ADAPTATION OF SERVICE-BASED APPLICATIONS: A MAINTENANCE PROCESS?

Stephen Lane
Lero – The Irish Software Engineering Research Centre,
University of Limerick, Ireland

Qing Gu
Dept. of Computer Science,
VU University Amsterdam, The Netherlands

Patricia Lago
Dept. of Computer Science,
VU University Amsterdam, The Netherlands

Ita Richardson
Lero – The Irish Software Engineering Research Centre,
University of Limerick, Ireland

10 Jan 2011

Contact

Address ..... Lero
International Science Centre
University of Limerick
Ireland
Phone ..... +353 61 233799
Fax ..... +353 61 213036
E-Mail ..... info@lero.ie
Website ..... http://www.lero.ie/

This work is partially supported by Science Foundation Ireland
under grant no. 03/CE2/I303-1

Lero Technical Report Lero-TR-2010-08

Adaptation of Service-Based Applications: A Maintenance Process?

Stephen Lane, Qing Gu, Patricia Lago, and Ita Richardson,

***Abstract*—In this work, we identified activities relevant to the adaptation of Service Based Applications (SBAs) from existing service engineering approaches as well as activities relevant to the software maintenance process. We then mapped these two sets of activities to a reference life-cycle model from a Large European project. The results highlight the software maintenance techniques that can be reused or tailored for SBA adaptation, and point out the gaps that demand further research. The findings of this research may provide input for improving existing service engineering approaches to fulfill the need of adaptation.**

***Index Terms*—Service-Based Application Life-Cycle, Service Adaptation, Maintenance Process.**

I. INTRODUCTION

Service Based Applications (SBAs) are software applications which are composed of software services, those services may be owned by the application developers or by a third party. When services are provided by a third party often there is 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. Due to this uncertainty, the ability of SBAs to adapt in order to chose more suitable services is a desirable attribute. 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 applications development life-cycle. The focus of this paper are the software process challenges, which we will attempt to address be eliciting adaptation related activities from existing service literature as well as maintenance process literature. 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 process model that addresses the remaining areas of the SBA development life-cycle. The life-cycle model that we have chosen to use is the S-Cube [1] life-cycle model. S-Cube is a large European research project that conducts research in the area of Software Services and Systems. The S-Cube life-cycle was chosen because it specifically aims to facilitate the adaptation of 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 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 adapatation needs*, *Identify adapatation strategy* and *Enact adaptation*. Within the complete life-cycle, there must also be a focus on o*operation and management*, and *deployment and provisioning*.

The S-Cube life-cycle as presented here is a work in progress, 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 [2][3][4][5]. The aim of this paper is to develop a set of activities for the processes of the adpatation cycle of the life-cycle. This adaptation cycle is the major difference between this life-cycle and standard software engineering life-cycles such as waterfall [6], or spiral [7] life-cycle models.

It may be desirable for an SBA to adapt for many reasons, such as business agility or failure recovery, in either of these cases it may be desirable to replace services within an SBA through self-adaptation or through manual adaptation. Adaptation of SBAs is different from maintenance in traditional software engineering in that 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 both adaptation and maintenance at a basic level 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 the software maintenance process. The maintenance process was chosen as a source of activities as it bears resemblance to the SBA adaptation process. We also elicit adaptation activities from existing service-based development approaches. By taking this approach existing activities are reused in a novel way that can fulfill the adaptation cycle of the S-Cube life-cycle. The use of the activities from the

Stephen Lane and Ita Richardson are with Lero - The Irish Software Engineering Institute, University *of* Limerick, Ireland.
E-mail: stephen.lane@lero.ie

Qing Gu and Patricia Lago are with Dept. of Computer Science, VU University Amsterdam, The Netherlands.

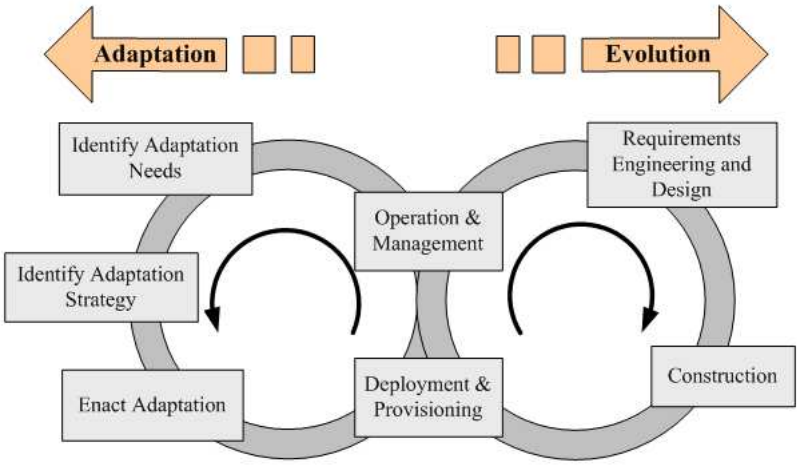

Fig. 1. The Life-Cycle of Adaptable SBAs.

maintenance process ensure that a level of quality assurance is built into the life-cycle.

This paper is organised as follows, Section II describes the motivation for carrying out this work, followed by Section III which describes our research methodology. Section III provides some background information on SBA adaptation and service engineering process models. The remainder of the paper contains the body of the work in three phases, followed by conclusions.

II. 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 process that accommodates their adaptation requirements.

Implementing a best practice software process ensures quality through the optimisation of the engineering processes and methods during the development life-cycle. This is particularly important when developing software within a critical domain. In their Evolving Critical Systems White paper [8], 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 [9]. One of the aims of this group is to achieve standardization, and 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 [10] 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.

