SOAdapt: a process reference model for developing adaptable service-based applications

SOAdapt: *A Process reference Model for Developing Adaptable Service-Based Applications*

Stephen Lane

B. Eng., The University of Limerick, 2005
M. Tech., The University of Limerick, 2007

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

Supervised by Dr. Ita Richardson

Submitted to the UNIVERSITY *of* LIMERICK, October 2012

Declaration

Title: SOAdapt: *A Process reference Model for Developing Adaptable Service-Based Applications*.

Author: Stephen Lane.

Award: Doctor of Philosophy.

Supervisor: Dr. Ita Richardson.

I hereby declare that this thesis is entirely my own work, and does not contain material previously published by any other author, except where due reference or acknowledgement has been made. Furthermore, I declare that it has not previously been submitted for any other academic award.

26 October 2012

Abstract

SOAdapt: *A Process reference Model for Developing Adaptable Service-Based Applications*

Stephen Lane

Context: The loose coupling of services and Service-Based Applications (SBAs) have made them the ideal platform for context-based run-time adaptation. There has been a lot of research into implementation techniques for adapting SBAs, without much effort focused on the software process required to guide the adaptation.

Objective: This thesis aims to bridge that gap by providing an empirically grounded software process model that can be used by software practitioners who want to build adaptable SBAs. The process model will focus only on the adaptation specific issues.

Method: The process model presented in this thesis is based on data collected through industrial expert opinion and two case studies. The complete study involved 20 practitioners occupying various roles within 14 different companies. The data was analyzed using qualitative data analysis techniques. The output was used to develop a set of activities, tasks, stakeholders and artifacts that were used to construct the process model. The model was evaluated by means an academic expert opinion based survey where 10 academic experts provided their feedback on the model's process attributes.

Results: The outcome of the data analysis process was a process model identifying nine sets of adaptation process attributes. These can be used in conjunction with an organisation's existing development life-cycle or another reference life-cycle.

Conclusion: SOAdapt, the process model developed in this thesis, provides a solid reference for practitioners who are planning to develop adaptable SBAs. It has advantages over similar approaches in that it focuses on software process rather than the specific adaptation mechanism implementation techniques.

Supervisor: Dr. Ita Richardson

Keywords: Service-Oriented Architecture, Service-Based Applications, Software Process, Adaptation

Acknowledgements

I would like to acknowledge several people who have helped and supported me during this PhD.

Firstly, I would like to thank my supervisor Dr. Ita Richardson for the endless help and encouragement she provided during this project.

I would also like to thank all of the academic and support staff as well as my fellow research students at Lero - the Irish Software Engineering Research Centre. From reviewing papers to providing helpful suggestions and generally providing a pleasant working environment, they certainly make the journey a lot easier. My research was also partially supported by Lero under Science Foundation Ireland Grants No. 03/CE2/I303.1 and 10/CE/I1855.

I am also indebted to the members of the S-Cube consortium with whom I have had the pleasure of working with, and to the S-Cube consortium in general who funded my study through the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483.

I am very grateful to all of the industrial and academic participants who took part in this research and gave so freely of their time.

Finally, I want to thank my family for their support over the years, especially my Mom, Dad, Grace and Aaron :)

Contents

Glossary

CBD - Component Based Development
BCDF -
BPEL - Business Process Execution Language
BPM - Business Process Modeling
BPMN - Business Process Modeling Notation
CIM - Computation Independent Model
CDL - Choreography Description Language
CMM - Capability Maturity Model
CMMI - Capability Maturity Model Integrated
ECOWS - European Conference on Web Services
EDOC-ECA - Enterprise Collaboration Architecture
FM - Formal Methods
GPS - Global Positioning System
GSM - Goal-Service Modeling
HTTP - Hypertext Transfer Protocol
ICWS - International Conference on Web Services
ISO - International Organization for Standardization
IEEE - Institute of Electrical and Electronics Engineers
MAS - Multi-Agent Software
MDA - Model Driven Architecture
MDD - Model Driven Development
OMG - Object Management Group
OOD - Object-Oriented Design
PIM - Platform Independent Model
PSM - Platform Specific Model
QoS - Quality of Service
RE - Requirements Engineering
RUP - Rational Unified Process
REST - Representational State Transfer
RPC - Remote Procedure Call
SBA - Service-Based Application
SOA - Service-Oriented Architecture
SOAD - Service-Oriented Analysis and Design
SOAP - Simple Object Access Protocol
SOC - Service-Oriented Computing
SOMA - Service-Oriented Modeling and Architecture
SOSE - Service-Oriented Software Engineering
SPL - Software Product Lines
TSE - Traditional Software Engineering
UDDI - Universal Description, Discovery and Integration

UML - Unified Modeling Language
DCOM - Distributed Component Object Model
CORBA - Common Object Request Broker Architecture
XP - Extreme Programming
XML - Extensible Markup Language
WSBA - Web Service Based Applications
WSDL - Web Service Definition Language

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

Service-Based Applications (SBAs) are loosely coupled, adaptable applications that are based on software services and, as such, are service consumers rather than service providers. SBAs may be comprised of individual services or more complex service compositions. SBAs have been often described to as the successors of component-based applications which are also modular and adaptable [42]. The difference between component-based applications and SBAs is that component-based applications are composed of software components rather than services. Software components usually do not follow specific standards to describe their interfaces and are not as easily consumed over networks by heterogeneous applications. Services, on the other hand, specifically target networks and the Internet and most service implementation types are standards compliant [88].

The standardization of service interfaces allows for services to be consumed with minimal effort. Alternative services that provide the same functionality can easily be substituted for one another by modifying the service end point locations within SBAs. There may be semantic issues in relation to service descriptions, however, these can be overcome by semantically annotating services with standardized descriptions.

This flexibility inherent in SBAs allows them to be easily adapted when services become unavailable or their performance deteriorates. Furthermore, this flexibility allows the functionality of SBAs' to adapt in order to meet new business requirements. However, we need to have the appropriate tools, techniques and methods in order to develop adaptable SBAs. We also need to know how to use these tools, techniques and methods effectively in order to develop adaptable SBAs.

These tools, techniques and methods combine to form software development processes. Software development processes are an integral part of software engineering. Mature software development processes ensure that software development organisations work effectively and produce quality software [111]. There are a wide variety of process reference models that software organisations can implement to improve their processes. Additionally, there are many software process improvement models that can be used to assess and improve on existing development processes. However, it is necessary to continuously upgrade and improve existing software process reference models with practices to meet the requirements of newly emerging software development paradigms.

1.2 Problem Statement and Research Question

For many years the Service-Oriented Computing (SOC) paradigm has been maturing and is now beginning to see widespread adoption in industry. There are several mature technologies and standards, notably Web Services, that can facilitate the implementation of SBAs. While SOC technologies and standards have been maturing, it is unfortunate that software development process references have not kept pace. Existing software process reference models such as the Rational Unified Process (RUP) [97] and improvement models such as the Capability Maturity Model Integrated (CMMI) [106] do not meet the needs of this new paradigm.

These reference models do not provide guidance on how to deal with software services that are not in control of the application developers. More specifically, they do not provide any guidance on how to leverage the adaptive capabilities of SBAs. Without the necessary support processes it is difficult for application developers to identify all of the necessary tasks that need to be completed in order to develop and operate adaptable SBAs. The smooth operation of adaptable SBAs requires many tasks and activities that are not typically enacted with more traditional software development.

In order to bridge this gap it will be necessary to identify new methods and techniques to support the development and operation of adaptable SBAs. There are existing

approaches in the literature that address some aspects of the development challenges associated with developing adaptable SBAs. However, there are no holistic approaches proposed to support the complete development life-cycle of adaptable SBAs. More crucially the approaches that are proposed are often very low level or technology specific without the empirical basis of a large field study.

The first objective of this research is the development of a process reference model that can be used by practitioners who want to develop adaptable SBAs. Ideally, the model should be based on best practices identified in the literature as well as empirical data sources. The empirical data sources should leverage the knowledge and expertise obtained by practitioners who are actively developing adaptable SBAs. In order to limit the scope of the research and to avoid duplication, the model should focus on process attributes that are concerned with the adaptation of SBAs. Since the process reference model is focused on adaptation it would be necessary to use it in conjunction with an existing life-cycle model to facilitate the rest of the development life-cycle. The second objective of the research is to ensure that the model is valid. Demonstrating the external validity of the mode would allow it to be used generally by practitioners. Demonstrating the internal validity of the model would strengthen its credibility.

1.3 Methodology

The research methodology used in this thesis is based on Ahlemann and Gastl's [4] process model for constructing empirically grounded reference models. The adapted approach taken in this thesis consists of a literature review and problem definition, exploratory study, two empirical studies and an evaluation. The SOAdapt reference model was constructed and modified during the two empirical studies and the final evaluation process. Many of the research stages are self-contained with three of them published as research articles.

The first stage of the research, the literature review, was conducted in two parts. The first part was a traditional networked search while the second was a Systematic Literature

Review (SLR). The SLR was used to conduct an exhaustive search for process models for developing adaptable SBAs. The complete SLR is reported in [65] (Appendix B). A problem definition follows the literature review setting the stage for the rest of the research.

The exploratory study in Chapter 4.2 was conducted as collaboration between the author and two service experts. The complete study is reported in [64] (Appendix C). The author's contributions are reported in Chapter 4.2 while a summary of the co-authors contributions are also provided with appropriate references.

The first empirical study, resulting in SOAdapt-V1, was based on industrial expert opinion as well as an industrial case study. SOAdapt-V1 as well as its interim evaluation are reported in [63] (Appendix D).

The second empirical study, resulting in SOAdapt-V2, was also based on industrial expert opinion, this time with the focus of the study being on weaknesses identified in SOAdapt-V1.

The final evaluation process was conducted with input from academic experts, resulting in the completed SOAdapt. The completed SOAdapt model is published as a self-contained chapter in this thesis.

1.4 Summary of Contributions

The first contribution of this thesis is a state of the art literature review conducted in Chapter 2. This literature review contains a systematic map of the SBA development processes identified in the literature as well as a detailed review of the approaches that support adaptation. This review was used as a data source for the construction of SOAdapt-FoR. In addition this review, particularly the SLR, provides an excellent starting point for further research in related areas.

The primary outcome of this research is, SOAdapt, an empirically grounded reference model for guiding the development of adaptable SBAs. The model is based on empirical data gathered from a cross section of software practitioners from a variety of industrial

backgrounds. This data, along with best practices reported in the literature, provide the best possible guidance for the development of adaptable SBAs. Additionally, the model has been demonstrated to show both internal and external validity. This shows that the model is reliable and transferable to contexts independent of this research.

For industry, the main target of this research, SOAdapt provides practitioners with a process reference model that can be used to guide the development of adaptable SBAs. Like most other reference models, SOAdapt identifies the stakeholders and development artifacts that are related to its processes and activities. SOAdapt can easily be integrated with a company's existing software processes without overlap since it deals only with adaptation specific process attributes.

For academia, the SOAdapt process reference model can be used as an adaptable SBA development framework to which tools and other research contributions can be added. SOAdapt can be used by academics as a guide to the research area where they can identify further research opportunities. For example, many of the tasks and activities specified in the model are technically challenging and would benefit from lower level implementation guidance.

Finally, this thesis addresses the challenges to the development of adaptable SBAs that were identified in Section 2.7. Section 10.5 contains a description of how these challenges are addressed by the thesis.

1.5 Thesis Structure

The thesis is structured as follows:

Chapter 2: Literature Review The literature review is composed of three sections. The first section introduces SOC and the web services implementation of SOA. The next section deals with software process concepts and introduces some of the more popular software process reference models. The final section introduces software adaptation and describes paradigms such as component-based development that have been used to realise adaptable applications. The section moves on to describe processes for adapting SBAs

elicited through a networked search of the traditional software engineering literature, the service engineering literature and from a systematic review. The chapter concludes with a discussion of the development challenges for adaptable SBAs discovered in the literature.

Chapter 3: Research Methodology The research methodology looks at the philosophy of research in order to formulate the appropriate philosophical underpinnings for the research and to identify appropriate data collection and research methods for the thesis. The second part of the research methodology chapter outlines a research design which satisfies the objectives of the research.

Chapter 4: Phase 1: SOAdapt-FoR Construction This chapter documents a study carried out as a collaboration between the author and several service engineering experts. The purpose of this study was to explore the literature and to uncover and categorize any adaptation related activities. These activities were later combined with the results of the SLR for the construction of a frame of reference model used during the empirical studies.

Chapter 5: Phase 2: SOAdapt-V1 Construction This chapter describes the analysis of the data obtained during Phase 2 of this research. It explains how the data is used for the construction of SOAdapt-V1. This chapter also contains a literature based interim evaluation carried out to determine the external validity of SOAdapt-V1.

Chapter 6: Phase 3: SOAdapt-V2 Construction This chapter describes the data analysis of the data obtained during Phase 3 of the research. It also explains how the data is used for the construction of SOAdapt-V2.

Chapter 7: Phase 4: SOAdapt Evaluation This chapter describes the evaluation of SOAdapt-V2 through an expert opinion, interview based, survey. The objective of the survey was to evaluate each of the attributes of the model with respect to fitness for purpose and transferability. Attributes were modified where appropriate based on the feedback received from the survey.

Chapter 8: SOAdapt Final Model Description This chapter describes the completed SOAdapt model. It is written as a self contained chapter without reference to the rest of the thesis. This chapter may be used by those interested in implementing the final model. The chapter begins with a brief overview of the use of the model followed by a description

of the models processes and activities with each process area illustrated as BPMN process models.

Chapter 9: Instantiation of SOAdapt In this chapter, SOAdapt is instantiated during a sample development project. This instantiation shows how the model can be used to guide the development of an adaptable SBA which is based on a real development scenario.

Chapter 10: Conclusion The concluding chapter presents a summary of the work that was completed in the thesis. Particular attention is paid to whether the challenges set out in the the literature review are met by SOAdapt. Finally, the directions for future work in the area are explored.

Chapter 2

Literature Review

2.1 Introduction

This state of the art review is intended to introduce the reader to the research areas relevant to this study and to critically analyse existing research publications within those areas. This is a cross cutting research project which encompasses the Service-Oriented Computing (SOC), software process and software adaptation research areas. The first section (Section 2.2) of the review will introduce SOC and related concepts such as Service-Based Applications (SBAs) which are the focus of this research project. Section 2.3 introduces the Software Process research area, leading on to Section 2.4 which deals with software process in the area of SOC. Finally Section 2.5 introduces the subject of software adaptation and how it relates to this research.

2.2 Service-Oriented Computing

Service-Oriented Computing (SOC) is a computing paradigm which uses software services as fundamental elements for developing software applications [89]. The paradigm encompasses the development of SBAs introduced in Section 1.1. Services which form the basis of SOC, are self contained computation elements which are made available through published interfaces. These published interfaces often contain service descriptions that allow them to be more easily consumed. In SOC, there are at least two stakeholders, service providers and service consumers. The service providers make services available for multiple service consumers to consume. There can also be a third stakeholder, called a service broker. The service broker may publish the services on behalf of

the service provider and negotiate on their behalf with service consumers. At their most basic level, services can be described, published, bound to and discovered. However, at a higher level, services can be combined into applications or service compositions which are monitored, managed and governed by business stakeholders. There are many service enabling technologies available such as Representational State Transfer (REST), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA) and Web Services. In recent times Web Services have become the de facto service enabling technology both in industry as well as academia. Conferences such as the International Conference on Web Services (ICWS) and the European Conference on Web Services (ECOWS) have been setup recently to cater for the demand for web services research.

Within the research areas of Service-Oriented Computing (SOC) the terminology used is often ambiguous and non-standard. Here a taxonomy is presented where a brief description of the most common terms used throughout the section. The SOC terms used in this review are defined the S-Cube project's knowledge model [2].

Service-Oriented Computing (SOC) refers to the computing paradigm that has evolved from object-oriented and component based design. The SOC paradigm facilitates ad-hoc, platform independent communication between unintroduced parties through the use of software services [2].

Services play a central role in SOC, they are self-contained loosely coupled computational elements that expose the functionality of underlying software. Services can be combined or composed into more complex services or they can be used in combination with other services to form Service-Based Applications. Services can be published on public registries and made available for use by third party clients [2].

Service Based Applications (SBAs) are applications that are composed of software services. They may consume services from third party services providers or service providers within the same organisation. A common example given for a SBA is the travel booking application: this is composed of a hotel booking service, a car

rental service and an airline reservation service each of which are provided by third parties. Synonyms in the literature include service-oriented systems or service-based systems [2].

2.2.1 Service-Oriented Architecture

Service-Oriented Architecture (SOA) is another important concept in SOC, a SOA is a multi-layered architecture made up of services which can be used across multiple domains [88]. Figure 2.1 [1] illustrates the various layers that make up a SOA. The first layer (6) of a SOA architecture contains the applications and databases that underly the SOA. The next layer (5) contains component-based service realizations that access the data and logic from layer 6. Layer 4 contains simple infrastructure services that expose functionality from the underlying component-based service realizations. The next layer (3) is the business services layer. Here, infrastructure services are combined into business functions such as record management (add order, delete order). In the next layer (2), business functions are orchestrated into meaningful business processes that can be accessed by business domains (1).

2.2.2 Service-Based Applications

At the heart of SOC and SOA are Service-Based Applications (SBAs) [5], these applications are above the "Business Domain" layer in Figure 2.1 and can be composed of simple or complex services. Simple services are services which simply expose a particular piece of functionality, while complex services are compositions of several simple services, which place them higher up in the SOA hierarchy. Simple services, while lower down in the SOA hierarchy, may still be called by SBAs. Calling simple services directly is considered suboptimal design. An often cited example of a SBA is the "travel reservation application" which is composed of services from several different service providers. The services are an airline ticket reservation service provided by Company X, a car rental

[1] Reproduced with permission from [88]

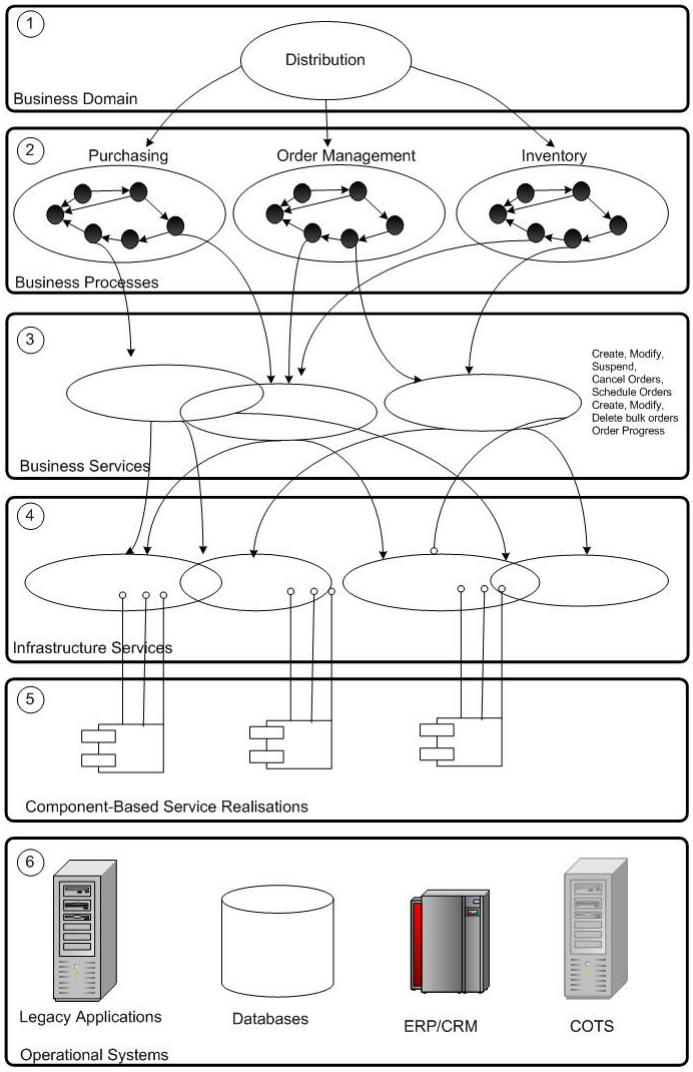

Figure 2.1: *SOA Layers [88]*

service provided by Company Y and a hotel reservation service provided by Company Z. With this example we can see that a useful SBA can be composed quickly be making use of third party services which complement one another.

Due to the nature of SBAs, they can easily be adapted to suit business or contextual requirements and adaptation can occur easily by replacing a service with a functionally identical service from an alternative provider. One scenario where this might be desirable is when different providers provide the same type of service but the cheapest provider varies with time. In this case the application's business stakeholders will always want to consume the cheapest service. This ability to adapt to market demands gives great power but also poses new challenges for SBA developers. If adaptation is a requirement

from the outset, this will have to be taken into consideration during the SBA development life-cycle.

2.2.3 Web Services

Web Services are becoming the de facto SOA enabling technology and are used as a basis for much of the current software service research, particularly research on topics such as service discovery, service negotiation and adaptation of SBAs [88]. For this reason Web Services, over other service implementation technologies, will be discussed in detail in this section.

Web Services are heterogeneous software components that can run in different operating environments to exchange messages and integrate with one another without requiring third party proprietary software or middleware. This is achieved by using standards based technologies over existing network protocols. Extensible Markup Language (XML) and Hypertext Transfer Protocol (HTTP) are the lowest common denominator technologies required for a system to implement Web Services. This permits complex applications to be constructed on the fly by anyone from anywhere by combining different services from multiple sources [88]. There are many specifications associated with Web Services, many of which are provided by different standards bodies such as Oasis[2] or the International Organization for Standardization [3]. Web Service specifications are provided for a wide range of functions from message handling to metadata exchange or security.

Web Services are self-describing, self-contained software modules which can be accessed directly, or can be accessed by means of a Web Service directory. Web Service directories serve several purposes such as showing potential Web Service clients that a particular type of Web Service is available for consumption. Another useful purpose of a Web Service directory is to provide alternatives of the same Web Service to potential clients. So, for instance, if a client uses a Web Service to process an electronic transaction, then the client can choose between the best payment Web Service available from

[2] http://www.oasis-open.org
[3] http://www.iso.org

the Web Service directory. The client can use criteria such as the processing fee to determine which Web Service to use. It is possible to obtain non-functional Web Service properties such as cost, availability or service level agreements from certain Web Service directory types. Many companies implement internal service directories with a catalog of their own internal services. There are currently no standards based service directories available publicly on the Internet. There are however several human readable web based service directories such as WebserviceX[4] that provide useful categorised lists of publicly available services.

Web Services vary greatly in terms of function and complexity. The most basic type of Web Services are usually Remote Procedure Call (RPC) type transactions, where the client sends a message to the Web Service publisher and then waits for a response. More complex Web Service, on the other hand, may provide access to software functionality rather than data or content. Web Services may be used to enable legacy or custom built software components to expose their functionality by means of a component wrapper. Web Services can be synchronous like RPCs or they can be asynchronous and do not provide a response.

Communication between Web Service clients and publishers tend to be coarse grained messages as opposed to the fine grained object level calls which are used in distributed computing. In order to make Web Services more consumable, they tend to wrap useful business functions in each Web Service realisation. If Web Services provided fine grained object level functionality, it would take a lot of effort to combine them into useful applications. Fine grained Web Services would also require a large number of service calls which would significantly increase network overhead as the messages are XML based. A key objective of Web Services is to provide useful business functions while abstracting the complex implementation of their underlying infrastructure.

In order for Web Services to communicate between heterogeneous systems, it is necessary to use a messaging protocol that is independent of an operating system or a programming environment. It is also desirable to have a messaging protocol that can be

[4]http://www.webservicex.net

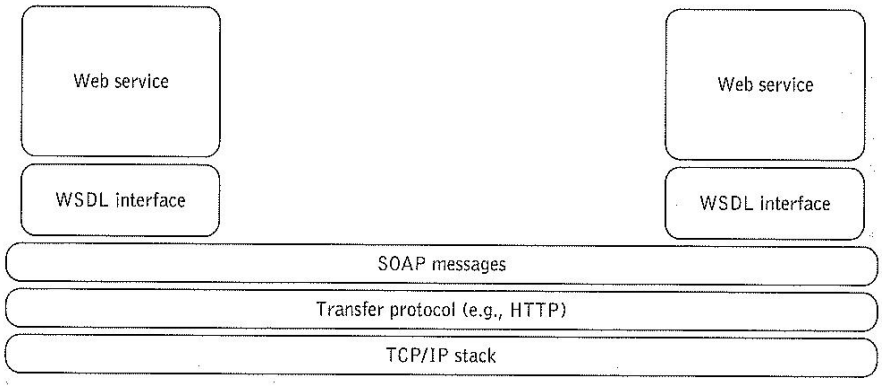

Figure 2.2: *The Web Services Communication and Messaging Framework [88]*

used on any network with the ability to seamlessly traverse firewalls and heterogeneous network hardware. The messaging protocol that has been adopted for Web Services is SOAP. SOAP is an XML based messaging protocol that usually uses HTTP as its transport protocol. Figure 2.2 demonstrates where SOAP messages fit in to the Web Services stack.

SOAP content requests or RPCs are often synchronous, while more complex intra-business exchanges tend to be asynchronous. Complex exchanges tend to be asynchronous because it is usually not practical to wait for a response from another business before continuing with an application's logic.

2.3 Software Process

The process used to design and build a software application is called the software process [111]. This process may include the organisational and technological aspects of a software development project. At its most basic form, the software process is the set of development tasks or activities used to develop a software application. The software process may consist of ad-hoc work activities or a formally documented process model. A software process model is the representation of a software process [27].

A software process model may be descriptive or prescriptive. A descriptive software process model describes an existing software process. A prescriptive software process

model proposes an optimised method of developing software. Organisations often create descriptive models of their software process so it can be analysed and improved. It is also common for software engineering researchers to use existing software process models as well as field studies to create prescriptive process models [83].

Watts and Kellner [111] define several criteria for an effective process model. They state that process models should be effective in communicating the software process between stakeholders involved in the development process. Process models should also be reusable given the cost and time associated with their development. Additionally, since real world processes evolve to meet business requirements, software process models should also be capable of evolving easily. Finally, process models should facilitate the management of the software process by allowing management to compare the activities of a project to those set out in the model.

2.3.1 Software Process Reference Models

While a process model may describe a particular process, the process that it describes may not be an exemplar process so the model is of no use as a reference model. A process reference model on the other hand may or may not describe an existing process but is intended to be an exemplar reference model. Therefore, a process reference model is a specific type of process model that is used as a reference. Winter and Schelp [113] describe process models as a specific type of reference model. Reference models also encompass models of entities such as data structures and functions. In other words, they state that reference models are generic conceptual models that can be used for guidance in specific domains.

In the software engineering literature, the term Process Reference Model has become a synonym for Process Model with both being widely used.

Henceforth, the term process reference model will be used in this thesis. In order to get a view of the state of the art, some of the more prominent software process reference models from the literature will be reviewed here.

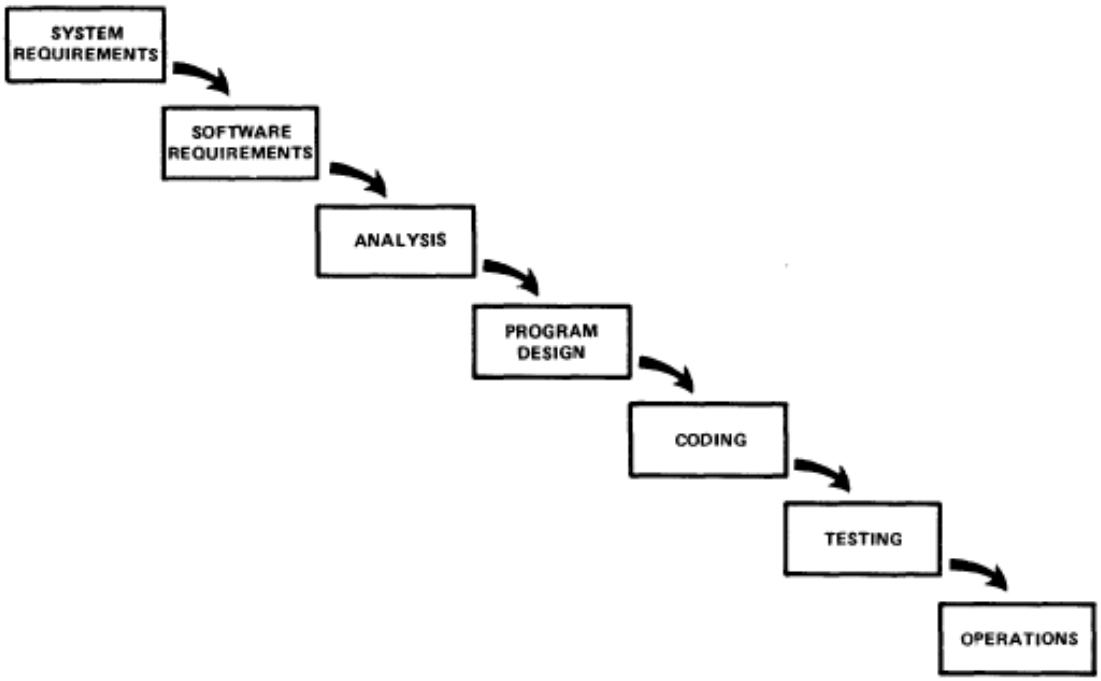

Figure 2.3: *Waterfall Model [100]*

2.3.1.1 Waterfall Model

The waterfall model [100] is one of the most recognisable software process reference models in the literature. It is a linear development process with each process followed by the next process once complete. When originally published in 1970 [100] the model contained the processes: System Requirements, Software Requirements, Analysis, Program Design, Coding, Testing and Operations. The model gets its name from the cascading shape of its graphical representation illustrated in Figure 2.3.

In an attempt to address the simplistic nature of the simple waterfall model, Royce, in his seminal paper, later introduces a modified version of the waterfall model with feedback loops between each process. This modified iterative waterfall model was designed to allow communication between each process providing greater flexibility for project stakeholders. The improved iterative waterfall model is illustrated in Figure 2.4.

It is difficult to find modern models that have not inherited some features from the waterfall model even though it does have critics. Watts [111] criticizes the simplicity of the model noting that in real life, software projects are never so straight forward. Often

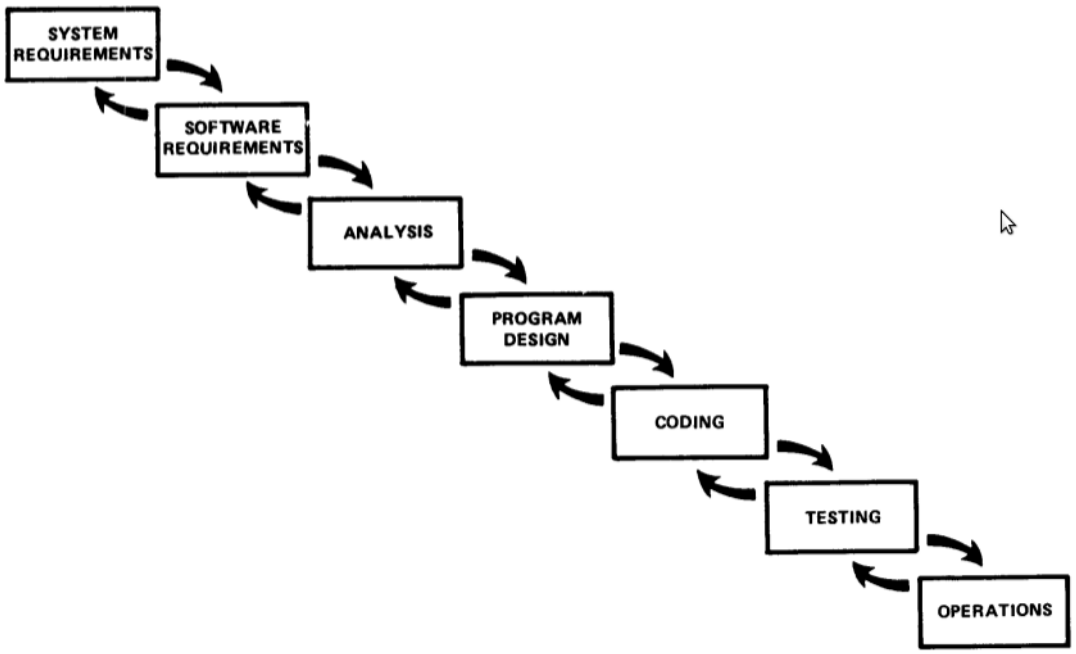

Figure 2.4: *Iterative Waterfall Model [100]*

the processes will run concurrently rather than sequentially and there is no way for the waterfall model to deal with requirements change during the project.

2.3.1.2 Spiral Model

The spiral model of software development is an evolutionary development life-cycle. It was proposed to be an incremental improvement on the waterfall model mentioned in the previous section [16]. Figure 2.5 shows how the spiral life-cycle reflects the cumulative cost of each development iteration. It aims to be a risk driven model where risk analysis is conducted on an on going basis. Taken more simply, the spiral model is a series of waterfall life cycle iterations, with subsequent iterations becoming more sophisticated.

The spiral model aimed to bridge some of the short comings of the life-cycle model that came before. Specifically, it addresses the reality that most software development projects aren't composed of self-contained discrete processes that occur one after another. Boehm recognised that software development is often an iterative process with several iterations of development necessary before a production application is produced. Boehm built a risk minimization strategy into the life-cycle with earlier versions of the application

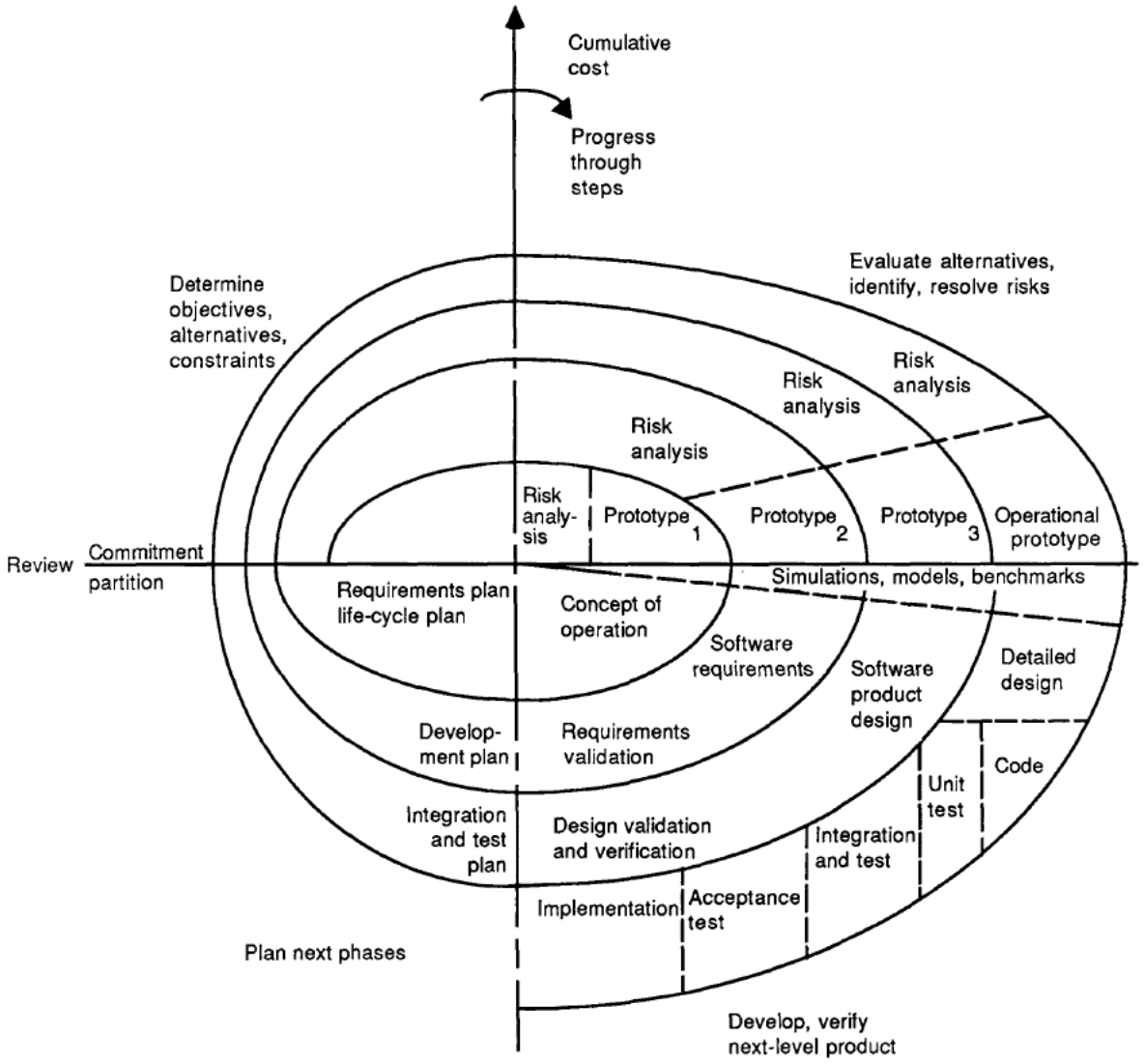

Figure 2.5: *Spiral Model [16]*

involving less investment. As the product matures more expensive processes such as detailed design and testing are introduced. These are introduced in later iterations because if the project fails at an early stage there will have been less investment lost. As the application matures and the risk of failure decreases the more costly processes necessary for producing a quality application are introduced.

2.3.1.3 Agile Software Development

Agile software development describes a set of software development methodologies that are communication and skills oriented, and can easily be adapted to different project re-

quirements [22]. Two prominent agile methodologies are eXtreme Programming (XP) and the scrum methodology.

Firstly, XP, as the name suggests is programming oriented. XP development is conducted by small groups of programmers who work together and usually involve at least one customer in the group [10]. Programmers will often work in pairs where two programmers share a workstation. This allows one programmer to formulate ideas while the other writes code. They will periodically switch roles in order to maintain focus. This type of collaboration is often referred to as paired programming. XP is an iterative development process with short development cycles of about 3 weeks. Release periods, where code is deployed to customers, occur every 2 to 5 development iterations. Each iteration consists of a requirements gathering activity that involves the on site customer. The customer describes the requirements as "user stories" and negotiates with the developers about which requirements get implemented during each iteration. Requirements are broken down into development tasks by the developers and assigned approximate development times. During each iteration, development is conducted using paired programming, with the developers constantly refactoring and simplifying the application code. All code is subject to 100% unit testing before it is accepted for the project.

The SCRUM methodology, named after the rugby set piece position, is a flexible development methodology where stakeholders form "a tight formation of forwards who bind together in specific positions when a scrumdown is called" [101]. The SCRUM methodology understands that there are certain aspects of the development process that are unpredictable so it treats them as a black box rather than defining them in advance. Then these black box processes are empirically defined during development. When they are defined, appropriate stakeholders are assigned to "scrum" in order to complete the process. This methodology allows flexibility within the development process which allows it to adapt with change. On the other hand, the unpredictability of the development process introduces risk which needs to be managed in order to avoid chaos.

Figure 2.6 illustrates the development process within the scrum methodology. The scrum methodology inherits features from linear models such as the waterfall model as

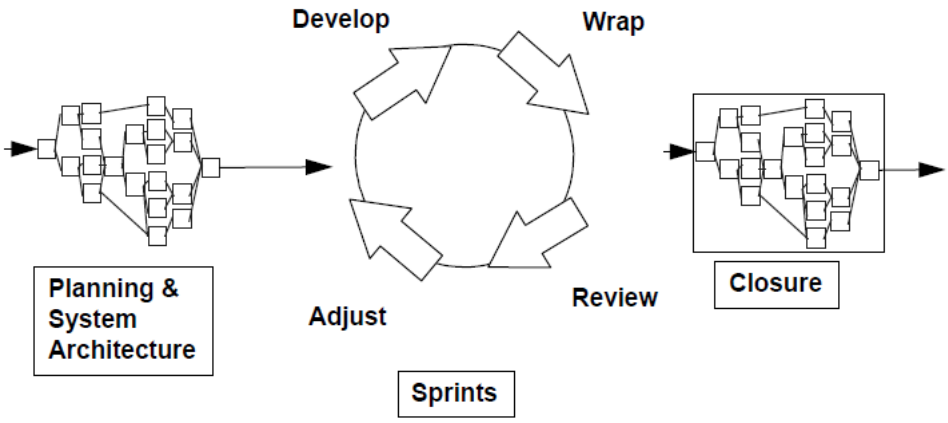

Figure 2.6: *The Scrum Development Methodology [101]*

well as iterative models such as the spiral model. Planning and system architecture occur linearly with the development sprints and the closure of the project. The development sprints that occur in the middle of the process are iterative development cycles that are empirically defined based on project as well as environmental variables.

2.4 Software Process in Service-Oriented Computing

There are many software development process models, such as those in Section 2.3.1, that can be successfully used for the development of complex software systems. However, each of these software development models were conceived with Traditional Software Engineering (TSE) in mind so they are not directly applicable to SOC. There are many reasons for this. For example, during TSE, each system component is usually developed by a single organization as part of a self-contained software application. However, during Service-Oriented System Engineering (SOSE), services are usually provided by one or more third parties and combined to form a Service Based Application (SBA). Similarly services are designed to be loosely coupled and combined in unpredictable ways. SBAs are often designed to be adaptable, which allows services to be exchanged for similar suitable services possibly from a different service provider. Traditional software systems are usually not designed to be adaptable and adaptation involves a costly maintenance process. For these reasons it is obvious that existing development models are not directly

applicable to this new paradigm. In order to address this issue it is necessary to either modify existing approaches or formulate new service specific models.

There have been several service-oriented development models proposed by proponents of SOA, for example IBM have proposed the SOMA [7] methodology while a group of industry practitioners have proposed the SOA practitioners guide [32]. Similarly, there have been efforts to tailor the RUP framework so that it can be used for service-oriented development. Many of these approaches are reported in grey literature such as white papers or technical reports. There may also be many other proprietary approaches maintained by organisations that are kept confidential for competitive advantage. In addition, there are many approaches published in scientific publications such as conference or workshop proceedings as well as peer reviewed journals.

Given the existing process reference models for developing SBAs, the reader my be wondering about the necessity for another. The reason is that none of the existing mature or empirically grounded models facilitate adaptation. Furthermore, the models that do support some aspects of adaptation, are either not based on empirical evidence or they are specific to particular areas of the development life-cycle.

2.4.1 Software Processes from Systematic Literature Review

During this research, a systematic mapping study was conducted in order to get an overview of the processes proposed for developing SBAs. This mapping study was part of a larger systematic literature review, following the procedure set out in [54]. The mapping study focused on identifying software development processes that focus on SOC. The second part of the review focused on identifying the subset of those approaches that in any way facilitate the development of adaptable SBAs.

Within the systematic mapping study, a list of SBA development processes were distilled from the relevant studies that were identified. These SBA development processes are presented in the current section. The complete review is reported in [65] which is contained in Appendix B.

Here, the process categories identified during the review will be discussed in detail.

2.4.1.1 End To End Models

Many of the studies encountered in the SLR propose end to end SBA development methodologies with processes for design, implementation and operation. These methodologies often borrow concepts from existing software development techniques or methodologies, for example, Kruger and Meisinger [112] propose an extension of the V-Model for use with SBAs. Similarly Christou et al. [21] propose the use of Agile methods along with the Rational Unified Process (RUP) development methodology for the development of SBAs. The use of Software Product Lines (SPL) development techniques are also suggested in combination with Agile methods by Karama et al. [105]. Deubler et al. [30] have developed a tool to support the development of SBAs, the tool's primary functions are to formally verify SBAs and to generate code.

Many of the end to end process models in the literature follow the classical waterfall development model with some modifications that are particular to SOC. Adamopoulos [118] proposes an iterative development methodology which includes many of the waterfall model's processes, however they are adapted to suit Web Service Based Applications (WSBAs). Papazoglou and van den Heuvel [90] suggest a process model that also bears a close resemblance to the waterfall model. However, it has processes such as "Execution and Monitoring" that are specific to SBAs. Processes in this model such as "Analysis and Design" bear resemblance to traditional processes not intended for SBA development. However, they are tailored specifically to meet the requirements of such applications.

Along with the fully defined process models, there are many "work in progress" models in the literature. Many of the "work in progress" models attempt to discover all of the service related concepts and aspects that make development of SBAs unique from the development of traditional software applications. One such work, by Engels and Assmann [35], has set out several development challenges of developing service-oriented enterprise architectures. Ivanyukovich et al. [14] are also working towards a service-oriented process model. They propose a model with the following three dimentions: managing the

software change, specifying the development process and targeting stakeholders goals.

The final type of end to end process model encountered in this review are interdisciplinary development models. These development models involve expertise from several domains. For example, Lamparter and Sure [76], propose a development methodology that includes aspects of web service engineering, market engineering and ontology engineering.

2.4.1.2 Analysis and Design

There are many analysis and design process models reported for the development of services and SBAs. Most approaches fall in to the categories of top down, meet in the middle or ground up service realisation [90]. The top down design approach involves working backwards from existing service interfaces. The components required to implement these services are constructed based on the functionality described by the service interfaces. This design technique is considered best practice as business requirements shape the service interfaces which in turn drive the development process. The bottom up approach differs in that the functionality of existing applications are made available by wrapping existing system functions with services. Finally, the meet in the middle approach combines features of both top down and bottom approaches. This technique is useful when a service needs to be created to implement an existing service interface. However, rather than create a new software component to implement this service, which would be done with top down design, an existing software component that has all of the required functionality would be wrapped with the required service.

The top down approach is more business oriented than its counterparts because services are created to meet specific business requirements. The same can be said for the meet in the middle approach when software components exist to suit service requirements. The ground up service realisation approach is the least business oriented in that existing software components are wrapped with services which limits the capability of resultant applications.

After reviewing the development models that address the analysis and design pro-

cesses, it appears that many of these models inherit characteristics from the popular Service-Oriented Modelling and Architecture (SOMA) methodology developed by Arsanjani et al. [7, 6]. The SOMA methodology is one of the most mature design process models encountered in the review. Its developers claim that the methodology was developed during involvement with hundreds of SOA projects. SOMA's design activities are divided into three processes: "Service Identification", "Service Specification" and "Service Realisation". These processes appear in many other models in some form or another. For example, Engels et al. [23] propose a method for engineering SOAs which contain the design processes of: "Identify top-level business services", "Refine business services" and "Specify business services". The SOMA processes and the other analysis and design processes encountered in the review will be discussed in more detail in the following paragraphs.

Service Identification is a process that identifies the candidate services required for developing a SBA. It involves the identification of the required services as well as the software that will realise those services. SOMA identified three techniques for the identification of services: goal-service modeling (GSM), domain decomposition, and existing asset analysis. Kenzi et al. [25] propose a Model Driven Architecture (MDA) framework for the development of adaptable SBAs that uses service identification models for the identification of candidate services.

Service Specification involves the development of service interfaces and the management of service dependencies, once the required services have been identified. Most service development methodologies either implicitly or explicitly contain a service specification process. Specifying services is one of the fundamental building blocks of developing SBAs.

Service Realisation, like service specification, is a vital step in the development of SBAs. Realisation is the final process before the implementation of services. It involves the implementation of service specifications into usable design documents that can be used by service implementers. An extremely common approach being proposed for the realisation of services is the application of Model Driven Architecture (MDA). This is a

model driven design approach with the added benefit of automatic code generation from design models. MDA will be discussed in more detail in Section 2.4.1.3.

Another description given to the analysis and design processes for SBAs is Service-Oriented Analysis and Design (SOAD). This is an extension of the Object-Oriented Development (OOD) and Component-Based Development (CBD) paradigms. SOAD is often used as a general term in the literature to describe analysis design approaches for service-oriented systems. Different SOAD methodologies take different approaches to service design. For example, Chang and Kim [82] propose a 5 phase design process within the following phases: "Identifying Business Processes", "Defining Unit Services", "Discovering Services", "Developing Services", and "Composing Services". Comparatively Kambhampaty [50] proposes a SOAD methodology with 4 processes which are quite different from Chang and Kim's methodology: "Activity Services Development", "Business Process Services Development", "Client Services Development" and "Data Services Development".

Service Discovery for static SBAs occurs during application design, while the discovery of services for dynamically adapting SBAs may occur during system runtime. There are many ways that service discovery can be achieved, from manually searching service directories to automatically scanning directories for candidate services that match a given criteria. Howard and Kerschberg [44] present a framework to support the preparation, publication, requisition, discovery, selection, configuration, deployment, and delivery of semantic web services. Their framework specifies that both published services and service clients contain semantic descriptions of their capabilities and various types of requirements.

Service Composition is a service design/construction process that involves the combination of services into new complex services or SBAs. Service composition spans several processes of the software development life-cycle - how services are composed needs to be established during the design of SBAs, the composition gets enacted during the development of a SBA and finally service compositions need to be monitored and possibly adapted during the operation of SBAs. Ren et al. [20] propose a framework that facilitates

the visual design, validation and development of web service compositions.

Agent-Orientation describes a software architecture that is made up of a system of intelligent agents. The agents have characteristics such as autonomy, social ability, reactivity, and pro-activeness [114]. Lu and Chhabra [71] propose a service-oriented design methodology that involved the wrapping of intelligent agents with web services which speeds up the development process and results in more competent web services.

2.4.1.3 Construction and Testing

Many of the process models encountered in the literature focus on the construction of services [40], SBAs [62] or both [53]. The construction of services is a role undertaken by service providers whereas the construction of SBAs is often undertaken by service consumers. Service providers and service consumers have differing roles but may be in the same organisation. The focus of this review is the construction of SBAs rather than services alone, however in the absence of suitable existing services, the construction of services is a sub-process of the SBA construction process.

The SBA development processes proposed vary in their approach. A common proposal is to adapt an existing traditional engineering approach or framework to suit the service-oriented paradigm. Meisinger and Kruger [112] suggest an adapted version of the traditional V-Model for the development of service-oriented systems. Similarly there have been many Model Driven Development (MDD) techniques suggested for the construction of SBAs. MDD is a popular approach for the design and implementation of object-oriented software. During MDD, analysis and design models can be used for code generation. MDD can greatly simplify the construction process of SBAs by abstracting complex implementation standards and service interface details from developers. This process will be discussed in more detail in the following section.

Model Driven Development (MDD) is a development paradigm that uses models to represent and reason about problem and solution domains [53]. Modeling is a useful method of working towards a solution which enables the representation of dependencies between the components being modeled. Once a model has been created it is possible

to view the effects of changing part of the model by examining dependencies within the models. A model can be used as a reference in order to construct a software system, alternatively using the appropriate technology a well defined model can be used to automatically generate a platform specific model and then application code which can form the skeleton of a system. Model-Driven Architecture (MDA) is a specific implementation of MDD maintained by the Object Management Group (OMG)[5]. MDA includes the development of Platform Independent Models (PIMs) using Domain Specific Languages (DSLs). The OMG suggest the use of the Service Oriented Architecture Modeling Language (SoaML) for defining service-oriented PIMs. PIMs are then transformed into Platform Specific Models (PSMs) that form the basis of executable code [17, 87].

This use of modeling to generate system code is referred to as Model-Driven Architecture (MDA).

Models are an intuitive method of developing solutions while at the same time they abstract the specific details of implementing the solution. This proposition is particularly appealing to SOC where there are many protocols and standards involved in implementing SBAs. In order to implement a basic SBA, the solution may involve the SOAP protocol for messaging, WSDL for describing service interfaces, and BPEL for implementing service compositions. Each of these are verbose XML based languages which are difficult to interpret by humans and are error prone when constructed manually. For this reason there have been many service development processes proposed that are based on MDD. In this systematic review 17 different MDD approaches were encountered.

A common approach proposed is to develop system models with UML [70] diagrams. This is a logical choice as there are many MDA tools in existence that can handle UML. Johnson and Brown [53] outline an MDD technique for developing SBAs using a UML profile suitable for service modelling. Many of the the MDD approaches proposed conform to MDA principles.

The following list is presented to briefly summarise some of the MDD approaches suggested in the literature:

[5]http://www.omg.org

Collaborative modeling is a collaborative MDD approach where stakeholders involved in system development may collaborate remotely using XML nets [52].

Semantic modeling is a MDD approach that uses semantic technologies to automatically translate between the different abstraction levels in MDA [87].

SOMA Bercovici et al. [13] propose a MDD approach that complies with IBMs SOMA methodology. It complements all of the SOMA development processes with the added benefits of MDD.

Context aware Samyr and Slimane [110] propose a MDD development approach that can be used to develop context aware SBAs. Context-aware applications have the added benefit of adapting to variable contextual parameters.

Web based Tavor et al. [40] propose a web based model editor which is part of a model driven service engineering process. It has advantages such as ease of accessibility, zero client foot print and simple consumption.

MDD automates many of the processes involved in software development such as analysis, design or implementation. However, the use of MDD does not negate the advantages of a software process model for the other processes in the development life-cycle. In conjunction with MDD, it is often necessary to implement processes such as requirements engineering, operation and management, and adaptation, all of which would benefit from the adherence to a suitable process model.

2.4.1.4 Deployment and Provisioning

Papazoglou and van den Heuvel [90] describe service provisioning as a process that contains practices such as service metering, service rating and service billing. In other words, it is an activity that allows service providers to monitor service usage and charge where appropriate. Provisioning may not be a concern when service producers and consumers are within a single organisation.

Service deployment on the other hand involves deploying executable service code and artifacts to application servers for consumption by service consumers which may include SBAs. Service publishing may also be undertaken which involves the publishing of service descriptions to a service registry or directory. Nineteen out of the 57 primary studies contained details about the activities required for deployment or provisioning.

2.4.1.5 Execution and Monitoring

The execution and monitoring processes are concerned with the runtime activities of services and SBAs [90]. Services need to be monitored during runtime in order to ensure that they are available and functioning correctly. Similarly SBAs need to monitor each of their component services to ensure that they are available and functioning correctly. Monitoring can be an automatic or manual process depending on the level of sophistication of the service or application. If the monitoring process for a SBA detects that a service becomes unavailable or its characteristics have changed to a state that are no longer suitable, then it may be desirable to adapt the application. Adaptation in the context of SOC, is a process which either automatically or manually facilitates the re-configuration of services within SBAs. Only four of the primary studies explicitly address execution and monitoring. However, nine primary studies address adaptation which is implicit in execution and monitoring. Curiously, only one of the papers that explicitly supports Execution and Monitoring also included processes supporting adaptation [72].

The *Security* process is usually regarded as an ongoing concern for services and SBAs. As with traditional software systems, security has to be considered at each stage of the development life-cycle. Fernandez et al. [107] propose the extension of a secure development methodology to the development of SBAs. Delessy and Fernandez [3] propose a pattern-driven security process for SBAs.

2.4.1.6 Formal Methods

Formal Methods (FM) is a software engineering discipline that centres around mathematical techniques that can be applied to processes such as formal specification, development

and verification of software systems [1]. These techniques are expensive to implement so they are usually reserved for safety or security critical applications. Due to the complexity of implementing these techniques they are applied sparingly. However, there have been many applications of FM proposed to aid in the analysis, design and construction of SBAs. FM techniques such as formal specification can be beneficial for the construction of SBAs as they will ensure that the complex applications are constructed correctly. The SENSORIA project [72], amongst others, proposes a method for Augmenting Service Engineering with various types of formal methods for the analysis, transformation and dynamicity of service-oriented systems. Formal methods are often used in conjunction with other software engineering techniques for the development of SBAs. For example, Nguyen et al. [80] propose an SBA engineering methodology developed using techniques from formal methods as well as MDA. One of the problems with using Formal Methods during the development of SBAs or using Formal Methods in general is the steep learning curve involved.

2.5 Software Adaptation

Software adaptation is the modification of a software application during runtime in order to best meet the requirements of its operating context. Software adaptation has been around for a very long time with the earliest software applications facilitating adaptation through self-modifying code [75]. However, in more recent times paradigms such as autonomous, ubiquitous and service-oriented computing have sparked new interest in adaptation. Software adaptation may be fully automatic with closed feedback loops monitoring the application for adaptation triggers [78], or it may be more manual with stakeholders manually acting monitored application states.

McKinley et al. [75] describe 2 types of adaptation, adaptation that is based on *static* or *dynamic* composition. Adaptation that is based on static composition involves the adaptive elements of the application being hard coded into the application at runtime. This method is somewhat restrictive as it does not allow additional application components

to be added at runtime. However, with static composition, there may be some runtime flexibility through configuration mechanisms. Configurable options that are stored in a configuration file or a database may be modified to facilitate a limited type of runtime adaptation. Applications that are based on dynamic composition on the other hand can have their functionality substantially modified during runtime. For example, new software services or components may be added at runtime or existing ones modified. This is the most flexible type of adaptation but it is also the most difficult to implement. From this point forward adapting through static composition will be referred to as static adaptation, while adapting through dynamic adaptation will be referred to as dynamic adaptation.

2.5.1 Component-based adaptation

Component-based software engineering is a software engineering paradigm used to develop software applications from reusable software components. It has often been referred to as the precursor to service-oriented computing making the adaptation of component-based applications relevant to the adaptation of SBAs [42].

Oreizy et al. [84] have proposed an approach for component-based adaptation (see Figure 2.7). Their approach contains two cycles that run in parallel, an evolution management cycle and an adaptation management cycle. The evolution management cycle manages modifications to the application through the use of tools and adaptation mechanisms. Modifications permitted include the addition, removal or replacement of application components and their connectors. The evolution management cycle examines changes to the application and prevents changes that might make the operation of the application unsafe or inconsistent. The adaptation management cycle on the other hand collects observations while the application is running. These observations are then analysed in order to determine if it is necessary to adapt the application. Should an adaptation be deemed necessary, then the adaptation cycle provides the necessary tools for deploying alternative components.

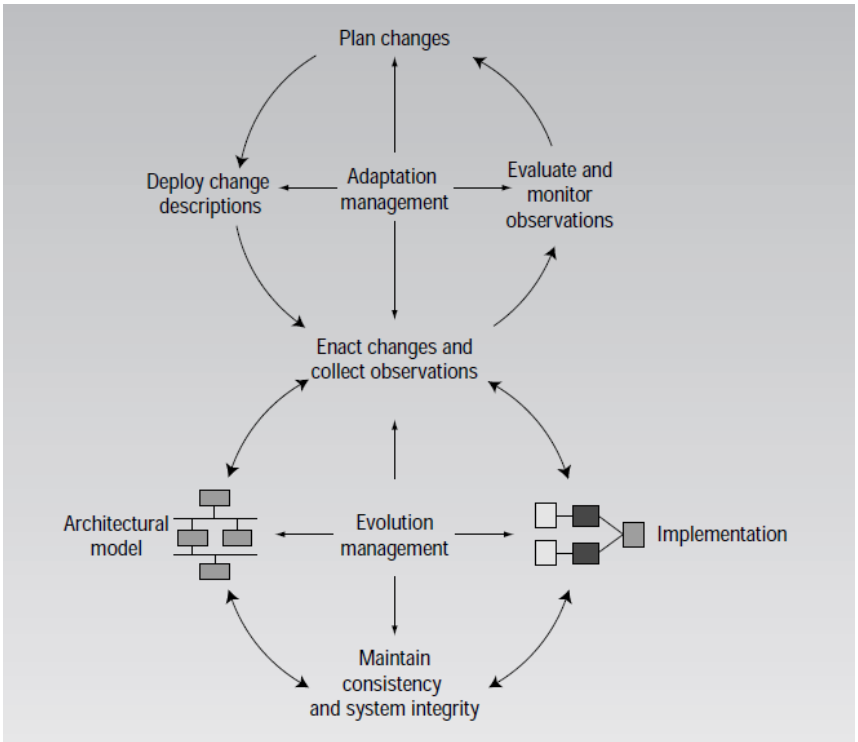

Figure 2.7: *Oriezy et al's Approach for Component-Based Adaptation [84]*

2.6 Adaptation of SBAs

Within the context of SBAs, adaptation is the modification of an application in order to satisfy adaptation requirements [12]. There are many adaptation requirements that can be desirable in SBAs, for example, the facilitation of interoperability amongst services [61], the optimisation of Quality of Service (QoS) [31] or the implementation of failure recovery [93]. SBA adaptation may involve the substitution, replacement, re-configuration or removal of component services from a SBA. Once adaptation requirements have been determined, it is then necessary to create an adaptation strategy. After the adaptation strategy has been developed, it is then possible to enact the adaptation.

This is in contrast to the evolution of SBAs which refers to the initial requirements, design, implementation and operation of SBAs. In order to appropriately determine whether or not adaptation is required, it is useful to monitor the execution of SBAs. Monitoring can be done automatically by an application or can be achieved manually by reviewing error logs. There have been many monitoring frameworks proposed. Pistore et al. (2004) [95] propose a methodology for monitoring SBAs, so they can be adapted if

an error occurs or if QoS requirements are not met.

Adaptation of an SBA can be partially or fully automatic. A scenario where adaptation is partially automated is where a service becomes unavailable requiring an actor to choose from alternative services using functionality built-in to a SBA. In a fully automatic SBA, this substitution could be enacted automatically by the application based on the QoS or availability of alternative services. SBAs can be adapted either statically or dynamically as per the definitions for static and dynamic adaptation provided in Section 2.5.

Section 2.6.1, Section 2.6.2 and Section 2.6.3 present approaches for adapting SBAs from the service engineering literature, software engineering literature, and the systematic literature review carried out for this research. These sections present the approaches, while associated development challenges are presented and discussed 2.7.

2.6.1 Processes for Adaption from Service Engineering Literature

After a networked literature search, the author et al. [64] have reported several SBA development process models which support adaptation. Some of these SBA development process models as well as some other relevant approaches from the literature will be reviewed. The complete list of approaches identified by the author et al. are reviewed in [64] (Appendix C).

2.6.1.1 S-Cube

The S-cube life cycle [5] is a high level life-cycle proposed for the development of adaptable SBAs, the life-cycle consists of two cycles as can be seen in Figure 2.8, the left hand side of the life-cycle is an operational cycle that deals with the runtime adaptation, while the right hand side represents the traditional development phases such as requirements engineering, design and construction. It contains many of the phases of the traditional waterfall model [100], while at the same time containing agile characteristics such as iterative development.

The S-Cube life cycle is an amended version of Papazoglou and Van Den Heuvel's [91]

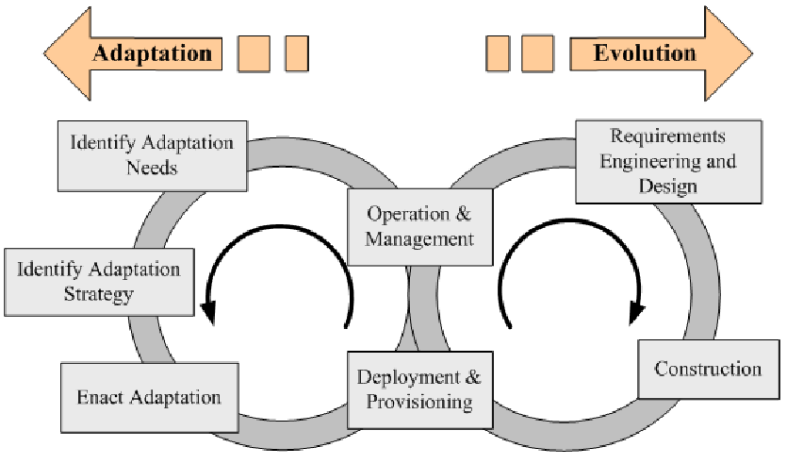

Figure 2.8: *S-Cube Life Cycle for Adaptable SBAs [5]*

service development life cycle. Papazoglou and Van Den Heuvel's original life-cycle was influenced by approaches such as the Rational Unified Process (RUP) [97], Business Process Modelling (BPM) [41] and Component-Based Development (CBD) [42].

This is a high level model which is an ideal reference point for research in the area as it has been developed by a community of researchers within a large European research project[6]. The S-Cube life-cycle as presented here is a work in progress and it presents the processes that need to be followed in order to develop adaptable SBAs. It does not, however, present the activities that need to be followed within each of the processes when developing SBAs.

2.6.1.2 Bucchiarone

Bucchiarone et al. [18] present a context-driven adaptation process for SBAs through the use of a context model with various dimensions such as, *Time Context*, *User Context* and *Service Context*. As illustrated in Figure 2.9 the activities and design artifacts from their process were mapped to an adaptable SBA life-cycle previously presented in [19] [7]. A thorough evaluation of the process model is given and its validity is illustrated by showing its applicability to an E-Government case study. This is one of the more process

[6] http://www.s-cube-network.eu/

[7] The white boxes represent the original actions and artifacts, while the gray boxes represent the new context specific additions.

focused approaches encountered in this review with low level details of many activities and development artifacts.

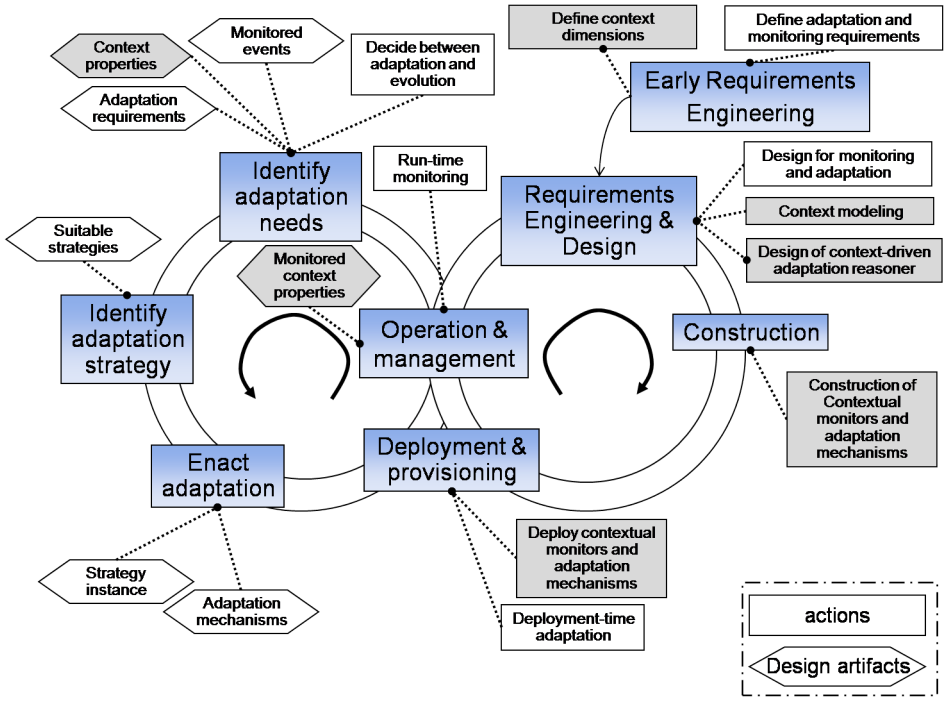

Figure 2.9: *Context-Aware Adaptable SBA Life-Cycle [18]*

Like many of the other approaches encountered, this process is not based on any empirical data which leaves it lacking somewhat in this respect. It does, however, provide a good frame of reference which could provide the foundation for an empirical study. As contributers to the S-Cube project, Bucchiarone et al. [18] map the activities within this process model to the life-cycle phases of the S-Cube reference life-cycle.

2.6.1.3 Web Services Development Life Cycle Methodology

The Web Services Development Life Cycle Methodology (SLDC) [91] is influenced by several established life-cycles such as RUP [60], CBD [42] and BPM [41]. The life-cycle contains one preparatory planning process and eight other incremental processes: “Analysis”, “Design”, “Construction”, “Testing”, “Provisioning”, “Deployment”, “Execution” and “Monitoring”. Along with the life-cycle, the methodology contains a number of principals such as service coupling, service cohesion and service granularity that aid in the development of SBAs. The SLDC methodology contains adaptation specific activities

such as QoS monitoring and alerts for compliance failures.

2.6.1.4 SeCSE methodology

The SeCSE methodology [8] is a set of functional areas and processes that focus on service-centric engineering, service engineering and service acquisition. The methodology also provides practitioners with the information required to adopt the various tools and methods developed by the SeCSE consortium. The SeCSE methodology is conveniently divided into two sections: design time processes and run-time processes. Design time processes contain many of the traditional software engineering processes such as analysis, design and development, while the run-time processes contain mostly service centric processes such as service binding/rebinding, run-time service composition and recovery management. Processes such as run-time service composition and service monitoring illustrate that the SeCSE methodology was designed with adaptation in mind.

2.6.2 Processes for Adaption from Software Engineering Literature

In traditional software engineering, the maintenance process shares many parallels with adaptation. Both are concerned with the modification of an application to meet evolving requirements. The obvious difference is that SBA are easier and cheaper to adapt then traditional applications. This is particularly true when SBAs are designed to be adaptable from the outset. The following subsection reviews some maintenance approaches and how they might be useful for adaptation.

2.6.2.1 Software Maintenance Process Models

There are many software maintenance models proposed in the literature. Models from Martin and McClure [74] or Parikh [92] are simple and loosely defined, while others such as Sharpley [102] and Yau [115] provide more detailed approaches. The age of maintenance models also differs greatly, with Bohem's [15] model dating back to 1976 when software engineering was in its infancy, right up to the present day standards such

as ISO/IEC 14764 [48].

The earliest models that can be found in the literature are generally less complex or detailed than the more recent literature. Bohem [15] proposed one of the earliest maintenance process models with three processes: understanding the software, modifying the software and re-validating the software.

At the other end of the scale both ISO/IEC and IEEE have published comprehensive standards for the software maintenance process. IEEE published IEEE 1219 [47] in 1998 which was an elaboration of the maintenance process from the IEEE 12207 [49] software life-cycle process standard. ISO/IEC published the maintenance standard ISO/IEC 14764 in 1999. However in 2006 ISO/IEC and IEEE combined forces and ISO/IEC/IEEE 14764-2006 replaced the previous versions of IEEE 1219 and ISO/IEC 14764. ISO/IEC/IEEE 14764-2006 is one of the most elaborate maintenance process models to date, with detailed explanations of the activities in each process of the model.

2.6.2.2 ISO/IEC 14764

ISO/IEC-15504 contains a detailed reference process model which covers most of the process areas in software engineering. The reference process model from ISO/IEC 15504 is also published as the separate standard ISO/IEC 12207. ISO/IEC 12207 was first published in 1994 and contained descriptions for sub-processes from the software maintenance process. ISO/IEC 12207 contains the following sub-processes: “Process Implementation”, “Problem and Modification Analysis”, “Modification Implementation”, “Maintenance Review/Acceptance”, “Migration and Retirement”.

The standard was updated in 2008 to include a purpose and outcome for the software maintenance process. The reference life-cycle from ISO/IEC 15504 has descriptions for each process in the software engineering life-cycle. They need to be relatively concise. Otherwise, completing a capability assessment would become too labor intensive. Generally, there are more detailed ISO/IEC standards for the individual process areas from the software engineering life-cycle. In the case of the maintenance process, there is a separate standard ISO/IEC 14764, which contains more detail than the process description

from ISO/IEC 15504 or ISO/IEC 12207. It specifies details of the inputs, tasks, controls, supports and outputs for each of the sub process for the maintenance process. Processes and their associated tasks in ISO/IEC 14764 are summarised here. Each process also has inputs, controls, supports and outputs which are not discussed.

Process Implementation requires maintenance plans and procedures to be created. The maintenance plan should document the plan for carrying out maintenance, while the maintenance procedures should contain more specific details for implementing this maintenance. Modification Request/Problem Report procedures are also listed. Procedures need to be put in place for receiving, recording and tracking modification requests and problem reports. A "Configuration Management" process also needs to be put in place to track the modification of an existing system.

Problem and Modification Analysis requires modification request and problem report analysis before deciding on how to proceed with changes. This may involve scoping the maintenance, documenting possible solutions and documenting impact on existing systems. Similarly the maintainer will need to verify or replicate the problem or issue. The maintainer needs to develop options for implementing the modification. Options to be developed include alternative work-arounds or solutions. Finally, the maintainer needs to document and ensure approval of the the modification request or problem report, the analysis and potential solutions.

Modification Implementation requires the maintainer to carry out analysis in order to determine which documents and software versions need to be modified. After the analysis, the required software changes should be implemented during the development process.

Maintenance Review/Acceptance is a process which involves the maintainer carrying out reviews to ensure the integrity of the modified system. Following this task, the maintainer seeks approval from the appropriate authority that the maintenance has been completed satisfactorily.

Migration begins with the identification of all software or data that is modified if migration from an old platform to a new platform is performed. If migration is going

to occur, it is necessary to create and document a migration plan and then execute the migration according to the plan. Prior to migration a notification of intent should be provided to all system users before migration occurs. Following migration the old and new environments should be run in parallel while providing training to end users in order to ensure a smooth transition. Once migration has been completed, a notification of completion needs to be sent to the appropriate stakeholders. Post migration review should be conducted after migration in order to assess the impact of the migration. Finally, all of the data associated with the old environment should be archived in accordance with the appropriate data protection and audit policies.

Software retirement takes place once a decision has been made to retire an active software product. A retirement plan should be developed and documented by the system maintainer. After deciding to retire software, a notification of retirement intent should be sent to the appropriate software product stakeholders. During retirement, a parallel operation of new and retiring software software should be carried out along with the training of end users. Once complete, notification should be sent to the appropriate stakeholders and finally, data relating to the retiring product should be archived should it be required at a later date.

2.6.2.3 CMMI™Maintenance Process Interaction

The Capability Maturity Model Integration (CMMI™) is divided into four primary process area groups, each of which contains several processes relating to that group. None of the process area groups contain a process specifically designed for software maintenance. In order to address the maintenance process, the CMMI™documentation points to the engineering process area group which contains processes for technical implementation. The CMMI™suggests the use of the engineering processes for both new development and maintenance activities. The engineering process area group contains the following processes: “Requirements Development”, “Requirements Management”, “Technical Solution”, “Product Integration”, “Verification” and “Validation”.

Since the CMMI™process model does not specifically address the maintenance pro-

Table 2.1: *Adaptation Specific Process Model Summary*

Adaptation Process Model	References	Adaptation Approach	Based On	Development Approach	Evaluation
ProDAOSS	[26]	SBA Adaptation	I-Tropos	Agent-oriented development	Case study
PLASTIC	[58]	Service Adaptation	Bespoke	Other	None
Muliview SOAD	[25]	Service Adaptation	SOAD	MDD	None
BCDF	[59]	SBA Adaptation	Bespoke	Other	Worked Example
Chang et al	[9, 51]	Service Adaptation	SOAD	Other	None
CSOMA	[17]	Service Adaptation	SOMA	MDD	None
CSOA	[110]	SBA Adaptation	Bespoke	MDD	Worked Example
Dino	[72]	SBA Adaptation	Bespoke	MDD	None

cess, concerns that are specific to maintenance may not be adequately represented during implementation. For example, the process for the analysis of problems and modifications as specified in ISO/IEC 14764 is not described and may not get addressed depending on ones interpretation of the CMMI™"Engineering" processes.

2.6.3 Processes for Adaptation from Systematic Review

In addition to the networked search conducted in the previous section (2.6.1), the SBA development approaches from the mapping study described in Section 2.4 were also examined for their ability to support adaptation. In this section, some of the more prominent approaches that support adaptation will be reviewed. The complete list of approaches are listed in Table 2.1. The first two columns of the table provide the name and citation for each of the models respectively. The third column states whether the model is concerned with service adaptation or the adaptation of SBAs. If the model is based on another model, the model that it is based on is shown in column four. If the model is not based on another model column four contains "Bespoke". Column five describes the development paradigm followed, while column six states whether or not the model has been evaluated.

Detailed reviews of each of these approaches are provided in the published SLR [65] (Appendix B).

2.6.3.1 ProDAOSS

Achbany et al. [26] propose a Process for Developing Adaptable and Open Service Systems (ProDAOSS) as a plugin for the I-Tropos framework. I-Tropos is a comprehens-

ive end to end agent-oriented development methodology which assigns a crucial role to requirements analysis and specification processes [37]. The ProDAOSS process model concentrates on three process areas. The first is "Organizational Modeling and Requirements Engineering" using a framework called FaMOs which models at the service level. The second contribution of ProDAOSS is an "Architectural Design" process which uses a Multi-Agent Software (MAS) architecture to facilitate the adaptation of SBAs. This aspect of ProDAOSS appears to focus on providing an architecture rather than specifically outlining process details. Finally, the last item described in ProDAOSS is a "Detail Design" process area that contains process details for a Reinforcement Learning Model which focuses on exploration and exploitation and a Probabilistic Reputation Model used to estimate the reputation of a service providers' QoS. The ProDAOSS process model has been applied to two case studies which contributes towards its validity.

2.6.3.2 BCDF

The Business Collaboration Design Framework (BCDF) [59] is a rule based development approach for the development of adaptable SBAs which facilitates ad-hoc business collaborations based on predefined rules. They propose many types of rules such as business rules, operational rules and services rules, all of which are specified using an XML based standard called RuleML. Specifying the rules as RuleML allows them to be transmitted to third parties as well as for use in automated processes. This is a useful approach that can be employed for automatic adaptation as the rules that are encoded in RuleML may be parsed at runtime to guide SBA adaptation. One of the key challenges with this approach is the accurate representation of business rules. If the coded rules do not accurately reflect reality then the consequences of any subsequent adaptation may be undesireable. Orriens et al. [59] illustrate the feasibility of BCDF with an example in the auto-insurance domain. Unfortunately, the example focuses on how rules are encoded and interpreted, rather than how BCDF fits into the overall development process.

2.6.3.3 CSOA

CSOA [110] is an MDD approach for context-aware development, however, in this case CSOA relates to the development of context-aware SBAs rather than services. The approach employs viewpoint principles from the Object Management Groups (OMG) Enterprise Collaboration Architecture (EDOC-ECA). Among the many viewpoints used in the modeling process is an adaptation view which facilitates the modeling of Business Components, Contextual Components and Connectors which handle bindings and middleware activities between business and context components. An application of the approach is illustrated using a mobile GPS application as an example, where the application adapts to meet a contextual environment when the user is a wheelchair user. In this case the application adapts by putting more emphasis on map items such as pedestrian crossings which are necessary for wheel chair users while also providing voice input functionality for hands free use. This is a useful approach with the added benefit of providing adaptation at the application level rather than just at the service level. Unfortunately, the approach focuses on specifying the MDD architecture and does not give much detail on how the approach fits into the overall development process.

2.7 Development Challenges for Adaptable SBAs

This final section of the literature review looks at some of the challenges faced by existing approaches for developing adaptable SBAs. Some of the approaches are reviewed in Section 2.6.1, Section 2.6.2 and Section 2.6.3 but it would be useful to have an impression of the common challenges that face them. The following sub-sections will look at the more common challenges encountered in the literature with reference to where they have been reported. These challenges will have to be overcome by any proposed process reference model.

2.7.1 Runtime Adaptation

There were five runtime adaptation related challenges identified in the literature. Runtime adaptation challenges relate to the class of challenges that occur while an adaptable SBA is in operation. These challenges are discussed in the following list.

