Grant Agreement N° 215483

Title:	*Integration Framework Baseline*
Authors:	*City, FBK, INRIA, LERO, PoliMI, UCBL, UniDue, UPM, USTUTT, Tilburg, TUW*
Editor:	*Marco Pistore, Raman Kazhamiakin, Antonio Bucchiarone (FBK)*
Reviewers:	*Ita Richardson (LERO)* *Barbara Pernici (PoliMI)* *Jean-Louis Pazat (INRIA)*
Identifier:	*Deliverable # CD-IA-3.1.1*
Type:	*Deliverable*
Version:	*1.0*
Date:	*27 March 2009*
Status:	*Final*
Class:	*External*

Management summary

This document describes the baseline of the S-Cube Integrated Research Framework. The objective of this framework, which will be constantly and incrementally refined and extended throughout the whole life of the project, is to provide a holistic vision that integrates, aligns and coordinates the research efforts and results of the joint research activities undertaken in JRA-1 and JRA-2. The baseline for the Integrated Research Framework described in this document consists of a set of views which define different perspectives on the S-Cube research: conceptual framework, reference life-cycle, logical run-time architecture, logical design environment. In this deliverable, we provide a description of these views and a first definition of the interfaces between the elements of the framework. We map the research efforts undertaken in the different joint research activities into these views. Finally, we define responsibilities for the different research work-packages, and relationships among them, in terms of their contributions to these views.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 215483 (S-Cube).

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator) – UniDue	Germany
Tilburg University – Tilburg	Netherlands
City University London – CITY	U.K.
Consiglio Nazionale delle Ricerche – CNR	Italy
Center for Scientific and Technological Research – FBK	Italy
The French National Institute for Research in Computer Science and Control – INRIA	France
The Irish Software Engineering Research Centre – Lero	Ireland
Politecnico di Milano – Polimi	Italy
MTA SZTAKI – Computer and Automation Research Institute – SZTAKI	Hungary
Vienna University of Technology – TUW	Austria
Universitá Claude Bernard Lyon – UCBL	France
University of Crete – UOC	Greece
Universidad Politécnica de Madrid – UPM	Spain
University of Stuttgart – USTUTT	Germany
University of Hamburg	Germany
VU Amsterdam	Netherlands

Published S-Cube documents

These documents are all available from the S-Cube Web Portal at `http://www.s-cube-network.eu/`

Contents

List of Figures

Chapter 1

Introduction

This deliverable describes the first, base version of the S-Cube Integrated Research Framework (IRF), which is one of the pivotal elements for ensuring overall integration, consistency and harmonisation of the research efforts undertaken by the network.

In this introductory chapter, we define the context of the IRF. We start by describing the objectives of IA-3, where the definition of the IRF belongs (Section 1.1). We then discuss the objectives for the IRF baseline described in this deliverable, and the approach that we adopted for its definition (Section 1.2). We finally discuss the structure of the rest of the deliverable (Section 1.3).

1.1 Activity IA-3: Integration Framework

The overall goal of Integration Activity 3 ("Integration Framework for Service-based Applications") is to guarantee the coherency and integration of the research efforts undertaken by S-Cube. This is achieved through the definition of a holistic framework that aligns and coordinates the results of the joint research activities. This overall goal will be achieved through the realization of the following detailed objectives (see also [1]):

- To define, and progressively refine, the Integrated Research Framework (IRF), i.e., a coherent, holistic framework for S-Cube research, which allows for integrating the principles, techniques, methods and mechanisms studied in S-Cube. In particular, the IRF should encompass those aspects of the research that are cross-cutting.
- To guarantee the overall coherence and alignment of all the research work-packages by defining their contributions to the IRF and by identifying boundaries and interfaces among the investigations undertaken by the different work-packages.
- To validate the IRF through suitable industrial case studies. The ultimate goal of the validation is to revise and improve the IRF. For this reason, it will be conducted iteratively for the whole duration of the activity.
- To identify the different classes of users that are involved with different roles in a Service Based Application (SBA), and to define customisations and refinements of the IRF that are tailored and personalised to these various classes of users.
- To help comparing the investigation undertaken by S-Cube with other proposals and approaches, both from industry and from academics; to help strengthening the relations of S-Cube with the research community; and to help identifying missing competences within the S-Cube consortium.

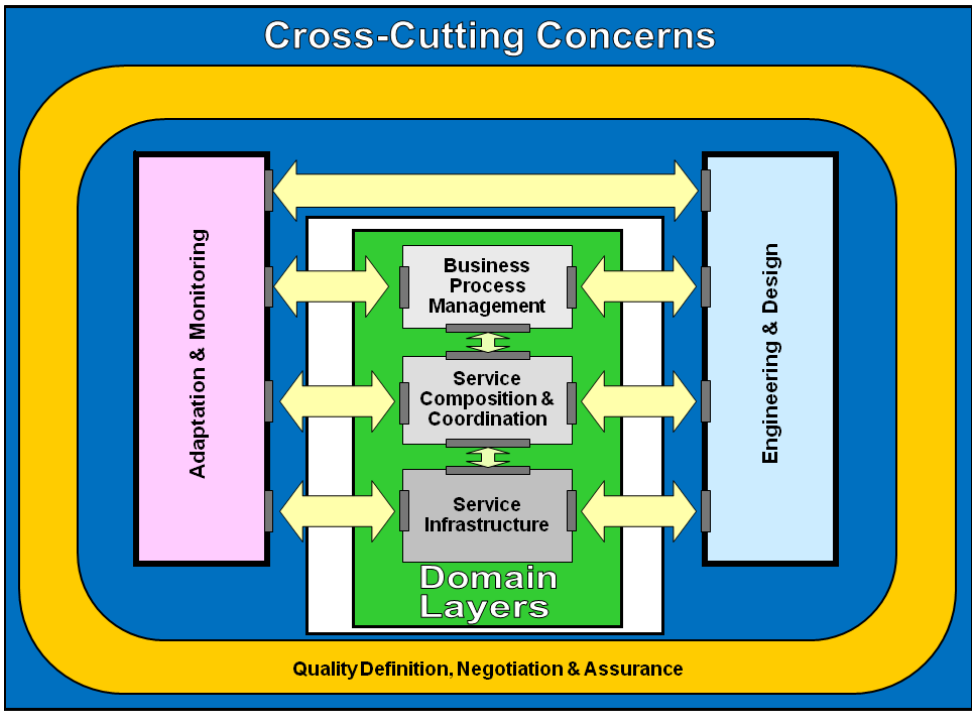

Figure 1.1: The conceptual framework used as staring point for the definition of the IRF

1.2 IRF Baseline: Objectives and Approach

This deliverable reports the outcomes of the activities undertaken in Task T-IA-3.1.1 of the IA-3 activity, whose aim is to *define the baseline upon which the IRF will be defined and progressively refined throughout the whole life of the project* [1].

This work has been conducted in close cooperation with the research work-packages, and is actually based on two fundamental contributions coming from these work-packages. Firstly, the work is based on the *systematic survey and analysis of the state of the art* undertaken during the first months of the project by the research work-packages. This survey and analysis has addressed engineering, adaptation, and quality assurance methodologies and processes [2, 3, 4] well as mechanisms and techniques for service infrastructure, service composition, and business process management [5, 6, 7] and has taken into account literature both from industry and from academics (see also [8]). Secondly, the work has exploited the *research visions and road-maps defined by the research work-packages*. These visions complement the analysis of the state of the art by identifying short-term and long-term research challenges for the work-packages.

The purpose of the activities undertaken in IA-3 so far has been to provide a holistic, coherent framework for the research activities of the different S-Cube work-packages. In particular, the purpose has been to identify gaps, overlaps and synergies among the different research efforts undertaken by S-Cube, and to identify clear responsibilities and interfaces for the efforts of the different research work-packages. This has been performed in an iterative way, in order to build an increasing alignment of the vision of the different research work-packages within the IRF. In particular, this integration in the vision has been consolidated during the S-Cube meetings[1], and through specific coordination activities undertaken with the research work-package and activity leaders.

The starting point for this has been the high-level description of the conceptual framework for the

[1] In particular, the IRF has been discussed in detail during the Global Meeting hold in Crete in September 2008, during the Integration Committee meeting hold in Amsterdam in February 2009, and during the Global Meeting hold in Lyon in March 2009. This deliverable reports the version of the IRF that has been discussed and approved during the Global Meeting in Lyon.

S-Cube research activities which is reported in Figure 1.1. This conceptual architecture identifies the six elements corresponding to the six work-packages in the two joint research activities, and defines some interfaces among those elements, corresponding to the mutual dependencies and expectations among the research work-packages.

The work undertaken in IA-3 for the definition of the IRF has led to the following main results:

- Firstly, the conceptual framework has been analysed, updated, and refined in order to better capture the roles and responsibilities of the different research work-packages. In particular, this evolution is based on a clear separation of responsibilities between the work-packages in JRA-2 and the work-packages in JRA-1. The former which focus the research on local technologies and mechanisms which are specific to one of the SOA domain layers. The latter which focus on overarching principles and techniques for issues that are cross cutting to these layers.

- Secondly, clear interfaces have been identified and defined based on the interactions between the elements of this conceptual framework. These interfaces are based on the capabilities that an element of the architecture offers to the others. In particular, capabilities are used to define the interfaces between the domain layers in JRA-2 and the cross-cutting issues in JRA-1 — namely, the capabilities provided by the domain layers will be exploited in order to manage the cross-cutting issues. In addition, capabilities are used to describe the interfaces and mutual dependencies among the different cross-cutting issues.[2]

- Thirdly, additional views have been added to the IRF. These views complete and complement the one given by the conceptual framework, by covering the aspects related to the life-cycle, the run-time architecture, and the design environment of SBAs. These complementary views are necessary to give a comprehensive overview of the research undertaken by S-Cube, and are essential to clarify the relations among the research efforts and to better understand responsibilities and mutual expectations of the different research work-packages.