The financial industry has been regulated by Sorbonnes-Oxley, mainly because it operates where there are business-critical and security-critical environments.

We also need to consider feature-interaction and how inclusion or exclusion of features could cause critical systems to fail. For example, traffic management systems are often discussed as cases where adaptation can occur [11]. But, the question is, what happens if the toll system swapped into the car also affects the braking system? Should these adaptable features not conform to regulations? We anticipate that the requirement for software engineering quality processes to be used will grow as these critical markets, such as medical device, automotive and financial domains, grow.

Given this 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 cycle practices to be defined.

In this paper we identify software engineering activities which should be carried out within the adaptation cycle of the S-Cube life-cycle, focusing both on service-oriented developemnt approaches as well as on traditional software engineering processes.

III. RESEARCH METHOD

Despite that adapting SBAs is of great importance in service engineering, SBA adaptation remains a challenge due to its

dynamic and unpredictable nature. With the aim of gaining insights into the state of the art of SBA adaptation, we see the need of reviewing the adaptation aspect of the current service engineering approaches. Moreover, given the similarities between software maintenance process and service adaptation, we are inspired to investigate if we could borrow some lessons from software maintenance processes which are mature and have been practiced for many years. To this end, we carry out this work through three distinct phases, which are illustrated in Figure 2.

In Phase I, we identified a set of concrete adaptation activities in the S-Cube life-cycle. While the S-Cube life-cycle shows the adaptation cycle and breaks it into three constituent processes - Identify adaptation needs, Identify adaptation strategy, Enact adaptation, these three processes are defined at a high level. In order to gain a better understanding on these adaptation processes, we analyzed S-Cube deliverables (CD-JRA-1.1.2, CD-JRA-1.2.1, CD-JRA-1.2.2 [12]) which provided us with more detailed description of the adaptation activities required for these processes. With this input, we were able to refine the S-Cube life-cycle with a set of concrete adaptation activities.

In Phase II, we carried out a literature review and identified a set of adaptation activities from service-oriented engineering approaches and maintenance activities from software maintenance process models. As illustrated in Figure 2, this phase has been carried out in two steps. In step 1, we studied a number of service-oriented engineering approaches, from which we identified several approaches that are concerned with adaptation. From each of these approaches, we elicited activities that are related to adaptation. In step 2, we studied a number of software maintenance process models, focusing particularly on ISO/IEC 14764, from which we elicited activities that potentially could also be used for adaptation.

Having the adaptation and maintenance activities from the literature (from Phase II), in Phase III we mapped them to the refined S-Cube life-cycle (from Phase I). By observing the mapping of the adaptation activities and the refined S-Cube life-cycle, we gained an overview of the relation between the existing adaptation approaches and identified research challenges with regard to service adaptation.

As adaptation is the modification of software in a dynamic environment, by observing the mapping of the maintenance activities to the refined S-Cube life-cycle, we expected that activities from static modification (maintenance) would or could also be of importance during SBA adaptation. In this way, we were able to highlight the software maintenance techniques that can be reused or tailored for service adaptation and hence improving existing service-oriented engineering approaches to fulfill the need of adaptation.

IV. BACKGROUND

A. SBA Adaptation

Within the context of SBAs, adaptation is the modification of an application in order to satisfy adaptation requirements [13]. There are many adaptation requirements that can be desirable in SBAs, for example, the facilitation of interoperability amongst services [14], the optimisation of Quality of Service (QoS) [15] or the implementation of failure recovery [16]. SBA adaptation may involves the substitution, replacement, reconfiguration 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 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) [17] 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 a change to the initial SBA implementation while dynamic adaptation occurs at run-time as a behavioural change. With dynamic adaptation the adaptation strategy has to be decided in advance because it has to be built into a SBA in terms of application logic. Dynamic adaptation of an SBA can by partially automated 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. Conversely with static adaptation the adaptation strategy does not have to be decided when the SBA is first implemented as long as it is decided before the adaptation is enacted. In this way the static adaptation of SBAs is comparable to the maintenance process of traditional software systems.

1) Processes for Adapting SBAs: The three adaptation processes required for the adaptation of SBAs as identified in [13] are: the identification of adaptation requirements, the development of an adaptation strategy and the enactment of the adaptation. These processes were identified during the classification of various adaptation concepts during a review of service adaptation literature in the S-Cube project [12].

In a fully or partially automated dynamic adaptation scenario the processes involved in the adaptation occur simultaneously with the development of a SBA. The adaptation requirements would need to be developed at the same time as the applications requirements and the adaptation strategy would have to be determined when the system is being designed. Then the final process, enactment of the adaptation, would occur automatically or semi-automatically during runtime. A comprehensive set of practices required to implement these processes have yet to be defined in the literature [18].

When there is no suitable mechanism in an application to enable dynamic adaptation, static adaptation is necessary to satisfy adaptation requirements. Static adaptation requires the same processes as dynamic adaptation but the processes