Adaptation needs to support evolution in the runtime environment [25, 12] This is the most frequently identified challenge in this category. This challenge, outlined in Multiview SOAD [25] and some of the approaches reviewed in [12], captures one of the key motivators to developing service-based applications. The ability to adapt to a changing runtime environment is one of the key benefits but also the key challenge to developing adaptable SBAs.

The evolution in runtime environment refers to the evolution of the application's execution context. This may be due to changes in user context, a change in operating system or platform, the failure of component services or an unforeseen execution path being introduced to the application. With the recent popularity of mobile and ubiquitous computing the ability of being able to adapt based on a users location or mobile device type provides many possibilities. However, this also adds many complexities into developing applications that can take advantage of this technology. These development issues need to be addressed by a process that can facilitate the development of these adaptable applications.

It is desirable to achieve automatic adaptation [12, 19, 108] ASTRO [108] and the approach described by Bucchiarone [19] identify that the ability for the application to be able to adapt, through automatic means, poses many challenges. For example, if the execution context of the application changes and the SBA needs to self-adapt it needs to have some execution engine that can calculate feasible execution paths. The execution engine would need to be able to calculate suitable execution paths within the operational parameters allowed by the application.

A more limited form of self-adaptation is where all of the adaptation configurations

are hard coded into the application at design time or added with an adaptation dashboard during runtime. In this case the list of adaptation configurations are finite and pre-conceived removing the uncertainty from the running SBA.

It is desirable to have some level of human control in automatic adaptation [18] In certain circumstances adaptable SBAs may require some level of governance prior to adaptation being enacted. In some mission or safety critical applications governance is necessary to prevent system downtime or dangerous events from occurring. Manual adaptation is more easily governed because the process intrinsically involves human actors who can decide if an adaptation is dangerous before it is enacted.

In dynamic environments it is necessary to adapt quickly [18, 12] This challenge relates to the speed at which adaptation can be affected during the operation of an SBA. The speed of adaptation is often relative to the type of adaptation being enacted and the type of application. In a manual adaptation scenario, quick adaptation may be a turnaround of a week or a few days. In an automatic adaptation scenario in an application with a high volume of transactions, adaptation may only be considered fast if each execution is a few milliseconds. Therefore, a requirement for fast adaptation needs to be considered in the context of the overall application and business requirements.

A key component in being able to adapt quickly is to be able to discover suitable services and consume them in a short period of time. Again, the speed of discovery here is relative to the type of adaptable SBA being developed. More specific challenges relating to service discovery follow in the service discovery section.

Ad-hoc runtime service composition adaptation is necessary [12] Runtime service composition or re-composition is a specific type of adaptation where services compositions are re-composed rather then individual services being exchanged. This type of adaptation is more challenging than service substitution because of the num-

ber of failure modes that are introduced into the application. This also makes troubleshooting application failures more difficult.

Again, service discovery and negotiation challenges are introduced here when the SBA is recomposed with services that were not already present in the application. Service re-composition is the most complicated type of adaptation and providing mechanisms for such is a big challenge to developers.

2.7.2 Development Process

There were several challenges identified in the literature that related to the development process for adaptable SBAs. Development process challenges in this context relate to challenges with the process for developing adaptable SBAs. The challenges are focused around the methods and techniques necessary to guide their development. Here a brief description is given of the process challenges identified.

A methodology for developing adaptable SBAs is required [18] This challenge was identified in several publications, including Bucchiarone [18], suggesting that it is a priority amongst researchers and practitioners. This is a high-level challenge which in part justifies the work being carried out in this thesis. Existing development methods do not lend themselves easily to the development of SBAs due to the distributed nature of services and the different roles played by service providers and service consumers.

Existing adaptable SBA development methodologies are too fragmented [91, 26, 58, 17, 72] This challenge, identified in the SDLC [91], proDAOSS [26], PLASTIC [58], CSOMA [17] and Dino [72] approaches, reflects the fact that there are many approaches available for many different aspects of adaptable SBA development. Another problem with the approaches available is that they are at different levels of abstraction. For example, some of the approaches deal with the business integration, some with the infrastructure layer and others with the application layer. A generic SBA development methodology or process needs to be at a higher level of abstraction than all

of these layers and provide guidance to each of the development project stakeholders.

In order to over come this challenge an adaptable SBA development process that spans the entire development life-cycle is required. This processes needs to show how each of the project stakeholders interact with the tasks and activities within the process. Additionally this processes needs to be generic enough that it can be used in conjunction with existing development life-cycles.

2.7.3 Monitoring

Monitoring in this context relates to the automatic or manual monitoring of attributes of an adaptable SBA to give an indication of when adaptation might be necessary. It is therefore a highly integral part of adaptation and this is reflected in the number of monitoring challenges that were identified from the SBA adaptation approaches. There were 6 approaches identified with monitoring challenges from the approaches reviewed in [12]. This reflects the general need for monitoring with adaptation. The challenges are described in the following sub-sections.

Efficient monitoring implementations and architectures are necessary [12] In the literature, it is generally accepted that monitoring is necessary for adaptation. Monitoring in itself is not a difficult problem but it can be difficult to implement monitoring mechanisms that do not have a performance footprint. The problem with monitoring performance comes into play in applications with a very high volume of transactions. Therefore, the challenge is to provide monitoring mechanisms that can signal adaptation triggers without affecting the performance of this type of application.

It is necessary to define runtime operational criteria for adaptable SBAs [12] Defining runtime operational criteria for an adaptable SBA is difficult since an application may consume unknown services. Therefore, the operational criteria needs to be

flexible enough to suit a wide variety of services while at the same time ensuring QoS and reliability.

The key operational criteria for services consumed by adaptable SBAs relate to their performance, cost and availability. In order to evaluate these criteria it is necessary to monitor non-functional service parameters. This monitoring will allow action to be taken when non-functional service parameters exceed their limits.

Monitoring approaches are highly fragmented [12] This challenge relates to low level monitoring implementations as well as general monitoring processes. Many monitoring approaches have been implemented in the literature with many focusing on specific aspects of monitoring on specific platforms. For example, a monitoring approach that just focuses on service performance for web services is not holistic enough to guide the monitoring of an adaptable SBA. Monitoring approaches need to facilitate the monitoring of all of the types of parameters necessary for functional and non-functional adaptation. Along with specific implementation approaches it is necessary to have process guidelines that will work with a variety of monitoring implementations and platforms. Existing approaches need to be consolidated and supplemented where necessary to fill any gaps.

The fragmented nature of existing monitoring approaches make diagonalisability difficult [12] This challenge related to the preceding challenge in that it relates to fragmented monitoring approaches. In this case it relates specifically to diagnosing application errors when they occur. If there is a single monitoring approach with monitoring logs and databases within a single repository then this makes diagonalisability more straightforward. If multiple monitoring mechanisms are used in the same solution then diagnosing a problem involves querying multiple monitoring repositories. Using multiple sources of monitoring data greatly adds to the complexity of troubleshooting even basic errors.

The execution context of the SBA needs to be monitored [12] monitoring the executing context of an adaptable SBA is difficult but it is necessary to provide the full

benefits of adaptation. When the application knows details such as the platform that it is being run from, the physical location of the running device and the application user profile, then many powerful adaptations can be made. In the case of a travel booking application, if the application knows that the user is a business person and the operating environment is a smart phone with a GPS chip then it can offer the business person suitable travel options based on location, desired destination and estimated budget.

The main problem with monitoring the execution context of an application is that the application needs mechanisms to interact with its environment. The application need strategies to determine a users profile which opens a wide variety of privacy issues. In addition if the application needs to interrogate external application programming interfaces then this introduces security and reliability issues into the application.

2.7.4 Adaptation Tools

Adaptation tools are the tools used during the development process for adaptable SBAs. Specialized tools are often used in the design and development of service specific artifacts since they are often low level and complex. In the following list the challenges that relate to adaptation tools will be discussed.

Service composition solutions are very complex [59] A key challenge with service composition is that by its nature it is complex. Looking at Web Services for instance there are complex standards for the description of service interfaces (WSDL), there are complex message standards (SOAP) and complex composition standards (BPEL). Adaptable SBAs that are based on platforms such as this require complex functionality to integrate the various components of service composition. In order to ameliorate this complexity the complex protocols and standards that are involved need to be abstracted so that the developers can focus on the complexity of the domain problems rather then the technology standards. In terms of software methodology,

tools that can solve these complexity problems need to be integrated into the development process.

Service composition solutions do not cater for adaptation [59, 110] As it stands many software composition solutions do not cater for the more advanced types of adaptation. There is a need for mature composition solutions to support adaptation types such as service re-composition and service re-negotiation. The state of the art composition solutions such as BPEL only support hard coded "if else" type adaptation. The solution to this problem may be to combine some type of adaptation engine with existing composition engines.

Existing adaptable SBA development tools are too fragmented [12] This challenge reflects the challenge of fragmented adaptable SBA development processes. Existing development tools need to be combined so that collectively they can be used to support the development of adaptable SBAs. As it stands existing tools focus on specific technologies and do not interoperate with one another. There are many modelling and design tools that are suitable for designing adaptable SBAs, but they often do not have capability of integrating with development tools such as code generators.

2.7.5 Service Terminology

Service terminology relates to the terms and definitions that are exclusively used in SOC and with SBAs. There is one challenge in the service terminology section. The challenge is that **Service definitions are too fragmented**. This challenge was elicited from Durvasula et al. [32] which is an industry developed process for the development of SBAs. While this challenge does not relate directly to adaptable SBAs, it is relevant to service discovery and negotiation which are necessary for adaptation. Conflicting service terminology poses problems for business integrators and developers that need to exchange service details before a business integration can take place. Conflicting terminology is also a problem when the terminology used in methodologies do not match the terminology in

tools and standards. This problem is magnified when fragmented SBA development tools and methodologies are used in a single project.

2.7.6 Service Adaptability

The service adaptability class of challenges relates to the challenges that occur when adapting simple services rather than SBAs or compositions of services.

There is single challenge in the service adaptability section. The challenge **Services need to be adaptable** [58, 25, 9, 51, 110], identified in PLASTIC [58], Multiview SOAD [25], Chang et al [9, 51], and CSOA [110], relates to the fact that it is often useful for services to have some level of customisability when there are consumed by multiple SBAs. This challenge is particularly important in the case of adaptable SBAs where it may not be possible to adapt the SBA to suit the service. This obstacle may be overcome by having functionality in the adaptable SBA to deal with service variability.

2.7.7 Service Discovery

In this section two challenges that relate to the service discovery are discussed. Service discovery relates to finding services either automatically or manually on service directories. Service discovery is a process in its own right but it is also a sub-process of adaptation in some cases.

Runtime service discovery is difficult [8, 110] This is quite an obvious challenge in that service discovery is clearly a challenging process. When Web Services were originally conceived, the service discovery scenario proposed was that services would be automatically discovered from databases such as Universal Description, Discovery and Integration (UDDI) registries. In practice this is rarely the case since companies were slow to publish their service on such registries.

A more typical service discovery scenario might involve searching human readable service directory websites and then short listing service alternatives in an internal

application database to be discovered by a running application. Which ever scenario is adopted, a service discovery process would be beneficial in making such a complicated process more straightforward.

There is a problem matching service providers to clients [9, 51] This challenge relates to the difficulty that service providers have making their services available to service consumers. One of the main reasons for this difficulty is the failure of public service directories to become established. This has left service publishers with only directory web sites of various quality to publish their services. Another problem with these directory web sites is that they are usually not computer readable making automatic service discovery and negotiation impossible.

2.7.8 Enterprise Integration

There are several SBA adaptation challenges in the enterprise integration category. They will be briefly discussed in the following list. These challenges relate to the integration of enterprise resources that are necessary to realise SOA and SBAs.

Enterprises need to be able to easily integrate their services [59] Enterprise integration with software services promises great flexibility to companies, but in order for this integration to be realised service integration must be straightforward. There must be a simple mechanism for companies to publish their services and allow other businesses to consume them.

With the demise of service registries such as UDDI this has become even more of a challenge. Large companies also don't often report their service descriptions on public web site directories for security and related reasons. A solution to this problem might be for companies to first initiate business contact with one another then exchange service details that can be used to support any business collaborations.

Organisations processes need to be interoperable [32] This challenge relates to the granularity of business processes within organisations. In order for business processes

and business services to be combined with one another they need to be at the same level of abstraction. A useful guide here is to define processes that can be used to complete logical tasks within a single domain. For example, in the logistics domain it might be useful to have business services to "pack orders", "ship orders" and "receive proof of delivery" for orders. These sample services can be combined with one another or with another organisation's services to complete business integrations.

2.7.9 Service Quality

It goes without saying that service quality is a vital factor for adaptable SBAs that rely on others for the provisions of services. Service quality relates to non-functional quality attributes like service availability and service response times.

Services should meet users QoS requirements [58] The two main QoS requirements of interest to service consumers are service availability and service response times. It is not good enough to have services that are available, they also need to be able to respond in a timely manner. Response time is particularly important in SBAs because multiple service delays get compounded into overall unacceptable application performance. Performance time is one of the major problems with SOA technologies such as Web Services due to the bandwidth and processing requirements of XML type messages.

Services are not under the control of the SBA developers [12] This challenge relates very much to the quality of services. If services are not under the control of the application or application developers, then QoS cannot be guaranteed. This issue cannot be completely overcome in SOA but measures can be put into place to ensure that service providers stick to service level agreements for specific quality attributes.

2.7.10 Service Semantics

Service semantics relate to concepts such as formal service descriptions and how these descriptions can automatically be interrogated by service consumers.

Services need to be described adequately for service consumers [8] . This service description challenge was identified in the SeCSE methodology [8]. In order for service consumers such as adaptable SBAs to be able to consume services which meet their requirements they need accurate service descriptions. These descriptions should be human readable for SBAs that do not adapt automatically and computer readable when adaptation is automatic. Computer parseable semantic service descriptions allow for powerful real-time service selection. In reality it is difficult to find different service providers that publish their services with standard semantic descriptions.

It is necessary to be able to evaluate service specifications for adaptation suitability [12] This challenge relates to the necessity to be able to interrogate service specifications to check that they are suitable for consumption by an adaptable SBA. Service specifications may include information on service security, messaging types, privacy, transaction, management and other such specifications. If a service is implementing particular specifications it is important for an adaptable SBA to be able to support those specifications before the service can be consumed. Using the correct specifications maximises factors such security, transaction management and a variety of other important factors.

2.7.11 Service Specifications

There is a single challenge in the service specifications category: **Services need to be at the same granularity as business processes** . This challenge, identified in the SDLC approach [91], is problematic for organisations that want to overlay a SOA over existing applications and components that are often not at the correct level of granularity. This

issue needs to be overcome so that business services can be combined into logical pieces of work. For example, a finance application should have services for entering orders, entering payment details and processing credit cards. It is not very useful to have low level services that, for example, returns a row in a database, returns an entity relationship or displays the value of a variable.

2.7.12 Domain Knowledge

The challenge **Domain knowledge needs to be integrated into adaptable SBAs** [12] relates to how domain knowledge should be encapsulated into adaptable SBAs so that it can be reasoned about for adaptation. Domain knowledge can be used to to guide the workflow of adaptable SBAs. Domain specific languages can describe valid business processes and rules that can be parsed by the SBAs. This prevents them from entering illegal states or states that will not end in successful execution.

2.7.13 Service Testing

Like a lot of other adaptable SBA challenges, the challenge that **Some SBA properties can only be evaluated during production settings** [12] cannot always be overcome but can be managed to minimise the impact on the application. SBA properties such as availability of services or the effect of combining services in unpredictable ways cannot be tested in advance. Therefore, it is necessary to monitor adaptable SBAs very carefully for problems after adaptation takes place and have a plan to revert to a working configuration if problems occur.

This is one of the key challenges that needs to be addressed in service engineering due to the unpredictable nature of software services. In adaptable SBAs with finite adaptation configurations, it is possible to test application properties during each of these configurations but a more innovative solution is needed when there are very many or infinite adaptation states.

2.8 Literature Summary

In this chapter I have looked at the state of the art in the areas of service-oriented computing, software process and software adaptation. These are all of the areas that are relevant to the development of adaptable SBAs.

The software process section provides a chronological view of software development process reference models. Starting with linear an iterative models, leading up to the more recent SCRUM hybrid model. Finally, the more advanced CMMI and ISO-15504 capability and improvement models were presented with reference to how they are applied compared to standard process reference models.

Based on this literature review, it seems like a linear or iterative waterfall type model would provide a suitable starting point for a reference model for developing adaptable SBAs. The waterfall model is a straightforward process with all of the key processes required for developing software applications. The waterfall model shares a lot of its process with many other mainstream life-cycle models. Therefore, it should be possible to map a process reference model based on the waterfall model to a lot of other life-cycle models. Basing the reference model on a more specialised life-cycle could make it difficult to integrate with other life-cycles.

In the adaptation section, software adaptation as a general concept was presented followed by descriptions of adaptation approaches for component-based and service-oriented development. After a networked literature search, several service-oriented adaptation approaches were identified. A more comprehensive systematic literature review, reported in Section 2.6.3, was conducted to identify additional approaches. This rigorous approach was undertaken because the approaches identified have been used as the starting point for the construction of a process reference model for adapting SBAs.

While there are several approaches proposed for the development of SBAs and for the development of adaptable SBAs, the literature shows a demand for a valid end to end process reference model for the development of adaptable SBAs that is based on empirical data. Similarly, it has been shown in Section 2.7 that the approaches proposed

for developing adaptable SBAs themselves face challenges that have not been overcome in their entirety by a single approach.

2.9 Research Objectives

From carrying out the literature review, it was established that, although models exist, they do not adequately support adaptation and are not empirically grounded. In order to address this gap, the first objective of this research is to develop a process model, using appropriate data sources and research methods, that can be used by practitioners who want to develop adaptable SBAs. The process model should only address adaptation processes and activities rather than being an all inclusive life-cycle model. This limits the scope of the problem and facilitates integration with existing life-cycle models such as the RUP [97], Spiral [16] or the waterfall models [100]. Section 3.3 proposes a research design to meet this objective.

Once the process model has been developed, the second objective is to ensure that the model is valid. Validation proves that the model is credible and can be applied generally to the problem of adapting SBAs. Internal validity can be achieved by following rigorous research methods while external validity can be demonstrated by evaluating the models generalisability and transferability.

Chapter 3

Research Methodology

3.1 Introduction

This chapter deals with the methods of inquiry used during the research and justifies them with reference to their underlying philosophical perspectives. Firstly, Section 3.2 identifies which philosophical paradigms were best suited to meet the objectives of this research project and also deals with the philosophical perspectives associated with those paradigms. Then once a philosophical basis was established for the research, suitable research methods were selected and constructed into a research design which is outlined in Section 3.3.

3.2 Research Philosophy

In order to decide upon the best research design it was important to consider the various methods of inquiry and their philosophical underpinnings. There were many research paradigms and philosophical perspectives to choose from, many of which are at different abstraction levels. These competing approaches are discussed in Section 3.2.1 while their philosophical perspectives are discussed in Section 3.2.2.

3.2.1 Research Paradigms

Research paradigms, the phrase that is commonly used in the literature [24] [45], refer to the high-level perspectives on how research should be conducted. These research paradigms in turn have related lower level ontological, epistemological and methodological perspectives that will be discussed later. Therefore, the selected paradigm or

paradigms will affect the selection of specific research methods for the research design. The first research paradigm considered was positivism.

Easterbrook et al. [33] define positivism as a view that:

> "knowledge must be based on logical inference from a set of basic observable facts".

Myres [79] describes positivism as the view that reality can only be described by properties that are independent of the researcher and the researcher's instruments. In other words the research process should not affect the outcome of the research. Detmar et al. [28] explain how Karl Popper's theory of falsification takes this one step further by stating that a theory is only scientific if its predictions can be observably falsified. The logic behind falsification is that, if a theory cannot be tested, then it is impossible to know if it is true, suggesting that it is a myth rather than a scientific theory. Orlikowski and Baroudi [86] have identified several characteristics of positivist research:

- Research with formal propositions.
- Research with variables that can be quantifiably measured.
- Research that tests hypotheses.
- Research that uses a population sample to draw inferences about a phenomenon.
- Research that aims to describe rather than interpreting a phenomenon.

Based on the analysis of the above list, positivism alone is not the best approach for collecting and analysing the qualitative data needed to develop a process reference model. While a positivism approach encourages repeatability and objectivity which is desirable, it does not allow the researcher the flexibility to interpret and reason about ambiguous or disjointed concepts.

Interpretivism is an opposing view of reality to that of positivism, interpretivists do not focus on measuring properties, hypotheses testing or similar "hard" scientific methods

[57]. They focus on understanding phenomena through social constructs such as shared understanding, language and consciousness. Howcroft and Fitzgerald refer to interpretivism as "soft" research but stress that "soft" interpretivist research is not inferior to "hard" positivist research because both approaches have their strengths and weaknesses.

Inevitably there are interpretive elements involved in research where the inquirer collects data by interviewing and interacting with research participants. The interview process involves shared understanding and subjectivity between the interviewer and the participant. Where suitable I will take an interpretivist approach to data collection and analysis. Positivism will be incorporated into aspects of the research design such as sampling, data coding and other activities that can be handled more objectively.

3.2.1.1 Critical Theory

Critical theory has its origins in critical social theory, where researchers have attempted to bring the restrictions of accepted social reality to light [79]. Ngwenyama and Lee [81] explain that critical social theorists have a responsibility to not only understand a social phenomena, but to be critical of unjust and inequitable situations. In terms of software engineering, the same applies, critical researchers must first attempt to understand a social phenomena within the research area and then affect it by highlighting its deficiencies and inadequacies.

In terms of this research project, there will be a critical element to the process model that is being developed. Once the model has been developed it will be critiqued by practitioners through a validation process to ensure that is is valid and does not contain deficiencies.

3.2.2 Philosophical Perspectives

Fitzgerald and Howcroft [45] identify four categories of research philosophies, each at different levels of abstraction: ontological level, epistemological level, methodological level and axiological level. The following subsections briefly define each level and their

containing philosophies that relate to this research.

3.2.2.1 Ontological Level

Ontology refers to the nature of reality, there are two primary stances on how reality exists. The first stance, relativism, claims that there are multiple versions of reality based on the relative subjectivity of different minds. The second stance, realism, is the opposite. It claims that there is only one reality that is independent of any individual's mind.

Both of these standpoints are logical, and, in a way, both are true. It is hard to escape the fact that individuals are subjective no matter how objectively they approach a task. Therefore, an individuals view of reality is bound to be subjective. On the other hand, in order for an individual to have a subjective perspective on reality there must exist some underlying constant reality on which their perspective is based.

The primary objective of this research is to construct a process model, with a primary data source being input from subject matter experts. Since it is acceptable that there is subjectivity involved when individuals describe their experiences, it is reasonable to expect that the input provided by subject matter experts will be subjective. It is therefore important that I take this into account when constructing the "true reality" of how best to adapt SBAs. During this research, the subjectivity of participant data will be accounted for by the triangulating of data sources.

3.2.2.2 Epistemological Level

Epistemology is the branch of philosophy concerned with the study of knowledge and the study of knowledge acquisition [103]. Before the research design is constructed it is important to consider relevant epistemological approaches and determine which ones are appropriate for this research. In the context of this thesis as well as any other research, it is important specifically to consider the various theories about how knowledge is acquired. Two knowledge acquisition epistemological perspectives, empiricism and rationalism, are relevant to this research.

Empiricism is a knowledge acquisition theory that claims all knowledge should be

based on sensory experience [73]. This theory is often embraced by software engineers as well as software engineering researchers. Software engineers are mostly concerned with "how" to build software and draw on positive past experiences to guide the software development process. Similarly software engineering researchers draw on the experiences of software practitioners in order to determine best practices for software developers [96].

This thesis is concerned with the development of a best practice process model, so an empirical approach which draws on the positive experiences of practitioners is appropriate. Empirical software engineering research usually consists of field work, such as case studies or surveys, where practitioners are observed or interviewed in their natural setting, or laboratory experiments where software development tasks are observer in a controlled, repeatable environment [94].

Rationalism is an opposing theory to empiricism; while empiricists believe that knowledge comes from experience, rationalists believe that knowledge is attained through intuition or deduction, or can be possessed innately [73]. Rationalism is also a useful standpoint in software engineering research. As well as providing valuable empirical evidence from experiences that they have had, software practitioners can also provide useful opinions and knowledge that they have derived through intuition, deduction or have known innately.

3.2.3 Methodological Level

Research methods usually have distinct data collection and analysis processes. The data that is collected may be quantitative, qualitative or a combination of both types, and the analysis may employ quantitative, qualitative or mixed analysis methods. Quantitative analysis techniques involve the analysis of quantitative data using positivist mathematical and statistical techniques [29]. As the name implies quantitative data is numerical or quantifiable data that can be used to represent properties of a phenomena being studied. On the other hand qualitative data which is usually text based, from sources like documentation or transcribed audio, can be analyzed using positivist, interpretivist or critical

techniques. Therefore, if one decides upon the positivist perspective both quantitative and qualitative methods are available for selection. Qualitative analysis methods are the only methods available to interpretivist or critical theorists since they rely on social constructs such as language and shared meanings which are represented by textual data.

While an interpretivist stance is suitable for the construction of process model, incorporating positivist elements into the research design where possible. This choice remains whether to chose quantitative, qualitative or mixed research methods. What is clear is that the development of a process model requires field work to collect data for the construction of the model. Edmonson and McManus [34] propose a framework for the selection of research methods when conducting field work. The model suggests that different methods are appropriate depending on the maturity of the research area. They describe three categories of research: nascent (immature), intermediate and mature. For each of these categories they suggest suitable, research questions, types of data to be collected, data collection methods, constructs and measures, goals of data analysis, data analysis methods and theoretical contribution. Since the adaptation of SBAs is an emerging topic the nascent category applies. Table 3.1 illustrates the methodological attributes for each step of the research design proposed by Edmonson and Mcmanus for nascent field work. The key attributes suggested are the collection of qualitative data for thematic content analysis which is a qualitative analysis technique. They suggest interviews, observations and the retrieval of documentation as suitable data collection methods.

The term “research methods” as it is used in the literature is quite ambiguous and can be used to describe data collection methods alone or entire research designs [98]. For example, the survey research method is typically used to describe a data collection activity using techniques such as interviews or questionnaires. However, the grounded theory research method refers to an entire research design. In this thesis, research methods will refer to entire research designs and data collection methods will be referred to separately.

State of Prior Theory and Research	Nascent
Research questions	Open-ended inquiry about a phenomenon of interest
Type of data collected	Qualitative, initially open-ended data that need to be interpreted for meaning
Illustrative methods for collecting data	Interviews; observations; obtaining documents or other material from field sites relevant to the phenomena of interest
Constructs and measures	Typically new constructs, few formal measures
Goal of data analyses	Pattern identification
Data analysis methods	Thematic content analysis coding for evidence of constructs
Theoretical contribution	A suggestive theory, often an invitation for further work on the issue or set of issues opened up by the study

Table 3.1: *Methodological Fit in Nascent Field Research (Edmonson and Mcmanus (2007))*

3.2.3.1 Qualitative Research Methods

There are many research designs to chose from some of which specify quantitative, qualitative or mixed data collection and analysis techniques. Since this is an emerging research topic, following Edmonson and McManus's advice, the author will focus on qualitative research methods. Here a brief review will be provided of several qualitative research methods, highlighting the strengths and weaknesses of each, in an attempt to select the most appropriate ones for this research project.

3.2.3.1.1 Action Research Like many other qualitative research methods action research has its roots in social science, although it is currently used in a variety of organisational research domains. Action research was formulated by Kurt Lewin in 1946 [67] as a method of generating theory by the researcher participating in the social system [104]. Therefore, through action research the researcher can influence the output of the research as well as gathering data. Susman and Evered [104] outline five steps in the process of action research: diagnosing, action planning, action taking, evaluating, and specifying learning. Figure 3.1 is a graphical representation of that cyclical process.

Since this researcher is a full time research student and not employed in an organ-

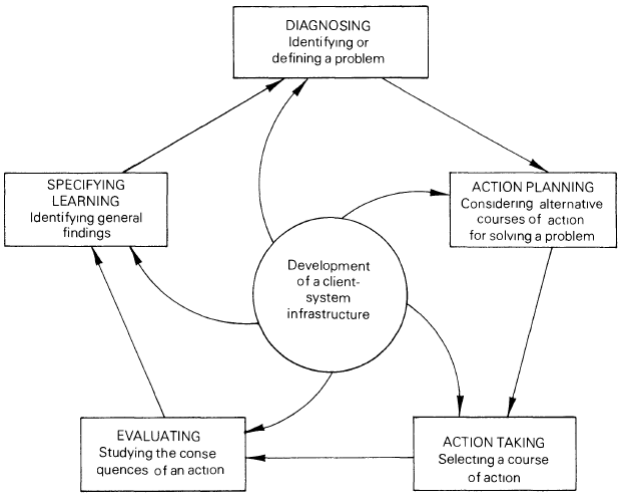

Figure 3.1: *Cyclical Process of Action Research [104]*

isation that develops service-oriented software it is not practical to participate in action research for this project. That being said, action research would be a useful research method for the elicitation of the data required to develop a reference model. Similarly being part of a SBA development organisation would facilitate the implementation of prototype reference models.

3.2.3.1.2 Grounded Theory The grounded theory research method, proposed initially by Glaser and Strauss [38], is a research method that attempts to construct theory by systematically gathering and analysing qualitative data [109]. The main steps involved in grounded theory are the data collection, coding, memoing and writing up the research.

The data collection activity involves the collection of qualitative data through methods such as interviewing and documentation gathering. Then any recorded voice data is transcribed so that it can be analysed. This is a time consuming and laborious process but is necessary in order to make sense of the mass of qualitative data collected during a study.

The next step is coding the textual data gathered and transcribed. There are three types of coding: open coding, axial coding and selective coding. Open coding involves reading each passage of text then assigning a code describing its context. Axial coding is a process where the researcher attempts to assign causal relationships to categories

of codes. Selective coding involves the selection of a core theory or code category and determining how all of the other theory categories relate to it.

Memoing is the activity of noting ideas about coded passages and how they might relate to one another. This is carried out throughout the coding process as ideas come to mind. It is important to memo ideas as they emerge during coding otherwise they may be lost permanently. Finally, the theory is distilled from the memos and coded data and written up, this is the main deliverable from the process that is shared with others.

The methodical activities of the grounded theory coding process is a promising way of processing the data gathered for the construction reference model. However, unstructured data analysis process is not a good match for the development of a structured reference model. Therefore, the coding process alone, adapted by Miles and Hubermann [77], will be chosen for this project.

3.2.3.1.3 Ethnography Ethnography is a research method that involves the observation of a community of people in order to understand how they interact with one another socially [98]. Again, this method has its roots in social science but it is widely used in organisational research. The goal of ethnography is to begin without any preconceived theories focusing exclusively on how community members interact with one another.

In software engineering research ethnography may be used to investigate how a particular development process or activity occurs by observing its participants. The observed community may be broad such as “java programmers in general” or tight such as a single development group [33]. The ethnographic process may involve observation and note taking of interactions relating to a research question and follow up interviews with community members to supplement those notes.

Since ethnography focuses primarily on observation it is very difficult to apply to large problems with complex and voluminous transactions between community members. In the case of the research the author is attempting to capture activities that occur during the entire development process. This suggests that ethnography may not be the appropriate approach in this case.

3.2.3.1.4 Case Study Research Yin describes the case study as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident."[116]. Easterbrook et al. describe two types of case study, exploratory case studies are used to describe some new phenomena or theory, while confirmatory case studies are used to test existing hypotheses.

Like other qualitative research methods it is first necessary to develop a research question before a case study is undertaken. This is then used to formulate a "study proposition" or a "study protocol" which is used to guide the selection of cases and how data is collected for each case. Depending on the research question there are several types of case selection approaches. An extreme case may be chosen to prove a point or falsify a hypothesis. Often it might be desirable to chose a typical case in order to get an overview of a particular phenomenon. In order to generalise about a phenomenon it is necessary to select a variety of cases that can show the phenomenon in different contexts.

Data collection in case studies involves the collection of qualitative data such interviews, observations and the collection of documentation. Data collection should focus around a "unit of analysis" such as a process reference model. It is important to limit the scope of the unit of analysis because case studies can turn up huge amounts of data.

Data analysis can be carried out with a variety of posivitist and interpretivist approaches. Posivitist approaches include coding and catagorising of concepts or categories. Interpretivist analysis approaches involve the researcher subjectively reasoning about and conceptualising the data.

The case study method suits this research project because it allows the researcher to efficiently document development processes within orgainisations. Within this research project, the unit of analysis is the participant orgainisation's software development process. Selecting a variety of different companies or cases allows the researcher to generalise results which is necessary during the development of reference models. An interpretivist approach for analyzing and reasoning about case study data provides the flexibility required to identify the generic process attributes required to develop a process reference

model.

3.2.3.1.5 Survey The survey research method is not just the instrument used to collect data, it is a complete research method with steps such as: setting objectives, designing the survey, developing the survey instrument, obtaining data and analysing data [56]. There are several types of survey mainly categorised by the data collection method employed. The two main methods of data collection are questionnaires and interviews with some overlap between those two methods. A questionnaire may be administered as a printed document, a web based document or can be administered orally as a structured interview. Interviews on the other hand may be structured (an orally administered questionnaire), semi-structured or unstructured.

The type of data analysis that is conducted after the survey depends on the nature of the data collection. If the survey instrument is a quantitative questionnaire then a variety of statistical methods may be used for analysis. Qualitative surveys are usually followed by some type of qualitative content analysis.

Surveys are typically used to describe or explore some phenomenon of interest. This is a particularly useful research method for the capture of organisational process data that is needed to construct a process reference model. Semi-structured and open ended interviews are flexible enough to enable research participants to describe the details of software engineering processes. Therefore, semi-structured interviews were used as a data collection method for this research project.

3.2.4 Axiological Level

The final philosophical perspective in the philosophy of research is at an axiological level. The debate here is between the value of relevance and the value of rigour [45]. The relevancy perspective emphasizes the value of research with respect to external factors such as suitability for a particular purpose. On the other hand the rigor perspective focuses on the internal validity of the research and how the research methodology is precisely followed.

Since the primary aim of this research is to produce a reference model that is useful practically to practitioners and researchers, the primary axiological focus of the research should be relevancy. On the other hand the quality of the research depends very much on the rigor of the research. Therefore, this research will employ rigorous methods to develop a reference model that is of practical value.

3.3 Research Design

3.3.1 Overview of Research Design

Edmonson and McManus [34] outline the key components of field studies: a research question, analysis of prior work, a research design and a contribution to the research.

The research design for this thesis largely follows the approach proposed by Ahlemann et al. [4] for the development of empirically grounded process reference models. The approach is derived from previous research results and has been validated in three previous process model construction projects. The design of Ahlemann et al.'s research approach consists of four phases: problem definition, construction of frame of reference, core construction and validation. This sequence has been modified to meet the requirements of this research project. For example, the problem definition phase is not included here because the problem has already been defined in the literature review.

The model developed in this thesis is named SOAdapt. It was developed during a research process consisting of four phases. The first phase - Phase 1: SOAdapt-FoR Construction, consists an exploratory study with an initial evaluation process, followed by a frame of reference construction process. Phase 1 was based on data from the literature as well as a case study. This frame of reference is used to guide two industrial empirical studies conducted during the project. The research output from this phase was a frame of reference construct called SOAdapt-FoR.

The second phase - Phase 2: SOAdapt-V1 Construction, contains the first empirical study that was conducted (Industrial Empirical Study (1)) and an interim literature based

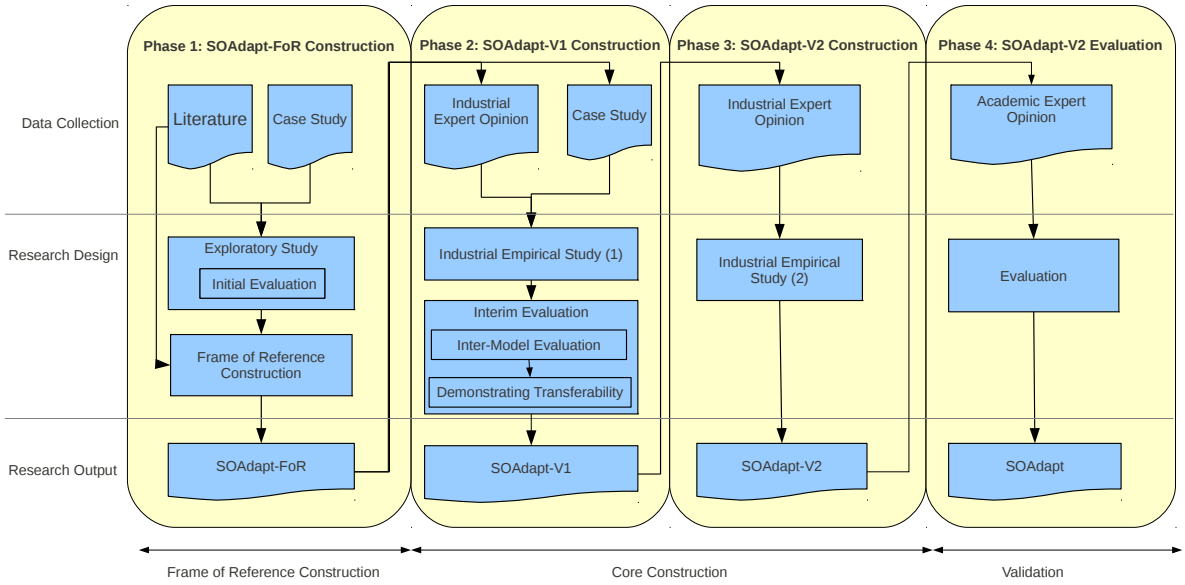

Figure 3.2: *Research Design*

evaluation resulting in SOAdapt-V1, the first complete version of the reference model. Phase 2 was based on industrial expert opinion and a case study.

The third phase - Phase 3: SOAdapt-V2 Construction, contains the second empirical study (Industrial Empirical Study (2)) resulting in SOAdapt-V2. This phase was based exclusively on data collected during an industrial expert opinion survey.

The final phase - Phase 4: SOAdapt-V2 Evaluation, is an evaluation process based on academic expert opinion resulting in, SOAdapt, the finished model. Figure 3.2 illustrates the phases that were used for the construction of the model. Phase 1 of the research can be mapped to the frame of reference construction phase of Ahlemann et al.'s process. Phase 2 and 3 can be mapped to the core construction phase of Ahlemann et al.'s process. Phase 4 can be mapped to the validation phase of Ahlemann et al.'s process.

3.3.2 Phase 1: SOAdapt-FoR Construction

The SOAdapt-FoR model was constructed by analysing existing publications that contain activities for adapting SBAs. Both peer-reviewed and non-peer-reviewed publications were analysed to compile these process details.

3.3.2.1 Systematic Literature Review

One of the key sources of adaptation activities was a Systematic Literature Review (SLR) of SBA development process models. The SLR, also reported in [65] (Appendix B), identified which of the process models supported SBA adaptation. The complete SLR in Appendix B provides a detailed account of the review protocol used to conduct the SLR. A total of 8 publications which contain processes and activities for adapting SBAs were identified in the SLR.

3.3.2.2 Exploratory Study

The author, in collaboration with two service engineering experts, examined 16 service-oriented engineering approaches for adaptation activities during an exploratory study. The exploratory study, which was carried out as a contribution to the S-Cube project, is published in [64]. The adaptation activities were validated during an evaluation process where each activity was mapped to a software process documented during a case study.

We found that 5 of these approaches supported adaptation in one way or another. Even if the approaches did not specifically support runtime adaptation they were still searched for activities that are implicit to adaptation. For example, in self-adapting systems, monitoring needs to occur in order to trigger adaptation. Although monitoring is not directly related to adaptation it is classified as an adaptation activity because it is needed by the application to adapt. Activities were classified as adaptation related if they could be related to the adaptation processes defined in the reference life-cycle proposed by the S-Cube consortium.

3.3.2.2.1 Initial Evaluation Additionally, the activities identified in the exploratory study were also mapped to the development process of a company who develop adaptable SBAs. The case is based on a company, SBA Solutions, that participated in Phase 3 of this project. The mapping carried out in this case study showed how the adaptation activities were able to enhance the companies software process. This demonstrated that the activities are applicable to the problem of developing adaptable SBAs.

3.3.2.3 Construction Process

Unfortunately, most of the approaches identified the in SLR [65] (Appendix B) and the networked search [64] (Appendix C) are not complete and lacking the required processes to fully guide SBA adaptation. Each of these publications as well as 2 other papers from Bucchiarone et al. [18][19] resulted in a total of 50 SBA adaptation activities. The activities are listed in Table 3.2 [63].

Within those 50 activities there was some overlap and duplication. Similar or duplicate activities were grouped together and assigned a generic activity name resulting in a smaller list of unique activities. The generic activity names are represented in the second column of the adaptation activities table (Table 3.2).

Source	Adaptation Activity	Generic Activity
Lane et al	Monitor message sequences amongst services and its partners	Runtime monitoring
Lane et al	Runtime service discovery	Adapt SBA
Lane et al	Requirements and analysis stage: define KPIs and management policies	Define Adaptation Requirements
Lane et al	Service specification: identify the service properties to specify	Define Adaptation Requirements
Lane et al	Specifying service decision model	Define Adaptation Strategies
Lane et al	Set warning thresholds and alerts for compliance failures	Define Monitoring Requirements
Lane et al	Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)	Define Monitoring Requirements
Lane et al	Service deployment: deploy the monitoring rules and recovery policies within the monitoring system	Deploy Monitoring Mechanisms
Lane et al	Gather QoS metrics on the basis of SLAs	Design Monitors
Lane et al	Evaluate SLA QoS metrics	Design Monitors
Lane et al	Readjust service weights for request queues	Design Monitors
Lane et al	Designing service adapters	Desing Adaptation Mechanisms
Lane et al	Recovery management: identify, by looking at the monitoring data, the needs for a recovery action	Event reasoning from Monitoring
Lane et al	Detect protocol violations	Runtime monitoring
Lane et al	Monitor service, application, middleware, OS, hardware, and network	Runtime monitoring
Lane et al	Monitor workloads	Runtime monitoring
Lane et al	Insert monitoring rules and recovery actions in concrete parts of the service composition executable description	Runtime monitoring
Lane et al	Monitor services	Runtime monitoring
Bucchairone et al	Define adaptation and monitoring requirements	Define Adaptation Requirements/Define Monitoring Requirements
Bucchairone et al	Design for monitoring and adaptation	Design Monitors/Design Adaptation Mechanisms
Bucchairone et al	Construction of monitors and adaptation mechanisms	Implement Monitoring and Adaptation Mechanisms
Bucchairone et al	Deployment-time adaptation	Define Adaptation Requirements
Bucchairone et al	Run-time Monitoring	Runtime monitoring
Bucchairone et al	Decide between adaptation and evolution	Define Adaptation Requirements
ProDAOSS	Organisational services realization paths are documented by a dynamic service hypergraph	Define Adaptation Requirements
ProDAOSS	Organisational services are designed as service centers in the architectural design discipline	Define Adaptation Requirements
ProDAOSS	Services realization environment is open and adaptable through the use of a reinforcement learning algorithm and a probabilistic reputation model	Event reasoning from Monitoring
PLASTIC	Runtime Analyser	Runtime monitoring
PLASTIC	SLA monitor	Runtime monitoring
PLASTIC	On-line validation	Runtime monitoring
PLASTIC	Evolution policies decoder	Event reasoning from Monitoring
BCDF	Take enterprises business and technical requirements as well as dependencies between them into consideration	Define Adaptation Requirements
BCDF	Enterprizes describe their purpose and high level requirements for a business collaboration	Define Adaptation Requirements
BCDF	Define the operational conditions under which businesses can cooperate	Define Adaptation Requirements
BCDF	Negotiation agreement describing the interactions among the services from the different parties	Desing Adaptation Mechanisms
Chang	Defining Target Services	Define Adaptation Requirements
Chang	Defining Unit Services	Define Adaptation Requirements
Chang	Planning service component aquisition	Desing Adaptation Mechanisms
Chang	Acquiring service components	Design Adaptation Mechanisms
Chang	Composing services	Adapt SBA
Multi-view SOAD	Develop use case models	Define Adaptation Requirements
Multi-view SOAD	Service identification by viewpoint	Define Adaptation Requirements
Multi-view SOAD	Service interface conception	Design Adaptation Mechanisms
Multi-view SOAD	Mapping to platform specific models and code generator	Event reasoning from Monitoring
CSOMA	Model services with variability points	Define Adaptation Requirements
CSOMA	Select orchestration schema based on the context of the incoming request	Event reasoning from Monitoring
CSOA	Design adaptation views for Platform Specific Models (PSMs)	Define Adaptation Requirements
Dino	Develop UML2 Modes Model	Define Adaptation Requirements
Dino	Generate requirements and capabilities of broker services	Define Adaptation Requirements
Dino	Set dynamic service broker configuration	Design Adaptation Mechanisms

Table 3.2: *Adaptation Activities*

The SOAdapt-FoR model was constructed using the Business Process Modeling Notation[1] (BPMN). The process constructs were limited to the constructs from the process meta-model defined in ISO/IEC 15288. BPMN was chosen over alternatives such as the Unified Modeling Language[2] (UML) since it is more expressive and understood by both technical and non-technical stakeholders. The ISO/IEC 15288 software process meta-model was chosen over alternatives such as the Software and Systems Process Engineering Meta-Model[3] (SPEM) because it has, in my opinion, fewer and more intuitive software process constructs.

Using UML class diagram notation, Figure 3.3 [46] illustrates the ISO/IEC 15288 process constructs and their interrelations. The figure uses UML classes and composition relationships to describe the meta-model. Classes are used to represent objects such as processes and activities. The composition relationships show how these objects relate to one another. Composition relationships are "has a" type relationships, for example, a process "has an" activity. Each end of the relationships have a small set of characters denoting the rules of the relationship. The characters allowed are "1", "0" or an asterisk. A "1" denotes that there is only allowed to be one instance of the object at that end of the relationship, a "0" means zero instances and an asterisk means multiple instances. The characters can be used in combination, for example, "1..*" mean one or multiple instances.

The figure shows that a Process can have zero or more Sub-Processes and one or more Activities. In turn, it shows that an Activity can have one or more tasks which can have zero or more Notes. It also shows that each process has a Name, a Purpose, and an Outcome. The meta-model will be slightly adapted for this research with the removal of the process outcomes and the addition of stakeholders and artifacts. Stakeholders are the individuals that participate in the activities of a process while artifacts are the physical entities such as code or documentation that are produced by or modified by activities.

The S-Cube life-cycle, illustrated in Figure 3.4 [5], was chosen as a starting point

[1] http://www.bpmn.org/
[2] www.uml.org
[3] http://www.omg.org/spec/SPEM/

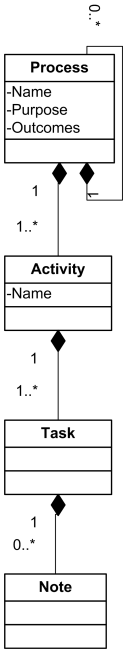

Figure 3.3: *ISO/IEC 15288 Process Constructs [46]*

to develop the SOAdapt-FoR model. The S-Cube life-cycle is a high-level life-cycle model skeleton proposed for the development of adaptable SBAs. The generic adaptation activities, that were identified in the SLR and Exploratory (Table 3.2), were mapped to the appropriate processes of the S-Cube life-cycle resulting in the SOAdapt-FoR model. This approach was taken in order to leverage the existing research which has been conducted by the S-Cube consortium. Additionally, the S-Cube life-cycle covers each aspect of the development cycle without being too detailed. This allows it to be easily modified and enhanced with activities reported in the service engineering literature. The SOAdapt-FoR model is a high-level model and only uses the Process and Activity constructs as defined in ISO/IEC 15288.

3.3.3 Phase 2: SOAdapt-V1 Construction

SOAdapt-V1 was constructed during Industrial Empirical Study (1) with data gathered from an industrial expert opinion survey and a case study. This data was then used to modify and extend SOAdapt-FoR. Therefore process attributes discovered during the interviews that were not present in SOAdapt-FoR represent the gaps in the literature. Altogether there were 198 passages of interview transcript which were assigned codes because

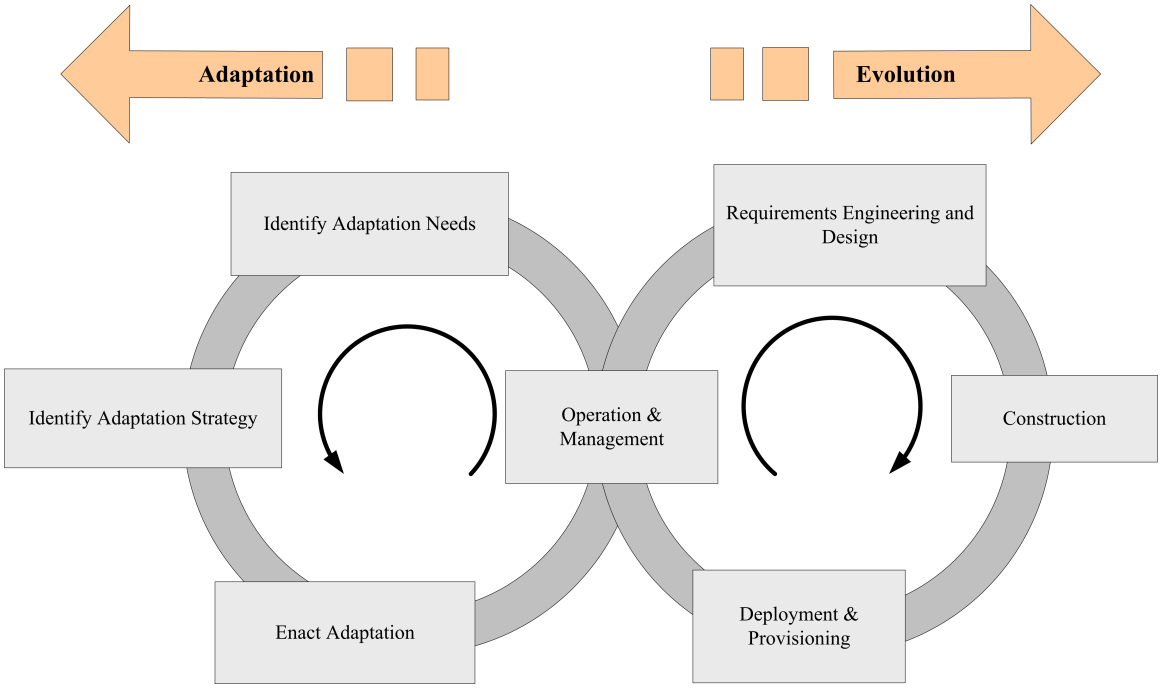

Figure 3.4: *The S-Cube Life-Cycle of Adaptable SBAs [5]*

they contained information about adaptation processes, activities, tasks, stakeholders, or artifacts. Additionally there were 12 passages of text which were coded because they contained information about the interview participants' backgrounds.

3.3.3.1 Industrial Expert Opinion Survey

The first stage in the expert-opinion survey was the development of an interview guide for use during the survey interviews. The interview guide was divided in two parts: the first part deals with the participants' backgrounds and their roles, while the second part focuses on the elicitation of adaptation activities. Participants discussed the activities within the SOAdapt-FoR model, and provided feedback. The respondents were asked whether or not they agreed with the activities contained in the SOAdapt-FoR model, and what changes they would make. Then they were asked to describe the tasks required to complete these activities, the stakeholders involved and the artifacts produced by the activities.

The interviews conducted were semi-structured interviews [99]. Interview guides for this type of interview contain open-ended questions which allow interviewers to ask follow on questions when necessary. This flexibility allowed the interviewer to ask the parti-

cipants detailed questions about their specific areas of expertise. This method allowed the model to be expanded in depth as well as breath, existing processes received much more detail while new processes were also added. Background questions were asked to provide context information for each of the participants. Each of the participants were asked back ground questions such as:

- What is you current position?
- Do you work with adaptable Service-Based Applications (SBAs)?

Then they were asked to identify adaptation process attributes that could be used to develop the SOAdapt-FoR further. For example, for each of its activities they were asked:

1. At what stage in the SBA development life-cycle should it occur, or should it occur at all?
2. What stakeholders should be involved in the activity?
3. What lower level activities or tasks should it contain?
4. What artifacts are required by or produced by the activity?

The purpose of these questions was to evaluate and refine where necessary the existing activities of SOAdapt-FoR. The second part of the interview, which sometimes came before these more formal questions, was about getting new adaptation specific process details based on their experience and opinion. For this part of the interview the participants were asked to describe the SBA development processes within their orgainisations, focusing specifically on there individual roles, where they could provide the most insight. These questions were used to elicit details of processes and activities that were not contained in SOAdapt-FoR and that could be used for its expansion.

A pilot interview was conducted using this interview format. This results of the pilot showed that this approach struck a good balance between flexibility and structure. A strictly structured questioning approach would have prevented the interviewer from probing concepts that were not considered while developing the SOAdapt-FoR model.

Participant	1	2	3	4
Role	Security R&D	Business Analyst	Sales/Product Development	Service Composition R&D, Commercialising Research
Experience with Adaptable SBAs	Yes	Yes	No	Yes
Location	USA	Mexico	UK	Germany
Company	A	B	C	D
Size	Very Large Multinational	Large	Large	Very Large Multinational
Sector	Hardware/Software Development	Banking	Mechanical Engineering	Communications
Participant	5	6	7	
Role	Performance Testing	Consultant	Business Process Improvement	
Experience with Adaptable SBAs	No	No	Yes	
Location	Germany	Netherlands	USA	
Company	E	F	G	
Size	Very Large Multinational	Very Large Multinational	Very Large Multinational	
Sector	Industrial,Healthcare	Software Development	Software Development/Services	

Table 3.3: *Survey Participants*

Once the interview guide was completed and piloted, the next step was to identify and contact suitable interview partners. This proved to be one of the most challenging aspects of the study given the specialised nature of the process model being constructed. In order to participate, interview partners needed to be experts in Service-Oriented Computing (SOC) and be able to provide opinion based on experience, on how best to adapt SBAs. Suitable practitioners were identified through user groups on the Linkedin[4] professional network as well as SOC conference proceedings. After sending out invitations to over 100 practitioners there were 15 replies with 7 eventually committing to interviews. At the end of each interview participants were asked to identify others who may participate with some initial positive responses. However, the initial interview partners did not provide leads for additional interviews indicating that chain sampling can be ineffective for such a specific research project topic.

Table 3.3 provides an anonymous profile of the SOC experts that participated in the survey. The company specific details are not shown to protect their identities.

Participant 1 outlined that his areas of expertise are in systems networking, communications, the security of large complex systems and more simple software applications. Over the past couple of years only he has gained experience in SOA-based business pro-

[4]http://www.linkedin.com

cess composition. The company that he works for sells hardware such as servers and personal computers while also providing professional software consultancy and development services.

Participant 2 was a business analyst in the banking sector. He worked with service-enabled financial applications which were often subject to adaptation during their operation. The bank that he worked for was one of the main banks in Mexico. Most of the applications that he supported were custom in house applications.

Participant 3, in his capacity as a software product development engineer, had experience in providing mechanical engineering software to clients. Some of this software is delivered to their clients through software services. He has done some preliminary research on integrating adaptation into the services provided by his company.

Participant 4 is an industrial researcher and is responsible for all the engineering research work within his company. His research focuses on service composition and he works within the company to turn research projects into products. The company he works for is a part of a group of vertically integrated companies that provide hardware and software solutions to the mobile telecommunications industry. The software that they provide is for the provisioning of mobile telecommunications.

Participant 5 is a test engineer working in the performance measurement area with a focus on queries performance, application overload and overload protection. He has experience with service-based applications in the health-care sector. The company that he works for is a major player in the health-care hardware sector. These hardware products are controlled by software thats developed within the company and many of the products are service enabled.

Participant 6 is a consultant within a middleware product division in his company. The middleware product that he works with facilitates service orchestration, service bus intercommunications and other technologies that tie applications together. The company that he works for also provides several other enterprise software applications.

Participant 7 focuses on business process management. His position involves engagement with his company's customers in order to improve their business services. He

Participant	8	9	10
Role	Chief Technolofy Officer	Business Anlayst	Developer
Experience with Adaptable SBAs	Yes	Yes	Yes
Location	UK	UK	UK
Company	H	H	H
Size	Medium	Medium	Medium
Sector	Software Development	Software Development	Software Development

Table 3.4: *Case Study Participants*

is involved with software adaptation research to meet the requirements of his customers' dynamic business processes. His company is a large hardware, software and software services supplier.

In advance of the interviews, the participants were sent a copy of the interview guide as well as some background information on the study, allowing them to familiarise themselves with the concepts and terminologies involved in the study. It also gave them extra time to think about the adaptation activities that they encountered while developing SBAs. The interviews were conducted either on site or through the Skype™Voice Over Internet Protocol (VOIP) service which allows the recording of interviews. Once all of the interviews were complete, they were transcribed in preparation for data analysis.

3.3.3.2 Case Study

A single case study was also conducted in order to supplement the survey conducted during this inquiry. The case study was carried out within a company with expertise in developing SBAs. The aim of this study [117] was to identify how a SBA development team approaches the adaptation of SBAs. The company where the case study was carried out will be referred to as SbaSoft.

Interviews were used in order to collect qualitative data during the case study. Three stakeholders involved in different aspects of the application development life-cycle were interviewed: The Chief Technology Officer (CTO), a business analyst and a developer. The interviews for the case study were conducted on site at SbaSoft. Table 3.4 provides an anonymous profile of the case study participants. Again, the company specific details are intentionally vague to protect its identity.

Participant 8 is the Chief Technology Officer (CTO) office within his company. He is responsible for high level technology related decisions and is involved in the management of each of the companies development projects. The company supplies one service-oriented enterprise application and sells solutions to its customers based around this product.

Participant 9 is a business analyst within the company. He works with their customers to capture their requirements and constraints. Once documented they are then used by the application designers, developers and testers.

Participant 10 is a software developer within the company and work directly on their product offerings. He has a lot of experience with the service-oriented aspects of their products.

The interviews for the case study were based on a semi-structured interviews similar to that used in Section 3.3.3.1. However, rather than focusing on adaptation specific activities, the questions were aimed at documenting SbaSoft's entire development life-cycle including adaptation specific activities. Participants in the case study were asked questions such as:

1. Does your company use service based applications, provide services for internal use or third parties: i.e. service client, service provider, or both.

2. Can you briefly outline how requirements are gathered for your services?

3. Can you briefly outline the process followed for designing web services in your company (i.e. people involved, activities carried out)?

4. Are your services designed to be adaptable?

Their entire development life-cycle was documented in order to view their adaptation specific activities in context. Analysed interview data provided input for the model.

Category	Process	Code
Activity	Requirements Engineering	activity!requirements engineering
	Design	activity!design
	Construction	activity!construction
	Deployment and Provisioning	activity!deployment and provisioning
	Operation and Management	activity!operation and management
	Adaptation Triggers	activity!adaptation triggers
	Adaptation Strategy	activity!adasequenceptation strategy
	Enact Adaptation	activity!enact adaptation
Artifacts	Requirements Engineering	artifacts!requirements engineering
	Design	artifacts!design
	Construction	artifacts!construction
	Deployment and Provisioning	artifacts!deployment and provisioning
	Operation and Management	artifacts!operation and management
	Adaptation Triggers	artifacts!adaptation triggers
	Adaptation Strategy	artifacts!adaptation strategy
	Enact Adaptation	artifacts!enact adaptation
Stakeholders	Requirements Engineering	stakeholders!requirements engineering
	Design	stakeholders!design
	Construction	stakeholders!construction
	Deployment and Provisioning	stakeholders!deployment and provisioning
	Operation and Management	stakeholders!operation and management
	Adaptation Triggers	stakeholders!adaptation triggers
	Adaptation Strategy	stakeholders!adaptation strategy
	Enact Adaptation	stakeholders!enact adaptation

Table 3.5: *Starting Codes*

3.3.3.3 Data Analysis

Once the data collection from the survey and case study was complete the interview transcripts were analysed using data analysis techniques as described in [77]. The raw data was first open coded with descriptive codes being added to segments of the data. A starting list of codes was based on the frame of reference model constructed in the first half of this study. Starting codes, as illustrated in Table 3.5, were created for activities, stakeholders and artifacts related to SOAdapt-FoR's high-level processes. New codes were added and existing ones modified to accurately reflect the raw data as the coding proceeded. Participant background information was coded in chunks containing information about their current roles which provides context for their interview responses. Background details of the participants, their companies and the software products that they work on are described in Section 3.3.5.

The coding was conducted with ulQda [39], a LaTeX[5] qualitative data analysis package. This package allows codes to be embedded within the raw data where they can be easily reviewed and edited in context. The package also allows the raw data to be typeset to a high quality document, and annotated with the appropriate codes. Finally, this package allows the codes and their corresponding raw data to be exported to a Comma Separated Value (CSV) file. CSV files are easily imported into spreadsheet applications for subsequent analysis.

During the coding process the interview transcripts were divided into passages with each passage being labeled to describe its process attributes. Some of the passages were conversational without any specific process details. Passages without process attributes or useful background information were not labeled. Once all of the passages were labeled they were all re-read to ensure that they were consistent with one another.

Once the open coding was complete the codes and their source data were imported into a spreadsheet. Here some of the codes that were assigned during the open coding process were further refined. Rather than being described by a single code, each passage was broken down into 5 categories of: processes, activities, tasks, stakeholders and artifacts. Processes, activities and tasks, in this context, relate the the process constructs defined in ISO/IEC 15288 where each process is made up of activities which are in turn made up of tasks. For example, the following passage was assigned the code "activity!operation and management!monitoring" during open coding:

> *I would monitor the state of the task, the external events, also the state of the information, and the documents and the things that are exchanged*

During the refinement process, a high-level process, an activity and 3 tasks were identified, with no stakeholders or artifacts being identified. The process constructs identified during this example are shown in Table 3.6.

In another example the following quotation provided the activity "Adaptation strategy based on system learning":

[5] http://www.latex-project.org/

Code	Text	Process	Activity	Task	Stakeholder	Artifact
activity!oam!monitoring	I would monitor the state of the task, the external events, also the state of the information, and the documents and the things that are exchanged	Operation and Management	Monitoring	Monitor processes		
		Operation and Management	Monitoring	Monitor events		
		Operation and Management	Monitoring	Monitor data		

Table 3.6: *Data Analysis Example*

> *..it would be great to have some type of learning system, in order to be able to learn from the jobs that have been run, failure modes and then to adapt the system according to what it has learnt, and any knowledge we have put in as to why the jobs might fail.*

Once these process attributes were identified they were organized into a conceptually clustered matrix [77]. Similar concepts, in this case the process attributes, were grouped together into a matrix with activities, stakeholders and artifacts assigned to the appropriate high-level life-cycle processes. They were then ordered into a logical process hierarchy making it easier to identify duplicate or disjointed attributes. Duplicate attributes were eliminated, while disjointed attributes were re-evaluated to see where they might fit into other processes or activities. Finally, the SOAdapt-FoR model was updated to reflect the processes, activities, tasks, stakeholders and artifacts identified during the data analysis. As well as additions, SOAdapt-V1 excluded some of the original SOAdapt-FoR constructs to reflect the findings of the empirical inquiry. All of the stakeholders and artifacts from SOAdapt-V1 have come from the empirical inquiry since there were none identified in SOAdapt-FoR. A graphical representation of SOAdapt-V1 was constructed using the Business Process Modeling Notation (BPMN) notation, showing the processes and activities. This approach was chosen because there were 38 stakeholders, 35 tasks and 17 artifacts identified which would clutter a complete BPMN representation. Tables were constructed to show the complete set of process attributes for each of the processes in the process model.

3.3.4 Interim Evaluation

SOAdapt-V1 was also compared to a related process model proposed by Oreizy et al. [85][84] for component-based software adaptation in an **inter-model evaluation**. This was to show that the process model is capable of adaptation. Oreizy et al.'s process model was validated through a worked example where it is used to adapt a cargo routing application. The objective of this comparison is to show that SOAdapt-V1 is suitable for runtime adaptation in general but not necessarily component-based adaptation. Therefore, the comparison will be at a high-level of granularity without emphasizing the lower-level component or service-based entities.

To demonstrate the **transferability** of the process model to a service-oriented environment, an evaluation was conducted in order to determine if it is compatible with a SBA development life-cycle proposed by Durvasula et al. [32]. Durvasula et al.'s life-cycle was collaboratively developed by 10 SBA development practitioners from 8 different companies. During the evaluation for this phase of the research, each of the activities from the adaptation process model was mapped to life-cycle processes within Durvasula et al.'s life-cycle. Mapping the activities to the life-cycle without conflict, shows that the process model is suitable for use with an empirically based SBA development life-cycle.

3.3.5 Phase 3: SOAdapt-V2 Construction

Industrial Empirical Study (2) was carried out to enhance SOAdapt-V1 with more detail, specifically within the adaptation cycle. In order to get more data for the adaptation cycle the original interview format was used. However, this time the interviewer focused more specifically on the questions relating to the adaptation cycle. As with Phase 2, the data collection method was an industrial expert opinion survey. There were 10 experts interviewed during this iteration of the research project. The participants, none of whom participated in Phase 2, have their profiles illustrated in Table 3.7.

Participant 1 is a lead architect in a large Irish software development company. In his role he provides guidance on the architecture and design of their service-oriented

Participant	1	2	3	4	5
Role	Lead Architect	Software Engineer	Service Engineer	Project Manager	Solutions Architect
Experience with Adaptable SBAs	Yes	Yes	Yes	Yes	Yes
Location	Ireland	Ireland	Ireland	Ireland	Ireland
Company	I	I	I	J	J
Size	Large	Large	Large	Very Large	Very Large
Sector	Software Development	Software Development	Software Development	Hardware/Software Development	Hardware/Software Development
Participant	6	7	8	9	10
Role	Requirements Engineer	SOA Researcher	Security Engineer	Research Manager	Researcher
Experience with Adaptable SBAs	Yes	Yes	Yes	Yes	Yes
Location	Ireland	UK	Argentina	India	Australia
Company	J	K	L	M	N
Size	Very Large	Very Large	Very Large	Very Large	Very Large
Sector	Hardware/Software Development	Telecommunications	Hardware	Software Development	Software Development

Table 3.7: *Survey Participants for SOAdapt-V2*

products. The company provides service-oriented enterprise applications to the public sector from their international operating locations. The company has three core product offerings all based around the same application framework.

Participant 2 is a software development engineer within the same company as Participant 1. He is involved in the design, coding and testing of their service-oriented products.

Participant 3 is a service engineer within the same company. He is responsible for service requests and deals with operational issues that come through the company's service desk.

Participant 4 is a project manager for the Irish devision of a multinational software services company. They provide professional software services for large client companies. Participant 4 is responsible for the project management of ongoing software projects with these clients. The company does not offer software products but work as a systems integrator as well as providing custom software development. Many of the solutions that the work on are distributed and rely on service-orientation to facilitate co-operation between systems.

Participant 5 is a solution manager in the same company as Participant 4. He is involved in the requirements engineering and design processes within the company. He

makes high level architectural decisions and provides design guidance on the company's projects.

Participant 6 is a requirements engineer within the same company. Her role involves working with customers to capture their requirements so that they can be used to design and develop the customers' solutions.

Participant 7 is a SOC researcher in a large British telecommunications company. His research involves the investigation of technologies that can be used to deliver adaptable SBAs. His company develops some of the software that is used to deliver their telecommunications services. Certain aspects of their provisioning software is service-oriented to allow distributed communications over their network.

Participant 8 is both a security architect and part of his company's product management team. One of his primary responsibilities is the management of a distributed service-oriented content management system. The company used this software platform to deliver software and content to specific hardware devices that are also provided by the company.

Participant 9 manages 80 software engineering researchers within a very large software services company based in India. One of the competencies of the research group is service-oriented computing. The group had published extensively and has several software patents in the area. The wider company delivers software services to companies in almost every sector. The profile of solutions delivered to their customers are usually large, often service-oriented, enterprise type applications.

Participant 10 is part of a distributed team of researchers that works in a large multinational software company. They work in collaboration with both the technical experts within the company as well as university researchers. SOC is one of the research groups competencies. The companies main business is the provision of enterprise management software to large organizations. Many of their products leverage service-orientation for the management of software and hardware within organisations.

3.3.5.1 Data Collection

In order to ensure that more adaptation related data was collected in the second round of interviews it was explained to the participants that there was a lot of data already collected for the evolution cycle. They were asked to briefly comment on the evolution cycle and to give as much insight as possible into how the adaptation processes should occur.

Lessons learned from Industrial Empirical Study (1) greatly helped with the elicitation of participants for the second round of interviews. In the first round of interviews industrial participants from service-oriented computing conferences had the highest response rate to interview invitations. Therefore, in the second round of interviews, industrial conference participants were also sought to participate. The second round of interviews saw participants from various domains and 6 different companies in total.

During the interviews, the same procedure as the first study applied. Each of the participants were sent an information sheet and the interview questions in advance of the interviews so that they could prepare and think about their answers. The interviews themselves were recorded and later transcribed.

3.3.5.2 Data Analysis

Data analysis was the same as for the Phase 2 interviews, described in Section 3.3.3.3. The data was then sorted by process and analysed for process attributes such as activities, tasks, stakeholders and artifacts. These process attributes were then used to update SOAdapt-V1 to give SOAdapt-V2. The primary objective of the second round of interviews was to to elicit process attributes for the the adaptation cycle. However, data was also elicited for the evolution cycle.

As the data from the second round of interviews was being analysed it was becoming clear that the model would need to facilitate process variability to deal with the different type of adaptable SBA development methods encountered in industry. An alternative to dealing with variability would be to keep the model at a high level of abstraction and then customise it for specific types of adaptable SBA development projects. The former option

was chosen because a process reference model that can be directly applied to different types of adaptable SBA development projects is more useful than a high level-model that is too abstract for any specific problem.

There were three distinct type of adaptation identified in Phase 2, static adaptation, dynamic manual-adaptation and dynamic self-adaptation. Each of these types of adaptation had their own associated process attributes and would need to be handled separately. As process attributes were identified during the second phase of the analysis they were analysed to see if they were related to generic adaptation or the three previously mentioned types. For example, in the "Identify adaptation triggers" process there were four activities identified:

1 Application adapts based on built in adaptation logical

2 Application dynamically adapts based on execution context

3 Application dynamically adapted by application user

4 Notify relevant stakeholders

After analysis it was determined that the "Notify relevant stakeholders" activity was the only generic activity that could be implemented during each type of adaptation. The other three were assigned to either static adaptation, dynamic manual-adaptation or dynamic self-adaptation.

Once each of the activities were assigned to an adaptation type, they were constructed in to a BPMN process diagram which reflected the different process flows for each of the adaptation types. In some processes, such as the deployment process, each of the activities were generic so there were no adaptation specific paths.

The high-level processes and activities from SOAdapt-V2 are represented in BPMN. Additionally, many of the activities from SOAdapt-V1 have been converted to sub-processes in SOAdapt-V2. According to ISO/IEC 15288 [46], activities can be converted to processes or sub-processes if they need to be specified in more detail. The new sub-processes

are also represented separately in BPMN diagrams. Finally, a set of process attribute tables have been created for each process and sub-process.

3.3.6 Phase 4: SOAdapt-V2 Evaluation

During Phase 4, an academic expert opinion survey was chosen for the evaluation of SOAdapt-V2 in order to perform a comprehensive evaluation of the model's usefulness. This evaluation method draws on the wealth of experience provided by the participating experts. Expert panels have been used extensively as a method for validating software engineering research [11]. Proponents claim that expert opinion is a reliable and accurate method of evaluation [66, 55]. For the evaluation, experts were drawn from a population with expertise in the various process areas of SOAdapt. We designate participants as experts if they have published widely in their area of expertise or have extensive professional experience in the area.

3.3.6.1 Expert Panel Survey

The experts that participated in the evaluation are members of S-Cube, the "European Network of Excellence in Software Services and Systems", of which the author is also a member. Each of the members have led research projects and published widely in their areas of expertise. Table 3.8 shows an anonymous profile of each of the participants that took part in the expert evaluation panel.

Participant 1 is one of the leading researchers in the area of requirements engineering and has also published in the area of requirements engineering for SOC. He has over 160 peer reviewed publications.

Participant 2 is a leading software engineering researcher and with particular interest in SOC and Web Services. He has over 20 publications in the area of SOC and more than 130 peer reviewed publications in total.

Participant 3 is a leading researcher in the areas of software architectures and middleware, and software process. She has more than 70 peer reviewed publications in her

Participant	1	2	3	4	5
Areas of Expertise	Requirements Engineering	Construction, Testing, Deployment	Design, Construction, Testing	Requirements Engineering	Adaptation
Location	UK	Italy	Italy	Ireland	Italy
Participant	6	7	8	9	10
Areas of Expertise	Operation and Management	Design	Adaptation	Deployment, Operation and Management	Design
Location	Spain	Germany	Germany	Spain	Netherlands

Table 3.8: *Expert Panel Participants*

areas of research.

Participant 4 is a researcher and software engineering lecturer with over 30 peer reviewed publications in the area of requirements engineering. He has previous industrial experience in the areas of SOC and software adaptation.

Participant 5 is a software engineering researcher with interests in a wide variety of software engineering research areas including SOC. He has over 230 peer reviewed publications and has been ranked among the top 50 software engineering researchers by the Association for Computing Machinery (ACM).

Participant 6 is a software engineering researcher with interests in logic programming, functional programming and SOC, particularly service composition and QoS. He has over 70 peer reviewed publication across his interest areas.

Participant 7 is a researcher in a prominent software engineering research center. His research interests include SBAs, adaptive systems and runtime verification. He has 8 peer-reviewed papers in the area of adaptive SBAs.

Participant 8 is a researcher within the same research center as Participant 7. His research interests are SBAs, adaptation and service testing. He has 4 peer reviewed publications in the area of adaptive SBAs.

Participant 9 is a researcher in the same research group as Participant 6. His primary research area is on the QoS of adaptive SBAs. He has 11 peer-reviewed publications in this area.

Participant 10 is a researcher in a dutch software engineering research center. His interests include business process modeling and the interactions between SBAs and business processes. He has over 20 peer-reviewed papers in the area.

Members of the panel were selected so that their primary area of expertise matched at least one of the process areas from SOAdapt. This will allow them to give in depth feedback about that particular aspect of the model.

The survey was administered as interviews that were administered over the phone or the Skype™Voice Over Internet Protocol (VOIP) service. These interviews were based on a questionnaire that contained open ended questions about SOAdapt-V2. Each of the interviews took approximately 25 minutes and were recorded for analysis.

3.3.6.2 Questionnaire Design

The aim of the survey was to determine if the process attributes from SOAdapt are useful, i.e. that they are capable of adapting SBAs and can be applied to a variety of contexts. In order to do this, the questionnaire was designed to find evidence that each of the process attributes has been used or should be used in real project scenarios. There were two choices for the questionnaire type: open ended or closed questions. There are several types of closed questions, for example, the participant may chose an answer from a set of options or they may be asked to rate a statement or question on a scale. Open ended questions, on the other hand, do not have a particular answer format and allow the participant to articulate the answer any way they want. Closed questions restrict the participant to provide data based around the parameters set by the questionnaire designed. Open ended questions were chosen for this validation allowing participants to freely give their opinions on the questions asked.

Participants were asked indirect questions about the process attributes. The questions were designed to investigate the usefulness of the attributes by asking questions about their use in real scenarios. Questions such as “can you give me an example of where that activity has been used” were asked. The answers to these type of question can show the usefulness of an activity while grounding the confirmation in real scenarios.

There were two question types used in the questionnaire of which there were many variations. The first question type sought examples of the process attributes while the second question type sought justification for inclusion of the process attributes in the model. Here are examples of the Example/Justification type questions that were asked:

- Can you give an example of where you have defined non-functional adaptation requirements?
- How does modeling adaptive business workflow help the requirements engineering process?
- Why is it necessary to define events that trigger adaptation?
- What is the cost/benefit of defining the feasibility of adaptation?

It was not feasible to ask each participant each question due to the large volume of attributes in SOAdapt-V2. There were a total of 111 attributes in SOAdapt-V2. It was decided that each evaluation questions would be asked only twice to keep the total number of questions manageable. Each question asked twice resulted in a total of 222 questions to be asked. It was also decided that 20 questions per expert was as much as could be expected from each interview. Dividing 222 question by a maximum of 20 questions per expert indicated a requirement of around 10 experts.

Experts were matched with questions that were closely related to their areas of expertise. For example, the requirements engineering process had about 25 questions. The two requirements engineering experts that participated were only asked questions from this process. However, in processes with smaller numbers of questions, similar processes were combined until about 25 questions were contained in the group. For example, all of the processes in the adaptation cycle were combined and “Adaptation” experts were asked the questions for this “Adaptation” process group.

3.3.6.3 Data Analysis

The data that was collected during the interviews was transcribed for analysis. In total there were 97 pages of transcribed data from 10 interviews that lasted a total of 5 hours and 52 minutes. The data was then collated and sorted so that each of the questions were grouped together. The answers for each question were analysed to see if they showed that the process attribute was useful and necessary for adaptation. It was also noted whether or not the usefulness and necessity was demonstrated through the experts' experience or their opinion. If both experts demonstrated the usefulness and necessity and necessity of an attribute it was kept unchanged. Otherwise a judgment call was made depending on the strength of evidence and whether the evidence was based on experience or opinion. At the end of this Phase 4, SOAdapt, which is presented in Chapter 8 has been completed.

3.4 Conclusion

In this chapter I have introduced some of the philosophy of research perspectives and paradigms and indicated which one are suitable for the objectives of this thesis. In some cases apparent philosophical dichotomies, such as rigor and relevancy, were both adapted as both have qualities relevant to the project.

The second part of the chapter outlines the research design for the thesis. The design contains the detailed methodology for each phase of the research. Phase 1 - SOAdapt-FoR Construction, of the research design describes how SOAdapt-FoR was constructed from the data gathered in the literature review and the exploratory study. Phase 2 - SOAdapt-V1 Construction, of the research design outlines how SOAdapt-V1 was the result of an empirical study and literature based evaluation. Phase 3 - SOAdapt-V2 Construction, describes the construction of SOAdapt-V2 from a second empirical study. Finally, Phase 4 - SOAdapt Evaluation, describes a final expert panel based evaluation resulting in SOAdapt.

Chapter 4

Phase 1: SOAdapt-FoR Construction

4.1 Introduction

In this first phase of the research, data, from an exploratory study and the systematic literature review are used to construct the frame of reference, SOAdapt-FoR. The frame of reference is used to guide the two empirical studies that were conducted. It is used to provide structure to industrial expert opinion interviews and is used for the development of the structured interview questions.

The complete systematic literature review is published in [65] (Appendix B). During the exploratory study a set of activities for developing adaptable SBAs were identified from the software and service engineering literature. The activities may be involved directly in adaptation or they may be adaptation support activities. The latter, while not essential for adaptation, provide for support functions such as change management. These support activities add to the overall quality of adaptable SBAs.

