SOCA (2014) 8:239–257
DOI 10.1007/s11761-013-0136-4

ORIGINAL RESEARCH PAPER

Towards a framework for the development of adaptable service-based applications

Stephen Lane · Qing Gu · Patricia Lago · Ita Richardson

Received: 19 April 2011 / Revised: 11 May 2013 / Accepted: 20 June 2013 / Published online: 14 July 2013

Abstract Service-oriented computing is a promising computing paradigm which facilitates the composition of loosely coupled and adaptable applications. Unfortunately, this new paradigm does not lend itself easily to traditional software engineering methods and principles due to the decentralised 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 service-based application development life-cycle. Existing software engineering literature as well as research results from service engineering research is 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

S. Lane (✉) · I. Richardson
Lero, the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
e-mail: stephen.lane@lero.ie

I. Richardson
e-mail: ita.richardson@lero.ie

Q. Gu · P. Lago
Department of Computer Science, VU University Amsterdam, Amsterdam, The Netherlands
e-mail: q.gu@vu.nl

P. Lago
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 adapting, it may be desirable to replace or reconfigure 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 relate to the implementation of adaptation mechanisms, while the software process challenges relate to the development methods required for adaptable SBAs. The focus of this paper is the software process challenges. There are several approaches that address the aspects of SBA development [1–3]. However, these approaches do not specifically facilitate adaptation.

We address the process challenges by eliciting adaptation-related activities from existing service literature and elicit adaptation support activities from the software maintenance 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 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 [4] reference life-cycle. S-Cube is

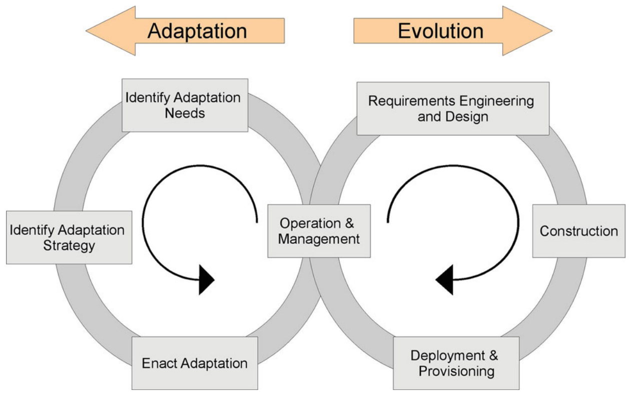
Adaptation
Evolution
Identify Adaptation Needs
Requirements Engineering and Design
Identify Adaptation Strategy
Operation & Management
Construction
Enact Adaptation
Deployment & Provisioning

Fig. 1 Life-cycle of adaptable SBAs

a European consortium that conducts research on software services and systems of which 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 Fig. 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 lifetime. The adaptation cycle, shown on the left-hand side, ensures that the software engineer follows the following 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*.

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 [5]. The aim of this paper is to develop the adaptation-related activities for the life-cycle. The 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.

Adaptation of SBAs is different from maintenance in traditional software engineering in that it is a less expensive 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 has 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 fulfil the adaptation cycle of the S-Cube life-cycle. The use of engineering practices from the maintenance process ensures that a level of quality assurance is built into the life-cycle.

In related work, Oreizy et al. [8] propose a development life-cycle for adaptable component-based applications with both development time and run-time 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.

Gu and Lago [9] have previously evaluated several service-oriented software engineering methodologies. They have evaluated them to determine whether they are truly service-oriented and which areas of the development life-cycle they apply. This evaluation illustrates methodologies that can provide input into this study.