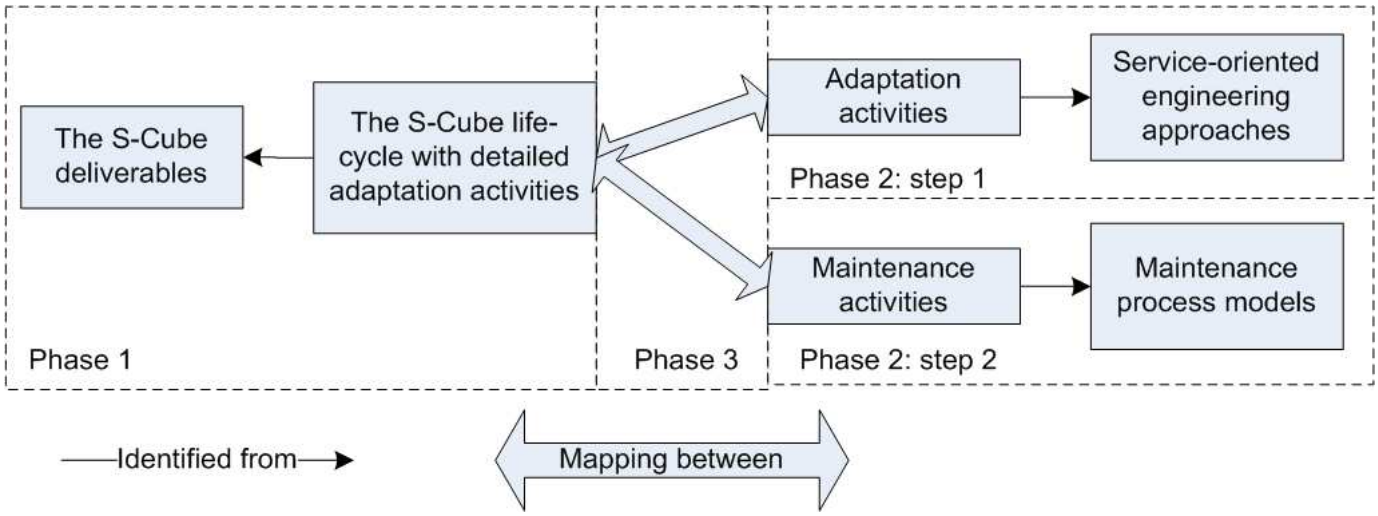

Fig. 2. An illustration of our research method

required for static adaptation do not need to be carried out during initial implementation. As previously mentioned this makes static adaptation comparable to the maintenance process of traditional software applications. Static adaptation differs from the maintenance process in that the adaptation of loosely coupled SBAs requires much less time and effort than the maintenance of a software system that was not conceived with ease of adaptation as an architectural feature.

2) Gap in Software Engineering Processes: 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 as compared to traditional software applications. 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 form 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 *Perfective 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.

B. Software Maintenance Definitions

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]:

- *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.
- *Perfective Maintenance* is performed to improve performance or maintainability.
- *Emergency Maintenance*: is unplanned maintenance that is carried out in order to keep a system operational.
- *Preventive Maintenance*: is maintenance carried out in a system to detect future errors in a software product.

V. PHASE I: REFINING S-CUBE LIFE-CYCLE

In this Phase we elicit adaptation related activities from the published S-Cube deliverables. S-Cube's existing deliverables are a rich source of information for service engineering principals and practices since Engineering and Adaptation Methodologies is one of the primary research tracks of the project. The relevant deliverables were inspected for activities relating to the adaptation processes of the s-Cube life-cycle namely *Identify adaptation need*, *Identify adaptation strategy*, and *Enact adaptation*. The *Identify adaptation need* process has two objectives. One of the objectives is to identify adaptation requirements that are either raised by the humans involved in the execution of SBAs or generated by the technological environment in which the system is running. Another objective is to decide if and when to take these requirement into consideration in that some requirements might conflict with each other. With this objective, relevant information of the behavior of the system has to be collected and evaluated.

Hence, *Monitoring* is also required in this processes. As soon as the need for adaptation is identified, the methods and strategies for adaptation should be decided in the *Identify adaptation strategy* process. The actual adaptation execution takes place in the *Enact adaptation* process.

We identified nine adaptation activities from the relevant S-Cube deliverables, these activities identified within the deliverables are necessary in order to develop adaptable SBAs. The three deliverables examined were:

- CD-JRA-1.1.2 Separate Design Knowledge Models for Software Engineering and Service Based Computing [22]
- 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 [23]
- CD-JRA-1.2.2 Taxonomy of Adaptation Principles and Mechanisms [24]

The first deliverable CD-JRA-1.1.2 is a knowledge model which contains information software engineering principals that are relevant to the area of service-oriented computing. Within the deliverable the concept of adaptable SBAs is introduced as well as some of the activities required to achieve this adaptation. The second deliverable PO-JRA-1.2.1 presents the state of the art of engineering principals for adaptable SBAs, this deliverable produced many adaptation related activities from the state of the art approaches encountered. The final deliverable studied, CD-JRA-1.2.2, produced a taxonomy of adaptation principals and mechanisms. This taxonomy is useful because it shows the interrelations between the activities encountered during the other deliverables.

The adaptation activities that were identified in the deliverables were mapped to the appropriate adaptation processes in the adaptation cycle of the S-Cube life-cycle. The practices are listed in Table I under headings that correspond to their associated adaptation processes. Figure 3 graphically illustrates where these practices occur within the S-Cube life-cycle[1]. They are arranged into logical sequential steps within their respective processes. The activities identified form a complete set of what needs to be achieved in order to complete the adaptation processes. The availability of a complete set of adaptation activities is based on the co-ordinated work of the S-Cube deliverables studied, which focus on identifying the principals and techniques required for developing adaptable SBAs.

VI. PHASE II: IDENTIFYING ADAPTATION ACTIVITIES

A. Adaptation Activities from Service-Oriented Engineering Approaches

The first Step (1) of this Phase (II) was to identify sub-activities that can be used to complement the adaptation activities identified in Section V, these activities can then ultimately be used to execute the adaption processes of the S-Cube life-cycle. There have been many software development processes and life-cycles proposed for the development of SBAs [18] as well as their underlying services. Lane and Richardson [18] highlight through a systematic literature review that many of these proposed approaches do not take the adaptation of SBAs into consideration. Several approaches such as those proposed by Cortellessa et al [25] or Adil kenzi et al [26] include adaptation as primary concern when developing services. However, these approaches are aimed at the development of services rather than compositions of services required by SBAs. 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.