4.2 Exploratory Study

The research presented in this section is the result of a collaboration between the author and two academic service engineering experts who identified adaptation activities from the service-engineering literature.

The complete study is presented in [64] (Appendix C), while I will present my contributions in this chapter. Summaries of the co-authors' contributions are provided where necessary with appropriate citations and references.

Table 4.1: *Adaptation Activity Categories from S-Cube Deliverables*

Activity Category	Description
Define adaptation requirements	Identify the aspects of the SBA model that are subject to change, and what the expected outcome of the adaptation process is.
Define requirements to the monitoring subject	In order to satisfy the adaptation requirements, this practice focuses on specifying what artifacts are expected to be monitored.
Define monitored property	Specify which properties of the monitoring subject should be monitored.
Provide monitoring functionality	Monitoring functionalities that satisfy the monitoring requirements are provided through monitoring realization mechanism.
Collect monitoring results for adaptation	Results of monitoring are collected and analyzed.
Trigger adaptation	Evaluate the results from the monitoring analysis against adaptation requirements. If the need for adaptation is identified, send a request to trigger adaptation process.
Design adaptation strategy	Design the ways through which the adaptation requirements are satisfied.
Select adaptation strategy	Decide which particular adaptation strategy to be chosen based on the specific adaptation needs.
Perform adaptation	The actual adaptation process is performed through adaptation realization mechanisms based on the selected adaption strategy.

4.2.1 Adaptation Activity Categories

Several adaptation activity categories were identified in [64]. These activity categories were identified from state of the art research reports published by the S-Cube Consortium [5] [12] [43] and are illustrated in Table 4.1. The first column of Table 4.1 are the activity catagories while the second column provides a brief description of each category.

The co-authors of [64] studied 16 service engineering approaches and identified 5 [108][32][91][8][9] that contained activities suitable for adaptation. The activities identified are shown in Table 4.2. They are grouped together into activity categories based on the categories identified in Table 4.2.

4.2.2 Adaptation Support Activities from Maintenance Process Models

Examining software maintenance processes, I identified activities which may be useful for supporting the adaptation of SBAs. There are many software maintenance processes, definitions, models and standards encountered in the literature [74] [92] [102][115] [47][49].

Table 4.2: *Adaptation Activities from Service-Oriented Engineering Approaches*

S-Cube Category	Source	Adaptation Activity
Define adaptation requirements		
Define requirements to the monitoring subject	SDLC	Set warning thresholds and alerts for compliance failures
	SDLC	Gather Quality of Service (QoS) metrics on the basis of Service Level Agreements (SLAs)
Define monitored property	SeCSE	Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
Provide monitoring functionality	ASTRO	Monitor message sequences amongst services and its partners
	BEA	Monitor service, application, middleware, OS, hardware, and network
	SDLC	Monitor workloads
	SeCSE	Monitor services
Collect monitoring results for adaptation	ASTRO	Detect protocol violations
	SDLC	Evaluate SLA QoS metrics
Trigger adaptation	SeCSE	Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Design adaptation strategy	Chang's	Specifying Service Decision Model
	Chang's	Designing Service Adapters
Select adaptation strategy		
Perform adaptation	SDLC	Readjust service weights for request queues
	SeCSE	Runtime Service Discovery
Evolution Activities	BEA	Requirements and analysis stage - define Key Performance Indicators (KPIs) and management policies
	SeCSE	Requirements and analysis stage - identify the service properties to specify
	SeCSE	Service deployment - insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description
	SeCSE	Service deployment - deploy the monitoring rules and recovery policies within the monitoring system

However, ISO/IEC 14764 was the only source that contained detailed activities.

In total there are 19 maintenance practices identified from ISO/IEC 14764. They are categorised in Table 4.3 according to the processes they come from in ISO/IEC 14764.

4.2.3 Categorisation of Adaptation Activities

Table 4.4 shows each of the adaptation activities that were identified in the Table 4.2 mapped to the appropriate phases of the S-Cube life-cycle. For example, the activities

Table 4.3: *Adaptation Support Activities from ISO/IEC 14764*

S-Cube Category	ISO/IEC 14764 Practice
Define adaptation requirements	Problem and Modification Analysis : Modification Request/Problem Report (MR/PR) analysis Migration: Migration
Define requirements to the monitoring subject	Modification Implementation: Analysis
Define monitored property	
Provide monitoring functionality	Process Implementation: MR/PR procedures
Collect monitoring results for adaptation	Problem and Modification Analysis : Verification Maintenance Review/Acceptance: Reviews Migration: Post-operation review Maintenance Review/Acceptance: Approval
Trigger adaptation	
Design adaptation strategy	Process Implementation: Maintenance plans and procedures Problem and Modification Analysis: Options Migration: Migration plan
Select adaptation strategy	Problem and Modification Analysis : Approval
Perform adaptation	Process Implementation: Configuration management Modification Implementation: Development process

"Define KPIs and management policies" and "Identify the service properties to specify" were both mapped to the "Requirements Engineering and Design" process. This mapping brings together all of the adaptation activities from the approaches that were examined. Where possible, the activities are grouped in related categories. The categories used are those identified from the S-Cube deliverables in Section 4.2.1. For example, the activities "Detect protocol violations" and "Evaluate SLA QoS metrics" are contained within the "Collect monitoring results for adaptation" category from the "Identify adaptation needs" process.

The **Astro** toolset contains a monitoring tool which facilitates the two adaptation activities: "Monitor message sequences amongst services and its partners" and "Detect protocol violations". The activity "Monitor message sequences amongst services and its partners" monitors messages exchanged between services and service consumers which could be used as an adaptation trigger. "Detect protocol violations" monitors whether service consumers behave as expected, if they do not, the monitoring activity could also trigger adaptation.

Table 4.4: *Adaptation Activities Mapped to S-Cube Life-Cycle*

Life-Cycle Process	Source	Activity
Requirements engineering and design	BEA	Define KPIs and management policies
	SeCSE	Identify the service properties to specify
Construction		
Deployment and provisioning	SeCSE	Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description
	SeCSE	Deploy the monitoring rules and recovery policies within the monitoring system
Operation and management		
Identify adaptation needs		**Define adaptation requirements**
		Define requirements to the monitoring subject
	SDLC	Set warning thresholds and alerts for compliance failures
	SDLC	Gather QoS metrics on the basis of SLAs
		Define monitored property
	SeCSE	Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
		Provide monitoring functionality
	ASTRO	Monitor message sequences amongst services and its partners
	BEA	Monitor service, application, middleware, OS, hardware, and network
	SDLC	Monitor workloads
	SeCSE	Monitor services
		Collect monitoring results for adaptation
	ASTRO	Detect protocol violations
	SDLC	Evaluate SLA QoS metrics
		Trigger adaptation
	SeCSE	Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Select adaptation strategy		**Design adaptation strategy**
	Chang's	Specifying Service Decision Model
	Chang's	Designing Service Adapters
		Select adaptation strategy
Enact adaptation		**Perform adaptation**
	SDLC	Readjust service weights for request queues
	SeCSE	Runtime Service Discovery

The **BEA** life cycle also contains monitoring related activities that can trigger adaptation, the "Define KPIs and management policies" activity could be used to determine which properties should be monitored, while "Monitor service, application, middleware, OS, hardware, and network" describes the monitoring of services and other system components.

Chang's approach contains two adaptation related activities: "Specifying service decision model" aims at specifying the variability between available services and expected services. "Designing service adapters" aims at bridging the variability between service providers and consumers by allowing services to be dynamically adapted.

SDLC defines five adaptation related activities which revolve around the monitoring of quality attributes and alerting system users when they have exceed predefined SLAs: "Gather QoS metrics on the basis of SLAs (Service Level Agreements)" refers to the collection of quality attribute data for monitoring, "Set warning thresholds and alerts for compliance failures" refers to the setting of threshold values for the monitored quality attributes, "Monitor workloads" refers to the monitoring of system utilisation, if utilisation is high and response times are affected then the service provider may have to take the appropriate actions to ensure SLAs are met. "Readjust service weights for request queues" refers to the re-evaluation of SLAs if they are not being met due to high demand or utilisation. "Evaluate SLA QoS metrics" involves the comparison of QoS metrics to predefined SLAs.

The **SeCSE** approach contains many detailed activities relating to monitoring ("Monitor services", "Specify monitoring rules according to the adopted SeCSE monitoring language") and runtime adaptation ("Runtime Service Discovery") of SBAs. It contains two activities which support corrective adaptation: "Service deployment: insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description" refers to the implementation of monitoring mechanisms, while "Recovery management: identify, by looking at the monitoring data, the needs for a recovery action" refers to the runtime corrective adaptation of a SBA. "Service specification: identify the service properties to specify" states that the service properties to be monitored are determined during the service specification phase of development. Finally, "Service deployment: deploy the monitoring rules and recovery policies within the monitoring system" states that the appropriate monitoring mechanism is deployed during the deployment phase.

4.2.4 Observations on Adaptation Activities

Unfortunately, the adaptation activities identified from the service-oriented engineering approaches do not form a complete view of all the necessary activities required to enable the adaptation of SBAs. This is due to the fact that the activities were identified from

many sources that do not treat service adaptation as a primary concern. As we can see from Table 4.4, eleven activities were in the process: “Identify adaptation needs”, while only 4 activities were identified in “Identify adaptation strategy” and “Enact adaptation”. This implies that the state of art of adaptation processes focuses much more on gathering requirements and identifying when adaption is needed. These are highly relevant to what needs to be monitored. However, as soon as the need for adaption is identified, little effort has been put in to defining, selecting and executing adaptation strategies.

Only two SOA approaches, SDLC and SeCSE, explicitly describe the actual execution of adaptation. Indeed, in these cases, the adaptation is limited to corrective adaptation - the replacement of services when quality attributes do not meet expectations. Other types of adaptation such as perfective adaptation and preventive adaptation are not supported. Perfective adaptation aims at improving or optimizing the quality attributes of a SBA even it runs correctly. Preventive adaptation aims at preventing potential or possible future faults before they occur. None of the existing service-oriented engineering approaches specify how to select an adaptation strategy. In the two approaches that actually describe the execution of adaptation, the adaptation strategies are pre-defined.

As these activities were to the S-Cube life-cycle, the research demonstrated that some adaptation related activities actually belong to the evolution cycle. For instance, “Define KPIs and Management policies” (from BEA) as well as “Identify service properties to specify” (from the SeCSE methodology), while adaptation related, are defined at the requirement engineering process.

4.2.5 Categorisation of Adaptation Support Activities

Out of the 19 adaptation support activities identified from ISO/IEC 14764, 13 of them were mapped to the S-Cube life-cycle. Table 4.5 shows the cumulative mapping of the adaptation as well as the adaptation support activities to the S-Cube life-cycle.

Process Implementation The process implementation process area from ISO/IEC 14764 has three activities: “Maintenance plans and procedures”, “Problem reports/modi-

Table 4.5: *Adaptation activities from ISO/IEC 14764 and Service Engineering Literature Mapped to S-Cube Life-Cycle*

Life-Cycle Process	Source	Activity
Requirements engineering and design	BEA	Define KPIs and management policies
	SeCSE	Identify the service properties to specify
Construction		
Deployment and provisioning	SeCSE	Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description
	SeCSE	Deploy the monitoring rules and recovery policies within the monitoring system
Operation and management		
Identify adaptation needs		**Define adaptation requirements**
	14764	Problem and Modification Analysis : MR/PR analysis
	14764	Migration: Migration
		Define requirements to the monitoring subject
	14764	Modification Implementation: Analysis
	SDLC	Set warning thresholds and alerts for compliance failures
	SDLC	Gather QoS metrics on the basis of SLAs
		Define monitored property
	SeCSE	Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
		Provide monitoring functionality
	14764	Process Implementation: MR/PR procedures
	ASTRO	Monitor message sequences amongst services and its partners
	BEA	Monitor service, application, middleware, OS, hardware, and network
	SDLC	Monitor workloads
	SeCSE	Monitor services
		Collect monitoring results for adaptation
	14764	Problem and Modification Analysis : Verification
	14764	Maintenance Review/Acceptance: Reviews
	14764	Migration: Post-operation review
	14764	Maintenance Review/Acceptance: Approval
	ASTRO	Detect protocol violations
	SDLC	Evaluate SLA QoS metrics
		Trigger adaptation
	SeCSE	Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Select adaptation strategy		**Design adaptation strategy**
	14764	Process Implementation: Maintenance plans and procedures
	14764	Problem and Modification Analysis: Options
	14764	Migration: Migration plan
	Chang's	Specifying Service Decision Model
	Chang's	Designing Service Adapters
		Select adaptation strategy
	14764	Problem and Modification Analysis : Approval
Enact adaptation		**Perform adaptation**
	14764	Process Implementation: Configuration management
	14764	Modification Implementation: Development process
	SDLC	Readjust service weights for request queues
	SeCSE	Runtime Service Discovery

fication requests (MR/PR) procedures" and "Configuration management". Each of these activities were mapped to one of the high level adaptation activities of the S-Cube life-cycle from Phase I. I mapped the "Implementation of MR/PR procedures" to "Provide monitoring functionality" in the life-cycle. The implementation of problem report procedures would allow application engineers to receive and track problem reports which would allow them to determine if adaptation is necessary. Similarly, a modification request procedure would allow engineers to track modification requests and determine if the modification request requires adaptation. The "Maintenance plans and procedures" activity was mapped to "Define adaptation strategy" in the S-Cube life-cycle. The "Define adaptation strategy" activity refers to the definition of plans and procedures for adapting SBAs, so it makes sense that "Maintenance plans and procedures" could be used for this activity given the commonalities between adaptation and maintenance. "Configuration management" was mapped to the "Enact adaptation" activity, because the resolution of problems after applications adapt would be much easier if configuration details of services compositions are recorded. Fang et al. [36] illustrate how the configuration management process would be beneficial to the adaptation of SBAs.

Problem and Modification Analysis The problem and modification analysis process area contains four activities that are useful for SBA adaptation: "Problem reports/modification requests (MR/PR) analysis", "Verification", "Options" and "Approval". In the context of software maintenance these activities are undertaken in order to analyze problem reports or modification requests and determine their impact on the application ("MR/PR analysis"). If the reports or modification requests are valid ("Verification") potential solutions are proposed ("Options") and approval is sought to implement the required changes ("Approval"). The "MR/PR analysis" activity is mapped to "Define adaptation requirements" in the S-Cube life-cycle. The "Analysis of maintenance requests and problem reports" could be altered to the "Analysis of adaptation requests and problem reports" to suit the adaptation of SBAs. This analysis activity could provide valuable input which could be used to "Define adaptation requirements" for a SBA. "Verification" is mapped to the "Collect monitoring results for adaptation" activity because replicating or verifying

the problem can be seen as analysis of the monitoring results. "Options" is mapped to "Design adaptation strategy" because options for implementing the modification can be seen as adaptation strategy. Finally, "Approval" is mapped to "Select adaptation strategy" because obtaining approval is part of selecting an adaptation strategy in that the it finalises the decision on the selection.

Modification Implementation contains two activities "Analysis" and "Development" which are mapped to "Define adaptation requirements" and "Perform adaptation" respectively. "Analysis" is usually carried out before any "Development" or maintenance activity in order to determine which artifacts need to be modified. This may also be useful during the requirements gathering phase of SBA adaptation in order to determine which parts of the application need to be changed. In the context of traditional software engineering, "Development" means the modification of application code in order to implement requirements. This activity could be tailored to mean the modification of an applications configuration to meet the adaptation requirements of a SBA.

Maintenance Review/Acceptance The Maintenance Review/Acceptance process area contains two activities: "Reviews" and "Approval". In the context of software maintenance, reviews are carried out to ensure that the maintenance is carried out appropriately. In terms of adaptable SBAs, reviews can be carried out to ensure that adaptation occurs correctly. The analysis of collected monitoring results can be used to perform a review of SBAs which is why the "Reviews" activity was mapped to "Collect monitoring results for adaptation". Following a "Review", "Approval" status may be given to an adaptation engineer on satisfactory adaptation of an SBA. If adaptation occurs automatically it is impossible to grant approval to the work of an individual(s) so it may be appropriate to grant approval to the adapted application.

Migration In the context of traditional software engineering, migration is the modification of a system thus allowing it to run in a new environment or context. Rather than migrate a SBA, it may be possible for the application to adapt in order to operate in a new environment. Therefore, the migration process area may contain some useful activities that can help a SBA adapt to context specific parameters. The maintenance process area

has three activities that are useful to the adaptation of SBAs: "Migration", "Migration plan" and "Post-operation review". "Migration" was mapped to "Define adaptation requirements" because it is important to determine which software artifacts or which data should be migrated (or adapted) during the requirements gathering stage. "Migration plan" was mapped to "Design adaptation strategy" because a migration plan can be seen as an adaptation strategy in that it specifies what tools are needed, how to convert software product data and how to execute migration. Finally, "Post-operation review" was mapped to "Collect monitoring results for adaptation" because the impact of changing to the new environment can be achieved by monitoring.

Other Mappings Many of the adaptation support activity mappings are apparent, for example "Maintenance plans and procedures" to "Design adaptation strategy" or "Modification request/problem report procedures" to "Provide monitoring functionality". Some of the other mappings however are not so apparent, such as the "Maintenance review/acceptance" activity that maps to "Collect monitoring results for adaptation".

During the analysis I discovered that some of the maintenance activities are also relevant to the evolution cycle of the S-Cube life-cycle. However, those mappings were excluded as the focus of this thesis is adaptation. As previously mentioned five activities from ISO/IEC 14764 could not be mapped to adaptation activities because they are too specific to the software maintenance process (see Table 4.6): the "Documentation activity" and "Migration" activities. The "Documentation" activity from the maintenance process does not get included or is paid little attention to in any of the adaptation activities covered in the literature. The four migration activities mentioned in Table 4.6 are specific to the maintenance of traditional software and can not be leveraged for service adaptation.

4.2.6 Observations on Adaptation Support Activities

The maintenance activities identified in this section were never previously identified in the service engineering literature as candidate activities for supporting SBA adaptation. Many of the activities identified from the service engineering literature tend to deal with the

Table 4.6: *Maintenance Activities not Mapped*

Maintenance Practices
2 Problem and Modification Analysis
Documentation
5 Migration
Notification of intent
Implement operations and training
Notification of completion
Data archival

technical details of adaptation rather than focusing on support process details. A strengths of eliciting activities from a software process standard is that there is a process focus with process details such as inputs, tasks, controls, supports and outputs. The activities elicited from the service literature tend to specify what needs to be done in order to adapt SBAs while the maintenance activities identified can be tailored to specify how to support the implementation of adaptation processes.

The suitability of maintenance activities for supporting SBA adaptation highlights the commonalities between SBA adaptation and software maintenance. Both of these processes involve the modification of software systems albeit in different contexts. Adaptation is a light weight process which may only require the modification of simple configuration details to facilitate adaptation, so it is important not to include maintenance activities which would add unnecessary overhead to the process. The "Documentation" activity falls into this category, "Documentation" would add a lot of overhead to the process which is unnecessary due to Ad Hoc nature of SBA adaptation during runtime.

The activities identified from the maintenance literature are designed for the maintenance process which involves many manual activities, such as the analysis of problem reports and the development of proposed changes. However, the adaptation process may be a manual or automatic process. If the adaptation is manual, many of the maintenance activities can be applied directly without modification. However, if the adaptation

is automatic then many of the maintenance activities may become obsolete or require re-interpretation. For example, the "Analysis of problem reports" activity by definition is a manual activity carried out by a system maintainer, in the case of automatic adaptation it becomes obsolete as the application analyses problems through its monitoring mechanisms.

4.2.7 Initial Evaluation

In this section, I present a case study that illustrates how the activities identified during this exploratory study can aid in the development of adaptable SBAs. This documents how a software company currently develops adaptable SBAs, then through an example I will illustrate how their development process is improved upon with the results of this study.

4.2.7.1 Introduction to SBA Solutions

SBA Solutions is a large multinational software development company with several off shore development groups, sales and service centres. They develop software products that are used primarily in the public sector. Their offshore development teams are self-contained development units that have the skill sets required to undertake complete development projects. They have found this to be an effective development approach as many of the barriers to Global Software Development (GSD) are overcome by allowing offshore teams to work effectively.

Their main product makes extensive use of services to exchange data and integrate processes with different public sector departments. This is a key requirement of their customers as public sector data systems are often non-centralised and usually have many legacy systems.

SBA Solutions have embraced SOA as an enabling technology that can help them to create the solutions that their clients need. They understand that SOA is a double edged sword, along with its benefits, SOA is complex and the composition of services is a

complex and skilled activity. In order to get the maximum benefit from SOC they follow best practice development activities set out by IBM. They follow SOAD [7] architectural practices where SOA is developed as a tiered architecture. There is a component tier with software components from enterprise applications. Then, above this, there is a service tier where these components are exposed as services. Finally, there is a business tier where services are combined into business services which are in turn composed into business processes used in functional domains.

4.2.7.2 SBA Development Process

The development process within SBA Solutions is similar to the Rational Unified Process (RUP) [60] development life-cycle. It is a four phase linear life-cycle with phases of requirements definition, architecture and design, coding and testing, and stabilisation and release. The activities within the architecture and design, and coding and testing phases are iterative resulting in scrum type development sprints. This is a typical development process with product documentation being developed throughout the life-cycle. It is worth noting that testing is an integral part of each development phase with test planning being included in the initial requirements phase.

Key to the software process in SBA Solutions is their process model that breaks down each process into discrete tasks, each assigned to relevant stakeholders. This model gives each member of the development team clear roles and responsibilities and eliminates redundancy. Since they employ GSD and their development is carried out iteratively, development iterations can be carried out concurrently or staggered to increase the development speed. This is a complicated development process so it is crucial for them to manage the process model effectively.

4.2.7.3 SBA Adaptation Process

SBA Solutions implement runtime adaptation in an interesting way. They have an adaptation dashboard that is used at runtime to make changes to the way that the application executes. This allows the modification of business workflows and the addition or removal

of business rules to suit real time business requirements. Another adaptation feature of one of their key applications is that the execution path of the application is dynamically determined based on parameters that are passed to the application. This is a type of built in static adaptation that is implemented using a rule engine. These rules may also be subsequently adapted using the adaptation dashboard that has been previously mentioned.

The distinguishing feature between this type of adaptation and that employed by others is the way that it is enacted. Many of the adaptation approaches reported in the literature have adaptation processes that are carried out on production applications. In SBA Solutions, adaptation is first tried on a test system to determine if there are any unforeseen consequences. If after adaptation, the test system operates correctly, then the adaptation configuration is transferred to the production system. This provides the safety of testing adaptation configurations while still facilitating dynamic runtime adaptation. The basic adaptation process for SBA Solutions can be seen in Figure 4.1.

One of the downsides to how adaptation is enacted in the products of SBA Solutions is that, while they have a rigorous development process model, runtime adaptation processes are ad-hoc and vary depending on their customers requirements. While it is important to consider customer requirements for adaptation, it would also be beneficial to have a process model to guide adaptation. This model could be adapted to customers needs which would still allow flexibility in the process.

4.2.7.4 Improved Process

This section illustrates how the adaptation process at SBA Solutions can be enhanced by following the adaptation activities identified during this study. Table 4.7 illustrates SBA Solution's development and adaptation processes mapped to the adaptation activities previously shown in Table 4.5. The processes from SBA Solutions actual software process have replaced the sample processes from the S-Cube reference life-cycle.

4.2.7.4.1 Evolution **Requirements definition** Within the requirements definition process there is the single activity of "Define KPIs and management policies". This activity can be used

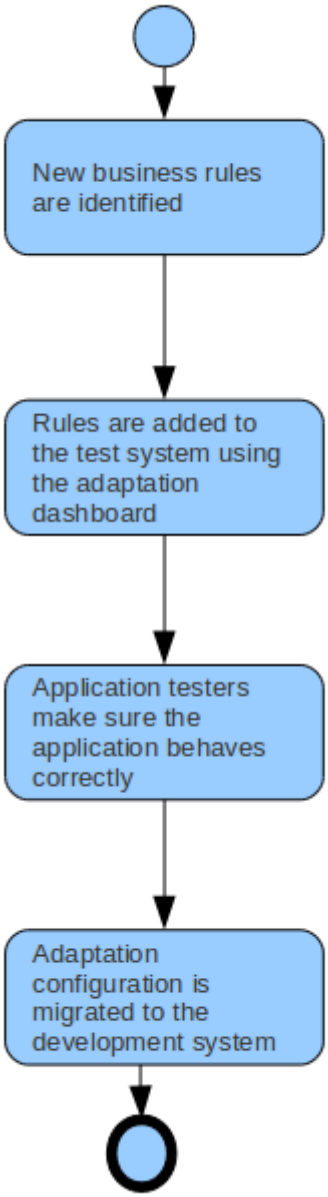

Figure 4.1: *SBA Solutions Adaptation Process.*

in SBA Solutions software process to specify SBA's Key Performance Indicators (KPIs) (non-functional adaptation requirements) such as service availability and performance. These are factors that can be decided upon during the initial development (evolution) of an SBA.

Architecture and design The architecture and design process has also one adaptation activity which is to "Identify the service properties to specify". This activity relates to the

Table 4.7: *Adaptation activities from ISO/IEC 14764 and Service Engineering Literature Mapped to SBA Solutions Life-Cycle*

Evolution	**Requirements Definition**	BEA: Define KPIs and management policies
	Architecture and Design	SeCSE: Identify the service properties to specify
	Coding and Testing	
	Stabilisation and Release	SeCSE: Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system
Adaptation	**New business rules are identified**	**Define adaptation requirements** 14764: Problem and Modification Analysis : MR/PR analysis 14764: Migration: Migration
		Define requirements to the monitoring subject 14764: Modification Implementation: Analysis SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs
		Define monitored property SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
	Application testers make sure the application behaves correctly	**Provide monitoring functionality** 14764: Process Implementation: MR/PR procedures ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services
		Collect monitoring results for adaptation 14764: Problem and Modification Analysis : Verification 14764: Maintenance Review/Acceptance: Reviews 14764: Migration: Post-operation review 14764: Maintenance Review/Acceptance: Approval ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics
		Trigger adaptation SeCSE: Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
	Rules are added to the test system using the adaptation dashboard	**Design adaptation strategy** 14764: Process Implementation: Maintenance plans and procedures 14764: Problem and Modification Analysis: Options 14764: Migration: Migration plan Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters
		Select adaptation strategy 14764: Problem and Modification Analysis : Approval
	Adaptation configuration is migrated to the development system	**Perform adaptation** 14764: Process Implementation: Configuration management 14767: Modification Implementation: Development process SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery

identification of service properties that need to be identified for monitoring which is in turn needed for adaptation. Within SBA Solutions most adaptation is related to business rules, so the service properties that might need to be identified are service functionality and functional specifications.

Coding and testing The activity "Insertion of monitoring rules and recovery actions

in concrete parts of the service composition executable description" describes how monitoring rules and recovery capabilities need to be added to the executable SBA so that they can be applied to adaptation scenarios during runtime. This activity is useful for SBA Solutions to guide the development of monitoring based adaptation in their applications.

Stabilisation and release The activity "Deploy the monitoring rules and recovery policies within the monitoring system" describes how monitoring rules are deployed separately from the SBA. This activity is necessary for SBA Solutions because each customer is supplied with a standard application which is updated with their specific configurations. In their case bare bones applications are updated with monitoring and adaptation configurations needed for out-of-box functionality.

4.2.7.4.2 Adaptation

New business rules are identified The first sub-process in SBA Solutions runtime adaptation is the identification of adaptation rules. This sub-process can be enhanced with several adaptation and adaptation support activities. The activity category "Define adaptation requirements" has two adaptation support activities that can be applied to SBA Soultions:

- The "Modification Request (MR) / Problem Report (PR) analysis" activity is used by SBA Solutions to manage requests for new business rules to be added to their adaptable SBAs. This activity provides a structured set of tasks for handling the business rules of SBA Solutions' customers as they are identified. This makes sure that they are valid and do not contain duplication.

- The "Migration" activity from IEEE 14764 can provide support tasks for when the SBA is adapting or migrating from one state to another. Migration activities related to planning and notifying users of changes would be useful for SBA solutions' application users.

The activity category "Define requirements to the monitoring subject" has one adaptation support activity and two adaptation activities that would be useful for SBA Solutions' "New business rules are identified" process:

- The "Analysis" activity provides useful guidance for the analysis of business rules before runtime adaptation is implemented. This activity could be for business rule validation so that illegal business rules are not added to the application causing it to adapt unsafely.

- The activity for "Setting warning thresholds and alerts for compliance failures" is useful for alerting stakeholders when the operating service-level agreements for SBA Solutions' customers' applications are being reached or approached.

- "Gathering QoS metrics on the basis of SLAs" is a useful activity for determining whether or not it is necessary to make an adaptation. This activity relates to non-functional adaptation. However, non-functional adaptation requirements also need to be considered when determining the impact if modifying business rules.

The final activities in the "Define monitored property" category is to "Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)". This activity is low-level as it specifies a particular monitoring language to be used for specifying monitoring rules. The activity is not useful for SBA Solutions given that they already have their own technical solutions for monitoring.

Rules are added to the test system using the adaptation dashboard In this adaptation sub-process business rules that were identified in the previous sub-process are added to the application. Execution workflows can also be changed using SBA Solutions adaptation dashboard. Our adaptation framework has three adaptation support activities and two core adaptation activities in the "Design adaptation strategy" category:

- The "Maintenance plans and procedures" activity from the IEEE 14764 process implementation activity group is necessary for implementing the adaptation processes and activities mentioned in this study. This activity is also useful for SBA Solutions' customers who want to implement the process formally.

- The "Options" activity from the IEEE 14764 problem and modification analysis activity group can be used by SBA Solutions' customers to evaluate the options for

adaptation business rules and workflows in their applications.

- The "Migration plan" activity from the IEEE 14764 migration activity group can be useful for the customers of SBA Solutions while they are planning adaptation.
- In the case of SBA Solutions' adaptable SBAs, the "Specifying service decision model" activity can be used when deciding on which services are going to be used to enact new business rules.
- The "Designing service adapters" activity can be used by SBA Solutions to make services adaptable when their products' existing services are not sufficient for an adaptation requirement.

In the final activity category in this process, "Select adaptation strategy" there is a single adaptation support activity. The "Approval" activity from the IEEE 14764 problem and modification analysis process group can be used by SBA Solutions to approve the selected adaptation configuration after the various options have been considered.

Application testers make sure application behaves correctly There are many adaptation activities, identified in this study, that can be used to assist this testing sub-process. I suggest that the monitoring related activities can help testers to collect data that can be used by them to determine the impact adaptations on the system. In the case of SBA Solutions, these monitoring and testing activities would take place on test systems rather than production systems.

To start with the "Provide monitoring functionality" category has four activities:

- The "Modification Request (MR) / Problem Report (PR) procedures" activity from the IEEE 14764 process implementation group involves the implementation of the Modification Request (MR) and Problem Report (PR) procedures. This is necessary so that the Modification Request (MR) and Problem Report (PR) analysis tasks can be carried out when new business rules are identified.
- The "Monitor message sequences amongst services and its partners" activity from the ASTRO composition tool set can be used to gather monitoring data that can be

used by testers to evaluate the impact of implementing new business rules in SBA Solution's applications.

- "Monitoring service, application, middleware, OS, hardware, and network" properties allow testers to evaluate the impact of adaptation on a wide variety of infrastructural characteristics.

- The activities "Monitor workloads" and "Monitor services" like the previous activity can be used to capture data that can be used by testers for diagnostic purposes.

There are four adaptation support and two adaptation activities in the "Collect monitoring results for adaptation" category:

- The "Verification" activity from the IEEE 14764 problem and modification analysis activity group can be used to support the verification of monitoring results. This activity may be useful for testing stakeholders in SBA Solutions.

- The "Reviews" activity from the IEEE 14764 maintenance review/acceptance activity group can be used by testers to review and verify the integrity of the test system after adaptation.

- The "Post-operation review" activity from the IEEE 14764 migration activity group should be conducted by testers on production systems to determine the impact of adaptation on live rather than test system. This post-operation review can be used to determine the "true" impact of adaptation by SBA solutions' testers.

- Once adaptation reviews have taken place it may be worthwhile for SBA Solutions to have an "Approval" activity where the introduction of new business rules are formally approved. This would close the loop in the business rules governance process and assign responsibility to the approvers to make sure the adaptations valid.

- The "Detect protocol violations" activity from the ASTRO toolset is useful for detecting protocol violations, using its monitoring capabilities, and storing the monit-

oring data. This monitoring data can be then used by system testers for diagnostic purposes.

- The "Evaluate SLA QoS metrics" activity from SLDC is another monitoring related activity that help to evaluate monitoring data to see if SLAs have been breached, this activity is useful for SBA Solutions' testers to check if modifying business rules or workflows cause application SLAs to be breached.

The final activity category for this sub-process, "Trigger adaptation", contains a single adaptation related activity: "Identify, by looking at the monitoring data, the needs for a recovery action". This activity again can be used by testers to trigger a recovery action if the application fails after new business rules or workflows are implemented.

Adaptation configuration is migrated to the test system there are two adaptation support and two core adaptation activities within the "Perform adaptation" category that can be used by SBA Solutions to help migrate adaptation configurations between test and production applications.

- The "Configuration management" activity from the IEEE 14764 process implementation activity group is a useful activity for SBA Solutions' customers. They can use it to keep track of their applications adaptation configurations. They would also be able to keep track of the business rules that get introduced into the application.

- The "Development process" activities from the IEEE 14764 modification implementation activity group can be used by SBA Solutions to implement adaptations when adaptation is implemented with adaptation scripts. The development of these adaptation scripts can be handled like small development projects.

- The "Readjust service weights for request queues" activity can be used by SBA Solutions' adaptable SBAs when their component services are unavailable or non-responsive. This can often be the case when services are provided by various distributed public sector departments.

- The "Runtime service discovery" activity is not particularly useful for SBA Solutions because their customers' services are usually listed in the application and discovery is not strictly necessary.

These mapping shows how SBA Solution's ad-hoc adaptation process can be greatly enhanced by adding more detailed adaptation as well as adaptation support activities. It may be the case that all activities are not needed depending on project requirements, but these redundant activities can simply be excluded from implementation.

4.3 Frame of Reference Construction

The frame of reference for SOAdapt (SOAdapt-FoR) was constructed using adaptation process attributes identified in a systematic literature review and the exploratory study described in the previous section. Ahlemann et al. describe a frame of reference model as the starting point for reference model development, with the purpose of facilitating navigation of the problem domain. This frame of reference was used to guide Industrial Empirical Studies (1) and (2) conducted in Phases 2 and 3 of this research. The study participants are then able to reason about that knowledge and improve it with their experience.

Ten high-level generic adaptation activities were distilled from a set of 50 activities that were identified in the systematic literature review and the exploratory study. There were also additional adaptation support activities identified in Section 4.2. These support activities will not be used in the construction of the frame of reference as it will focus on the activities core to adaptation.

During the systematic literature review, there were 722 studies identified in the initial search with 77 found to contain process details for developing SBAs. A total of 8 studies from the SLR were found to have supported adaptation. The adaptation activities from these 8 studies, the studies identified in the exploratory study (Section 4.2) and one other relevant study [18] are listed in Table A.19 in Appendix D.

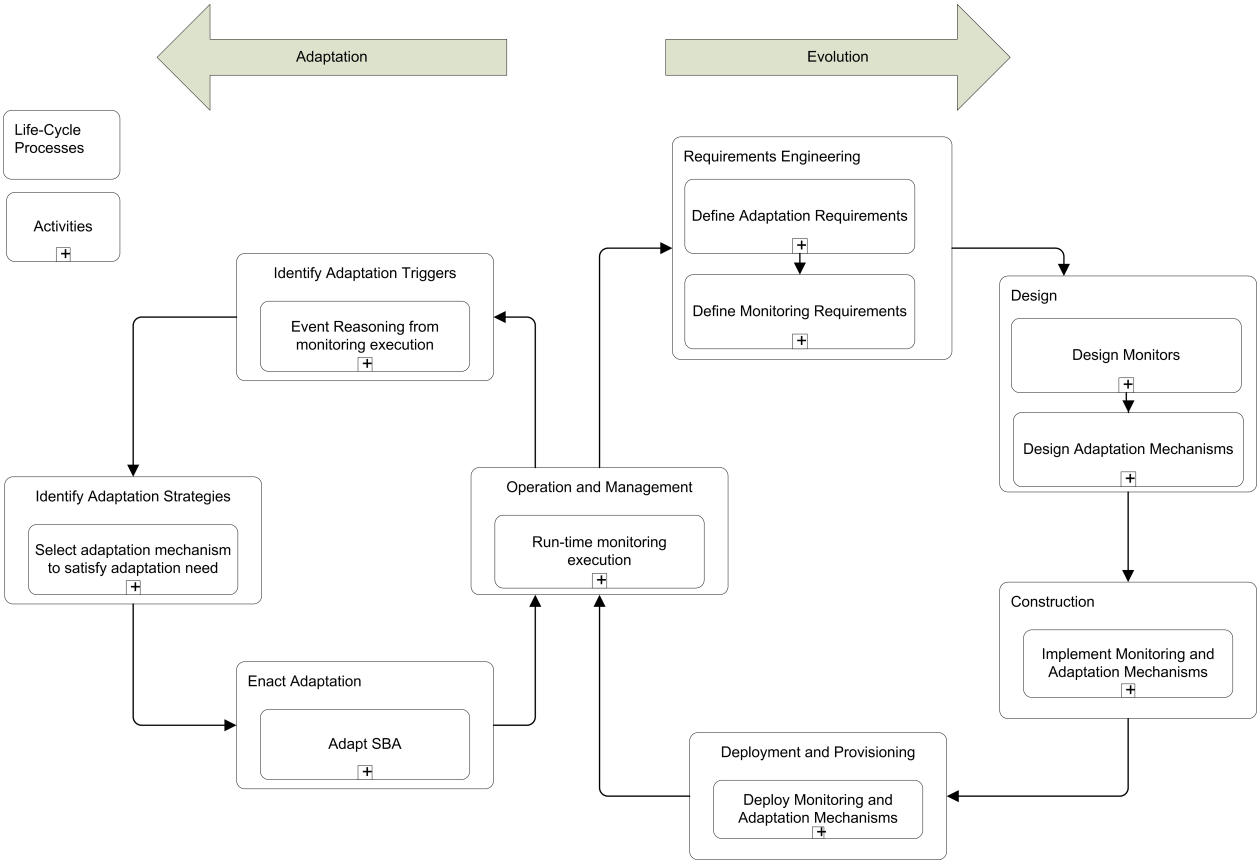

Figure 4.2: *SOAdapt-FoR Model*

The 10 generic activities (Table 4.8) and their parent processes from the S-Cube life-cycle were ordered into a sequential workflow. Figure 4.2 illustrates the SOAdapt-FoR model represented in BPMN.

Generic Activities
Adapt SBA
Define Adaptation Requirements
Define Adaptation Strategies
Define Monitoring Requirements
Deploy Monitoring Mechanisms
Design Monitors
Design Adaptation Mechanisms
Event reasoning from Monitoring
Implement Monitoring and Adaptation Mechanisms
Runtime monitoring

Table 4.8: *Generic Adaptation Activities*

The high level process such as “Requirements Engineering” and “Design” that make

up the skeleton of the model are primarily inherited from the S-Cube life cycle model with some exceptions. The "Requirements Engineering and Design" process inherited from the S-Cube life-cycle model was separated into two separate processes to accommodate the numerous activities identified for each process. The S-Cube life-cycle was chosen as a starting point because it is generic enough for practitioners to relate to while also including processes for adaptation.

The frontier between SOAdapt-FoR and its inclusion in existing life-cycle models depends on the type of model with which it is being integrated. In the evolution cycle, adaptation activities should occur before the other activities in the process have been completed. This is necessary so that other activities can see the impact of the activities from SOAdapt-FoR. For example, adaptation may require a particular service granularity and service hierarchy and these aspects need to be decided before the remainder of the application is developed. The activities from the adaptation cycle (left-hand side) are contained in a self-contained loop. Therefore, these processes may occur during the operation of the application regardless of the application's development life-cycle.

4.4 Conclusion

In Section 4.2, activities and support activities which should be considered when carrying out the adaptation of SBAs have been identified. This has been done through the identification of activities within service-oriented development approaches and within the software engineering maintenance process. The importance of this work is that the consolidation of existing adaption activities is enhanced with support activities from mature software engineering process standards. While not directly involved with adaptation, these support activities should support the development of adaptable SBAs. The result is a collection of adaptation activities that can be used to guide practitioners in the development of adaptable SBAs.

During the exploratory study, an industrial case study was used to show how the activities identified were able to enhance the process of a software company that specialises in

developing adaptable SBAs.

After the exploratory study and the systematic literature review, documented in Section 2.6, a list of adaptable SBA development activities were complied. These activities were then used to construct SOAdapt-FoR.

Chapter 5

Phase 2: SOAdapt-V1 Construction

5.1 Introduction

In this Chapter I will present the updates to SOAdapt-V1 based on the findings from the first empirical study - Industrial Empirical Study (1). I also describe the analysis of the 10 interview transcripts (275 minutes of audio) used to construct the SOAdapt-V1. Once the content analysis is complete, the SOAdapt-FoR model is updated to represent the study's findings. The source of each process attribute will be discussed with exemplar quotations provided as justification for inclusion in the model.

5.2 Industrial Empirical Study (1)

In the first subsection a description of the changes to the high-level parent processes will be described. This will be followed by a more detailed analysis of the changes within each of the process groups from SOAdapt-FoR. In this phase, the model is updated with low level attributes adding a lot more detail than SOAdapt-FoR.

I will present the reasoning behind the inclusion of each attribute with the inclusion of supporting quotations where appropriate. Reasoning will also be provided for cases where process attributes are not taken directly from the data but added to fill the gaps where they are logically needed. Activities, tasks and stakeholders will be shown in inverted commas when mentioned while artifacts will be shown in bold.

5.2.1 High-Level Processes

Interview participants agreed that the high-level processes from the SOAdapt-FoR model should remain largely unchanged and that adaptation activities should occur in the traditional requirements engineering, design, construction, deployment and provisioning and operation and management phases. However, the need for a separate testing process was identified. Another notable deviation from the original high-level processes was the lack of emphasis placed on the provisioning sub-process from the deployment and provisioning process.

The processes from the adaptation cycle of the SOAdapt-FoR model caused confusion. For example, participants did not understand the difference between terms such as strategies, mechanisms and their applications. These processes as well as their contained activities have now been modified to make them more intuitive. For example, the activity "Event reasoning from monitoring execution" was changed to the more straightforward "Event reasoning from monitoring".

Changes to the activities contained within the high-level processes were more extensive than changes the changes to their parent processes. This is to be expected given that the high-level life-cycle processes from the evolution cycle are almost de facto, featuring in prominent life-cycle models such as RUP, ISO/IEC 12207 and the waterfall model [100]. All of the changes made to the processes and activities from SOAdapt-FoR are listed in Table 5.1. The first column of the table shows whether the processes or activities were added or removed. In most of the cases activities were modified rather than being removed completely. This is shown in the table as a *Remove* followed by an *Add* of the modified version. Rows with both a *Process* and *Activity* entry refer to the activity and show the process for reference, while rows with just a process entry refer directly to the process.

In cases where there is no direct supporting data, changes were made to activities so that they correspond to the activities that have supporting data. For example, the modification of "Deploy Monitoring and Adaptation Mechanisms" to "Deploy Adapta-

Add/Remove	Process	Activity
Remove	Design	Design Adaptation Mechanisms
Add	Design	Design Adaptation Strategies
Remove	Construction	Implement Monitoring and Adaptation Mechanisms
Add	Construction	Implement Adaptation Strategies
Add	Construction	Implement Monitors
Remove	Provisioning	
Remove	Deployment	Deploy Monitoring and Adaptation Mechanisms
Add	Deployment	Deploy Adaptation Strategies and Monitors
Remove	Operation and Management	Run-time monitoring execution
Add	Operation and Management	Run-time monitoring
Remove	Identify Adaptation Triggers	Event Reasoning from monitoring execution
Add	Identify Adaptation Triggers	Event Reasoning from monitoring
Remove	Identify Adaptation Strategies	
Add	Select Adaptation Strategies	
Remove	Select Adaptation Strategies	Select adaptation mechanisms to satisfy adaptation need
Add	Select Adaptation Strategies	Select adaptation strategies to satisfy adaptation need
Remove	Enact Adaptation	Adapt SBA
Add	Enact Adaptation	Execute selected adaptation strategy
Add	Testing	

Table 5.1: *SOAdapt-V1 Process and Activity Changes*

tion Strategies and Monitors" did not have direct supporting evidence. This activity was changed in order to correspond with the activity "Design Adaptation Strategies" which did have supporting data. The "Provisioning" sub-process was removed from the "Deployment and Provisioning" process, not as a result of comments from the interviewees to remove it, but because when probed, they never offered any details on adaptation activities that need to occur during provisioning.

5.2.2 Requirements Engineering

Activities, Tasks and Artifacts

The interview participants agreed with the 2 Requirements Engineering (RE) activities from the SOAdapt-FoR model. There were 3 tasks identified for the "Define Adaptation Requirements" activity as a result of the data analysis. The first task in "Define adaptation requirements" is to "Define events that trigger adaptation". Several variations of this activity were identified by many of the interview participants, for example:

> ..knowledge that we have about why we would need to adapt, so at some level we would want to have some fuzzy logic, under the circumstances maybe you should contemplate, looking for the cheapest system, looking for a reliable system, so at some level there would be a service level agreement..

Several of the interview participants suggest that it is necessary to identify the events relating to application reliability, availability and quality during requirements engineering. The participants suggest that artifacts such as **use cases**, **workflows** and **architectural overviews** can be used to capture these adaptation requirements.

The second task identified in the "Define adaptation requirements" activity is to "Choose between manual and automatic adaptation". This task was not elicited directly from interview participants but was added because some participants suggested that adaptation should be manual while others suggested automatic adaptation. The last task within the "Define adaptation requirements" activity is to "Validate adaptation requirements". General requirements are often validated so adaptation requirements should be no different:

> ..we need to have something that will test and validate the requirements, and that usually comes from letting the end user try something and he says, boy you guys are bosos, thats not what i told you, even though its completely within the specification..

There were 2 tasks identified for the "Define monitoring requirements" activity. The first task was to "Define application components that have to be monitored". This task was identified by several of the interview participants with each suggesting several component types that should be monitored so that adaptation requirements can be met:

> I would monitor the state of the task, the external events, also the state of the information, and the documents and the things that are exchanged.

Another participant identified that SBAs should be highly available and that they should be able to deal with unavailable services. In order to do this he suggested that it is necessary to monitor the availability of individual services:

> Yes because, one of the basic requirements the customer always has is the the system integration should be highly available meaning that you can cope with downtime of service etc. etc.

The second task identified for this monitoring activity is to "Define how monitored events are reported". Participants stated that not only was monitoring important but that

it was important to convey monitoring results to the applications administrators and end users. Some of the participants also suggest the type of mechanisms that should be used in order to report monitored events:

> If you provide an XML service then the error message wants to be in XML format as well.

The adaptation activities and tasks identified are self contained and can be carried out in parallel with the other RE tasks that are necessary for a project. A number of adaptation related artifacts and stakeholders were also identified for the process (see Table 5.3). Unlike the activities and tasks, the artifacts and stakeholders identified are not exclusively concerned with adaptation. For example, all of the adaptation stakeholders listed here usually have other roles in other RE activities.

Stakeholders

For the purposes of brevity the artifacts included in the process model are summarised in Table 5.2, each with a supporting exemplar quotation taken from the data.

There were two types of business stakeholders identified in the requirements engineering process. "Business analysts" are part of the development team and the "Business stakeholders" are the project's business contacts. The "Business analysts" were suggested to be responsible for determining the adaptation requirements. Another participant suggested that "Solution architects" and "System administrators" should be involved with adaptation requirements for adaptation types such as service substitution. This is when services are substituted with one another rather than being recomposed to create different functionality. The stakeholders "Engineer" and "Developer" were identified as important to the requirements engineering process. In some organisations these may be the same role, i.e. writing code, however both have been kept in case engineer more specifically means software engineer rather than just developer. The "Information security team" stakeholder was described by one of the participants who himself is part of such a team. He suggested that this team might participate in tasks such as governance, risk management and compliance. The final stakeholders that were introduced were "End users", as

Artifact	Exemplar Quotation
Business analyst	"..maybe in the requirements phase the analyst have to think about the adaptive requirements.."
Business stakeholders	"..so people need to be involved, also you need to involve possibly the business people around.."
Solution architects	"So if its at the level of the endpoint determination, then the solution architects and possibly system administrators."
Engineers	" ..obviously there would always be some kind of discussion within engineers to clarify that [xx] it means in most specific [xx] and most possibly"
Developers	"..so the stakeholders if you like as i see them apart form managers there is the people who are developing the adaptation tools.."
Information security team	"There is actually quite a few stakeholders potentially who need to be involved, so depending on the size of the organisation you probably have people involved in the governance, risk management, compliance, maybe you have an information Security team with policies that are documented within your company,n security team"
System administrators	"So if its at the level of the endpoint determination, then the solution architects and possibly system administrators."
End users	"Only if there is a failure you have to come back to this, the end user decides what happens"

Table 5.2: *RE Artifacts with supporting Quotations.*

it was suggested that end users need to specify the actions to be taken when there is an adaptation failure.

All of the enhancements to SOAdapt-V1 for the RE process, based on the data collected during the first empirical study, are shown in Table 5.3.

5.2.3 Design

Activities, Tasks and Artifacts

There was one prominent activity change between the SOAdapt-FoR and SOAdapt-V1 for the design process. The "Design Adaptation Mechanisms" activity was changed to "Design Adaptation Strategies". Interviewees felt that the word "Mechanisms" was too ambiguous to describe the way in which SBAs adapt:

> Yeah, so I would call these three things out on the right side that I design my monitor, my strategy and my adaptation. I do those three things because my monitor leads to

REQUIREMENTS ENGINEERING
Adaptation Tasks Added
Define Adaptation Requirements:
-Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS Based)
-Choose between manual and automatic adaptation
-Validate adaptation requirements
Define Monitoring Requirements:
-Define application components that have to be monitored
-Define how monitored events are reported
Adaptation Artifacts Added
Adaptation workflow
Use Cases
Architectural Overview
Adaptation Stakeholders Added
Business analyst
Business stakeholders
Solution Architects
Engineers
Developers
Information security team
System Administrators
End users

Table 5.3: *Detailed Changes to the Requirements Engineering Process of SOAdapt-FoR*

correct strategy.

Adaptation strategies such as service discovery and service substitution were also identified during the inquiry. There were 8 adaptation tasks identified for designing these adaptation strategies. The task "Design adaptation strategies [service discovery, service substitution]" was described by several participants, each describing the various types of strategy that could be enacted. One of the participants described that during service substitution it is necessary for the service **Interface definitions** to be the same.

Another theme that was common amongst the interviewees for the design process was to "Define adaptation workflows" where possible. **Service topologies** were identified as necessary when designing adaptation workflows. In this context a service topology is an artifact that defines the services within service compositions and how they relate to one another. These adaptation workflows should then be incorporated into the application's **Enterprise architecture**.

..and you would have to define the workflows of their tasks are adaptive or not, so in

> some cases you have six workflows that you can model for business process modeling notation, and then they is no adaptation on the structure of the workflow, however in workflows such as hospitals the tasks can adapt to the business process and then if this task are interconnected through web services then you need to adapt the web services but I think it depends on the workflow, behind the systems..

In the case of manual adaptation there were many suggestions that the "Design of an adaptation dashboard" is necessary to control adaptation configurations during runtime. The participants suggest that the dashboard should have features that allow users to "Use a guide for selecting services" within the dashboard:

> "Yea in that case maybe the user needs some guide to select, if he has to select the service manually.."

> "Yes, so but hes not going to code WSDL or the URL, the end user, is going to decide based on descriptions.."

In the case of automatic adaptation a dashboard would be redundant, the adaptation logic would have to be programmed within the application. It still may be desirable to configure certain aspects of SBAs that adapt automatically, for example, to change SLA levels and other QoS parameters. The task "Design logic to match monitored events to adaptation strategies" was included to allow monitored events to be assigned to adaptation strategies during operation:

> ..it should be automatic although there are manual events, so if somebody is doing a task, and maybe after performing this manual task, you consider to define an event because its relevant..

The task "Design service interfaces" is necessary for both adaptable and non-adaptable SBAs. The reason that it is specified in this adaptation model is to emphasise the importance of service interfaces in adaptable applications. It is paramount that service interfaces remain constant otherwise adaptation strategies such as substitution become more difficult. The **Input/Output Data** artifact is directly related to this activity as each interface design should have a specification for input and output data.

In order to design adaptable SBAs it is necessary to "Chose service alternatives" for adaptation strategies such as service substitution and service negotiation, factors such as price and trust are relevant when choosing these alternatives:

> ..yeah, so there is a few, one comes down to actually choosing the services, it's a very compeditive decision process so you may have preferred suppliers that you choose to use because you have an ongoing relationship with them and you trust them, you may not always choose the cheapest supplier that seems to me your needs for example..

The reliability of adaptation components was also a concern for the interview participants. In order to alleviate this concern they suggested tasks such as "Adaptation strategies prototyping" and to "Design tests for adaptation strategies". Prototyping ensures that the proposed adaptation strategies will actually work, while designing tests for these components ensure that the end product works:

> ..and you need to have some type of prototyping at least int the early days, to try and understand what the captured requirements are in my experience, for capturing requirements you talk to people like myself, and i've told you one thing and you've heard something else, and you end up with something different..

There were two tasks identified within the "Design monitors" activity. The first task was to "Design monitoring services", which is quite an obvious activity, but more interestingly participants stressed the need to "Determine performance impact of monitors". One participant suggested that it is important to monitor performance indicators such as the volume of composition errors and basic response times.

All of the design activities and tasks in the design process should feed back into the "Functional design" and "Software specifications" documentation for the overall application. These artifacts are not specific to adaptation but form the bridge between these adaptation activities and the rest of the development process.

Stakeholders

There were 4 adaptation stakeholders identified for the design process. The stakeholders identified, like the RE process, are not exclusive to adaptation related activities. These

stakeholders with supporting exemplar quotations are illustrated in Table 5.4.

Artifact	Exemplar Quotation
Requirements engineers	"Mainly architects, requirement engineers, and maybe a system architect.."
Solution Architects	"..for designing the system I would say a system architect and developers.."
Developers	"..for designing the system I would say a system architect and developers.."
Maintainers	"Stakeholders are functional application architects, technical maintenance people, sometimes there are different people and sometimes there enterprise architects, because you have to have people present that know about he application and the application architecture, and what the overall strategy should be when some monitored event takes place.."

Table 5.4: *Design Artifacts with supporting Quotations.*

From the data collected about stakeholders it was often not possible to assign the stakeholders to specific tasks. Often the participants would specify that the stakeholders are relevant to the entire process rather then specific tasks or activities.

All of the enhancements to SOAdapt-FoR for the design process, based on the data collected during the study, are shown in Table 5.5.

5.2.4 Construction

The construction activity "Implement monitoring and adaptation mechanisms" was updated to "Implement adaptation strategies" to reflect the new design activity "Design adaptation strategies". The "Implement Monitors" activity was isolated as a separate activity to show the "Construct monitors" task which was identified in its specific context.

Activities, Tasks and Artifacts

Within the activity of "Implement Adaptation Strategies" there were four tasks identified during the analysis process. In the "Implement monitors" activity there was a single task. Most of the tasks relate to the implementation of the design features so nearly all have a corresponding task in the design process. For example "Construct adaptation strategies" relates to the design activity of "Designing adaptation strategies". These tasks were added to logically follow on from the design tasks but there is also evidence in

DESIGN
Adaptation Tasks Added
Design Adaptation Strategies
-Design adaptation strategies [service discovery, service substitution]
-Define adaptation workflows
-Design adaptation dashboard (manual adaptation)
-Design logic to match monitored events to adaptation strategies (automatic adaptation)
-Design servcie interfaces
-Choose service alternatives
-Use guide for selecting services
-Adaptation strategies prototyping
-Design tests for adaptation strategies
-Design monitoring services (Post mortem, real time monitoring)
-Determine performance impact of monitors
Design Monitors
-Design monitoring services (Post mortem, real time monitoring)
-Determine performance impact of monitors
Adaptation Artifacts Added
Service topology
Interface definitions
Input/Output Data
Software specifications
Functional Design
Enterprise architecture
Adaptation Stakeholders Added
Requirements engineers
Solution Architects
Developers
Maintainers

Table 5.5: *Detailed Changes to the Design Process of SOAdapt-FoR*

the data to support them. The tasks to "Construct adaptation strategies" and "Construct monitors" can also be induced from participant descriptions such as:

> Have monitor, and instead of the word mechanism, have monitor design the strategy and design the way to enact an adaptation. And then in the construction, I construct those same three things. I construct the monitor. I construct the strategy, because in that strategy I'll probably have some conditions saying if such and such, do such and such. There's probably a strategy selection. You'll finally call the correct enactment.

Participants in the study identified that **Middleware** platforms are key enablers of SBAs' adaptation. Middleware platforms provide tools that are useful for integrating disparate systems exposing homogeneous services that can be dynamically orchestrated

to achieve adaptation.

Another two activities that follow on from the design process are to “Implement adaptation dashboards” and “Early testing”. The “Implement adaptation dashboard” task is necessary to implement the **Dashboards** that were designed during the design process, while the “Early testing” task uses the test plans created in the design process. Testing during the construction process was also identified by one of the participants:

> Besides early testing , not really, I guess not only unit testing but also kind of system testing if possible, should be done as early as possible.

The “Refinement” task follows on from the early testing task by improving the adaptation mechanisms of the adaptable SBA based on feedback from the testing. This task was also suggested by the participant who proposed early testing of adaptation components:

> .. so for me construction is more or less kind of not only implementation but also testing, and refinement..

All of the design tasks and activities that result in design documents and artifacts need to have these artifacts added to the overall **Technical design** document of the adaptable SBA.

Stakeholders

There were two stakeholders identified for the construction process, “Engineers” as expected as well as “Maintainers”. The maintainers while not directly involved with construction would need to familiarise themselves with the adaptation components during construction so that they could support them when they go into production:

> The people involved are always technical maintenance people, they have to support the system when it goes into production, and application maintenance, those are the people that see at a functional level what happens during the complete process would span across multiple physical systems, you IT infrastructure and some type of important stakeholder is somebody from the business, who does acceptance.

All of the changes to the process attribute details for the construction process, including 3 artifacts and 2 stakeholders, are summarised in Table 5.6.

CONSTRUCTION
Adaptation Tasks Added
Implement Adaptation Strategies
-Construct adaptation strategies
-Implement adaptation dashboards
-Early Testing
-Refinement
Implement Monitors
-Construct monitors
Adaptation Artifacts Added
Middleware
Technical Design
Dashboards
Adaptation Stakeholders Added
Engineers
Maintainers

Table 5.6: *Detailed Changes to the Construction Process of SOAdapt-FoR*

5.2.5 Testing

The testing process was added to SOAdapt-V1 as a result of 3 testing activities that were identified by participants. Along with the testing process there were adaptation related testing activities identified for the "Design", "Construction" and "Deployment" processes. Since the testing activities were identified from scratch, rather than being derived from SOAdapt-FoR activities, the interviews did not provide a lot of detail about tasks, artifacts and stakeholders.

Activities, Tasks

There were three adaptation tasks identified in total, with only one having specific tasks identified. The "Functional testing of adaptation configurations" has the single task to "Prove to customer that application will work in all adaptation cases". It can be argued that this activity and task are necessary for all types of applications, but it is particularly important with adaptable applications because the number of adaptation configurations may be vast. In some cases the number of adaptation may be up to infinity making the testing of all configurations impossible. One of the participants felt that if each adaptation configuration is not tested, it is impossible to be confident that the application is reliable, leading to liability issues for software vendors:

> This is hardly required because if this application breaks you are liable later on. So you need to actually prove that the application would work in every case as reliably

> in what you're saying the static application would. So the streets need for or up to someone you can still need the same methodology for testing as you would in the traditional sense. But in some cases you actually need different technology to be able to even test these economic things.

Another two traditional testing activities were identified for the testing process, "System integration testing" and "User Acceptance testing of adaptation configurations". System integration testing, while useful for many application types, becomes very useful for component or service-based architectures. However, the problem with this task is that if service composition combinations are not known in advance then this task cannot be accomplished. The same applies for user acceptance testing of adaptation configurations. Therefore, these tests only apply when there is a finite number of service combinations and they are all known during the development cycle. One of the participants mentioned that testing at this stage is the only way to measure the quality of the work products being delivered by the development team:

> ..so if somebody wants to report it then we have a way to monitor it, but up until then we don't know the level of quality being delivered from the development team, except for at, what we call integration system testing, and user acceptance testing stages..

There were no adaptation artifacts or stakeholders identified during the study for this sub-process. The activities and single task identified are illustrated in Figure 5.7.

TESTING
Adaptation Tasks Added
Functional testing of adaptation configurations
-Prove to customer that application will work in all adaptation cases
System integration testing
User Acceptance testing of adaptation configurations
Adaptation Artifacts Added
Adaptation Stakeholders Added

Table 5.7: *Detailed Changes to the Testing Process of SOAdapt-FoR*

5.2.6 Deployment

Activities, Tasks

The "Deployment" process originated as the "Deployment and Provisioning" process from the SOAdapt-FoR model. As previously mentioned (section 5.2.1) the "Provisioning" process was dropped because participants did not mention it in relation to adaptation. The process contains one adaptation activity: "Deploy Adaptation Strategies and Monitors" which was changed from "Deploy Monitoring and Adaptation Mechanisms" in order to match the terminology of preceding processes. The deployment of monitors and strategies were kept as a single activity as the deployment tasks apply generally to both.

There were three tasks identified, the "Deploy components on separate systems" was identified by a participant who suggested that since SBAs are often composed of distributed services, it is often necessary to deploy components to distributed locations. This is also true of adaptation components, if adaptation components are contained in middleware then is common for the middleware to be in a seperate deployment:

> Maybe depending on the complexity of the system, if it is distributed you have to deploy most of the systems that are collaborating, if they are web services maybe you have this middleware with adaptation mechanisms in one side and you have the same middleware in another place, they are working together..

Another related activity is to "Monitor system dependencies", when adaptation components are deployed in distributed locations it may be necessary to monitor components as they are deployed in order to make sure the complete deployment is successful. One of the participants suggest that a deployment dashboard can be used to achieve this.

Finally, the task "Functional testing" was proposed during deployment to ensure that the various adaptation configurations work as well during deployment as they do in the development environment:

> ..of course you need to do your functional testing, so you need to run into test cases, or into systems states to test all your defined changes..

Stakeholders

There were two obvious stakeholders identified for the deployment process and two not so obvious. The obvious ones were the "Developers" and the "Maintainers". It is very often the same stakeholders that are involved in both the construction and deployment processes. The developers would have the most detailed knowledge of the application while the maintainers would take over the management of the application once it goes into production.

Interestingly "Service consumers" and "Service providers" were identified as necessary for the deployment process. One of the participants suggest that it is necessary to interact with the providers of services that the application depends upon. This is to ensure that the service are available and to assist with any production issues. The primary "Service consumer" in the case of an adaptable would be the developers themselves, however, if the application uses composite services then there may be consumers between the original service providers and the SBA who might need to be contacted.

> ..what I want to say is that considering adaptable service etc. they always have to have stakeholders in place from providers and consumers, or automatically of course, its quite hard to do that, the holy grail I suppose, in practice if you want to do something automatically with your provided services to another infrastructure or what ever you always have to deal with consumers.

As illustrated in Table 5.8, there were no adaptation artifacts and 4 adaptation stakeholders added to SOAdapt-FoR along with the tasks previously mentioned.

5.2.7 Operation and Management

Activities, Tasks and Artifacts

The "Operation and Management" process has retained its activity "Run-time monitoring", from the SOAdapt-FoR model, and has gained an activity called "Governance". The governance activity, as the name implies, contains tasks for governing how the SBA adapts during run-time. The "Run-time monitoring" activity has been updated with tasks

DEPLOYMENT
Adaptation Activities Added
Deploy Adaptation Strategies and Monitors
-Deploy components on separate systems
-Monitor System Dependencies
-Functional Testing
Adaptation Artifacts Added
Adaptation Stakeholders Added
Developers
Maintainers
Service providers
Service Consumers

Table 5.8: *Detailed Changes to the Deployment and Provisioning Process of SOAdapt-FoR*

for monitoring data, events and processes as well as a user notification task. The "Governance" activity has tasks for the governing the adaptation as well as negotiating Service-Level Agreements (SLAs).

The first task within the "Run-time monitoring" task is to "Audit service delivery infrastructure". This is an important task to make sure the infrastructure underlying the adaptable SBA is working within defined parameters. A key concern of one of the participants was that once one has gone to the trouble of defining service-level agreements and installing monitors, it is important to follow up with auditing the infrastructure against those service levels.

The next task in this activity is to "Notify users of critical events" when they occur. This task is relevant when adaptation is not completely automatic. When a user is notified of the event they would begin proceedings for adaptation if they deem it necessary. One of the participants suggest that it may be necessary to notify maintainers or administrators via a wireless mechanism when they are not actively monitoring the application.

The last three tasks in the "Run-time monitoring" activity all relate to monitoring different things. They relate to "Monitoring processes", "Monitoring events" and "Monitoring data". Participants felt that it was necessary to monitor these artifacts in order to get an impression of how the application is performing at various levels of abstraction. Monitoring processes refers to the business processes that are implemented as service compositions with events and data being at the individual service execution level.

In the new "Governance" activity that was identified, participants identified the tasks

of "Governing adaptation" and "Negociate SLAs". Several of the participants feel strongly about business stakeholders having a level of governance over service configuration. This governance adds some overhead to the adaptation process and prevents certain ad-hoc automatic adaptations. It may be appropriate to govern the types of adaptation that occur when the adaptation may be concerned with business criticality, safety or other critical factors.

The inability of service consumers to govern the services that they use is cited by one of the practitioners as one of the disadvantages of SBAs:

> I would say that that's the biggest challenge, cause you can get the sales pitch which tells you its all of this, but how do you prove that, there's no like, global quality of service, so this is why I was talking about governance before, there's no governance body that says, if you go to this service on Google it gives you this level of performance.

A "Sate model" was the only adaptation artifact identified without any relevant stakeholders being identified (see Table 5.9).

OPERATION & MANAGEMENT
Adaptation Activities Added
Run-time monitoring
-Audit service delivery infrastructure
-Notify users of critical events
-Monitor processes
-Monitor events
-Monitor data
Governance
-Govern adaptation
-Negociate SLAs
Adaptation Artifacts Added
State model
Adaptation Stakeholders Added

Table 5.9: *Detailed Changes to the Operation and Management Process of SOAdapt-FoR*

5.2.8 Identify Adaptation Triggers

The "Identify adaptation triggers" process has not changed in SOAdapt-V1, it still retains the single Activity: "Event Reasoning from monitoring", with no lower level tasks.

However, this activity was confirmed as a relevant adaptation task during the interviews:

> What are the kind of criteria that we would want for this type of adaptation, the trigger points for doing some adaptation.

As Table 5.10 illustrates, there were no adaptation artifacts or stakeholders identified during the interview process.

IDENTIFY ADAPTATION TRIGGERS
Adaptation Activities Added
Event Reasoning from monitoring
Adaptation Artifacts Added
Adaptation Stakeholders Added

Table 5.10: *Detailed Changes to the Identify Adaptation Triggers Process of SOAdapt-FoR*

5.2.9 Select Adaptation Strategies

The "Select adaptation strategies" process, like the "Identify adaptation triggers" process, retains its original activity from the SOAdapt-FoR model. However, in this case the 'Select adaptation strategies" activity has been updated with 3 tasks for selecting adaptation strategies. The first task identified was to "Select adaptation strategies based on a recommendation". An interview participant felt that end users would be suitable candidates to recommend services for adaptation:

> End users, that they like or dislike one, can be one mechanism for recommending.

The task to "Select adaptation strategy from an end used adaptation dashboard" reflects some of the data in the earlier processes that suggests that dashboards should be designed in order to facilitate the selection of adaptation mechanisms druing the operation of the application. The task "Select adaptation strategy to satisfy adaptation need" reflects the goal of its parent activity to select an adaptation strategy that meets the adaptation requirements of the application. A single stakeholder, "End user", has also been identified for the process (see Table 5.11), the "End user" is assigned to the tasks of choosing or recommending strategies.

An adaptation strategy, in the context of this study, is the means or mechanism by which an SBA adapts. Two strategies encountered during the study were service substitution and service re-composition. Service substitution involves exchanging one service for a functionally identical alternative. Service re-composition on the other hand involves re-composing the service workflow to an alternative configuration.

SELECT ADAPTATION STRATEGIES
Adaptation Activities Added
Select adaptation strategies to satisfy adaptation need -Select adaptation strategy based on recommendation -Select adaptation strategy from an end used adaptation dashboard -Select adaptation strategy to satisfy adaptation need
Adaptation Artifacts Added
Adaptation Stakeholders Added
End User

Table 5.11: *Detailed Changes to the Select Adaptation Strategies Process of SOAdapt-FoR*

5.2.10 Enact adaptation

The final process, "Enact adaptation", has also not had many process attributes identified. The wording of its single activity "Execute selected adaptation strategy" had been reworded to match the terminology of other activities but its intent remains the same. The "End user" stakeholder has also been identified for this process (see Table 5.12).

ENACT ADAPTATION
Adaptation Activities Added
Execute selected adaptation strategy
Adaptation Artifacts Added
Adaptation Stakeholders Added
End User

Table 5.12: *Detailed Changes to the Enact Adaptation Process of SOAdapt-FoR*

Figure 5.1 is a BPMN representation of the empirically grounded process model. The illustration shows high-level life-cycle processes and activities.

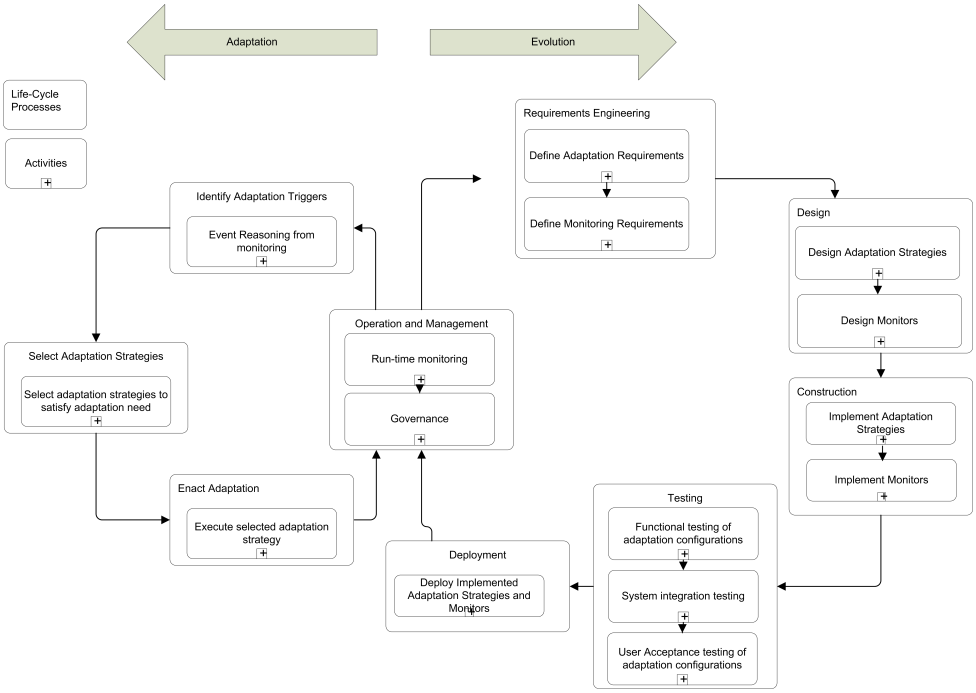

Figure 5.1: *SOAdapt-V1 - Process Model for Adapting SBAs*

5.3 Interim Evaluation

Two methods were employed to determine the external validity of SOAdapt-V1. In Section 5.3.1 I describe the comparison between the process model and a previously validated component-based software adaptation process model, indicating whether the model contains the required activities for adaptation. In Section 5.3.2 the activities from the process model are mapped to an SBA development life-cycle which demonstrates their transferability.

5.3.1 Inter-model Evaluation

To evaluate SOAdapt-V1's capability of adapting SBAs we compared to a similar model proposed and validated by Oreizy et al [85, 84] for adapting component-based applications. Component-based applications are similar to service-based applications, with both application types being composed of loosely coupled self-contained software modules. This similarity allows comparisons to be made between the two approaches.

The two models will be compared at an activity level since Oreizy et al's model does

not specify the life-cycle processes within which its activities occur. Each of Oreizy et al's activities will be analysed to see if they have equivalent or comparable activities within SOAdapt-V1. The degree of similarity between the activities will also be judged by degree of similarity between their component tasks. Oreizy et al's do not explicitly list the tasks within each activity. However, they are described verbally within their paper. These task descriptions have been extracted for the comparative analysis.

5.3.1.1 High-Level Comparison

As with SOAdapt-V1, Oreizy et al's adaptation model has two cycles: an adaptation cycle and an evolution cycle. Both models implement changes to applications during an offline evolution cycle. However, the overall approaches for adaptation are different. In SOAdapt-V1, adaptation mechanisms are built into the application at runtime. Then changes can be made to the operational application using these mechanisms. Oreizy et al change the composition of the application during evolution, while during adaptation implement the changes enacted during evolution. They argue that making changes during evolution is safer because it allows the application to be tested thoroughly before reflecting those changes in the running system. This is a valid consideration which is addressed in SOAdapt-V1 by the task "Prove to customer that application will work in all adaptation cases". This task is completed during the testing process and ensures that possible application configurations are tested off line during evolution before the application is made operational. The differences in the adaptation approach may be rooted in the fact that components are more tightly coupled than services and hence require a greater deal of testing and integrity checks during evolution. The two approaches share many of the same activities. For example, both approaches have runtime monitoring processes to detect when adaptation should be enacted. Both methods also have activities to govern and manage runtime adaptations.

5.3.1.2 Detailed Comparison

Table 5.13 was constructed in order to get a better picture of whether SOAdapt-V1 contains equivalent adaptation activities and tasks to those outlined by Oreizy et al. The left hand column contains the complete list of activities and tasks from Oreizy et al while the right hand side shows comparable or similar tasks and activities from SOAdapt-V1.

Oreizy et al	SOAdapt-V1
A1: Maintain concistency and system integrety	**Governance/Design Adaptation Strategies/Functional testing of adaptation configurations**
T1: Preserve consistant model of application architecture	Prove to customer that application will work in all adaptation cases
T2: Preserve strict correspondence between architectural model and execution implementation	Govern adaptation
T3: Develop architecture evolution manager (AEM) to mediate changes to the AEM	Design adaptation dashboard (manual adaptation)
A2: Enact Changes	**Implement Adaptation Strategies**
T4: Use an interactive editor to construct architectures and describe modifications	
T5: Use design tools to critiqe architectures as they are constructed	Early Testing
T6: Use domain specific tools to check for semantic errors	Refinement
T7: Use modification interpreter to interpret change description language scripts as AEM primitives	Construct adaptation strategies
A3: Collect observations	**Run-time monitoring**
T8: Observe and provide notifications of exceptional events such as resource shortages	Monitor Events/Noitfiy users of critical events
T9: Dynamic modification of monitors and monitored events	
T10: Model application behavior as patterns of events	
T11: Remote monitoring by human actors	Audit service delivery infrastructure
A4: Evaluate and monitor observations	**Event Reasoning from monitoring/Functional testing of adaptation configurations**
T12: Monitor behaviors of the running application and compere them to behavioral requirements	
T13 Determine all possible architectural configurations to use for concistancy checks	Prove to customer that application will work in all adaptation cases
T14: Automatic runtime concsitancy checks	
A5: Plan Changes	**Define Adaptation Requirements/Define monitoring requirements**
T15: Plan which observations are required for enacting adaptations	Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS Based)
T16: Plan which adaptations to make and when to make them	Define application components that have to be monitored
A6: Deploy change descriptions	**Execute selected adaptation strategy**
T17: Use AEM to translate change descriptions into specific application modifications	

Table 5.13: *Oreizy et al vs SOAdapt-V1*

There are six high-level adaptation activities in Oreizy et al's model labeled from A1 to A6. Each of these activities have one or more tasks labeled form T1 to T17. The activities, A1 to A6. were matched to activities from SOAdapt-V1, with 6 close matches being made. The same procedure was carried out for the tasks between the two models. It is worth noting that the tasks from SOAdapt-V1 that were mapped to Oreizy et al's model

do not necessarily belong to the SOAdapt-V1 activities that they are shown with in Table 5.13. The tasks were mapped on a best fit basis regardless of their parent activity.

It is evident from the table that all of the adaptation activities from Oreizy et al have comparable activities in SOAdapt-V1. This shows that, at a high-level at least, SOAdapt-V1 contains the basic ingredients for adaptation. The table does not show the adaptation activities identified in SOAdapt-V1 which do not have counterparts in Oreizy et al's model. SOAdapt-V1 goes beyond the basic set of activities for adaptation and considers adaptation more explicitly during evolutionary processes such as design, deployment and testing. There are some activities where Oreizy et al has more detail than SOAdapt-V1 particularly with regard to its monitoring activities. The similarities that can be seen between the two models suggest that SOAdapt-V1 is a valid adaptation model. This, however, is based on the assumption that Oreizy et al's model is itself valid and that adapting component-based applications is a comparable process to adapting SBAs.

5.3.2 Demonstrating Transferability

In order to determine the transferability of SOAdapt-V1, each of its activities were mapped to a component-based application development life-cycle. This life-cycle proposed by Durvasula et al. [32], suggests a high-level process as well as best practice guidelines for developing SBAs. The best practice guidelines are given in a non-sequential format, providing practitioners with a menu of best practices that they can select to use in conjunction with the proposed life-cycle process model. If the activities and tasks from SOAdapt-V1 can be intuitively mapped to Durvasula et al's development life-cycle, this shows the transferability of the model to a real-world development scenario.

Table 5.14 illustrates Durvasula et al's development life-cycle annotated with the activities from SOAdapt-V1 which are highlighted in bold. The mapping process here is more high-level that the previous mapping with activities and processes rather than tasks being considered.

The granularity of development activities from Durvasula et al's life-cycle and SOAdapt-

V1 are somewhat different, with Durvasula et al embedding some life-cycle processes within others. For example, Deployment is embedded as an activity within the IT Operations process. This is not a problem, but it does have some knock on effects to the mappings contained in the Table 5.14. Some of the activities from SOAdapt-V1 are represented as tasks to satisfy the different level of process granularity used by Durvasula et al.