- Fourthly, an initial mapping of the research work-packages into the IRF has been defined. This mapping describes the research concerns of the different work-packages and their contributions to the overall research framework in terms of their contributions to the views defining the IRF.

We note that the IRF is not a static object, but will evolve and be refined and extended throughout the whole life of S-Cube. These revisions may involve the introduction of additional views, in case new perspectives will become useful to describe the research framework. Moreover, the specific content of the different views and the mapping of the research efforts and results into the IRF will also be updated and refined in order to reflect the progress of the activities in the project.

1.3 Structure of the Deliverable

The structure for the remaining chapters in the deliverable is the following. Chapter 2 will define the baseline version of the IRF by describing the four different views it provides. In addition to this, the chapter will also describe the interfaces among the different elements of the conceptual framework. Chapter 3 will provide a description of the responsibilities and scope of the research work-packages by mapping them into the IRF. Chapter 4 discusses the next steps in the definition of the IRF, the strategies that we foresee for its management and update, and its usage for the validation of the research undertaken by S-Cube. Finally, Chapter 5 provides some concluding remarks. A list of acronyms used in the document can be found in Appendix A.

[2] According to [1], the description of the interfaces among the elements of the conceptual framework is the content of a different document, namely Deliverable PO-IA-3.1.2. Since the definition of such interfaces is strictly connected with the definition of the IRF, the content of the PO-IA-3.1.2 deliverable has been merged within the current document.

Chapter 2

Integrated Research Framework

In this chapter we define the baseline for the S-Cube Integrated Research Framework (IRF). As discussed in the Introduction, this framework, which will be constantly and incrementally refined and extended throughout the whole life of the project, has the objective of providing a common understanding of the research efforts undertaken by S-Cube.

The baseline of the IRF which is described in this chapter will provide a first integrated and coherent description of the integrated S-Cube framework which is being investigated in the joint research activities. To this purpose, *four different views of the research undertaken by S-Cube* have been identified, which provide different perspectives of the integrated research efforts. The first view is given by the **conceptual research framework**, which provides a high-level conceptual view of the S-Cube research activities. This view corresponds to the refinement and consolidation of the initial conceptual framework represented in Figure 1.1. This view clarifies the relations and *complementary roles of the research undertaken on the domain layers* (service infrastructure, service composition and coordination, and business process management) *and of the research undertaken on the cross-cutting issues* (engineering and design, adaptation and monitoring, and quality assurance). More in general, it defines *clear interfaces among the research efforts undertaken by the different research work-packages*, and clarifies mutual expectations and dependencies among them.

The three additional views which define the IRF have been introduced to complete the high-level, static view of the organisation of the research within S-Cube which is provided by the conceptual research framework. More precisely, these three additional views have been introduced to provide different perspectives to the definition and integration of the research efforts of the project. These three views define:

- A **reference life-cycle**, which shows how the different research activities contribute to support a coherent design and run-time life-cycle of an SBA. The core aspect of the proposed life-cycle is that it is *composed of two main cycles, one for the classical analysis and design phase and another for the run-time perspective, including monitoring and adaptation*. In fact, one of the key aspects of an SBA is that a strong interplay between the design and the run-time is needed in order to meet the expectations of these applications in terms of dynamicity, pro-activeness, adaptability, and quality assurance. As we will see, the proposed two-cycle approach allows for a smooth transition between the runtime operation and the analysis and design phases in order to allow for a continuous improvement and update of the SBA. As a consequence, by associating the research efforts of S-Cube within the reference life-cycle, we guarantee that they are aligned with these expectations.

- A **logical run-time architecture**, which illustrates the research efforts in terms of contributions to a hypothetical infrastructure for the execution of SBAs. The proposed run-time architecture is *service-oriented*, that is, it assumes that *both the engines providing infrastructural services* (e.g., services supporting the execution monitoring, testing, negotiation of SBAs) *and the application*

specific services are exposed on the same communication back-bone. By adopting this logical run-time architecture, we guarantee that the run-time mechanisms studied within the different work-packages define a coherent picture. By adoption a service-oriented architecture, we guarantee that these mechanisms can be integrated and exploited in a synergistic way, at least at the conceptual level.

- A **logical design environment** for SBAs, whose purpose is similar to that of the run-time architecture, but addresses the design phase. More precisely, this view provides a context where to place the envisioned techniques and mechanisms investigated by S-Cube to support the analysis and design of a SBA. This is achieved by *placing these techniques in a matrix that shows their relevance for the three domain layers, as well as according to the provided modelling, generation, deployment, verification functionalities*. By adopting this logical design environment, we guarantee that the efforts undertaken by the different research work-packages define a coherent picture that supports the different aspects of the design of a SBA.

We note that the four views defining the IRF are not independent. On the contrary, they are strongly interconnected, as they provide different perspectives for visualising and exploring the whole corpus of research undertaken by S-Cube.

In the rest of this chapter, we define the IRF by describing in detail the four views just introduced. More precisely, Section 2.1 describes the conceptual research framework, while Section 2.2 provides a detailed description of the interfaces among the different blocks of this framework. The following three sections focus on the three other views, namely, the reference life-cycle (Section 2.3), the logical run-time architecture (Section 2.4), and the logical design environment (Section 2.5). Section 2.6 completes the chapter with some concluding remarks.

2.1 Conceptual Research Framework

The conceptual research framework (see Figure 2.1) is the core element in the definition of the IRF. Its aim is to organise the joint research activities within S-Cube by providing a *high-level conceptual architecture for the principles and methods* for engineering service-based applications, as well as *for the technologies and mechanisms* which are used to realize those applications.

The framework consists of six components, which are in 1-to-1 relation with the six research work-packages of the network. Moreover, the framework distinguishes between the horizontal components corresponding to the "traditional" *domain layers* of a SBA, i.e., "Service Infrastructure", "Service Composition and Coordination", and "Business Process Management", and the vertical components, which correspond to the *cross-cutting issues* addressed by the project, namely "Engineering and Design", "Adaptation and Monitoring", and "Quality Definition, Negotiation and Assurance".

We note that the distinction between the two kinds of components is one of the core elements of the S-Cube approach. Indeed, an element that makes the S-Cube framework unique when compared to the traditional "layered" approach is that the framework *systematically addresses cross-cutting issues*. The framework sets out to *make explicit the knowledge of the horizontal layers* that is relevant for these cross-cutting issues, and that currently is mostly hidden in languages, standards, mechanisms, and so on that are defined and investigated in isolation at the different layers. More precisely, the approach underlying the framework is that the domain layers offer (design, monitoring, adaptation, verification) capabilities that are relevant for the cross-cutting issues. The research efforts in the vertical components are responsible of defining over-arching principles and methodologies for addressing cross-cutting issues by exploiting in suitable ways the capabilities exposed by the horizontal components.

In order to support the vision just described, the responsibilities of the components of the conceptual research framework are defined by following a clear separation of concerns between the two kinds of components in the conceptual research framework, as discussed in the next paragraphs.

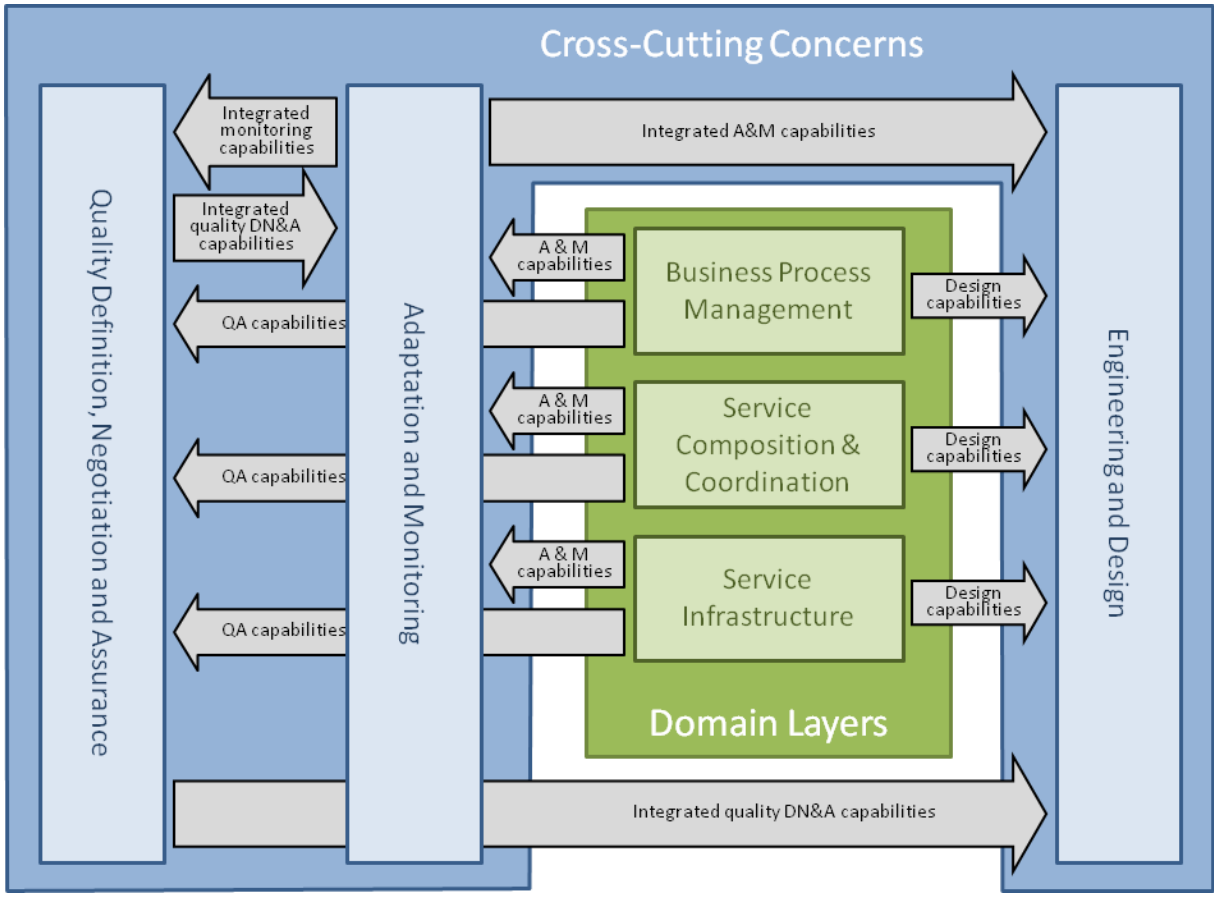

Figure 2.1: The conceptual research framework

2.1.1 Domain Layers: Domain-Specific Technologies and Mechanisms

Similar to the traditional SOA layers, the three domain layers of the framework are responsible for mechanisms and methods which are applicable locally within the layers. Also, concrete service technologies fall under the responsibility of the domain layers. More precisely:

- The "*Service Infrastructure*" layer studies a high-performance execution platform supporting adaptation and monitoring of SBAs (e.g., self-* mechanisms). The platform provides a set of core services, like service registries, discovery capabilities, and virtualisation services to the other layers.
- The "*Service Composition and Coordination*" layer focuses on novel service composition languages and techniques. In particular, it provides mechanisms to adapt and monitor service compositions.
- The "*Business Process Management*" layer addresses the aspects related to the modelling, designing, deploying, monitoring and managing of service networks, business processes and Key Performance Indicators (KPIs).

We note that the concern of the domain layers is primarily focused on studying and proposing *technologies and mechanisms at the different layers*. Their investigation also covers those *principles and methods* for the engineering, adaptation, and quality assurance *that are local to a specific layer*.

2.1.2 Cross-Cutting Issues: Over-Arching Principles and Methods

The concern of the vertical components of the conceptual framework is to address those over-arching principles, techniques and methods that are cross-cutting the domain layers. More precisely:

- The "*Engineering and Design*" concern covers the issues related to the life-cycle of services and SBAs. This includes principles and methods for identifying, designing, developing, deploying, finding, applying, provisioning, evolving, and maintaining services, while exploiting novel technologies from the functional layers. In particular, this aspect focuses on the quality of the SBA development process, on the roles and placement of the contextual properties of SBAs and human involvement, and on exploiting future service search engines for bottom-up SBA design.
- The "*Adaptation and Monitoring*" concern covers the issues related to the adaptation of a SBA. Specifically, this comprises languages and methods for defining adaptation goals and different adaptation strategies, which are triggered by monitoring events. An example for an adaptation technique that falls into the responsibility of this aspect is a strategy that correlates the monitoring events across the functional layers, thereby avoiding conflicting adaptations, or the one that aims to predict the potential SBA problems and perform adaptation activities pro-actively.
- The "*Quality Definition, Negotiation and Assurance*" concern involves principles and methods for defining, negotiating and ensuring quality attributes and Service Level Agreements (SLAs). Negotiating quality attributes requires understanding and aggregating quality attributes across the functional layers as well as agreeing on provided levels of quality. To ensure agreed quality attributes, techniques which are based on monitoring, testing or static analysis (e.g., model checking) are employed and extended by novel techniques exploiting future technologies (e.g., Web 2.0).

We note that this concern is primarily focused on providing *overarching principles and methods*, and to exploit the mechanisms and technologies studies by the domain layers. However, it also covers those *technologies and mechanisms that are not local to a specific domain layer*, and that are hence outside the scope of the concerns of the horizontal components. This is the case, for instance, of mechanisms for cross-layer monitoring and adaptation, or for quality negotiation.

2.2 Interfaces in the Conceptual Research Framework

In this section, we describe the interfaces that have been identified among the components of the conceptual research framework, and that are described in Figure 2.1. These interfaces are based on the concept of *capabilities* that a component of the architecture offers to the others. More precisely, for each component of the conceptual research framework, interfaces are defined that describe the mechanisms that the component makes available to the other components of the framework. In particular, capabilities are used to define the interfaces between the domain layers and the cross-cutting issues. These capabilities refer to design, adaptation and monitoring, and quality assurance mechanisms that are defined at the level of the single domain layers, and that are suitably exploited in order to manage the design, adaptation and monitoring, quality assurance of the whole SBA. In addition, capabilities are used to describe the interfaces and mutual dependencies among the different cross-cutting issues.

We note that the interface between two components implicitly defines a relation between the components that is inverse to the one defined in terms of the capabilities that the first component offers to the second one. This inverse relation corresponds to the *specifications and/or directives* that the second component has to provide to the first one in order to exploit the provided capabilities for enabling future, innovative SBAs. This inverse relation appears implicitly in the detailed description of the different interfaces that we are now going to provide. We will give some explicit examples of this inverse relation at the end of this section.

2.2.1 Interfaces Between Domain Layers and Cross-Cutting Issues

Monitoring Capabilities. These capabilities correspond to mechanisms and technologies defined at each domain layer for collecting information about the behaviour of the application and the changes in the environment. Indeed, monitoring capabilities at all layers are necessary to understand whether the SBA is executed and evolves in a normal mode and whether there are some deviations or violations of the desired or expected functionality. Each layer of Figure 2.1 provides monitoring capabilities to "Adaptation and Monitoring". This component is responsible of integrating these capabilities in order to achieve an over-arching, cross-layer monitoring of the SBA. To achieve this, additional mechanisms and technologies can be exploited that are not local to any domain layer and that are in the scope of the "Adaptation and Monitoring" component: this is the case, for instance, of mechanisms for correlating monitoring information coming from different layers.

In the "Business Process Management" layer, monitoring capabilities include real-time monitoring of business activities, the measurement of KPIs, as well as mechanisms for a proactive identification and notification of deviations. The "Service Composition and Coordination" layer provides monitoring capabilities that address the problem of checking whether certain predefined properties are satisfied when the service composition is executed. Monitored events at this layer are deviations of the expected model of execution of the services and of their compositions. Finally, in the "Service Infrastructure" layer, we have monitoring mechanisms that report data on the behaviour of the infrastructure on which the SBA is executed.

Adaptation Capabilities. This interface is used to specify adaptation capabilities that the different domain layers offer to the "Adaptation and Monitoring" component. This component is responsible of the over-arching approach for SBA adaptation that, once unexpected events or deviations are detected by the monitoring mechanisms, identifies a global, cross-layer adaptation strategy and realizes it by exploiting in suitable ways the adaptation capabilities of the different domain layers.

Adaptation capabilities at the "Business Process Management" layer may range from temporary modifications to specific components of the business processes to permanent reconfigurations of the whole business model and business network — the latter case often requires human intervention and re-design activities, on top of the adaptation capabilities offered by the domain layers. Adaptation capabilities at the "Service Composition and Coordination" layer correspond to the activation of pre-codified adaptation strategies. These strategies can be defined in various ways, ranging from procedural approaches (concrete actions to be performed are specified), over declarative approaches (the goals and requirements to be achieve are specified), to hybrid approaches. Adaptation capabilities at the "Service Infrastructure" layer provides mechanisms related to resource brokering, load balancing and renegotiation of Quality of Service (QoS) parameters, as well as mechanisms for service re-configuration and re-deployment.

Design Capabilities. Each layer of Figure 2.1 provides capabilities to the "Engineering and Design" of SBAs; these capabilities correspond to languages and mechanisms for modelling and specifying those aspects of a SBA that are specific to a domain layer. For example, the "Business Process Management" layer offers capabilities for modelling business processes (e.g., BPMN, or UML Activity Diagrams), as well as for specifying aspects related to the integration and execution of these business processes. The "Service Composition and Coordination" layer provides capabilities for modelling the single services, as well as service compositions (e.g., WSDL, BPEL). Finally, the "Service Infrastructure" layer provides capabilities for service discovery, for accessing service registries, and for managing service execution.

Quality Assurance Capabilities. Each domain layer provides capabilities that are exploited to achieve an end-to-end, cross-layer quality definition and assurance for the SBA. At the "Business Process Management" layer, these capabilities correspond to understanding how to express the relevant quality at-

tributes (e.g., KPIs) and the possibility of doing a static verification of the business process models, as well as of running simulations in order to predict and analyse the expected behaviour of these models. At the "Service Composition and Coordination" layer, the capabilities cover understanding the relevant quality attributes and how to do both static verification and simulation of single services and of service compositions. At this layer, capabilities may also concern the possibility of testing the service composition. The "Service Infrastructure" layer, finally, provides capabilities for expressing relevant infrastructual quality attributes, and capabilities for exploiting the infrastructures for running simulations or to test cases on SBAs.

2.2.2 Interfaces Among Cross-Cutting Issues

Integrated Adaptation and Monitoring Capabilities. The "Adaptation and Monitoring" component is responsible of defining overall, cross-layer monitoring and adaptation strategies that are then realized by exploiting the capabilities offered by the domain layers. These overall monitoring and adaptation strategies are in turn capabilities that the "Adaptation and Monitoring" component offers to the "Engineering and Design" component. Indeed, the knowledge of the capabilities and mechanisms for monitoring and adaptation, which will be available at run time, is crucial at design time in order to design and construct a SBA that is able to exploit those capabilities. Indeed, by "design for monitoring" and "design for adaptation" we refer to the possibility of designing SBAs whose behaviour relies on a full exploitation of the monitoring and adaptation capabilities offered by the framework.