1) Service-Oriented Engineering Approaches: In this section we will discuss the service-oriented development approaches that do have activities that can be used for the adaptation of SBAs.

ASTRO [27] is a toolset that is made up from four component tools: WS-gen, WS-mon, WS-console and WS-animator. The aim of this project 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 activitiess.

The BEA reference lifecycle [28] outlines the activities for each of the following SBA life-cylce 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 lifecycle also has a business dashboard which monitors the lifecycle as it progresses. Along with the dashboards the lifecycle had a governance process which promotes interoperability, discoverability and standardisitation of service technologies. The BEA lifecycle caters for adaptability to some extinct as it provices service monitoring, runtime correctness analysis and operational management activities.

Chang [29] proposes a process model which focuses on developing highly adaptable web services. It follows the sequence of steps specified in the SOAD [30] 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

[1]The shadowed boxes denote S-Cube life cycle processes and white boxes denote the activities that may occur in the adaptation-related processes. The arrow lines between the practices represent the dependencies between them

TABLE I
ADAPTATION ACTIVITIES FROM S-CUBE DELIVERABLES

Identify adaptation need	
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.
Identify adaptation strategy	
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.
Enact adaptation	
Perform adaptation	The actual adaptation process is performed through adaptation realization mechanisms based on the selected adaption strategy.

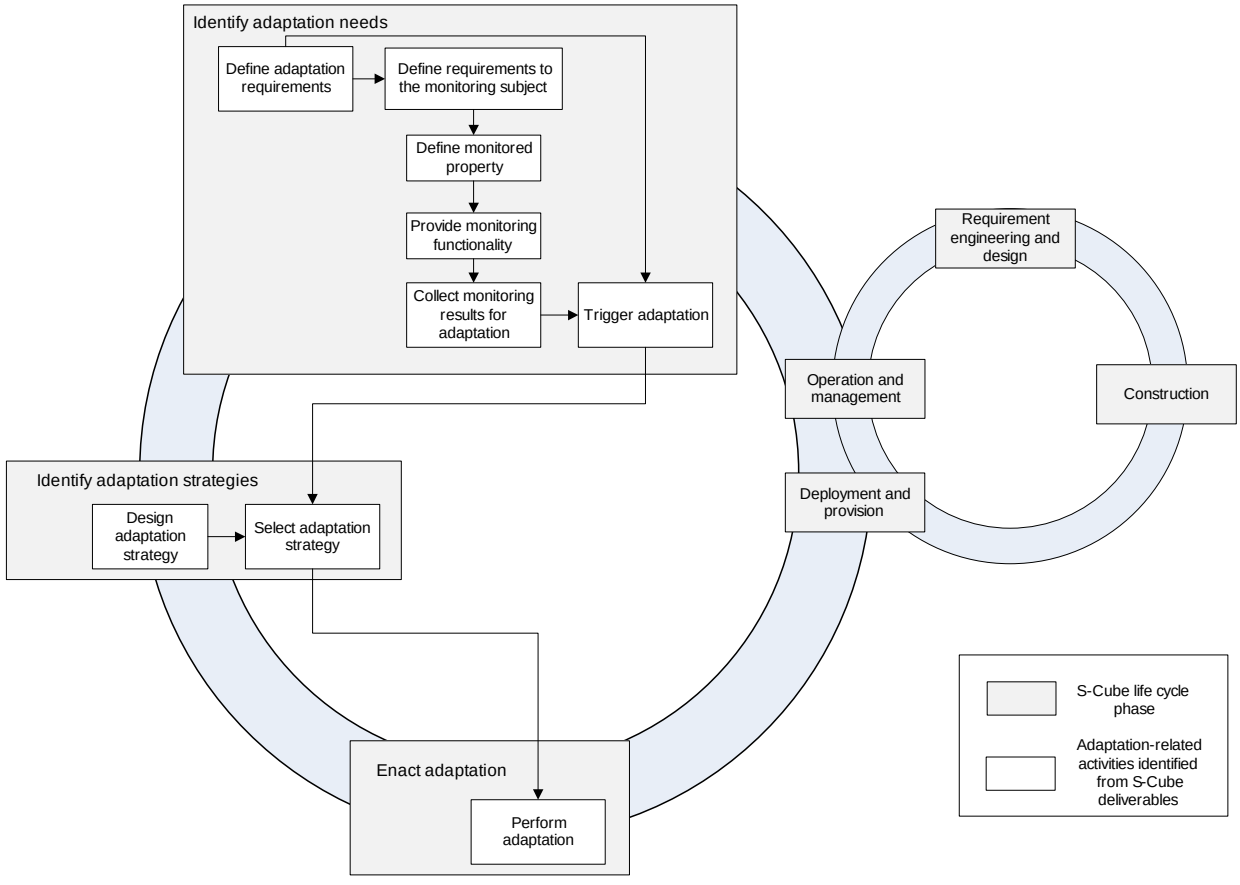

Fig. 3. S-Cube Life-Cycle with Adaptation Activities

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) [31] is influenced by several established life-cycles such as RUP [32], CBD [33] and BPM [34]. 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 quality of service monitoring and alerts for compliance failures.

The SeCSE methodology [35] 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) Activities Identified: Having reviewed these five approached in detail any activities encountered relating to adaptation or monitoring were recorded. The monitoring activities were recorded because adaptation cannot take place without the monitoring, so in a way monitoring can be seen as a sub-process of adaptation. The activities that were recorded are summerised in Table II.