Life-Cycle Process	**Activity**	**Tasks**
Requirements and Analysis	Map high-level business processes Prioritise business services Capture business service requirements **Define Adaptation Requirements** **Define Monitoring Requirements**	
	Architecture Review	Review requirements Review alternatives and estimate effort Propose solution
	Prioritise and add to solutions portfolio	
Design and development	Assign resources to solutions development team Design solutions - identify reuse opportunity **Design Adaptation Strategies** **Design Monitors**	
	Develop, QA and conduct UAT for business solution	**Implement Adaptation Strategies** **Implement Monitors** **Functional testing of adaptation configurations** **System integration testing** **User Acceptance testing of adaptation configurations**
IT Operations	Assign resources to service operations teams Identify infrastructure needs and establish systems environment	
	Deploy business solution	**Deploy Adaptation Strategies and Monitors**
	Maintain souution to business requirements	Run-time monitoring Governance
	Identify Adaptation Triggers	**Event Reasoning from monitoring**
	Select Adaptation Strategies	**Select adaptation strategies to satisfy adaptation need**
	Enact Adaptation	**Execute selected adaptation strategy**

Table 5.14: *Mapping SOAdapt-V1 to Durvasula et al's SBA Life-Cycle*

It is evident from Table 5.14 that the activities from SOAdapt-V1 fit into the SBA development life-cycle indicating their transferability. Changes in granularity have been made in order to make the mapping. However, this is to be expected given the variation of process attribute granularity in SBA development process models such as those reviewed in [65].

In the requirements engineering, design and construction processes all of the artifacts can be mapped to Durvasula et al's life-cycle except for the software specifications and functional design artifacts. These are common artifacts and are often present in software development projects. They may not be mentioned by Durvasula et al as they focus on

service-specific artifacts.

Most of the processes from the adaptation cycle of SOAdapt-V1 suh as “Identify adaptation triggers” and “Select adaptation strategies” do not have equivalents in Durvasula et al’s life-cycle. This is because Durvasula et al’s life-cycle does not cater for adaptation. Therefore, the addition of these adaptation processes to the life-cycle essentially converts it into a life-cycle with a runtime adaptation capability.

It is also useful to evaluate the degree of transferability between the deliverables or artifacts of each process model. The mapping in Table 5.15 was constructed in order to see the level of compatibility between the artifacts . The first column lists the artifacts from SOAdapt-V1 while the third column shows the artifacts from Durvasula et al’s life-cycle. The second column shows how the artifacts from SOAdapt-V1 would relate to the artifacts from Durvasula et al’s life-cycle during a development project.

The state model artifact from the operation and management process was the only other artifact without a match in the mapping. This shows a high level of interoperability between the artifacts of the two models. Mapping the artifacts from SOAdapt-V1 has had no negative impact on the artifacts already present in Durvasula et al’s life-cycle. The artifacts in Durvasula et al’s life-cycle focus on development of service-oriented applications in general while the artifacts in SOAdapt-V1 are primarily concerned with adaptation. Therefore, the artifacts from SOAdapt-V1 interact with those in Durvasula et al’s life-cycle as specialised artifacts which enhance the overall development process.

5.4 Conclusion

In this chapter I presented the construction process for SOAdapt-V1 based on the findings from the first empirical study conducted for this thesis. The construction process consisted of some high-level changes to SOAdapt-FoR but the most of the chapter focused on putting some flesh on the bones of the SOAdapt-FoR skeleton. This process involved the identification of activities, tasks, artifacts and stakeholders for each of the processes and sub-processes of the model.

SOAdapt-V1 Artifact	Relationship	Durvasula Artifact
Requirements Engineering		
Adaptation workflow	is integrated into	Design Models: UML, SCA service assembly model and others
Use Cases	is integrated into	Design Models: UML, SCA service assembly model and others
Architectural Overview	uses	Bindings: JMS, RMI, IIOP, HTTP(s), and others
Design		
Service topology	is integrated into	Design Models: UML, SCA service assembly model and others
Interface definitions	is integrated into	Design Models: UML, SCA service assembly model and others
Input/Output Data	is integrated into	Design Models: UML, SCA service assembly model and others
Software specifications		
Functional Design		
Enterprise architecture	uses	Bindings: JMS, RMI, IIOP, HTTP(s), and others
Construction		
Middleware	uses	Product specific metadata for configuration as well as service execution
Technical Design	is used to guide development of	Source code, Jave Documents, Release Notes
Dashboards	form part of	Source code
Operation and Management		
State Model		

Table 5.15: *Mapping SOAdapt-V1's Artifacts to Durvasula et al's Artifacts*

These process attributes were identified during a qualitative analysis process detailed in the research methodology chapter. Here, the process attributes were isolated from the interview transcripts and added to the model where relevant. In this chapter an overview of the new attributes are given with some quotations showing their sources.

An interim evaluation process was conducted on SOAdapt-V1 that contained an inter-model evaluation to show its adaptation capability while it was mapped to a SBA development life-cycle to show its transferability. Internal validity was ensured through rigorous analysis techniques and the triangulation of data sources.

Chapter 6

Phase 3: SOAdapt-V2 Construction

6.1 Introduction

In this Chapter I will present the updates to SOAdapt-V2 based on the findings from the second empirical study - Industrial Empirical Study (2). The focus of this study is on the adaptation cycle of the model, addressing the gaps in the data from the first empirical study - Industrial Empirical Study (1). The changes that are made to SOAdapt-V2 are described along with the data analysis process that was followed to identify those changes. Most of the changes to the model are attribute additions while there were a few attribute modifications and removals. As with the first study, quotations are provided where possible to support changes.

6.2 Industrial Empirical Study (2)

The data analysis in Phase 3 uncovered a lot of process attributes that can be used to enhance SOAdapt-V1. Not surprisingly, many process details were uncovered for the runtime adaptation cycle since this was the primary focus of this study. Additionally, there were many process details uncovered for the evolution cycle.

It became clear during the data analysis that there were several variations of SBA adaptation that would need to be facilitated with an adaptation process reference model. The adaptation approaches encountered in the study were either dynamic (adaptation enacted during runtime) or static (adaptation coded during development). Furthermore, within dynamic adaptation the application may self-adapt or it may be adapted manually by an application user. SOAdapt-V2 has been updated to meet the requirements of each of

these types of adaptation. Flexibility points have been added to the model which provide different options depending on the adaptation approach being followed.

While SOAdapt-V2 facilitates the major types of adaptation such as static and dynamic adaptation, it does not attempt to facilitate every conceivable type of adaptation approach. Since every real world project is different there will always be a need for some level of customisation. Therefore, the aim was to include enough detail to facilitate a generic model without including low level implementation specific details.

6.2.1 Requirements Engineering

Activities and Tasks

In the requirements engineering process, 1 task was modified and 2 artifacts were removed. The task "Choose between manual and automatic adaptation" was changed to "Choose adaptation type" to accommodate the three types of adaptation identified in the second part of the study. The artifact **Use cases** was removed as it is not specific to the adaptation of SBAs. The artifact **Architectural Overview** was removed because it could not be related to any of the activities or tasks in the model.

Apart from these changes to the existing process attributes there were 8 new tasks and 4 new artifacts added. The first task to be added was to "Define functional adaptation requirements". This task is a specialisation of the task to "Define adaptation adaptation requirements" from the first version of the model. This task has been added to reflect the importance that the participants put on differentiating between functional and non-functional adaptation requirements.

> I suppose not that...we would... I would kind of... I would concentrate on would be the more functional stuff. So it wouldn't be around system failure or anything like that. And I to want every type...you know, the requirements document would be broken down to it's smallest part. And we would follow that right through, all the way to testing.

The "Define non-functional adaptation requirements" task for capturing all other non-

functional adaptation requirements such as QoS requirements, performance and security requirements. In terms of performance monitoring, it is necessary to define the expected limits of the application's infrastructure as well as the SBA itself. QoS requirements might involve negotiations with service providers to work on service-level agreements and other quality agreements. Security requirements have also been identified as important non-functional requirements. Decisions such as which security model to use and permissions for each application user should be determined at this time. One of the participants suggested that system developers and testers should not have direct access to production systems to prevent accidental damage to the system.

Both the functional and non-functional adaptation requirements need to be added to the **Technical requirements** for the project which is often part of a projects **Requirements Specification** document.

The next task suggested was to "Model adaptive business process workflow". This task was suggested by participants who are proponents of model driven development. One of the participants suggested that modelling is key to the requirements engineering process. She suggests that the challenge lies in capturing the real world process and modelling it so that it can be incorporated into an application. In the case of adaptable SBAs the adaptive business processes need to be modelled.

Another important task that was added was the ability to "Determine feasibility of adaptation". Once adaptation requirements have been identified it is worth checking to see if this type of adaptation is possible with the tools and infrastructure that will be used for the project. Similarly if an existing application is being adapted there may be constraints built within the application that prevent adaptation.

> Exactly. Yeah, yeah. I think that when the developers can often find that they can see a really good solution it, but because they are constrained by what's already there, you know, we might have to work around what's there to...get them what they want.

In the previous version of the model there was a task to chose between static and dynamic adaptation. This task has been made more generic to suit the other more spe-

cific types of adaptation that have been identified in the second study. The task has been renamed, allowing analysts to simply "Choose adaptation type" rather than being constrained to the two previously mentioned.

The "Define alternative execution paths for adaptation" activity was identified as useful during the RE process for dynamic adaptation. When there are exhaustive adaptation requirements listing all of the adaptation scenarios, the execution paths for these requirements need to be captured so that they can be later incorporated into the designs of the application.

The "Define business rules for adaptation" task is intended to be used to set up rules in a rule engine that would be used for dynamic adaptation. Like the "Define alternative execution paths for adaptation" task, this is only possible where all of the possible execution paths are known in advance. In the case of dynamic self-adaptation where adaptation is determined on the fly, this task does not apply. One of the participants described how this might be achieved in practice:

> Yes well there is actually two ways we do it that way or we could also just have... hard coded business logic.. java classes which will in turn call services...so one example I can tell you about ...we done in Australia is ...your recording a persons address..concept of primary and secondary addresses and depending on information stored in the system you have a system invoking your system with a persons details, addresses and stuff.. some of them only have secondary addresses, for secondary address you wouldn't call the main address function...our main address service, you call the secondary address service....you get multiple address services ...primary...secondary ...

The task "Define runtime flexibility points" is common during the development of many traditional application types with flexibility points. This allows a basic type of static adaptation with a limited set of alternative execution paths. This adaptation activity is intended for static adaptation where all of the adaptation logic is decided and hard coded during the initial development of the application. One of the interview participants referred to these flexibility points as **change cases**:

> Oh absolutely yes. And, I mean, from our perspective, these things become what we call change cases at the requirement stage. So if the requirements talk about user experience stuff and business data that needs to be recorded but also the change cases that need to exist within that.

There was only one extra task identified for the "Define Monitoring Requirements" process. The task "Calculate the cost/benefit of the proposed monitors" was proposed by one of the participants who was concerned with the performance impact monitors on adaptable SBAs. He suggested that the impact of monitors is particularly important for applications with high transaction volumes.

Artifacts

There was one artifact identified that does not directly relate to the adaptation tasks and activities identified. The **Process documentation** is the document that specifies all of the process steps for the development and operations cycles. The document contains all of the process details necessary for the project and is necessary to implement the SOAdapt model itself.

The complete set of changes to the requirements engineering process are listed in Table 6.1. The quantity of process attributes identified for the requirements engineering process was surprising given that the focus was on the adaptation cycle during the interviews. Most of the activities that were identified relate to the "Define adaptation requirements" process. The findings indicate that the data from the first study had also not reached saturation point. It also points to the fact that when new adaptation tasks are identified they often require corresponding tasks in the evolution cycle.

6.2.2 Design

Activities and Tasks

The design process has had many new process attributes even though the focus of the second round of interviews was the adaptation cycle. This may be accounted for by the fact that it is difficult to speak about runtime adaptation activities without mentioning

REQUIREMENTS ENGINEERING
Adaptation Tasks Removed
-Choose between manual and automatic adaptation
Adaptation Tasks Added
Define Adaptation Requirements
-Define Functional Adaptation Requirements
-Define Non-Functional Adaptation Requirements
-Model adaptive business process workflow
-Determine feasibility of adaptation
-Choose adaptation type
-Define alternative execution paths for adaptation
-Define business rules for adaptation
-Define runtime flexability points
Define Monitoring Requirements
-Calculate the cost/benefit of the proposed monitors
Adaptation Artifacts Removed
Use Cases
Architectural Overview
Adaptation Artifacts Added
Change Cases
Technical requirements
Requirements Specification
Process documentation

Table 6.1: *Detailed Changes to the Requirements Engineering Process of SOAdapt-V1*

evolutionary activities that facilitate them.

At a high level the activity "Design adaptation strategies" activity was changed to "Design adaptation functionality" to reflect the fact that other adaptation mechanisms have to be designed in addition to adaptation strategies.

At a lower level 2 tasks were removed, "Design adaptation dashboard" was merged with the more generic "Design UI for manual dynamic adaptation" and "Design service interfaces" was merged with "Design business processes at appropriate granularity for adaptation".

A total of 7 new tasks and 7 new artifacts added to the design process, many of these specific to the different types of adaptation (static, dynamic). The were 6 identified for the "Design adaptation strategies" activity and a single task for the "Design monitors" activity.

The first task added in the "Design adaptation strategies" activity is to "Design business processes at appropriate granularity for adaptation". Some of the participants were concerned that some companies who want to adopt SOA do not design their services at the correct level of granularity. This is a particular problem for adaptation as services

cannot be combined usefully when they are not at the same level of granularity.

> ..some people have taken their scattergun approach, they've gone in and taken out and technical equivalent of shotgun and said we need some services , catchinnnggg !..

The "Specify adaptation architecture" task is aimed at the lower level technology specific architectural decisions. Many of the participants suggest that the use of platforms such as an Open Services Gateway initiative framework OSGi would enable adaptation. Other participants suggested implementing all of the adaptation functionality using custom adaptation patterns.

> So, while the programs is still running and you can swap around bundles. You can also receive events about bundles. Bundles can in fact initiate life cycle events themselves. So it's a very powerful framework for what you'd call self-adaptation. The [xxxxxxxx] product is now using OSGI technology, and we're also using it in another research project. It's basically a quality assurance tool for emulating a very large scale environment.

The tasks "Design mechanisms for static adaptation", "Design UI for manual dynamic adaptation" and "Design UI to configure self-adaptation parameters" were identified for static, manual dynamic and dynamic self-* adaptation types respectively. The interview participants each described the type of products and their development methods. During the analysis process it was determined that each of their adaptation types could be categorised as one of the aforementioned adaptation types.

This is one of the key flexibility points of the model where the selection is based on the adaptation type chosen in the RE process. The task selected here influences many of the tasks which follow in the process model, particularly those in the adaptation cycle.

Some of the interview participants suggested that it may be necessary to have different adaptation strategy types for non-functional adaptation requirements, hence the task "Design adaptation strategies for non-functional adaptation" was added. They feel that many non-functional adaptation requirements can be handled at the infrastructure level

rather than the application level. For example, one participant suggests that performance issues can be handled by spawning new virtual machines or servers.

> I see this as a process where the evolution and adaptation of the process, to say we have manual process for which we define requirements. The monitor might be static today. Of course we have the dynamic monitoring, not only for performance for a bunch of things. The adaptation mechanism might be deploying a new physical server. Or it might be, starting up a new virtual machine.

The final task in the "Specify implementation architecture for monitors" was added to the "Design monitors" activity. This task was added to reflect several different adaptation architectures proposed by the interview participants. The interviewees suggested that monitoring should be part of the infrastructure rather than the application, particularly for non-functional monitoring requirements.

Artifacts

The artifacts **Software specifications** and **Functional design** were merged into **Software design specifications** since functional designs are usually contained in the specification document. The term **Software design specifications** was also most commonly used by the participants. The artifact **Enterprise architecture** was replaced with **Application architecture** since this model is focused at the single application level, rather than multiple applications that comprise the **Enterprise architecture**. Finally **Input/Output Data** was replaced with **Interface definitions** to reflect the participants' comments. The complete set of changes is shown in Table 6.2.

6.2.3 Construction

Activities and Tasks

The construction process remains largely unchanged in SOAdapt-V2. The activity "Implement adaptation strategies" was changed to "Implement adaptation functionality" to match the terminology introduced in the design process. Additionally the task "Construct mechanisms for static/dynamic adaptation" was identified during the interviews.

DESIGN
Adaptation Tasks Removed
Design adaptation strategies
-Design adaptation dashboard
-Design service interfaces
Adaptation Tasks Added
Design Adaptation Functionality
-Design business processes at appropriate granularity for adaptation
-Specify adaptation architecture
-Design mechanisms for static adaptation
-Design UI for manual dynamic adaptation
-Design UI to configure self-adaptation parameters
-Design adaptation strategies for non-functional adaptation
Design Monitors
-Specify implementation architecture for monitors
Adaptation Artifacts Removed
Software specifications
Functional design
Enterprise architecture
Input/Output Data
Adaptation Artifacts Added
Configuration files design
User Interface design
Application architecture
Adaptation strategy designs
Adaptation mechanism designs
Software Design Specification

Table 6.2: *Detailed Changes to the Design Process of SOAdapt-V1*

Various types of adaptation mechanisms were suggested for this task. Rule/script engines, flag based editors and process editors were all suggested as mechanisms by which static and dynamic adaptation could be achieved.

Artifacts

The artifact **Monitors** was introduced as necessary for the activities and tasks relating to monitoring. The complete set of changes are listed in Table 6.3.

CONSTRUCTION
Adaptation Tasks Removed
Implement adaptation strategies
Adaptation Tasks Added
Implement Adaptation Functionality
-Construct mechanisms for static/dynamic adaptation
Adaptation Artifacts
Monitors

Table 6.3: *Detailed Changes to the Construction Process of SOAdapt-V1*

6.2.4 Testing

There are no changes to the testing process in SOAdapt-V2. While there were several adaptation related testing activities and tasks identified in the interviews they already existed in the previous version of the model. There were no new tasks or stakeholders identified either in this version of the model.

6.2.5 Deployment

The deployment process has had only minor changes in this version of SOAdapt. Its sole activity had been renamed from "Deploy Adaptation Strategies and Monitors" to "Deploy Adaptation Functionality and Monitors" in order to match the design and construction processes. The task "Plan deployment of adaptation components" was added in response to participants describing various ways that adaptation components can be deployed. A single artifact **Adaptation components** was added since it is adaptation components that get deployed during the deployment process.

DEPLOYMENT
Adaptation Tasks Removed
Deploy Adaptation Strategies and Monitors
Adaptation Tasks Added
Deploy Adaptation Functionality and Monitors
-Plan deployment of adaptation components
Adaptation Artifacts Added
Adaptation components

Table 6.4: *Detailed Changes to the Deployment Process of SOAdapt-V1*

6.2.6 Operation and Management

Activities and Tasks

The operation and management process has one new activity "Configuration management" introduced in SOAdapt-V2. The existing "Runtime monitoring" and "Negotiate SLAs" activities have had several tasks added, with no new artifacts or stakeholders being identified.

The "Run-time monitoring" activity had 3 activities added during the second study. The first activity identified for this activity was to "Monitor services properties against predefined rules". This task is relevant to most types of adaptation whether it be static or dynamic. This monitoring may be automatic or manual depending on the type of adaptation. The predefined rules may be coded into the application or be deployed as polices by end users who are monitoring the application:

> Then they also provide monitoring mechanisms for these predefined rules or policies to be monitored under the required service. [Okay] What happens with their role service, which is why the ..thinking policies, static trigger in the, what you call the console.

The next task within the "Run-time monitoring" activity is to "Monitor non-functional service properties". Rather than being a completely new task, it is a specialisation of the previous task about monitoring services properties. In this case the task more specifically monitors non-functional properties when the application has non-functional adaptation mechanisms.

The "Governance" activity has had 2 tasks added. The first was to "Manage adaptation processes". This task was added because participants were concerned that adaptation might have unexpected affects that do not get managed by any process. One of the participants was happy of a proposed reference model as that in itself would provide a level of governance around the adaptation cycle.

Another concern with participants was that adaptation may pose some sort of financial, safety, ethical or other type of tangible risk. In order to prevent this risk participants

suggested that application users or managers should “Authorise adaptations where necessary”. Adaptation cases that require authorisation are context specific. If an SBA makes trades on a stock market it may be necessary to approve an adaptation if the adaptation makes the transaction more risky.

There was a single task identified for the newly added “Configuration management” activity. The “Manage service versions during adaptation” task like many others in the model depends on the type of adaptation. Participants used configuration management for many reasons such as troubleshooting error, being able to revert to known working versions when errors occur or just to be able to keep a log of the applications’ states. Some of the participants stored their configurations in database tables while others used metadata engines to store configurations.

The complete set of changes to the reference model are listed in Table 6.5.

OPERATION & MANAGEMENT
Adaptation Tasks Added
Run-time monitoring
-Monitor services properties against predefined rules
-Monitor non-functional service properties
Governance
-Manage adaptation processes
-Authorise adaptations where necessary
Configuration management
-Manage service versions during adaptation

Table 6.5: *Detailed Changes to the Operation and Management Process of SOAdapt-V1*

6.2.7 Identify adaptation triggers

Activities and Tasks

The identify adaptation triggers process has completely evolved since the previous version of SOAdapt. This is not surprising since the adaptation cycle was the primary focus of the second study. The primary activity has changed from "Event Reasoning from monitoring" to "Trigger adaptation from monitored events" so that the activity can apply to both static and dynamic adaptation scenarios. It has three tasks that can accommodate either static, dynamic-manual or dynamic-self adaptation. It has a final task that notifies users of adaptations that have occurred.

The first task "Application adapts based on built in adaptation logic" was elicited from participants describing static adaptation where all adaptation decisions are hard coded at design time. One implementation of this task was described where policy makers decide on adaptation rules during design time with purpose built monitors to check for policy violations. This would be a closed loop system with no end-user input during the operation of the application:

> But the thing is, you have people who are providing this design time policy, design construct. Then they also provide monitoring mechanisms for these predefined rules or policies to be monitored under the required service. [I1: Okay] What happens with their role service, which is why the ..thinking policies, static trigger in the, what you call the console.

This type of static adaptation is often used in application infrastructures for performance related adaptation such as fail-over and recovery type operations. It is also commonly used to implement business logic that is not subject to change. One of the participants suggested that this type of adaptation would be useful for one of their adaptable SBAs where availability of services is a concern.

The next task in this activity is that the "Application dynamically adapts based on execution context". This is an automatic task that occurs when the execution context changes. The task does not require any input from end users or administrators but it

has been added as an alternate for the static or manual dynamic adaptation operational tasks. In some circumstances it may be desirable to have some level of control over this automatic task. In some cases it may also be desirable to manually add parameters or limits that control the automatic adaptation.

The "Application dynamically adapted by application user" task describes how administrators and end users manually adapt the SBA during runtime. This type of adaptation gives the flexibility of run time adaptation without the added complexity having to calculate its adaptation path on the fly. One of the participant companies used this adaptation type in one of their SBAs to facilitate frequently changing business rules:

> Yes, so the way we deal with that is that our administrative application allows you to define what external services are in use, where there hosted, what a formal fail over solution is "morin yeah", so for us this one flips back over, you know what I mean, because you can just go into the admin application and as you can for anything else, say oh know that's hosted somewhere else tappidy tap

Other participants described how this type of adaptation can be used to adapt an SBA in response to performance issues. For example, one participant suggested that an adaptation dashboard could be used to add additional hardware resources to an application cloud when the volume of users reach a certain threshold. The dashboard could use services to enable this type of infrastructure, with services representing each hardware or virtual hardware instance.

The final task in the "Trigger adaptation from monitored events" activity is to "Notify relevant stakeholders" that an adaptation had been triggered. Participants suggested that the notification might be just to alert end users to an adaptation or it may be to notify them on an action that they must take in relation to the adaptation. For example, in the case of an order processing application, if an aspect of the process fails then the application may adapt to cancel the order and release the stock. In this instance it might be a good idea to notify relevant stakeholders of the adaptation/cancellation so that the order can be rebooked.

A summary of the changes to the "Identify adaptation triggers" process is illustrated in Table 6.6.

IDENTIFY ADAPTATION TRIGGERS
Adaptation Tasks Removed -Event Reasoning from monitoring
Adaptation Tasks Added
Trigger adaptation from monitored events
-Application adapts based on built in adaptation logic
-Application dynamically adapts based on execution context
-Application dynamically adapted by application user
-Notify relevant stakeholders

Table 6.6: *Detailed Changes to the Identify Adaptation Triggers Process of SOAdapt-V1*

6.2.8 Initiate adaptation

Activities and Tasks

The "Initiate adaptation" process has been changed in name from the "Select adaptation strategy" process. This name change is to reflect the fact that there are other activities in the process apart from selecting the appropriate adaptation strategy. For example, the process now has two new activities for managing adaptation strategies and determining the impact that adaptation has on the overall Information Technology (IT) infrastructure.

All of the 4 existing tasks from the previous version of the reference model have been changed or expanded to suit the data collected in the second empirical study. The "Select adaptation strategy from an end used adaptation dashboard" task was moved to the "Enact adaptation" process since participants felt this is where it should occur. The "Select adaptation strategies to satisfy adaptation need" task was broken down into more detailed tasks which will be introduced later in this section. The task "Select adaptation strategy based on recommendation" was changed slightly to "Determine adaptation strategy based on recommendation" based on user feedback. Finally, the task "Select adaptation strategy to satisfy adaptation need" was removed from the process since participants considered it too general.

The activity "Manage adaptation strategies" was added to the process with the single activity "Add new adaptation strategies to the application". This task facilitated the addition of new adaptation strategies at runtime, greatly adding to the flexibility of the SBA.

One of the participants described an administration suite that their company uses to add new adaptation strategies to their SBAs using custom scripts. These scripts contain custom adaptation logic that can be used when existing strategies do not suit new runtime adaptation requirements.

The activity "Application adapts based on static business rules" encapsulates all of the tasks that should occur when initiating static adaptation during runtime. Since static adaptation is automatic, based on hard coded business rules, there may be no task to be completed. Several of the participants described different SBAs that employ static adaptation. In all cases there was no human input involved with some suggesting that governance might be useful.

The new activity "Automatically adapt to satisfy adaptation need" was added with two tasks. This activity is intended for dynamic self-adaptation. This type of adaptation is like static adaptation in that it happens automatically, but different in that the adaptation path is determined dynamically based on the application runtime context.

One of the participants described dynamic self-adaptation in one of their SBAs based on performance properties. If the SBA becomes overloaded it has to decide whether to spawn new server instances based on the number of users connected, available hardware resources, the profile of connected users (whether they are idle or active) and other performance related characteristics. The SBA takes all of these characteristics into account and makes an adaptation decision based on the information available.

Application properties used to make dynamic adaptation decisions were identified in the study as execution context or system based properties, with a single task identified to determine adaptation strategies based on context.

The "Determine adaptation strategies based on context" task was identified by participants who described adaptation based on execution contexts such as user role, user location, user preferences, system properties such as operating system:

> Exactly, yeah, the user context or user permissions. Like whether the user has certain permissions to execute the code, or it depends on which role the user belongs to, certain parts of the application would be not available. You know what I mean, this

sort of stuff.

The next activity, “Manually adapt SBA with adaptation UI” was added to accommodate the descriptions received from interview participants of manual type dynamic adaptation. During the interviews several participants from one of the companies described how most of their products allow this type of adaptation by means of administration user interfaces for adaptation.

Two tasks were identified for this activity. One was to “Determine adaptation strategy based on recommendation”. Since adaptation gets initiated manually in this case it was suggested that the appropriate adaptation strategy should be selected based on recommendations from business stakeholders.

The second task “Develop adaptation script” should be completed during the “Manually adapt SBA with adaptation UI” activity when there is no adaptation strategy available to meet the adaptation requirements:

> So we’ve got a DSL, say, a rules engine, that’s crafted for the area. There’s certain things, they’re not unique to the area, but certainly you don’t see them much else. Like the need to have the concept of temporal rules that apply for a certain amount of time; you have that in other areas as well. But there are even nuances in the way we need to do it versus whatever other do.

The script should be then introduced to the application with the adaptation UI. Finally, the “Determine impact of adaptation on IT infrastructure” was added with 2 tasks. This activity touched on the more practical issues of adaptation such as whether the IT infrastructure is actually capable for the adaptation being initiated. There were two related tasks introduced: to “Determine if new hardware is needed” and to “Co-ordinate with IT department”. These activities and tasks were elicited from a description provided by an interviewee who has to take their practical issues into account during the daily management of a large scale adaptable SBA:

> So what will happen is, the adaptation will not happen at, so all of the monitoring happens at runtime of course,we can tell in a matter of minutes if the infrastructure

is adapting correctly or not,well its not adapting, its supporting the requirements correctly, if we need to do any corrective action lets say right,then all of the processes for kicking out a new vm or any of that ,its more like an IT process,so we need support from IT,it might take time to create the new vm or any physical server depending on the case,but what we do is more of a proactive thing,we measure the requirements that we have,we measure the infrastructure to see how many transactions per second we have to support, and then we do an estimation on how much infrastructure we need.

The complete list of changes to the process are shown in Table 6.7.

INITIATE ADAPTATION
Adaptation Tasks Removed
Select adaptation strategies to satisfy adaptation need
-Select adaptation strategy based on recommendation
-Select adaptation strategy from an end used adaptation dashboard
-Select adaptation strategy to satisfy adaptation need
Adaptation Tasks Added
Manage adaptation strategies
-Add new adaptation strategies to the application
Application adapts based on static business rules
Automatically adapt to satisfy adaptation need
-Determine adaptation strategies based on context
-Determine adaptation strategies based on system properties
Manually adapt SBA with adaptation UI
-Determine adaptation strategy based on recommendation
-Develop adaptation script
Determine impact of adaptation on IT infrastructure
-Determine if new hardware is needed
-Co-ordinate with IT department

Table 6.7: *Detailed Changes to the Select Adaptation Strategy Process of SOAdapt-V1*

6.2.9 Enact adaptation

Activities and Tasks

The enact adaptation process has also had many activities and tasks added in SOAdapt-V2. The previous activity “Execute selected adaptation strategy” has been replaced with activities relating to the various types of adaptation. The three activities in this process each match one of the adaptation types identified earlier in the analysis process. The “Adapt application using UI” activity corresponds with manual dynamic adaptation, “Automatic adaptation based on context” with dynamic self adaptation and finally “Automatic

adaptation" with static adaptation.

There were four tasks identified in the "Adapt application using UI" activity of the final process, "Enact adaptation". The tasks in this activity were constructed from descriptions provided by the interview participants on how they enact manual dynamic adaptation. The first task "Adapt service composition using process editor" was added in response to descriptions of how service compositions are adapted during runtime with process editors:

> Yeah. And if you don't have it, just a general purpose scripting engine, you've got to actually code those rules into your stuff as well. So when you're writing a new rule set or a new script, you need to write the stuff to handle this temporality. Whereas we now have that in our engines, so you don't need to worry about it basically. So for that stuff we have a dedicated, for example, rules; we have a dedicated editor, a GUI that we've built in house, with drag-and-drop ... do all this stuff.

The next task, "Adapt service end-points using a configuration manager", is similar but involves the modification of configuration parameters rather than process diagrams directly. This is a more low-tech option where configurations are simply changed by modifying source files or through simple interfaces for selecting predefined configuration options. The ability to modify configuration files directly was cited as a powerful way to adapt applications but it was also considered to be more technically challenging with developers required to carry out the adaptation.

A more basic type of adaptation and possibly a more risky way of enacting manual dynamic adaptation that was identified during the study was to "Enact adaptation by patching live applications". This type of adaptation relates to patching core software modules rather than re-configuring how they execute. One participant suggested a less risky approach to patching a live system using the OGGi platform to manage adaptations.

One of the target companies employed a hybrid approach to manual dynamic adaptation where the combined more traditional evolution processes during the adaptation cycle to enact adaptation. During runtime they enact adaptation on development systems, then

migrate it to a test system and finally to the live application. This gives them the benefit of development and test cycles of their adaptation configurations while also facilitating runtime adaptation. The task "Enact adaptation in test and development application before production application" was added to reflect this participant data.

The next activity, "Automatic adaptation based on context", had only a single task identified. Since automatic adaptation is usually completely automatic the only task suggested by participants was that it may be desirable to "Approve suggested adaptation" before the adaptation is enacted by the SBA automatically. As previously mentioned in some domains it may be compulsory to approve adaptation before enactment due to safety or business critical concerns.

The final activity of the "Enact adaptation" process that of "Automatic adaptation". As mentioned this activity has been added to represent cases where the SBA adapts dynamically by itself during runtime (dynamic self-adaptation). There were not many reports from industry for implementations of this type of adaptation. Therefore, much of the data used to construct the tasks for these activity types were anecdotal or opinion.

A concrete implementation example given for the enactment of automatic adaptation was provided by one of the participants from the banking sector. He described a type of broker that facilitated automatic runtime adaptation where the adaptation path was chosen by the application based on message parameters. He described that various types of XML service messages could be sent to the broker. The message types are not all known in advance, the broker examines the message and tries to route it based on some fuzzy logic rules:

> It was service-oriented and we had a facade service in the first place and then depending on what sort of request is coming in, where if one of the properties of a service request was the name of the method to be invoked, that facade service then invokes some kind of change request to one of the available services.

The task "Automatic adaptation based on service-request properties" was added to reflect this example. The next task, "Automatic adaptation based on user input", was

suggested by a participant as a way to enact self-adaptation based on properties of the user using the application. The final task, "Automatic adaptation based on business rules", is for self-adaptation based on business rules. The business rules are used to set limits for an autonomous adaptation engine rather than to specify adaptation paths:

> Or it's more likely to take the format of rules about the business process, so for example, you can add the rule about the man and his preferred bonds, so what you do is put into the development system.

Table 6.8 shows the complete set of changes to the "Enact adaptation" process, there were no additional stakeholders or artifacts added.

ENACT ADAPTATION
Adaptation Tasks Removed
Execute selected adaptation strategy
Adaptation Tasks Added
Adapt application using UI
-Adapt service composition using process editor
-Adapt service end-points using a configuration manager
, -Enact adaptation by patching live application
-Enact adaptation in test and development application before production application
Automatic adaptation based on context
-Approve suggested adaptation
Automatic adaptation
-Automatic adaptation based on service-request properties
-Automatic adaptation based on user input
-Automatic adaptation based on business rules

Table 6.8: *Detailed Changes to the Enact Adaptation Process of SOAdapt-V1*

6.3 Conclusion

In Phase 3 a second empirical study, Industrial Empirical Study (2), was conducted to further develop SOAdapt-V1 resulting in SOAdapt-V2. Phase 3 focused on the areas that remained under developed after Phase 2. The study focused specifically on the adaptation cycle while also contributing the other evolution processes.

Data elicited during Phase 3 corroborated Phase 2 but also added important new contributions. Industrial Empirical Study (2), showed that the model should be able to facilitate static as well as the different types of dynamic adaptation if it is to be truly generic.

The study also uncovered other adaptation specific process attributes that added necessary detail to the adaptation cycle. Furthermore, the study showed that the data had reached saturation point with a diminishing return of process attributes as the data analysis concluded.

Chapter 7

Phase 4: SOAdapt-V2 Evaluation

7.1 Introduction

In Phase 4 of this research project, the external validity of SOAdapt-V2 was evaluated. The following sections contain a detailed analysis of each of the model's attributes, outlining their modification or removal where necessary.

Each section contains a table representing a process area from SOAdapt-V2. The tables contain the process attributes, the type of evaluation question, and the determination as to whether the attribute is valid, based on the answers provided by the experts.

As mentioned in the methodology chapter there were two question types (example and justification type questions) used to determine the validity of each attribute. For example, a participant might have been asked to give an example of where they have seen one of the tasks performed in a real-world scenario. If they can provide an example, then the example indicates that the task has been observed in a context independent of the study. This independent observation shows the transferability of the task and that it is fit for purpose in a real-world scenario. In this instance the question type would be recorded as **Example** in the process table.

Table 7.1 illustrates the process attributes from the requirements engineering process area of SOAdapt-V2 along with its evaluation questions and results. The questions are provided in Table 7.1 for illustrative purposes and are omitted for brevity in the later sections. The question type is provided in all of the sections.

In columns P1 and P2 the letters K (Keep), M (Modify) or D (Delete) will be used to indicate whether the answers given by the participants (P1, P2,..) suggest that the attribute should be kept, modified or deleted. The final column "Decision", also using the K , M

and D abbreviations, represents the authors final decision on the attribute based on the input from the experts.

Since there is a large volume of attributes, the detailed evaluation of each attribute will not be described here. Detail are provided for various examples which illustrate how evaluation decisions were made. The detailed commentary in the following paragraphs can be matched to the evaluation table with the attribute identification number from the evaluation table annotated to each paragraph.

7.2 Requirements Engineering

There were 26 questions asked to participants P1 and P2 as illustrated in the requirements engineering evaluation table (Table 7.1). Based on their answers, four attributes were removed with one being modified. The remainder of the section details some of the interesting examples or examples where consensus was not reached by both participants.

When asked "Can you provide an example of where you have defined functional adaptation requirements?" (**R1**), P1 said he could not provide an example as the concept is novel, while P2 was able to provide a novel real life example for adaptation requirements in a safety critical transportation scenario. This real life example, added to the weak rejection by P1, suggests the attribute should be kept.

With the question, "How does modeling adaptive business workflow help the requirements engineering process" (**R3**), the participants commented on how this task might not explicitly be carried out on projects. P1 suggested that in his experience requirements are usually textual while P2 suggested that requirements engineers may not treat adaptation and non-adaptation requirements differently. For this reason, this task will be removed on the basis that any modeling will be carried out during the traditional requirements engineering activities where necessary.

For the question "What is the cost/benefit of defining the feasibility of adaptation" (**R5**), P1 thinks that requirements engineering is about describing the problem domain rather than checking the feasibility of adaptation. P2 sees this task as beneficial providing

ID	Adaptation Activities & Tasks	Validation Question	Type	P1	P2	Decision
	Define Adaptation Requirements:					
R1	-Define Functional Adaptation Requirements	Can you provide an example of where you have defined functional adaptation requirements?	Example	D	K	K
R2	- Define Non-Functional Adaptation Requirements	Can you give an example of where you have defined non-functional adaptation requirements?	Example	D	K	K
R3	- Model adaptive business process workflow	How does modelling adaptive business workflow help the requirements engineering process?	Justification	D	D	D
R4	- Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS Based)	Why is it necessary to define events that trigger adaptation?	Justification	K	K	K
R5	- Determine feasibility of adaptation	What is the cost/benefit of defining the feasibility of adaptation?	Justification	D	K	K
R6	- Choose adaptation type	What is involved in the activity of choosing an adaptation type?	Example	K	K	K
R7	- Define alternative execution paths for adaptation	How would alternative execution paths for adaptation be defined?	Example	K	K	K
R8	- Define business rules for adaptation	Why is it necessary to define business rules to enable hard-coded adaptation?	Justification	K	K	K
R9	- Define runtime flexibility points	Can you describe how adaptation/flexibility points are identified during the requirements engineering process?	Example	D	K	K
R10	- Validate adaptation requirements	What is the advantages and disadvantages of validating adaptation requirements?	Justification	K	K	K
	Define Monitoring Requirements					
R11	- Define application components that have to be monitored	How should the monitoring of application components be specified in the RE process?	Example	K	M	M
R12	- Calculate the cost/benefit of the proposed monitors	How would you calculate the cost/benefit of SBA monitors?	Example	M	D	D
R13	- Define how monitored events are reported	Can you explain how reporting monitored events might lead to adaptation?	Example	K	K	K
	Adaptation Artifacts					
R14	Change Cases	Can you give an example of how change cases are used in the RE process?	Example	K	K	K
R15	Adaptation workflow	Can you give an example of how an adaptation workflow is used in the RE process?	Example	D	K	K
R16	Technical requirements	What should be in the technical requirements of an adaptable SBA?	Example	M	K	K
R17	Requirements Specification	How should the adaptation requirements interact with the overall requirements specification?	Example	D	D	D
R18	Process documentation	How should process documentation be used in the development of adaptable SBAs?	Example	D	K	K
	Adaptation Stakeholders					
R19	Business analyst	What adaptation related activities should business stakeholders do in the requirements engineering process?	Example	K	K	K
R20	Business stakeholders	What adaptation related activities should solution architects do in the requirements engineering process?	Example	D	D	D
R21	Solution Architects	What adaptation related activities should engineers do in the requirements engineering process?	Example	K	K	K
R22	Engineers	What adaptation related activities should engineers do in the requirements engineering process?	Example	K	K	K
R23	Developers	What adaptation related activities should developers do in the requirements engineering process?	Example	K	K	K
R24	Information security team	What adaptation related activities should security stakeholders do in the requirements engineering process?	Example	K	K	K
R25	System Administrators	What adaptation related activities should system administrators do in the requirements engineering process?	Example	K	K	K
R26	End users	What adaptation related activities should end users do in the requirements engineering process?	Example	K	K	K

Table 7.1: *Evaluation of the RE Process*

an opportunity to see if adaptation or evolution is more appropriate depending on the feasibility of adaptation. The task will be kept as it is important for requirements engineers to examine all options when specifying an application.

Both participants were able to provide examples of " What is involved in the activity of choosing an adaptation type" (**R6**). P1 gave an abstract example while P2 gave another example from a transport domain to strengthen his explanation. Based on this evidence the task will be kept in the process.

P1 answered the question "How would alternative execution paths for adaptation be defined?" (**R7**) by suggesting that the task is necessary but difficult to achieve. P2 suggested that this type of task is common in requirements engineering and can be precisely represented with logic programming languages such as Prolog. The consensus, therefore, is to keep the task even though it is difficult to enact.

When asked "Can you describe how adaptation/flexibility points are identified during the requirements engineering process?" (**R9**), both participants suggested that the activity of defining adaptation points is difficult with P2 providing an example of the task. The problem that P1 sees with this task is that the operating context of the application needs to be modeled before the adaptation points can be modeled. He suggests that it may be difficult to model the operating context of the application since real-life contexts are constantly changing. P2 provided an example of how flexibility points can be modeled. He described modeling adaptation points in a real project by logically evaluating all of the possible execution paths at different points in the application. He warned that it is not an exact science as each analyst may think differently.

For the question "What are the advantages and disadvantages of validating adaptation requirements?" (**R10**), P1 provided the disadvantage that the task would be difficult, however, also described how the task might be achieved. P2 stated that he could not see any disadvantages and that the task could be used as a safety blanket to make sure that the requirements were correct.

Both of the participants had difficulty answering the question "How should the monitoring of application components be specified in the requirements engineering process?"

(**R11**). P1 agreed that the task is useful but said that it can be difficult deriving monitoring requirements. On the other hand P2 suggests that the task of specifying adaptation, while necessary, may be beyond the scope of the requirements engineering process. P2 also suggested that a specification language may be useful in defining application components to be monitored. Since these suggestions are more design related the task will be merged with the "Design monitoring services" task in the design process.

When asked "How would you calculate the cost/benefit of SBA monitors" (**R12**) both P1 and P2 were of the opinion that the task is impractical in the requirements engineering process. P1 suggested that the task would be only be possible after monitors have been implemented or prototyped. P2 was also of the opinion that, if monitors are necessary then poor performance is not reason enough to exclude them. Therefore, testing performance is not necessary as no action would be taken based on the results of the tests.

Before answering "Can you explain how reporting monitored events might lead to adaptation?" (**R13**), P1 highlighted that in real projects it is unlikely that a new evolution cycle would commence based on monitoring reports. This, while slightly besides the point, reinforces the justification for runtime adaptation. P2 explained that reporting of monitoring events might be part of a monitoring, analysis, planning, execution adaptation cycle with reporting being necessary for analysis.

Both of the participants believed that, although adaptation requirements should be added to the requirements specification document (**R17**), there is nothing adaptation specific about this task or artifact so it is not necessary in an adaptation specific reference model.

P1 argued that process models, specifically process model diagrams, are too rigid when asked "How should process documentation be used in the development of adaptable SBAs?" (**R18**). He said the requirements engineering process was too complex with too many variations to be represented with any model. He suggested that there may be other less rigid ways to guide the process without the constraints of a reference model.

Both participants suggested that the "Business stakeholders" (**R20**) do not play a significant role in any adaptation related activities in the RE process. P1 suggested, if involved, business stakeholders should take instruction from the business analysts.

7.3 Design

The results of the evaluation for the design process are summarized in Table 7.2. There were 28 questions asked to three different participants P3, P4 and P10. During the interview with P3 there was a technical issue and the answers to the first two questions did not get recorded. To compensate for for this, data from P10 who answered these questions is used. His answers are only used for the missing segments as he stated that the design process was not his primary area of expertise.

ID	Adaptation Activities & Tasks	Type	P3	P4	Decision
	Design Adaptation Functionality				
D1	- Design business processes at appropriate granularity for adaptation	Example	M	K	K
D2	- Define adaptation workflows	Example	M	M	K
D3	- Specify adaptation architecture	Example	K	K	K
D4	- Design adaptation strategies [service discovery, service substitution]	Justification	K	K	K
D5	- Choose service alternatives	Example	K	K	K
D6	- Design mechanisms for static adaptation	Example	K	K	K
D7	- Use guide for selecting services	Justification	K	K	K
D8	- Design UI for manual dynamic adaptation	Example	K	K	K
D9	- Design UI to configure self-adaptation parameters	Example	K	K	K
D10	- Design logic to match monitored events to adaptation strategies	Example	K	M	K
D11	- Adaptation strategies prototyping	Justification	K	K	K
D12	- Design adaptation strategies for non-functional adaptation	Example	M	M	M
D13	- Design tests for adaptation strategies	Justification	K	K	K
	Design Monitors				
D14	- Specify implementation architecture for monitors	Example	K	M	M
D15	- Design monitoring services (Post mortem, real time monitoring)	Example	D	D	D
D16	- Determine performance impact of monitors	Justification	K	K	K
	Adaptation Artifacts				
D17	Service topology	Example	K	K	K
D18	Interface definitions	Example	K	K	K
D19	Configuration files design	Example	K	D	K
D20	User Interface design	Example	K	K	K
D21	Application architecture	Example	K	K	K
D22	Adaptation strategy designs	Example	K	K	K
D23	Adaptation mechanism designs	Example	K	K	K
D24	Software Design Specification	Example	K	K	K
	Adaptation Stakeholders				
D25	Requirements engineers	Example	K	K	K
D26	Solution Architects	Example	K	K	K
D27	Developers	Example	K	K	K
D28	Maintainers	Example	K	K	K

Table 7.2: *Evaluation of the Design Process*

When asked to provide an "example of how designing business processes at appro-

priate granularity improve the design of adaptation functionality" (**D1**), both P10 and P4 suggested that business processes granularity is more related to the design process. P1 suggested more fine grained business processes can give rise to more fine grained adaptation requirements.

When asked to provide an example of where he defined an adaptation workflow, P10 stated that he developed many workflows in many domains and the process is the same for adaptive and non-adaptive systems. This suggests that the "Define adaptation workflow" (**D2**) task is not adaptation specific. P4 mentioned that he has seen no practical examples of this task but has seen it in papers. This task will be kept in the model since P4 provided academic examples. A note will be added to the final version of the model suggesting that the task could be combined with non-adaptation specific modeling tasks as suggested by P10.

P3 suggested that service alternatives might not always be chosen at design time (**D5**), when asked how service alternatives should be chosen during the design process. She suggested that this may be also a runtime activity for dynamically adapting SBAs. P4 gave an example of this task being carried out manually with a recommendation that it should be automatically enacted through the use of semantic service descriptions.

Both participants thought that it was advantageous to use a guide to aid the selection of services (**D7**). P3 mentioned that guides makes large quantities of services available while P4 highlighted that guides provide a standard method of finding services for designers. P3 was concerned about how the integrity of the services within the guides would be ensured. P4 was also concerned that the use of guides means that the service provider is not readily available to negotiate the consumption of their services. On balance, the advantages outweigh the disadvantages and the task will be retained in the model.

P3's answer for the question "How might user interfaces be used to configure self-adaptation parameters during runtime?" (**D9**) was unintelligible from the audio so data from P10 was used once again. P10 articulated an example of how this task might get executed, strengthening its case for retention in the model. P4 suggested that this task is only appropriate in applications with this type of adaptation capability. In the model itself

this task is in fact specified only for dynamic adaptation as P4 suggests.

P3 pointed out that one would only "Design logic to match monitored events to adaptation strategies" (**D10**) when dealing with certain static adaptation types. P4 on the other hand thinks that the task is necessary but might be more suited to the RE process since monitoring rules and the adaptation strategies are derived from adaptation requirements. Since there is no consensus on moving the task it will be kept in the design process.

When asked "How should non-functional adaptation strategies be designed" (**D12**), P3 said that they should not be designed any differently for functional adaptation. She also said that non-functional adaptation in her experience is one of the most common adaptation types. P4 found it difficult to answer the question because he said that he was not aware of any approaches for designing adaptation strategies. He said that this was a gap in the area and specific guidance for designers of adaptable SBAs would be welcome. Based on this feedback this task will be combined with the "Design adaptation strategies for functional adaptation" task.

Both participants agreed that monitoring architecture is necessary for adaptable SBAs (**D14**). P3 did not provide any examples of the task, while P4 provided an example of an application framework that has built in monitoring capabilities. This suggests that the design of monitoring architecture is only necessary when this capability is not already part of the deployment infrastructure. This caveat will be added to the task in the final version of the model.

When asked to provide examples of monitoring services such as postmortem or real time monitoring services (**D15**), P3 suggested that real-time monitoring is necessary while postmortem monitoring is not useful for SBA adaptation. She did not comment on whether or not it is useful to provide monitoring capabilities as a service. P4 was not able to cite any examples of monitoring services. Since neither participant acknowledged a benefit of providing monitoring capabilities as a service it will be removed from the model.

Once again data from P3 was unavailable for the question "Can you give an example of where interface definitions are used in the design process?" (**D18**). In this case unfor-

tunately P10 also did not answer that question. In his answer, P4 articulated how service interfaces are used in the design process, providing a real-life example to support his point. Based on the strength of evidence from P4 the task will be kept in the model.

P3 gave an example of where configuration files are used in adaptive SBAs (**D19**), while P4 did not see the relevancy in specifying the storage medium of configuration data. Since P3 provided an example and P4 did not provide strong basis for exclusion, the task will be kept in the model.

When discussing how adaptation specifications should be treated in relation to the overall design specification (**D24**) both participants felt that it was a good idea to keep the specifications separate but to store them all within the design specification document. Therefore, the design specification artifact will remain as it serves this purpose.

7.4 Construction

The results of the evaluation of the construction process is summarized in Table 7.3. There were 12 attributes with the same number of questions asked to participants P3 and P5. All of the original attributes were retained unmodified after the evaluation process.

ID	Adaptation Activities & Tasks	Type	P3	P5	Decision
	Implement Adaptation Functionality				
C1	- Construct adaptation strategies	Example	K	K	K
C2	- Construct mechanisms for static/dynamic adaptation	Example	K	K	K
C3	- Implement adaptation dashboards	Example	K	K	K
C4	- Early Testing	Justification	K	K	K
C5	- Refinement	Justification	K	K	K
	Implement Monitors				
C6	- Construct monitors	Example	K	K	K
	Adaptation Artifacts				
C7	Middleware	Example	K	D	K
C8	Technical Design	Example	D	D	K
C9	Dashboards	Example	K	K	K
C10	Monitors	Justification	K	K	K
	Adaptation Stakeholders				
C11	Engineers	Example	K	K	K
C12	Maintainers	Example	K	K	K

Table 7.3: *Evaluation of the Construction Process*

When asked how adaptation strategies should be constructed (**C1**), P3 said it depends

on the type of adaptation while P5 gave an example of how dynamic adaptation might be implemented. Neither of these responses indicate that the task should be removed from the model or modified.

Both participants agreed that it was necessary to "Implement adaptation dashboards" (**C3**) for some or all adaptation types. P3 limited her recommendation to use dashboards only when human actors are involved in the adaptation process. P5 on the other hand suggested that adaptation dashboards would be beneficial for most types of adaptation even if human actors were not directly involved in the adaptation process.

For the tasks "Early testing" and "Refinement" (**C4,C5**), both participants agreed that adaptation components would receive the same benefit as typical software components. In that regard it could be argued that these adaptation tasks are not adaptation specific and should be built into any mature design process.

The single task "Construct monitors" (**C6**) from the "Implement monitors" activity was evaluated with the question "Can you explain how monitors should be implemented?". P3 provided a single example as justification for the task while P5 provided two examples of how monitors should be constructed.

P3 was able to give an elaborate description of how middleware can form part of an adaptable SBA (**C7**). She described how middleware can be used in the construction of certain adaptation mechanisms. P5 was not able to provide an example so the artifact will be weakly accepted.

Both participants thought that the "Technical design" (**C8**) document was not specific to adaptation. They suggested that the adaptation designs be added to the technical design document like all the other application design features. Even though the artifact is not adaptation specific it will be kept in the model to show where SOAdapt interacts with complete development life-cycles.

7.5 Testing

The evaluation table for the testing process is illustrated in Table 7.4. There were only 3 questions in this process which were answered by P3 and P5. Two of the activities were kept unchanged while the other was modified.

ID	Adaptation Activities & Tasks	Type	P3	P5	Decision
T1	Functional testing of adaptation configurations - Prove to customer that application will work in all adaptation cases	Justification	K	K	K
T2	System integration testing	Example	M	M	M
T3	User Acceptance testing of adaptation configurations	Example	K	K	K
	Adaptation Artifacts				
	Adaptation Stakeholders				

Table 7.4: *Evaluation of the Testing Process*

The first question asked to the participants in this section was "What are the advantages and disadvantages of proving that the application will work in all adaptation cases?" (**T1**). The objective here was to justify the inclusion of the task in the model. P3 agreed that the task would be useful but there was a problem in that it is not possible to identify all change cases in dynamic systems. P5 also identified the same limitation of this task but suggested that one could test all known or desired adaptation cases and quarantine the rest of them.

The problem that P3 had with integration testing of adaptation configurations (**T2**) is that the integration tests become invalid when the SBA adapts. She suggested that integration testing should be kept as close as possible to execution and that it might be useful to build in some form of integration testing into the application. P5 suggests that to solve this problem, integration tests could be conducted using assumed application behaviors in the place of real behaviors. Therefore the task will be kept and annotated with these suggestions.

The final task "User Acceptance testing of adaptation configurations" (**T3**) was subject to the same limitations as the "System integration testing" task according to the interview participants. This, they say, is because it is not possible to test adaptation configurations that have not been pre-determined. Therefore, this suggests that this task is only suitable

for static adaptation.

7.6 Deployment

Table 7.5 shows the evaluation table for the deployment process. All of the attributes remain unchanged apart from one that was removed.

ID	Adaptation Activities & Tasks	Type	P5	P6	Decision
	Deploy Adaptation Functionality and Monitors				
DP1	- Plan deployment of adaptation components	Example	D	D	D
DP2	- Deploy components on separate systems	Justification	D	K	K
DP3	- Monitor System Dependencies	Example	K	D	K
DP4	- Functional Testing	Example	K	K	K
	Adaptation Artifacts				
DP5	Adaptation components	Example	D	D	K
	Adaptation Stakeholders				
DP6	Developers	Example	K	K	K
DP7	Maintainers	Example	K	K	K
DP8	Service providers	Example	K	K	K
DP9	Service Consumers	Example	K	M	K

Table 7.5: *Evaluation of the Deployment Process*

Both participants suggested that planning the deployment of adaptation components (**DP1**) works much the same as planning the deployment of typical software components. They said that they should be packaged with the rest of the application and deployed at the same time. For this reason this task will be removed from the model as the consensus is that it is not adaptation specific.

P5 was not able to provide an example of where adaptation components should be deployed on separate systems (**DP2**). P6 on the other hand suggested that this would be necessary for distributed business processes where different system nodes enact different parts of the process. Based on this one example the task will be kept in the model.

In the case of "Monitoring system dependencies" (**DP3**), P5 described at length a scenario where the monitoring of system dependencies at deployment time would ensure that all of the expected components are available. He also stressed that in the case of adaptable SBAs, system dependencies might need to be checked after the application adapts as well as during the deployment process. P6 was not able to provide any data for

this task so it will remain unchanged.

Both participants again see functional testing (**DP4**) as useful during the deployment process. P5 says the testing schedule would depend on whether one wants to test the SBA itself or the adaptation configurations that are possible with the application. P6 says that functional testing during deployment allows engineers delay choice of specific component services until candidates have been tested.

When asked about the "Adaptation components" (**DP5**) artifact, participants were in consensus that they shouldn't be treated any differently to the rest of the application. However, previously there was some agreement that adaptation components in some circumstances might need to be deployed on different systems. Therefore, it is still necessary to consider adaptation components as separate entities during some deployment processes. For this reason the adaptation components artifact will be kept in the model.

While acknowledging the role of the "Service consumers" (**DP9**), P6 suggested that the service consumers are really just the developers and their tasks include monitoring and negotiating Service-Level Agreements (SLAs).

7.7 Operation and Management

Table 7.6 contains the evaluation details for the operation and management process attributes. All of the attributes remain unchanged after the evaluation process.

P7 felt that it would be very complicated to monitor functional service properties against pre-defined rules (**O2**). He said that it would be easier to monitor the properties of simple services but becomes more challenging for complex services or service compositions. However, the difficulty of the task doesn't warrant the modification or removal of the task from the model.

P6 suggested that the task of "Monitoring non-functional service properties" (**O3**) would be very similar to the task of "Monitoring functional service properties". P7 disagreed and suggested that it is much easier to monitor non-functional services such as availability and response times by logging this data when services are called.

ID	Adaptation Activities & Tasks	Type	P6	P7	Decision
	Run-time monitoring				
O1	- Audit service delivery infrastructure	Example	K	K	K
O2	- Monitor services properties against predefined rules	Example	K	K	K
O3	- Monitor non-functional service properties	Example	K	K	K
O4	- Notify users of critical events	Example	K	K	K
O5	- Trigger adaptation from monitored events	Example	K	K	K
	Governance				
O6	- Manage adaptation processes	Example	K	K	K
O7	- Authorize adaptations where necessary	Justification	K	K	K
O8	- Negotiate SLAs	Justification	K	K	K
	Configuration management				
O9	- Manage service versions during adaptation	Example	K	K	K
	Adaptation Artifacts				
O10	State Model	Example	K	K	K
	Adaptation Stakeholders				

Table 7.6: *Evaluation of the Operation and Management Process*

P6 discussed some basic examples of how an "adaptation is triggered from a monitored event" (**O5**) in an adaptable SBA. P7 went into detail about when the triggering of adaptation should be automatic and when it should be based on a human in the loop. He said that the task can be automatic when the adaptation would do no harm to the application or the humans, but not otherwise. He said that there are also some ethical concerns that should be addressed when the application is being designed.

P6 and P7 had different views on how to "Manage adaptation processes" (**O6**). P6 believes that the key to managing adaptation is to adapt conservatively minimizing any problems that might occur. P6 believes that the key to managing adaptation successfully is to document everything and lock down the system. He admitted that locking down an adaptable SBA would be difficult but that it should be possible to lock down the non-adaptive parts.

P6 feels that authorization should be sought for adaptation when the adaptation affects the general execution process of the SBA rather than the execution of a single process instance (**O7**). P7 provided a more philosophical answer and said that it would be necessary to authorize adaptations if the adaptation could cause harm to humans. He also said authorization would be necessary if the adaptation had the potential to cause financial loss to the application's owner.

P6 was not able to provide an example when asked "Can you provide an example of service version management during adaptation?" (**O9**). P7 said that service version management or configuration management for SBAs would be very important if the application contained rules or configurations that pertained to law. Since law is continuously changing it would be necessary, as he sees it, to keep track of old configurations so that they can be reviewed within the context of the times that particular laws were enacted. Based on this compelling reason the task will be retained.

When asked to provide an example of the use of state models (**O10**), P6 said they should be used for reasoning about the current state as well as potential adaptation configurations in adaptable SBAs. P7 sees this type of reasoning only as practical when the adaptable SBA has a small amount of adaptation configurations. He says that it is computationally impossible to reason about the state of an application that has an exponential number of adaptation configurations.

7.8 Identify Adaptation Triggers

The evaluation table, "Identify adaptation triggers", for the first process in the adaptation cycle is shown in Table 7.7. All of the attributes remain unchanged after the evaluation process.

ID	Adaptation Activities & Tasks	Type	P8	P9	Decision
	Trigger adaptation from monitored events	Example	K	K	K
I1	- Application adapts based on built in adaptation logic	Example	K	K	K
I2	- Application dynamically adapts based on execution context	Example	K	K	K
I3	- Application dynamically adapted by application user	Example	K	K	K
I4	- Notify relevant stakeholders				
	Adaptation Artifacts				
	Adaptation Stakeholders				

Table 7.7: *Evaluation of the Identify Adaptation Triggers Process*

The first question posed to participants P8 and P9 for the adaptation in the identify adaptation triggers process category was "How might adaptation get triggered from built

in adaptation logic?" (**I1**). Both participants were able to articulate how this might be achieved using real examples. P8 suggested a rule based approach while P9 suggested quality prediction and monitoring approaches.

P8 suggested that the enactment of the task "Application dynamically adapts based on execution context" (**I2**) is similar to that of the previous task "Application adapts based on built in adaptation logic". P9 wasn't able to provide much feedback on the task but said that user task models might be useful for context based adaptation. Neither participant provided negative feedback on the task so it will be retained in the model.

Both participants see the value in "Notifying relevant stakeholders" (**I4**) that an adaptation has occurred. P8 highlighted that if there is too much feedback to end users then they might become a cause of annoyance. P9 didn't see any disadvantages to keeping end users "notified or aware of adaptation in order to be prepared for example for co-sequences of adaptation".

7.9 Initiate Adaptation

The evaluation of the process attributes for the "Initiate adaptation" process are shown in Table 7.8. The task "Develop adaptation script" was removed from the model while all of the other attributes remained unchanged.

Both participants agreed that the "Application adapts based on static business rules" (**IA2**) was the typical classic example of static adaptation that is enacted in both service and non-service based adaptable applications. P8 provided an example of a travel reservation application that would have business rules associated with its execution context.

P8 gave a basic example of how dynamic adaptation might involve "Determining adaptation strategies based on context" (**IA3**) by using properties discovered over the internet as context. P9 on the other hand was not able to provide any examples of this task. On balance the task will be kept as there was no negative feedback.

When asked to provide an example of "where an SBA adapts based on system properties?" (**IA4**) both participants provided different answers. P8 suggested that system

ID	Adaptation Activities & Tasks	Type	P8	P9	Decision
	Manage adaptation strategies				
IA1	- Add new adaptation strategies to the application	Example	K	K	K
IA2	Application adapts based on static business rules	Example	K	K	K
	Automatically adapt to satisfy adaptation need				
IA3	- Determine adaptation strategies based on context	Example	D	K	K
IA4	-Determine adaptation strategies based on system properties	Example	K	K	K
	Manually adapt SBA with adaptation UI				
IA5	- Determine adaptation strategy based on recommendation	Example	K	K	K
IA6	- Develop adaptation script	Example	D	D	D
	Determine impact of adaptation on IT infrastructure				
IA7	- Determine if new hardware is needed	Justification	K	K	K
IA8	- Co-ordinate with IT department	Justification	K	K	K
	Adaptation Artifacts				
	Adaptation Stakeholders				
IA9	End user	Example	K	K	K

Table 7.8: *Evaluation of the Initiate Adaptation Process*

properties such as device type could be taken into consideration for certain adaptation types. P9 took a different approach and suggested that service substitution could be enacted if individual services' performance falls outside of acceptable ranges.

In the "Manually adapt SBA with adaptation UI" (**IA5**) activity, both participants were positive about the suggestion of determining "adaptation strategies based on recommendation". P8 said that it is important to give users some control and to make them feel like they are in control.

P8 was not sure when it was appropriate to "develop adaptation scripts" (**IA6**) for manual adaptation. P9 on the other hand was confused with the question and felt that the task was too low level to have included in a generic process reference model. For this reason, this task will be removed form the model.

Both participants approved of the tasks to "Determine if new hardware is needed" (**IA7, IA8**) and to "Co-ordinate with IT department". They agreed that it would be necessary to work with the IT department if performance of the adaptable SBA was impacted by any type of adaptation during runtime.

When asked about the role of "End users" (**IA9**) in the initiate adaptation process, once again, both participants were in agreement and suggested that the role of the end

user very much depended on the type of adaptation being considered.

7.10 Enact Adaptation

Table 7.9 is the final process evaluation table showing the evaluation of the attributes from the enact adaptation process. All attributes remain unchanged with consensus by the participants for all questions apart from E1, E2 and E3.

ID	Adaptation Activities & Tasks	Type	P8	P9	Decision
	Adapt application using UI				
E1	- Adapt service composition using process editor	Example	K	D	K
E2	- Adapt service end-points using a configuration manager	Example	K	D	K
E3	- Enact adaptation by patching live application	Example	K	D	K
E4	- Enact adaptation in test and development application before production application	Example	K	K	K
	Automatic adaptation based on context				
E5	- Approve suggested adaptation	Justification	K	K	K
	Automatic adaptation				
E6	- Automatic adaptation based on service-request properties	Example	K	K	K
E7	- Automatic adaptation based on user input	Example	K	K	K
E8	- Automatic adaptation based on business rules	Example	K	K	K
	Adaptation Artifacts				
	Adaptation Stakeholders				
E9	End user	Example	K	D	K

Table 7.9: *Evaluation of the Enact Adaptation Process*

When asked about adapting "service composition using process editor" (**E1**) neither participants gave detailed answers. P8 mentioned a travel application scenario that might use a process editor but was not able to provide a real example. P9 was not able to provide any example of the task being implemented. Since there was a single scenario description and no negative feedback the task will be kept.

The feedback for the task "Adapt service end-points using a configuration manager" (**E2**) was very similar to the previous task. P8 provided a trivial example and P9 did not provide any example. The lack of data here seems to stem from lack of familiarity with the concepts rather than disagreement with the tasks. Again since there was no explicit disagreement the task will be kept.

Again, P9 did not provide an answer when asked for an example of "where an SBA has been adapted by patching a live application" (**E3**). P8 suggested that this task is necessary when a security update needs to be pushed out to a live application.

The final three tasks in this process relate to automatic adaptation based on service-request properties, user input and business rules (**E6,E7,E8**). Both participants provided practical examples of when each task might be enacted with P8 providing more detail in his answers.

For the final stakeholder, "End user" (**E8**), P8 thought that the end user should be available to provide high-level guidance to the adaptation process when adaptation is not completely automatic. P9 didn't think the "End user" needs to be involved in the enactment process. The stakeholder will be kept in the model despite the feedback from P9 since in my opinion they are critical for manual adaptation scenarios.

7.11 Conclusion

In Phase 4, the final phase of the research, SOAdapt-V2 was evaluated through an expert opinion survey. Each attribute from the model was the basis for an evaluation question in the survey. During the survey each question was answered by at least two participants. The results of the survey were used to determine whether the attributes from the model should be modified, removed or left unchanged.

Based on the findings from the evaluation, the completed model, SOAdapt, is presented in Chapter 8.

Chapter 8

SOAdapt Final Model Description

8.1 Introduction

The process reference model, SOAdapt, outlined here, is intended for the development of adaptable Service-Based Applications (SBAs). It is composed of tasks, activities, artifacts and stakeholder processes attributes.

The model was developed using data from an extensive literature search and has been iteratively enhanced with data from several industry based empirical studies. The empirical studies consisted of expert opinion interviews and case studies. Finally, the model was validated through an expert opinion evaluation process with academic experts.

The model has nine development processes each containing several activities which are in turn composed of executable tasks. Additionally, the model outlines the stakeholders and artifacts that are associated with each of the processes. Figure 8.1 is a Business Process Modelling Notation (BPMN) representation of the high-level activities and processes of SOAdapt.

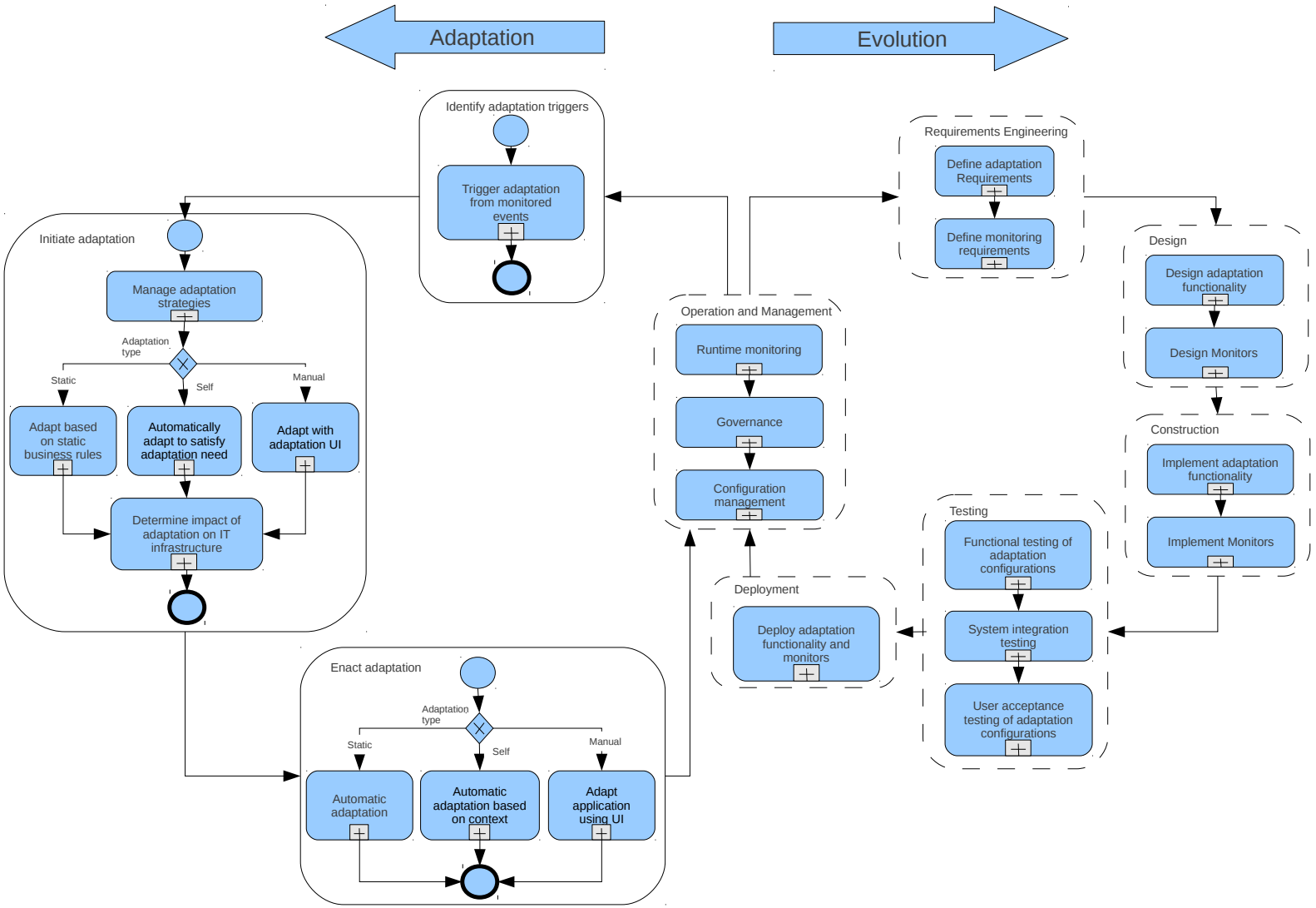

Figure 8.1: *SOAdapt Process Reference Model*

8.2 Model Usage

This model does not attempt to guide the entire application development life-cycle. Instead, it focuses specifically on adaptation related activities. Therefore, it needs to be used in conjunction with a software development process that deals with the non-adaptation related aspects of software development. It may be used with an organisation's existing software development process or a software development process reference model such as the Rational Unified Process (RUP). Throughout this chapter, process tables list the process attributes for each process.

In addition to the process tables, each process is represented in BPMN which shows the sequence of activities within each process. The BPMN diagrams also show how the software development artifacts interact with the model's activities.

8.2.1 Requirements Engineering

In the SOAdapt model, we are interested in capturing adaptation specific requirements for SBAs. Adaptable SBAs will also have many other requirements that are not related to adaptation, but those are outside the scope of the SOAdapt model. Within SOAdapt there are 2 requirements engineering activities that are specific to adaptation.

Define Adaptation Requirements refers to the activity of defining or eliciting requirements relating to why it is necessary for the SBA to adapt. An adaptation requirement may be, for example, if the Quality of Service (QoS) of service A falls below a minimum threshold then the application adapts by choosing an alternative service. The adaptation requirement may also state how the adaptation occurs. Adaptation might simply involve choosing between 2 hard coded service end points or in a more complex way by negotiating with a service directory. An adaptation requirement may also specify whether or not the adaptation occurs with user intervention or automatically with minimal or no user input.

Define Monitoring Requirements Once adaptation requirements have been determined,

monitoring requirements need to be defined in order to monitor service attributes that relate to adaptation requirements. For example, in order to enable an application to adapt based on QoS, it is necessary to monitor relevant service quality attributes such as service latency and service reliability. So, in this case, the *monitoring of service latency and reliability* is a monitoring requirement, relating to the *adapt if service QoS falls below a minimum threshold* adaptation requirement.

The complete process table is shown in Table 8.1 [1], with its BPMN representation illustrated in Figure 8.2. Here we can see that the the process workflow splits in two with alternative tasks for dynamic and static adaptation. Additionally, it shows how the various adaptation tasks provided input into the requirements engineering document for the overall development process.

[1] In the activities and tasks section of the process tables, activities are listed with their associated tasks underneath them. The requisite artifacts and process stakeholders are also listed.

REQUIREMENTS ENGINEERING
Adaptation Activities & Tasks
Define Adaptation Requirements
- Define Functional Adaptation Requirements
- Define Non-Functional Adaptation Requirements
- Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS Based)
- Determine feasibility of adaptation
- Choose adaptation type
- Define alternative execution paths for adaptation
- Define business rules for adaptation
- Define runtime flexibility points
- Validate adaptation requirements
, Define Monitoring Requirements
- Define how monitored events are reported
Adaptation Artifacts
Change cases
Adaptation workflow
Technical requirements
Process documentation
Adaptation stakeholders
Business analyst
Solution architects
Engineers
Developers
Information security team
System administrators
End users

Table 8.1: *Process Attributes for the Requirements Engineering Process*

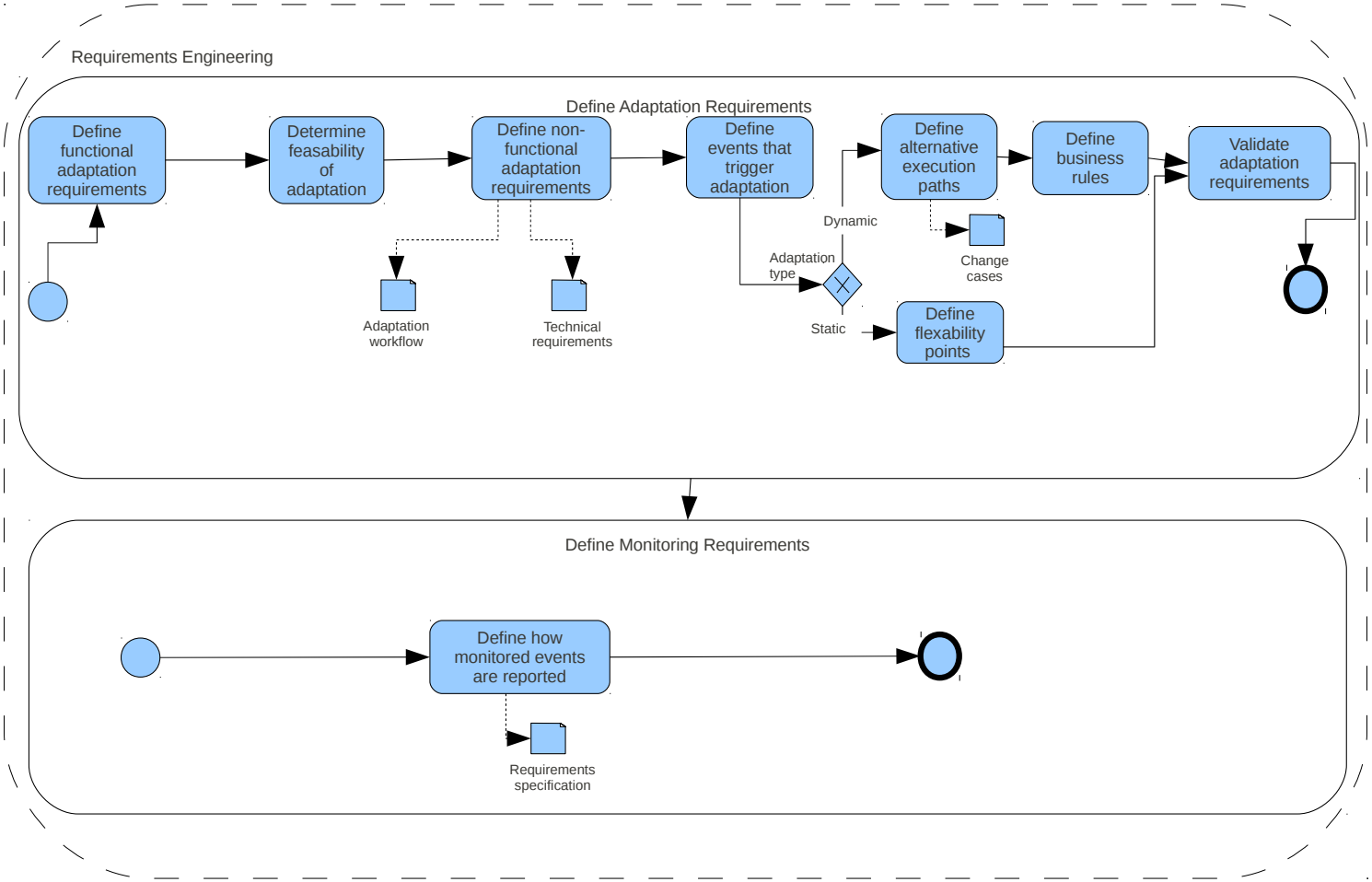

Figure 8.2: *Requirements Engineering Process Represented in BPMN*

8.2.2 Design

There are two adaptation specific design activities within the design process of SOAdapt:

Design Monitors is a design activity where monitors are designed in order to satisfy the monitoring requirements defined during the requirements engineering process. Details such as implementation technology and infrastructure need to be considered at this stage. The monitor designs need to be included with a design specification document or a similar alternative that can be used by the application developers.

Design Adaptation Functionality Once monitors have been designed, a means to adapt the application based on monitored events must be designed. The application must have the functionality to respond to monitored events and adapt by enacting an adaptation strategy. Bucchiarone et al [18] identified many possible adaptation strategies such as substituting services with known alternatives or service renegotiation which involves negotiation with a service directory in order to locate alternative services. It is at this point that it must be determined which of these strategies will be used in order to adapt during runtime. Once adaptation strategies have been decided upon, the technical and infrastructural details of how they are implemented and triggered, need to be designed.