Integrated Monitoring Capabilities. The capabilities of the "Adaptation and Monitoring" component which are related to the overall, cross-layer monitoring strategies are provided not only to the "Engineering and Design" component, but also to the "Quality Definition, Negotiation and Assurance" component. Indeed, these monitoring capabilities are not only essential for triggering the adaptation, but they are also a cornerstone for assessing the overall quality of the SBA.

Integrated Quality Definition, Assurance, and Negotiation Capabilities. These capabilities of the "Quality Definition, Negotiation and Assurance" component are offered to the "Engineering and Design" component, so that they can be exploited during the design and construction of a SBA. More precisely, these capabilities concern languages that can be exploited for defining the expected quality of a SBA; they concern mechanisms for negotiating quality attributes between service consumers and providers; and mechanisms for static analysis, simulation and testing of SBAs.

These capabilities are also offered to the "Adaptation and Monitoring" component, for the purpose of enabling pro-active adaptation on the basis of the analysis of the past, current and future quality of the SBA. Indeed, pro-active adaptation will exploit the testing, simulation and quality prediction mechanisms studied by the "Quality Definition, Negotiation and Assurance" component.

2.2.3 Interfaces: Concluding Remarks

In the whole section, we have based the definition of the interface between two components of the conceptual research framework on the description of the capabilities that the first component offers to the second one. We note, however, that each interface also defines implicitly an inverse relation between the two components, which corresponds to the *specifications and/or directives* that the second component has to provide to the first one in order to exploit the capabilities. So, for instance, the interfaces between the "Adaptation and Monitoring" component and the domain layers can be described in terms of the adaptation and monitoring directives that the component pushes to the layers in order to exploit in a suitable way their local monitoring and adaptation mechanisms. These directives correspond, for instance, to instructions on the monitoring mechanisms to be activated, on the properties to be monitored, and on the adaptation mechanisms and strategies to be instantiated. Similarly, the "Quality Definition, Negotiation

and Assurance" component provides to the "Engineering and Design" capabilities for expressing quality requirements. Conversely, the "Engineering and Design" component provides to "Quality Definition, Negotiation and Assurance" specifications of the expected quality properties that need to be assured by the SBA.

2.3 Reference Life-Cycle

In this section we describe the second view of the IRF, which corresponds to the life-cycle of SBAs which has been adopted as a reference within S-Cube. The key purpose of this view is to complement the static view provided by the conceptual research framework and to relate the research efforts undertaken by the network to the different phases of the life of the SBAs.

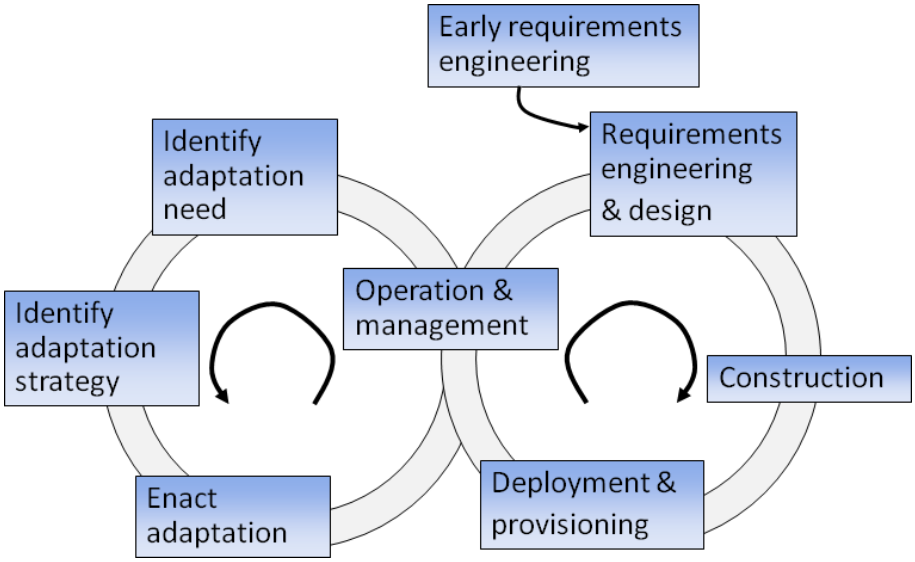

Figure 2.2: The reference life-cycle

The reference life-cycle is depicted in Figure 2.2. It is composed of two main cycles: the one on the right hand side corresponds to the classical application design, deployment and provisioning; the one on the left hand side corresponds to the adaptation perspectives. In fact, one of the key aspects of service based applications is that they need to be able to accommodate and manage various changes at runtime. By adopting this two-cycle approach, not only must applications undergo the transition between the runtime operation and the analysis and design phases in order to be continuously improved and updated, but they must provide mechanisms that, during runtime, continuously and automatically a) detect new problems, changes, and needs for adaptation, b) identify possible adaptation strategies, and c) enact them. These three steps are shown in the left hand side of the figure and lead to the deployment and provisioning of the modified application. The identification of the changes in the environment and of the problems in the execution of the SBA (e.g., failures) is obtained through monitoring and run-time quality assurance (they are part of the management activities typically performed during execution). This monitoring activity triggers the iteration of the adaptation cycle, whose effect is to inject changes directly into the application being operated and managed.

2.3.1 Description of the Phases of the Life-Cycle

A description of the life-cycle phases is provided below.

Early Requirements Engineering. This phase corresponds to the analysis and understanding of the problem by studying the existing organisational and business setting. In this phase, the requirements are expressed in terms of high-level concepts that correspond to the actors that are relevant in the setting, and to their goals, needs, and mutual dependencies, without any reference to the system-to-be.

The peculiarities in the development of SBAs make this phase particularly relevant. This is related to the high dynamicity of SBAs and to the necessity to guarantee the continuous adaptability and the evolvability of these applications. Indeed, in a context where the "system" is in continuous evolution and is characterised by very blurred boundaries, the study of those requirements that exist *a priori* in the organisational and business setting, and that are hence largely independent from the solution, becomes very important. These requirements are the most stable part in the description of a SBA[1] and serve as a basis for the engineering and the design of the whole application.

The requirements collected from the stakeholders in this phase should cover not only the functional aspects; they should cover also the quality expectations, expressed in high-level concepts such as KPIs; and they should cover the adaptation requirements and expectations of the actors.

Requirements Engineering and Design. This phase includes the usual requirement engineering and design activities, which take into account both functional and quality aspects of the SBA. While the main objectives of this phase are similar to the ones of any classical software development, there are some peculiarities that make development of SBAs different from others.

A first difference is that in many cases SBAs are built by exploiting already existing and running services. This means that, unlike traditional software engineering, the availability of services drives the requirement engineering as well as the design phase in such a way that the usage of these services is possible.

A second difference is that the requirements engineering and, even more, the design of a SBA have to be performed taking into account the three domain layers that define such an application. The models and languages provided by the different layers for the design of a SBA and for expressing its goals and properties need to be taken into account in this phase. This is also the phase where static analysis can be applied to the different models and specifications of a SBA, both locally to a specific layer and cross-layer.

A third difference is that the SBA has to be built to be able to react to new and/or critical conditions by triggering proper adaptation actions. At the level of requirement engineering this means that new classes of requirements have to be elicited and understood. These include the requirements for adaptation that define the circumstances in which the SBA should be adapted, their prioritization, and the objectives of the adaptations to be performed. At the level of design this means that proper adaptation strategies have to be designed together with the monitoring mechanisms that allow the adaptation needs to be identified.

Construction. After the design phase, the construction of the system can start. The peculiar element that has to be taken into account in this phase is that SBAs are obtained by the integration of different services. Specifically, this means that for establishing the desired end-to-end quality of those SBAs, contracts between the service providers and the service requestors (also known as service consumers) on quality aspects of services have to be established. Typically this requires some form of SLA negotiation and agreement.

The service composition construction could be *manual*, if the service integrator defines the service composition using appropriate specification languages (BPEL for example). Moreover the service composition construction could be *model-driven*, if service orchestration models are derived from abstract models. Finally, the service composition could be *automatically generated* starting from the service models and from higher level models of the business network and of the service choreography that are

[1] This fact is shown also graphically in Figure 2.2, where the Early Requirements Engineering phase is depicted outside the loop on the right hand side of the figure.

defined in the "Business Process Management" layer. We note that, in all cases, the service composition construction should cover not only the functional requirements, but also the QoS aspects and the adaptability requirements for the SBA.

Finally, in addition to the service composition, the construction phase will also realize all those mechanisms that are necessary for supporting the monitoring, adaptation, and quality assurance of the SBA.

Deployment and Provisioning. The deployment and provisioning phase comprises all the activities needed to make the SBA available to its users. A SBA can be itself a service: in this case, a proper description of its interface should be provided and published on some registry. In the literature, the early proposals of syntactic descriptions of services have been proven to be insufficient for the proper understanding of the service functionality. Consequently, semantic service descriptions of various kinds have been proposed. These include the description of the QoS characteristics of a service and enable for the definition of SLAs. In the case of adaptable SBAs, we could imagine that QoS and SLA information includes data on the adaptation characteristics of the SBA.

Operation and Management. The operation and management phase specifies all the activities needed for operating and managing a SBA. The literature also uses the term governance to mean all activities that govern the correct execution of SBAs (and related services) by ensuring that they respect the expected QoS level during execution. In this context, the identification of problems in the SBA (e.g., failures) plays a fundamental role. This identification is obtained by means of monitoring mechanism and, more in general, of mechanisms for run-time quality assurance. These mechanisms are able to detect failures, or critical conditions requiring the triggering of an adaptation mechanism needed to adapt SBAs.