Here we will give a brief description of each of the adaptation activities identified in Table II:

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 first 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. The second activity (*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.

Changs approach contains two adaptation related activities: *Specifying service decision model* aims at specifying the variability between available services and expected services. The *Designing Service Adapters* aims at bridging the variability between service providers and consumers by allowing services to be dynamically adapted.

TABLE II
ADAPTATION ACTIVITIES FROM SERVICE-ORIENTED ENGINEERING APPROACHES

1 Astro
Monitor message sequences amongst services and its partners Detect protocol violations
2 BEA
Requirements and analysis stage: define KPIs and management policies Monitor service, application, middleware, OS, hardware, and network
3 Chang's
Specifying Service Decision Model Designing Service Adapters
4 SDLC
Gather QoS metrics on the basis of SLAs Set warning thresholds and alerts for compliance failures Monitor workloads Evaluate SLA QoS metrics Readjust service weights for request queues
SeCSE
Service specification: identify the service properties to specify Specify monitoring rules according to the adopted SeCSE monitoring language (SECMOL) Service deployment: insertion of monitoring rules and recovery actions in concrete parts of the service composition executable description Service deployment: deploy the monitoring rules and recovery policies within the monitoring system Monitor services Recovery management: identify, by looking at the monitoring data, the needs for a recovery action Runtime Service Discovery

SDLC defines five adaptation related activities which revolve around the monitoring of quality attributes and alerting system users they exceed predefined SLAs: *Gather QoS (Quality of Service) metrics on the basis of SLAs (Service Level Agreements)* refers to the collect quality data to be monitored, *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 responce 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.

B. Adaptation Activities from Maintenance Process Models

The second step (2) of this phase (II) we aimed to identify activities from the software maintenance process that may be useful for the adaptation of SBAs. There were many software maintenance processes, definitions, models and standards encountered during the literature review carried out in Section VI-B1, however, ISO/IEC 14764 was the only source that contained detailed activities. For this reason ISO/IEC 14764 was chosen as the sole source of adaptation activities for this Step (2). In the next Section (VI-B1) a review will be given of the maintenance process models we encountered leading us to ISO/IEC 14764 as a source of adaptation activities.

1) Software Maintenance Process Models: There are many software maintenance models proposed in the literature. Models from Martin and McClure (1983) [36] or Parikh (1982) [37] are simple and loosely defined, while others such as Sharpley (1977) [38] and Yau (1980) [39] provide more detailed approaches. The age of maintenance models also differs greatly, with Bohem's [40] model dating back to 1976 when software engineering was in its infancy, right up to the present day standards such as ISO/IEC 14764.

The earliest models that can be found in the literature are generally less complex or detailed than the more recent literature. Bohem [40] 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 [41] in 1998 which was an elaboration of the maintenance process from the IEEE 12207 [42] 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 processes of the model.

a) ISO Maintenance Process Models: ISO/IEC-15504 primarily known as SPICE (Software Process Improvement and Capability Determination) 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 the outcome for the software maintenance process. The reference lifecycle from ISO/IEC 15504 has descriptions for each process in the software engineering life-cycle, therefore they need to be relatively concise otherwise completing a capability assessment may become too labour 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 much 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 carrying out 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 need to document and have approved 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. Then the required software changes need to 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 notification of retirement intent should be sent to the appropriate software product stakeholders. During retirement parallel operation of new and retiring software software should be carried out along with training of end users. Once complete notification should be sent to the appropriate stakeholders and finally data relating to the retiring product should be achieved should it be required at a later date.

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

c) Maintenance Process in Reference Models: Many software engineering reference lifecycles and assessment models do not make direct reference to software maintenance. It seems like 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 [43] and 70 percent of the budget [44] of a software project. April et al [45] propose a Software Maintenance Maturity Model (SMMM) which can be used as an add-on to the CMMI™, it takes best practice processes and activities from a variety of sources such as ISO/IEC 14764, IEEE 1219, ISO/IEC 12207, CMMI™ and SWEBOK [46] in order to construct the model.

2) Activities Identified: In total there were 19 maintenance activities identified from the software engineering literature (ISO/IEC 14764). They are categorised in Table III according to the 5 processes they come from in ISO/IEC 14764. These 5 processes represent the complete set of activities that need to be carried out to implement a maintenance process according to ISO/IEC 14764. 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. The individual activities will be discussed in detail with respect to their applicability to SBA adaptation in Phase III.

TABLE III
MAINTENANCE ACTIVITIES FROM ISO/IEC 14764

Maintenance Practices
1 Process Implementation
Maintenance plans and procedures
MR/PR procedures
Configuration management
2 Problem and Modification Analysis
MR/PR analysis
Verification
Options
Documentation
Approval
3 Modification Implementation
Analysis
Development process
4 Maintenance Review/Acceptance
Reviews
Approval
5 Migration
Migration
Migration plan
Notification of intent
Implement operations and training
Notification of completion
Post-operation review
Data archival

VII. PHASE III: MAPPING ADAPTATION ACTIVITIES

In this Phase (III) we will map the adaptation activities identified in the previous Phase (II) to the refined S-Cube life-cycle from Phase (I). The adaptation activities in the refined S-Cube life-cycle are high level activities so the more granular activities identified in Phase (II) will be mapped as sub-activities of those high level activities. The mapping process will be carried out in two steps, first the activities from the service-oriented engineering approaches will be mapped to the refined S-Cube life-cycle, then activities from ISO/IEC 14764 will be mapped to the life-cycle.