This paper is organised as follows: Sect. 2 describes the motivation for carrying out this work, followed by Sect. 3 which provides some background information on SBA adaptation and service engineering process models. Section 4 describes our research methodology. The remainder of the paper contains the body of the work in Sects. 5 and 6. A case demonstrating an application of the framework is presented in Sect. 8. Section 7 presents a set of metrics that can be used to evaluate the process, followed by the conclusions in Sect. 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 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 promises to provide greater power and flexibility than predecessors such as electronic data interchange, distributed components and e-mail. 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 run-time adaptation capabilities. Existing development approaches usually do not have any processes to support run-time 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 [10], 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 Hersteller Initiative Software (HIS) process assessment working group [11]. One of the aims of this group is to achieve standardisation, 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 [12] which 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 Background

3.1 SBA adaptation definitions

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 involve 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 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. [17] propose a methodology for the monitoring of web service-based applications, so they can be adapted if an error occurs or 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 run-time. Modifying the adaptation logic for an 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 run-time.

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 into an SBA. In a fully automatic SBA, this substitution could be enacted automatically by the application based on the QoS or availability of alternative services.

3.2 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 % of the time [18] and 70 % of the budget [19] of a software project. April et al. [20] propose a Software Maintenance Maturity Model (SMMM) that 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 [21] 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 [22–24]:

– *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 response to new functionality requirements.
– *Preventive Maintenance* is maintenance carried out in a system to detect future errors in a software product.
– *Emergency Maintenance* is an unplanned maintenance that is carried out in order to keep a system operational.

3.2.1 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 becomes 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 that are missing from the software engineering literature; each of the processes are based on similar software maintenance processes:

– *Perfective Adaptation* aims at improving or optimising the quality attributes of an SBA even it runs correctly. This corresponds with *Perfective Maintenance*.
– *Corrective Adaptation* aims at removing any faults in the behaviour of an SBA. This corresponds with *Corrective Maintenance*.
– *Adaptive Adaptation* modifies an 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 an 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 an SBA can be seen as unplanned maintenance activities.

4 Research method

The aims of this paper are to determine the design and run-time 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 Fig. 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 run-time 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 [4].

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

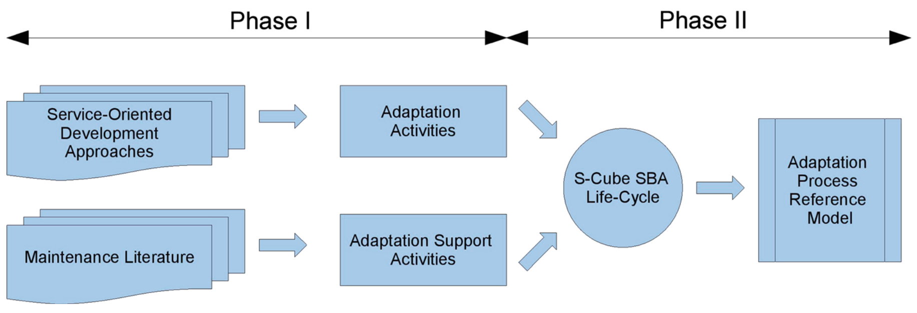
Phase I
Phase II
Service-Oriented Development Approaches
Maintenance Literature
Adaptation Activities
Adaptation Support Activities
S-Cube SBA Life-Cycle
Adaptation Process Reference Model

Fig. 2 Research methodology

reports from the S-Cube consortium (CD-JRA-1.1.2 [4], CD-JRA-1.2.1 [13], CD-JRA-1.2.2 [25]).

In order to satisfy 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.

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 literature. 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 activity 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 [4]
– 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 [13]
– CD-JRA-1.2.2 Taxonomy of Adaptation Principles and Mechanisms [25].

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 [4]. Several approaches such as those proposed

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 artefacts 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 realisation mechanism
Collect monitoring results for adaptation	Results of monitoring are collected and analysed
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 realisation mechanisms based on the selected adaption strategy

by Cortellessa et al. [26] or Adil kenzi et al. [27] 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 [28] 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 BPEL as input and generating a composition based on the BPEL specification. BPEL is a Business Process Execution Language tailored to meet the needs of Web Services. WS-mon is a monitoring tool that 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 that 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 [29] outlines the activities for each of the following SBA life-cycle 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 processes 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 that promotes interoperability, discoverability and standardisation of service technologies. To some extent, the BEA life-cycle caters for adaptability as it provides service monitoring, run-time correctness analysis and operational management activities.