Identify Adaptation Need. The information gathered during execution, the observation of the properties of the application, and the context of SBA constitute the elements on which the decision on the need for the SBA to adapt is based. Such decision may be automatically taken on the basis of monitoring requirements derived from adaptation requirements, or it may require human intervention (end user, system integrator, application manager). Moreover, such decision may be taken in a reactive way, when the problem has already occurred, or in a pro-active way, where the need is to prevent a potential problem.

Identify Adaptation Strategy. When the adaptation needs are understood, the corresponding adaptation strategies should be identified and selected. Among the possible adaptation strategies we mention service substitution, SLA re-negotiation, SBA re-configuration or service re-composition. The selection of the strategy may be automatic if either the SBA or the execution platform decide the action to perform, or it can be done by a human user.

Enact Adaptation. After the choice of the adaptation strategy, the adaptation mechanisms that implement it have to be activated. For example, service substitution, re-configuration, re-composition may be obtained using automated service discovery and dynamic binding mechanisms, while re-composition may be achieved using existing automated service composition techniques. As these examples show, the enactment of an adaptation strategy usually requires the exploitation of mechanisms provided by different layers, in particular by the "Service Composition and Coordination" and by the "Service Infrastructure" layers.

As for the previous two phases, human users may be involved in the implementation of the adaptation strategy, if the available adaptation mechanisms are not sufficient or not adequate to the adaptation need.

2.4 Logical Run-Time Architecture

The delivery of an integrated software environment is not among the objectives of the S-Cube project. However, a logical description of the envisioned run-time environment is important for aligning the joint research activities undertaken by S-Cube. For this reason, one of the views we defined for the IRF defines a logical run-time architecture for S-Cube. The purpose of this view is to guarantee a coherent picture for all the run-time mechanisms studied by S-Cube, that is, for all the mechanisms that are adopted in the operation and management phase and in all the left hand side of the reference life-cycle in Figure 2.2.

We stress that this architecture is logical — issues related to the physical realization and deployment of such an architecture are not addressed, at least in this first version of the IRF.

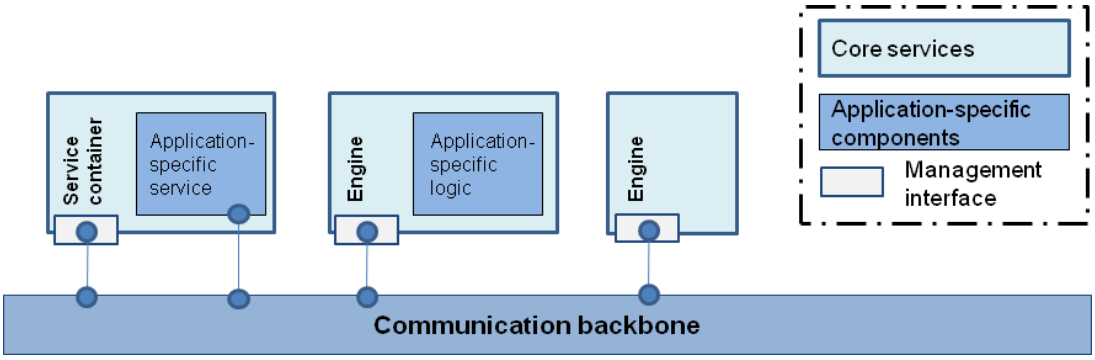

Figure 2.3: Run-time architecture: high-level view

2.4.1 High-Level Architecture

A high-level view of the run-time architecture is given in Figure 2.3. This picture illustrates all the relevant concepts of the architecture.

The proposed run-time architecture is *service-oriented*, that is, it assumes that all the run-time mechanisms and components are realized as services and are exposed on the same communication back-bone. By adopting a service-oriented architecture, we guarantee that the run-time mechanisms realized in the project can be integrated and exploited in a synergistic way, at least at the conceptual level.

In the picture, we distinguish between core services and application-specific services. The *core services* are middleware services that the run-time architecture provides to all SBA in order to support the different aspects of the SBA execution. Examples of such core services are a discovery service, an engine for executing service compositions, or an engine for monitoring the behaviour of a SBA or the performance of a business network. We note that these core services may belong to any of the domain layers of the conceptual research framework or may also belong to the "Adaptation and Monitoring" and "Quality Definition, Negotiation and Assurance" components. Some of these core services act as containers for *application-specific services*, i.e., services that are specific of the SBA in execution, and that encapsulate part of the application-specific logic. This is the case of the engine for executing service compositions.

The core services that act as containers for application-specific services are also referred to as *service containers*[2]. Other core services contain other parts of the application-specific logics, which are however not exposed as services. This is the case, for instance, of the monitoring engine, which will contain the application-specific properties to be monitored. For lack of a more precise name, these core services will

[2]We note that, in a physical realization of the proposed architecture, all services, regardless of their type, will be deployed onto a container. This includes also the core services. Since we are discussing a logical architecture here, we want to distinguish between core infrastructural services and application-specific services, and we use the term "service container" just for the containers of the second type of services.

also be referred to as *engines*. Finally, there are core services that have no application-specific contents. This is the case, for instance, of discovery services. When a SBA is deployed, specific pieces of the application logics will hence be deployed in the service containers and in the engines — depending on whether these pieces of application logics are themselves services or not — but no application-specific pieces are deployed in this third kind of core services.

The *communication backbone* supports the communication among any kind of services, regardless of whether they are core services or application-specific services. In particular, in the case of service containers, the communication backbone allows accessing both the core service and the application-specific services deployed within the container. The core service will offer a management interface for controlling the behaviour of the container and in particular the deployment and operation of application-specific services. The application-specific services will expose application-specific interfaces which allow for accessing the specific functionalities of these services. The fact that core services and application services are both exposed on the same communication backbone, and the possibilities offered by this solution in terms of interactions between the two logical levels, are key elements of the proposed architecture. Of course this is possible only for the pieces of application logics that are themselves services. In the case of engines, only the management infrastructure corresponding to the core service is exposed on the communication backbone. The pieces of application logics deployed within these engines are hence not accessible as services, even if they can be managed though the service interface exposed by the engine. So, for instance, it is possible exploit the backbone to interact with a monitoring engine to get information on the monitored properties or to activate or deactivate the actual monitoring of a certain property. It is however not possible to "invoke" the monitoring logics (i.e., the property that is monitored), since it is not a service.

2.4.2 Detailed Architecture

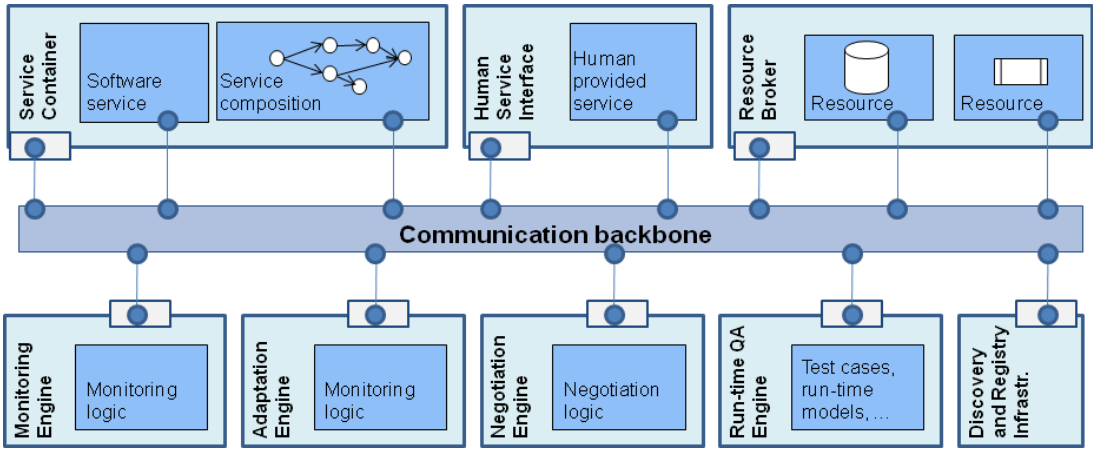

Figure 2.4: Runtime architecture: detailed view

Figure 2.4 provides a more detailed illustration of the run-time architecture of S-Cube, which contains specific instantiations of the different core services. More precisely, the figure represents explicitly the three kinds of service containers identified so far in the project. Besides the traditional containers for software services (web services and service compositions being two instances of this kind), we foresee containers also for human-provided services, and for the resources. That is, the communication backbone will have to provide access not just to the software components of an SBA. Also the resources exploited by the application and the parts of the application logics executed by humans have to be accessible from the backbone as services. This will be guaranteed by resource brokers and human interaction interfaces.

The figure also represents the different kinds of engines foreseen in the project, namely the engines for monitoring, for adaptation, for negotiation, and for run time quality assurance through testing and

simulation. All of them contain different kinds of application-specific logics. Finally, the figure contains an additional core service that offers discovery and registry functionalities.

Other services can be added to the run-time architecture in the refinements and extensions of the architecture that will be undertaken during the whole life of the project.

We note that the logical run-time architecture just presented will help integrating the S-Cube research activities in two ways:

- *Integration at the application-specific level*. Indeed, the architecture provides a description of the different pieces of application logics that define a SBA. That is, the (conceptual) implementation of a SBA consists of the definition of the different (software and human-provided) services, resources and other pieces of application logic to be deployed within the different elements of the architecture.

- *Integration at the infrastructural level*. Indeed, the architecture identifies different kinds of service containers, engines, and other elements. Each kind of architectural element has often more than one instantiation: e.g., there are different software service containers, different monitoring or negotiation engines, and so on. All the instances of a given kind of element will have to provide a similar management interface. The definition of this interface will hence correspond to the definition of the key characteristics and functionalities offered by the architectural element. So, for instance, the definition of the interface of software service containers or of monitoring engines will allow for a better (conceptual) interoperability among the different implementations of these architectural elements.