A. Mapping Service Engineering Activities to S-Cube Life-Cycle.

Figure 4 shows the adaptation activities identified from the service-oriented engineering approaches studied mapped to the high level adaptation activities of the refined S-Cube life-cycle. For instance, *Detect protocol violations* from ASTRO aims at catching the misbehaviors by external partners according to the business process protocol; *Evaluate SLA QoS metrics* aims at assessing quality attributes of the system of interest based on the corresponding SLAs. Both of these two activities assess the execution of the system against pre-defined requirements

by collecting and analyzing monitoring results. Therefore, we mapped these two activities to *collect monitoring results for adaptation* within the refined S-Cube life-cycle.

The figure shows a three layer hierarchy with the three S-Cube adaptation processes: *Identify adaptation needs*, *Identify adaptation strategies*, and *Enact adaptation* at the highest level. The next level down in the hierarchy are represented as boxes within the three primary adaptation processes. These are the high-level adaptation activities that were identified from the several relevant S-Cube deliverables. These activities, which make up the lowest level in the hierarchy, are used to classify the adaptation activities elicited from the service-oriented engineering approaches in Phase II.

B. Observations on Mapping of Service Engineering Activities to the S-Cube Life-Cycle

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 Figure 3, many activities were in the process: *Identify adaptation needs*, while only a few processes were identified in the other two processes. This implies that the state of art of adaptation processes focuses much more on gathering requirements and identifying when adaption is needed. These are tightly 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 limited to corrective adaptation - the replacement of services when quality attributes do not meet the 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. They are not directly used by adaptation practices but are relevant in that specifying these attributes makes corresponding monitoring and assessment possible.

C. Mapping Activities from Maintenance Process Models

Out of the 19 maintenance activities identified in Phase II (see Table III, 13 of them were mapped to adaptation activities identified in Phase I of this work (see Figure 5). Many of these 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”. Here we will explain the mappings made in Figure 5.

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 when 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 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, it was mapped to this activity because the resolution of problems after applications adapt would be much easier if configuration details of component services are recorded. Fang *et al* [47] 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

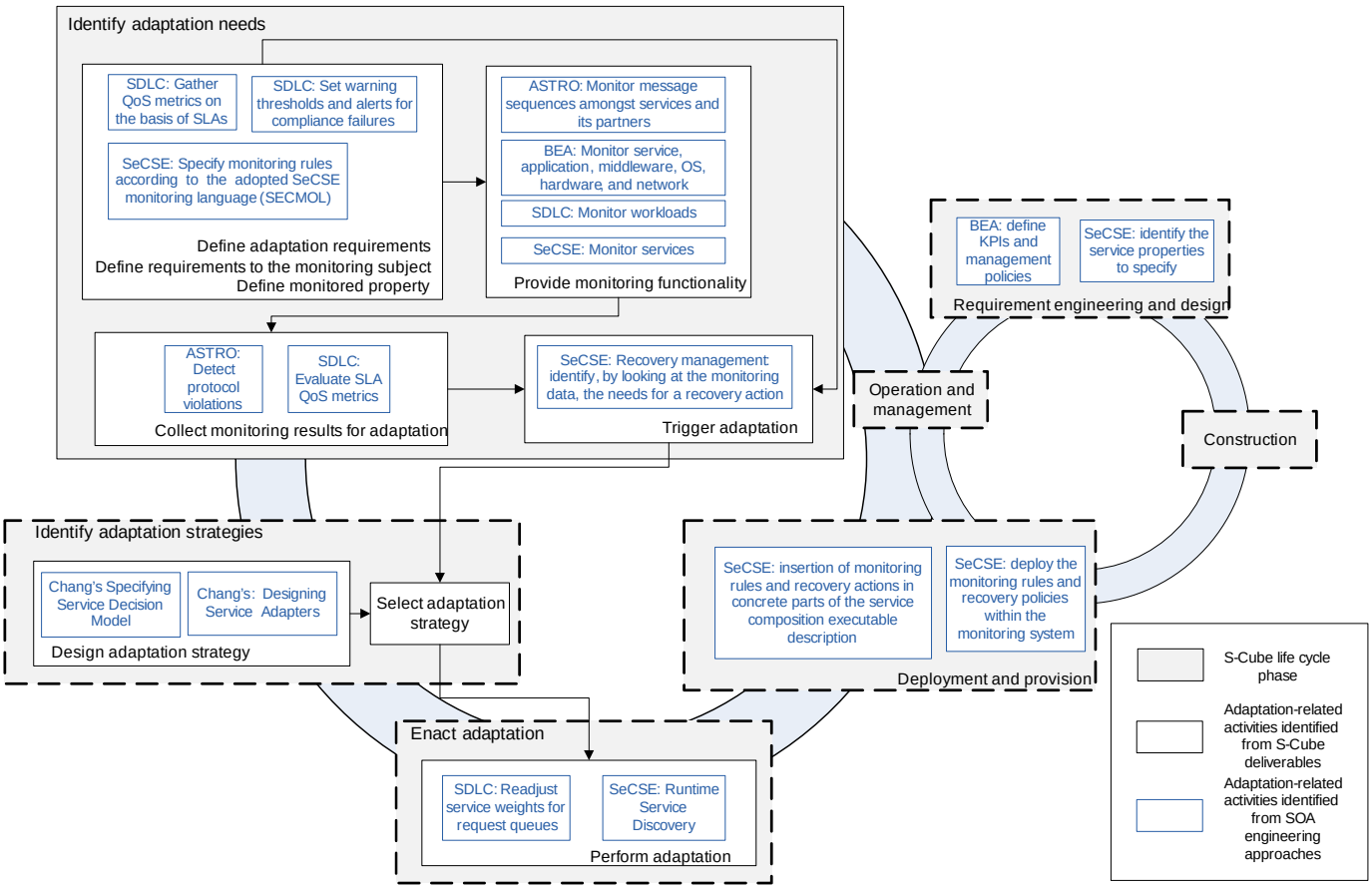

Fig. 4. Mapping Service-Oriented Engineering Activities to S-Cube Life-Cycle

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 in order 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.