The complete process table is shown in Table 8.2, with its BPMN representation illustrated in Figure 8.3. The process has three different workflows for static, self and manual adaptation. Another interesting feature of the diagram is how the requirements engineering document is required for many of the design activities and many of the activities also feed into the design specification document. This shows how these adaptation related activities relate to the overall application development life-cycle.

DESIGN
Adaptation Activities & Tasks
Design Adaptation Functionality
- Design business processes at appropriate granularity for adaptation
- Define adaptation workflows
- Specify adaptation architecture
- Design adaptation strategies [service discovery, service substitution]
- Choose service alternatives
- Design mechanisms for static adaptation
- Use guide for selecting services
- Design UI for manual dynamic adaptation
- Design UI to configure self-adaptation parameters
- Design logic to match monitored events to adaptation strategies
- Adaptation strategies prototyping
- Design tests for adaptation strategies
Design Monitors
- Specify implementation architecture for monitors when necessary
- Determine performance impact of monitors
Adaptation Artifacts
Service topology
Interface definitions
Configuration files design
User Interface design
Application architecture
Adaptation strategy designs
Adaptation mechanism designs
Software design specification
Adaptation Stakeholders
Requirements engineers
Solution architects
Developers
Maintainers

Table 8.2: *Process Attributes for the Design Process*

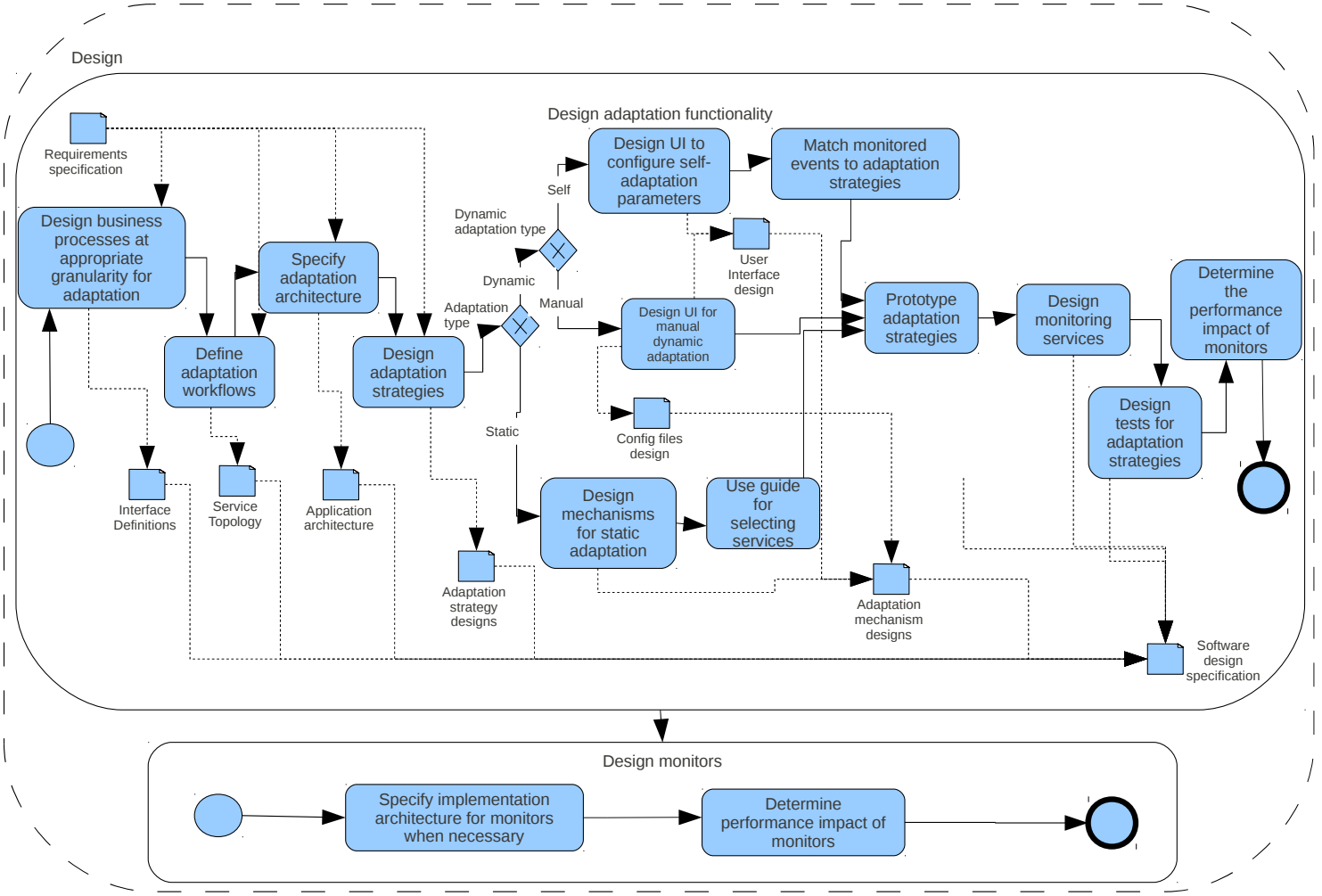

Figure 8.3: *Design Process Represented in BPMN*

8.2.3 Construction

There are two adaptation specific construction activities included in SOAdapt.

Implement Monitors This activity involves the implementation of the application monitors that are designed in the design process.

Implement Adaptation Functionality This activity involves the implementation or coding of the adaptation functionality that was described in the design process. This activity would occur along with all of the other application implementation activities within the software development project. There are no large design features to be decided at this point as they are specified during the design process. However, due to the complex nature of adaptation functionality, this implementation is one of the most challenging parts of the development process.

The complete process table is shown in Table 8.3, with its BPMN representation illustrated in Figure 8.4. The tasks in both activities are sequential without any alternative paths that depend on adaptation type. This is because all construction tasks identified are suitable for each type of adaptation.

CONSTRUCTION
Adaptation Activities & Tasks
Implement Adaptation Functionality
- Construct adaptation strategies
- Construct mechanisms for static/dynamic adaptation
- Implement adaptation dashboards
- Early Testing
- Refinement
Implement Monitors
- Construct monitors
Adaptation Artifacts
Middleware
Technical design
Dashboards
Monitors
Adaptation Stakeholders
Engineers
Maintainers

Table 8.3: *Process Attributes for the Construction Process*

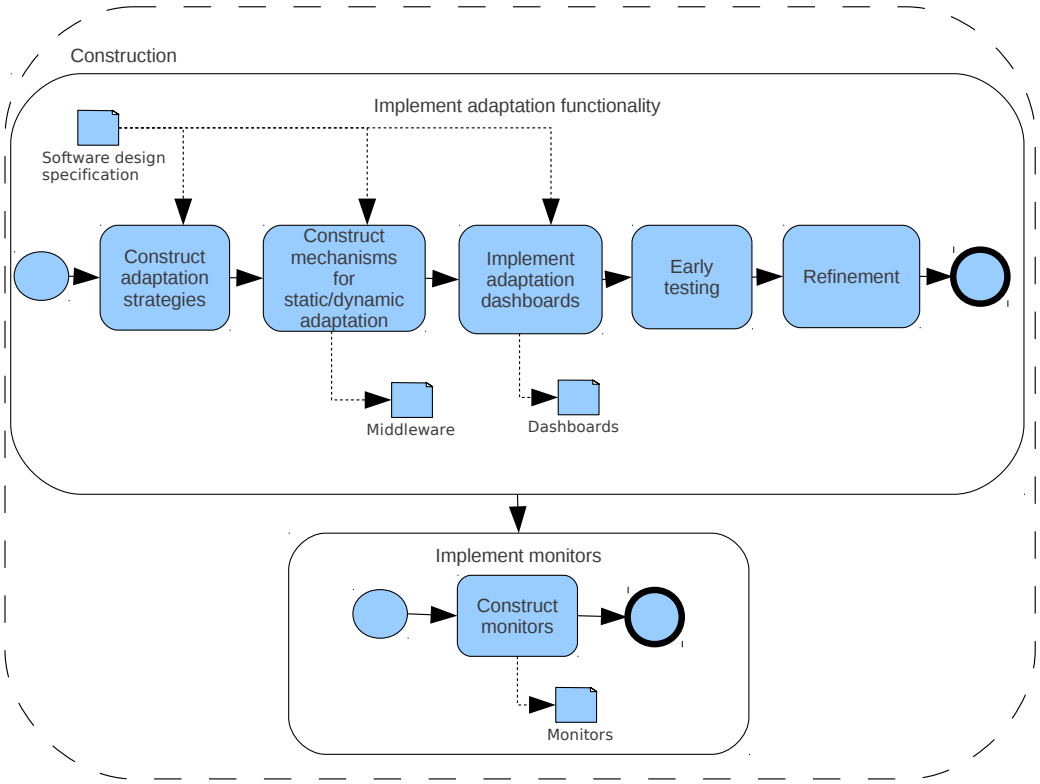

Figure 8.4: *Construction Process Represented in BPMN*

8.2.4 Testing

In SOAdapt, the testing process contains three activities that are used to test the adaptation related components of an SBA.

Functional testing of adaptation configurations This activity is for testing all the possible application configurations of an adaptable SBA. This activity can only be carried out when adaptation configurations are known in advance. Therefore, this activity is not possible in dynamically adapting SBAs where adaptation configurations maybe infinite depending on the operating context of the application.

System integration testing is an activity concerned with the testing of groups of software modules or components. This activity is therefore useful for adaptable SBAs since services are often combined to create novel functionality. Like the functional testing activity, this activity can only occur in statically adapting SBAs or dynamically adapting SBAs with a small number of possible adaptation configurations. If possible, it may be useful to automatically conduct integration testing during runtime when specific adaptation configurations are not known in advance.

User Acceptance testing of adaptation configurations is carried out to ensure user satisfaction with adaptation configurations. When there is a small set of possible adaptation configurations, user acceptance testing can be carried out on all possible configurations. In a scenario with a large or infinite set of configurations, user acceptance testing of each configuration is not possible.

The complete process table is shown in Table 8.4, with its BPMN representation illustrated in Figure 8.5. As can be seen from the diagram, there are still 2 activities that do not have component tasks. They are represented as collapsed activities, in sequence with the “Functional testing of adaptation configurations” activity which has one task.

TESTING
Adaptation Activities & Tasks
Functional testing of adaptation configurations - Prove to customer that application will work in all adaptation cases System integration testing of predicted adaptation configurations User Acceptance testing of adaptation configurations
Adaptation Artifacts
Adaptation Stakeholders

Table 8.4: *Adaptation Process Attributes for the Testing Process*

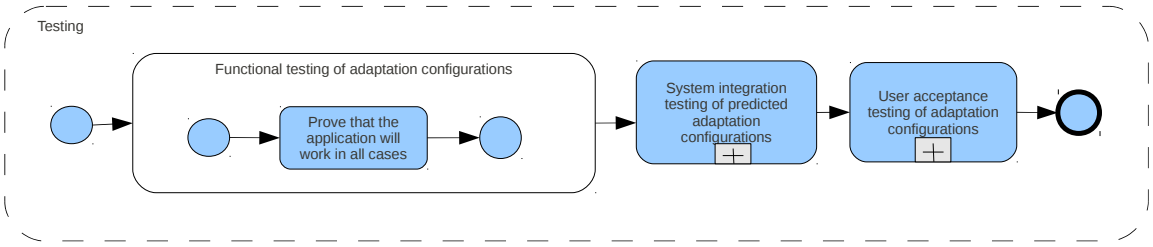

Figure 8.5: *Testing Process Represented in BPMN*

8.2.5 Deployment

This process specifies how the adaptation related components of an SBA are deployed. There is one adaptation specific deployment activity in the SOAdapt model.

Deploy Monitoring and Adaptation Functionality The deployment of monitoring and adaptation functionality is most likely going to occur when the other parts of the SBA get deployed and may not require any special treatment. However, if the monitoring or the adaptation functionality is built in to a service infrastructure which is external to the application, they will have different deployment requirements.

The complete process table is shown in Table 8.5, with its BPMN representation illustrated in Figure 8.6. It shows how the “Adaptation components” are used by the first two tasks in the “Deploy adaptation functionality and monitors” activity.

DEPLOYMENT
Adaptation Activities & Tasks
Deploy Adaptation Functionality and Monitors - Deploy components on separate systems - Monitor System Dependencies - Functional Testing
Adaptation Artifacts
Adaptation components
Adaptation Stakeholders
Developers Maintainers Service Providers Service Consumers

Table 8.5: *Process Attributes for the Deployment Process*

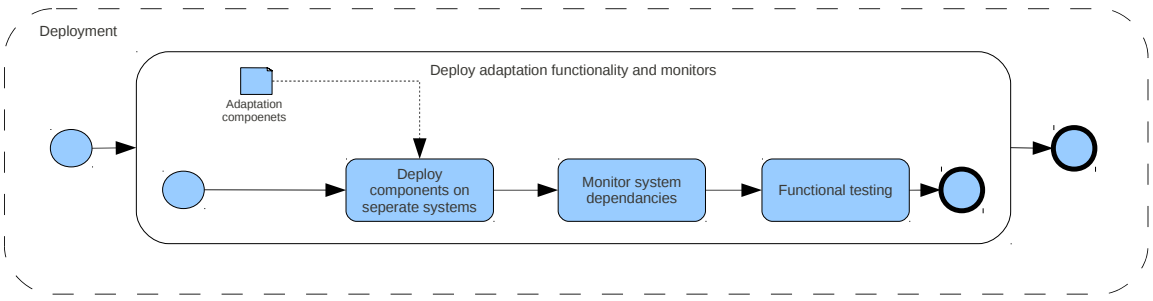

Figure 8.6: *Deployment Process Represented in BPMN*

8.2.6 Operation and Management

The operation and management of the SBA is one of the most critical processes with respect to adaptation. Whether it be manual or automatic, it is during the operation of the application when certain monitored events prompt the adaptation of the application. Within the model, the operation and management process is the link between the evolution cycle which typically occurs during maintenance and the adaptation cycle which allows the application to adapt. As well as providing the link between the adaptation and evolution cycles it also has its own activities.

Run-time monitoring execution Run-time monitoring is a key activity within the operation and management process, it monitors events that signal whether or not run time adaptation should take place. This could be a manual or fully automatic activity depending on the application, but most likely it would be necessary to have some form of involvement by a relevant stakeholder.

Governance The governance activity provides a level of control to the operation of an adaptable SBA. Even where automatic adaptation is technically possible it is often desirable to have some level of governance for tasks such as negotiating service SLAs and authorising application adaptations.

Configuration Management is an activity typically used to manage different configurations during the evolution of applications. However, this activity can also be used in adaptable SBAs to manage and track the different runtime adaptation configurations. This can be particularly useful when troubleshooting runtime errors that occur after the application adapts.

The complete process table is shown in Table 8.6, with its BPMN representation illustrated in Figure 8.7.

OPERATION & MANAGEMENT
Adaptation Activities & Tasks
Run-time monitoring
- Audit service delivery infrastructure
- Monitor services properties against predefined rules
- Monitor non-functional service properties
- Notify users of critical events
- Trigger adaptation from monitored events
Governance
- Manage adaptation processes
- Authorise adaptations where necessary
- Negociate SLAs
Configuration management
- Manage service versions during adaptation
Adaptation Artifacts
State Model
Adaptation Stakeholders

Table 8.6: *Process Attributes for the Operation and Management Process*

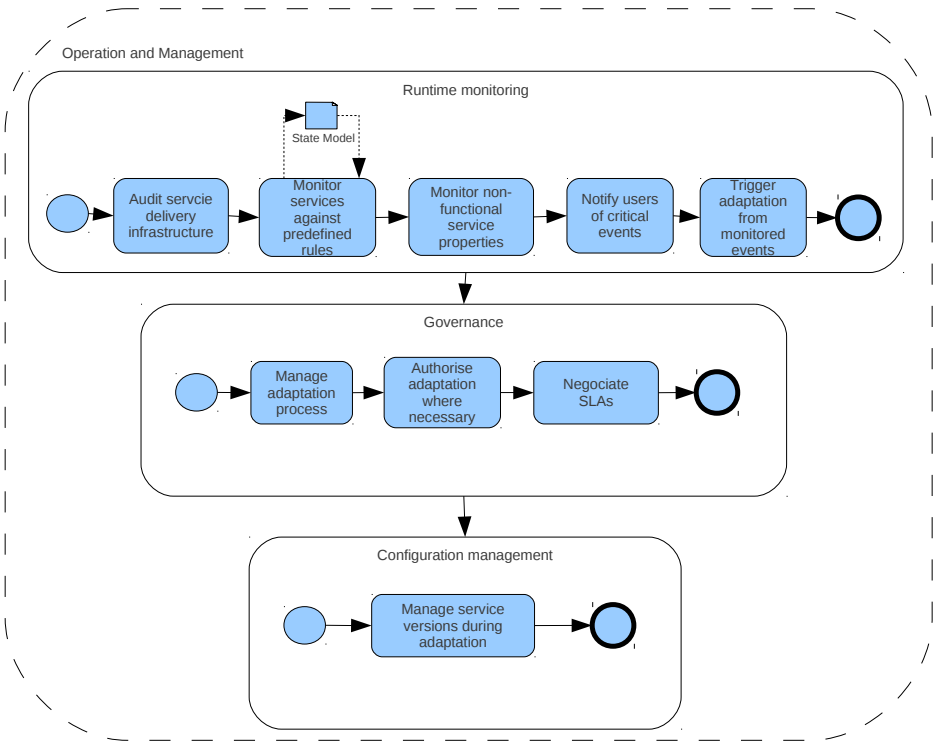

Figure 8.7: *Operation and Management Process Represented in BPMN*

8.2.7 Identify adaptation triggers

Identify adaptation triggers is a process which makes use of monitoring mechanisms in order to identify reasons to adapt. This process occurs during the adaptation cycle, thus all of its sub-activities are related to adaptation. Within SOAdapt there is one activity in this process.

Trigger adaptation from monitored events is an activity that involves the identification of monitored events from application monitors. These events may be reasoned or identified manually depending the application. In a fully automated system, events would be identified based on application logic, however, in a manual scenario, events might be manually identified by end users or developers through inspection of application logs.

The complete process table is shown in Table 8.7, with its BPMN representation illustrated in Figure 8.8. It shows how the workflow branches out to cater for the type of adaptation that is occurring.

IDENTIFY ADAPTATION TRIGGERS
Adaptation Activities & Tasks
Trigger adaptation from monitored events - Application adapts based on built in adaptation logic - Application dynamically adapts based on execution context - Application dynamically adapted by application user - Notify relevant stakeholders
Adaptation Artifacts
Adaptation Stakeholders

Table 8.7: *Process Attributes for the Identify Adaptation Triggers Process*

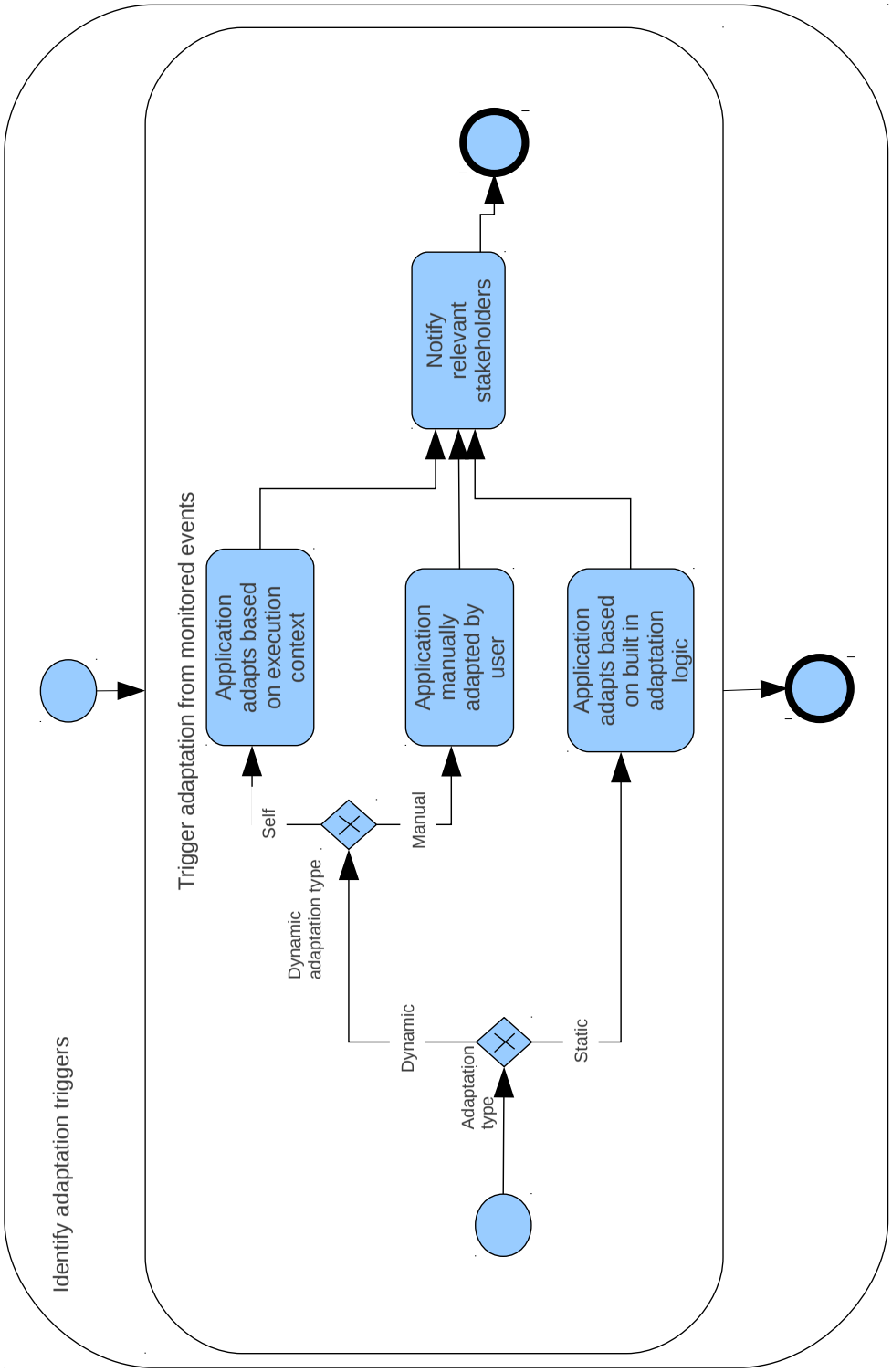

Figure 8.8: *Identify Adaptation Triggers Process Represented in BPMN*

8.2.8 Initiate adaptation

During this process adaptation strategies are selected to suit the monitoring events that were identified in the previous process. Different event types may be associated with different adaptation strategies, and again, this association may be built in to the application's logic or may be made at runtime by a stakeholder. One scenario is that an application has many built in adaptation strategies, then the most appropriate one is chosen by an end user at runtime.

Manage adaptation strategies This activity refers to the management of adaptation strategies in an application where it is possible to modify those strategies at runtime. The ability to introduce new adaptation strategies at runtime would greatly enhance the flexibility of the application.

Application adapts based on static business rules This is an automatic adaptation activity that adapts the application by executing hard coded business rules at runtime.

Automatically adapt to satisfy adaptation need This is an adaptation activity that identifies an optimum adaptation strategy based on the execution context of an adaptable SBA.

Determine impact of adaptation on Information Technology (IT) infrastructure If an SBA needs to adapt due to non-functional adaptation requirements such as performance or availability this often necessitates the provision of additional hardware or other IT resources. This activity addresses those concerns with tasks for assessing the need for additional hardware and co-coordinating with an organisation's IT department.

The complete process table is shown in Table 8.8, with its BPMN representation illustrated in Figure 8.9. This process has separate paths for each of the adaptation types identified in study.

INITIATE ADAPTATION
Adaptation Activities & Tasks
Manage adaptation strategies
- Add new adaptation strategies to the application
Application adapts based on static business rules
Automatically adapt to satisfy adaptation need
- Determine adaptation strategies based on context
Manually adapt SBA with adaptation UI
- Determine adaptation strategy based on recommendation
Determine impact of adaptation on IT infrastructure
- Determine if new hardware is needed
- Co-ordinate with IT department
Adaptation Artifacts
Adaptation Stakeholders
End User

Table 8.8: *Process Attributes for the Select Adaptation Strategies Process*

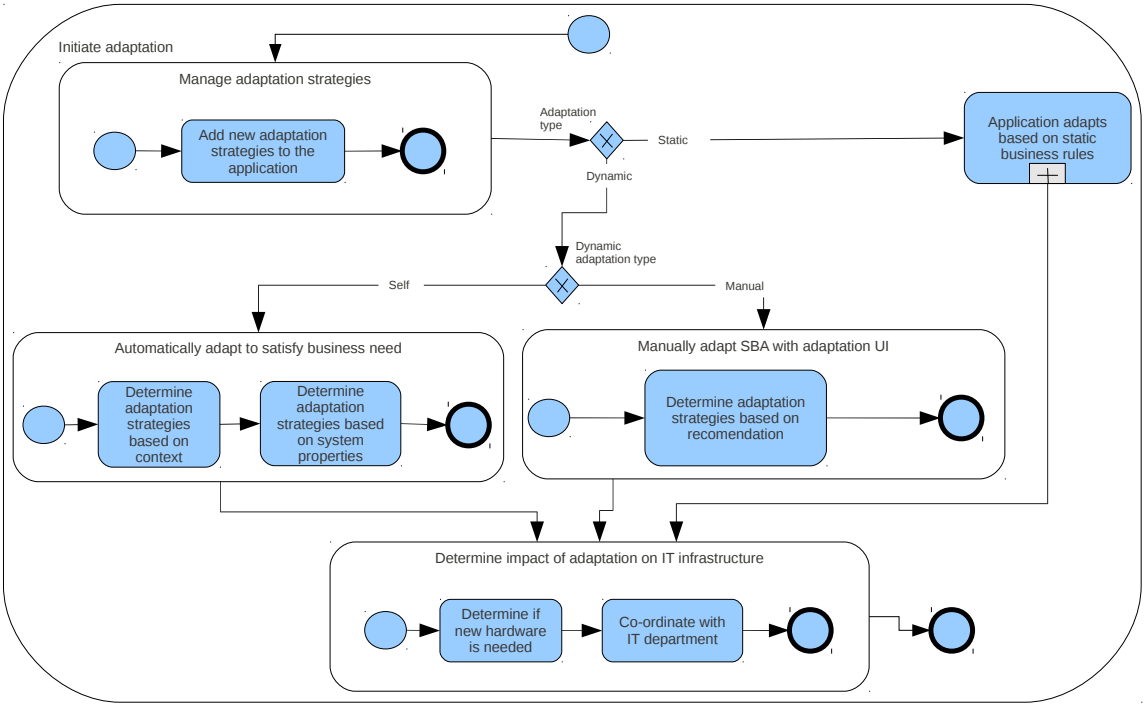

Figure 8.9: *Initiate Adaptation Process Represented in BPMN*

8.2.9 Enact adaptation

Adapt application using User Interface (UI) This task is intended for SBAs that adapt manually during runtime (manual dynamic-adaptation). A user or administrator adapts the SBA by editing its configuration or business rules using an administrative UI.

Automatic adaptation based on context is an automatic adaptation activity that adapts the SBA based on its execution context (dynamic self-adaptation). The exact execution context of an SBA may be impossible to predict in advance so there are potentially infinite contexts that each need specific adaptation strategies.

Automatic adaptation is an automatic adaptation activity that is based on hard coded business rules (static adaptation). This is the most straight forward type of adaptation as each possible adaptation configuration is programmed during the SBA's evolution cycle. During runtime there is often no user input with this type of adaptation. However, it may be desirable to have some level of governance to approve or be notified of a given adaptation.

The complete process table is shown in Table 8.9, with its BPMN representation illustrated in Figure 8.10. As with the two previous processes, enact adaptation has separate activities for each type of adaptation.

ENACT ADAPTATION
Adaptation Activities & Tasks
Adapt application using UI
- Adapt service composition using process editor
- Adapt service end-points using a configuration manager
- Enact adaptation by patching live application
- Enact adaptation in test and development application before production application
Automatic adaptation based on context
- Approve suggested adaptation
Automatic adaptation
- Automatic adaptation based on service-request properties
- Automatic adaptation based on user input
- Automatic adaptation based on business rules
Adaptation Artifacts
Adaptation Stakeholders
End User

Table 8.9: *Process Attributes for the Enact Adaptation Process*

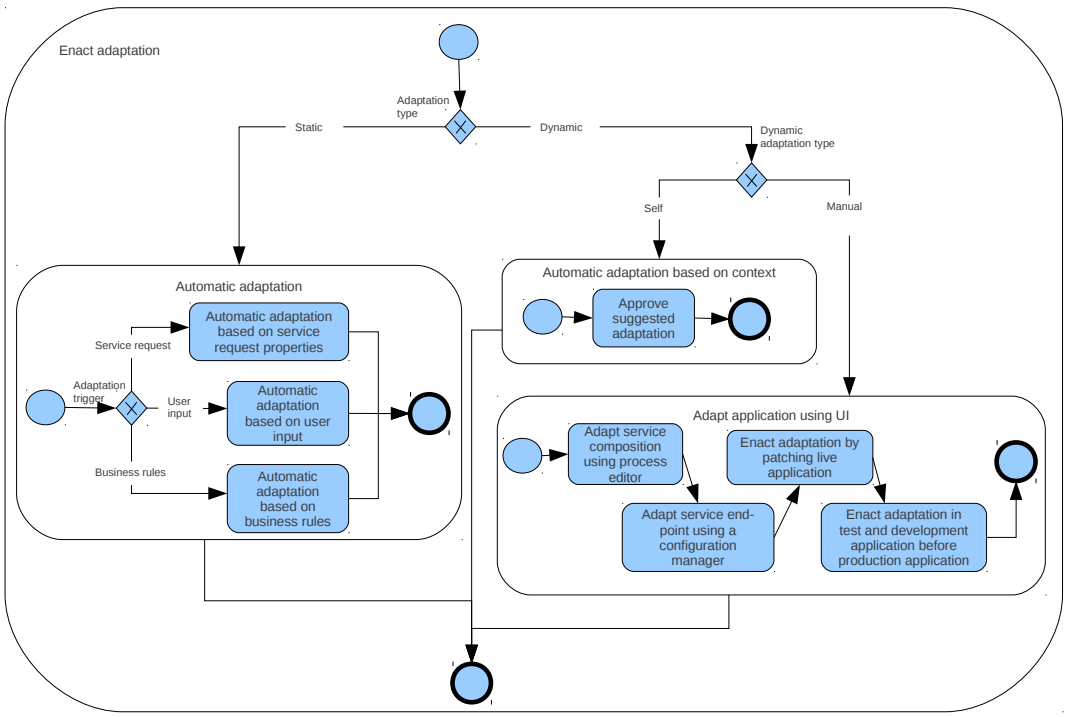

Figure 8.10: *Enact Adaptation Process Represented in BPMN*

8.3 Conclusion

This chapter contains the final completed version of the SOAdapt process reference model. SOAdapt contains nine processes spanning an evolution and adaptation cycle. The three processes in the adaptation cycle are complete and self-contained while the activities within the processes of the evolution cycle need to be used in conjunction with equivalent processes from a life-cycle model. Each process contains a detailed set of adaptation activities, tasks, artifacts and stakeholders.

SOAdapt's process attributes are listed in process attribute tables and its tasks, activities and processes are also represented in BPMN. The BPMN diagrams show the relationships between the attributes that cannot be represented in the attribute tables while the attribute tables provide a quick reference for the contents of each process. The chapter can be used in isolation by researchers or practitioners who are interested in investigating or implementing the model.

Chapter 9

Instantiation of SOAdapt

In this chapter, SOAdapt is applied to the development of a hotel reservation application. The hotel reservation application selected is a SBA which has been prototyped by a group of distributed computing students[1]. This case was chosen because the application's architecture allows for runtime adaptation, the focus of SOAdapt.

9.1 Hotel Reservation Application

The hotel reservation application is a SBA with a broker that identifies the hotel that most closely matches the requirements of the end user. Along with being able to chose a hotel, the end user is also able to pay in his currency of choice. The foreign exchange transactions are managed by a currency broker that consumes foreign exchange services from multiple banks. The high-level architecture of the application is illustrated in Figure 9.1.

The architecture of the application allows hotels and foreign exchange services to easily be added or removed. As it stands, the application is not designed to be adaptable and each of its components services are hard-coded. I will show how this application could be designed, developed and operated following best practice specified by SOAdapt. I will show that, with adaptable SBAs, it is necessary to consider adaptation as a first class concern from the requirements engineering process through to the operation and management process.

[1] http://uwf.edu/nwilde/soaResources/

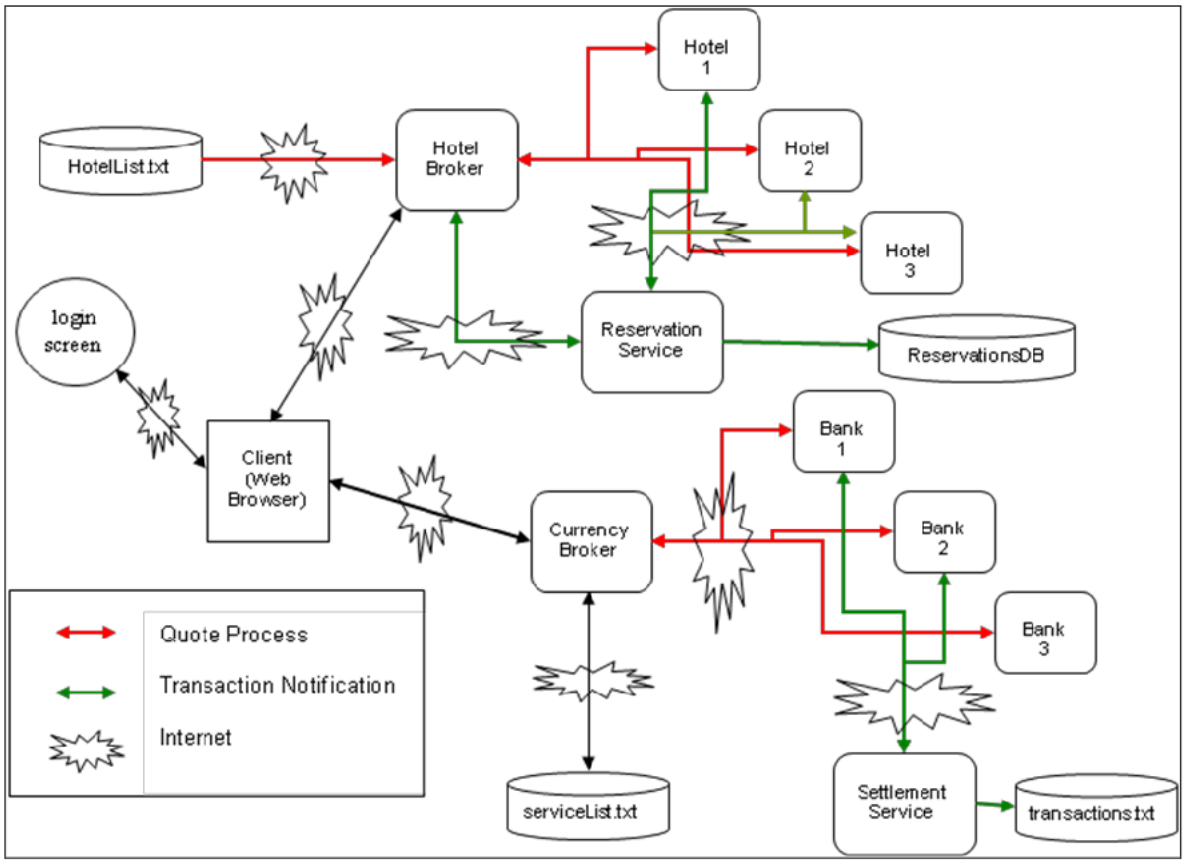

Figure 9.1: *Hotel Reservation Application Architecture*

9.2 Requirements Engineering Elaboration

The SOAdapt requirements engineering process can be applied without modification to the hotel reservation application scenario. In terms of software process, the "Process documentation" artifact from the requirements engineering process description would guide the entire process.

The "Adaptation workflow" artifact would document the choices between the banks and hotels that can be chosen during the execution of the application. The "Change cases" deliverable would describe the adaptation choices from the "Adaptation workflow". For example, a change case would be the selection of a specific hotel if the other hotels are unavailable or more expensive. In the hotel application scenario the business analyst in cooperation with the other stakeholders would document the "Adaptation workflows" and "Changes cases" with guidance from the tasks and activities of SOAdapt's requirements

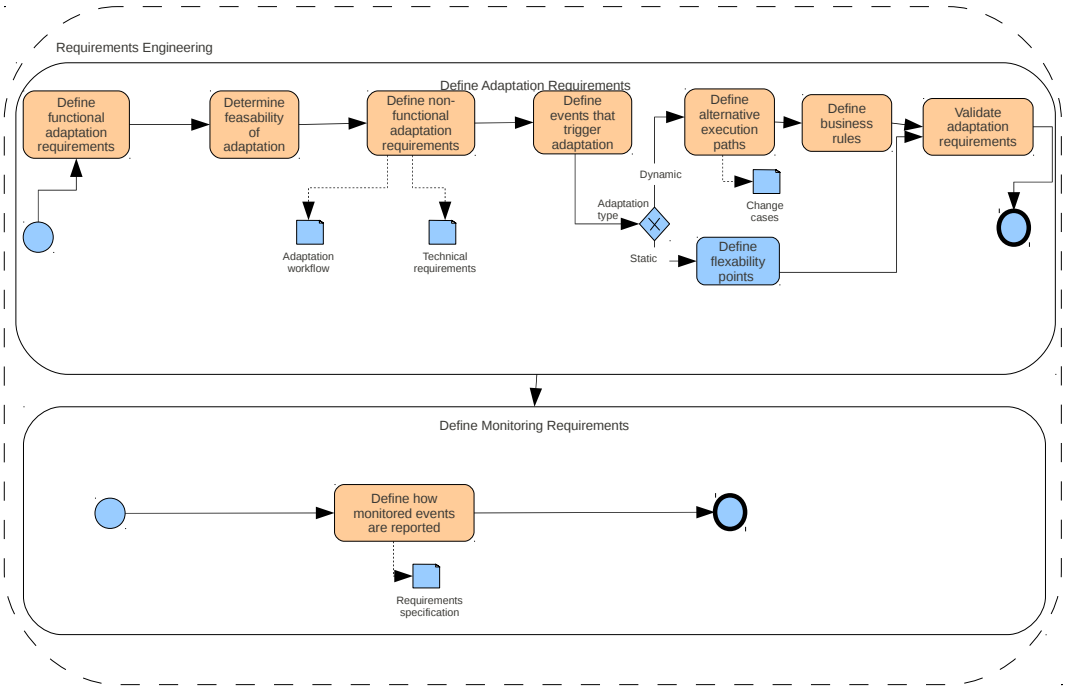

Figure 9.2: *Requirements Engineering Process for the Hotel Reservation Application*

engineering process. These artifacts would be interpreted by the technical stakeholders such as the "Solution architects" and "Engineers" who would provide feedback for the "Technical requirements" of the application.

The requirements engineering process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.2.

9.3 Design Elaboration

SOAdapt specifies each of the activities and tasks that are required to design an adaptable SBA. The design process can be used to guide either dynamic, manual or static adaptation. The hotel reservation application needs to be dynamically adaptable so that the appropriate hotel and bank services can be called during runtime.

Dynamic adaptation would allow runtime characteristics, such as availability and currency rates, to be factored in to adaptation decisions. There are two types of dynamic adaptation catered for in SOAdapt, dynamic manual-adaptation and dynamic self-

adaptation. Since the end users of the application make the final decision as to which currency and hotels they choose, dynamic manual-adaptation is the most appropriate type of adaptation. Once the adaptation type has been chosen the activities and tasks from that workflow are followed.

Once the dynamic manual-adaptation specific activities have been completed, then activities such as prototyping and designing tests, which are common to each type of adaptation, are completed. The monitoring activities in the design process are also common to each type of adaptation.

Some of the artifacts from the design process are particularly interesting to the hotel reservation application. The "Service topology" artifact refers the type of service arrangement required by the SBA. In the case of the hotel reservation application, the service topology consists of a network of services with decision points that facilitate the selection of different hotels and banks during runtime. The service topology is illustrated in Figure 9.1. "Interface definitions" in this type of application are very important because the logic required to consume services with different interface definitions is much more complicated than if the interfaces are uniform. It is, therefore, very important to finalise "Interface definitions" during design.

The Stakeholders specified for the design process are typical of design teams and do not have any specific implications for the hotel reservation application.

The design process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.3.

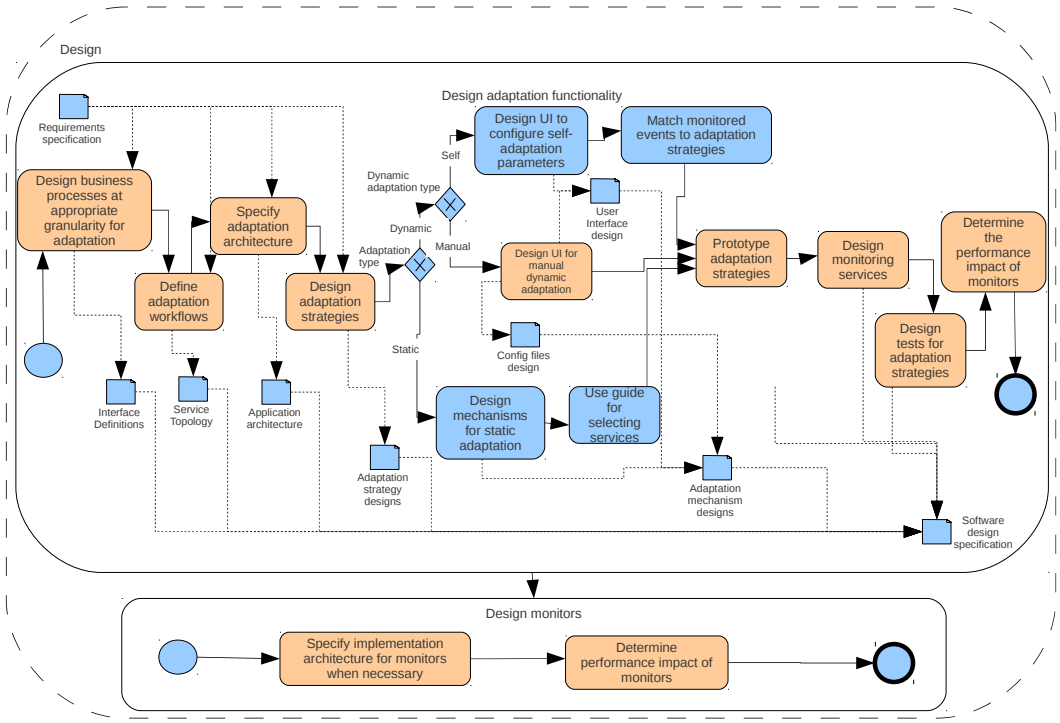

Figure 9.3: *Design Process for the Hotel Reservation Application*

9.4 Construction Elaboration

The construction process describes how the adaptation designs specified in the design process can be implemented. Each of the activities from this process, except for “Implement adaptation dashboards”, applies to the hotel reservation process. Dashboards would only be necessary if there was some adaptation configuration required which was not provided by the end users. The artifacts required for the hotel reservation application would be the adaptation “Middleware” as well as the application “Monitors”. “Engineers” are the primary stakeholders in the hotel reservation application with “Maintainers” familiarising themselves with the application for later maintenance. It is common for the initial development engineers to be the subsequent maintainers.

The construction process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.4.

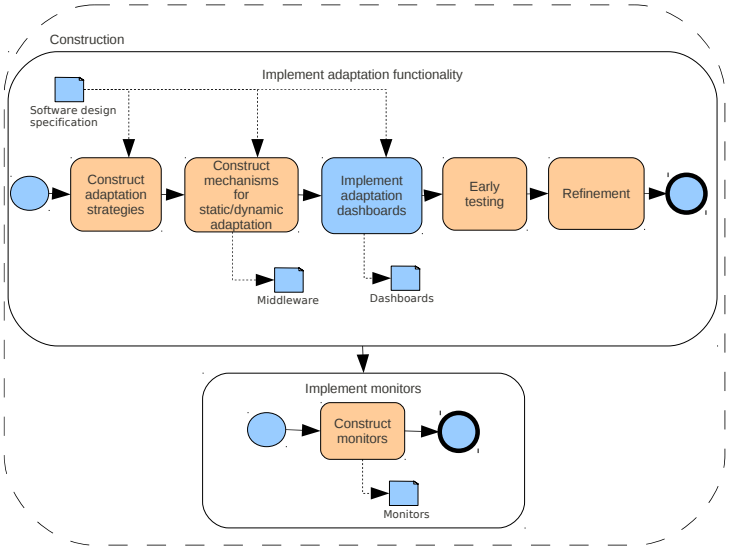

Figure 9.4: *Construction Process for the Hotel Reservation Application*

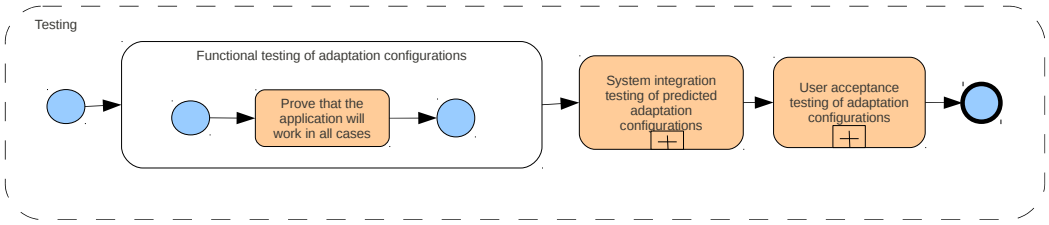

Figure 9.5: *Testing Process for the Hotel Reservation Application*

9.5 Testing and Deployment Elaboration

Like the construction process, the testing and deployment processes have a single workflow regardless of the type of adaptation being implemented. All of the activities from the testing process are applicable to the hotel reservation application. There are no artifacts or stakeholders involved in this process. Perhaps of most importance to the application from testing is the "System integration testing of predicted adaptation configurations". Given the low numbers of services consumed, it is possible to test all of the possible adaptation configurations.

The testing process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.5.

Each of the activities from the deployment process also apply to the hotel reservation

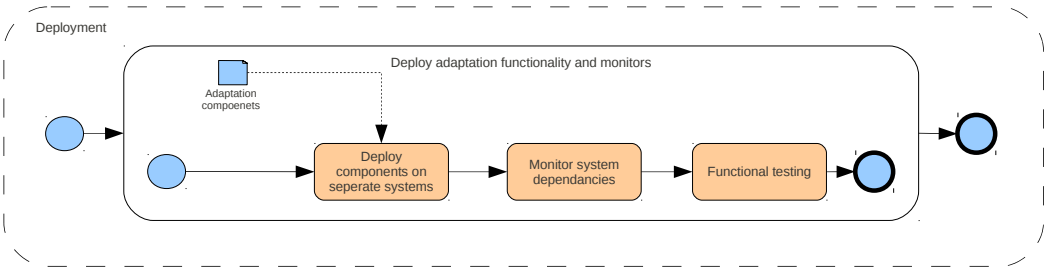

Figure 9.6: *Deployment Process for the Hotel Reservation Application*

application without special consideration. The "Service providers", which are the banks and hotels, may need to be consulted at this stage if there are any issues with consuming their services after deployment. The "Service consumer" stakeholder is not relevant since there are no third party service consumers providing service compositions for the application.

The deployment process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.6.

9.6 Operation and Management Elaboration

The operation and management process does not have workflows that are specific to any of the adaptation types. However, some of the activities are more applicable to particular types of adaptation than others. For example, the activity, "Monitor service against predefined rules" is more applicable to static adaptation and is not very useful to the hotel reservation application. The activity "Trigger adaptation from monitored events" may be applicable to each type of adaptation but is particularly useful for dynamic self-adaptation which is also outside of the scope of the hotel reservation application.

The tasks from the "Governance" activity are not particularly relevant since the users of the reservation application make all of the adaptation decisions without external input. The "Configuration Management" tasks would be useful to keep track of the versions of the services being used by the application. The ability to store service versions and configuration details would allow the application to roll back to known good configurations

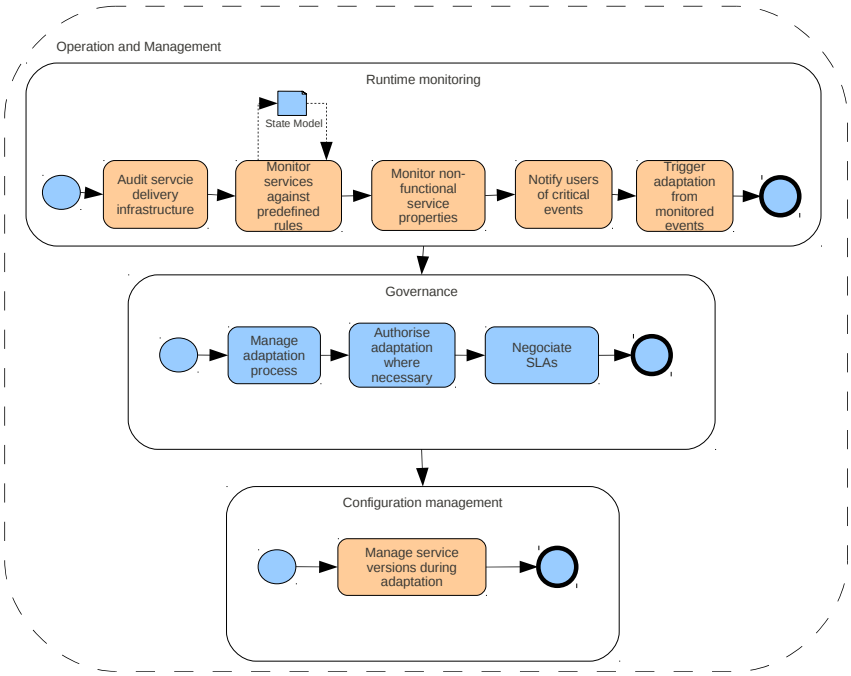

Figure 9.7: *Operation and Management Process for the Hotel Reservation Application*

in the event of unexpected runtime behavior.

The operation and management process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.7.

9.7 Identify Adaptation Triggers Elaboration

The "Identify adaptation triggers" process has three paths depending on the type of adaptation. The path of interest to the hotel reservation application is dynamic manual-adaptation. This path has the single task "Application manually adapted by the user". The common task "Notify relevant stakeholders" is also relevant to the application. The relevant stakeholders in this case would be the hotels where the reservations are made. The banks would not need to be notified of the currency lookup as this lookup is informational and doesn't involve a financial transaction.

The identify adaptation triggers process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.8.

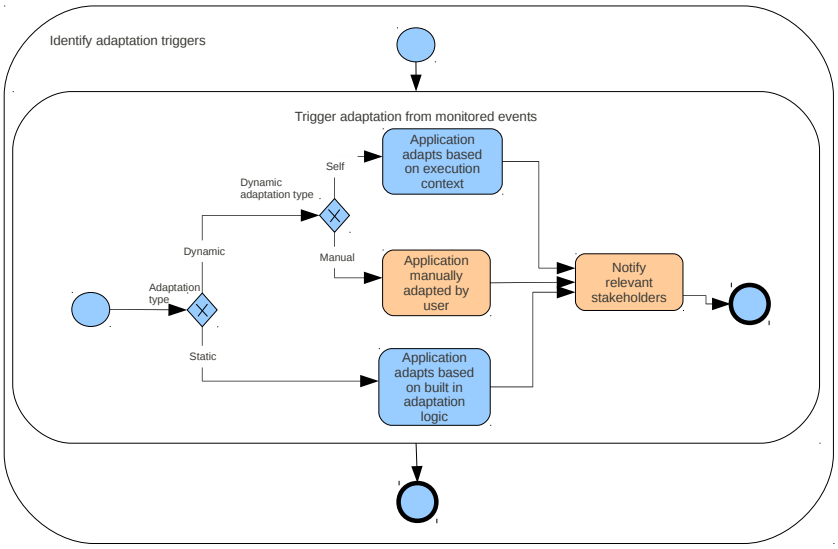

Figure 9.8: *Identify Adaptation Triggers Process for the Hotel Reservation Application*

9.8 Initiate Adaptation Process Elaboration

Following the dynamic manual-adaptation path in the initiate adaptation process there there are four adaptation tasks. Like some of the other process in SOAdapt, some of the tasks from this process are are more applicable to the hotel reservation application than others. For example, the task "Add new adaptation strategies to the application" is more suitable for complex applications with different types of adaptation strategy such as service recomposition or service renegotiation.

Service substitution is the only strategy that is needed in the hotel reservation application. Services are substituted for one another based on simple rules and based on interactions with the end user. A more advanced version of the application might involve service recomposition where adaptation configurations are based on user context information such as location or personal details.

The task "Determine adaptation strategies based on recommendations" from the "Manually adapt SBA with adaptation UI" activity might be realized through a social rating feature. With this feature end users could comment and rate hotels that they have previously used. These comments and ratings would then allow follow on users to make selection decisions and adapt the application according to their preferences.

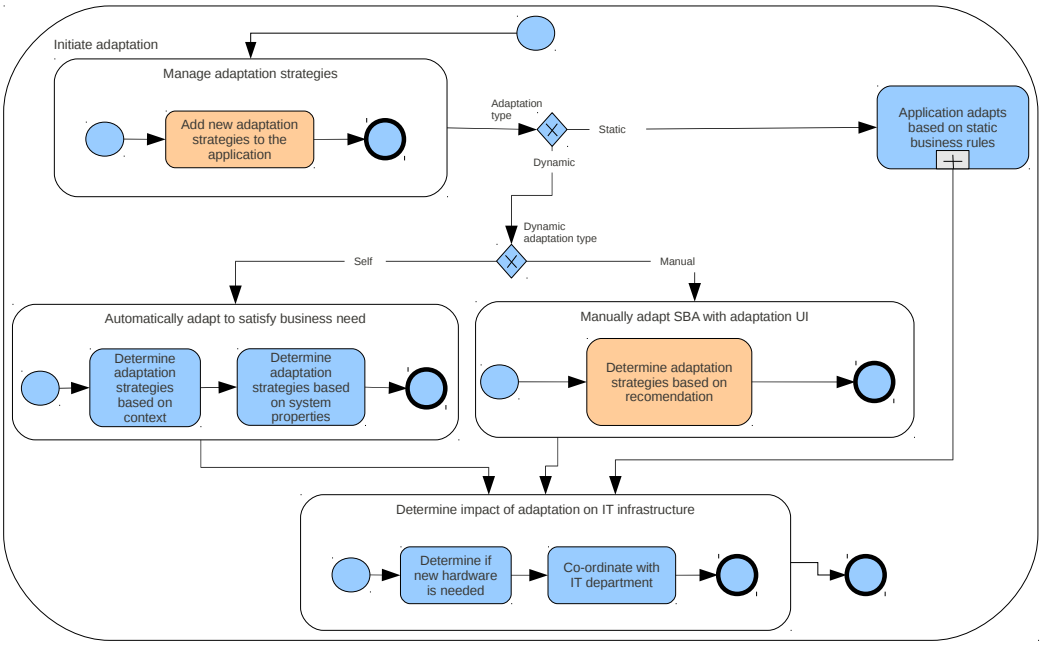

Figure 9.9: *Initiate Adaptation Process Process for the Hotel Reservation Application*

The IT related activities from this process are aimed at adaptable SBAs where adaptation configurations might have a performance or scalability impact on the application. In the case of the hotel reservation application each transaction would involve exactly two service calls so the performance of the application would not be affected by any of the possible configurations. These activities are more concerned with adaptable SBAs that support large adaptable processes.

The initiate adaptation process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.9.

9.9 Enact Adaptation Process Elaboration

Again, the enact adaptation process has three workflow paths, one of which is applicable to the hotel reservation application. There are four tasks in the "Adapt application using UI" activity from the dynamic manual-adaptation path.

The task, "Adapt service compositions using process editor" needs to be adapted to suit the hotel reservation application. A task such as "Adapt service compositions us-

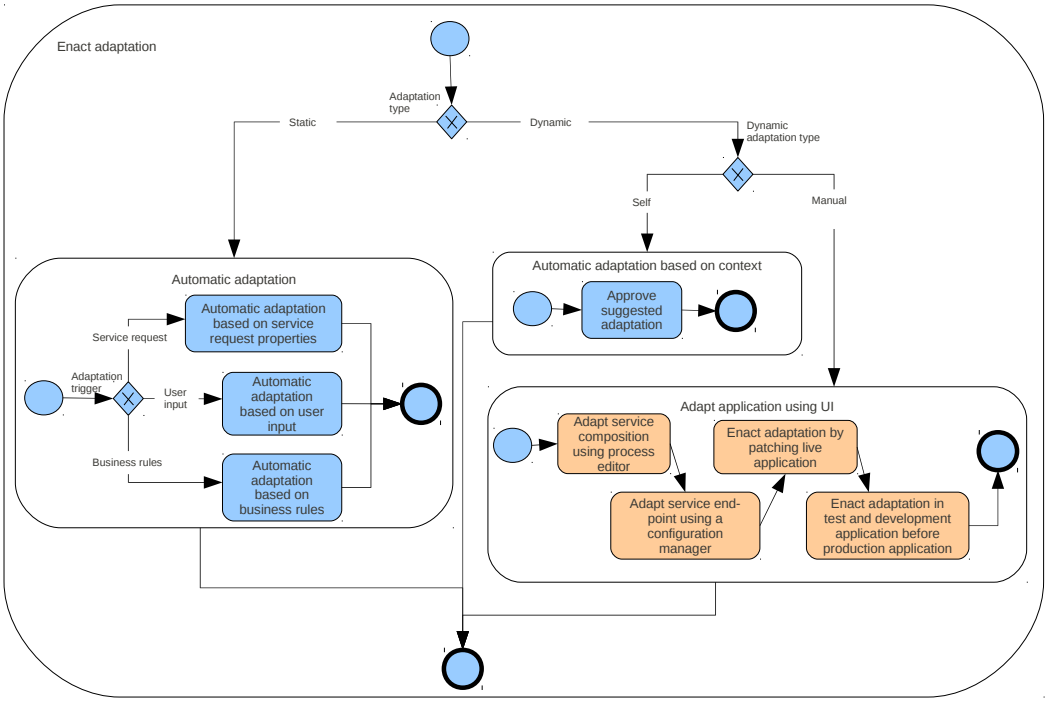

Figure 9.10: *Enact Adaptation Process Process for the Hotel Reservation Application*

ing the application's user interface" would be more applicable. The task "Adapt service end-points using a configuration manager" can be used to add or remove bank and hotel services form the application. The task "Enact adaptation by patching live application" can be used to adapt parts of the application not involved in the booking process. For example, the adaptation mechanisms within the application might be adapted with a live patching task. The task " Enact adaptation in test and development application before production application" relates to testing service configurations on a test system before migrating them to a production system. This task would be beneficial when introducing new services to the application.

The enact adaptation process, with tasks relevant to the hotel reservation application highlighted in orange, is illustrated in Figure 9.10.

9.10 Conclusion

In this chapter, it was shown how SOAdapt could be used to guide the development of an adaptable SBA. The SBA chosen for the demonstration was a hotel reservation application that consumes services from hotels and banks. The hotels' services are used to facilitate reservation bookings while the banks' services enable currency exchange calculations. In this scenario the SBA can adapt to choose different services depending on the requirements of the application's end users.

During the illustration the activities that were relevant to the hotel reservation application were highlighted. In the processes with different workflows, depending on adaptation types, the appropriate workflows were highlighted. In some instances, activities were not relevant to the scenario so they were excluded. This illustration shows a practical demonstration of the model while also demonstrating its external validity.

Chapter 10

Conclusion

10.1 Introduction

In this thesis, SOAdapt, a process reference model for the development of adaptable service-based applications, has been developed during four phases. Phase 1 was a literature study based on a systematic literature review and an exploratory study resulting in SOAdapt-FoR. An initial validation process based on an industrial case study was used to validate the adaptation activities that were used to construct SOAdapt-FoR. Phases 2 and 3 were industrial empirical studies, based on industrial expert opinion and an industrial case study, that resulted in SOAdapt-V1 and SOAdapt-V2 respectively. An inter-model evaluation and an evaluation demonstrating the models transferability was carried out on SOAdapt-V1 before Phase 3 commenced. Phase 4 contained an evaluation process used to validate SOAdapt-V2 through an academic expert opinion evaluation resulting in SOAdapt. Finally, SOAdapt was applied to an adaptable SBA development scenario to demonstrate the use of the model in a real-life application.

10.2 Meeting Research Objectives

The research objectives of this thesis were to develop and validate an empirically grounded process reference model for the development of adaptable SBAs. In order to meet these objectives, I studied the philosophical perspectives and more particularly epistemological perspectives that would suit the development of such a model. The epistemological perspectives considered for this type of research were rationalism and empiricism. Empiricism was chosen as an appropriate perspective for the research due to the nature of

reference model construction. Empiricism suits reference model construction since the best practice contained in the model is more naturally obtained through empirical observation. These empirical observations were elicited from the participants who took part in the research through the use of qualitative data collection and analysis methods. An interpretivist qualitative approach was taken to data collection and analysis while also incorporating positivism into aspects of the data collection and analysis processes where appropriate.

For the second objective, SOAdapt was evaluated using an academic expert opinion based survey with open ended questions. This evaluation method follows the qualitative approach taken for the development of the model. An implementation of SOAdapt was then illustrated to demonstrate the transferability of the model.

10.3 Review of Research Design

To fulfill the first research objective of this thesis, SOAdapt was developed using qualitative data collected during two industrial expert opinion based empirical studies that followed from a comprehensive literature study. There was an interim evaluation between the two empirical studies followed by a more comprehensive survey based evaluation after the second study.

The interview for the first empirical study was based on predefined as well as ad-hoc probing questions. The predefined questions were based on a frame of reference model that was based on the literature. The predefined questions attempted to get the participant's view of the frame of reference and to add more detail to it. The probing questions were used to get an idea of the participants background and experience. They were also used to illicit any information that was relevant to the construction of the model that were not covered by the structured questions. This format worked well, while adding structure to the interviews, it allowed me to probe the participants on areas that might not have been covered. Some of the participants worked better with structured questions while others enjoyed the freedom allowed by open ended ad-hoc questions. The structured

questions were a good method of gleaming information from participants who tended to contribute short answers. The structured questions were also a good way to guide the interview and prevent participants from going too far off topic.

The inter-model evaluation and the evaluation, demonstrating the transferability of the model, were intended to evaluate the external validity of the SOAdapt-V1 before the second study commenced. The external validity was evaluated with respect to generalizability and fitness for purpose. Its purpose was to make sure that SOAdapt-V1 was on the right track and that it did not have to be drastically changed for the second study.

The interview format for the second empirical study was similar to the first study. The same set of structured questions were used for second study as the first study. However, in addition to the structured questions based on the frame of reference, the participants were probed more specifically on areas of the model that had some gaps from the first version. In this study, the participants were asked briefly about the frame of reference while being probed in depth about the adaptation cycle. They were asked about any experience they had relating to runtime adaptation rather then evolutionary adaptation activities.

The evaluation that took place after the second study attempted to gage the external validity of SOAdapt's process attributes, fulfilling the second objective of this thesis. Two question types were used to gage the external validity of each artifact, example and justification type questions. The challenge with composing these types of questions is to phrase them so that participants don't feel like they are repeatedly being asked the same type of question. The instantiation of SOAdapt to the hotel reservation application scenario further demonstrated the external validity of the model, particularly its transferability.

One benefit of completing the study in two stages was that lessons learned from the first study could be applied to the second study. Another benefit was that the first version of the model could be reviewed for gaps allowing the focus of the second study to address these gaps.

10.4 Validation

Lincoln and Guba [69] argue that qualitative research should be internally valid, externally valid, reliable and objective. Internal validity relates to "how" the research is carried out, and whether the methods used are credible. Two suggested methods of ensuring internal validity are data source and method triangulation [68]. External validity refers to the transferability of the research results to other similar contexts. If research results are transferable then they are arguably much more useful than results that are specific to a particular context.

The internal validity of this research has been sought through the use of data source and research method triangulation. There were several data sources used to construct the process model, SOAdapt-FoR was constructed with data from relevant literature sources documenting various adaptation approaches, while the empirically grounded versions of the model were constructed using data gathered from an industrial case study and input from industrial experts. SOAdapt-V2 was evaluated by a panel of academic experts adding another data source to the construction of the model. There were also two research methods employed, survey research where individuals were interviewed and case study research comprising two case studies. The first case study, in SBA Solutions, was used to validate the adaptation activities identified in Phase 1 while the second case study in SbaSoft provided input for SOAdapt-V1 in Phase 2.

The primary threat to the external validity of this research is the generalisability of the model constructed. Every effort was made to select interview partners from a diverse set of companies and a diverse set of roles, but this does not guarantee that the results are generally applicable. Both the interim evaluation of SOAdapt-V1 and the survey based evaluation of SOAdapt-V2 were used in order to mitigate these risks. SOAdapt-V1 was mapped to a SBA development life-cycle that was developed independently of this research. The goodness of fit that was achieved during the mapping helps to show that the model can be generally applied. The evaluation of SOAdapt-V2 was also used to determine whether its processes' attributes could be used in general SBA adaptation

contexts.

In Chapter 9, it was shown how SOAdapt could be used to guide the development of an adaptable SBA. In this scenario each of the models attributes were evaluated to see how they could be implemented. The application used in the scenario was a dynamic manually-adapting hotel reservation application. Some of the attributes from the model were non-applicable since they are intended for static or dynamic self-adaptation. This practical scenario demonstrated the usefulness of the model for developing a real-world adaptable SBA.

I strived for reliability and objectiveness during all stages of the research by having each phase audited by my supervisor. For example, during the systematic literature review my supervisor audited the study selection process by repeating some of the decisions independently. This was to make sure that the selection process was reliable and objective. Similarly, the coding and analysis processes were audited for repeatability showing that they were reliable and objective.

10.4.1 Threats to Validity

A remaining threat to the validity of the research is whether or not the data collected reached a point of saturation. Saturation point for this study would be reached when subsequent field work identifies no new process attributes. The later interviews did repeat much of the earlier findings with the exception of the life-cycle processes in the adaptation cycle, indicating that the study had neared or reached saturation point.

10.5 Meeting the Challenges to developing adaptable SBAs

In section 2.7, I discussed some of the challenges for developing adaptable SBAs that were identified in the literature. The ability of SOAdapt to overcome some of these challenges shows the maturity and capability of the model. In this section, SOAdapt is reviewed with respect to each of these challenges. Table 10.1 summarizes the challenges that SOAdapt specifically addresses.

Runtime Adaptation The runtime adaptation challenges such as the ability to facilitate fast, reliable and automatic adaptation were the main focus of this thesis. SOAdapt facilitates runtime adaptation through the specification of design and runtime processes and activities that are necessary for the development of adaptable SBAs.

Development Process The development process challenges related to the provision of a non-fragmented method for developing adaptable SBAs. SOAdapt addresses these challenges by specifying a coherent development process for guiding all aspects of adaptable SBA development and operation.

Monitoring The main challenges relating to monitoring suggested that monitoring implementations needed to be efficient, non-fragmented and easy to diagnose. These monitoring challenges are implementation specific and outside the scope of a generic reference model. However, following an efficient reference process such as SOAdapt should enhance the quality of all implemented components. This is achieved by eliminating unnecessary development tasks and making sure that development tasks are completed in the correct sequence.

Adaptation Tools The challenges in the adaptation tools category were that service composition tools are too complex and adaptation tools are too fragmented. These are ongoing challenges with SOA development in general. SOAdapt attempts to guide the use of tools where appropriate but does not contain or endorse any specific adaptation tools. The SOAdapt frame of reference is based on the S-Cube reference life-cycle which has many advanced adaptation tools associated with its processes.

Service Terminology In an attempt to address the challenge of disjointed service terminology the literature review carried out in this thesis adapted a common service nomenclature allowing approaches with different terminologies to be compared. This nomenclature was then carried forward into the empirical studies and the development of SOAdapt.

Service Adaptability The challenge that services need to be adaptable to be useful to

many service consumers is not addressed by SOAdapt. SOAdapt is intended for the development of SBAs which are a type of service consumer. Since SOAdapt is designed from a service consumer perspective it does not attempt to guide the development of adaptable services. SOAdapt seeks to find services that are suitable without adaptation rather than address the challenge of service adaptability.

Service Discovery There were two service discovery challenges identified in relation to developing adaptable SBAs. The first was that runtime service discovery is difficult and the second was that it is difficult to match service clients to providers. In the case of completely automatic adaptation, the implementation details for automatic service discovery or service negotiation mechanisms are outside the scope of a process model. In the case of manual adaptation SOAdapt suggests the use of an adaptation dashboard to manage adaptation strategies. This would include the discovery of new services for inclusion in an adaptation strategy.

Enterprise Integration the enterprise integration challenges refer to the ease that an adaptable SBA can integrate with other applications within an organisation or enterprise. The organisations' processes also need to be at a level of granularity where they can be represented by software processes that can be composed into adaptable compositions. SOAdapt handles these challenges in the design process with the task "Design business processes at appropriate granularity for adaptation".

Service Quality A major problem with SBAs is that services are not always under the control of the application owners. Therefore, to solve the problem of unreliable services, SBAs can adapt and choose more reliable alternatives. SOAdapt facilitates this adaptation since it is a reference model for developing adaptable SBAs.

Service Semantics The challenges in the service semantics category are concerned with services being annotated with semantic descriptions that allow them to be computer readable. This facilitates the automatic discovery and negotiation that occurs during automatic adaptation. SOAdapt describes these types of functionality as "adapta-

tion mechanisms" and specifies how to design and implement these mechanisms at a high level. Providing implementation specific tasks to deal with these types of mechanisms, risks reducing the generalizability of SOAdapt.

Service Specifications The single challenge for service specifications was that " Services need to be at the same granularity as business processes ". SOAdapt directly caters for this challenge with the design task: " Design business processes at appropriate granularity for adaptation".

Domain Knowledge In order to integrate domain knowledge with adaptable SBAs, SOAdapt has tasks such as "Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS Based)" and "Define business rules for adaptation". During the empirical studies several participants suggested the use of domain specific languages to capture domain knowledge. However, this task was rolled up into the tasks mentioned as it was deemed too low level for a generic process.

Service Testing is problematic for adaptable applications because once the applications adapt, previous tests become obsolete. SOAdapt contains a testing process containing integration and user acceptance testing activities. During the evaluation of SOadapt-V2 one of the participants suggested integration testing of service compositions during the execution process before adaptation takes place. This recommendation has been added to the final version of SOAdapt.

10.6 Contributions

The first contribution of the research is a state of the art literature review. This review covers the areas of software process, SOC and software adaptation. Within the SOC section a systematic literature review was carried out to identify software process models for developing SBAs. The models identified in systematic literature review were analysed in order to identify the ones that support the adaptation of SBAs. This literature review and

Challenge	Addressed by SOAdapt
Runtime Adaptation	Yes
Development Process	Yes
Monitoring	No
Adaptation Tools	Partial
Service Terminology	Yes
Service Adaptability	No
Service Discovery	Partial
Enterprise Integration	Yes
Service Quality	Yes
Service Semantics	Partial
Service Specifications	Yes
Domain Knowledge	Yes
Service Testing	Yes

Table 10.1: *Challenges addressed by SOAdapt*

contained systematic review provide a useful starting for further research into adaptable SBAs.

The main deliverable of the research is SOAdapt, this process reference model provides an important consolidation of best practices form the literature as well as empirical sources. The model is based around a high-level life-cycle allowing it to be mapped to other life-cycles with similar processes. The validity of the model has been demonstrated with respect to both internal and external validity. For academia, the reference model can be used as a framework to which new techniques and tools can be added. For Industry, this reference model can be used to guide their development to ensure that they have an optimum process when developing adaptable SBAs.

Finally, the thesis addresses the challenges to the development of adaptable SBAs that were identified at the beginning of the thesis.

10.7 Future Work

The next stage in SOAdapt's development would be an industrial implementation to evaluate the effectiveness of the model is a real industrial context. The implementation would likely identify any short comings that the model may have that were not previously identified. A documented implementation would also give confidence to other early adopters increasing the uptake of the model.

I encountered several companies during this research that would be suitable for an implementation of the model. Two in particular stood out, Companies H and I. I conducted a case study in Company H for Industrial Empirical Study (1). This was a relatively small company which would make it easier to implement a new process. The product that they deliver facilitates static adaptation which is one of the adaptation types supported by SOAdapt. Company H on the other hand is larger than Company I meaning an implementation would be more challenging. The reason that it would be a suitable target for the implementation of SOAdapt is because its main product encompasses advanced dynamic adaptation. This would provide a more thorough evaluation of the model.

Another research path which could be taken would be to analyse the compatibility of SOAdapt with industry specific standards. For example, if developing software for the automotive or health care domains, customers may insist compliance with development standards specific to their domains. If SOAdapt is to be implemented in organizations where this is the case, future work will have to include its evaluation for standards compatibility.

10.8 Conclusion

In this thesis I have presented SOAdapt, a model that was constructed and evaluated from the study of literature and industrial practice. The model has been assessed using literature based evaluations as well as an interview based expert opinion survey. It is ready for implementation in industry to support organisations in their efforts to develop adaptable SBAs. This model gives practitioners the benefit of SBA adaptation best practices, elicited from practitioners who have already invested in developing their own adaptation processes.

The primary barrier to adaption of SOAdapt by industry is the inherent inertia within companies to adopting new practices. During the field work that was carried out for this research, the experts that I spoke implied that companies needed a lot of persuasion to change their existing processes. This is especially true when they believe that they

existing processes are sufficient. The challenge is to demonstrate to practitioners that adaptable SBAs are distinct from more traditional application types. It also needs to be highlighted that adaptable SBAs need to be developed with a process model that takes the subtleties of these types of applications into consideration.

In Section 2.3, I discussed three attributes that Watts and Kellner [111] suggest an effective process model should have. Effective process model should be effective in communicating a software process, reusable given the cost and time associated with their development, and evolveable since real software processes inevitably evolve. I argue that SOAdapt possesses all of these qualities. Its process attribute tables and BPMN representations ensure that it is effective in communicating its software process. Its generalisability, achieved through the use of multiple data sources and evaluation processes, allows the model to be reused by practitioners developing different types of adaptable SBAs. Finally, through the iterative model development process presented this thesis, I have shown how the model can be easily evolved to incorporate new information.

References

[1] NASA LaRC formal methods program: What is formal methods? http://shemesh.larc.nasa.gov/fm/fm-what.html.

[2] S-Cube knowledge model. http://www.s-cube-network.eu/km, 2010.

[3] D.X. Adamopoulos. An integrated development environment for synthesising web services. In *International Conference on e-Business Engineering (ICEBE 2008)*, pages 776–779, 2008.

[4] Frederik Ahlemann and Heike Gastl. Process model for an empiracally grounded reference model construction. In Peter Fettke and Peter Loos, editors, *Reference Modeling for Business Systems Analysis*. Idea Group Publishing, 2007.

[5] Vasilios Andrikopoulos. Separate design knowledge models for software engineering and service based computing. Deliverable CD-JRA-1.1.2, S-Cube Consortium, May 2009.

[6] A. Arsanjani and A. Allam. Service-Oriented modeling and architecture for realization of an SOA. In *IEEE International Conference on Services Computing*, pages 521–521, 2006.

[7] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley. SOMA: a method for developing service-oriented solutions. *IBM Systems Journal*, 47:377–396, 2008. 3.

[8] ATOS. SeCSE methodology, version 3. Technical report, March 2007.

[9] Lisa Bahler, Francesco Caruso, and Josephine Micallef. A practical method and tool for systems engineering of Service-Oriented applications. In *The 8th International Conference on Web Information Systems Engineering*, volume 4831 of *Lecture Notes in Computer Science*, pages 472–483. Springer, 2007.

[10] Kent Beck. *Extreme Programming Explained: embrace change*. Addison-Wesley Professional, 2001.

[11] S Beecham, T Hall, C Britton, M Cottee, and A Rainer. Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software*, 76:251–275, June 2005.

[12] Salima Benbernou. State of the art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of SBAs. Deliverable PO-JRA-1.2.1, S-Cube Consortium, July 2008.

[13] Avivit Bercovici, Fabiana Fournier, and Alan Wecker. From business architecture to SOA realization using MDD. In *Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Applications*, pages 381–392, 2008.

[14] Veli Bicer, Steffen Lamparter, York Sure, and Ali H. Dogru. Towards an interdisciplinary methodology for service-oriented system engineering. In *24th International Symposium on Computer and Information Sciences*, pages 486–491, 2009.

[15] B. W. Boehm. Software engineering. *IEEE Transactions on Computers*, 100(25):1226–1241, 1976.

[16] B. W Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, May 1988.

[17] Khouloud Boukadi, Lucien Vincent, and Patrick Burlat. Modeling adaptable business service for enterprise collaboration. In *IFIP WG 5.5 Working Conference on Virtual Enterprises (PRO-VE 2009)*, pages 51–60, Thessaloniki, Greece, 2009. Springer.

[18] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, and V. Mazza. A context-driven adaptation process for service-based applications. In *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems*, page 50–56, 2010.

[19] Antonio Bucchiarone, Cinzia Cappiello, Elisabetta di Nitto, Raman Kazhamiakin, Valentina Mazza, and Marco Pistore. Design for adaptation of Service-Based applications: Main issues and requirements. Stockholm, Sweden, 2009.

[20] C. Bussler, D. Fensel, M. E. Orlowska, J. Yang, Z. H. Ren, B. H. Jin, and J. Li. A new web application development methodology: Web service composition. In *International Workshop on Web Services, E-Business, and the Semantic Web (WES 2003)*, volume 3095 of *Lecture Notes In Computer Science*, pages 134–145, 2004.

[21] I. Christou, S. Ponis, and E. Palaiologou. Experiences of using the agile unified process in the banking sector. *Software, IEEE*, PP:1–1, 2009. 99.

[22] A. Cockburn. *Agile Software Development*. 2000.

[23] J. Cordeiro, J. Filipe, G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J. P. Richter, M. Voss, and J. Willkomm. A method for engineering a true service-oriented architecture. In *International Conference on Enterprise Information Systems (ICEIS 2008)*, volume 2, pages 272–281, Barcelona, Spain, 2008.

[24] E. G Cuba and Y. S Lincoln. Competing paradigms in qualitative research. In N. Denzin, editor, *Handbook of qualitative research*, pages 105–117. Sage Publications, Inc, Thousand Oaks, CA, 1994.

[25] A. Dan, W. Lamersdorf, Simon K. Johnson, and Alan W. Brown. A model-driven development approach to creating service-oriented solutions. In *International Conference on Service-Oriented Computing (ICSOC 2006)*, volume 4294 of *Lecture Notes In Computer Science*, pages 624–636. Springer, 2006.