2.5 Logical Design Environment

In this section, we describe the last view of the IRF, which is complementary to the run-time architecture of the previous section, since it aims at providing a logical description of the foreseen design environment for SBAs. More precisely, the purpose of this view is to provide a context where to place the envisioned techniques and mechanisms that support the analyst and designer in the design of a SBA. While the run-time architecture covers the left hand side part of the SBA life-cycle described in Figure 2.2, the design environment covers the phases corresponding to the right hand side of the life-cycle, from early requirements engineering to deployment and provisioning. By adopting this logical design environment, we guarantee that the efforts undertaken by the different research work-packages define a coherent picture that supports the different aspects of the SBA design and engineering.

Similarly to the view in the previous section, also the design environment described in this section is logical, as the aspects related to the realization of a concrete integrated development platform are outside the scope of S-Cube.

The logical design environment is defined through two different pictures. The first one provides a taxonomy, or map, of the foreseen design techniques in terms of the functionality they provide and of their relevance for the three S-Cube domain layers. The second picture places the different design techniques into the reference life-cycle for SBAs that has been introduced in Section 2.3. The following two subsections describe these two pictures of the logical design environment.

2.5.1 Map of the Design Environment

In this subsection, we describe the foreseen design environment by mapping the different techniques we expect will constitute it — see Figure 2.5. These techniques are placed in a matrix, where the columns correspond to the different functionalities the foreseen design environment should provide, and the rows correspond to the three domain layers for Business Process Management, Service Composition and Coordination, and Service Infrastructure. Cross-layer techniques are represented by boxes that span over more rows.

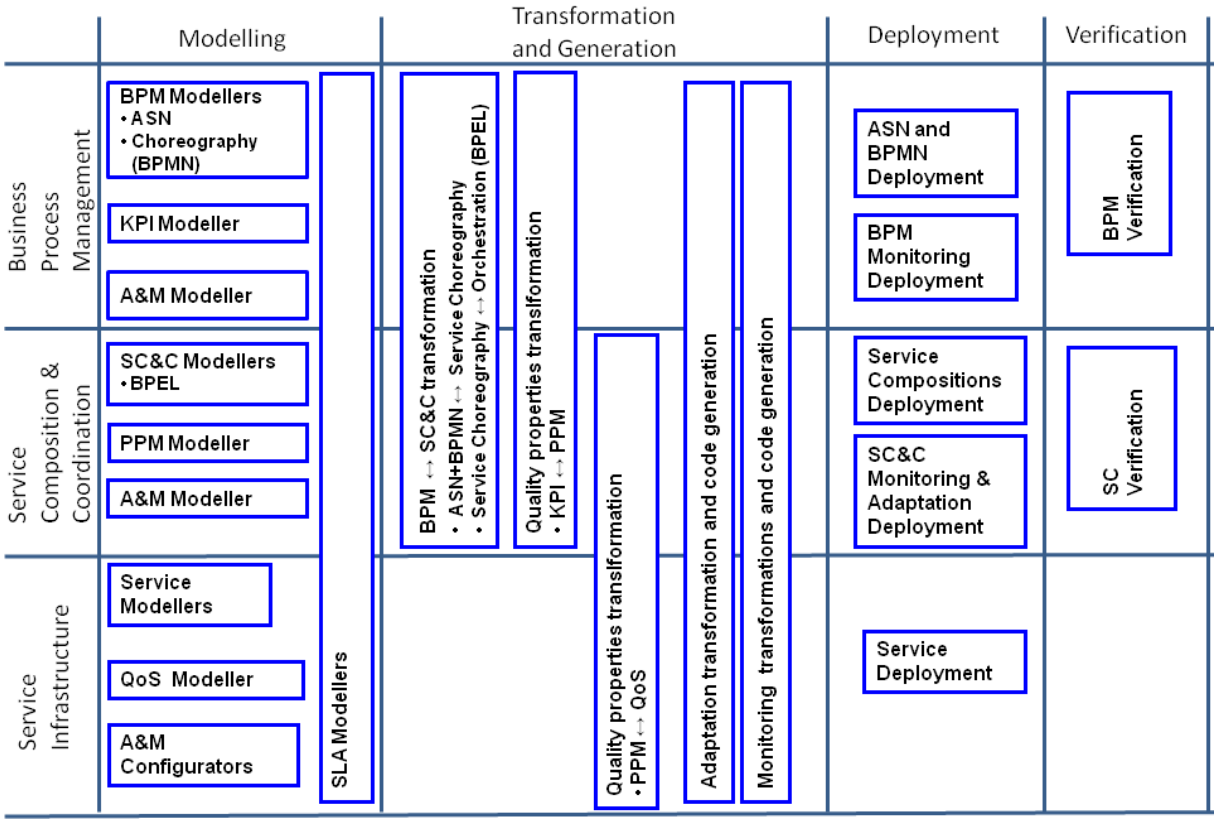

Figure 2.5: Logical design environment: matrix

Modelling Techniques. This set of functionalities corresponds to the techniques for modelling a SBA at the different domain layers, as well as for modelling the cross-cutting aspects of a SBA. More precisely, for each layer we must be able to provide techniques for modelling our SBA, for modelling the indicators that are used to evaluate the quality of the SBA, and for modelling the monitoring and adaptation aspects that are used to control and adapt the application at run-time.

At the "Business Process Management" layer, we model a SBA using ASNs and choreographies represented thorough BPMN diagrams. Quality aspects are modelled using KPI editors. Finally we use adaptation and monitoring modellers.

At the "Service Composition and Coordination" layer we model our SBA using service-based modelling techniques and languages based, e.g., on BPEL and WSDL. These techniques are able to describe single services as well as complex services compositions. Modelling techniques are also provided for specifying the monitoring and adaptation mechanisms that are used at run-time to check and adapt the SBA when some deviation/fault arises. In particular, the definition of monitoring techniques based on Process Performance Metrics (PPMs) is foreseen at this layer.

At the "Service Infrastructure" layer, the design environment must provide techniques for modelling services and the associated QoS aspects. These techniques will be exploited to provide the descriptions of services that will be made available in the service registries. Moreover it must provide techniques for monitoring and adapting configurations at runtime.

The modelling techniques foreseen in Figure 2.5 also include one cross-layer aspect, which is related to SLAs. Indeed, SLA modelling is based on different techniques that model different aspects of a SLA which are relevant at different layers. For example, SLA modelling at business process layer corresponds to the establishment of a business agreement, while SLA modelling at service composition layer means defining a contract between service provider and its consumer which covers the functional and quality aspects of the services.

Transformation and Generation Techniques. These techniques allow for transforming high-level models of the behaviour of a SBA into lower-level executable models, and vice-versa. They include for instance transformation techniques that generate BPEL code from BPMN, or that transform choreographies into orchestrations, and vice-versa.

Moreover, they contain techniques to transform high-level specifications of quality properties into lower-level specifications of the same properties and vice-versa, e.g. KPI to/from PPM models.

Finally, they include techniques for generating code in automatic way from the design models, as well as mechanisms to transform adaptation and monitoring specifications from one layer to another one. An example are mechanisms for transforming the monitoring and adaptation strategies specified by the designer into engine mechanisms that the service infrastructure will provide.

Deployment techniques. The third functionality corresponds to techniques for deploying the artifacts corresponding to a SBA specification at the different layers. This corresponds to deploying service networks, as well as the real/physical deployment of services on a service infrastructure.

This functionality also covers deployment techniques for the adaptation and monitoring mechanisms and specifications.

Verification techniques. To validate our models with respect to functional and non functional properties, the design environment must provide techniques to verify their correctness and completeness. Such verification techniques are available both at the Business Process Management and at the Service Composition and Coordination layers.

2.5.2 Design Environment Techniques in the Life-cycle

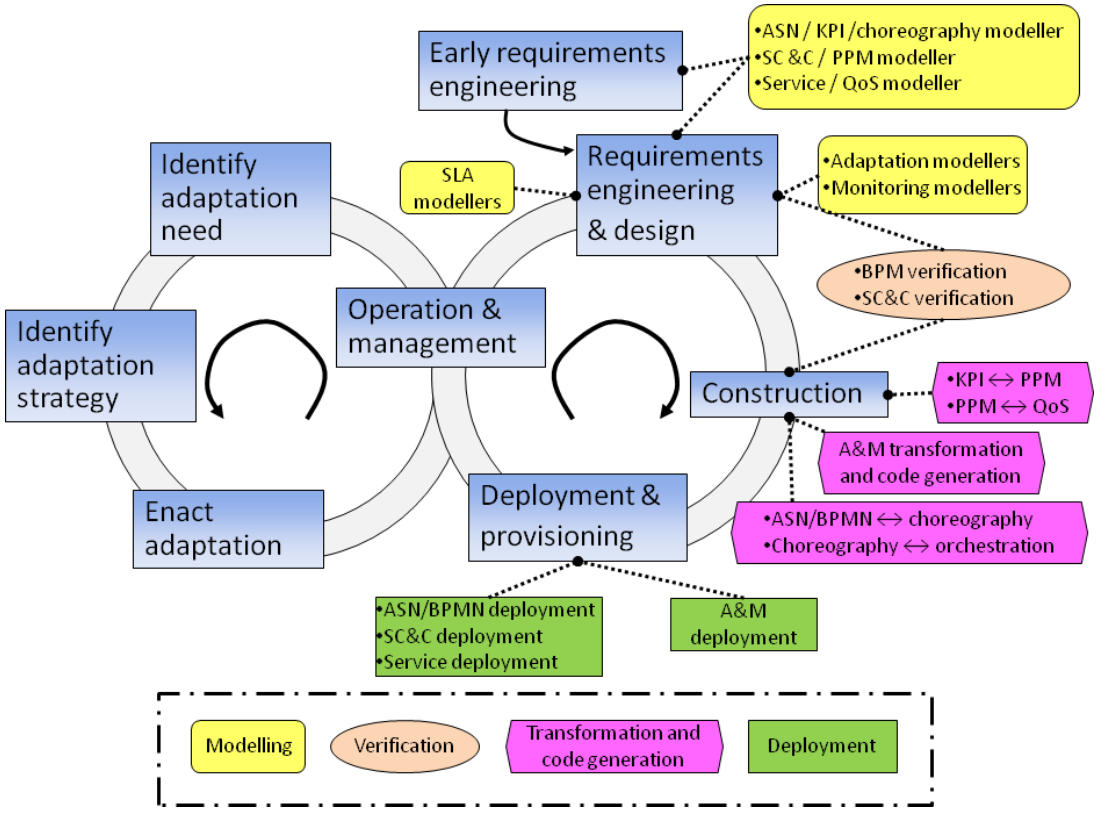

Figure 2.6: Design environment techniques mapped into the life-cycle

Figure 2.6 shows the strong relations existing between the different functionalities and techniques of the logical design environment and the life-cycle for SBAs defined in Section 2.3.