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 several of the activities from ISO/IEC 14764 could not be mapped to adaptation activities because

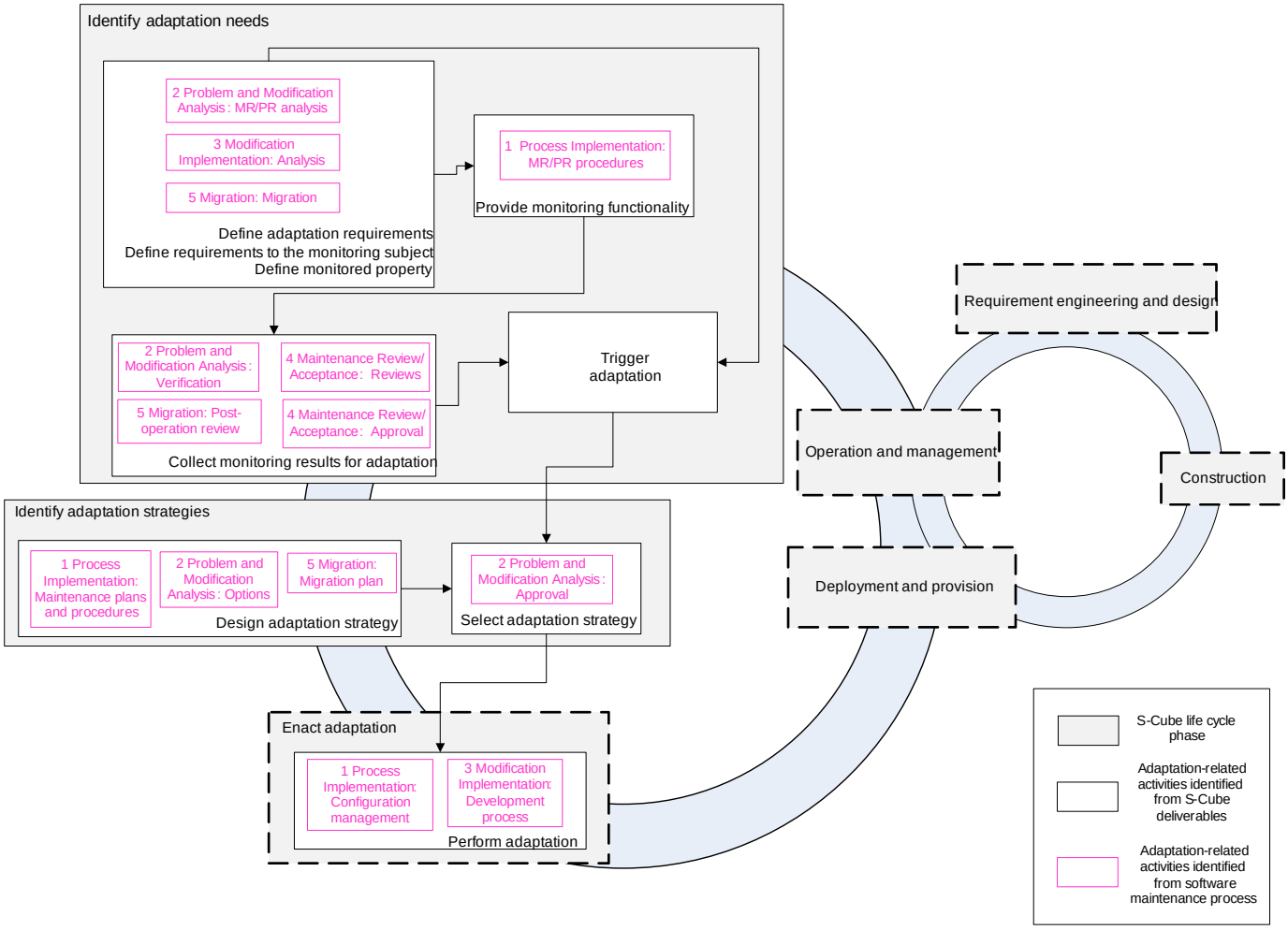

Fig. 5. Mapping Maintenance Activities to S-Cube Life-Cycle.

TABLE IV
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

they are too specific to the software maintenance process (see Table IV). There were 4 activities excluded from the mapping: the *Documentation activity* and 4 *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 4 migration activities mentioned in Table IV are specific to the the maintenance of traditional software and cannot be leveraged for service adaptation.

D. Observations on mapping Activities from Maintenance Process Models to S-Cube Life-Cycle

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 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 form 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 the agile nature of SBA adaptation.

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.

VIII. CONCLUSION

In this paper, we defined activities which should be considered when carrying out the adaptation processes of: *Identify adaptation needs*, *Identify adaptation strategies* and *Enact adaptation*.

This has been done through the identification of activities within S-Cube project deliverables, service-oriented development models, and the software engineering maintenance process. The importance of this work is that while it consolidates existing work for service-oriented development into a SBA development life-cycle, it enhances this with activities from software engineering.

During Phase II We noted that there are potential weaknesses if we examine service-oriented development approaches only. Our analysis has highlighted some of the weakness of the service approaches. However, we identified that the inclusion of software maintenance activities can remove some of these weaknesses. We identified a set of service adaptation activities through analyzing both the service-oriented development approaches and software maintenance activities. Issues which are being dealt with through the implementation of software engineering activities into the S-Cube life cycle include defining, selecting and executing adaptation strategies.