[26] Nelly A. Delessy and Eduardo B. Fernandez. A pattern-driven security process for SOA applications. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2226–2227, Fortaleza, Ceara, Brazil, 2008. ACM.

[27] J. C Derniame, B. A Kaba, and D. Wastell. *Software process: principles, methodology, and technology*. Springer Verlag, 1999.

[28] Straub Detmar, David Gefen, Marie-Claude Boudreau, and Dennis Galletta. The ISWorld quantitative, positivist research methods website. http://dstraub.cis.gsu.edu:88/quant/, 2004.

[29] Straub Detmar W., David Gefen, and Marie-Claude Boudreau. Quantitative research. In D. Avison and J. Pries-Heje, editors, *Research in Information Systems: A Handbook for Research Supervisors and Their Students*, pages 221–238. Butterworth-Heinemann, 2005.

[30] M. Deubler, J. Grunbauer, G. Popp, G. Wimmel, and C. Salzmann. Tool supported development of service-based systems. In *11th Asia-Pacific Software Engineering Conference*, pages 99–108, Busan, Korea, 2004. IEEE Computer Society.

[31] Elisabetta Di Nitto, Massimiliano Di Penta, Alessio Gambi, Gianluca Ripa, and Maria Villani. Negotiation of service level agreements: An architecture and a Search-Based approach. In *Service-Oriented Computing – ICSOC 2007*, volume 4749 of *Lecture Notes in Computer Science*, pages 295–306. Springer, 2009.

[32] S. Durvasula, M. Guttmann, A. Kumar, J. Lamb, T. Mitchel, B. Oral, Y. Pai, T. Sedlack, H. Sharma, and S.R. Sundaresan. Introduction to service lifecycle. SOA practitioner's guide. part 3, 2007.

[33] S. Easterbrook, J. Singer, M. A Storey, and D. Damian. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering*, page 285–311, 2008.

[34] A. C Edmondson and S. E McManus. Methodological fit in management field research. *Academy of Management Review*, 32(4):1155, 2007.

[35] G. Engels and M. Assmann. Service-Oriented enterprise architectures: Evolution of concepts and methods. In *12th International IEEE Enterprise Distributed Object Computing Conference*, pages xxxiv–xliii. IEEE Computer Society, 2008.

[36] Ru Fang, Ying Chen, Liana Fong, Linh Lam, David Frank, Christopher Vignola, and Nan Du. A version-aware approach for web service client application. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management, Vols 1 and 2*, pages 401–409, 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, GERMANY, MAY 21-25, 2007.

[37] F. Giunchiglia, J. Mylopoulos, and A. Perini. The tropos software development methodology: processes, models and diagrams. In *Agent-Oriented Software Engineering*, LNCS 2585, pages 162–173. Springer Verlag, 2002.

[38] B. G Glaser and A. L Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company, Chicago, IL, USA, 1968.

[39] I. Griffin and I. Richardson. Using LATEX for qualitative data analysis. *The PracTEX Journal*, (2010-1), 2010.

[40] M. H. Hamza, M. Deubler, J. Grunbauer, G. Popp, G. Wimmel, and C. Salzmann. Towards a model-based and incremental development process for service-based systems. In *Proceedings of the IASTED International Conference on Software Engineering*, pages 183–188, 2004.

[41] Paul Harmon. Second generation business process methodologies. *Business Process Trends*, 1(5), 2003.

[42] Peter Herzum and Oliver Sims. *Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*. John Wiley & Sons, Inc., 2000.

[43] J. Hielscher, A. Metzger, and R. Kazhamiakin. Taxonomy of adaptation principles and mechanisms. Technical Report PO-JRA-1.2.2, S-Cube Consortium, 2009.

[44] R. Howard and L. Kerschberg. A framework for dynamic semantic web services management. *International Journal of Cooperative Information Systems*, 13(4):441–485, 2004.

[45] D. Howcroft and B. Fitzgerald. Towards dissolution of the IS research debate: from polarization to polarity. *Journal of Information Technology*, 13:313–326, 1998.

[46] IEEE, IEC, and ISO. ISO/IEC 15288, systems and software engineering - system life cycle processes. Technical report, 2008.

[47] IEEE, ISO, and IEC. *IEEE Standard for Software Maintenance*. Institute of Electrical and Electronics Engineers, New York NY, 1998.

[48] IEEE, ISO, and IEC. *Software engineering–software life cycle processes–maintenance International standard.* Institute of Electrical and Electronics Engineers, New York, NY, 2006.

[49] ISO/IEC. *Systems and software engineering - Software life cycle processes*. Geneva, 2008.

[50] Shankar Kambhampaty. Service oriented analysis and design process for the enterprise. In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science*, pages 366–371, Venice, Italy, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[51] Marcel Karam, Sergiu Dascalu, Haidar Safa, Rami Santina, and Zeina Koteich. A product-line architecture for web service-based visual composition of web applications. *Journal of Systems and Software*, 81:855–867, 2008. 6.

[52] R. Kaschek, C. Kop, C. Steinberger, G. Fliedl, Thomas Karle, and Andreas Oberweis. Collaborative model driven software development for SOA-based systems. In *International United Information Systems Conference (UNISCON 2008)*, volume 5 of *Lecture Notes in Business Information Processing*, pages 189–200, Klagenfurt, Austria, 2008.

[53] A. Kenzi, B. El Asri, M. Nassar, and A. Kriouile. A model driven framework for multiview service oriented system development. In *International Conference on Computer Systems and Applications*, pages 404–411. IEEE Computer Society, 2009.

[54] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University, 2007.

[55] Barbara Kitchenham, Shari Lawrence Pfleeger, Beth McColl, and Suzanne Eagan. An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64(1):57–77, October 2002.

[56] Barbara A. Kitchenham, Shari L. Pfleeger, Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. Personal opinion surveys. In *Guide to advanced empirical software engineering*. Springer, October 2007.

[57] Heinz K. Klein and Michael D. Myers. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1):67–93, March 1999. ArticleType: research-article / Full publication date: Mar., 1999 / Copyright © 1999 Management Information Systems Research Center, University of Minnesota.

[58] B. J. Kramer, K. J. Lin, P. Narasimhan, M. Autili, L. Berardinelli, V. Cortellessa, A. Di Marco, D. Di Ruscio, P. Inverardi, and M. Tivoli. A development process for self-adapting service oriented applications. In *International Conference on Service-Oriented Computing (ICSOC 2007)*, volume 4749 of *LNCS*, pages 442–448, 2007.

[59] Ingolf Krüger, Michael Meisinger, and Massimiliano Menarini. Applying Service-Oriented development to complex systems: BART case study. In *Monterey Conference on Reliable Systems on Unreliable Networked Platforms*, Lecture Notes in Computer Science, pages 26–46. Springer, 2007.

[60] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison Wesley, 3 edition, December 2003.

[61] Stuart K.Williams, Steven A. Battle, and Javier Esplugas Cuadrado. Protocol mediation for adaptation in semantic web services. In *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 635–649. Springer, 2006.

[62] S. Lamparter and Y. Sure. An interdisciplinary methodology for building service-oriented systems on the web. In *IEEE International Conference on Services Computing*, volume 2, pages 475–478. IEEE Computer Society, 2008.

[63] Stephen Lane, Antonio Bucchiarone, and Ita Richardson. SOAdapt: a process reference model for developing adaptable service-based applications. *Information and Software Technology*, 2011.

[64] Stephen Lane, Qing Gu, Patricia Lago, and Ita Richardson. Towards a framework for the development of adaptable Service-Based applications. *Journal of Service Oriented Computing and Applications*, 2012.

[65] Stephen Lane and Ita Richardson. Process models for service based applications: A systematic literature review. *Information and Software Technology*, 2010.

[66] Soren Lauesen and Otto Vinter. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering*, 6:37–50, February 2001.

[67] K. Lewin. Action research and minority problems. *Journal of social issues*, 2(4):34–46, 1946.

[68] Pranee Liamputtong and Douglas Ezzy. *Qualitative Research Methods*. Oxford University Press, USA, 2 edition, 2005.

[69] Y. S Lincoln and E. G Guba. *Naturalistic inquiry*. Sage Publications, Inc, 1985.

[70] Marcos López-Sanz, César J. Acuòa, Carlos E. Cuesta, and Esperanza Marcos. Modelling of Service-Oriented architectures with UML. In *Proceedings of the 6th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2007)*, volume 194 of *Electronic Notes in Theoretical Computer Science*, pages 23–37. Elsevier, 2008. 4.

[71] Hongen Lu and Manish Chhabra. A methodology for agent oriented web service engineering. In *Pacific Rim International Workshop on Multi-Agents (PRIMA 2006)*, Lecture Notes in Computer Science, pages 650–655. Springer, 2006.

[72] T. Margaria, B. Steffen, Martin Wirsing, Matthias Hoelzl, Lucia Acciai, Federico Banti, Allan Clark, Alessandro Fantechi, Stephen Gilmore, Stefania Gnesi, Laszlo Goenczy, Nora Koch, Alessandro Lapadula, Philip Mayer, Franco Mazzanti, Rosario Pugliese, Andreas Schroeder, Francesco Tiezzi, Mirco Tribastone, and Daniel Varro. SENSORIA patterns: Augmenting service engineering with formal analysis, transformation and dynamicity. In *Leveraging Applications of Formal Methods, Verification and Validation, Third International Symposium*, volume 17 of *Communications in Computer and Information Science*, pages 170–190, 2008.

[73] Peter Markie. Rationalism vs. empiricism. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[74] James Martin and Carma L. McClure. *Software Maintenance: Problems and Its Solutions*. Prentice Hall PTR, April 1983.

[75] P. K McKinley, S. M Sadjadi, E. P Kasten, and B. H.C Cheng. A taxonomy of compositional adaptation. Technical report, 2004.

[76] Michael Meisinger and Ingolf H. Kruger. A Service-Oriented extension of the V-Modell XT. In *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 256–268, Tucson, Arizona, March 2007. IEEE.

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

[78] Hausi Müller, Mauro Pezzè, and Mary Shaw. Visibility of control in adaptive systems. In *Second International Workshop on Ultra-Large-Scale Software-Intensive Systems*, pages 23–26, 2008.

[79] Michael D. Myers. Qualitative research in information systems. *MIS Quarterly*, 21(2):241–242, June 1997. ArticleType: research-article / Full publication date: Jun., 1997 / Copyright © 1997 Management Information Systems Research Center, University of Minnesota.

[80] Dinh Nguyen, Willem-Jan van den Heuvel, Mike Papazoglou, Valeria de Castro, and Esperanza Marcos. GAMBUSE: a gap analysis methodology for engineering SOA-Based applications. *Conceptual Modeling: Foundations and Applications*, pages 293–318, 2009.

[81] Ojelanki K. Ngwenyama and Allen S. Lee. Communication richness in electronic mail: Critical social theory and the contextuality of meaning. *MIS Quarterly*, 21(2):145–167, June 1997. ArticleType: research-article / Full publication date: Jun., 1997 / Copyright © 1997 Management Information Systems Research Center, University of Minnesota.

[82] L. Oconner, Bart Orriens, Jian Yang, and Mike Papazoglou. A rule driven approach for developing adaptive service oriented business collaboration. In *International Conference on Service-Oriented Computing (ICSOC 2005)*, volume 3826 of *Lecture Notes in Computer Science*, pages 182–189, 2005.

[83] Pádraig O'Leary, Fergal McCaffery, Steffen Thiel, and Ita Richardson. An agile process model for product derivation in software product line engineering. *Journal of Software Maintenance and Evolution: Research and Practice*, 2010.

[84] P. Oreizy, M.M. Gorlick, R.N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D.S. Rosenblum, and A.L. Wolf. An architecture-based approach to self-adaptive software. *Intelligent Systems and their Applications, IEEE*, 14(3):54–62, 1999.

[85] P. Oreizy, N. Medvidovic, and R.N. Taylor. Architecture-based runtime software evolution. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, pages 177–186, 1998.

[86] W. J Orlikowski and J. J Baroudi. Studying information technology in organizations: Research approaches and assumptions. *Information systems research*, 2(1):1–28, 1991.

[87] Claus Pahl. Semantic model-driven architecting of service-based software systems. *Information and Software Technology*, 49:838–850, 2007. 8.

[88] Dr Michael Papazoglou. *Web Services: Principles and Technology*. Prentice Hall, 1 edition, July 2007.

[89] M. P Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, 2003.

[90] Michael P. Papazoglou and Willem-Jan Van Den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 2:412–442, 2006. 4.

[91] Michael P. Papazoglou and Willem-Jan van den Heuvel. Service-oriented design and development methodology. *Int. J. Web Eng. Technol.*, 2(4):412–442, 2006.

[92] Garish Parikh. *Techniques of Program and Systems Maintenance*. Q E D Pub Co, 2 sub edition, June 1988.

[93] Barbara Pernici. Automatic learning of repair strategies for web services. In *Fifth European Conference on Web Services (ECOWS '07)*, pages 119 –128, Halle, Germany, November 2007. IEEE Computer Society.

[94] Dewayne E Perry, Adam A Porter, and Lawrence G Votta. Empirical studies of software engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, page 345–355, New York, NY, USA, 2000. ACM.

[95] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and monitoring web service composition. In *Artificial Intelligence: Methodology, Systems, and Applications*, Lecture Notes in Computer Science, pages 106–115. Springer, 2004.

[96] Norah Power. *A grounded theory of requirements documentation in the practice of software development*. Unpublished PhD thesis, Dublin City University, Ireland, 2002.

[97] Rational. Rational unified process - best practices for software development teams. Technical Report TP026B, 1998.

[98] H. Robinson, J. Segal, and H. Sharp. Ethnographically-informed empirical studies of software practice. *Information and Software Technology*, 49(6):540–551, 2007.

[99] Colin Robson. *Real world research: a resource for social scientists and practitioner-researchers*. Wiley-Blackwell, 2002.

[100] W. W Royce. Managing the development of large software systems. In *Proceedings of IEEE Wescon*, volume 26, page 9, 1970.

[101] K. Schwaber. Scrum development process. In *OOPSLA Business Object Design and Implementation Workshop*, 1997.

[102] W. K. Sharpley. Software maintenance planning for embedded computer systems. In *Proceedings of the IEEE COMPSAC*, pages 520–526, 1977.

[103] Matthias Steup. Epistemology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2010 edition, 2010.

[104] Gerald I. Susman and Roger D. Evered. An assessment of the scientific merits of action research. *Administrative Science Quarterly*, 23(4):582–603, December 1978. ArticleType: research-article / Full publication date: Dec., 1978 / Copyright © 1978 Johnson Graduate School of Management, Cornell University.

[105] H. Tan, B. Wu, and Wang Chengjun. Applying pattern oriented software engineering to web service development. In *International Seminar on Future Information Technology and Management Engineering (FITME 2008)*, pages 214–217, 2008.

[106] CMMI Product Team. CMMI for development, version 1.2. Technical Report CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, 2006.

[107] A. M. Tjoa, R. R. Wagner, Eduardo B. Fernandez, Preethi Cholmondeley, and Olaf Zimmermann. Extending a secure system development methodology to SOA. In *Database and Expert Systems Applications (DEXA'07)*, pages 749–754, 2007.

[108] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, and P. Traverso. Astro: Supporting composition and execution of web services. In *Lecture notes in computer science*, volume 3826, page 495, 2005.

[109] C. Urquhart, H. Lehmann, and M. D Myers. Putting the 'theory'back into grounded theory: guidelines for grounded theory studies in information systems. *Information systems journal*, 20(4):357–381, 2010.

[110] Samyr Vale and Slimane Hammoudi. Model driven development of context-aware service oriented architecture. In *The 11th IEEE International Conference on Computational Science and Engineering - Workshops*, pages 412–418, 2008.

[111] Humphrey S. Watts and Marc I. Kellner. Software process modeling: Principles of entity process models. Technical Report CMU/SEI-89-TR-002, Carnegie Mellon University, Pittsburgh, Pennsylvania, February 1989.

[112] Yves Wautelet, Youssef Achbany, Jean-Charles Lange, and Manuel Kolp. A process for developing adaptable and open service systems: Application in supply chain management. In *International Conference on Enterprise Information Systems (ICEIS 2009)*, volume 24 of *Lecture Notes in Business Information Processing*, pages 564–576, Milan, Italy, 2009. Springer.

[113] R. Winter and J. Schelp. Reference modeling and method construction: a design science perspective. In *Proceedings of the 2006 ACM symposium on Applied computing*, page 1561–1562, 2006.

[114] M. Wooldridge and N. R Jennings. Intelligent agents: Theory and practice. *Knowledge engineering review*, 10(2):115–152, 1995.

[115] S.S. Yau and J.S. Collofello. Some stability measures for software maintenance. *Software Engineering, IEEE Transactions on*, SE-6(6):545–552, 1980.

[116] R. K Yin. *Case study research: Design and methods*, volume 5. Sage Publications, Inc, 2009.

[117] Robert K. Yin. *Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series, Vol 5*. Sage Publications, Inc, 3rd edition, December 2002.

[118] L. J. Zhang, W. VanderAalst, P. C. K. Hung, Soo Ho Chang, and Soo Dong Kim. A service-oriented analysis and design approach to developing adaptable services. In *IEEE International Conference on Services Computing*, pages 204–211, 2007.

Appendix A

Publications from the Research

[P1] Stephen Lane, Qing Gu, Patricia Lago, and Ita Richardson. Towards a framework for the development of adaptable Service-Based applications. Journal of Service Oriented Computing and Applications (in review, accepted with minor changes), 2012.

[P2] Stephen Lane, Antonio Bucchiarone, and Ita Richardson. SOAdapt: a process reference model for developing adaptable service-based applications. Information and Software Technology, 2011.

[P3] Stephen Lane and Ita Richardson. Process models for service based applications: A systematic literature review. Information and Software Technology, 2010.

[P4] Vasilios Andrikopoulos, Antonio Bucchiarone, Elisabetta Di Nitto, Raman Kazhamiakin, Stephen Lane, Valentina Mazza, and Ita Richardson. Service Engineering. Service Research Challenges and Solutions for the Future Internet: S-Cube - Towards Engineering, Managing and Adapting Service-Based Systems. LNCS 6500, 2010.

[P5] Ita Richardson, Stephen Lane. Coordinated design knowledge models for software engineering and service-based computing. S-Cube Consortium, 2009.

[P6] Sajid Ibrahim Hashmi, Stephen Lane, Dimka Karastoyanova, and Ita Richardson, A CMMI based configuration management framework to manage the quality of service based applications. In Industrial Proceedings of the 17th conference on Software Process Improvement, 2010.

Appendix B

Systematic Literature Review

Information and Software Technology 53 (2011) 424–439

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Process models for service-based applications: A systematic literature review

Stephen Lane *, Ita Richardson

Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland

ARTICLE INFO

Article history:
Received 8 June 2010
Received in revised form 6 December 2010
Accepted 12 December 2010
Available online 17 December 2010

Keywords:
SOA
Service-based application
Software process
Systematic literature review

ABSTRACT

Context: Service-Oriented Computing (SOC) is a promising computing paradigm which facilitates the development of adaptive and loosely coupled service-based applications (SBAs). Many of the technical challenges pertaining to the development of SBAs have been addressed, however, there are still outstanding questions relating to the processes required to develop them.
Objective: The objective of this study is to systematically identify process models for developing service-based applications (SBAs) and review the processes within them. This will provide a useful starting point for any further research in the area. A secondary objective of the study is to identify process models which facilitate the adaptation of SBAs.
Method: In order to achieve this objective a systematic literature review (SLR) of the existing software engineering literature is conducted.
Results: During this research 722 studies were identified using a predefined search strategy, this number was narrowed down to 57 studies based on a set of strict inclusion and exclusion criteria. The results are reported both quantitatively in the form of a mapping study, as well as qualitatively in the form of a narrative summary of the key processes identified.
Conclusion: There are many process models reported for the development of SBAs varying in detail and maturity, this review has identified and categorised the processes within those process models. The review has also identified and evaluated process models which facilitate the adaptation of SBAs.

Contents

* Corresponding author.
E-mail addresses: stephen.lane@lero.ie (S. Lane), ita.richardson@lero.ie (I. Richardson).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.12.005

1. Introduction

Repeatable software development process are a key component in the development of high quality software [1]. Implementing a repeatable process ensures that all of the necessary development tasks get completed, and in the correct sequence. In order to make a process repeatable it needs to be documented so that project stakeholders can adhere to the process. In software engineering it is common to document a process by means of a process model. Software process models come in many forms from high level models such as the traditional waterfall model [2] to comprehensive improvement models such as CMMI [3] or the ISO-15504 [4]. Many domains such as the medical device or automotive industries require that software used in their products follow one of these comprehensive process models.

With the recent emergence of Service-Oriented Computing (SOC) many industries are interested in reaping its benefits. These benefits include the loose coupling of services which facilitates software reuse and the flexibility of service-based applications (SBAs) which allows them to be adapted with minimal effort. Unfortunately, because of the diversity of business requirements and operating contexts that accompanies the development of Service-Oriented Architectures (SOA), older software development paradigms cannot be blindly applied [5]. Many existing process models were designed with object-oriented or component based development in mind [6], therefore, since SOC requires a new development approach it will also require new supporting process models.

In this work we will have completed a systematic literature review in order to document the state of art in development processes models for developing service-based applications (SBAs). This will provide a useful starting point for further research in the area. Section 3 outlines the research method for this review, followed by Section 4 which outlines the results of the review. Section 5 contains a meta-model of the processes identified in the study, with Section 6.1 providing detailed descriptions of each of the process from the meta-model. Section 6.2 looks in detail at each study that facilitates adaptation in response to the second research question of the study. Section 7 contains a discussion of the results, followed by the conclusions in Section 8.

2. Background

2.1. Processes models for service-based applications

There have been many service-oriented process models proposed by proponents of SOC, for example IBM have proposed the SOMA [7] methodology while a group of industry practitioners have proposed the Service-Oriented Architecture (SOA) practitioners guide [8]. Similarly there have been efforts to tailor the RUP framework so that it can be used for SBA development. There are many SBA process models reported in grey literature such as white papers or technical reports. Lane et al. [9] report many of these, however, there may also be many other proprietary models maintained by organisations who keep them confidential for competitive advantage. Apart from proprietary models or the models reported in grey literature, there are many models published in scientific publications such as conference or workshop proceedings as well as peer reviewed journals. In order conduct an exhaustive review of relevant SBA process models using these sources, a systematic literature review (SLR) was conducted by the authors. A SLR is a comprehensive review carried out based on a set of predefined search criteria, which follows documented and repeatable guidelines. This is a popular review method in areas such as evidence based medical research and is being conducted increasingly in software engineering. The particular SLR procedure used for this review was proposed by Kitchenham and Charters [10] to meet the requirements of a software engineering review. In the next sections the review method and results of the review will be described in detail.

2.2. Taxonomy

Within the research areas of Software Process and Service-Oriented Computing (SOC) the terminology used is often ambiguous and non-standard. Having a consistent set of terms is important when evaluating or comparing different studies during a systematic review. In this section a taxonomy is presented where a brief description of the most common terms used throughout the review will be given with reference to synonyms found in the literature. The software process definitions used in the review

are from Derniame et al.'s (1999) "Software process: principles, methodology, and technology" [11] which gives a comprehensive view of the software process research area. The SOC terms used in this review are defined the S-Cube project's knowledge model [12].

Software process refers to the organisation, management, measurement and improvement of the activities involved in software development. Every software development project has a process whether or not it is explicitly or implicitly defined. At a bare minimum, software development will consist of a set of related activities whether or not they are measured or improved. Synonyms in the literature for software process include: software development process or software life-cycle [11].

Software process models or process models are representations of real world software processes. Ideally process models should perfectly reflect the software processes they are trying to represent. Organisations often measure and improve their software process by documenting it and comparing it to reference process models such as CMMI or ISO-15504. Similarly, organisations may choose to implement an existing process model from the many types available [11].

Service-Oriented Computing (SOC) refers to the computing paradigm that has evolved from object-oriented and component based design. The SOC paradigm facilitates ad-hoc, platform independent communication between unintroduced parties through the use of software services [12].

Services play a central role in SOC, they are self-contained loosely coupled computational elements that expose the functionality of underlying software. Services can be combined or composed into more complex services or they can be used in combination with other services to form service-based applications. Services can be published on public registries and made available for use by third party clients [12].

Service-based applications (SBAs) are applications that are composed of software services, the services may be controlled by the application developers or they may be third party services. A common example given for a SBA is the travel booking application: this is composed of a hotel booking service, a car rental service and an airline reservation service each of which are provided by third parties. Synonyms in the literature include service-oriented systems or service-based systems [12].

3. Research method

This review follows the guidelines set out in Kitchenham and Charters' [10] guide for SLRs in software engineering. This research method provides a verifiable method of summarizing existing approaches as well as identifying gaps in the current research.

3.1. Systematic literature review protocol

A review protocol is a written plan that is completed before the review begins. Therefore, the success of the review depends on the quality of the protocol. The protocol also provides a means by which the review itself can be repeated or updated at a later date to include subsequent publications. The protocol describes every aspect of the review from searching electronic databases to creating the final report and lays the groundwork for an unbiased systematic review Because the review is a snapshot in time it is impossible to replicate exactly. It would, however, be possible to update an existing review by including only new studies from a subsequent literature search.

3.1.1. Research questions

The review protocol begins with the construction of the research question or questions. Every systematic review has at least one primary research question with the possibility of one or more secondary questions. The research questions can be freely formed or they can contain a Population, Intervention, Comparison, and Outcomes as per Kitchenham and Charters' guide. The following are the primary and secondary research questions for this study:

1. What software processes models are proposed for developing SBAs ? (RQ-1)
   - *Population:* software
   - *Intervention:* process for SBAs
   - *Comparison:* process for traditional software
   - *Outcomes:* set of processes and techniques.
2. Do proposed SBA development processes models facilitate SBA adaptation ? (RQ-2).

3.1.2. Data sources

For this SLR, the electronic data sources in Table 1 were searched as these are the primary sources for software engineering research publications. Other software engineering SLRs such as [13],[14] use similar or the same sources. Other resources such as CiteSeer and Google Scholar are powerful search tools which return many of the same results as the databases mentioned. This duplication as well as the volume of non-relevant publications means that they were excluded as sources for this review.

3.1.3. Search string

The primary research question was be broken down to give a list of search terms to use in the electronic databases. Since the secondary question is based on the first, it is not necessary to have separate search terms. The search string will use the logical operator OR to include synonyms for each search term, and the logical

Table 1
Literature sources.

Source	URL
IEEE Explore	http://Ieeexplore.ieee.org
ACM Digital Library	http://portal.acm.org
ISI Web of Knowledge	http://www.isiknowledge.com
SpringerLink	http://www.springerlink.com
ScienceDirect	http://www.sciencedirect.com
Wiley InterScience	http://www3.interscience.wiley.com

Table 2
Search synonyms.

Software process	Service-based
Development technique	SBA
Development approach	SOSE
Development process	SOA
Development life-cycle	Service based
Development life cycle	Service-oriented
Development methodology	Service oriented
Process model	Service-centric
Process framework	Service centric
Development model	Service computing
Development framework	Service engineering
sdlc	
cmm	
cmmi	
15504	
spice	
12207	
15288	

Table 3
Inclusion/exclusion criteria sources.

Included
- Peer reviewed research papers from electronic databases.
- Papers that describe more than a single specialised process.
- Must describe processes for developing SBAs.

Excluded
- Non-English papers.
- Papers specific to a particular technology.
- Papers that only address once specific process area.
- Papers that are application domain specific.
- Papers which are obviously not related to the research questions in this protocol.
- Letters and editorials.
- Duplicate publications on the same approach.

operator AND to link together each set of synonyms. Table 2 lists all of the search synonyms used in the construction of the search string. When concatenated using the appropriate boolean expressions the following generic search string was produced:

("software process" or "software engineering process" or "engineering process" or "engineering methodology" or "sose framework" or soad or soma or "architecture framework" or "architecture frameworks" or "engineering methodology" or "development technique" or "development approach" or "development methodology" or "development process" or "development processes" or "development life-cycle" or "development life cycle" or "process model" or "process framework" or "development model" or "development framework" or sdlc or cmm or cmmi or 15504 or spice or 12207 or 15288) and (sba or sose or soa or "service-based" or "service based" or "service-oriented" or "service oriented" or "service-centric" or "service centric" or "service computing" or "service engineering").

Some of the databases such as IEEEXplore, can take this string directly while others, such as Web of Science, required slight modification. SpringerLink has a 10 term length on search strings which meant that a series of strings had to be used instead of one generic string. In all cases the string was applied to the abstracts contained in the databases. Applying the string to titles produced too few results while applying the string to full texts produced thousands of inaccurate results.

3.1.4. Study selection

The studies that were selected for inclusion in this study were identified from on-line electronic databases during October 2009. In order to determine whether or not a study is included, the abstract was evaluated based on the inclusion/exclusion criteria in Table 3. The inclusion and exclusion criteria we selected in order to exclude irrelevant publications while keeping studies of interest. Most of the criteria are self explanatory, however, some of the criteria need a little explanation. The criterion: "Papers that describe more than one process" was included because hundreds of the the results returned in the initial search detailed single low level processes that were too specialised to be applied generally by practitioners. We were looking for studies that were aimed at guiding the high level development process such as life-cycle models or process models with several high-level processes. The criterion: "Must describe service-oriented process activities/practices" was included to filter out studies that do not contain processes/activities that can be used to develop SBAs.

In many cases it was not possible to determine whether or not a study should be included due to the quality of its abstract. In these cases the studies were either included or excluded based on their entire full text. After the initial set of studies were selected their details were recorder using the citation management tool Zotero.[1] Full texts were obtained where possible for the data extraction part of the SLR. It is also important to note that many papers were excluded based on their full text as it became apparent that they failed to meet the inclusion/exclusion criteria during the data extraction process.

The study selection process is one of the most critical processes of the review process and also one of the most time consuming. It is therefore vital that the search string only returns studies that are closely related to the research questions to prevent unnecessary study selection effort. It is also vital not to exclude important relevant studies by using search strings with an overly narrow focus. To strike a balance between too many and too few results of the selection process, the search string was validated using a selection of 5 known relevant publications. The search string was tested against the publisher's websites for those known publications to ensure that the studies were returned in the publisher's search engine results.

The selection process was undertaken by the first author, with a random selection of 20 of the original results audited by the second author in order to ensure minimal selection bias. There was a close correlation between the studies chosen for selection by the two authors which indicates minimal selection bias.

3.1.5. Data extraction

After the full text of the studies were retrieved the data extraction phase begun. For this activity a custom data extraction form was created using MS Access™. This allowed for convenient data entry as a paper's full text was being reviewed. Another benefit of using this approach was the ease with which reports could be generated from the underlying data table. Meta-data such as author, title and publication source was collected with descriptive data fields such as study type and focus during this activity. Table 4 shows the data fields that were extracted from each study along with their descriptions and their associated research questions. The first research question, RQ1, which refers to the identification of process models for developing SBAs requires all of the data extraction fields for analysis. The data fields are used to illustrate frequency distributions as well as to map the various processes for developing SBAs. Some of the more content-oriented fields such as "Processes" and "Study Focus" are used to determine whether the identified processes facilitate SBA adaptation, in answer to the second research question, RQ2.

Like the study selection process, the data extraction process was conducted by the primary author and audited by the secondary

[1] http://www.zotero.org/.

Table 4
Data extraction fields.

Data field	Purpose	Research questions
Reviewer	Name of reviewer	RQ1
Title	Name of study	RQ1
Authors	Study authors	RQ1
Publication source	Where the paper is published, name of journal, conference, etc.	RQ1
Publication type	Is the paper a conference paper, journal paper, etc.	RQ1
Year of publication	When study was published	RQ1
Primary/secondary	Deos the study use primary or secondary data	RQ1
Study type	Type of research methods used	RQ1
Study context	Industry, lab based study, etc.	RQ1
Study population	Study participants, students, academics, industry experts, etc.	RQ1
Research questions	Research question(s) of the study	RQ1, RQ2
Study focus	Primary objective of the study	RQ1, RQ2
Processes	Process described in the study	RQ1, RQ2
Findings/conclusions	Main conclusions from the study	RQ1, RQ2
IS valid	Was it a valid study	RQ1, RQ2

author. A random selection of 10 of the selected studies were reviewed by the second author and the appropriate data fields extracted. The fields that were extracted by both authors were compared and found to be sufficiently similar not to warrant further validation.

3.1.6. Data synthesis

The data synthesis phase of this review aims at addressing the two research questions posed at the beginning of the review. The first research question was to identify process models for developing SBAs. In order to address this research question, details of the process models contained in the selected studies were extracted. In order to get an idea of the scope of the process models identified, the individual processes identified within each process model were synthesised into a meta-model. This meta-model provides an overview of the area and will also highlights gaps in the research. Once the meta-model is completed a narrative summary of each of the process areas is provided based on the studies that make up each category. The second research question asks whether or not the development processes that are identified address adaptation of SBAs. In order to answer this question studies which contain SBA adaptation related processes are also critically reviewed through a narrative summary.

A meta-model is similar to a map from a mapping study as outlined by [15,16]. Rather than looking at each study in detail, the aim is to scope out the breath of the research area. The narrative summary of each process within the meta-model is more typical of SLRs than mapping studies. For this reason the data synthesis process is divided into two parts, a mapping study to scope the area and an SLR to evaluate the included studies. Petersen et al. [15] recommend that rather than use these synthesis approaches in isolation, they can be used to complement one another. For example, in this case, the systematic map is first created, then the specific processes are critically evaluated through the narrative summary. A noteworthy exclusion from this study is a quality assessment, as these are often present in systematic reviews. However, in this study there are many qualitative studies included which makes it difficult to apply a non-subjective quality score.

For this study, a UML[2] (Unified Modeling Language) meta-modeling approach was taken as an alternative to the bubble plot approach often used in mapping studies. The processes used in this meta-model were constructed using Noblit and Hare's reciprocal translation technique [17]. This is the synthesis technique suggested by Kitchenham [10] when researchers are attempting to create an additive summary of the literature. Reciprocal translation summarises concepts by translating similar concepts into a single concept. This processes is continued until it is not possible to translate any more concepts into one another. For example, if the following terms are encountered: "service-based application adaptation", "service-oriented system adaptation" or "dynamic modification of service-oriented systems" they may all be translated into "service-based application adaptation". This technique then produces a normalised set of development processes covering the entire research area assuming that the included studies are representative. Other statistical data such as frequency distributions were reported in tables which illustrate how the research area changes over time.

In our meta-model we have made use of the following UML constructs some of which, for simplicity, have been tailored slightly from their official definitions:

- *Class:* A class (square box) in our mata-model is used to represent concepts or processes.
- *Association:* An association in our model is used to represent a relation between two concepts (processes, sub-processes). In our model we have used standard associations (arrow without arrowhead) and directional associations (arrow with open arrowhead). Directional associations imply precedence and sequence between classes (processes).
- *Aggregation:* An aggregation (arrow with diamond head) is used in the context of this model to represent "has a" relationships.
- *Generalization:* A generalization (arrow with closed arrowhead) relation in our model represents a relationship between two classes (processes) where one class is a sub-type of the other.

The narrative summary synthesis method is used twice during this study, firstly to summarise the key processes from the meta-model and secondly to summarise and critically discuss each of the studies that contain processes for adapting SBAs. Since there are to many studies to discuss individually, it seems appropriate to discuss the process themes emerging from the studies. SBA development processes that also facilitate adaptation are expected to be a much smaller subset of the total number of studies which permits a narrative summary at an individual study basis. Therefore the meta-model and process summaries address RQ-1, while the summaries of each study with adaptation processes address RQ-2.

4. Results

After an exhaustive literature search, our data contained 722 studies. The two largest set of results came from IEEE Xplore (185) and the Web of Science (333). Studies were then excluded if they focused on a single process or technique or were specific to a particular application domain. If the exclusion criteria were relaxed there would be many more interesting studies. However they are beyond the scope of this review. After the abstracts and titles of each study were read the number of studies was reduced to 77. However, on fully reading each of these papers a further 20 were excluded leaving a total of 57 valid studies. The categorical data that was extracted from the selected studies is shown in Table 5.

[2] http://www.uml.org.

Table 5
Categorical data extracted from selected studies.

Study	Source	Lifecycle processes	Sub processes	End to end model	Supports adaptation
[S1]	Conference paper	RED, CON	MDD	No	Yes
[S2]	Conference paper	RED, CON	MDD	No	No
[S3]	Journal paper	RED, CON, DP, EM	Discovery, composition	Yes	No
[S4]	Conference paper	RED, CON, DP	MDD	Yes	No
[S5]	Workshop paper	RED, CON	Agent-oriented development	No	No
[S6]	Conference paper	RED, CON	MDD	No	Yes
[S7]	Conference paper	RED, CON	MDD	No	No
[S8]	Workshop paper	RED, CON	Service composition	Yes	No
[S9]	Conference paper	RED, CON	MDD, security	No	No
[S10]	Conference paper	RED, CON, DP	MDD	Yes	No
[S11]	Conference paper	RED	Agent-oriented development	No	Yes
[S12]	Journal paper	RED, CON, DP	Service composition	No	No
[S13]	Conference paper	RED, CON	Service composition	Yes	Yes
[S14]	Conference paper	RED, CON	Service specification	No	Yes
[S15]	Conference paper	RED, CON, DP	Service specification	No	No
[S16]	Conference paper	RED	Service identification, service specification, service realisation, service composition	Yes	Yes
[S17]	Conference paper	RED	Service identification, service specification, service realisation, service composition	Yes	No
[S18]	Conference paper	RED, CON, DP,		Yes	No
[S19]	Conference paper	RED, CON, DP, EM	Service discovery, service composition	No	No
[S20]	Conference paper	RED, CON, DP		No	No
[S21]	Workshop paper	RED, CON	MDD	No	No
[S22]	Conference paper	RED, CON	Collaborative modelling	Yes	No
[S23]	Thesis	RED, CON	MDD	No	No
[S24]	Journal paper	RED, CON, DP		No	No
[S25]	Workshop paper	RED	Security	No	No
[S26]	Workshop paper	RED, CON	Formal methods	No	No
[S27]	Conference paper	RED, CON	MDD for SOMA	No	No
[S28]	Book section	RED, CON	Service composition	No	No
[S29]	Journal paper	RED	Formal methods	No	No
[S30]	Conference paper	RED, CON	Context aware MDD	No	Yes
[S31]	Journal paper	RED, CON	MDD	No	No
[S32]	Conference paper	RED, CON	MDD	No	Yes
[S33]	Conference paper	RED	Formal methods	No	No
[S34]	Conference paper	RED, CON	MDD	No	No
[S35]	Workshop paper	RED, CON	MDD	No	No
[S36]	Conference paper	RED, CON	Formal methods	No	No
[S37]	Journal paper	RED, CON	Semantic modelling	No	No
[S38]	Symposium	RED, CON, DP, EM	MDD, service specification, service realisation, service composition	Yes	Yes
[S39]	Conference paper	RED, CON	MDD	No	No
[S40]	Conference paper	RED, CON		No	No
[S41]	Journal paper	RED, CON, DP, EM		Yes	No
[S42]	Conference paper	RED, CON		No	No
[S43]	Conference paper	RED, CON	Service identification, service specification, service realisation	No	No
[S44]	Conference paper	RED, CON	Service identification	No	No
[S45]	Conference paper	RED, CON		No	No
[S46]	Workshop paper	RED, CON	MDD	No	No
[S47]	Journal paper	RED, CON, DP	MDD, service identification, service specification, service realisation	Yes	No
[S48]	Conference paper	RED, CON	Aspect-oriented development	No	No
[S49]	Workshop paper	RED, CON, DP		Yes	No
[S50]	Conference paper	RED, CON	Aspect-oriented development	No	No
[S51]	Conference paper	RED, CON, DP	MDD	Yes	No
[S52]	Conference paper	RED, CON, DP	Web based MDD	Yes	No
[S53]	Conference paper	RED, CON, DP	MDD	Yes	No
[S54]	Journal paper	RED, CON, DP		Yes	No
[S55]	Journal paper	RED, CON	MDD	No	No
[S56]	Conference paper	RED, CON, DP	Service discovery	Yes	No
[S57]	Conference paper	RED, CON, DP, EM		Yes	No

RED – requirement engineering and design.
CON – construction.
DP – deployment and provisioning.
EM – execution and management.
MDD – model driven development.

4.1. Summary of selected studies

From the 57 papers selected there was a wide variation in the approaches described, the quality of the research and the maturity of the studies. There were 45 secondary research and 12 primary research studies which suggest that the research area is lacking in primary studies. Analysis shows that there were 40 academic studies compared to 17 industry based studies. Although secondary or academic studies may be as valid as their counterparts this trend suggests that the area is lacking mature industry-based

Table 6
Types of publication by year.

Publication type	Year						Total
	04	05	06	07	08	09	
Book section					1		1
Conference paper	3	3	3	9	14	4	36
Journal paper	1	1	1	2	2	3	10
Symposium					1		1
Thesis				1			1
Workshop paper	1		3	3		1	8
Total	5	4	7	15	18	8	57

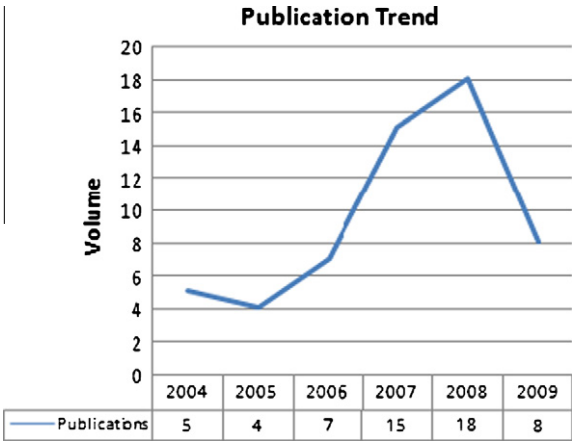

Fig. 1. Publication trend.

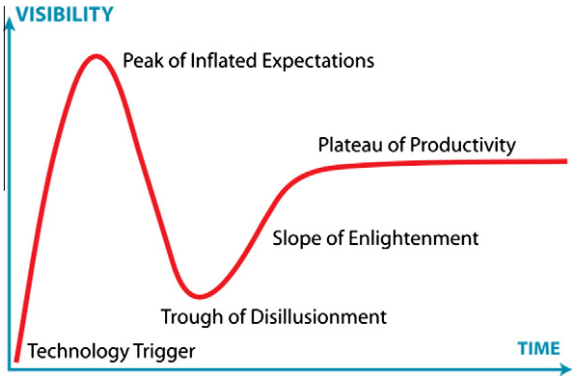

Fig. 2. Gartner Hype Cycle.

research. A good indication of the maturity of an area is the type of publications in that area. Journal articles are often more mature than conference or workshop papers. Table 6 shows the number of each type of publication by year. Conference papers, workshop papers and journal articles make up the majority of publications. There are only 10 journal articles with a total of 44 workshop and conference papers.

As illustrated in Fig. 1, the first service-oriented development approaches appeared in 2004, with the number of publications rising rapidly from 2006 to 2008. Then interestingly the number of publications drop off again in 2009. One interpretation for the 2009 drop is that some publications were not yet indexed by the electronic databases when the searches were conducted in October 2009. Another interpretation for the drop in 2009 may be that the publication trend is following the Gartner Hype-Cycle[3] in Fig. 2, in this case the trend would suggest that the research area has already reached its peak of inflated expectations with 2009 representing its trough of disillusionment. This corroborates the Hype-Cycle of Service-Oriented Software Engineering (SOSE) challenges reported in [13]. Its is advisable not to draw draw conclusions from the trend without knowing the true number of publications indexed in 2009.

Table 7
Study type.

Literature based	25
Discussion	12
Example application	9
Case study	8
Exploratory case study	2
Experience report	1

Another interesting way to view the selected studies is by study type, this may also give us a view into the quality and maturity of the selected studies. As can be seen in Table 7, the most popular study type was literature based studies. In this context, "Literature based" means that existing approaches from the literature were used or derived from to create service-oriented development approaches. The next most popular study type were discussion based papers. The discussion based papers that were reviewed were mostly short papers based on the authors' experiences rather than being based on primary or secondary research. The remaining studies were either example applications or case studies. Example applications were used to illustrate tool supported approaches such as Model Drivel Architectures (MDA).

5. RQ-1: Mapping study

In this section a quantitative mapping analysis of the selected studies is provided, each study is categorised according to the processes they contain and mapped to a process meta-model. The publication and study type distributions are shown in tables, while the process category relationships are mapped using a subset of UML (Unified Modeling Language). This approach was chosen because it is an intuitive representation format with the ability to represent associations between processes.

5.1. Software process meta-model

The meta-model shown in Fig. 3 displays the processes identified in this study along with their inter-relations.

The meta-model construction was a multi-stage process that was completed after the data collection activity of the SLR. The first step of constructing the meta-model was to classify the processes extracted from the literature into groups of related processes using reciprocal translation. One of the decisions during reciprocal translation is which process description to use in order to describe a group of processes. In our earlier example we encountered the following processes: "service-based application adaptation", "service-oriented system adaptation" or "dynamic modification of service-oriented systems". Any of these individual terms can be used to describe the process in our meta-model. In order to keep the resultant processes consistent with one another the processes from Papazoglou and Van Den Heuvel's [5] Service Development Life-Cycle (SLDC) approach were used as a base point for the translation process. The four life-cycle processes from the SLDC which were used as the base point are highlighted in Fig. 3. This approach was chosen as a base point because it is a detailed approach that has high-level processes for the entire development life-cycle.

[3] http://en.wikipedia.org/wiki/Hype_cycle.

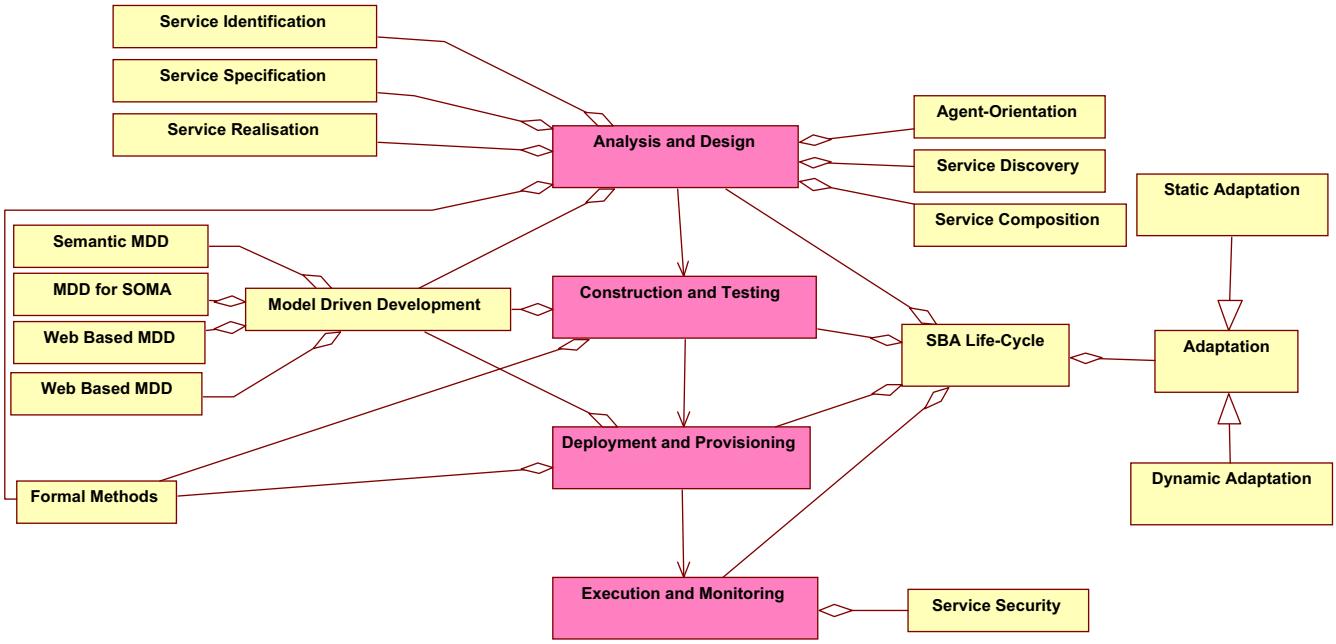

Fig. 3. Service development process meta-model.

There are several other high-level life-cycle models that could have been chosen for the base point, however, the same meta-model should result regardless of which one is chosen if the reciprocal translation process is followed accurately. Processes encountered during the review were evaluated as to whether or not they could be translated into one of the base point processes. If they could be translated they were eliminated. If not, they were added to the meta-model as a separate process or sub process.

The next step was to construct these related processes into a meta-model where associations between processes were represented. The completed meta-model gives a comprehensive view of the state of the art in software processes for SBAs. In the following sections an overview will be given of each of the prominent software processes proposed in the studies selected for this review.

6. Systematic review

6.1. RQ-1: Process categories

6.1.1. End to end models

Many of the studies encountered in this review propose complete SBA development methodologies with processes for designing, implementing and operating them. These methodologies often borrow concepts from existing software development techniques or methodologies, for example, Kruger and Meisinger [S15] propose an extension of the V-Model for use with SBAs. Similarly Christou et al. [S24] propose the use of Agile methods along with the Rational Unified Process (RUP) development methodology for the development of SBAs. The use of Software Product Lines (SPL) development techniques are also suggested in combination with Agile methods by Karama *at al* [S12]. Deubler et al. [S51] have developed a tool to support the development of SBAs, the tool's primary functions are to formally verify service-based systems as well as the generation of system code.

Many of the end to end process models in the literature follow the classical waterfall development model with some modifications that are particular to SOC. Adamopoulos [S18] proposes an iterative development methodology which includes many of the waterfall model's processes, however they are adapted to suit Web Service Based Applications (WSBAs). Papazoglou and van den Heuvel [S41] suggest a process model that also bears a close resemblance to the waterfall model. However, it has processes such as "Execution and Monitoring" that are specific to SBAs. Processes in this model such as "Analysis and Design" bear resemblance to traditional processes not intended for SBA development. However, they are tailored specifically to meet the requirements of such applications.

Along with the fully defined process models, there are many "work in progress" models in the literature. Many of the "work in progress" models attempt to discover all of the service related concepts and aspects that make development of SBAs unique from the development of traditional software applications. One such work by Engels and Assmann [S42] set out the several development challenges of service-oriented enterprise architectures. Ivanyukovich et al. [S54] are also working towards a service-oriented process model They propose a model with the following three dimentions: managing the software change, specifying the development process and targeting stakeholders goals.

The final type of end to end process model encountered in this review are interdisciplinary development models. These development models involve expertise from several domains. For example, Lamparter and Sure [S19], propose a development methodology that includes aspects of web service engineering, market engineering and ontology engineering.

6.1.2. Analysis and design

There are many analysis and design process models reported for the development of services and SBAs. Most approaches fall in to the categories of top down, meet in the middle or ground up service realisation [S41]. The top down design approach involves working backwards from existing service interfaces. The components required to implement the service are constructed based on the functionality described by these interfaces. This design technique is considered best practice as business requirements shape the service interfaces which in turn drive the development process. The bottom up approach differs in that the functionality of existing applications are made available by wrapping existing system functions with services. Finally, the meet in the middle

approach combines features of both top down and bottom approaches. This technique is useful when a service needs to be created to implement an existing service interface. However, rather than create a new software component to implement this service, which would be done with top down design, an existing software component that has all of the required functionality would be wrapped with the required service.

The top down approach is more business oriented than its counterparts because services are created to meet specific business requirements. The same can be said for the meet in the middle approach when software components exist to suit service requirements. The ground up service realisation approach is least business oriented in that existing software components are wrapped with services which limits the capability of resultant applications.

After reviewing the development models that address the analysis and design processes, it appears that many of these models inherit characteristics from the popular Service-Oriented Modelling and Architecture (SOMA) methodology [S47,S43] developed by Arsanjani et al. at IBM. The SOMA methodology is one of the most mature design process models encountered in the review. Its developers claim that methodology was developed during involvement with hundreds of SOA projects. SOMA's design activities are divided into three processes: "Service Identification", "Service Specification" and "Service Realisation". These processes appear in many other models in some form or another. For example, Engels et al. [S4] propose a method for engineering SOAs which contain the design processes of: "Identify top-level business services", "Refine business services" and "Specify business services". The SOMA processes: Service Identification, Service Specification and Service Realisation and the other analysis and design processes encountered will be discussed in more detail in the following subsections.

Service identification is a process that identifies the candidate services required for developing a SBA. It involves the identification of the required services as well as the software that will realise those services. SOMA identified three techniques for the identification of services: goal-service modeling (GSM), domain decomposition, and existing asset analysis. Kenzi *at al* [S6] propose a Model Driven Architecture (MDA) framework for the development of adaptable SBAs that uses service identification models for the identification of candidate services.

Service specification involves the development of service interfaces and the management of service dependencies, once the required services have been identified. Most service development methodologies either implicitly or explicitly contain a service specification process specifying services is one of the fundamental building blocks of SBAs.

Service realisation, like service specification, is a vital step in the development of SBAs. Realisation is the final process before the implementation of services. It involves the implementation of the service specification into usable design documents that can be used by the service implementers. Service design documents need to include artifacts such as component implementation details and SOA solution stack details. An extremely common approach being proposed for the realisation of services is the application of Model Driven Architecture (MDA). This is a model driven design approach with the added benefit of automatic code generation from design models. MDA will be discussed in more detail later in this review.

Another description given to the analysis and design processes for SBAs is service-oriented analysis and design (SOAD). This is an extension of the Object-Oriented Development (OOD) and Component-Based Development (CBD) paradigms. SOAD is often used as a general term in the literature to describe analysis design approaches for service oriented systems. Different SOAD methodologies take different approaches to service design. For example, Chang and Kim [S17] propose a 5 phase design process within the following phases: "Identifying Business Processes", "Defining Unit Services", "Discovering Services", "Developing Services", and "Composing Services". Comparatively Kambhampaty [S40] proposes a SOAD methodology with four processes which are quite different from Chang and Kim's methodology: "Activity Services Development", "Business Process Services Development", "Client Services Development" and "Data Services Development".

Service discovery for static SBAs occurs during application design , while the discovery of services for dynamically adapting SBAs may occur during system runtime. There are many ways that service discovery can be achieved, from manually searching service directories to automatically scanning directories for candidate services that match a given criteria. Howard and Kerschberg [S3] present a framework to support the preparation, publication, requisition, discovery, selection, configuration, deployment, and delivery of semantic web services. Their framework specifies that both published services and service clients contain semantic descriptions of their capabilities and various types of requirements.

Service composition is a service design/construction process that involves the combination of services into new complex services or SBAs. Service composition spans several processes of the software development life-cycle – how services are composed needs to be established during the design of SBAs, the composition gets enacted during the development of a SBA and finally service compositions need to be monitored and possibly adapted during the operation of SBAs. Ren et al. [S8] propose a framework that facilitates the visual design, validation and development of web service compositions.

Agent-orientation describes a computer architecture that is made up of a system of intelligent agents. The agents have characteristics such as autonomy, social ability, reactivity, and pro-activeness [18]. Lu and Chhabra [S5] propose a service-oriented design methodology that involved the wrapping of intelligent agents with web services which speeds up the development process and results in more competent web services.

6.1.3. Construction

Many of the process models encountered in the literature focus on the construction of services [S52], SBAs [S10] or both [S7]. The construction of services is a role undertaken by service providers whereas the construction of SBAs is a role undertaken by service consumers. Service providers and service consumers have differing roles but may be in the same organisation. The focus of this review is the construction of SBAs rather than services alone, however in the absence of suitable existing services, the construction of services is a sub-process of the SBA construction process.

The SBA development processes proposed vary in their approach. A common proposal is to adapt an existing traditional engineering approach or framework to suit the service-oriented paradigm. Meisinger and Kruger [S15] suggest an adapted version of the traditional V-Model for the development of service-oriented systems. Similarly there have been many Model Driven Development (MDD) techniques suggested for the construction of SBAs. MDD is a popular approach for the design and implementation of object-oriented software. During MDD analysis and design models can be used to automatically generate system code automatically constructing the system. MDD can greatly simplify the construction process of SBAs by abstracting complex implementation standards and service interface details from developers. This process will be discussed in more detail in the next section.

Model Driven Development (MDD) is a development paradigm that uses models to represent and reason about problem and solution domains [S7]. Modeling is a useful method of working towards a solution which enables the representation of dependencies between the components being modeled. Once a model has been

created it is possible to view the effects of changing part of the model by examining dependencies within the models. A model can be used as a reference in order to construct a software system, alternatively using the appropriate technology a well defined model can be used to automatically generate a platform specific model and then application code which can form the skeleton of a system. This use of modeling to generate system code is referred to as Model-Driven Architecture (MDA).

Models are an intuitive method of developing solutions while at the same time they abstract the specific details of implementing the solution. This proposition is particularly appealing to SOC where there are many protocols and standards involved in implementing SBAs. In order to implement a basic WSBA the solution may involve the SOAP protocol for messaging, WSDL for describing service interfaces, and BEPL for implementing service compositions. Each of these are verbose XML based languages which are difficult to interpret by humans and are error prone when constructed manually. For this reason there have been many service development processes proposed that are based on MDD. In this systematic review 17 different MDD approaches were encountered.

A common approach proposed is to develop system models with UML [S35] diagrams. This is a logical choice as there are many MDA tools in existence that can handle UML. Johnson and Brown [S7] outline an MDD technique for developing SBAs using a UML profile suitable for service modelling. Many of the MDD approaches proposed conform to MDA principles where there are several models created during development and these different model types can be transformed into one another and finally into source code. The most common model types proposed in the literature were Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM) [S32,S37]. Each of these models are one order of abstraction higher than their successor, for example, a CIM models domain specific artifacts which would use vocabulary familiar to domain practitioners, while PIM models refer to the structure of a system without referring to platform specific details [19].

There are too many MDD approaches in the literature to review in detail, so a list is presented to briefly summarise some of the approaches suggested:

Collaborative modeling is a collaborative MDD approach where stakeholders involved in system development may collaborate remotely using XML nets [S22].

Semantic modeling is a MDD approach that uses semantic technologies to automatically translate between the different abstraction levels in MDA [S37].

SOMA Bercovici et al. [S27] propose a MDD approach that complies with IBMs SOMA methodology. It complements all of the SOMA development processes with the added benefits of MDD.

Context aware Samyr and Slimane [S30] propose a MDD development approach that can be used to develop context aware SBAs. Context-aware applications have the added benefit of adapting to variable contextual parameters.

Web based Tavor et al. [S52] (2008) propose a web based model editor which is part of a model driven service engineering process. It has advantages such as ease of accessibility, zero client foot print and simple consumption.

MDD automates many of the proceses involved in software development such as analysis, design or implementation. However, the use of MDD does not negate the advantages of a software process model for the other processes in the development life-cycle. In conjunction with MDD it is often necessary to implement processes such as requirements engineering, operation and management, or adaptation all of which would benefit from the adherence to a suitable process model.

6.1.4. Deployment and Provisioning

Papazoglou and van den Heuvel [S41] describe service provisioning as a process that contains practices such as service metering, service rating and service billing. In other words it is an activity that allows service providers to monitor service usage and charge where appropriate. Provisioning may not be a concern when service producers and consumers are within a single organisation. Service deployment on the other hand involves making services available to service consumers which may include SBAs. There were 19 out of the 57 primary studies which contained details about the activities required for deployment or provisioning.

6.1.5. Execution and Monitoring

The execution and monitoring processes are concerned with the runtime activities of services and SBAs [S41]. Services need to be monitored during runtime in order to ensure that they are available and functioning correctly. Similarly SBAs need to monitor each of their component services to ensure that they are available and functioning correctly. Monitoring can be an automatic or manual process depending on the level of sophistication of the service or application. If the monitoring process for a SBA detects that a service becomes unavailable or its characteristics have changed to a state that are no longer suitable then it may be desirable to adapt the application. Adaptation in the context of (SOC), (covered in more detail in Section 6.1.7) is a process which either automatically or manually facilitates the re-configuration of services within SBAs to a more desirable level. Only four of the primary studies explicitly address Execution and Monitoring. However, nine primary studies address adaptation which is implicit in Execution and Monitoring. Curiously, only one of the papers that explicitly supports Execution and Monitoring also included processes supporting adaptation [S38].

The *Security* process is usually regarded as an ongoing concern for services and SBAs. As with traditional software systems, security has to be considered at each stage of the development life-cycle. Fernandez et al. [S25] propose the extension of a secure development methodology to the development of SBAs. Delessy and Fernandez [S9] propose a pattern-driven security process for SBAs.

6.1.6. Formal Methods

Formal Methods (FM) is a software engineering discipline that centres around mathematical techniques that can be applied to processes such as formal specification, development and verification of software systems [20]. These techniques are expensive to implement so they are usually reserved for safety or security critical applications. Due to the complexity of implementing these techniques they are applied sparingly. However, there have been many applications of FM proposed to aid in the analysis, design and construction of SBAs. FM techniques such as formal specification can be beneficial for the construction of SBAs as they will ensure that the complex applications are constructed correctly. The SENSORIA project [S38], amongst others, proposes a method for Augmenting Service Engineering with various types of formal methods for the analysis, transformation and dynamicity of

service-oriented systems. Formal methods are often used in conjunction with other software engineering techniques for the development of SBAs. For example, Nguyen et al. [S29] propose an SBA engineering methodology developed using techniques from formal methods as well as Model Driven Architecture (MDA). One of the problems with using Formal Methods during the development of SBAs or using Formal Methods in general is the steep learning curve involved. Since Formal Methods uses mathematics in order to validate or specify software applications this can be off-putting for practitioners.

6.1.7. Adaption

One of the research questions set out at the beginning of this review was whether or not the software process models encountered would facilitate the adaptation of SBAs. Adaptation has varying definitions depending on the context in which it is used. In terms of SOC, adaptation is most often defined as the enactment of an adaptation strategy in order to satisfy adaptation requirements [21]. Adaptation requirements can range from failure recovery, to the consumption of services from providers who offer better business propositions. Prior to the emergence of SOC, software adaptation has been defined as a discipline that provides techniques which enable the reuse of existing pieces of software to create systems with new functionality [22]. This type of adaptation is facilitated through the use of software adaptors which enable software component reuse. Prior to the concept of adaptation, software engineers have been using Component-Based Software Engineering (CBSE) [23] or Component-Based Development (CBD) [24] to reuse software components. Traditionally, it has been very difficult for software engineers to find software components that exactly meet their requirements so implementing CBSE/CBD has proven difficult. Software adaptation, however, solved this problem by facilitating the adaptation of components to meet the necessary requirements. Adaptation of SBAs can be achieved with greater ease than traditional component based systems because services implementations such as Web Services are loosely coupled and expose standard interfaces. Standard interfaces provide greater interoperability between service providers and consumers, thus alleviating the requirement for components to be adapted. SBAs can easily be adapted by exchanging their services for alternative services that expose the required functionality.

Services can also be adapted; as we have seen SBAs can be adapted by re-arranging their component services. Chang and Kim [S14] outline a service-oriented analysis and design process for the development of adaptable services. In certain circumstances it may also be desirable to adapt services in order to meet the requirements of SBAs. In a scenario where a service has many service clients which all may differ slightly, it may be beneficial for the service to adapt to meet each of the individual client's requirements. This, however, may also be viewed as suboptimal design as there can be many versions of the same service or many similar versions in operation at the same time. This can lead to problems with maintainability as well as problem resolution. A detailed narrative summary of each of the process models from the selected studies follows in Section 6.2.

6.2. RQ-2: Adaptation process models

In this section our second research question is addressed. Process models from the selected studies which contain adaptation related processes are critically reviewed with emphasis on the approaches taken as well as any validations that have been conducted. A summary of these process models is illustrated in Table 8 with detailed reviews in the following sub-sections.

6.2.1. ProDAOSS

Achbany et al. [S11] propose a Process for Developing Adaptable and Open Service Systems (ProDAOSS) as a plugin for the I-Tropos framework. I-Tropos is a comprehensive end to end agent-oriented development methodology which assigns a crucial role to requirements analysis and specification processes [25]. The ProDAOSS process model concentrates on three process areas. The first is Organizational Modeling and Requirements Engineering using a framework called FaMOs which models at the service level. The second contribution of ProDAOSS is an Architectural Design process which uses a Multi-Agent Software (MAS) architecture to facilitates the adaptation of SBAs. This aspect of ProDAOSS appears to focus on providing an architecture rather than specifically outlining the process details of how to implement it in an adaptable SBA. Finally, the last item described in ProDAOSS is a Detail Design process area that contains process details for a Reinforcement Learning Model which focuses on exploration and exploitation and a Probabilistic Reputation Model used to estimate the reputation of a service providers Quality of Service (QoS). The ProDAOSS process model has been applied to two case studies which contributes towards its validity.

6.2.2. PLASTIC

Autili et al. [S1] take a different approach to adapting SBAs where the services rather than the applications themselves adapt. When an application's context changes, adaptable SBAs should adapt by choosing a service more suited to the new context. However, if individual service adapt to meet context changes it negates the need to adapt the application itself. The advantage of service adaptability over SBA adaptation is that it is more straight forward to implement adaptable services, however adaptable SBAs are more flexible and allow the dynamic binding of services from alternative providers during run-time. The PLASTIC development process addresses adaptation in relation to context change only, and does not support other forms of adaptation. It consists of a design phase where all possible run-time contexts are models, where these model are used to automatically generate code skeletons that are manually coded to satisfy each context type. Then, at run-time, the code segment that best suits the current context is executed to achieve the desired functionality. It is a useful adaptation approach

Table 8
Adaptation specific process model summary.

Adaptation process model	References	Adaptation approach	Based on	Development approach	Evaluation
ProDAOSS	[S11]	SBA adaptation	I-Tropos	Agent-oriented development	Case study
PLASTIC	[S1]	Service adaptation	Bespoke	Other	None
Muliview SOAD	[S6]	Service adaptation	SOAD	MDD	None
BCDF	[S13]	SBA adaptation	Bespoke	Other	Worked example
Chang et al.	[S14, S16]	Service adaptation	SOAD	Other	None
CSOMA	[S32]	Service adaptation	SOMA	MDD	None
CSOA	[S30]	SBA adaptation	Bespoke	MDD	Worked example
Dino	[S38]	SBA adaptation	Bespoke	MDD	None

but is limited to context only adaptation and by the fact that the services adapt rather than the SBA itself.

6.2.3. Muliview SOAD

Kenzi et al. [S6] propose a service-oriented anaylsis and design (SOAD) development process with the same gaol as PLASTIC to adapt services rather than SBAs Therefore, their approach is subject to the same pros and cons as PLASTIC mentioned in the previous sub-section. Kenzi et al. take a different approach however, opting to for a Model Driven Development (MDD) process. The process focuses on providing a base or core set of functionalities for all service clients with specific functionality being provided by means of multiple views to clients with specific adaptation requirements. The services required to implement these base and multiview functionalities are modeled at design time. Then, these models are used to generate implementation code. Once the services are implemented at runtime the base and multiview services are combined in order to provide the necessary functionality to meet the contextual requirements of each client. This is a useful approach but it lacks the ability to adapt by binding to services which are provided by third parties.

6.2.4. BCDF

The Business Collaboration Design Framework (BCDF) [S13] is a rule based development approach for the development of adaptable SBAs which facilitate ad-hoc business collaborations based on predefined rules. They propose many types of rules such as business rules, operational rules and services rules, all of which are specified using an XML based standard called RuleML. Specifying the rules as RuleML allows them to be transmitted to third parties as well as for use in automated processes. This is a useful approach that can be employed for automatic adaptation as the rules that are encoded in RuleML may be parsed at runtime to guide SBA adaptation. One of the key challenges with this approach is the accurate representation of business rules. If the coded rules do not accurately reflect reality then the consequences of any subsequent adaptation may be undesireable. Orriens et al. [S13] illustrate the feasibility of BCDF with an example in the auto-insurance domain. Unfortunately the example focuses on how rules are encoded and interpreted, rather than how BCDF fits into the overall development process.

6.2.5. Chang et al.

Chang et al. [S14,S16] provide details of a SOAD approach which can be used to develop adaptable services. This is another of those approaches that achieves adaptability through the adaptation of services rather that the adaptation of SBAs. They motivate their work by hilighting the benefits of software reuse; pointing out that adaptable services are more suitable for reuse than static services. They explain that service variability is necessary when there are many clients each with slightly different functional requirements and contexts. The SOAD process that they present contains five phases: Defining Target Services, Defining Unit Services, Planning Service Components Acquisition, Acquiring Service Components and Acquiring Service Components. Throughout the development phases attention is paid to provision of adaptable services that can cope with variation in factors such as compositions, interfaces and logic. This is a useful service adaptation approach and has the added benefit of being very process focused with low level details of adaptation phases, processes, activities and artifacts.

6.2.6. CSOMA

The Contextual Service-Oriented Modeling and Analysis (CSOMA) [S32] development methodology is a SOMA approach aimed at designing services that can adapt during runtime in order to meet contextual adaptation requirements. The approach, which includes a UML modeling profile for creating Platform Independent Models (PIMs), proposed an outline for a wider MDD approach. The authors highlight that their PIM modeling profile has advantages of being platform independent (not tied to Web Services) as well as specifically addressing adaptation requirements at design time. The CSOMA development approach consists of process details for three layers which are also common with many MDD approaches: a Computational Independent Model (CIM), their UML based PIM profile, and a Platform Specific Model (PSM).

The CSOMA PIM modeling profile contains modeling constructs which can represent variability in business logic and orchestration logic. When used as part of a wider MDD approach, the modeled variability can be transformed into application code which can facilitate the runtime adaptation of business services. This MDD approach is useful because it hides the complexities of low level implementation technologies from system developers.

6.2.7. CSOA

Like the previous approach (CSOMA), CSOA [S30] is an MDD approach for the context-aware development, however, in this case CSOA relates to the development of context-aware SBAs rather than services. The approach employs viewpoint principles from the Object Management Groups (OMG) Enterprise Collaboration Architecture (EDOC-ECA). Among the many viewpoints used in the modeling process is an adaptation view which facilitate the modeling of Business Components, Contextual Components and Connectors which handle bindings and middleware activities between business and context components. An application of the approach is illustrated using a mobile GPS application as an example, where the application adapts to meet a contextual environment where the user is a wheelchair user. In this case the application adapts by putting more emphasis on map items such as pedestrian crossings which are necessary for wheel chair users while also providing voice input functionality for hands free use. This is a useful approach with the added benefit of providing adaptation at the application level rather than just at the service level. Unfortunately the approach focuses on specifying the MDD architecture and does not give much detail on how the approach fits into the overall development process.

6.2.8. Dino

SENSORIA was a large European project aimed at developing a comprehensive service engineering approach, Wirsing et al. [S38] describe it as:

> SENSORIA is developing a novel comprehensive approach to the engineering of service-oriented software systems where foundational theories, techniques and methods are fully integrated into pragmatic software engineering processes.

A key goal of this project was to facilitate the design of adaptable open ended service-oriented systems that can adapt at runtime to meet a changing environmental conditions or by optimizing an existing composition by selecting cheaper or better quality services. A deliverable from the project which attempts to meet these goals is a service broker engine called Dino [26] with an associated development methodology. The approach is based on MDD engineering create models in UML2 using a Mode profile. A Mode in terms of SOC abstract a set of services that contribute to a shared goal. They are particularly useful when modeling systems that self-heal, self-optimise and self-assemble.

This approach has all the advantage of the other MDD approaches such as the abstraction of low level interface details and service descriptions. However, the use of MDD has an associated learning curve which may be off putting to many developers and architects.

7. Discussion

The results of this review have given us a useful insight into the state of the art of the research in the area of service-oriented software process models. What is interesting to note is that out of all of the approaches reviewed, only a few of them [7],[5] appear mature enough for use in the development of real-life SBAs. Arsanjani et al.'s process model is the only one that is based on extensive empirical evidence. They claim that it was constructed based on the experiences of developing hundreds of service based projects. Unfortunately, in their publication they do not clearly show how the processes and activities from SOMA were elicited from the experiences on which it is based. Papazoglou et al.'s SLDC do not draw on empirical evidence but they have based their approach on RUP and CBSE which are both mature proven approaches. Even with the lack of empirical validation, the SLDC is a complete life-cycle model with enough detailed activities to guide the development of SBAs.

On reviewing the data collected in the review, specifically summaries, one of the strongest research themes in the complete dataset is Model Driven Development (MDD) of SBAs. Given the complex standards used in SBAs such as Business Process Execution Language (BPEL) and Web Service Definition Language (WSDL), many authors believe that the model driven approach is the only way to achieve a high level of quality and productivity while developing these systems.

7.1. Literature Gaps

There is a lot of opportunity for further research in this area. Most of the studies that were reviewed only address particular areas of the development life-cycle. Of the studies that attempt to address the entire life-cycle, few of them are validated with real-life scenarios. The process that are spread over many different process models and studies need to be consolidated in order to address the entire SBA development life-cycle. Then, once more complete process models have been developed, they need to be validated so that they can be adopted with confidence by SBA practitioners.

Other obvious areas of opportunity are the phases of the processes that are not addressed often in the literature. The most obvious of these processes are operation and management, and deployment and provisioning. Similarly, the type of research that has been completed in this area are primarily secondary studies. This lack of primary studies suggest that existing primary studies are being repeatedly cited producing similar secondary studies.

7.2. Limitations of the Review

As with SLRs, there are some limitations associated with this SLR. With SLRs in general there is often a lot of subjectivity involved in the study selection as well as the data extraction processes. In order to minimise the subjectivity involved in this study the reviewers strictly adhered to the instructions set out in the review protocol. The majority of the study selection and data extraction activities were conducted by the first author of this report. In order to test for reviewer bias a random selection of studies were reviewed and extracted by the second author and compared to the results of the first author. The difference between the results of the two authors were small enough not to warrant further investigation.

Another limitation of this study is how the search string was constructed and validated. This process often involves trial and error in order to strike the balance between the minimum set of representative results and getting thousands of results which would be impossible to process. During this study it was found that the initial set of terms used were too generic and returned far too many results. For example, using the term "process" alone generates thousands of results, however, using the term "development process" greatly reduces the number of results while keeping relevant studies. The search strings in this study were validated by ensuring that some well known relevant studies were returned with the results from the various electronic databases. The construction and validation of the search string was found to be one of the most critical factors in the design of the review protocol. A high quality search string provides a smaller set of relevant results which allows the reviewer to focus more effort on reviewing the relevant studies rather than spending a lot of time filtering irrelevant studies. The key is not to have an overly conservative search string which could risk the exclusion of relevant studies.

Two exclusion criteria used in this study which had a big impact on the number of results returned were the exclusion of technology-specific and application domain-specific process models. A decision was chosen to exclude these studies in an attempt to identify studies that contained more generic processes that can be used by anyone not just individuals from a specific domain or who employ a certain technology. The downside to this is that many useful approaches may have been excluded at the expense choosing only generic studies. A more positive observation from using these criteria is that many technology specific or domain specific approaches are specialisations of previously reported generic approaches which are already included in the review.

The final noteworthy limitation of this review is that grey literature such as technical reports or unpublished material are excluded from the study. From a search point of view, excluding this material make the SLR more straight-forward and repeatable but at the cost of potentially excluding valuable studies. It is common for process models to be reported in standards documents which are often excluded from research databases so, unfortunately, they are also excluded from this report.

8. Conclusions and future work

There are clearly a lot of studies in the public domain with a many process models that can be used to guide the development of SBAs. This review has identified and categorised the processes within those process models to give an overview of the entire research area. It has also identified studies that contain processes that can facilitate the adaptation of SBAs. This review is a useful resource for practitioners and researchers who want to find process models for specific SBA development processes or the entire SBA development life-cycle. It is a particularly useful resource for those interested in SBA adaptation.

Within the results it was found that there were some strong process themes emerging, such as MDD is a particularly popular theme. Another common approach was the modification of existing development methodologies to meet the needs of SOC. A promising indication from the results is that 8 process models include adaptation processes. This is a welcome result since adaptability is one of the key benefits of using a SOA.

One of they key findings within the study is that the published process models lack empirical validation. Existing approached while promising need to be strengthened with data from empirical research to prove their applicability in real-life scenarios. Future work planned for this research area is to codify the results of this review using content analysis as well as enhancing existing research with empirical data gathered through industry interaction.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme

FP7/2007-2013 under grant agreement 215483 (S-Cube) and has been supported, in part, by Lero – the Irish Software Engineering Research Centre, Science Foundation Ireland Grant No. 03/CE2/I303.1.

We would like to thank the reviewers and Dr. Sarah Beecham for their valuable comments.

Appendix A. Selected studies

[S1] M. Autili, L. Berardinelli, V. Cortellessa, A. Di Marco, D. Di Ruscio, P. Inverardi, et al., A development process for self-adapting service oriented applications, in: International Conference on Service-Oriented Computing (ICSOC 2007), 2007, pp. 442–448.

[S2] H. Wada, J. Suzuki, K. Oba, A feature modeling support for non-functional constraints in service oriented architecture, in: IEEE International Conference on Services Computing (SCC 2007), IEEE Computer Society, 2007, pp. 187–195.

[S3] R. Howard, L. Kerschberg, A framework for dynamic semantic web services management, International Journal of Cooperative Information Systems, 13 (2004) 441–485.

[S4] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.P. Richter, et al., A method for engineering a true service-oriented architecture, in: International Conference on Enterprise Information Systems (ICEIS 2008), Barcelona, Spain, 2008, pp. 272–281.

[S5] H. Lu, M. Chhabra, A methodology for agent oriented web service engineering, in: Pacific Rim International Workshop on Multi-Agents (PRIMA 2006), Springer, 2006, pp. 650–655.

[S6] A. Kenzi, B. El Asri, M. Nassar, A. Kriouile, A model driven framework for multiview service oriented system development, in: International Conference on Computer Systems and Applications, IEEE Computer Society, 2009, pp. 404–411.

[S7] S.K. Johnson, A.W. Brown, A model-driven development approach to creating service-oriented solutions, in: International Conference on Service-Oriented Computing (ICSOC 2006), Springer, 2006, pp. 624–636.

[S8] Z.H. Ren, B.H. Jin, J. Li, A new web application development methodology: web service composition, in: International Workshop on Web Services, E-Business, and the Semantic Web (WES 2003), 2004, pp. 134–145.

[S9] N.A. Delessy, E.B. Fernandez, A pattern-driven security process for SOA applications, in: Proceedings of the 2008 ACM Symposium on Applied Computing, ACM, Fortaleza, Ceara, Brazil, 2008, pp. 2226–2227.

[S10] L. Bahler, F. Caruso, J. Micallef, A practical method and tool for systems engineering of service-oriented applications, in: The 8th International Conference on Web Information Systems Engineering, Springer, 2007, pp. 472–483.

[S11] Y. Wautelet, Y. Achbany, J. Lange, M. Kolp, A process for developing adaptable and open service systems: application in supply chain management, in: International Conference on Enterprise Information Systems (ICEIS 2009), Springer, Milan, Italy, 2009, pp. 564–576.

[S12] M. Karam, S. Dascalu, H. Safa, R. Santina, Z. Koteich, A product-line architecture for web service-based visual composition of web applications, Journal of Systems and Software, 81 (2008) 855–867.