Chang [30] proposes a process model that focuses on developing highly adaptable web services. It follows the sequence of steps specified in the SOAD [31] framework, namely service *identification*, service *specification* and service *realisation*. The process model contains six processes each of which contain several activities. The processes are analysing 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 artefacts 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) [32] is influenced by several established life-cycles such as RUP [33], CBD [34] and BPM [35]. 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 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 [36] 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 a sub-process of adaptation. The activities are summerised in Table 2. They are categorised based on the activity categories identified in Sect. 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 run-time adaptation.

The **Astro** toolset contains a monitoring tool that 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

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: Run-time 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

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* 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 that 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, and *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 that 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 run-time adaptation (*Runtime Service Discovery*) of SBAs. It contains two activities that 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 run-time corrective adaptation of an 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 Sect. 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 that 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 subprocesses 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 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 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 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 that 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

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

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.

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 Sect. 5.2.2 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 Sect. 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 the 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 limited 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) predefined.

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.

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.

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: Run-time Service Discovery

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 whether adaptation is necessary. Similarly, a modification request procedure would allow engineers to track modification requests and determine whether 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. [37] 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 analyse 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

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: Run-time Service Discovery

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 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 artefacts 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 artefacts 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 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 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 activ-

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

ities 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 run-time.

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 reinterpretation. 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.

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 allows 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 [38] 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 [39]. 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 measuring the impact of each process.

Table 7 Software Metrics [38]

Category	Metrics	Measurement unit
Project performance	Budget performance	Currency
	Schedule performance	Time
	Earned value performance	Time
	Technical performance	Time
Process performance	Rework rate	Quantity of Development Cycles
	Defect rates	Quantity of Bugs
Product quality	Product complexity	Days per Function Point
	Defect density	Bugs per Function Point
Personnel performance	Personnel productivity	Days per Function Point
Orgainisational performance	Process maturity	Process Level Determined by Auditor
	Product quality	Quantity of Bugs
	Productivity	Days per Function Point
Enterprise performance	Profitability	Currency
	Return on equity	Currency
	Cost of sales	Currency
	Competitiveness	Currency

Table 8 Adaptation framework and assessment metrics

Process	Activity	Metrics
Requirements engineering and design	BEA: Define KPIs and management policies	Productivity, Personnel productivity, Product complexity
	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	Defect rates, Rework rate, Defect density
	SeCSE: Deploy the monitoring rules and recovery policies within the monitoring system	
Operation and management		
Identify adaptation needs	Define adaptation requirements	Productivity, Personnel productivity, Product complexity
	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	Productivity, Personnel productivity
	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	

Table 8 continued

Process	Activity	Metrics
Enact adaptation	Perform adaptation	Rework rate, Defect rates, Defect density
	14764: Process Implementation: Configuration management	
	14767: Modification Implementation: Development process	
	SDLC: Readjust service weights for request queues	
	SeCSE: Run-time Service Discovery	

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, organisational and personnel productivity metrics are suggested for the "Identify adaptation needs" and "Select adaptation strategies" processes because these processes are often manual.

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 sub-sections, we will introduce SBA solutions and their 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 [4]. 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 company's 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 [40].

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 run-time adaptation in an interesting way. They have an adaptation dashboard that is used at run-time 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 the 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 whether 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 Fig. 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, run-time adaptation processes

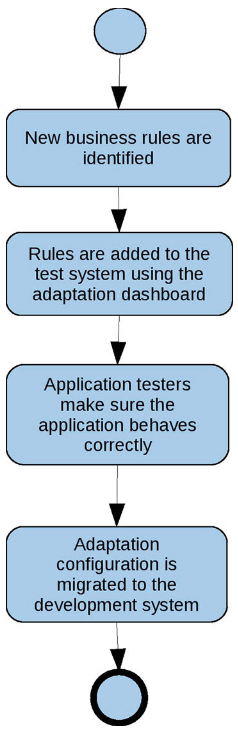
New business rules are identified
Rules are added to the test system using the adaptation dashboard
Application testers make sure the application behaves correctly
Adaptation configuration is migrated to the development system

Fig. 3 SBA solutions adaptation process

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.