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 the maintenance standard ISO/IEC 14764. However, we have not considered whether emerging methods, such as agile methods, can be used in a similar format. There is potential for some research to be carried out on this topic.

In addition, while we have identified the activities which should be used during the adaptation cycle, we have not discussed how each of these activities should be implemented. We see this as the next stage of this research project.

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 Grant No. 03/CE2/I303.1

REFERENCES

[1] CITY, FBK, Lero-UL, POLIMI, and Tilburg, "Separate design knowledge models for software engineering and service based computing," Tech. Rep., 2009.

[2] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "A unified formal model for controlled evolution of services*," in *(submitted to ICSOC09)*, 2009.

[3] A. Bucchiarone, C. Cappiello, E. di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore, "Design for adaptation of Service-Based applications: Main issues and requirements," in *(to be submitted)*, 2009.

[4] Q. Gu, P. Lago, and E. D. Nitto, "Guiding the service engineering process: the importance of service aspects," in *2nd IFIP WG5.8 Workshop on Enterprise Interoperability (IWEI 2009)*. Valencia, Spain: Springer, 2009.

[5] N. Maiden, K. Zachos, A. Metzger, A. Gelhert, K. Karlsen, and N. Seyff, "Using scenarios to discover requirements for Service-Based applications," in *(to be submitted to REFSQ10)*, 2009.

[6] W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE Wescon*, vol. 26, no. 1, 1970, p. 9.

[7] B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.

[8] Lero, "Evolving critical systems white paper," 2009.

[9] "Herstellerinitiative software, working group assessment: HIS." [Online]. Available: http://portal.automotive-his.de/images/pdf/ProcessAssessment/his-wg-assessments

[10] "Amazon." [Online]. Available: http://www.amazon.com/

[11] 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, p. 5462, 1999.

[12] "S-cube deliverables." [Online]. Available: http://www.s-cube-network.eu/results/deliverables

[13] 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, Jul. 2008. [Online]. Available: http://www.s-cube-network.eu/results/deliverables/wp-jra-1.2/PO-JRA-1.2.1-State

[14] 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.

[15] E. D. 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.

[16] 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.

[17] 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.

[18] S. Lane and I. Richardson, "Process models for service based applications: A systematic literature review," *to be submitted: Journal of IST*, 2010.

[19] E. B. Swanson, "The dimensions of maintenance," in *Proceedings of the 2nd international conference on Software engineering*, 1976, p. 492497.

[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] V. Andrikopoulos, "Separate design knowledge models for software engineering and service based computing," S-Cube Consortium, Deliverable CD-JRA-1.1.2, May 2009, the following institutions contributed to this deliverable: City University London, Center for Scientific and Technological Research, Lero - The Irish Software Engineering Research Centre, Politecnico di Milano and Tilburg University.

[23] 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, the following institutions contributed to this deliverable: Université Claude Bernard Lyon, Politecnico di Milano, Center for Scientific and Technological Research, MTA SZTAKI Computer and Automation Research Institute, University of Duisburg-Essen, The French National Institute for Research in Computer Science and Control, Consiglio Nazionale delle Ricerche and University of Stuttgart.

[24] 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, the following institutions contributed to this deliverable: City University London, Center for Scientific and Technological Research,The French National Institute for Research in Computer Science and Control, Politecnico di Milano, MTA SZTAKI Computer and Automation Research Institute, Tilburg University, Université Claude Bernard Lyon and University of Duisburg-Essen.

[25] 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.

[26] 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.

[27] 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.

[28] S. Durvasula *et al.*, "Introduction to service lifecycle. SOA practitioners guide. part 3," 2007.

[29] 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.

[30] A. Arsanjani, "Service-oriented modeling and architecture," Nov. 2004. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/

[31] 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

[32] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Addison Wesley, Dec. 2003.

[33] 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.

[34] P. Harmon, "Second generation business process methodologies," *Business Process Trends*, vol. 1, no. 5, 2003.

[35] ATOS, "SeCSE methodology, version 3," Tech. Rep., Mar. 2007.

[36] J. Martin and C. L. McClure, *Software Maintenance: Problems and Its Solutions*. Prentice Hall PTR, Apr. 1983.

[37] G. Parikh, *Techniques of Program and Systems Maintenance*, 2nd ed. Q E D Pub Co, Jun. 1988.

[38] W. K. Sharpley, "Software maintenance planning for embedded computer systems," in *Proceedings of the IEEE COMPSAC*, 1977, p. 520526.

[39] S. Yau and J. Collofello, "Some stability measures for software maintenance," *Software Engineering, IEEE Transactions on*, vol. SE-6, no. 6, pp. 545–552, 1980.

[40] B. W. Boehm, "Software engineering," *IEEE Transactions on Computers*, vol. 100, no. 25, p. 12261241, 1976.

[41] IEEE, ISO, and IEC, *IEEE Standard for Software Maintenance*. New York NY: Institute of Electrical and Electronics Engineers, 1998.

[42] ——, *Systems and software engineering software life cycle processes*. Geneva: ISO/IEC-IEEE, 2008.

[43] S. D. Conte, H. E. Dunsmore, and Y. E. Shen, *Software engineering metrics and models*, 1986.

[44] M. M. Lehman, "Program evolution." *INFO. PROC. MANAGE.*, vol. 20, no. 1, p. 1936, 1984.

[45] 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, p. 197223, 2005.

[46] A. Abran, P. Bourque, R. Dupuis, and J. W. Moore, "Guide to the software engineering body of Knowledge-SWEBOK," 2001.

[47] 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.