[S13] B. Orriens, J. Yang, M. Papazoglou, A rule driven approach for developing adaptive service oriented business collaboration, in: International Conference on Service-Oriented Computing (ICSOC 2005), 2005, pp. 182–189.

[S14] S.H. Chang, S.D. Kim, A service-oriented analysis and design approach to developing adaptable services, in: IEEE International Conference on Services Computing, 2007, pp. 204–211.

[S15] M. Meisinger, I.H. Kruger, A service-oriented extension of the V-Modell XT, in: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, IEEE, Tucson, Arizona, 2007, pp. 256–268.

[S16] C. Soo Ho, A systematic analysis and design approach to develop adaptable services in service oriented computing, in: Congress on Services (Services 2007), 2007, pp. 375–378.

[S17] S. Chang, S. Kim, A systematic approach to service-oriented analysis and design, in: International Conference on Product-Focused Software Process Improvement (PROFESS 2007), Springer, 2007, pp. 374–388.

[S18] D. Adamopoulos, An integrated development environment for synthesising web services, in: International Conference on E-Business Engineering (ICEBE 2008), 2008, pp. 776–779.

[S19] S. Lamparter, Y. Sure, An Interdisciplinary Methodology for building service-oriented systems on the web, in: IEEE International Conference on Services Computing, IEEE Computer Society, 2008, pp. 475–478.

[S20] W. Chengjun, Applying pattern oriented software engineering to web service development, in: International Seminar on Future Information Technology and Management Engineering (FITME 2008), 2008, pp. 214–217.

[S21] I. Krüger, M. Meisinger, M. Menarini, Applying service-oriented development to complex systems: BART case study, in: Monterey Conference on Reliable Systems on Unreliable Networked Platforms, Springer, 2007, pp. 26–46.

[S22] T. Karle, A. Oberweis, Collaborative model driven software development for SOA-based systems, in: International United Information Systems Conference (UNISCON 2008), Klagenfurt, Austria, 2008, pp. 189–200.

[S23] S. Mittal, Devs unified process for integrated development and testing of service oriented architectures, University of Arizona, 2007.

[S24] I. Christou, S. Ponis, E. Palaiologou, Experiences of Using the Agile Unified Process in the Banking Sector, Software, IEEE. PP (2009) 1–1.

[S25] E.B. Fernandez, P. Cholmondeley, O. Zimmermann, Extending a secure system development methodology to SOA, in: Database and Expert Systems Applications (DEXA'07), 2007, pp. 749–754.

[S26] L. Bocchi, J.L. Fiadeiro, A. Lapadula, R. Pugliese, F. Tiezzi, From architectural to behavioural specification of services, in: Proceedings of the Sixth International Workshop on Formal Engineering Approches to Software Components and Architectures (FESCA 2009), 2009, pp. 3–21.

[S27] A. Bercovici, F. Fournier, A. Wecker, From business architecture to SOA realization using MDD, in: Proceedings of the 4th European Conference on Model Driven Architecture: Foundations and Applications, 2008, pp. 381–392.

[S28] N. Protogeros, D. Tektonidis, A. Mavridis, C. Wills, A. Koumpis, FUSE: A framework to support services unified process, in: Enterprise Interoperability III: New Challenges and Industrial Approaches, Springer, 2008, pp. 209–220.

[S29] D. Nguyen, W. van den Heuvel, M. Papazoglou, V. de Castro, E. Marcos, GAMBUSE: a Gap Analysis Methodology for Engineering SOA-Based Applications, Conceptual Modeling: Foundations and Applications, 2009, pp. 293–318.

[S30] S. Vale, S. Hammoudi, Model driven development of context-aware service oriented architecture, in: The 11th IEEE International Conference on Computational Science and Engineering – Workshops, 2008, pp. 412–418.

[S31] J.K. Strosnider, P. Nandi, S. Kumaran, S. Ghosh, A. Arsanjani, Model-driven synthesis of SOA solutions, IBM Systems Journal, 47 (2008) 415–432.

[S32] K. Boukadi, L. Vincent, P. Burlat, Modeling adaptable business service for enterprise collaboration, in: IFIP WG 5.5 Working Conference on Virtual Enterprises (PRO-VE 2009), Springer, Thessaloniki, Greece, 2009, pp. 51–60.
[S33] E. Colombo, J. Mylopoulos, P. Spoletini, Modeling and analyzing context-aware composition of services, in: International Conference on Service Oriented Computing (ICSOC 2005), Springer, 2005, pp. 198–213.
[S34] M. Rychly, P. Weiss, Modeling of Service Oriented architecture – from business process to service realisation, in: International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2008), 2008, pp. 140–146.
[S35] M. López-Sanz, C.J. Acuóa, C.E. Cuesta, E. Marcos, Modelling of service-oriented architectures with UML, in: Proceedings of the 6th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2007), Elsevier, 2008, pp. 23–37.
[S36] R. Huigui, Z. Ning, J. Min, W. Jiaxin, Research on service-oriented framework of interface prototype driven development, in: International Conference on Computer Science and Software Engineering, IEEE Computer Society, 2008, pp. 552–557.
[S37] C. Pahl, Semantic model-driven architecting of service-based software systems, Information and Software Technology, 49 (2007) 838–850.
[S38] M. Wirsing, M. Hoelzl, L. Acciai, F. Banti, A. Clark, A. Fantechi, et al., SENSORIA patterns: augmenting service engineering with formal analysis, transformation and dynamicity, in: Leveraging Applications of Formal Methods, Verification and Validation, Third International Symposium, 2008, pp. 170–190.
[S39] M. Wirsing, M. Hoelzl, N. Koch, P. Mayer, A. Schroeder, Service engineering: the sensoria model driven approach, in: Proceedings of Software Engineering Research, Management and Applications (SERA 2008), Prague, Czech Republic, 2008, pp. XIV–XVI.
[S40] S. Kambhampaty, Service oriented analysis and design process for the enterprise, in: Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science, World Scientific and Engineering Academy and Society (WSEAS), Venice, Italy, 2007: pp. 366-371.
[S41] M.P. Papazoglou, W.V.D. Heuvel, Service-oriented design and development methodology, International Journal of Web Engineering and Technology, 2 (2006) 412–442.
[S42] G. Engels, M. Assmann, Service-Oriented Enterprise architectures: evolution of concepts and methods, in: 12th International IEEE Enterprise Distributed Object Computing Conference, IEEE Computer Society, 2008, pp. xxxiv–xliii.
[S43] A. Arsanjani, A. Allam, Service-Oriented Modeling and architecture for realization of an SOA, in: IEEE International Conference on Services Computing, 2006, pp. 521–521.
[S44] H. Karhunen, M. Jantti, A. Eerola, Service-oriented software engineering (SOSE) framework, in: Proceedings of the International Conference on Services Systems and Services Management, 2005, pp. 1199–1204 Vol. 2-1199–1204 Vol. 2.
[S45] A. Erradi, S. Anand, N. Kulkarni, SOAF: An architectural framework for service definition and realization, in: International Conference on Services Computing (SCC 2006), 2006, pp. 151–158.
[S46] D. Dori, SODA: Not just a drink! From an object-centered to a balanced object-process model-based enterprise systems development, in: Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, 2006, pp. 3–14.
[S47] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, K. Holley, SOMA: a method for developing service-oriented solutions, IBM Systems Journal, 47 (2008) 377–396.
[S48] I.H. Kruger, R. Mathew, Systematic development and exploration of service-oriented software architectures, in: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture, 2004: pp. 177–187.
[S49] J. Koehler, R. Hauser, J. K++ster, K. Ryndina, J. Vanhatalo, M. Wahler, The Role of Visual Modeling and Model Transformations in Business-driven Development, in: Proceedings of the Fifth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2006), Elsevier, Budapest, Hungary, 2008: pp. 5-15.
[S50] B. Demchak, C. Farcas, I.H. Kruger, The Treasure Map for Rich Services, in: IEEE International Conference on Information Reuse and Integration, 2007: pp. 400-405.
[S51] M. Deubler, J. Grunbauer, G. Popp, G. Wimmel, C. Salzmann, Tool supported development of service-based systems, in: 11th Asia-Pacific Software Engineering Conference, IEEE Computer Society, Busan, Korea, 2004: pp. 99-108.
[S52] A. Anaby-Tavor, D. Amid, A. Sela, A. Fisher, K. Zhang, O.T. Jun, Towards a Model Driven Service Engineering Process, in: IEEE Congress on Services – Part I, 2008: pp. 503–510.
[S53] M. Deubler, J. Grunbauer, G. Popp, G. Wimmel, C. Salzmann, Towards a model-based and incremental development process for service-based systems, in: Proceedings of the IASTED International Conference on Software Engineering, 2004: pp. 183–188.
[S54] A. Ivanyukovich, G.R. Gangadharan, V. D'Andrea, M. Marchese, Towards a service-oriented development methodology, Journal of Integrated Design and Process Science, 9 (2005) 53–62.
[S55] V. De Castro, E. Marcos, R. Wieringa, Towards a service-oriented MDA-based approach to the alignment of business processes with IT systems: from the business model to a web service composition Model, International Journal of Cooperative Information Systems. 18 (2009) 225–260.
[S56] V. Bicer, S. Lamparter, Y. Sure, A.H. Dogru, Towards an interdisciplinary methodology for service-oriented system engineering, in: 24th International Symposium on Computer and Information Sciences, 2009: pp. 486–491.
[S57] C. Hong-Mei, Towards Service Engineering: Service Orientation and Business-IT Alignment, in: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 2008: pp. 114–114.

References

[1] W.S. Humphrey, Managing the Software Process, Addison-Wesley Professional, 1989.
[2] W.W. Royce, Managing the development of large software systems, in: Proceedings of IEEE Wescon, vol. 26, 1970, p. 9.
[3] C.P. Team, CMMI for Development, Version 1.2. Tech. Rep. CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, 2006.
[4] I. 1/SC, ISO/IEC 15504-5:2006, Information Technology – Process Assessment – Part 5: An Exemplar Process Assessment Model, International Organization for Standardization, 2007.
[5] M.P. Papazoglou, W.V.D. Heuvel, Service-oriented design and development methodology, International Journal of Web Engineering Technology 2 (4) (2006) 412–442.
[6] P. Kruchten, The Rational Unified Process: An Introduction, third ed., Addison Wesley, 2003.
[7] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, K. Holley, SOMA: a method for developing service-oriented solutions, IBM Systems Journal 47 (3) (2008) 377–396.
[8] S. Durvasula, M. Guttmann, A. Kumar, J. Lamb, T. Mitchel, B. Oral, Y. Pai, T. Sedlack, H. Sharma, S. Sundaresan, Introduction to Service lifecycle. SOA Practitioner's Guide. Part 3, 2007.
[9] S. Lane, Q. Gu, P. Lago, I. Richardson, Adaptation of service based applications: a maintenance process? Tech. Rep. Lero-TR-2010-08, Lero, the Irish Software Engineering Research Centre, University of Limerick, 2010

[10] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering. Tech. Rep., Keele University, 2007.
[11] J.C. Derniame, B.A. Kaba, D. Wastell, Software Process: Principles, Methodology, and Technology, Springer Verlag, 1999.
[12] S-Cube Knowledge Model, 2010. <http://www.s-cube-network.eu/km.
[13] Q. Gu, P. Lago, Exploring service-oriented system engineering challenges: a systematic literature review, Service Oriented Computing and Applications (2009) 1–18.
[14] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Information and Software Technology 50 (9–10) (2008) 833–859.
[15] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering, 2008.
[16] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using mapping studies in software engineering, in: Proceedings of PPIG, 2008, pp. 195–204.
[17] G.W. Noblit, R.D. Hare, Meta-ethnography: Synthesizing Qualitative Studies, SAGE, 1988.
[18] M. Wooldridge, N.R. Jennings, Intelligent agents: theory and practice, Knowledge Engineering Review 10 (2) (1995) 115–152.
[19] X. Cao, H. Miao, Y. Chen, Transformation from computation independent model to platform independent model with pattern, Journal of Shanghai University (English Edition) 12 (6) (2008) 515–523.
[20] NASA LaRC Formal Methods Program: What is Formal Methods?. <http://shemesh.larc.nasa.gov/fm/fm-what.html>.
[21] S. Benbernou, State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs, S-Cube Consortium, 2008.
[22] C. Canal, J.M. Murillo, P. Poizat, Software adaptation, L'object 12 (1) (2006) 9–13.
[23] W. Kozaczynski, G. Booch, Component-based software engineering, IEEE Software 15 (5) (1998) 34–36.
[24] P. Herzum, O. Sims, Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, John Wiley & Sons, Inc., 2000.
[25] F. Giunchiglia, J. Mylopoulos, A. Perini, The Tropos software development methodology: processes, models and diagrams, in: Agent-Oriented Software Engineering, LNCS 2585, Springer Verlag, 2002, pp. 162–173.
[26] H. Foster, A. Mukhija, D. Rosenblum, S. Uchitel, A model-driven approach to dynamic and adaptive service brokering using modes, in: Service-Oriented Computing – ICSOC 2008, 2008. pp. 558–564.

Appendix C

Exploratory Study

Noname manuscript No.
(will be inserted by the editor)

Towards a Framework for the Development of Adaptable Service-Based Applications

Stephen Lane · Qing Gu · Patricia Lago · Ita Richardson

Received: date / Accepted: date

Abstract Service-oriented computing is a promising computing paradigm which facilitates the composition of loosely coupled and adaptable service-based applications. Unfortunately, this new paradigm does not lend itself easily to traditional software engineering methods and principles due to the decentralized nature of software services. The goal of this paper is to identify a set of engineering activities that can be used to develop adaptable service-based applications. Rather than focusing on the entire service-based application development life-cycle, this paper will focus on adaptation specific processes and activities and map them to an existing high-level SBA development life-cycle. Existing software engineering literature as well as research results from service engineering research are reviewed for relevant activities. The result is an adaptation framework that can guide software engineers in developing adaptable service-based applications.

Keywords Service-Based Application Life-Cycle · Service-Based Application Adaptation · Maintenance Process · Software Process

Contents

Stephen Lane and Ita Richardson
Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland
E-mail: stephen.lane@lero.ie
E-mail: ita.richardson@lero.ie

Qing Gu and Patricia Lago
Dept. of Computer Science, VU University Amsterdam, The Netherlands
E-mail: q.gu@vu.nl
E-mail: p.lago@vu.nl

1 Introduction

Service-Based Applications (SBAs) are software applications which are composed of software services and those services may be owned by the application developers or by a third party. The ability of SBAs to adapt in order to choose more suitable services is a desirable attribute. When services are provided by a third party there is often no guarantee that they will be available when required. Another concern is that their functional or non-functional parameters such as cost or quality may change without notice. SBAs may be required to adapt for many reasons such as business agility or failure recovery. When adpating it may be desirable to replace or re-configure services within an SBA through self-adaptation or through manual adaptation.

In order for SBAs to be adaptable there are both technical and software process challenges. The technical challenges refer to the implementation details of the adaptation mechanisms, while the software process challenges refer to the way in which adaptation affects the application's development life-cycle. The focus of this paper is the software process challenges. We address the process challenges by eliciting adaptation related activities from existing service literature and elicit adaptation support activities from the software maintenance lit-

erature. The maintenance process was chosen as a source of activities because of the similarities that can be drawn between software adaptation and software maintenance.

Since we are only focusing on adaptation related activities in this paper, they will need to be used in conjunction with a life-cycle model that addresses the remaining areas of the SBA development life-cycle. The life-cycle model that we will use is the S-Cube [1] reference life-cycle. S-Cube is a European consortium that conducts research on software services and systems of which the the authors of this paper are contributors. One of the aims of the S-Cube consortium is to develop a life-cycle for the development of adaptable SBAs. The S-Cube reference life-cycle is a skeleton life-cycle model that will be populated with tools, techniques and methods by S-Cube participants. This paper is one such contribution relating specifically to adaptation activities for the development of adaptable SBAs.

The S-Cube life-cycle consists of two cycles (see Figure 1). In the evolution cycle, shown on the right hand side of the figure, the software engineer concentrates on the development of the SBA through the traditional stages of requirements engineering, design, construction and deployment, while also focusing on quality assurance. However, as adaptation is a desirable feature in SBAs, the software engineer must also consider how the application will adapt during its life-time. The adaptation cycle, shown on the left hand side, ensures that the software engineer follows the processes: *Identify adaptation needs*, *Identify adaptation strategy* and *Enact adaptation*. Within the complete life-cycle, there must also be a focus on *Operation and management* and *Deployment and provisioning*.

In related work, Oreizy et al [2] propose a development life-cycle for adaptable component-based applications with both development time and runtime cycles. This paper proposes a development process and a supporting application architecture. However, since this approach is not specifically focused on service-oriented computing and is not entirely process focused the S-Cube life-cycle was chosen as a basis for this paper.

The S-Cube life-cycle as presented here is a conceptual framework and it presents the processes that need to be followed in order to develop adaptable SBAs. It does not, however, present the activities that need to be followed within each of the processes when developing SBAs. The activities required for many of the processes within the evolution cycle of the life-cycle are currently being investigated by participants of the S-Cube project [3]. The aim of this paper is to develop the adaptation related activities for the the life-cycle. The adaptation cycle is the major difference between this life-cycle and standard software engineering life-cycles such as waterfall [4], or spiral [5] life-cycle models.

Adaptation of SBAs is different from maintenance in traditional software engineering in that it is a less inexpensive process that usually involves the substitution of component services compared to expensive maintenance which usually involves rewriting parts of an application. However, because at a basic level both adaptation and maintenance involve the modification of an application, similarities can be drawn between the two.

Once a set of activities have been developed for each of the processes of the S-Cube life-cycle it will provide a useful guide for software engineers intending to build adaptable SBAs. In order to contribute to this life-cycle model, we elicit adaptation activities from existing service-based development approaches. We also elicit activities from software maintenance literature that can support SBA adaptation. The maintenance process was chosen as a source of practices as it bears resemblance to the SBA adaptation process. By taking this approach existing engineering practices are reused in a novel way to fulfill the adaptation cycle of the S-Cube life-cycle. The use of engineering practices from the maintenance process ensure that a level of quality assurance is built into the life-cycle.

This paper is organised as follows: Section 2 describes the motivation for carrying out this work, followed by Section 3 which describes our research methodology. Section 4 provides some background information on SBA adaptation and service engineering process models. The remainder of the paper contains the body of the work in Sections 5 and 6, a case demonstrating an application of the framework in Section 8 followed by conclusions in Section 9.

2 Motivation

The adaptation of SBAs is important because they are meant to operate in open-world contexts. Services are dynamically integrated in larger service compositions and/or SBAs, whose structure, features, location and qualities are unknown when they are developed. Their execution environments are distributed, non-deterministic, unpredictable, heterogeneous and highly dynamic. All these variables demand that SBAs be highly adaptable, and that they are developed using a software development process that accommodates their adaptation

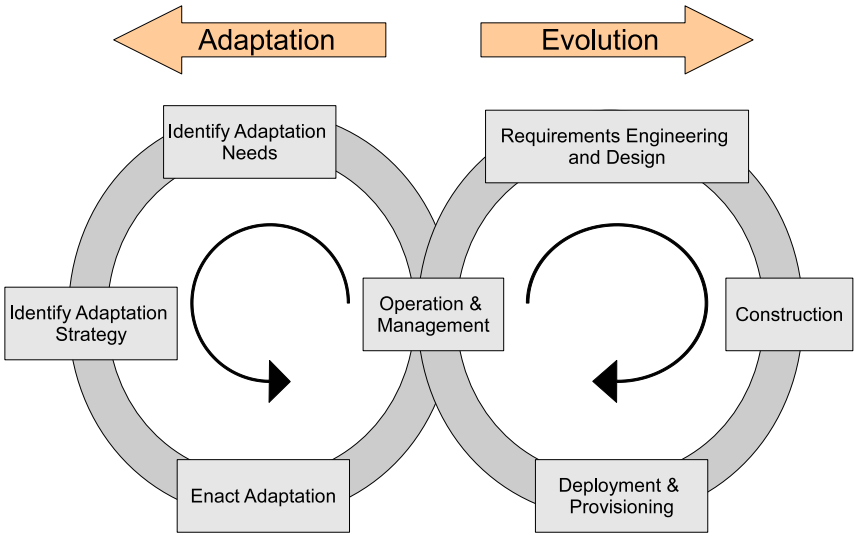

Fig. 1 The Life-Cycle of Adaptable SBAs.

requirements. Implementing a best practice software development process ensures quality through the optimisation of the engineering processes and methods during the development life-cycle.

Service-Oriented Computing (SOC) promises companies the ability to conduct ad-hoc business collaborations that are supported by software services that can be orchestrated to meet the business requirements of each participating company. Software supported ad-hoc business collaborations are not new but SOC promise to provide greater power and flexibility than predecessors such as electronic data interchange, distributed components and email. The benefit of SOC is that it is platform and technology neutral with the lowest common denominators usually XML and HTTP capabilities. In order to realise the benefits of SOC it is important for SBAs to be able to adapt to meet changing business needs and service quality characteristics as previously mentioned.

The problem with existing development approaches is that they do not suit development SBA that are composed of distributed services with runtime adaptation capabilities. Existing development approaches usually do not have any processes to support runtime adaptation. The closest engineering process to adaptation is the maintenance process. Although the maintenance process can provide some level of support to adaptation it is clear that specific adaptation processes and activities are necessary.

A key benefit of adaptation is that it facilitates agility, reliability and resilience of applications. These attributes are particularly important for applications operating within critical domains. Therefore, a framework that can facilitate adaptation is valuable for developing applications within these domains. In their Evolving Critical Systems white paper [6], Lero researchers discuss four types of criticality: safety-critical, mission-critical, business-critical and security-critical. Failure of safety-critical systems can cause serious injury or even death to individuals. Such cases normally come under the auspices of regulation bodies. These include the medical device, automotive and financial domains, where software is becoming more prevalent and regulations are inherent within the domain. For example, development of software for medical devices is governed in many jurisdictions by the U.S. Food and Drugs Administration (FDA). In Europe, major car companies - Audi, BMW group, DaimlerChrysler, Porsche and Volkswagen - have come together to form the Herstellerinitiative Software (HIS) process assessment working group [7]. One of the aims of this group is to achieve standardization, and they require that suppliers of software follow particular process models. Another view of criticality to be considered is that of business-critical. Of course, for organisations depending on regulation, not achieving certification will result in the company being prevented from entering or continuing in a particular market. However, systems down-time can also be business-critical. This would be the case, with a company such as Amazon [8] who sells much of its product on the web. In this case, the reliability of the service is important because down-time could cause significant loss of business.

Given the growth and increased availability of services, many SBAs are being used in these critical environments. These systems are expected to be adaptable, and, as software engineers, we need

to ensure that during the adaptation cycle of the SBA, the software continues to be operationally successful. To do this, software engineers need adaptation activities to be defined.

3 Research Method

The aims of this paper are to determine the design and runtime activities required for SBA adaptation and to identify suitable support activities that can be used to supplement them. These activities are then mapped to the S-Cube SBA development life-cycle at the point where they can be enacted. As illustrated in Figure 2, the work is divided into two phases. The adaptation activities are identified in Phase I. These activities are then mapped to the S-Cube life-cycle in Phase II. The end result is a framework of adaptation activities that can be used to guide software practitioners in the development of adaptable SBAs.

To determine the activities required for SBA adaptation we examined 16 service-oriented engineering approaches. We found that 5 of these approaches supported adaptation in one way or another. Even if the approaches did not specifically support runtime adaptation they were still searched for activities that are implicit to adaptation. For example, in self-adapting systems monitoring needs to occur in order to trigger adaptation. Although monitoring is not directly related to adaptation it is classified as an adaptation activity because it is needed by the application to adapt. Activities were classified as adaptation related if they could be related to the adaptation processes defined in the reference life-cycle proposed by the S-Cube consortium [1].

The set of disjointed adaptation activities identified were grouped together into categories of related activities. These categories were based on high-level adaptation activities that were identified in several technical reports from the S-Cube consortium (CD-JRA-1.1.2[1], CD-JRA-1.2.1[9], CD-JRA-1.2.2 [10]).

In order to satisfy the the second part of the research objective, identification of adaptation support activities, we examined relevant maintenance standards and process reference models. This established adaptation support activities that could be used to supplement the core activities identified in the previous step.

Once the activities were identified they were mapped to the skeleton SBA development life-cycle proposed by S-Cube. This mapping shows the activities in context and lays the foundations for a process model that can be used for developing adaptable SBAs.

4 Background

4.1 SBA Adaptation Definitions

Within the context of SBAs, adaptation is the modification of an application in order to satisfy adaptation requirements [9]. There are many adaptation requirements that can be desirable in SBAs, for example, the facilitation of interoperability amongst services [11], the optimisation of Quality of Service (QoS) [12] or the implementation of failure recovery [13]. SBA adaptation may involves the substitution, replacement, re-configuration or removal of component services from a SBA. Once adaptation requirements have been determined, it is then necessary to create an adaptation strategy. After the adaptation strategy has been developed, it will then be possible to enact the adaptation.

This is in contrast to the evolution of SBAs which refers to the initial requirements, design, implementation and operation of SBAs. In order to appropriately determine whether or not adaptation is required, it is useful to monitor the execution of SBAs. Monitoring can be done automatically by an application or can be achieved manually by reviewing error logs. There have been many monitoring frameworks proposed. Pistore *et al* (2004) [14] propose a methodology for the monitoring of web services based applications, so they can be adapted if an error occurs of if QoS requirements are not met.

Adaptation strategies depend on many factors. One such factor is whether adaptation will be dynamic or static. Static adaptation involves the adaptation logic being hard coded into the initial SBA implementation while dynamic adaptation allows adaptation logic to be introduced or altered at runtime. Modifying the adaptation logic for a SBA with static adaptation requires that the application code is changed during a maintenance or evolution cycle. Dynamic adaptation, on the other hand, allows the introduction of new adaptation logic or the reconfiguration of existing adaptation logic at runtime.

Adaptation of an SBA can by partially or fully automatic. A scenario where adaptation is partially automated is where a service becomes unavailable requiring an actor to choose from alternative services using functionality built-in to a SBA. In a fully automatic SBA this substitution could be enacted automatically by the application based on the QoS or availability of alternative services.

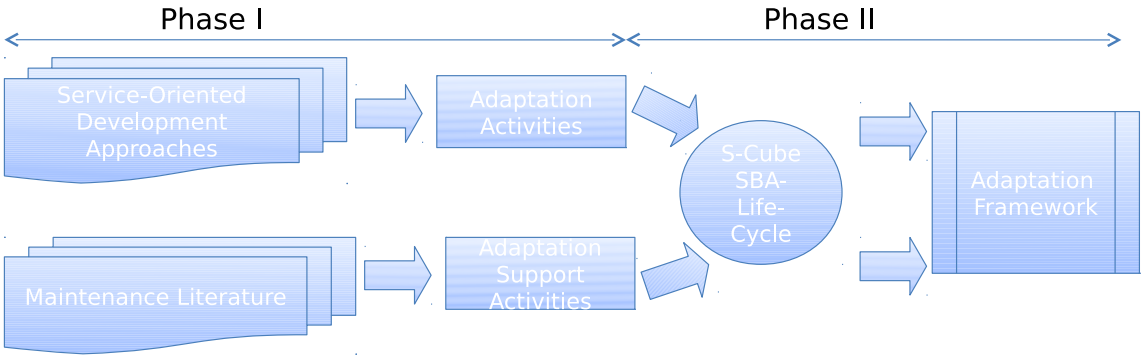

Fig. 2 Research Methodology

4.2 Software Maintenance Definitions

Many software engineering reference life-cycles and assessment models do not make direct reference to software maintenance. We have observed that there is little or no coverage of the maintenance process in the major assessment models despite the fact that software maintenance can take up to 60 percent of the time [15] and 70 percent of the budget [16] of a software project. April et al [17] propose a Software Maintenance Maturity Model (SMMM) which can be used as an add-on to the CMMI^{TM}. It takes best practice processes and activities from a variety of sources such as ISO/IEC 14764, IEEE 1219, ISO/IEC 12207, CMMI^{TM}and SWEBOK [18] in order to construct the model.

Software maintenance has a variety of definitions. However, most agree that it is the process of modifying software after initial delivery. The following list outlines the five most recognised types of software maintenance [19] [20] [21]:

- *Perfective Maintenance* is performed to improve performance or maintainability.
- *Corrective Maintenance* is carried out in response to system failures.
- *Adaptive Maintenance* is carried out in response to a change in operating environment or in responce to new functionality requirements.
- *Preventive Maintenance*: is maintenance carried out in a system to detect future errors in a software product.
- *Emergency Maintenance*: is unplanned maintenance that is carried out in order to keep a system operational.

4.2.1 Gap in Traditional Software Engineering

When comparing the engineering of SBAs to the engineering of traditional software applications, the focus of engineering SBAs is shifted to developing compositions of services, the control of services is passed from their users to their owners, and the ability of adapting to ever-changing requirements become more important. Due to the different focus and additional requirements, traditional software engineering approaches are no longer sufficient for engineering SBAs.

In particular, the ability to be self-adaptable is an important research topic in the service development community. We propose the following adaptation processes which are missing from the software engineering literature, each of the processes are based on similar software maintenance processes:

- *Perfective Adaptation* aims at improving or optimizing the quality attributes of a SBA even it runs correctly. This corresponds with *Perfective Maintenance*.
- *Corrective Adaptation* aims at removing any faults in the behavior of a SBA. This corresponds with *Corrective Maintenance*.
- *Adaptive Adaptation* modifies a SBA when its execution environment changes. This corresponds with *Adaptive Maintenance*.
- *Preventive Adaptation* aims at preventing potential or possible future faults before they occur. This corresponds with *Preventive Maintenance*.
- *Extending Adaptation* extends a SBA by adding new functionalities as required. To an extent, this corresponds with *Emergency Maintenance* in that adding new functionalities that are required during the execution of a SBA can be seen as unplanned maintenance activities.

5 PHASE I: Identifying Adaptation Activities

In this section we present the activities that can be used to develop adaptable SBAs which we identified in the software and service engineering liter-

ature. The activities may be involved directly in adaptation or they may be adaptation supporting activities. The latter, while not essential for adaptation, provide for support activities, such as change management. These support activities add to the overall quality of adaptable SBAs.

5.1 Adaptation Activities Categories

Table 1 lists high-level conceptual adaptation activities proposed by the S-Cube consortium for the adaptation of adaptable SBAs. These activities were used to categorise the adaptation activities identified in the service-engineering literature. The three S-Cube deliverables examined were:

- CD-JRA-1.1.2 Separate Design Knowledge Models for Software Engineering and Service Based Computing [1]
- PO-JRA-1.2.1 State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs [9]
- CD-JRA-1.2.2 Taxonomy of Adaptation Principles and Mechanisms [10]

5.2 Adaptation Activities from Service-Oriented Engineering Approaches

There have been many software development processes and life-cycles proposed for the development of SBAs as well as their underlying services. Many of these proposed approaches do not take the adaptation of SBAs into consideration [1]. Several approaches such as those proposed by Cortellessa et al [22] or Adil kenzi et al [23] include adaptation as a primary concern when developing services. However, these approaches are aimed at the development of services rather than compositions of services required by SBAs.

5.2.1 Service-Oriented Engineering Approaches

For the research presented here, we analysed 16 SOA approaches, and note that only five approaches explicitly mentioned some activities or tasks that are related to adaptation. These five approaches are presented in this section.

ASTRO [24] is a toolset that is made up from four component tools: WS-gen, WS-mon, WS-console and WS-animator. The aim of ASTRO is to support the automated composition of distributed business processes. Distributed business processes are represented as distributed software services, and these services can automatically be composed with the ASTRO tools to make a useful combined business process. The WS-gen tool is used to generate business process or service compositions by taking BPEL4WS as input and generation a composition based on the BPEL4WS specification. BPEL4WS is a Business Process Execution Language tailored to meet the needs of Web Services. WS-mon is a monitoring tool which is used to implement and deploy monitors to monitor the composed business processes. The WS-console tool is a front end which displays the status of the monitors deployed by the WS-mon tool and the final tool WS-animator is a graphical tool which allows the execution of the composed services/processes. ASTRO facilitates service composition which makes it a suitable candidate to look at for service adaptation activities.

The BEA reference life-cycle [25] outlines the activities for each of the following SBA lifecylce processes: Requirements and Analysis, Design, Service Development and IT Operations. For each of these processes it looks at the concerns such as actors, tools, deliverables, key considerations, recommended process and best practices. The life-cycle also has a business dashboard which monitors the life-cycle as it progresses. Along with the dashboard the life-cycle had a governance process which promotes interoperability, discoverability and standardisation of service technologies. To some extinct the BEA life-cycle caters for adaptability as it provides service monitoring, runtime correctness analysis and operational management activities.

Chang [26] proposes a process model which focuses on developing highly adaptable web services. It follows the sequence of steps specified in the SOAD [27] framework, namely: service *identification*, service *specification* and service *realisation*. The process model contains six processes each of which contain several activities. The processes are: analyzing target services, defining unit services and compositions, planning for acquiring service compositions, acquiring service components, developing service adapters and verifying service components. Each of the processes are targeted at the end result of developing adaptable web services. Similarly, each of the processes refer to one or more of the key artifacts in SOAD. The process model, although concise, addresses a lot of key concerns relating to adaptable services.

The Web Services Development Life Cycle Methodology (SLDC) [28] is influenced by several established life-cycles such as RUP [29], CBD [30] and BPM [31]. The life-cycle contains one preparatory planning process and eight other incre-

Table 1 Adaptation Activities from S-Cube Deliverables

Activity	Description
Define adaptation requirements	Identify the aspects of the SBA model that are subject to change, and what the expected outcome of the adaptation process is.
Define requirements to the monitoring subject	In order to satisfy the adaptation requirements, this practice focuses on specifying what artifacts are expected to be monitored.
Define monitored property	Specify which properties of the monitoring subject should be monitored.
Provide monitoring functionality	Monitoring functionalities that satisfy the monitoring requirements are provided through monitoring realization mechanism.
Collect monitoring results for adaptation	Results of monitoring are collected and analyzed.
Trigger adaptation	Evaluate the results from the monitoring analysis against adaptation requirements. If the need for adaptation is identified, send a request to trigger adaptation process.
Design adaptation strategy	Design the ways through which the adaptation requirements are satisfied.
Select adaptation strategy	Decide which particular adaptation strategy to be chosen based on the specific adaptation needs.
Perform adaptation	The actual adaptation process is performed through adaptation realization mechanisms based on the selected adaption strategy.

mental processes: Analysis, Design, Construction, Testing, Provisioning, Deployment, Execution and Monitoring. Along with the life-cycle, the methodology contains a number of principles such as service coupling, service cohesion and service granularity that aid in the development of SBAs. The SLDC methodology contains adaptation specific activities such as Quality of Service (QoS) monitoring and alerts for compliance failures.

The SeCSE methodology [32] is a set of functional areas and processes that focus on service-centric engineering, service engineering and service acquisition. The methodology also provides practitioners with the information required to adopt the various tools and methods developed by the SeCSE consortium. The SeCSE methodology is conveniently divided into two sections: design time processes and run-time processes. Design time processes contain many of the traditional software engineering processes such as analysis, design and development, while the run-time processes contain mostly service centric processes such as service binding/rebinding, run-time service composition and recovery management. Processes such as run-time service composition and service monitoring illustrate that the SeCSE methodology was designed with adaptation in mind.

5.2.2 Adaptation Activities Identified

Having reviewed these five approaches in detail the activities encountered relating to adaptation or monitoring were recorded. We included monitoring activities because adaptation cannot take place without monitoring, so monitoring is sub-process of adaptation. The activities are summerised in Table 2. They are categorised based on the activity categories identified in Section 5.1. We include four activities in an evolution activity category. These activities, while not directly involved with run-time adaptation, need to be carried out during SBA evolution in order to facilitate runtime adaptation.

The **Astro** toolset contains a monitoring tool which facilitates the two adaptation activities: *Monitor message sequences amongst services and its partners* and *Detect protocol violations*. The activity *Monitor message sequences amongst services and its partners* monitors messages exchanged between services and service consumers which could be used as an adaptation trigger. *Detect protocol violations* monitors whether service consumers behave as expected, if they do not, the monitoring activity could also trigger adaptation.

The **BEA** life cycle also contains monitoring related activities that could trigger adaptation, the *Define KPIs and management policies* activity could be used to determine which properties should be monitored, while *Monitor service, application, middleware, OS, hardware, and network* describes the monitoring of services and other system components.

Chang's approach contains two adaptation related activities: *Specifying service decision model*

Table 2 Adaptation Activities from Service-Oriented Engineering Approaches

Define adaptation requirements
Define requirements to the monitoring subject
SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs
Define monitored property
SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
Provide monitoring functionality
ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services
Collect monitoring results for adaptation
ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics
Trigger adaptation
SeCSE: Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Design adaptation strategy
Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters
Select adaptation strategy
Perform adaptation
SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery
Evolution Activities
BEA: Requirements and analysis stage - define KPIs and management policies SeCSE: Requirements and analysis stage - identify the service properties to specify SeCSE: Service deployment - insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Service deployment - deploy the monitoring rules and recovery policies within the monitoring system

aims at specifying the variability between available services and expected services. *Designing Service Adapters* aims at bridging the variability between service providers and consumers by allowing services to be dynamically adapted.

SDLC defines five adaptation related activities which revolve around the monitoring of quality attributes and alerting system users when they exceed predefined SLAs: *Gather QoS metrics on the basis of SLAs (Service Level Agreements)* refers to the collection of quality attribute data for monitoring, *Set warning thresholds and alerts for compliance failures* refers to the setting of threshold values for the monitored quality attributes, *Monitor workloads* refers to the monitoring of system utilisation, if utilisation is high and response times are affected then the service provider may have to take the appropriate actions to ensure SLAs are met. *Readjust service weights for request queues* refers to the re-evaluation of SLAs if they are not being met due to high demand or utilisation. *Evaluate SLA QoS metrics* involves the comparison of QoS metrics to predefined SLAs.

The **SeCSE** approach contains many detailed activities relating to the monitoring (*Monitor services*, *Specify monitoring rules according to the adopted SeCSE monitoring language*) and runtime adaptation (*Runtime Service Discovery*) of SBAs. It contains two activities which support corrective adaptation: *Service deployment: insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description* refers to the implementation of monitoring mechanisms, while *Recovery management: identify, by*

looking at the monitoring data, the needs for a recovery action refers to the runtime corrective adaptation of a SBA. *Service specification: identify the service properties to specify* states that the service properties to be monitored are determined during the service specification phase of development. Finally *Service deployment: deploy the monitoring rules and recovery policies within the monitoring system* states that the appropriate monitoring mechanism is deployed during the deployment phase.

5.3 Adaptation Activities from Maintenance Process Models

We also identified activities from the software maintenance process as useful for the adaptation of SBAs. There are many software maintenance processes, definitions, models and standards encountered in the literature. However, ISO/IEC 14764 was the only source that contained detailed activities. In the next Section (5.3.1) we review ISO/IEC 14764 as well as the other ISO/IEC standards from which it inherits some of its attributes.

5.3.1 ISO/IEC 14764

ISO/IEC-15504 also known as the Software Process Improvement and Capability Determination (SPICE) model contains a detailed reference process model which covers most of the process areas in software engineering. The reference process model from ISO/IEC 15504 is also published as the separate standard ISO/IEC 12207. ISO/IEC 12207 was first published in 1994 and contained descriptions for sub-processes from the software maintenance process. ISO/IEC 12207 contains the following sub-processes: Process Implementation, Problem and Modification Analysis, Modification Implementation, Maintenance Review/Acceptance, Migration and Retirement.

The standard was updated in 2008 to include a purpose and outcome for the software maintenance process. The reference life-cycle from ISO/IEC 15504 has descriptions for each process in the software engineering life-cycle. They need to be relatively concise. Otherwise, completing a capability assessment would become too labor intensive. Generally, there are more detailed ISO/IEC standards for the individual process areas from the software engineering life-cycle. In the case of the maintenance process there is a separate standard ISO/IEC 14764, which contains more detail than the process description from ISO/IEC 15504 or ISO/IEC 12207. It specifies the details of the inputs, tasks, controls, supports and outputs for each of the sub process for the maintenance process. Processes and their associated tasks in ISO/IEC 14764 are summarised here. Each process also has inputs, controls, supports and outputs which are not discussed.

Process Implementation requires maintenance plans and procedures to be created. The maintenance plan should document the plan for carrying out maintenance, while the maintenance procedures should contain more specific details for implementing this maintenance. Modification Request/Problem Report procedures are also listed. Procedures need to be put in place for receiving, recording and tracking modification requests and problem reports. A Configuration Management process also needs to be put in place to track the modification of an existing system.

Problem and Modification Analysis requires modification request and problem report analysis before deciding on how to proceed with changes. This may involve scoping the maintenance, documenting possible solutions and documenting impact on existing systems. Similarly the maintainer will need to verify or replicate the problem or issue. The maintainer needs to develop options for implementing the modification. Options to be developed include alternative work-arounds or solutions. Finally, the maintainer needs to document and ensure approval of the the modification request or problem report, the analysis and potential solutions.

Modification Implementation requires the maintainer to carry out analysis in order to determine which documents and software versions need to be modified. After the analysis the required software changes should be implemented during the development process.

Maintenance Review/Acceptance is a process which involves the maintainer carrying out reviews to ensure the integrity of the modified system. Following this task, the maintainer seeks approval from the appropriate authority that the maintenance has been completed satisfactorily.

Migration begins with the identification of all software or data that is modified if migration from an old platform to a new platform is performed. If migration is going to occur, it is necessary to create and document a migration plan and then execute the migration according to the plan. Prior to migration a notification of intent should be provided to all system users before migration occurs. Following migration the old and new environments should be run in parallel while providing training

to end users in order to ensure a smooth transition. Once migration has been completed, notification of completion needs to be sent to the appropriate stakeholders. Post migration review should be conducted after migration in order to assess the impact of the migration. Finally, all of the data associated with the old environment should be achieved in accordance with the appropriate data protection and audit policies.

Software retirement takes place once a decision has been made to retire an active software product. A retirement plan should be developed and documented by the system maintainer. After deciding to retire software, a notification of retirement intent should be sent to the appropriate software product stakeholders. During retirement, a parallel operation of new and retiring software software should be carried out along with the training of end users. Once complete, notification should be sent to the appropriate stakeholders and finally, data relating to the retiring product should be archived should it be required at a later date.

5.3.2 Activities Identified

In total there were 19 maintenance practices identified from ISO/IEC 14764. They are categorised in Table 3 according to the processes they come from in ISO/IEC 14764, and represent the complete set of activities that need to be carried out to implement a maintenance process . The first set of activities refer to the actual implementation of the required process guidelines, while the other activities detail the execution of those guidelines.

Table 3 Adaptation Support Activities from ISO/IEC 14764

Define adaptation requirements
14764: Problem and Modification Analysis : MR/PR analysis 14764: Migration: Migration
Define requirements to the monitoring subject
14764: Modification Implementation: Analysis
Define monitored property
Provide monitoring functionality
14764: Process Implementation: MR/PR procedures
Collect monitoring results for adaptation
14764: Problem and Modification Analysis : Verification 14764: Maintenance Review/Acceptance: Reviews 14764: Migration: Post-operation review 14764: Maintenance Review/Acceptance: Approval
Trigger adaptation
Design adaptation strategy
14764: Process Implementation: Maintenance plans and procedures 14764: Problem and Modification Analysis: Options 14764: Migration: Migration plan
Select adaptation strategy
14764: Problem and Modification Analysis : Approval
Perform adaptation
14764: Process Implementation: Configuration management 14767: Modification Implementation: Development process

6 PHASE II: Mapping Adaptation Activities

6.1 Adaptation Activities Mapped to the S-Cube Life-Cycle

Table 4 shows each of the adaptation activities that were identified in Section 5.3.2 mapped to the appropriate phases of the S-Cube life-cycle. Where possible the activities are grouped in related categories. The categories used are those identified from the S-Cube deliverables in Section 5.1. The activities within each life-cycle phase are not in a specific order and they can be executed as needed.

6.2 Observations on Adaptation Activities

Unfortunately, the adaptation activities identified from the service-oriented engineering approaches do not form a complete view of all the necessary activities required to enable the adaptation of SBAs. This is due to the fact that the activities were identified from many different sources that do not treat service adaptation as a primary concern. As we can see from Table 4, many activities were in the process: *Identify adaptation needs*, while only a few processes were identified in *Identify adaptation strategy* and *Enact adaptation*. This implies that the state of art of adaptation processes focuses much more on gathering requirements and identifying when adaption is needed. These are highly relevant to what needs to be monitored. However, as soon as the need for adaption is identified, little efforts have been put in to defining, selecting and executing adaptation strategies.

Only two SOA approaches, SDLC and SeCSE, explicitly describe the actual execution of adaptation. Indeed, in these cases, the adaptation is lim-

Table 4 Adaptation Activities Mapped to S-Cube Life-Cycle

Requirements engineering and design	BEA: Define KPIs and management policies SeCSE: Identify the service properties to specify
Construction	
Deployment and provisioning	SeCSE: Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system
Operation and management	
Identify adaptation needs	**Define adaptation requirements**
	Define requirements to the monitoring subject SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs
	Define monitored property SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
	Provide monitoring functionality ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services
	Collect monitoring results for adaptation ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics
	Trigger adaptation SeCSE: Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Select adaptation strategy	**Design adaptation strategy** Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters
	Select adaptation strategy
Enact adaptation	**Perform adaptation** SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery

ited to corrective adaptation - the replacement of services when quality attributes do not meet expectations. Other types of adaptation such as perfective adaptation, adaptive adaptation, preventive adaptation and extending adaptation are not supported.

None of the existing service-oriented engineering approaches specifies how to select an adaptation strategy. In the two approaches that actually describe the execution of adaptation, the adaptation strategies are (implicitly) pre-defined.

While adding these activities to the S-Cube life-cycle, we noticed that some of them belong to the adaptation cycle, while there are others which, while coming under adaptation within service-oriented engineering approaches and S-Cube life-cycle literature, actually belong to the evolution cycle of the S-Cube life-cycle. For instance, KPIs and management policies (from BEA) as well as service properties (from the SeCSE methodology) are defined at the requirement engineering process. There not directly used by adaptation practices but are relevant in that specifying these attributes makes corresponding monitoring and assessment possible.

6.3 Adaptation Support Activities Mapped to the S-Cube Life-Cycle

Out of the 19 adaptation support activities identified from ISO/IEC 14764, 13 of them were mapped to the S-Cube life-cycle. Table 5 shows the cumulative mapping of the adaptation as well as the adaptation support activities to the S-Cube life-cycle.

Process Implementation The process implementation process area from ISO/IEC 14764 has three activities: *Maintenance plans and procedures*, *Problem reports/modification requests (MR/PR) procedures* and *Configuration management* each of which were mapped to one of the high level adaptation activities of the S-Cube life-cycle from Phase I. The implementation of *MR/PR procedures* was mapped to *Provide monitoring functionality* in the life-cycle. The implementation of problem report procedures would allow application engineers to receive and track problem reports which would allow them to determine if adaptation is necessary. Similarly, a modification request procedure would allow engineers to track modification requests and determine

Table 5 Adaptation activities from ISO/IEC 14764 and Service Engineering Literature Mapped to S-Cube Life-Cycle

Requirements engineering and design	BEA: Define KPIs and management policies SeCSE: Identify the service properties to specify
Construction	
Deployment and provisioning	SeCSE: Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system
Operation and management	
Identify adaptation needs	**Define adaptation requirements** 14764: Problem and Modification Analysis : MR/PR analysis 14764: Migration: Migration
	Define requirements to the monitoring subject 14764: Modification Implementation: Analysis SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs
	Define monitored property SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
	Provide monitoring functionality 14764: Process Implementation: MR/PR procedures ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services
	Collect monitoring results for adaptation 14764: Problem and Modification Analysis : Verification 14764: Maintenance Review/Acceptance: Reviews 14764: Migration: Post-operation review 14764: Maintenance Review/Acceptance: Approval ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics
	Trigger adaptation SeCSE: Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
Select adaptation strategy	**Design adaptation strategy** 14764: Process Implementation: Maintenance plans and procedures 14764: Problem and Modification Analysis: Options 14764: Migration: Migration plan Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters
	Select adaptation strategy 14764: Problem and Modification Analysis : Approval
Enact adaptation	**Perform adaptation** 14764: Process Implementation: Configuration management 14767: Modification Implementation: Development process SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery

if the modification request requires adaptation. The *Maintenance plans and procedures* activity was mapped to *Define adaptation strategy* in the S-Cube life-cycle. The *Define adaptation strategy* activity refers to the definition of plans and procedures for adapting a SBA, so it makes sense that *Maintenance plans and procedures* could be used for this activity given the commonalities between adaptation and maintenance. *Configuration management* was mapped to the *Enact adaptation* activity, because the resolution of problems after applications adapt would be much easier if configuration details of component services are recorded. Fang *et al* [33] illustrate how the configuration management process would be beneficial to adaptation of SBAs.

Problem and Modification Analysis The problem and modification analysis process area contains four activities that are useful for SBA aadaptation: *Problem reports/modification requests (MR/PR) analysis*, *Verification*, *Options* and *Approval*. In the context of software maintenance these activities are undertaken in order to analyze problem reports or modification requests and determine their impact on the application (*MR/PR analysis*). If the reports or modification requests are valid (*Verification*) potential solutions are proposed (*Options*)

and approval is sought to implement the required changes (*Approval*). The *MR/PR analysis* activity is mapped to *Define adaptation requirements* in the S-cube life-cycle. The *analysis of maintenance requests and problem reports* could be altered to the *analysis of adaptation requests and problem reports* to suit the adaptation of SBAs. This analysis activity could provide valuable input which could be used to *Define adaptation requirements* for a SBA. *Verification* is mapped to the *Collect monitoring results for adaptation* activity because replicating or verifying the problem can be seen as an analysis on the monitoring results. *Options* is mapped to *Design adaptation strategy* because options for implementing the modification can be seen as adaptation strategy. Finally *Approval* is mapped to *Select adaptation strategy* because obtaining approval is part of adaptation strategy selection in that the it finalises the decision on the selection.

Modification Implementation contains two activities *Analysis* and *Development* which are mapped to *Define adaptation requirements* and *Perform adaptation* respectively. *Analysis* is usually carried out before any *Development* or maintenance activity in order to determine which artifacts need to be modified. This may also be useful during the requirements gathering phase of SBA adaptation in order to determine which parts of the application need to be changed. In the context of traditional software engineering *Development* means the modification of application code in order to implement requirements, this activity could be tailored to mean the modification of an applications configuration to meet the adaptation requirements of a SBA.

Maintenance Review/Acceptance The Maintenance Review/Acceptance process area contains two activities: *Reviews* and *Approval*. In the context of software maintenance, reviews are carried out to ensure that the maintenance is carried out appropriately. In terms of adaptable SBAs, reviews can be carried out to ensure that adaptation occurs correctly. The analysis of collected monitoring results can be used to perform a review of SBAs which is why the *Reviews* activity was mapped to *Collect monitoring results for adaptation*. Following a *Review*, *Approval* status may be given to an adaptation engineer on satisfactory adaptation of an application. If adaptation occurs automatically it is impossible to grant approval to the work of an individual(s) so it may be appropriate to grant approval to the adapted application.

Migration In the context of traditional software engineering, migration is the modification of a system thus allowing it to run in a new environment or context. Rather than migrate a SBA, it may be possible for the application to adapt in order to operate in a new environment. Therefore, the migration process area may contain some useful activities that can help a SBA adapt to context specific parameters. The maintenance process area has three activities that are useful to the adaptation of SBAs: *Migration*, *Migration plan* and *Post-operation review*. Migration was mapped to *Define adaptation requirements* because it is important to determine which software artifacts or which data should be migrated (or adapted) during the requirements gathering stage. *Migration plan* was mapped to *Design adaptation strategy* because a migration plan can be seen as an adaptation strategy in that it specifies what tools are needed, how to convert software product and data and how to execute migration. Finally *Post-operation review* was mapped to *Collect monitoring results for adaptation* because the impact of changing to the new environment can be achieved by monitoring.

Other Mappings Many of the adaptation support activity mappings are apparent, for example *Maintenance plans and procedures* to *Design adaptation strategy* or *Modification request/problem report procedures* to *Provide monitoring functionality*. Some of the other mappings however are not so apparent, such as the "maintenance review/acceptance" activity that maps to *Collect monitoring results for adaptation*.

During our analysis we discovered that some of the maintenance activities are also relevant to the evolution cycle of the S-Cube life-cycle. However, those mappings were excluded as we are focusing on adaptation in this paper. As previously mentioned five activities from ISO/IEC 14764 could not be mapped to adaptation activities because they are too specific to the software maintenance process (see Table 6): the *Documentation activity* and *Migration* activities. The *Documentation* activity from the maintenance process does not get included or is paid very little attention to in any of the adaptation activities covered in the literature. The four migration activities mentioned in Table 6 are specific to the the maintenance of traditional software and should not be leveraged for service adaptation.

6.4 Observations on Adaptation Support Activities

The maintenance activities identified in this section were never previously identified in the service engineering literature as candidate activities for the adaptation of SBAs. Many of the activities

Table 6 Maintenance Activities not Mapped

Maintenance Practices
2 Problem and Modification Analysis
Documentation
5 Migration
Notification of intent
Implement operations and training
Notification of completion
Data archival

identified from the service engineering literature tend to deal with the technical details of adaptation rather than focusing on process details. One of the strengths of eliciting activities from a software process standard is that there is a process focus with process details such as inputs, tasks, controls, supports and outputs. The activities elicited from the service literature tend to specify *what* needs to be done in order to adapt SBAs while the maintenance activities identified can be tailored to specify *how* to implement the adaptation processes.

The suitability of maintenance activities for SBA adaptation highlights the commonalities between SBA adaptation and software maintenance. Both of these processes involve the modification of software systems albeit in different contexts. Adaptation is a light weight process which may only require the modification of simple configuration details to facilitate adaptation, so it is important not to include maintenance activities which would add unnecessary overhead to the process. The *Documentation* activity falls into this category, *Documentation* would add a lot of overhead to the process which is unnecessary due to Ad Hoc nature of SBA adaptation during runtime.

Since we are reusing activities from a process model designed for the maintenance process we cannot be guaranteed that the activities we have chosen form the complete set of activities required for adaptation. However, when combined with the activities from the service literature the resultant set of activities are one step closer to the complete set of activities required for SBA adaptation.

The activities identified from the maintenance literature are designed for the maintenance process which involves many manual activities, such as the analysis of problem reports and the development of proposed changes. However, the adaptation process may be a manual or automatic process. If the adaptation is manual, many of the maintenance activities can be applied directly without modification. However, if the adaptation is automatic then many of the maintenance activities may become obsolete or require re-interpretation. For example, the *Analysis of problem reports* activity by definition is a manual activity carried out by a system maintainer, in the case of automatic adaptation it becomes obsolete as the application analyses problems through its monitoring mechanisms.

Table 7 Software Metrics (Reifer,2006)

Category	**Examples**
Project performance	Budget performance Schedule performance Earned value performance Technical performance
Process performance	Rework rate Defect rates
Product quality	Product complexity Defect density
Personnel performance	Personnel productivity
Orgainisational performance	Process maturity Product quality Productivity
Enterprise performance	Profitability Return on equity Cost of sales Competitiveness

7 Measuring the Performance of the Process Framework

7.1 Process Metrics

When implementing a software process reference model or framework it is important to have the ability to measure the performance of that new process. This allow the software engineer to evaluate any improvement or disimprovement that may have occurred. There are many quantitative metrics that can be used to measure the performance impact of implementing a software process.

Reifer [34] proposes a comprehensive set of performance metrics that can be used to measure a software organisation's performance. The set is illustrated in Table 7.

An improved software process promises to improve product quality and organisational performance [35]. Improvements in these metric categories should also have a cascading effect on higher level metrics such as Enterprise performance. To measure the impact of the framework proposed in this paper, metrics from the process performance, product quality, personnel performance and orgainisational performance categories of Table 7 are suggested. These categories were chosen because process improvement should be readily reflected by metrics from these categories. Table 8 illustrates

the framework developed in this paper. A column has been added representing the metrics suggested for the measuring the impact of each process.

The productivity and complexity related metrics were suggested for the "Requirements engineering" process since defect type metrics don't apply this early in the life-cycle. Defect type metrics were suggested for the "Deployment and provisioning", and the "Enact adaptation" processes because it is expected that defects in adaptable SBAs may be identified at these points. Finally productivity type metrics are suggested also for the "Identify adaptation needs" and the "Select adaptation strategies" processes since the efficiency of these processes would be reflectivity of organisational and personnel productivity.

8 Case Study

In this section we present a real-life case study that illustrates how the activities identified during this research can aid in the development of adaptable SBAs. The case documents how a software company currently develops adaptable SBAs, then through an illustrated example we will show how their development process is improved upon with the results of this research. In the following subsections we will introduce SBA Solutions and there existing process for the development of adaptable SBAs. Then we will apply our framework to their process illustrating its effectiveness.

8.1 Introduction to SBA Solutions

SBA Solutions is a large multinational software development company with several off shore development groups, sales and service centres. Their offshore development teams are self-contained development units with the required skillets to undertake complete development projects. For them, this is an effective approach, as many Global Software Development complexities are bypassed. Their main product makes extensive use of services to exchange data and integrate processes with different public sector departments.

8.2 Research within SBA Solutions

Our research within SBA solutions was undertaken as part of the S-Cube project [1]. We carried out interviews with a lead architect, a developer who works within the construction process and a member of the sales department who previously worked within the companys services organisation. Interviews focused on their roles within the company, the development of adaptable SBAs, the development process and their approach to adaptation. They gave insight as to why SBA Solutions chose the SOA paradigm. Supporting documentation such as process models, development artefacts and company presentations were made available. Interview transcripts were analysed using Miles and Huberman's content analysis techniques [36].

8.3 Case Study Discussion

A key to the software process in SBA Solutions is their process model that breaks down each process into discrete tasks. These are each assigned to relevant stakeholders. This gives each member of the development team clear roles and responsibilities and eliminates redundancy. Since they employ GSD and their development is carried out iteratively, development iterations can be carried out concurrently or staggered to increase the development speed. This is a complicated development process so the process model is crucial for them to effective management.

SBA Solutions implement runtime adaptation in an interesting way. They have an adaptation dashboard that is used at runtime to make changes to the way that the application executes. This allows the modification of business workflows and the addition or removal of business rules to suit real time business requirements. Another interesting adaptation feature of one of their key applications is that the execution path of the application is dynamically determined based on parameters that are passed to the application. This is a type of built-in static adaptation that is implemented using a rule engine. These rules may also be subsequently adapted using the adaptation dashboard.

In SBA soft adaptation is first tried on a test system to determine if there are any unforeseen consequences. If, after adaptation, the test system operates correctly, then the adaptation configuration is transferred to the database of the production application. The basic adaptation process for SBA Solutions can be seen in Figure 3.

One of the downsides to how adaptation is enacted in the products of SBA solutions is that, while they have a rigorous development process model, runtime adaptation processes are ad-hoc and vary depending on their customers requirements. While it is important to consider customer requirements for adaptation, it would also be beneficial to have a process model to guide adaptation.

Table 8 Adaptation Framework and Assessment Metrics

Process	Activity	Metrics
Requirements engineering and design	BEA: Define KPIs and management policies SeCSE: Identify the service properties to specify	Productivity, Personnel productivity, Product complexity
Construction		
Deployment and provisioning	SeCSE: Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system	Defect rates, Rework rate, Defect density
Operation and management		
Identify adaptation needs	**Define adaptation requirements** 14764: Problem and Modification Analysis : MR/PR analysis 14764: Migration: Migration	Productivity, Personnel productivity, Product complexity
	Define requirements to the monitoring subject 14764: Modification Implementation: Analysis SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs	
	Define monitored property SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)	
	Provide monitoring functionality 14764: Process Implementation: MR/PR procedures ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services	
	Collect monitoring results for adaptation 14764: Problem and Modification Analysis : Verification 14764: Maintenance Review/Acceptance: Reviews 14764: Migration: Post-operation review 14764: Maintenance Review/Acceptance: Approval ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics	
	Trigger adaptation SeCSE: Recovery management: identify by looking at the monitoring data, the needs for a recovery action	
Select adaptation strategy	**Design adaptation strategy** 14764: Process Implementation: Maintenance plans and procedures 14764: Problem and Modification Analysis: Options 14764: Migration: Migration plan Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters	Productivity, Personnel productivity
	Select adaptation strategy 14764: Problem and Modification Analysis : Approval	
Enact adaptation	**Perform adaptation** 14764: Process Implementation: Configuration management 14767: Modification Implementation: Development process SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery	Rework rate, Defect rates, Defect density

This model could be adapted to customers needs which would still allow flexibility in the process.

8.4 Improved SBA Adaptation Process

In this section we will illustrate how the adaptation process at SBA Solutions can be enhanced by following the adaptation activities identified in this paper. Table 9 illustrates SBA Solution's devel-

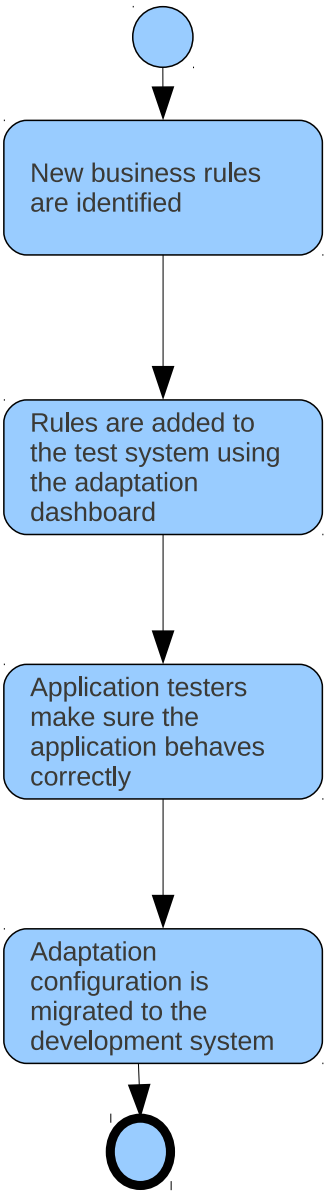

Fig. 3 SBA Solutions Adaptation Process.

opment and adaptation processes mapped to the adaptation framework previously shown in Table 5. These processes have replaced the sample processes from the S-Cube reference life-cycle.

These mapping shows how SBA solution's ad-hoc adaptation process can be greatly enhanced by adding more detailed adaptation as well as adaptation support activities. It may be the case that all activities are not needed depending on project requirements, but these redundant activities can simply be excluded from implementation.

9 Conclusion

In this paper, we identified activities and support activities which should be considered when carrying out the adaptation of SBAs. This has been done through the identification of activities within service-oriented development models, and the software engineering maintenance process. The importance of this work is that while consolidating existing work for service-oriented development into a SBA development life-cycle, it enhances this with support activities from mature software engineering process standards. The result is an adaptation framework that can be used to guide practitioners in the development of adaptable SBAs. This point has been illustrated through the documentation of a case where the framework has been mapped to the existing development processes of a company that develops adaptable SBAs.

We have observed that, in isolation, the service-oriented development approaches that we reviewed do not adequately facilitate adaptation. We have also observed that, when combined, the adaptation activities identified may not cover every aspect of adaptation. They do however, present a set of activities that can be combined into a useful framework for developing adaptable SBAs. In addition to the core adaptation activities, we identified a set of adaptation support activities that add a level of governance and control to the adaptation process. While not directly involved with adaptation, these best practice support activities increase the quality of the adaptable applications being developed.

We have also observed that adaptation and maintenance, while separate processes, share common attributes. Maintenance involves the modification of an application's source code while adaptation allows an application to adapt without modifying its code. However, the similarities between the two processes allow us to reuse many maintenance activities as adaptation support activities.

During this research, we have seen how the the SBA adaptation cycle can be detailed using service-oriented development and software engineering activities, based particularly on service development models and on the maintenance standard ISO/IEC 14764.

In addition, while we have identified the activities which should be used during the adaptation cycle, we have also demonstrated how these can be implemented within an industrial case study. The next stage of this research project will focus on identifying SBA adaptation practices from other industrial case studies. These will be combined with the activities elicited from the literature to create an incremented version of the model presented here. We then hope to run experiments with the framework to evaluate its effectiveness and de-

Table 9 Adaptation activities from ISO/IEC 14764 and Service Engineering Literature Mapped to SBA Solutions Life-Cycle

Evolution	**Requirements Definition**	BEA: Define KPIs and management policies
	Architecture and Design	SeCSE: Identify the service properties to specify
	Coding and Testing	
	Stabilisation and Release	SeCSE: Insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system
Adaptation	**New business rules are identified**	**Define adaptation requirements** 14764: Problem and Modification Analysis : MR/PR analysis 14764: Migration: Migration
		Define requirements to the monitoring subject 14764: Modification Implementation: Analysis SDLC: Set warning thresholds and alerts for compliance failures SDLC: Gather QoS metrics on the basis of SLAs
		Define monitored property SeCSE: Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)
	Application testers make sure the application behaves correctly	**Provide monitoring functionality** 14764: Process Implementation: MR/PR procedures ASTRO: Monitor message sequences amongst services and its partners BEA: Monitor service, application, middleware, OS, hardware, and network SDLC: Monitor workloads SeCSE: Monitor services
		Collect monitoring results for adaptation 14764: Problem and Modification Analysis : Verification 14764: Maintenance Review/Acceptance: Reviews 14764: Migration: Post-operation review 14764: Maintenance Review/Acceptance: Approval ASTRO: Detect protocol violations SDLC: Evaluate SLA QoS metrics
		Trigger adaptation SeCSE: Recovery management: identify, by looking at the monitoring data, the needs for a recovery action
	Rules are added to the test system using the adaptation dashboard	**Design adaptation strategy** 14764: Process Implementation: Maintenance plans and procedures 14764: Problem and Modification Analysis: Options 14764: Migration: Migration plan Chang's: Specifying Service Decision Model Chang's: Designing Service Adapters
		Select adaptation strategy 14764: Problem and Modification Analysis : Approval
	Adaptation configuration is migrated to the development system	**Perform adaptation** 14764: Process Implementation: Configuration management 14767: Modification Implementation: Development process SDLC: Readjust service weights for request queues SeCSE: Runtime Service Discovery

termine the overhead of implementing it in real projects.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube) and has been partially supported by Lero - the Irish Software Engineering Research Centre, Science Foundation Ireland Grants No. 03/CE2/I303.1 & 10/CE/I1855.

References

1. V. Andrikopoulos, "Seperate design knowledge models for software engineering and service based computing," S-Cube Consortium, Deliverable CD-JRA-1.1.2, May 2009.
2. P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 3, pp. 54–62, 1999.
3. I. Richardson and S. Lane, "Coordinated design knowledge models for software engineering and service-based computing," S-Cube Consortium, Deliverable CD-JRA-1.1.4, 2009.
4. W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE Wescon*, vol. 26, no. 1, 1970, p. 9.
5. B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.
6. M. Hinchey and L. Coyle, "Evolving critical systems," Lero - The Irish Software Engineering Research Centre, Tech. Rep. Lero-TR-2009-00, 2009.
7. "Herstellerinitiative software (his) process assessment working group." [Online]. Available: http://portal.automotive-his.de/images/pdf/ProcessAssessment/his-wg-assessments_v31_07-06-08.pdf
8. "Amazon." [Online]. Available: http://www.amazon.com/
9. S. Benbernou, "State of the art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of sbas," S-Cube Consortium, Deliverable PO-JRA-1.2.1, July 2008.
10. J. Hielscher, A. Metzger, and R. Kazhamiakin, "Taxonomy of adaptation principles and mechanisms," S-Cube Consortium, Contractual Deliverable CD-JRA-1.2.2, May 2009.
11. S. K.Williams, S. A. Battle, and J. E. Cuadrado, "Protocol mediation for adaptation in semantic web services," in *The Semantic Web: Research and Applications*, ser. Lecture Notes in Computer Science. Springer, 2006, vol. 4011, pp. 635–649.
12. E. Di Nitto, M. D. Penta, A. Gambi, G. Ripa, and M. Villani, "Negotiation of service level agreements: An architecture and a Search-Based approach," in *Service-Oriented Computing ICSOC 2007*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 4749, pp. 295–306.
13. B. Pernici, "Automatic learning of repair strategies for web services," in *Fifth European Conference on Web Services (ECOWS '07)*. Halle, Germany: IEEE Computer Society, Nov. 2007, pp. 119 –128.
14. M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso, "Planning and monitoring web service composition," in *Artificial Intelligence: Methodology, Systems, and Applications*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 106–115.
15. S. D. Conte, H. E. Dunsmore, and Y. E. Shen, *Software engineering metrics and models*, 1986.
16. M. M. Lehman, "Program evolution." *INFO. PROC. MANAGE.*, vol. 20, no. 1, pp. 19–36, 1984.
17. A. April, J. H. Hayes, A. Abran, and R. Dumke, "Software maintenance maturity model (smmm): the software maintenance process model," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, no. 3, pp. 197–223, 2005.
18. A. Abran, P. Bourque, R. Dupuis, and J. W. Moore, "Guide to the software engineering body of Knowledge-SWEBOK," 2001.
19. E. B. Swanson, "The dimensions of maintenance," in *Proceedings of the 2nd international conference on Software engineering*, 1976, pp. 492–497.
20. IEEE, ISO, and IEC, *Software engineering–software life cycle processes–maintenance International standard.* New York, NY: Institute of Electrical and Electronics Engineers, 2006.
21. I. Sommerville, *Software Engineering, 7th Edition*, 7th ed. Addison Wesley, Jun. 2004.
22. M. Autili, L. Berardinelli, V. Cortellessa, A. D. Marco, D. D. Ruscio, P. Inverardi, and M. Tivoli, "A development process for self-adapting service oriented applications," in *Service-Oriented Computing ICSOC 2007*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 4749, pp. 442–448.
23. A. kenzi, B. E. Asri, M. Nassar, and A. Kriouile, "A model driven framework for multiview service oriented system development," in *ACS/IEEE International Conference on Computer Systems and Applications*. IEEE Computer Society, 2009, pp. 404–411.
24. M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, and P. Traverso, "Astro: Supporting composition and execution of web services," in *Lecture notes in computer science*, vol. 3826, 2005, p. 495.
25. S. Durvasula *et al.*, "Introduction to service lifecycle. SOA practitioners guide. part 3," 2007.
26. S. H. Chang, "A systematic analysis and design approach to develop adaptable services in service oriented computing," in *Services, 2007 IEEE Congress on*, 2007, pp. 375–378.
27. A. Arsanjani, "Service-oriented modeling and architecture," Nov. 2004. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/
28. M. P. Papazoglou and W. V. D. Heuvel, "Service-oriented design and development methodology," *Int. J. Web Eng. Technol.*, vol. 2, no. 4, pp. 412–442, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1358575.1358582
29. P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Addison Wesley, Dec. 2003.
30. P. Herzum and O. Sims, *Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise.* John Wiley & Sons, Inc. New York, NY, USA, 2000.
31. P. Harmon, "Second generation business process methodologies," *Business Process Trends*, vol. 1, no. 5, 2003.

32. ATOS, "SeCSE methodology, version 3," Tech. Rep., Mar. 2007.
33. R. Fang, Y. Chen, L. Fong, L. Lam, D. Frank, C. Vignola, and N. Du, "A version-aware approach for web service client application," in *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 401–409.
34. D. J. Reifer, "Metrics and management: A primer," in *Software Management, 7th ed.*, D. J. Reifer, Ed. IEEE Computer Society, 2006, pp. 397–401.
35. W. Humphrey, *A Method for Assessing the Software Engineering Capability of Contractors: Preliminary Version*. Addison Wesley, 1987.
36. M. B. Miles and A. M. Huberman, *Qualitative data analysis: An expanded sourcebook*. SAGE publications, Inc, 1994.

Appendix D

SOAdapt-V1 Paper

Information and Software Technology xxx (2011) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

SOAdapt: A process reference model for developing adaptable service-based applications

Stephen Lane [a,*], Antonio Bucchiarone [b], Ita Richardson [a]

[a] Lero, The Irish Software Engineering Research Centre, University of Limerick, Ireland
[b] Fondazione Bruno Kessler, Via Santa Croce 77, 38100 Trento, Italy

ARTICLE INFO

Article history:
Received 15 April 2011
Received in revised form 24 October 2011
Accepted 25 October 2011
Available online xxxx

Keywords:
SOA
Service-based application
Software process
Process reference model

ABSTRACT

Context: The loose coupling of services and Service-Based Applications (SBAs) have made them the ideal platform for context-based run-time adaptation. There has been a lot of research into implementation techniques for adapting SBAs, without much effort focused on the software process required to guide the adaptation.
Objective: This paper aims to bridge that gap by providing an empirically grounded software process model that can be used by software practitioners who want to build adaptable SBAs. The process model will focus only on the adaptation specific issues.
Method: The process model presented in this paper is based on data collected through interviews with 10 practitioners occupying various roles within eight different companies. The data was analyzed using qualitative data analysis techniques. We used the output to develop a set of activities, tasks, stakeholders and artifacts that were used to construct the process model.
Results: The outcome of the data analysis process was a process model identifying nine sets of adaptation process attributes. These can be used in conjunction with an organisation's existing development life-cycle or another reference life-cycle.
Conclusion: The process model developed in this paper provides a solid reference for practitioners who are planning to develop adaptable SBAs. It has advantages over similar approaches in that it focuses on software process rather than the specific adaptation mechanism implementation techniques.

1. Introduction

The Service-Oriented Computing (SOC) paradigm is centered around the publication and consumption of loosely coupled computational elements known as services. These software services are often owned and controlled by third parties rather than the application developers who consume them. A key benefit of SOC is the ability to rapidly compose distributed software services into useful Service-Based Applications (SBAs). These applications are able to offer complex and flexible functionalities in widely distributed environments by composing different types of services. Such services are often not under the control of systems developers, but they are simply exploited to obtain a specific functionality.

The possibility of rapidly composing SBAs opens up new opportunities for conducting ad hoc business transactions. However, with this new freedom there are also new challenges, for example, when services are owned and controlled by third parties their reliability or availability are not guaranteed. Adaptable SBAs change their behavior, reconfigure their structure and evolve over time reacting to changes in the operating conditions, so as to continuously meet users' expectations. This concept is fundamental since those systems live in distributed and mobile devices which have frequently changing environments. Also, user goals and needs may change dynamically, and systems should adapt their functionalities accordingly without intervention from technicians.

When designing and developing traditional software, modeling and implementing the application's logic is usually the only problem to consider. With SBAs, adaptability should also be considered during the requirements engineering, design and construction phases of SBA development. Similarly the adaptation of an SBA needs to be monitored and controlled when these applications are in operation.

Many of the existing SBA development methodologies are based on the results carried out in the fields of classical software and system engineering and do not easily facilitate SBA adaptation [1–3]. Some of the reported SBA development approaches such as SOUP (Service Oriented Unified Process) [4], ASTRO [5] or an approach by Linner et al. [6] do support some level of adaptation, however, lack sufficient process details. Lane and Richardson [7] carried out a systematic literature review of SBA development approaches,

* Corresponding author.
E-mail addresses: stephen.lane@lero.ie (S. Lane), bucchiarone@fbk.eu (A. Bucchiarone), ita.richardson@lero.ie (I. Richardson).

0950-5849/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2011.10.003

they identified 57 such approaches of which there were only eight that specifically dealt with adaptation. Only four of the eight approaches were concerned with adaptation of SBAs, the others were concerned with the adaptation of services.

Each of the four approaches show interesting features, but even those that enable the definition of various adaptation strategies lack a coherent design approach to support designers in this complex task. Moreover, they tend to focus on the implementation process without considering what impact adaptation has on the rest of the development and operational life-cycle [8–11]. Finally, they also tend to focus on particular types of adaptation, such as adaptation due to requirements violations [12], or substitution services due to application constraints [13], so it is difficult to elicit generic adaptation specific processes from them. In summary, each of these four approaches focused on the analysis and design processes without consideration for any other development or runtime processes. The solution proposed in [14] can be considered as a first step in this direction. The authors of this paper have proposed a process model for SBAs where adaptation is a first class concern. This process model provides a number of steps that, if implemented during the life-cycle of a SBA, will allow the SBA to adapt in response to changing runtime contexts.

1.1. Objectives

The primary objective of this paper is to develop an empirically grounded process model that can be used as a reference for service practitioners who want to develop adaptable SBAs. The process model will focus on activities related to adaptation only, meaning that it will have to be used in conjunction with an other SBA development life-cycle. In order for an SBA to be adaptable, certain adaptation mechanisms need to be put in place when the application is initially developed. These mechanisms then facilitate adaptation when the application is in operation.

There are two types of adaptation, static adaptation and dynamic adaptation [15]. With static adaptation, adaptation mechanisms are hard coded into the application at development time and the adaptation logic cannot be changed without recoding. Dynamic adaptation on the other hand allows the adaptation logic to be modified or replaced during runtime without shutting the system down. Dynamic adaptation is more flexible than static adaptation but it requires some process to guide the manual intervention during runtime.

Process details will be elicited from relevant SBA development and adaptation approaches as well as qualitative data gathered from interviews with industry practitioners. The validity of the SBA adaptation process model will be demonstrated through a comparison with a previously validated adaptation process model. This comparison will show that the process model contains the minimum set of activities required for runtime software adaptation. The minimum set of activities for adaptation exclude activities that are not specifically related to adaptation. The processes and activities from the process model will also be mapped to an empirically based SBA development life-cycle. This will illustrate whether or not the model is suitable for use in a real world SBA development context.

1.2. Taxonomy

The terminology used in the areas of software process and service engineering research are often ambiguous and conflicting. In order to define what is meant by the more common software process and service engineering terms used in this paper a brief outline will be given for each. The definitions for the service engineering terms are based on the S-Cube knowledge model [16] while the software process terms are based on Derniame et al. [17].

Service-based applications are composite applications composed of multiple software services. Software services or services are computational elements that expose their functionality over computer networks. Because services are self-contained and loosely coupled it is possible to build SBAs from unrelated services that may or may not be in the control of the SBA developers. An example SBA is the travel booking application, which is composed of three third party services: a car rental, flight reservation and hotel reservation services. These three unrelated services are composed to provide useful SBA, synonyms for SBAs include service-oriented systems or service-based systems.
Adaptable service-based applications. ASBAs are a specific type of SBA that adapt during operation by following built-in adaptation strategies. Adaptation of SBAs is facilitated by monitoring mechanisms that monitor changes in monitored properties. Adaptation may be desirable for many reasons such as changes in quality characteristics, change in business requirements or change in the cost of services being consumed. Adaptation may occur through the selection of alternative services from pre-defined lists or through discovery of new services from service directories.
Software process. Software process refers to the set of processes and activities undertaken to develop software. This process is often measured so that it can be managed or improved. Some software developers follow predefined processes while others have a more ad hoc software process. Nevertheless, whether defined or not, each software development project will follow a particular set of processes. Synonyms for software process include: software development process or software life-cycle.
Process models are a documented representation of a set of real world processes, activities and their interrelations. A *software process model* is specific type of process model for software development. Creating a process model for a software development process allows reasoning about the model so that it can be improved. Ideally a process model should perfectly articulate the real world processes and activities that they are attempting to describe.
Process reference models are best practice or exemplar *process models* that can be used as a reference by individuals or groups to improve on their own processes. A specific type of process reference model is a *software process reference model* which is an exemplar process model used for developing software, for example, the Capability Maturity Model Integration (CMMI) and ISO-15504 document process reference models.
Life-cycle models in software engineering are *process models* that define the entire software development life-cycle. This is different from a *process model* which may only address particular parts of the development cycle.
Life-cycle process is a collection of software development activities that occur in sequence and make up one logical part of the software development life-cycle. For instance, "Requirements engineering" is a *software life-cycle process* that may contain activities such as "Elicit requirements" or "Develop requirements specifications".