SBA solutions employ a variety of software engineering tools to mange their software process. Key to the management of their process are their issue tracking and configuration management tools. They use version control software called Perforce to manage their software configurations. Perforce[1] allows them to keep track of all of the changes made to their source code facilitating traceability and recovery if bugs are introduced. Code and configurations are deployed from Perforce to their development, test and production environments.

They also rely on a commercial software package called Jira[2] to manage the status of software deliverables such as requirements specifications, software designs and code deliverables. Jira also manages test incidents and bugs identified in their software. A powerful feature of Jira is that it guides the workflow of software deliverables ensuring that they follow the correct process.

Along with Jira to manage their workflow and deliverables, they use a content management system (CMS) to store their software process documentation so that it is available to all of their employees. The CMS is easily updated, so that improvements made to their process can be reflected in their process documentation and made available immediately via the web-based CMS.

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 development 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 with-

[1] http://www.perforce.com/.

[2] http://www.atlassian.com/software/jira.

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: Run-time Service Discovery

out 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 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 determine the overhead of implementing it in real projects.

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 partially supported by Lero—the Irish Software Engineering Research Centre, Science Foundation Ireland Grants No. 03/CE2/I303.1 & 10/CE/I1855.

References

1. Cai H, Bu F, Jiang L (2012) A business-driven methodology for service-oriented information system development. In: Parallel and distributed processing symposium workshops Ph.D. forum (IPDPSW), 2012 IEEE 26th, international, pp 2292–2299
2. Retter R, Fehling C, Karastoyanova D, Leymann F, Schleicher D (2012) Combining horizontal and vertical composition of services. Serv Oriented Comput Appl 6:117–130. doi:10.1007/s11761-011-0095-6
3. Xu B, Luo S, Yan Y, Sun K (2012) Towards efficiency of qos-driven semantic web service composition for large-scale service-oriented systems. Serv Oriented Comput Appl 6:1–13. doi:10.1007/s11761-011-0085-8
4. Andrikopoulos V (2009) Separate design knowledge models for software engineering and service based computing. S-Cube Consortium, Deliverable CD-JRA-1.1.2
5. Richardson I, Lane S (2009) Coordinated design knowledge models for software engineering and service-based computing. S-Cube Consortium, Deliverable CD-JRA-1.1.4
6. Royce W (1970) Managing the development of large software systems. Proc IEEE Wescon 26(1):9
7. Boehm B (1986) A spiral model of software development and enhancement. ACM SIGSOFT Softw Eng Notes 11(4):14–24
8. Oreizy P, Gorlick MM, Taylor RN, Heimhigner D, Johnson G, Medvidovic N, Quilici A, Rosenblum DS, Wolf AL (1999) An architecture-based approach to self-adaptive software. IEEE Intell Sys Appl 14(3):54–62
9. Gu Q, Lago P (2011) Guiding the selection of service-oriented software engineering methodologies. Serv Oriented Comput Appl 5:203–223. doi:10.1007/s11761-011-0080-0
10. Hinchey M, Coyle L (2009) Evolving critical systems. Lero—The Irish Software Engineering Research Centre, Technical Report Lero-TR-2009-00
11. Herstellerinitiative software (his) process assessment working group. Available: http://portal.automotive-his.de/images/pdf/ProcessAssessment/his-wg-assessments-v31-07-06-08.pdf
12. Amazon. Available: http://www.amazon.com/
13. Benbernou S (2008) 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
14. Williams SK, Battle SA, Cuadrado JE (2006) Protocol mediation for adaptation in semantic web services. In: The semantic web: research and applications. ser. Lecture Notes in computer science, vol 4011. Springer, Berlin, pp 635–649