In particular, the modelling techniques are used by SBA engineers during the early requirements phase, where business-level models (ASNs, business metrics, KPIs) are defined, and in the requirements engineering and design phase, where the implementing services, service compositions and configurations, and the associated SLAs are defined.

The construction and deployment phases make intensive use of various transformation techniques. Following the model-driven approach, they allow for generating more specific, executable models from more abstract declarative models, e.g., executable orchestrations from choreography specifications or monitor programs and adaptation rules from the corresponding properties and strategies.

The verification techniques may be exploited both during the design and the construction phases, to check the correctness both of the specifications (e.g., validation of a BPMN specification or a choreography) and the implementations (e.g., verification of the properties of a service composition).

At the deployment and provisioning phase, specific techniques are needed to bring the elements of the SBA into operation. This includes techniques for deploying services and service compositions, monitor programs, and adaptation logic.

2.6 Concluding remarks

In this chapter we have presented the baseline for the IRF by defining four different view which provide different perspectives to this framework. The *conceptual research framework* identifies the different research areas in S-Cube and defines clear competences for these different areas and clear interfaces among them. The *reference life-cycle* describes the research in S-Cube by placing it within the different phases of the life and evolution of a SBA. The *run-time architecture* and *design environment*, finally, allow for a logical integration of the different research efforts in S-Cube by placing these efforts within hypothetical platforms for the design and run-time of SBAs.

Before moving to the next chapter, where we map into these views the contributions of the different S-Cube research work-packages, we stress again the strong synergy and the complementary roles played by the different views in the definition of the IRF. Indeed, while the conceptual research framework provides a view of the IRF that is static and focuses on the organisation of the research, the other three views concentrate on three main aspects of SBAs and of their life and evolution, namely their life-cycle, their run-time and their design. Each research component identified in the conceptual research framework contributes to several phases of the life-cycle and studies techniques, mechanisms and technologies that contribute in various ways to the run-time architecture and to the design environment. Vice-versa, different instances of a specific elements of the run-time architecture or of the design environment can be in the scope of different components of the conceptual framework. So, for instance, monitoring engines correspond to mechanisms studied by each domain layer, as well as to cross-layer mechanisms studied within the “Adaptation and Monitoring” component. A similar situation exists for the techniques that allow the specification of the quality properties of a SBA within the design environment. Strong relations also exist between the life-cycle and the view corresponding to the design environment and run-time architecture. Indeed, these two views correspond to the two loops that define the life-cycle, and refine the definition of the phases in the life of a SBA with a description of the techniques and mechanisms that support these phases.

Chapter 3

Contributions of the S-Cube Research WPs

In this chapter, we provide a first description of the contributions of the different S-Cube research work-packages to the IRF defined in the previous chapter. This description is still preliminary, as it is just the starting point of the process, which will take place during the second year of the project, and which will lead to the consolidation of the definition of the IRF and of the integration in the framework of results and challenges of the S-Cube research activities[1].

The chapter is organised in six sections, corresponding to the six research work-packages in JRA-1 and JRA-2. For each work-package, we define the contributions to the four different views of the IRF, and we illustrate the dependencies with other work-packages. The chapter is concluded by a section with final remarks.

3.1 Engineering Principles, Techniques and Methodologies for Hybrid, Service-based Applications (WP-JRA-1.1)

The main goal of WP-JRA-1.1 is to gather a clear understanding of how we can engineer SBAs that can live in the open world, by adapting and evolving themselves to best fit the situation in which they live. In particular, the work-package aims at investigating all the main phases of the life-cycle of Figure 2.2 in order to understand what the main principles for building SBAs are.

3.1.1 Contribution to the Conceptual Research Framework

In the conceptual framework, the work-package will contribute to the corresponding cross-cutting concern. This contribution will primarily focus on the provision to the other work-packages of the main principles and abstractions for developing adaptable SBAs, and on the analysis of the specific techniques and mechanisms that will be provided by the others to check if they coherently fit in the S-Cube reference life-cycle.

3.1.2 Contribution to the Reference Life-Cycle

The work-package is the owner of the life-cycle and is responsible for maintaining and improving it. Moreover, it mainly focuses on the identification of the specific principles and methods that can suitably used in the right hand side of the cycle in order to build adaptable SBAs.

[1]The outcomes of this consolidation activity will be reported in Deliverable CD-IA-3.1.2 “First Version of the Integration Framework”, due at Month 21.

3.1.3 Contribution to the Logical Run-Time Architecture

This work-package has no direct influence on the run-time architecture, but it is influenced by it, as the way a SBA is developed and operated depends also on the kinds of features and services that are available at the level of the run-time. For instance, the presence of the monitoring and adaptation infrastructure enables the development of applications ready to be monitored and adapted, and, in particular, the specific adaptation mechanisms that are available determine the kind of adaptation strategies that can be defined and enacted for the system. In addition, the work-package is also responsible to investigate the involvement of users during the run-time, e.g., to take decisions concerning the adaptation of the SBA.

3.1.4 Contribution to the Logical Design Environment

The work-package coordinates the various requirement engineering and modelling tools and techniques that will be made available by the other work-packages by providing guidelines and, for some proof of concept examples, glue code to use them together in a coherent way.

3.1.5 Relations with Other WPs

This work-package interacts with all others by providing design principles and methods and receiving specific techniques and approaches. In particular, the approaches for Agile Service Networks (ASNs) and business process definition developed by WP-JRA-2.1 will be incorporated into the requirement engineering and high-level design phase of the life-cycle. The same work-package will also offer inputs on the identification of the needs for adaptation that typically arise at the level of the business processes. Vice-versa, WP-JRA-2.1 will receive from the engineering work-package hints and suggestions on how the ASNs and business process aspects are incorporated into the comprehensive life-cycle.

The service composition metamodel and techniques provided by WP-JRA-2.2 will be incorporated into the construction phase. Vice-versa, WP-JRA-1.1 will provide to the other work-package new requirements for extending the composition metamodel and techniques to account for the design for monitoring and design for adaptation principles. For instance, a way to incorporate into the composition approach a proper model of the execution context should be identified. Other minor interactions with WP-JRA-2.2 concern the other phases of "deployment and provision" and all the phases of the adaptation life-cycle (left hand side of the figure).

The infrastructural services offered by WP-JRA-2.3 will be exploited and properly incorporated within the life-cycle, partially, in the design phases (e.g., the discovery mechanisms) and, partially, in the run-time phases (e.g., the low-level adaptation mechanisms and, again, the discovery mechanisms).

Finally, the relationships between WP-JRA-1.1 and the other WP-JRA-1.* are quite strict, and therefore, the work will be, in many cases, conducted in strict collaboration with these work-packages. More in detail, WP-JRA-1.2 owns the left hand-side of the life-cycle and coordinates with WP-JRA-1.1 to receive developed applications that are ready to be monitored and adapted. WP-JRA-1.3 offers all those techniques that are needed to provide some quality guarantees on the SBAs. Such quality guarantees can be provided at various different levels of abstraction and can concern various, if not all, phases of the life-cycle. The task of WP-JRA-1.1 is then to incorporate these techniques within the life-cycle and to apply those software engineering principles that lead to the design of verification-ready applications.

3.2 Adaptation and Monitoring Principles, Techniques and Methodologies for Service-based Applications (WP-JRA-1.2)

This work-package focuses on principles, techniques, and methods for adaptation and monitoring of SBAs. More specifically, it aims at providing novel principles and approaches for the integrated, cross-layer SBA adaptation and monitoring; it will elaborate on the solutions that exploit contextual information within the adaptation and monitoring activities; it will target the approaches that consider different

forms of involving the user in the monitoring process, such as user-driven monitoring, and in the adaptation process, such as human-in-the-loop adaptation. Finally, the work-package will also devise prediction models and approaches that anticipate the need for changes and thus trigger proactive adaptation.

3.2.1 Contribution to the Conceptual Research Framework

In the conceptual framework the adaptation and monitoring work-package will contribute to the corresponding cross-cutting aspect. This contribution, however, will primarily focus on cross-layer adaptation and monitoring, while the corresponding mechanisms at particular domain layers will be provided by the technology work-packages.