2. Background

2.1. A life-cycle to develop adaptable SBA

The life-cycle model shown in Fig. 2, proposed by the S-Cube consortium [18], highlights the typical design-time iteration cycle that leads to the design of adaptable SBAs. It introduces an iteration cycle at run-time that is undertaken in all cases in which

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

the adaptation needs are addressed on-the-fly. The two cycles coexist and support each other during the lifetime of the application. In particular, the design-time activities allow for *evolution* of the application, that is, for the introduction of permanent and, usually, important changes, while the run-time activities allow for temporary *adaptation* of the application to the specific circumstances that are occurring at a certain time.

2.2. Design guidelines for adaptation

The design of SBAs with runtime adaptation capabilities may be motivated by variety of needs. Such needs may concern the component services or the context of SBAs. These include:

Changes in the service functionality due to variation of the service interface (e.g., signatures, data types, semantics), variation of service interaction protocol (e.g., ordering of messages) or Application failures.
Changes in the service quality due to service availability, degrade of QoS parameters, violation of SLA or decrease of service reputation (e.g., black lists), etc.
Changes in the service context as a result of changes in the business context, changes in agile service networks, or new business regulations and policies.
Changes in the computational context such as different devices, protocols, and networks.
Changes in the user context such as different user groups and profiles, social environment, physical settings (e.g., location/time), and different user activities.

Some of these aspects may be interleaved. For example, if a user moves to a new location (change in the user context), a new set of services may be available (change in the business context) with different bandwidth (change in the computational context). Each factor can be associated with a set of adaptation strategies that are suitable to re-align the application within the system and/or context requirements. In order to select the adaptation strategy which should be applied, it is necessary to consider that adaptation needs may be associated with other requirements that are important for designing and performing adaptation. In particular, the scope of the change (whether the change affects only a single running instance of the SBA or influences the whole model) and the impact of the change (the possibility of the application to accomplish its current task) should be considered. Depending on these parameters different strategies may apply.

Among the adaptation strategies, it is possible to distinguish domain-independent or domain-dependent strategies. The former are applicable in almost every application context while the adoption of the latter is limited to specific execution environments. Table 1 defines the most common domain-independent adaptation strategies.

The identification of the most suitable strategy is supported by a reasoner that also bases its decisions on multiple criteria extracted from the current situation and from the knowledge obtained from previous adaptations and executions. After this selection, the enactment of the adaptation strategy is automatically or manually performed. The execution of all activities and phases in all runtime phases may be performed autonomously by SBAs or may involve active participation of the various human actors.

2.3. Component-based software engineering

Component-Based Software Engineering (CBSE) is often described as the precursor to service-oriented computing [19]. In CBSE software applications are composed of self contained software components that contain all of the functionality necessary for a system of business process. In component-based applications communication occurs through interfaces that are described with some interface description language. Components do not have to be implemented using the same programming languages providing there is some intermediary mechanism that can serialize data and objects between components.

In several Service-Oriented Software Engineering (SOSE) methodologies components are first implemented before being exposed as services over a network [3]. This shows the high degree of overlap between the two development paradigms. Although, services and components may differ on their levels of granularity. For example, software components are usually composed of more fine grained objects while in turn they often form the basis of more coarse grained services [20].

When looking at service-oriented adaptation techniques it may first be beneficial to review related component-oriented approaches given the relationship between the two paradigms. Oreizy et al. [21] have proposed such an approach for component-based adaptation. Their approach contains two cycles that run in parallel, and evolution management cycle and an adaptation management cycle. The evolution management cycle manages modifications to the application through the use of tools and adaptation mechanisms. Modifications permitted include the addition, removal or replacement of application components and their connectors. The evolution management cycle examines changes to the application and prevents and changes that might make the operation of the application unsafe or inconsistent. The adaptation management cycle on the other collects observations while the application is running. These observations are then analyzed in order to determine if it is necessary to adapt the application. Should

Table 1
Description of the most common domain-dependent adaptation strategies.

Adaptation strategy	Description
Service substitution	Reconfiguration of the SBA with a dynamic substitution of the a service with another one
Re-execution	The possibility of going back in the process to a point defined as safe for redoing the same set of tasks or for performing an alternative path
(Re-)negotiation	Simple termination of the service used on the requester side and re-negotiation of the SLA properties to complex management on reconfiguration activities on the provider side
(Re-)composition	Reorganization and rearrangement of the control flow that links the different service components in the business application
Compensation	Definition of ad hoc activities that can undo the effects of a process that fails to complete
Trigger evolution	Insertion of workflow exception able to activate the application evolution
Log/update adaptation information	Storage of all the information about the adaptation activities for different goals (e.g., service reputation, QoS analysis, outcome of adaptation)
Fail	The system reacts to the changes by storing the system status and causing the failure of the service and re-executing it

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

an adaptation be deemed necessary, then the adaptation cycle provides the necessary tools for deploying alternative components.

3. Research method

There were two phases involved in the construction of the process model during this study. The first phase involved the development of a Frame of Reference (FoR), named SOAdapt-FoR, based on relevant literature. This was enhanced with empirical evidence gathered during the second phase of the study, resulting in the SOAdapt process model. The research design largely follows Ahlemann and Gastl's [22] process for empirically grounded reference model construction.

3.1. Construction of the SOAdapt-FoR model

The SOAdapt-FoR model was constructed by analysing existing publications that contain activities for adapting SBAs. Both peer-reviewed and non-peer-reviewed publications were analyzed to compile these process details. One of the key sources of adaptation activities was a Systematic Literature Review (SLR) of SBA development process models. The SLR, that was carried out by Lane and Richardson [7], identified which of the process models supported SBA adaptation. Many of the activities identified were similar and at varying levels of abstraction. In order to eliminate duplication and to show the activities at the same level of abstraction, each activity was assigned a generic name. Similar activities were assigned the same generic name, resulting in a smaller list of unique generic activities.

In their Systematic Literature Review (SLR), Lane and Richardson identified 8 publications which contain high-level processes and activities for adapting SBAs. Similarly Lane et al. [23] identified several additional publications, during an ad hoc literature search, which contained SBA adaptation activities. Unfortunately, most of the approaches identified the in SLR [7] and the ad hoc search [23] are non-holistic and lacking the required processes to fully guide SBA adaptation. Each of these publications as well as two other papers from Bucchiarone et al. [24,14], were searched, resulting in a total of 50 SBA adaptation activities (see Appendix A).

There is a wide variety of process modeling notations such as the Business Process Modeling Notation[1] (BPMN) or the Unified Modeling Language[2] (UML), and domain specific process meta-models such as the Software and Systems Process Engineering Meta-Model[3] (SPEM) of the meta-model described in ISO/IEC 15288 [25]. BPMN and UML both facilitate the graphical representation of process models which is not the goal of this exploratory research. SPEM is a very detailed meta-model which also uses UML in order to instantiate process models. Rather than these graphical modeling approaches, the aim of this research is to document key process attributes such as activities, tasks, artifacts and stakeholders. The process meta-model defined in ISO/IEC 15288 contains these key process attributes.

The "activity" abstraction level, as defined in ISO/IEC 15288, was chosen for the generic activity names. ISO/IEC 15288 defines activities as a group of related tasks that are required to achieve the outcomes of a process. Using UML class diagram notation, Fig. 1 [25] illustrates the ISO/IEC 15288 process constructs and their interrelations. The figure shows that a Process can have zero or more Sub-Processes and one or more Activities. In turn, it shows that an Activity can have one or more tasks which can have zero or more Notes. It also shows that each process has a Name, a Purpose, and an Outcome. The meta-model will be slightly adapted for this research with the removal of the process outcomes and the addition of stakeholders and artifacts. Stakeholders are the individuals that participate in the activities of a process while artifacts are the physical entities such as code or documentation that are produced by or modified by activities.

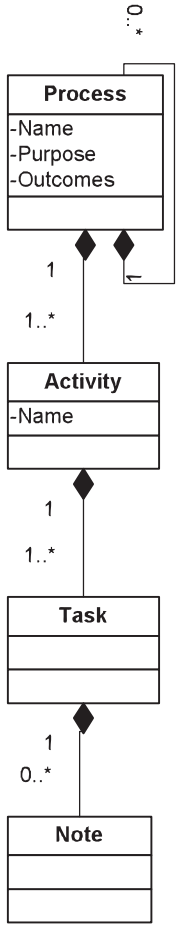

Fig. 1. ISO/IEC 15288 process constructs.

The S-Cube life-cycle, illustrated in Fig. 2 [18], was chosen as a starting point to develop the SOAdapt-FoR model. The S-Cube life-cycle is a high-level life-cycle model skeleton proposed for the development of adaptable SBAs. The generic adaptation activities identified were mapped to the appropriate processes of the S-Cube life-cycle resulting in the SOAdapt-FoR model. This approach was taken in order to leverage the existing research which has been conducted by the S-Cube consortium. Additionally, the S-Cube life-cycle covers each aspect of the development cycle without being too detailed. This allows it to be easily modified and enhanced with activities reported in the service engineering literature. The SOAdapt-FoR model is a high-level model and only uses the Process and Activity constructs as defined in ISO/IEC 15288.

3.2. Construction of SOAdapt

SOAdapt was constructed with data gathered from an expert-opinion survey and an interview-based case study. This data was then used to modify and extend SOAdapt-FoR. Therefore process attributes discovered during the interviews that were not present in SOAdapt-FoR represent the gaps in the literature. Altogether there were 198 passages of interview transcript which were assigned codes because they contained information about adaptation

[1] http://www.bpmn.org/.
[2] www.uml.org.
[3] http://www.omg.org/spec/SPEM/.

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

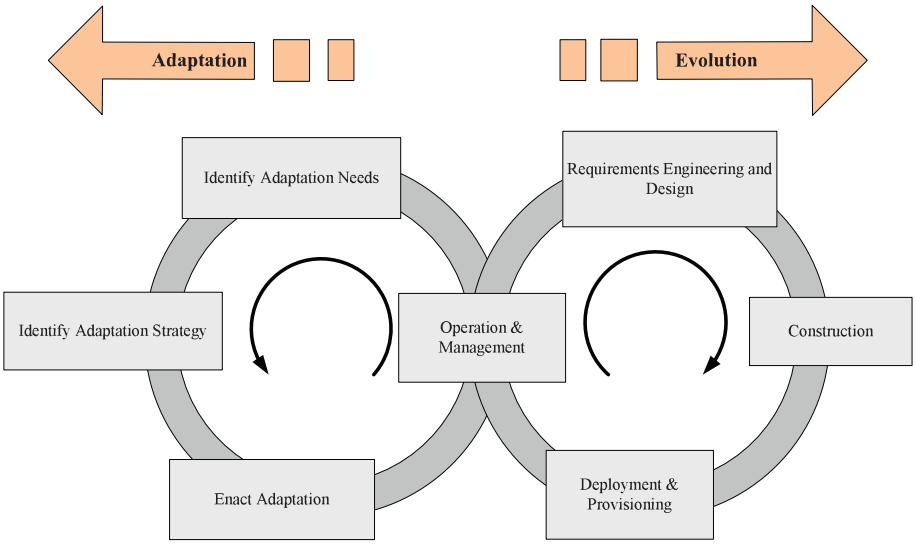

Fig. 2. The S-Cube life-cycle of adaptable SBAs.

processes, activities, tasks, stakeholders, or artifacts. Additionally there were 12 passages of text which were coded because they contained information about the background's of interview participants.

3.2.1. Expert opinion survey

The first stage in the expert-opinion survey was the development of an interview guide for use during the survey interviews. The interview guide was divided in two parts: the first part deals with the participants backgrounds and their roles, while the second part focuses on the elicitation of adaptation activities. Participants discuss the activities within the SOAdapt-FoR model, and to provide feedback. The respondents were asked whether or not they agreed with the activities contained in the SOAdapt-FoR model, and what changes they would make. Then they were asked to describe the tasks required to complete these activities, the stakeholders involved and the artifacts produced by the activities.

The interviews conducted were semi-structured interviews [26]. Interview guides for this type of interview contain open-ended questions which allow interviewers to ask follow on questions when necessary. This flexibility allows the interviewer to ask the interviewees detailed questions about their specific areas of expertise. This methods allowed the model to be expanded in depth as well as breath, existing processes received much more detail while new processes were also added. Background questions were asked to provide context information for each of the participants. Each of the participants were asked back ground questions such as:

- What is you current position?
- Do you work with adaptable Service-Based Applications (SBAs)?

Then they were asked to identify adaptation process attributes that could be used to develop the SOAdapt-FoR further. For example, for each of its activities the were asked:

1. At what stage in the SBA development life-cycle should it occur, or should it occur at all?
2. What stakeholders should be involved in the activity?
3. What lower level activities or tasks should it contain?
4. What artifacts are required by or produced by the activity?

They were asked to describe the SBA development processes within their organisations, focusing specifically on there individual roles, where they could provide the most insight. These questions were used to elicit details of processes and activities that were not contained in SOAdapt-FoR.

A pilot interview was conducted using this interview format. This results of the pilot showed that this approach struck a good balance between flexibility and structure. A strictly structured questioning approach would have prevented the interviewer from probing concepts that were not considered while developing the SOAdapt-FoR model.

Once the interview guide was completed and piloted, the next step was to identify and contact suitable interview partners. This proved to be one of the most challenging aspects of the study given the specialized nature of the process model being constructed. In order to participate, interview partners needed to be experts in Service-Oriented Computing (SOC) and be able to provide opinion based on experience, on how best to adapt SBAs. Suitable practitioners were identified through user groups on the Linkedin[4] professional network as well as SOC conference proceedings. After sending out invitations to over 100 practitioners there were 15 replies with seven eventually committing to interviews. At the end of each interview participants were asked to identify others who may participate with some initial positive responses. However, the initial interview partners did not provide leads for additional interviews indicating that chain sampling can be ineffective for such a specific research project topic. Table 2 provides an anonymous profile of the SOC experts that participated in the survey. The company specific details are intentionally vague to protect their identities.

In advance of the interviews the interviewees were sent a copy of the interview guide as well as some background information on the study, allowing them to familiarise themselves with the concepts and terminologies involved in the study. It also gave them extra time to think about the adaptation activities that they encountered while developing SBAs. The interviews were conducted either on site or through the Skype™Voice Over Internet Protocol (VOIP) service which allows the recording of interviews. Once all of the interviews were complete, they were transcribed in preparation for data analysis.

[4] http://www.linkedin.com.

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 2
Survey participants.

Participant	1	2	3	4
Role	Security R&D	Business analyst	Sales/product development	Service composition R&D, commercialising research
Experience with adaptable SBAs	Yes	Yes	No	Yes
Location	USA	Mexico	UK	Germany
Company	A	B	C	D
Size	Very large multinational	Large	Large	Very large multinational
Sector	Hardware/software development	Banking	Mechanical engineering	Communications

	5	6	7
Role	Performance testing	Consultant	Business process improvement
Experience with adaptable SBAs	No	No	Yes
Location	Germany	Netherlands	USA
Company	E	F	G
Size	Very large multinational	Very large multinational	Very large multinational
Sector	Industrial, healthcare	Software development	Software development/ services

Table 3
Case study participants.

Participant	8	9	10
Role	Chief technology officer	Business analyst	Developer
Experience with adaptable SBAs	Yes	Yes	Yes
Location	UK	UK	UK
Company	H	H	H
Size	Medium	Medium	Medium
Sector	Software development	Software development	Software development

3.2.2. Case study

A single case study was also conducted in order to supplement the survey conducted during this inquiry. The case study was based on a typical company with expertise in developing SBAs. The aim of this study [27] was to identify how a SBA development team approaches the adaptation of SBAs. The company where the case study was carried out will be referred to as SbaSoft.

Interviews were used in order to collect qualitative data during the case study. Three stakeholders involved in different aspects of the application development life-cycle were interviewed: the Chief Technology Officer (CTO), a business analyst and a developer. The interviews for the case study were conducted on site at SbaSoft. Table 3 provides an anonymous profile of the case study participants. Again, the company specific details are intentionally vague to protect its identity.

The interviews for the case study were based on a semi-structured interviews similar to that used in Section 3.2.1. However, rather than focusing on adaptation specific activities, the questions were aimed at documenting SbaSoft's entire development life-cycle including adaptation specific activities. Participants in the case study were asked questions such as:

1. Does your company use service based applications, provide services for internal use or third parties: i.e. service client, service provider, or both.
2. Can you briefly outline how requirements are gathered for your services?
3. Can you briefly outline the process followed for designing web services in your company (i.e. people involved, activities carried out)?
4. Are your services designed to be adaptable?

Their entire development life-cycle was documented in order to view their adaptation specific activities in context. After the interviews were complete they were analyzed and provided input for the model.

3.2.3. Data analysis

Once the data collection from the survey and case study was complete the interview transcripts were analyzed using data analysis techniques as described in Miles and Huberman's [28] expanded source book. The raw data was first open coded with descriptive codes being added to segments of the data. A starting list of codes was based on the frame of reference model constructed in the first half of this study. Starting codes, as illustrated in Table 4, were created for activities, stakeholders and artifacts related to SOAdapt-FoR's high-level processes. New codes were added and existing ones modified to accurately reflect the raw data as the coding proceeded. Participant background information was coded in chunks containing information about their current roles as well experience which provides context for their interview responses.

The coding was conducted with ulQda [29], a LaTeX[5] [5] qualitative data analysis package. This package allows codes to be embedded within the raw data where they can be easily reviewed and edited in context. The package also allows the raw data to be typeset to a high quality document, and annotated with the appropriate codes. Finally, this package allows the codes and their corresponding raw data to be exported to a Comma Separated Value (CSV) file. CSV files are easily imported into spreadsheet applications for subsequent

[5] http://www.latex-project.org/.

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 4
Starting codes.

Category	Process	Code
Activity	Requirements engineering	activity!requirements engineering
	Design	activity!design
	Construction	activity!construction
	Deployment and provisioning	activity!deployment and provisioning
	Operation and management	activity!operation and management
	Adaptation triggers	activity!adaptation triggers
	Adaptation strategy	activity!adaptation strategy
	Enact adaptation	activity!enact adaptation
Artifacts	Requirements engineering	artifacts!requirements engineering
	Design	artifacts!design
	Construction	artifacts!construction
	Deployment and provisioning	artifacts!deployment and provisioning
	Operation and management	artifacts!operation and management
	Adaptation triggers	artifacts!adaptation triggers
	Adaptation strategy	artifacts!adaptation strategy
	Enact adaptation	artifacts!enact adaptation
Stakeholders	Requirements engineering	stakeholders!requirements engineering
	Design	stakeholders!design
	Construction	stakeholders!construction
	Deployment and provisioning	stakeholders!deployment and provisioning
	Operation and management	stakeholders!operation and management
	Adaptation triggers	stakeholders!adaptation triggers
	Adaptation strategy	stakeholders!adaptation strategy
	Enact adaptation	stakeholders!enact adaptation

analysis.her than being described by a single code, each passage was broken down into 5 categories of: processes, activities, tasks, stakeholders and artifacts. Processes, activities and tasks, in this context, relate the the process constructs defined in ISO/IEC 15288 where each process is made up of activities which are in turn made up of tasks. For example, the following passage was assigned the code "activity!operation and management!monitoring" during open coding:

> *I would monitor the state of the task, the external events, also the state of the information, and the documents and the things that are exchanged*

During the refinement process, a high-level process, an activity and 3 tasks were identified, with no stakeholders or artifacts being identified. The process constructs identified during this example are shown in Table 5.

In another example the following quotation provided the activity "Adaptation strategy based on system learning":

> *. . .it would be great to have some type of learning system, in order to be able to learn from the jobs that have been run, failure modes and then to adapt the system according to what it has learnt, and any knowledge we have put in as to why the jobs might fail.*

Once these process attributes were identified they were organized into a conceptually clustered matrix [28]. Similar concepts, in this case the process attributes, were grouped together in to a matrix with activities, stakeholders and artifacts assigned to the appropriate high-level life-cycle processes. They were then ordered into a logical process hierarchy making it is easier to identify duplicate or disjointed attributes. Duplicate attributes were eliminated, while disjointed attributes were reevaluated to see where they might fit into other processes or activities. Finally, the SOAdapt-FoR model was updated to reflect the processes, activities, tasks, stakeholders and artifacts identified during the data analysis. As well as additions, SOAdapt excluded some of the original SOAdapt-FoR constructs to reflect the findings of the empirical inquiry. All of the stakeholders and artifacts from SOAdapt have come from the empirical inquiry since there were none identified in SOAdapt-FoR. A graphical representation of SOAdapt was constructed using the Business Process Modeling Notation (BPMN) notation, showing the processes and activities. This approach was chosen because there were 38 stakeholders, 35 tasks and 17 artifacts identified which would clutter a complete BPMN representation. Tables were constructed to show the complete set of process attributes for each of the processes in the process model.

3.2.4. Evaluation

The model was evaluated to get an impression of its external validity. Several studies were chosen from the literature to be used during this evaluation. These studies were not used in the construction of the reference model since they could not later be used for an impartial evaluation. Oreizy et al.'s process model [30,21] was chosen for the evaluation rather than construction since component-oriented development is related rather than part of service-oriented development. Durvasula et al.'s life-cycle [31] was reserved for evaluation since it could be used to evaluate each of the attributes of our model during a development life-cycle scenario.

To show that the process model is suitable for adaptation it was also compared to a related process model proposed by Oreizy et al. [30,21] for component-based software adaptation. Oreizy et al.'s process model was validated through a worked example where it is used to adapt a cargo routing application. The objective of this comparison is to show that SOAdapt is suitable for runtime adaptation in general but not necessarily component-based adaptation. Therefore, the comparison will be at a high-level of granularity without much emphasis on the lower-level component or service-based entities. If this comparison shows that SOAdapt is useful for adaptation in general, it remains to be seen whether SOAdapt is useful in a service-oriented environment. This will be investigated in the next part of the evaluation.

To demonstrate the transferability of the process model to a service-oriented environment, an evaluation was conducted in order to determine if it is compatible with a SBA development life-cycle proposed by Durvasula et al. [31]. Durvasula et al.'s life-cycle was collaboratively developed by 10 SBA development practitioners from 8 different companies. During the evaluation, each of the activities from the adaptation process model was mapped to life-cycle processes within Durvasula et al. life-cycle. Mapping the activities to the life-cycle without conflict, shows that the process

Table 5
Data analysis example.

Code	Text	Process	Activity	Task	Stakeholder	Artifact
Activity!oam!monitoring	I would monitor the state of the task, the external events, also the state of the information, and the documents and the things that are exchanged	Operation and management	Monitoring	Monitor processes		
		Operation and management	Monitoring	Monitor events		
		Operation and management	Monitoring	Monitor data		

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

model is suitable for use with an empirically based SBA development life-cycle.

4. Company backgrounds

All of the companies that participated in this study have successfully developed adaptable SBAs or frameworks that facilitate the development of adaptable SBAs. These applications consisted of Commercial Off-The-Shelf (COTS) products or bespoke customer solutions. Each of the applications have strengths and weaknesses and each have been developed using different development processes. By reviewing the type of applications developed by the participant companies, we can judge the relevance of the participants to the project. Here are some examples of those applications:

Company A provide a service monitoring COTS framework which facilitates SBA adaptation. The framework monitors application events that may trigger adaptation. These events are also permanently stored for later investigation or troubleshooting. Here is an excerpt from Participant 1's description of the framework:

> . . .*and* it will actually store all of these audit events that come in there and so later even the year two years later you can see was I compliant with this particular control objective during this particular time window two years ago.

On the other hand Company B have implemented a bespoke adaptation solution to facilitate adaptation in their in house enterprise application suite. This solution consists of a service broker that intercepts service and re-routes their path depending on their properties. Here is a description of the application:

> . . .*on one of the systems I worked on there were rules that allowed some adaptation, and we had to interview financial users that they knew the rules, so for example they were saying that if you have a message of number type 100 in one filed, and one field was filled with some value then this message has to be sent to this office if not it has to be sent to some to some other office. But sometimes you have to decide when you receive the message where to send it, so there was a kind of automatic broker.*

Company 4 like Company 1 also provide a COTS adaptation framework, the application focused on compositional adaptation. A problem with this framework is that there is no runtime monitoring or governance which will rescue the application during an expected event, so all adaptation configurations are tested thoroughly during design time.

> *With our mechanism you can use in run time so the adaptation process itself can enforce boundaries but there's no, there's no multi level that will somehow, you know, play some kind of arbitrator role and stop the execution if something goes wrong. But that's exactly the reason why it's so important to actually be able to test in advance and guarantee that nothing will actually go wrong.*

Company H where the case study was conducted was another COTS development company. In this case they developed user interface software for large enterprise customers. Their application allow the consumption of data services that could be merged into collated views. Their framework allows their clients to adapt their SBAs through "drag and drop" admin interfaces. Here is an excerpt from a description of the framework:

> . . .*yeah* customers can do it if they got like some, you know a degree of technical ability, because it is just basically a drag and drop environment, so yeah we do that, quite a lot of customers have wanted to do it for themselves so they get training courses from us.

5. SOAdapt-FoR model

Ten generic adaptation activities were distilled from a set of 50 activities that were identified in from Lane and Richardson's SLR [7] as well as other related literature. The SLR included a methodical search of computer science related digital libraries such as IEEE Xplore and the Web of Science. In total 722 studies were identified in the initial search with 77 found to contain process details for developing SBAs. These SBA development processes were then investigated to see whether or no they support the development of adaptable SBAs. A total of 8 studies from the SLR were found to have supported adaptation. These 8 studies along with two others found in the literature are listed in Appendix A.

The 10 generic activities and their parent processes identified in the S-Cube life-cycle were ordered into a sequential workflow which shows how each activity relates to one another creating a process model. BPMN is used to visualise the model. BPMN has a standard modeling notation which should be understandable by a large number of business as well as technical stakeholders [32]. Fig. 3 illustrates the SOAdapt-FoR model represented in BPMN in contrast with the block diagram used to construct the S-Cube life-cycle.

5.1. SOAdapt-FoR model description

Our description of the SOAdapt-FoR model pays particular attention to the activities and their interrelations. The high level process such as "Requirements Engineering" and "Design" that make up the skeleton of the model are primarily inherited from the S-Cube life cycle model with some exceptions. The "Requirements Engineering and Design" process inherited from the S-Cube life-cycle model was separated into two separate processes to accommodate the numerous activities identified for each process.

The S-Cube life cycle is an amended version of Papazoglou and Van Den Heuvel's [2] service development life cycle with an adaptation cycle added to it's left hand side [18]. Papazoglou and Van Den Heuvel's original life-cycle was influenced by approaches such as the Rational Unified Process (RUP) [1], Business Process Modelling (BPM) [33] and Component-Based Development (CBD) [19].

It is important to reiterate at this stage that the SOAdapt-FoR model is not a complete life-cycle model as it only contains activities that are related to adaptation. These activities are mapped to the S-Cube life-cycle model which indicates where they should occur during the software development life-cycle. The activities could also be mapped to similar processes within other development life-cycles with similar high level process of the S-Cube life cycle.

The frontier between SOAdapt-FoR and its inclusion in an existing life-cycle models depends on the type of model it is being integrated with. In the evolution cycle, adaptation activities should occur before the other activities in the process have been completed. This is necessary so that other activities can see the impact of the activities from SOAdapt-FoR. For example, adaptation may require a particular service granularity and service hierarchy and these aspects need to be decided before the remainder of the application is developed. The activities from the adaptation cycle (left-hand side) are contained in complete processes. Therefore, these processes may occur autonomously during the operation of the application regardless of the application's development life-cycle.

If SOAdapt-FoR was used in a development project employing a waterfall development method, each of the process groups in the evolution cycle would have to be mapped to their equivalent process in the waterfall model. This is a straightforward example as each of the procesess from the evolution cycle of SOAdapt-FoR usually have an equivalent in most waterfall model instantiations. The

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

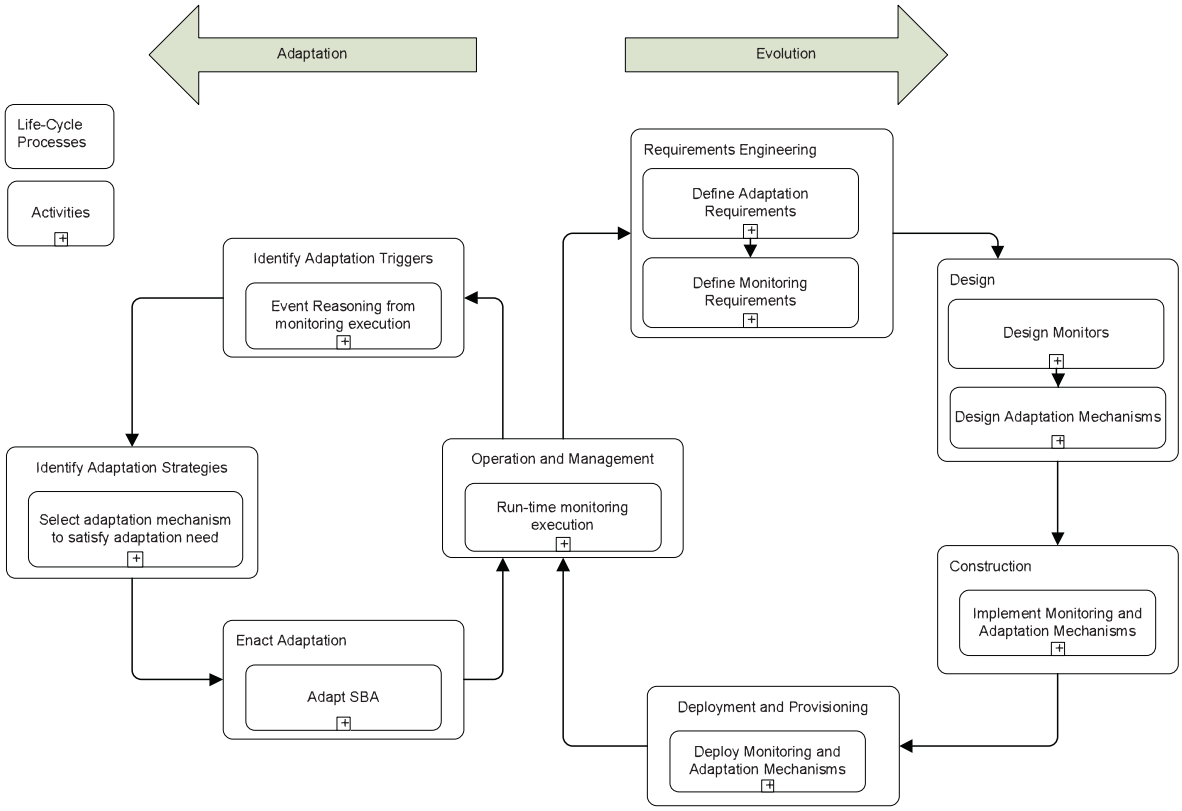

Fig. 3. SOAdapt-FoR model.

adaptation cycle of SOAdapt-FoR would then need to occur in series with the other processes from the waterfall. This would result in a hybrid linear/iterative development life-cycle not to dissimilar to the SCRUM development life-cycle. If was mapped to an existing SCRUM model the result would be a hybrid linear/iterative life-cycle with an iterative development cycle and an iterative adaptation cycle in series with the other linear SCRUM processes.

5.1.1. Requirements engineering

In the SOAdapt-FoR model we are interested in the capturing adaptation specific requirements for SBAs. Adaptable SBAs will also have many other requirements that are not related to adaptation, but those are outside the scope of the SOAdapt-FoR model. Within SOAdapt-FoR there are two requirements engineering activities that are specific to adaptation:

Define adaptation requirements refers to the activity of defining or eliciting requirements relating to why it is necessary for the SBA to adapt. An adaptation requirement may be, for example, if the Quality of Service (QoS) of service A falls below a minimum threshold then the application adapts by choosing an alternative service. The adaptation requirement may also state how the adaptation occurs, adaptation might simply involve choosing between two hard coded service end points or in a more complex way by negotiating with a service directory. An adaptation requirement may also specify whether or not the adaptation occurs with user intervention or automatically with minimal or no user input.

Define monitoring requirements Once adaptation requirements have been determined, monitoring requirements need to be defined in order to monitor service attributes that relate to adaptation requirements. For example, in order to enable an application to adapt based on QoS, it is necessary to monitor relevant service quality attributes such as service latency and service reliability. So, in this case, the *monitoring of service latency and reliability* is a monitoring requirement, relating to the *adapt if service QoS falls below a minimum threshold* adaptation requirement.

5.1.2. Design

There are two adaptation specific design activities within the design process of SOAdapt-FoR:

Design monitors is a design activity where monitors are designed in order to satisfy the monitoring requirements defined during the requirements engineering process. Details such as implementation technology and infrastructure need to be considered at this stage. The monitor designs need to be included with a design specification document or a similar alternative that can be used by the application programmers.

Design adaptation mechanisms. Once monitors have been designed, a means to adapt the application based on monitored events must be designed. The adaptation mechanism must have the ability to respond to monitored events and adapt by enacting an adaptation strategy. Bucchiarone et al. [24] identified many possible adaptation strategies such as substituting services with known alternatives or service re-negotiation which involves negotiation with a service directory in order to locate alternative services. It is at this point that it must be determined which of these strategies will be used in order to adapt during runtime. Once adaptation strategies have been decided upon the technical and infrastructural details of how they are implemented and triggered need to be designed.

5.1.3. Construction

The adaptation specific construction activity included in SOAdapt-FoR is outlined below:

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Implement monitoring and adaptation mechanisms. This activity involves the implementation or coding of the monitoring and adaptation mechanisms that were discussed in Section 5.1.2. This activity would occur along with all of the other application implementation activities within the software development project. There are no large design features to be decided at this point as they are specified during the design process. However, due to the complex nature of adaptation mechanisms, this implementation in one of the most challenging parts of the development process.

5.1.4. Deployment and provisioning

The S-Cube life-cycle has inherited its deployment and provisioning process from Papazoglou and Van Den Heuvel's [2] service development methodology, which itself in influenced by the RUP methodology. The process specifies how a software system is deployed as well as provisioned or metered, which would allow charging for a service. Here is an outline of the single, adaptation specific, deployment and provisioning activity from the SOAdapt-FoR model:

Deploy monitoring and adaptation mechanisms. The deployment of monitoring and adaptation mechanisms are most likely going to get deployed with the other parts of the SBA and may not require any special treatment. However, if the monitoring or the adaptation mechanisms are built into a service infrastructure which is external to the application they will have different deployment requirements.

5.1.5. Operation and management

The operation and management of the SBA is one of the most critical processes with respect to adaptation. Whether its manual or automatic, its during the operation of the application when certain monitored events prompt the adaptation of the application. Within the SOAdapt-FoR model and the S-Cube life-cycle, the operation and management process is the link between the evolution cycle which undergoes typical maintenance and the adaptation cycle which allows the application to adapt. As well as providing the link between the adaptation and evolution cycles it also has its own activities, and one in particular that relates to adaptation:

Run-time monitoring execution. Run-time monitoring is a key activity within the operation and management process; it monitors events that signal whether or not run time adaptation should take place. This could be a manual activity or fully automatic depending on the application, but most likely it would be necessary to have some form of involvement by a relevant stakeholder.

5.1.6. Identify adaptation triggers

Identify adaptation triggers is a process which makes use of monitoring mechanisms in order to identify reasons to adapt. This process occurs during the adaptation cycle, thus all of its sub-activities are related to adaptation. Within SOAdapt-FoR there is one activity in this process which will now be explained:

Event reasoning from monitoring execution is an activity that involves the identification of monitored events from application monitors. These events may be reasoned or identified manually depending the complexity of the application. In an fully automated system events would be identified based on application logic, however, in a manual scenario, events might be manually identified by end users or developers through inspection of application logs.

5.1.7. Identify adaptation strategies

During this process, adaptation strategies are selected to suit the monitoring events that were identified in the previous process. Different event types may be associated with different adaptation strategies, and again, this association may be built into the applications logic may be made at runtime by a stakeholder. One scenario is that an application has many built in adaptation strategies where the most appropriate one is selected by an end user at runtime.

Select adaptation mechanism to satisfy adaptation need. This activity is the only activity within the identify adaptation strategies process. This activity completes all of the goals of its parent process which is to select and adaptation mechanism which is most suited to the monitored event that triggered the adaptation. The specifics actions within this activity are difficult to determine with out having some knowledge about the adaptation engine used in the underlying application.

5.1.8. Enact adaptation

This process is the last process in the adaptation cycle which simply involves the enactment of the adaptation strategy that was selected. Within the context of SOAdapt-FoR, this process has only one activity which is to adapt the SBA.

Adapt SBA. This activity refers to physically adapting the application. If it is an autonomous application this activity is automatic. If the application is not autonomous enacting the adaptation of the SBA may require configurations modifications by end users or developers.

6. SOAdapt

In this section we will present the results of the content analysis that carried out on the 10 interview transcripts (275 min of audio) conducted during this study. Once the content analysis was complete the SOAdapt-FoR model was updated to represent the study's findings. Fig. 4 is a BPMN representation of the empirically grounded process model. The illustration shows high-level life-cycle processes and activities. Tasks, stakeholders and artifacts are listed separately in the following sub-sections.

6.1. Processes and activities

Interview participants agreed that the high-level processes from the SOAdapt-FoR model should remain largely unchanged and that adaptation activities should occur in the traditional requirements engineering, design, construction, deployment and provisioning and operation and management phases. However, a need for a separate testing process was identified. Another notable deviation from the original high-level processes was the lack of emphasis placed on the provisioning sub-process from the deployment and provisioning process.

The processes from the adaptation cycle of the SOAdapt-FoR model caused confusion among some participants during the interviews. There was confusion between terms such as strategies, mechanisms and their applications. These processes as well as their contained activities have now been modified to make them more intuitive.

Changes to the activities contained within the high-level processes were more extensive than changes to their parent processes. This is to be expected given that the high-level life-cycle processes from the evolution cycle are almost De facto, featuring in prominent life-cycle models such as RUP, ISO/IEC 12207 and

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

the waterfall model [34]. All of the changes made to the processes and activities from the SOAdapt-FoR are listed in Table 6. The first column of the table shows whether the processes or activities were added or removed. In most of the cases activities were modified rather than being removed completely. This is shown in the table as a *remove* followed by an *add* of the modified version. Rows with both a *process* and *activity* entry refer to the activity and show the process for reference, while rows with just a process entry refer directly to the process. In cases where there is no directly supporting data, changes were made to activities so that they correspond to the activities that have supporting data. For example, the modification of "Deploy monitoring and adaptation mechanisms" to "Deploy adaptation strategies and monitors" did not have directly supporting evidence. This activity was changed in order to correspond with the activity "Design adaptation strategies" which did have supporting data. The "Provisioning" sub-process was removed from the "Deployment and provisioning" process, not as a result of comments from the interviewees to remove it, but because when probed, they never offered any details on adaptation activities that need to occur during provisioning.

6.2. Detailed process attributes

The following subsections provide an outline of the activities in each of the process groups from SOAdapt. The activities and tasks in Tables 7–15 are color coded depending on whether or not they can be modified to meet the needs of a particular project scenario. The main flexibility points relate to whether or not the type of adaptation chosen for the project will be static (adaptation paths determined at design time) or dynamic (adaptation paths chosen at runtime).

6.2.1. Requirements engineering

The two Requirements Engineering (RE) activities from the SOAdapt-FoR model remain unchanged in SOAdapt, with three tasks having being identified for the "Define Adaptation Requirements" activity and 2 for "Define monitoring requirements" (see Table 7). There were several types of adaptation triggers identified for "Define events that trigger adaptation" task. For example, learning-based, context-based, availability based, rule-based and QoS Based triggers were identified. The availability and QoS based adaptation trigger was elicited form the following quotation:

> . . .*so* if one of the systems go down, to be able to say, hey one of the systems has gone down, identify what are possible reasons for it to go down, and then make decisions based on the information that you've got as to whether to retry it on that system, retry it on another system, go back to the customer, or what ever, thats the kind of think that we were looking at.

These adaptation activities and tasks are self contained and can be carried out in parallel with the other RE tasks that are necessary for a project. A number of adaptation related artifacts and stakeholders were also identified for the process.

Unlike the activities and tasks, the artifacts and stakeholders identified are not exclusively concerned with adaptation. For example, all of the adaptation stakeholders listed here usually have other roles in other RE activities.

6.2.2. Design

There was one prominent activity change between the SOAdapt-FoR and SOAdapt for the design process. The "Design Adaptation Mechanisms" activity was changed to "Design Adaptation Strategies". Interviewees felt that Mechanisms was too ambiguous to describe the was in which SBAs adapt:

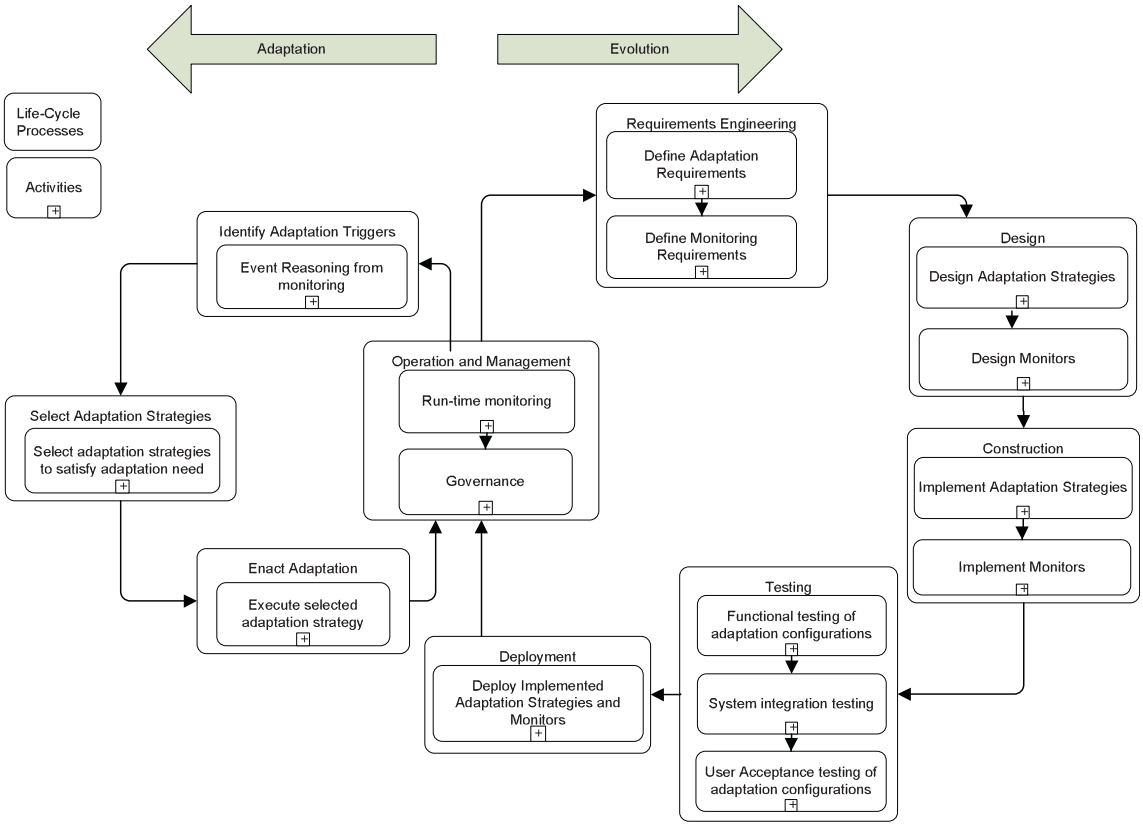

Fig. 4. SOAdapt – process model for adapting SBAs.

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 6
SOAdapt process and activity changes.

Add/ remove	Process	Activity
Remove	Design	Design adaptation mechanisms
Add	Design	Design adaptation strategies
Remove	Construction	Implement monitoring and adaptation mechanisms
Add	Construction	Implement adaptation strategies
Add	Construction	Implement monitors
Remove	Provisioning	
Remove	Deployment	Deploy monitoring and adaptation mechanisms
Add	Deployment	Deploy adaptation strategies and monitors
Remove	Operation and management	Run-time monitoring execution
Add	Operation and management	Run-time monitoring
Remove	Identify adaptation triggers	Event reasoning from monitoring execution
Add	Identify adaptation triggers	Event reasoning from monitoring
Remove	Identify Adaptation strategies	
Add	Select adaptation strategies	
Remove	Select adaptation strategies	Select adaptation mechanisms to satisfy adaptation need
Add	Select adaptation strategies	Select adaptation strategies to satisfy adaptation need
Remove	Enact adaptation	Adapt SBA
Add	Enact adaptation	Execute selected adaptation strategy
Add	Testing	

"Yeah, so I would call these three things out on the right side that I design my monitor, my strategy and my adaptation. I do those three things because my monitor leads to correct strategy".

Adaptation strategies such as service discovery and service substitution were also identified during the inquiry. There were eight adaptation tasks identified for designing these adaptation strategies, along with two tasks for designing monitors (see Table 8). There were eight adaptation artifacts and four adaptation stakeholders also identified for this process. The artifacts and stakeholders identified, like the RE process, are not exclusive to adaptation related activities. In this process artifacts such as "Functional Design" documents are contributed to be adaptation activities, but are also contributed to be non-adaptation activities.

6.2.3. Construction

The construction activity "Implement monitoring and adaptation mechanisms" was updated to "Implement adaptation strategies" to reflect the new design activity "Design adaptation strategies". The "Implement Monitors" activity was isolated as a separate activity to show the "Construct monitors" task in its specific context. A noteworthy task identified for the implementation process was "Early Testing" with the other tasks specifically related to construction. The complete adaptation process attribute details for the construction process, including three artifacts and two stakeholders, can be seen in Table 9.

6.2.4. Testing

The testing process was added to SOAdapt as a result of three testing activities that were identified by interviewees. Along with the testing process there were adaptation related testing activities identified for the "Design", "Construction" and "Deployment" processes. Since the testing activities were identified from scratch, rather than being derived from SOAdapt-FoR activities, the interviews did not provide a lot of detail about tasks, artifacts and stakeholders. The activities and single task identified are illustrated in Fig. 10.

Table 7
Adaptation process attributes for the requirements engineering process.

Requirements engineering
Adaptation activities and tasks
Define adaptation requirements:
– Define events that trigger adaptation (learning-based,context-based, availability based, rule-based, QoS Based)
– Choose between manual and automatic adaptation
– Validate adaptation requirements
Define monitoring requirements:
– Define application components that have to be monitored
– Define how monitored events are reported

Adaptation artifacts
Adaptation workflow
Use cases
Architectural overview

Adaptation stakeholders
Business analyst
Business stakeholders
Solution architects
Engineers
Developers
Information security team
System administrators
End users

Table 8
Adaptation Process Attributes for the Design Process.

Design
Adaptation activities and tasks
Design adaptation strategies:
– Design adaptation strategies (service discovery, service substitution)
– Define adaptation workflows
– Design adaptation dashboard (manual adaptation)
– Design logic to match monitored events to adaptation strategies (automatic adaptation)
– Design service interfaces
– Choose service alternatives
– Use guide for selecting services
– Adaptation strategies prototyping
– Design tests for adaptation strategies
– Design monitoring services (post-mortem, real time monitoring)
– Determine performance impact of monitors
Design monitors:
– Design monitoring services (post-mortem, real time monitoring)
– Determine performance impact of monitors

Adaptation artifacts
Service topology
Interface definitions
Input/output data
Software specifications
Functional design
Enterprise architecture

Adaptation stakeholders
Requirements engineers
Solution architects
Developers
Maintainers

6.2.5. Deployment

The "Deployment" process originated as the "Deployment and Provisioning" process from the SOAdapt-FoR model. As previously mentioned (Section 6.1) the "Provisioning" process was dropped because interviewees did not mention it in relation to adaptation. The process contains one adaptation activity: "Deploy Adaptation Strategies and Monitors" which was changed from

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 9
Adaptation process attributes for the construction process.

Construction
Adaptation activities and tasks
Implement adaptation strategies:
– Construct adaptation strategies
– Implement adaptation dashboards
– Early testing
– Refinement
Implement monitors:
– Construct monitors
Adaptation artifacts
Middleware
Technical design
Dashboards
Adaptation stakeholders
Engineers
Maintainers

Table 10
Adaptation process attributes for the testing process.

Testing
Adaptation activities and tasks
Functional testing of adaptation configurations:
– Prove to customer that application will work in all adaptation cases
System integration testing
User acceptance testing of adaptation configurations
Adaptation artifacts
Adaptation stakeholders

Table 11
Adaptation process attributes for the deployment process.

Deployment
Adaptation activities and tasks
Deploy adaptation strategies and monitors:
– Deploy components on separate systems
– Monitor system dependancies
– Functional testing
Adaptation artifacts
Adaptation stakeholders
Developers
Maintainers
Service providers
Service consumers

Table 12
Adaptation process attributes for the operation and management process.

Operation and management
Adaptation activities and tasks
Run-time monitoring:
– Audit service delivery infrastructure
– Notify users of critical events
– Monitor processes
– Monitor events
– Monitor data
Governance:
– Govern adaptation
– Negociate SLAs
Adaptation artifacts
State model
Adaptation stakeholders

Table 13
Process attributes for the identify adaptation triggers process.

Identify adaptation triggers
Adaptation activities and tasks
Event reasoning from monitoring
Adaptation artifacts
Adaptation stakeholders

Table 14
Process attributes for the select adaptation strategies process.

Select adaptation strategies
Adaptation activities and tasks
Select adaptation strategies to satisfy adaptation need:
– Select adaptation strategy based on recommendation
– Select adaptation strategy from an end used adaptation dashboard
– Select adaptation strategy to satisfy adaptation need
Adaptation artifacts
Adaptation stakeholders
End user

Table 15
Process attributes for the enact adaptation process.

Enact adaptation
Adaptation activities and tasks
Execute selected adaptation strategy
Adaptation artifacts
Adaptation stakeholders
End user

"Deploy Monitoring and Adaptation Mechanisms" in order to match the terminology of preceding processes. The deployment of monitors and strategies were kept as a single activity as the deployment tasks apply generally to both. As illustrated in Table 11, there were 0 adaptation artifacts and four adaptation stakeholders identified.

6.2.6. Operation and management

The Operation and Management process has retained its monitoring "Run-time monitoring", from the SOAdapt-FoR model, and has gained an activity called "Governance". The governance activity, as the name implies, contains tasks for governing how the SBA adapts during run-time. The "Run-time monitoring" activity has been updated with tasks for monitoring data, events and processes as well as a user notification task. The "Governance" activity has tasks for the governing the adaptation as well as negotiating Service-Level Agreements (SLAs). A "State model" was the only adaptation artifact identified without any relevant stakeholders being identified (see Table 12). A state model is an important artifact in the operation of autonomous adaptable SBAs. The state model is the blue print which tells the application how to respond to certain states or events.

6.2.7. Identify adaptation triggers

The "Identify adaptation triggers" process has not changed in SOAdapt, it still retains the single Activity: "Event Reasoning from monitoring", with no lower level tasks. However, this activity was confirmed as a relevant adaptation task during the interviews:

> *"What are the kind of criteria that we would want for this type of adaptation, the trigger points for doing some adaptation".*

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

As Table 13 illustrates, there were no adaptation artifacts or stakeholders identified during the interview process.

6.2.8. Select adaptation strategies

The "Select adaptation strategies" process, like the "Identify adaptation triggers" process, retains its original activity from the SOAdapt-FoR model. However, in this case the 'Select adaptation strategies" activity has been updated with three tasks for selecting adaptation strategies. A single stakeholder, "End user", has also been identified for the process (see Table 14). End users play an important role in identifying the correct adaptation strategies when the adaptation process is not fully automatic.

6.2.9. Enact adaptation

The final process, "Enact adaptation", has also not had many process attributes identified. The wording of its single activity "Execute selected adaptation strategy" had been reworded to match the terminology of other activities but its intent remains the same. The "End user" stakeholder has also been identified for this process (see Table 15). Again end users have the final say on enacting adaptation when the application is not fully automatic.

7. Evaluation

We employed two methods to determine the external validity of the process model developed during this research. In Section 7.1, we describe the comparison between the process model and a previously validated component-based software adaptation process model, indicating whether the model contains the required activities for adaptation. In Section 7.2, the activities from the process model are mapped to an SBA development life-cycle which demonstrates their transferability.

7.1. Inter-model evaluation

To evaluate SOAdapt's capability of adapting SBAs we compared to a similar model proposed and validated by Oreizy et al. [30,21] for adapting component-based applications. Component-based applications are similar to service-based applications, with both application types being composed of loosely coupled self-contained software modules. This similarity allows comparisons to be made between the two approaches. However, the different development paradigms being addressed by each means that a like for like comparison is not being made.

The two models will be compared at an activity level since Oreizy et al's model does not specify the life-cycle processes within which its activities occur. Each of Oreizy et al.'s activities will be analyzed to see if they have equivalent or comparable activities within SOAdapt. The degree of similarity between the activities will also be judged by degree of similarity between their component tasks. Oreizy et al.'s do not explicitly list the tasks within each activity, however they are described verbally within their paper. These task descriptions have been extracted for the comparative analysis.

7.1.1. High-level comparison

As with SOAdapt, Oreizy et al.'s adaptation model has two cycles: an adaptation cycle and an evolution cycle. Both models implement changes to applications during an offline evolution cycle. However, the overall approaches for adaptation are different. In SOAdapt, adaptation mechanisms are built into the application at runtime. Then changes can be made to the operational application using these mechanisms. Oreizy et al change the composition of the application during evolution, while during adaptation implement the changes enacted during evolution. They argue that making changes during evolution is safer because it allows the application to be tested thoroughly before reflecting those changes in the running system. This is a valid consideration which is addressed in SOAdapt by the task "Prove to customer that application will work in all adaptation cases". This task is completed during the testing process and ensures that possible application configurations are tested off line during evolution before the application is made operational. The differences in the adaptation approach may be rooted in the fact that components are more tightly coupled than services and hence require a greater deal of testing and integrity checks during evolution. The two approaches share many of the same activities. For example, both approaches have runtime monitoring processes to detect when adaptation should be enacted. Both methods also have activities to govern and manage runtime adaptations.

7.1.2. Detailed comparison

Table 16 was constructed in order to get a better picture of whether SOAdapt contains equivalent adaptation activities and tasks to those outlined by Oreizy et al. The left hand column contains the complete list of activities and tasks from Oreizy et al. while the right hand side shows comparable or similar tasks and activities from SOAdapt.

It is evident from the table that all of the adaptation activities from Oreizy et al. have comparable activities in SOAdapt. This shows that, at a high-level at least, SOAdapt contains the basic ingredients for adaptation. The table does not show the adaptation activities identified in SOAdapt which do not have counterparts in Oreizy et al.'s model. SOAdapt goes beyond the basic set of activities for adaptation and considers adaptation more explicitly during evolutionary processes such as design, deployment and testing. There are some activities where Oreizy et al. has more detail than SOAdapt particularly with regard to its monitoring activities. The similarities that can be seen between the two models suggest that SOAdapt is a valid adaptation model. This, however, is based on the assumption that Oreizy et al.'s model is itself valid and that adapting component-based applications is a comparable process to adapting SBAs.

7.2. Transferability

In order to determine the transferability of SOAdapt, each of its activities were mapped to a component-based application development life-cycle. This life-cycle proposed by Durvasula et al. [31], suggests a high-level process as well as best practice guidelines for developing SBAs. The best practice guidelines are given in a non-sequential format, providing practitioners with a menu of best practices that they can select to use in conjunction with the proposed life-cycle process model. If the activities and tasks from SOAdapt can be intuitively mapped to Durvasula et al.'s development life-cycle, this shows the transferability of the model to a real-world development scenario.

Table 17 illustrates Durvasula et al.'s development life-cycle annotated with the activities from SOAdapt which are highlighted in bold. The granularity of development activities from Durvasula et al.'s life-cycle and SOAdapt are somewhat different, with Durvasula et al. embedding some life-cycle processes within others. For example, Deployment is embedded as an activity within the IT Operations process. This is not a problem, but it does have some knock on effects to the mappings contained in the Table 17. Some of the activities from SOAdapt are represented as tasks to satisfy the different level of process granularity used by Durvasula et al.

It is evident from Table 17 that the activities from SOAdapt fit into the SBA development life-cycle indicating their transferability. Changes in granularity have been made in order to make the mapping. However, this is to be expected given the variation of process

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 16
Oreizy et al. vs SOAdapt.

Oreizy et al.	SOAdapt
A1: Maintain concistancy and system integrity	Governance/design adaptation strategies/functional testing of adaptation configurations
T1: Preserve consistant model of application architecture	Prove to customer that application will work in all adaptation cases
T2: Preserve strict correspondence between architectural model and execution implementation	Govern adaptation
T3: Develop architecture evolution manager (AEM) to mediate changes to the AEM	Design adaptation dashboard (manual adaptation)
A2: Enact changes	Implement adaptation strategies
T4: Use an interactive editor to construct architectures and describe modifications	
T5: Use design tools to critique architectures as they are constructed	Early Testing
T6: Use domain specific tools to check for semantic errors	Refinement
T7: Use modification interpreter to interpret change description language scripts as AEM primitives	Construct adaptation strategies
A3: Collect observations	Run-time monitoring
T8: Observe and provide notifications of exceptional events such as resource shortages	Monitor events/notify users of critical events
T9: Dynamic modification of monitors and monitored events	
T10: Model application behavior as patterns of events	
T11: Remote monitoring by human actors	Audit service delivery infrastructure
A4: Evaluate and monitor observations	Event reasoning from monitoring/functional testing of adaptation configurations
T12: Monitor behaviors of the running application and compere them to behavioral requirements	
T13 Determine all possible architectural configurations to use for concistancy checks	Prove to customer that application will work in all adaptation cases
T14: Automatic runtime concistancy checks	
A5: Plan changes	Define adaptation requirements/define monitoring requirements
T15: Plan which observations are required for enacting adaptations	Define events that trigger adaptation (learning-based,context-based,availability based, rule-based, QoS based)
T16: Plan which adaptations to make and when to make them	Define application components that have to be monitored
A6: Deploy change descriptions	Execute selected adaptation strategy
T17: Use AEM to translate change descriptions into specific application modifications	

attribute granularity in SBA development process models such as those reviewed in [7].

It is also necessary to evaluate the degree of transferability between the deliverables or artifacts of each process model. The mapping in Table 18 was constructed in order to see the level of compatibility between the artifacts. The first column lists the artifacts from SOAdapt while the third column shows the artifacts from Durvasula et al.'s life-cycle. The second column shows how the artifacts from SOAdapt would relate to the artifacts from Durvasula et al.'s life-cycle during a development project.

In the requirements engineering, design and construction processes all of the artifacts can be mapped to Durvasula et al.'s life-cycle except for the software specifications and functional design artifacts. These are common artifacts and are often present in software development projects. They may not be mentioned by Durvasula et al. as they focus on service-specific artifacts. The state model artifact from the operation and management process was the only other artifact without a match in the mapping. This shows a high level of interoperability between the artifacts of the two models. Mapping the artifacts from SOAdapt has had no negative impact on the artifacts already present in Durvasula et al.'s life-cycle. The artifacts in Durvasula et al.'s life-cycle focus on development of service-oriented applications in general while the artifacts in SOAdapt are primarily concerned with adaptation. Therefore, the artifacts from SOAdapt interact with those in Durvasula et al.'s life-cycle as specialized artifacts which enhance the overall development process.

8. Discussion

The research presented in this paper has addressed a gap. In the first instance, while there have been many implementation techniques proposed for adapting services and SBAs, they are not process focused. Secondly, studies have not been based on the experiences and opinions of expert SBA development practitioners. When developing a process model for adapting SBAs it is important to look at the broader perspective of the entire SBA development life-cycle, taking into account the various actors and stakeholders involved. By analysing interviews carried out with developers, analysts, architects, and business stakeholders from various domains, a useful real-world SBA adaptation process model has been constructed.

The approach taken in this paper has leveraged the existing literature, which was used to construct a conceptual framework to guide the fieldwork and qualitative data analysis. There are arguments against this approach, with some suggesting [35] that the use of a conceptual framework influences research participants and results in biased findings. Miles and Huberman [28], however, argue that all qualitative research is inherently biased by the researcher and that the benefits of using a conceptual framework outweigh the drawbacks. In completing this study, we found that participants offered insights that they might not have thought about in the absence of prompts from the conceptual framework.

The data analysis conducted followed a best practice positivistic approach which allows the results to be audited and reviewed by secondary researchers. This rigorous, repeatable approach aims to eliminate researcher bias and enhance the credibility of the research. Furthermore, this approach allows for the findings to be extended by conducting additional interviews and analysing the additional data with the original analysis method.

It was discovered during the interview process that participants were able to readily relate to the processes in the evolution cycle and to provide relevant insights into the adaptation activities that should occur during these phases. This is likely to be due to the fact

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table 17
Mapping SOAdapt to Durvasula et al.'s SBA life-cycle.

Life-cycle process	Activity	Tasks
Requirements and analysis	Map high-level business processes	
	Prioritise business services	
	Capture business service requirements	
	Define adaptation requirements	
	Define monitoring requirements	
	Architecture review	Review requirements
		Review alternatives and estimate effort
		Propose solution
	Prioritise and add to solutions portfolio	
Design and development	Assign resources to solutions development team	
	Design solutions – identify reuse opportunity	
	Design adaptation strategies	
	Design monitors	
	Develop, QA and conduct UAT for business solution	Implement adaptation strategies
		Implement monitors
		Functional testing of adaptation configurations
		System integration testing
		User acceptance testing of adaptation configurations
IT operations	Assign resources to service operations teams	
	Identify infrastructure needs and establish systems environment	
	Deploy business solution	Deploy adaptation strategies and monitors
	Maintain solution to business requirements	Run-time monitoring
		Governance
	Identify adaptation triggers	Event reasoning from monitoring
	Select adaptation strategies	Select adaptation strategies to satisfy adaptation need
	Enact adaptation	Execute selected adaptation strategy

Table 18
Mapping SOAdapt's artifacts to Durvasula et al.'s artifacts.

SOAdapt artifact	Relationship	Durvasula artifact
Requirements engineering		
Adaptation workflow	is integrated into	Design models: UML, SCA service assembly model and others
Use cases	is integrated into	Design models: UML, SCA service assembly model and others
Architectural overview	uses	Bindings: JMS, RMI, IIOP, HTTP (s), and others
Design		
Service topology	is integrated into	Design models: UML, SCA service assembly model and others
Interface definitions	is integrated into	Design models: UML, SCA service assembly model and others
Input/output data	is integrated into	Design models: UML, SCA service assembly model and others
Software specifications		
Functional design		
Enterprise architecture	uses	Bindings: JMS, RMI, IIOP, HTTP (s), and others
Construction		
Middleware	uses	Product specific metadata for configuration as well as service execution
Technical design	is used to guide development of	Source code, Jave documents, release notes
Dashboards	form part of	Source code
Operation and management		
State model		

that these life-cycle processes are present in many of the formal development life-cycles that are reported in the literature. Respondents found it more difficult to relate to the processes introduced in the adaptation cycle, which resulted in significantly less data being collected for this these processes. Another interesting finding was that many practitioners did not consider the use of a service specific life-cycle. They normally develop SBAs using traditional waterfall style methods. However, on reviewing the frame of reference model with an additional adaptation cycle they considered this to be a useful approach.

8.1. Threats to validity

The primary threat to the validity of this research is the generalisability of the adaptation model constructed. Every effort was made to select interview partners from a diverse set of companies and a diverse set of roles, but this does not guarantee that the results are generally applicable. In order to mitigate these risks the process mode was mapped to a SBA development life-cycle that was empirically developed independently of this research. The goodness of fit that was achieved during the mapping helps to show the model can be generally applied.

In the absence of having used the model in a real-life scenario it is difficult to determine its capability of adapting SBAs. There are two conceivable methods of determining its real world adaptation capabilities, the first being to actually use it in a real adaptable SBA development project, the second being to compare it to a similar model that has been validated in a real scenario. The latter option was chosen mainly for practical reasons. While it is arguably better to validate the model in real scenario, there is also a lot to be learned by comparison with an existing approach. In this paper the comparison shows that the adaptation process model does contain the activities required to safely adapt an SBA during runtime.

The final threat considered is whether or not the data collected reached a point of saturation where activities and tasks identified

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

Table A.19
Adaptation activities.

Source	Adaptation activity	Generic activity
Lane et al.	Monitor message sequences among services and its partners	Runtime monitoring
Lane et al.	Runtime service discovery	Adapt SBA
Lane et al.	Requirements and analysis stage: define KPIs and management policies	Define Adaptation Requirements
Lane et al.	Service specification: identify the service properties to specify	Define Adaptation Requirements
Lane et al.	Specifying service decision model	Define Adaptation Strategies
Lane et al.	Set warning thresholds and alerts for compliance failures	Define Monitoring Requirements
Lane et al.	Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL)	Define Monitoring Requirements
Lane et al.	Service deployment: deploy the monitoring rules and recovery policies within the monitoring system	Deploy Monitoring Mechanisms
Lane et al.	Gather QoS metrics on the basis of SLAs	Design Monitors
Lane et al.	Evaluate SLA QoS metrics	Design Monitors
Lane et al.	Readjust service weights for request queues	Design Monitors
Lane et al.	Designing service adapters	Design Adaptation Mechanisms
Lane et al.	Recovery management: identify, by looking at the monitoring data, the needs for a recovery action	Event reasoning from Monitoring
Lane et al.	Detect protocol violations	Runtime monitoring
Lane et al.	Monitor service, application, middleware, OS, hardware, and network	Runtime monitoring
Lane et al.	Monitor workloads	Runtime monitoring
Lane et al.	Insert monitoring rules and recovery actions in concrete parts of the service composition executable description	Runtime monitoring
Lane et al.	Monitor services	Runtime monitoring
Bucchairone et al.	Define adaptation and monitoring requirements	Define Adaptation Requirements/ Define Monitoring Requirements
Bucchairone et al.	Design for monitoring and adaptation	Design Monitors/Design Adaptation Mechanisms
Bucchairone et al.	Construction of monitors and adaptation mechanisms	Implement Monitoring and Adaptation Mechanisms
Bucchairone et al.	Deployment-time adaptation	Define Adaptation Requirements
Bucchairone et al.	Run-time Monitoring	Runtime monitoring
Bucchairone et al.	Decide between adaptation and evolution	Define Adaptation Requirements
ProDAOSS	Organisational services realization paths are documented by a dynamic service hypergraph	Define Adaptation Requirements
ProDAOSS	Organisational services are designed as service centers in the architectural design discipline	Define Adaptation Requirements
ProDAOSS	Services realization environment is open and adaptable through the use of a reinforcement learning algorithm and a probabilistic reputation model	Event reasoning from Monitoring
PLASTIC	Runtime Analyser	Runtime monitoring
PLASTIC	SLA monitor	Runtime monitoring
PLASTIC	On-line validation	Runtime monitoring
PLASTIC	Evolution policies decoder	Event reasoning from Monitoring
BCDF	Take enterprises business and technical requirements as well as dependencies between them into consideration	Define Adaptation Requirements
BCDF	Enterprises describe their purpose and high level requirements for a business collaboration	Define Adaptation Requirements
BCDF	Define the operational conditions under which businesses can cooperate	Define Adaptation Requirements
BCDF	Negotiation agreement describing the interactions among the services from the different parties	Design Adaptation Mechanisms
Chang	Defining Target Services	Define Adaptation Requirements
Chang	Defining Unit Services	Define Adaptation Requirements
Chang	Planning service component acquisition	Design Adaptation Mechanisms
Chang	Acquiring service components	Design Adaptation Mechanisms
Chang	Composing services	Adapt SBA
Multi-view SOAD	Develop use case models	Define Adaptation Requirements
Multi-view SOAD	Service identification by viewpoint	Define Adaptation Requirements
Multi-view SOAD	Service interface conception	Design Adaptation Mechanisms
Multi-view SOAD	Mapping to platform specific models and code generator	Event reasoning from Monitoring
CSOMA	Model services with variability points	Define Adaptation Requirements
CSOMA	Select orchistration schema based on the context of the incoming request	Event reasoning from Monitoring
CSOA	Design adaptation views for Platform Specific Models (PSMs)	Define Adaptation Requirements
Dino	Develop UML2 Modes Model	Define Adaptation Requirements
Dino	Generate requirements and capabilities of broker services	Define Adaptation Requirements
Dino	Set dynamic service broker configuration	Design Adaptation Mechanisms

begun to re-occur in the later interviews. The later interviews did repeat much of the earlier findings with the exception of the life-cycle processes in the adaptation cycle. Had the interview questions relating to the adaptation cycle been focused on more during the interviews this would have resulted in more data. The problem seemed to be that the questions for the evolution cycle often took a lot of time and by the time the adaptation cycle questions came around the respondents enthusiasm diminished resulting in less data being collected. A solution may have been to focus solely on the adaptation cycle in later interviews, however, this would have been at the cost of valuable data collected by the evolution related questions. We plan to take this approach in future research.

8.1.1. Validation

Lincoln and Guba [36] argue that qualitative research should be internally valid, externally valid, reliable and objective. In this paper we focused on internal and external validity. Internal validity relates to "how" the research is carried out, and whether the methods used are credible. Two suggested methods of ensuring internal validity are data source and method triangulation [37]. Externally

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003

validity refers to the transferability of the research results to other similar contexts. If research results are transferable then they are arguably much more useful than results that are specific to a particular context.

The internal validity of this research has been sought through the use of data source and research method triangulation. There were two data sources used to construct the process model, SOAdapt-FoR was constructed with data from relevant literature sources, while the empirically grounded version of the model was constructed using data gathered during field work. There were also two research methods employed, survey research where individuals were interviewed and case study research where a the development process of SbaSoft was documented through interviews with several practitioners working at SbaSoft.

8.2. Conclusions and future work

During this research, the authors conducted a review of the literature and identified adaptation activities that could be used to adapt SBAs. These activities in combination with a skeleton life-cycle model proposed by the S-Cube consortium [18] formed the basis for a frame of reference process model for adapting SBAs. This frame of reference was used to guide interviews with SBA development practitioners who had experience with or who could provide expert opinion on how to adapt SBAs. The data that was collected in these interviews was transcribed and analyzed using qualitative content analysis techniques. The resulting adaptation activities and tasks were constructed into a detailed process model identifying the relevant stakeholders and development artifacts for each stage of the process. The model's transferability and capability were demonstrated during an evaluation process where the model was systematically compared to a component-based application adaptation model and an empirically based SBA development life-cycle. The approach taken in this paper has advantages over similar approaches in that its process focused and is based on input provided by experts from the field.

Future work will focus in refining the process model and investigating whether or not the activities identified for the adaptation cycle can be enriched with more detail. In order to determine the applicability of the model in the field, efforts will be made to use the model during an adaptable SBA development project.

Appendix A. Adaptation activities

See Table A.19.

References

[1] Rational, Rational unified process – best practices for software development teams, Tech. Rep. TP026B, 1998.
[2] M.P. Papazoglou, W.v.d. Heuvel, Service-oriented design and development methodology, Int. J. Web Eng. Technol. 2 (4) (2006) 412–442. URL <http://portal.acm.org/citation.cfm?id=1358575.1358582>.
[3] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, K. Holley, SOMA: a method for developing service-oriented solutions, IBM Syst. J. 47 (3) (2008) 377–396. URL <http://portal.acm.org/citation.cfm?id=1466610.1466613>.
[4] K. Mittal, Service oriented unified process, 2009, <http://www.kunalmittal.com/html/soup.html>. URL <http://www.kunalmittal.com/html/soup.html>.
[5] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, P. Traverso, Astro: supporting composition and execution of web services, in: Lecture Notes in Computer Science, vol. 3826, 2005, p. 495.
[6] D. Linner, H. Pfeffer, I. Radusch, S. Steglich, Biology as inspiration towards a novel service life-cycle, in: International Conference on Autonomic and Trusted Computing (ATC 2007), 2007, p. 94102.
[7] S. Lane, I. Richardson, Process models for service based applications: a systematic literature review, Information and Software Technology (2011).
[8] Y. Wautelet, Y. Achbany, J. Lange, M. Kolp, A process for developing adaptable and open service systems: application in supply chain management, in: International Conference on Enterprise Information Systems (ICEIS 2009), Lecture Notes in Business Information Processing, vol. 24, Springer, Milan, Italy, 2009, pp. 564–576.
[9] L. Oconner, B. Orriens, J. Yang, M. Papazoglou, A rule driven approach for developing adaptive service oriented business collaboration, in: International Conference on Service-Oriented Computing (ICSOC 2005), Lecture Notes in Computer Science, vol. 3826, 2005, pp. 182–189.
[10] S. Vale, S. Hammoudi, Model driven development of context-aware service oriented architecture, in: The 11th IEEE International Conference on Computational Science and Engineering – Workshops, 2008, pp. 412–418.
[11] T. Margaria, B. Steffen, M. Wirsing, M. Hoelzl, L. Acciai, F. Banti, A. Clark, A. Fantechi, S. Gilmore, S. Gnesi, L. Goenczy, N. Koch, A. Lapadula, P. Mayer, F. Mazzanti, R. Pugliese, A. Schroeder, F. Tiezzi, M. Tribastone, D. Varro, SENSORIA patterns: augmenting service engineering with formal analysis, transformation and dynamicity, in: Third International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, Communications in Computer and Information Science, vol. 17, 2008, pp. 170–190.
[12] G. Spanoudakis, A. Zisman, A. Kozlenkov, A service discovery framework for service centric systems, in: IEEE International Conference on Services Computing, 2005, vol. 1, 2005, pp. 251–259.
[13] K. Verma, K. Gomadam, A.P. Sheth, J.A. Miller, Z. Wu, The METEOR-S approach for configuring and executing dynamic web processes, LSDIS METEOR-S Project Technical Report, 2005.
[14] A. Bucchiarone, C. Cappiello, E. di Nitto, R. Kazhamiakin, V. Mazza, M. Pistore, Design for adaptation of service-based applications: main issues and requirements, Stockholm, Sweden, 2009.
[15] P. McKinley, S. Sadjadi, E. Kasten, B. Cheng, Composing adaptive software, Computer 37 (7) (2004) 56–64, doi:10.1109/MC.2004.48.
[16] S. Consortium, S-Cube knowledge model, 2011, <http://www.s-cube-network.eu/km>. URL <http://www.s-cube-network.eu/km>.
[17] J.C. Derniame, B.A. Kaba, D. Wastell, Software Process: Principles, Methodology, and Technology, Springer Verlag, 1999.
[18] S. Consortium, State of the art report on software engineering design knowledge and survey of HCI and contextual knowledge, Tech. Rep. PO-JRA-1.1.1, July 2008.
[19] P. Herzum, O. Sims, Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, John Wiley & Sons, Inc., 2000.
[20] R. Haesen, M. Snoeck, W. Lemahieu, S. Poelmans, On the definition of service granularity and its architectural impact, in: Z. Bellahsne, M. Lonard (Eds.), Advanced Information Systems Engineering, vol. 5074, Springer, Berlin Heidelberg, 2008, pp. 375–389. URL <http://www.springerlink.com/content/g17h73u685227g63/>.
[21] P. Oreizy, M. Gorlick, R. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, A. Wolf, An architecture-based approach to self-adaptive software, IEEE Intell. Syst. Appl. 14 (3) (1999) 54–62, doi:10.1109/5254.769885.
[22] F. Ahlemann, H. Gastl, Process model for an empiracally grounded reference model construction, in: P. Fettke, P. Loos (Eds.), Reference Modeling for Business Systems Analysis, Idea Group Publishing, 2007.
[23] S. Lane, Q. Gu, P. Lago, I. Richardson, Adaptation of service based applications: a maintenance process?, Tech. Rep. Lero-TR-2010-08, Lero, the Irish Software Engineering Research Centre, Limerick, Ireland, 2010. URL <http://www.lero.ie/sites/default/files/files/Lero-TR-2010-08.pdf>.
[24] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, V. Mazza, A context-driven adaptation process for service-based applications, in: Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems, 2010, pp. 50–56.
[25] IEEE, IEC, ISO, ISO/IEC 15288, systems and software engineering – system life cycle processes, Tech. Rep., 2008.
[26] C. Robson, Real World Research: A Resource for Social Scientists and Practitioner-Researchers, Wiley-Blackwell, 2002.
[27] R.K. Yin, Case Study Research: Design and Methods, Applied Social Research Methods Series, third ed., vol. 5, Sage Publications, Inc., 2002.
[28] M.B. Miles, A.M. Huberman, Qualitative Data Analysis: An Expanded Sourcebook, SAGE publications, Inc, 1994.
[29] I. Griffin, I. Richardson, Using LATEX for qualitative data analysis, The PracTEX Journal, 2010-1.
[30] P. Oreizy, N. Medvidovic, R. Taylor, Architecture-based runtime software evolution, in: Proceedings of the 1998 International Conference on Software Engineering, 1998, pp. 177–186. doi:10.1109/ICSE.1998.671114.
[31] S. Durvasula, M. Guttmann, A. Kumar, J. Lamb, T. Mitchel, B. Oral, Y. Pai, T. Sedlack, H. Sharma, S. Sundaresan, Introduction to Service Lifecycle, SOA Practitioners Guide. Part 3, 2007.
[32] Business process modeling notation (BPMN) information, 2010, <http://www.bpmn.org/Documents/FAQ.htm>. URL <http://www.bpmn.org/Documents/FAQ.htm>.
[33] P. Harmon, Second generation business process methodologies, Business Process Trends 1 (5) (2003).
[34] W.W. Royce, Managing the development of large software systems, in: Proceedings of IEEE Wescon, vol. 26, 1970, p. 9.
[35] B.G. Glaser, Basics of Grounded Theory Analysis, Sociology Press Mill Valley, CA, 1992.
[36] Y.S. Lincoln, E.G. Guba, Naturalistic Inquiry, Sage Publications, Inc., 1985.
[37] P. Liamputtong, D. Ezzy, Qualitative Research Methods, second ed., Oxford University Press, USA, 2005.

Please cite this article in press as: S. Lane et al., SOAdapt: A process reference model for developing adaptable service-based applications, Inform. Softw. Technol. (2011), doi:10.1016/j.infsof.2011.10.003