15. Di Nitto E, Penta MD, Gambi A, Ripa G, Villani M (2009) Negotiation of service level agreements: an architecture and a search-based approach. In: Service-oriented computing ICSOC 2007. ser. Lecture notes in computer science, vol 4749. Springer, Berlin, pp 295–306
16. Pernici B (2007) Automatic learning of repair strategies for web services. In: Fifth European conference on web services (ECOWS '07). IEEE Computer Society, Halle, Germany, pp 119–128
17. Pistore M, Barbon F, Bertoli P, Shaparau D, Traverso P (2004) Planning and monitoring web service composition. In: Artificial intelligence: methodology, systems, and applications. ser. Lecture notes in computer science. Springer, Berlin, pp 106–115
18. Conte SD, Dunsmore HE, Shen VY (1986) Software engineering metrics and models. Benjamin-Cummings Publishing Co. Inc., CA
19. Lehman MM (1984) Program evolution. Info Proc Manag 20(1):19–36
20. April A, Hayes JH, Abran A, Dumke R (2005) Software maintenance maturity model (smmm): the software maintenance process model. J Softw Maint Evol Res Pract 17(3):197–223
21. Abran A, Bourque P, Dupuis R, Moore JW (2001) Guide to the software engineering body of Knowledge-SWEBOK
22. Swanson EB (1976) The dimensions of maintenance. In: Proceedings of the 2nd international conference on software engineering, pp 492–497
23. IEEE, ISO, and IEC,(2006) Software engineering-software life cycle processes-maintenance International standard. Institute of Electrical and Electronics Engineers, New York
24. Sommerville I (2004) Software engineering, 7th edn. Addison Wesley, Reading
25. Hielscher J, Metzger A, Kazhamiakin R (2009) Taxonomy of adaptation principles and mechanisms. S-Cube Consortium, Contractual Deliverable CD-JRA-1.2.2
26. Autili M, Berardinelli L, Cortellessa V, Marco AD, Ruscio DD, Inverardi P, Tivoli M (2009) A development process for self-adapting service oriented applications. In: Service-oriented computing ICSOC 2007, ser. Lecture notes in computer science, vol 4749. Springer, Berlin, pp 442–448
27. kenzi A, Asri BE, Nassar M, Kriouile A,(2009) A model driven framework for multiview service oriented system development. In: ACS/IEEE international conference on computer systems and applications. IEEE Computer Society, pp 404–411
28. Trainotti M, Pistore M, Calabrese G, Zacco G, Lucchese G, Barbon F, Bertoli P, Traverso P (2005) Astro: Supporting composition and execution of web services. In: Lecture notes in computer science, vol 3826, p 495
29. Durvasula S et al (2007) Introduction to service lifecycle. SOA practitioners guide. part 3
30. Chang SH (2007) A systematic analysis and design approach to develop adaptable services in service oriented computing. In: IEEE Congress on services, 2007, pp. 375–378
31. Arsanjani A (2004) Service-oriented modeling and architecture. Available: http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/
32. Papazoglou MP, Heuvel WVD (2006) Service-oriented design and development methodology. Int J Web Eng Technol 2(4):412–442, Available: http://portal.acm.org/citation.cfm?id=1358575.1358582
33. Kruchten P (2003) The rational unified process: an introduction, 3rd edn. Addison Wesley, Reading

34. Herzum P, Sims O (2000) Business components factory: a comprehensive overview of component-based development for the enterprise. Wiley, New York
35. Harmon P (2003) Second generation business process methodologies. Bus Process Trends 1(5)
36. ATOS (2007) SeCSE methodology, version 3. Technical Report
37. Fang R, Chen Y, Fong L, Lam L, Frank D, Vignola C, Du N (2007) A version-aware approach for web service client application. In: 10th IFIP/IEEE international symposium on integrated network management, IM'07, pp 401–409
38. Reifer DJ (2006) Metrics and management: a primer. In: Reifer DJ (ed) Software management, 7th edn. IEEE Computer Society, Silver Spring, pp 397–401
39. Humphrey W (1987) A Method for assessing the software engineering capability of contractors: preliminary version. Addison Wesley, Reading
40. Miles MB, Huberman AM (1994) Qualitative data analysis: an expanded sourcebook. SAGE publications Inc., Beverley Hills