3.2.2 Contribution to the Reference Life-Cycle

Figure 3.1 illustrates the contributions of this work-package to the S-Cube life-cycle. In particular,

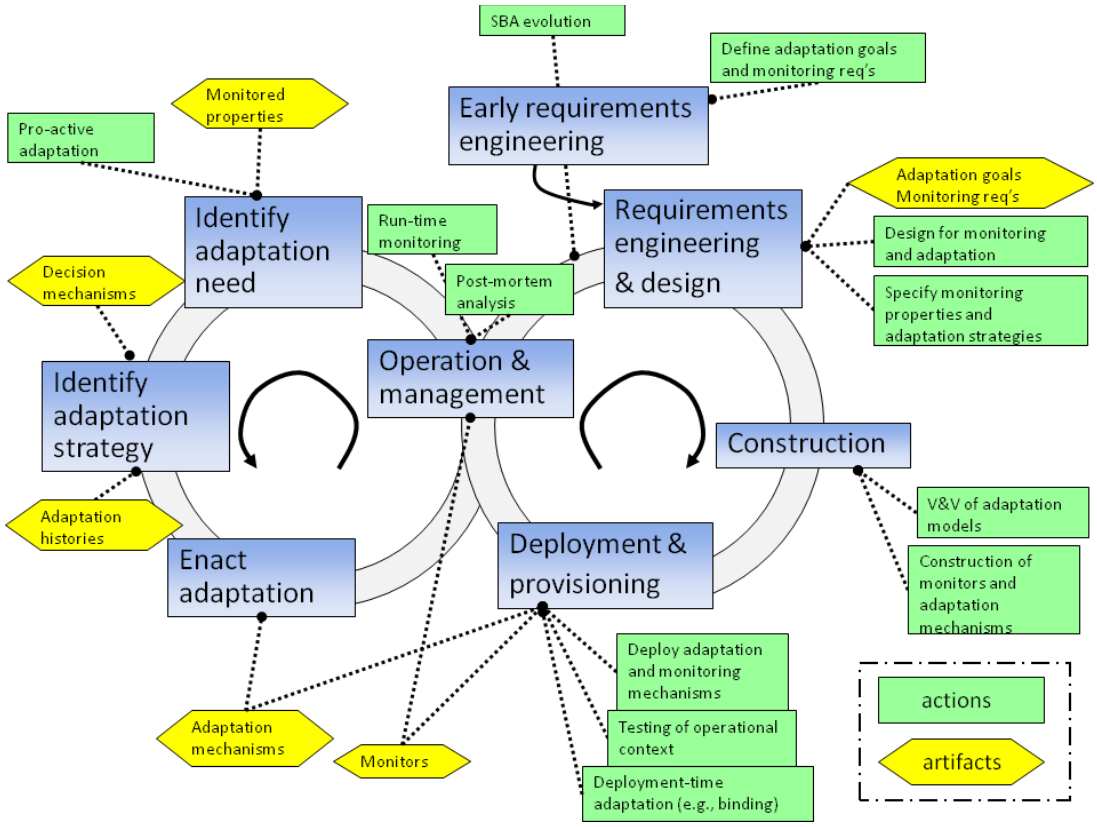

Figure 3.1: Reference life-cycle: contributions of JRA-WP-1.2

the figure shows how various adaptation- and monitoring-specific actions (hexagons) are carried out throughout the life-cycle of the SBA and how the corresponding artifacts (boxes) are exploited within those activities. We remark that different adaptation and monitoring activities and artifacts are highly interleaved through the SBA life-cycle and affect each other. Consequently, their definition, development, and exploitation should be performed in a holistic manner motivating the need for cross life-cycle monitoring and adaptation methods and approaches.

During *early requirements engineering*, the adaptation goals and monitoring requirements are defined. The goals and requirements are based on the quality model of the developed SBA and may involve various aspects, quality characteristics and attributes. In this phase, it is important not only to devise the appropriate monitoring and adaptation facilities, but also to see how the adaptation goals should be

achieved, that is, for instance, whether the application has to be modified pro-actively or re-actively, and what is the role of the contextual and user-specific information.

At the *requirements engineering and design* phase the goals and requirements are used to perform the design for adaptation and monitoring. In particular, the appropriate monitoring and adaptation architecture, the decision mechanisms, and the corresponding techniques are being selected, adopted, and instantiated. Using this model, the monitoring requirements and adaptation goals are being transformed into the monitoring properties and the adaptation strategies respectively.

During SBA *construction*, together with the construction of the SBA, the corresponding monitors and the adaptation mechanisms are being realized. It is also important in this phase to ensure that the developed adaptation mechanisms are neither contradictory nor in conflict with the application logic, i.e., to perform adaptation-specific quality assurance activity, such as validation and verification of the adaptation strategies, specifications, and realizations.

The *deployment* phase also involves the activities related to adaptation and monitoring: deployment of the monitors and monitoring mechanisms; deployment time adaptation actions (e.g., binding), testing and validation of operational context (e.g., evaluation of QoS metrics). After the *operation and management* phase, where the monitoring and adaptation activities interleave the application execution, specific post-execution activities may take place. This includes, for instance, the analysis of the previous SBA executions and adaptations, as well as the evolution of SBA and related mechanisms, bringing the latter back to the design phase.

The adaptation and monitoring mechanisms, tools, and facilities are actively exploited during the phases related to the modification of SBA, i.e., during the identification and realization of the adaptation needs. In particular, the most critical monitored properties characterise a serious deviation of the SBA functioning from the expected one, and therefore *identify adaptation needs*. Depending on the adaptation requirements this identification may be done re-actively or pro-actively. Various *adaptation strategies* developed during the design-time phase are instantiated and selected using the corresponding decision mechanisms, based on the current situation and/or on the knowledge obtained from the previous adaptations and executions. Finally, the *enactment of the adaptation strategy* is performed by the developed adaptation mechanisms. We remark that the implementation of these activities and phases may be performed by the SBA autonomously or may involve active participation of the various human actors (human-in-the-loop adaptation).

3.2.3 Contribution to the Logical Run-Time Architecture

The work-package will provide two main contributions to the run-time architecture (see also Figure 3.6).

First, the work-package will provide general guidelines for monitoring and adaptation instances, in terms of the functionalities that these will have to provide and of interfaces that they will have to expose. As a consequence, management protocols and usage patterns for monitoring and adaptation will be defined in terms of communications and service invocations occurring through the communication backbone. This will support the interoperability among the different (layer-specific or concern-specific) engines that contribute to the monitoring and adaptation of an SBA, and will contribute to the definition and management of a comprehensive, integrated monitoring and adaptation framework.

Second, the work-package will contribute with cross-layer monitoring and adaptation engines. Cross-layer mechanisms and techniques are necessary to overcome the limitations of the local mechanisms. Moreover, a comprehensive view of monitoring and adaptation at the level of the whole SBA is necessary to be able to decide at run-time among conflicting adaptation requirements.

3.2.4 Contribution to the Logical Design Environment

With respect to the design environment, the work-package will provide the principles for realizing cross-layer adaptation and monitoring mechanisms. These principles will play an important role for the definition, realization, and integration of various transformation and code generation tools. Concerning the

modelling environment and techniques, their definition will be a part of the design for monitoring and adaptation activity within the engineering and design work-package. Analogously, the tools for verification and validation of adaptation specification will be provided together with the end-to-end quality work-package.

3.2.5 Relations with Other WPs

The relationship between the monitoring and adaptation work-package with other WPs is represented with the following elements. WPs on the realization mechanisms (WP-JRA-2.*) will provide isolated adaptation and monitoring mechanisms at different functional domain layers (Business Activity Monitoring and post-mortem process analysis at BPM layer; mechanisms for run-time composition monitoring, and analysis, as well as automated service composition and adapter generation techniques at SC layer; Grid monitoring, SLA-negotiation, and self-* techniques at SI layer). These techniques are used to report relevant events and to satisfy adaptation requests at the corresponding domain layers. In order to enable coordination and management between isolated mechanisms, adaptation and monitoring work-package will come up with integrated cross-layer monitoring and adaptation approaches based on the mechanisms at different domain layers.

The SBA quality model and its elements (WP-JRA-1.3) is used as the basis to define the adaptation goals and monitoring requirements. Different quality dimensions provide a way to extract different types and different assets of the expected functionalities of the SBA, and therefore drive the design and definition of the corresponding properties to monitor and the adaptation strategies to be applied upon their violations.

In order to enable adaptation and monitoring functionalities of the SBA, the design and engineering process (subject of research in WP-JRA-1.1) will accommodate their specific activities. The role of the engineering and design includes also the need to deliver appropriate adaptation decision mechanisms (captured, e.g., with the appropriate monitoring properties) and strategy decision mechanisms. Furthermore, in certain cases the problems revealed by the SBA monitoring may require re-design/re-engineering of the whole SBA.

3.3 End-to-End Quality Provision and SLA Conformance (WP-JRA-1.3)

The general research objective of work-package WP-JRA-1.3 ('End-to-End Quality Provision and SLA Conformance') is to define novel principles, techniques and methods for defining, negotiating and assuring end-to-end quality and conformance to quality contracts (e.g., Service Level Agreements) across the domain layers as well as across networks of service providers and consumers. With respect to these quality activities the following research objectives will be pursued:

- *Quality Definition*, which concerns the establishment of a model or language for the definition of quality contract terms, which is understood and shared by the contracting parties.
- *Quality negotiation*, which concerns the set of tasks required for defining an actual quality contract, based on the model or language for the definition of contract terms.
- *Quality assurance*, which concentrates novel techniques for run-time quality assurance and proactive assurance of quality.

3.3.1 Contribution to the Conceptual Research Framework

In the conceptual framework, this work-package will contribute to the cross-cutting issues on "Quality Definition, Negotiation and Assurance". This includes understanding how the mechanisms at the different domain layers can contribute to the techniques devised in this work-package. Also, this WP will

contribute knowledge on the quality definition, negotiation and assurance techniques to the other cross-cutting issues. In the case of "Engineering and Design", this will be exploited to better understand the SBA life-cycle and the issues related to the definition of quality requirements. In the case of "Adaptation and Monitoring", this will be used to understand how to pro-actively trigger adaptation.

3.3.2 Contribution to the Reference Life-Cycle

Figure 3.2 illustrates the contributions of this work-package to the S-Cube life-cycle. In particular, the

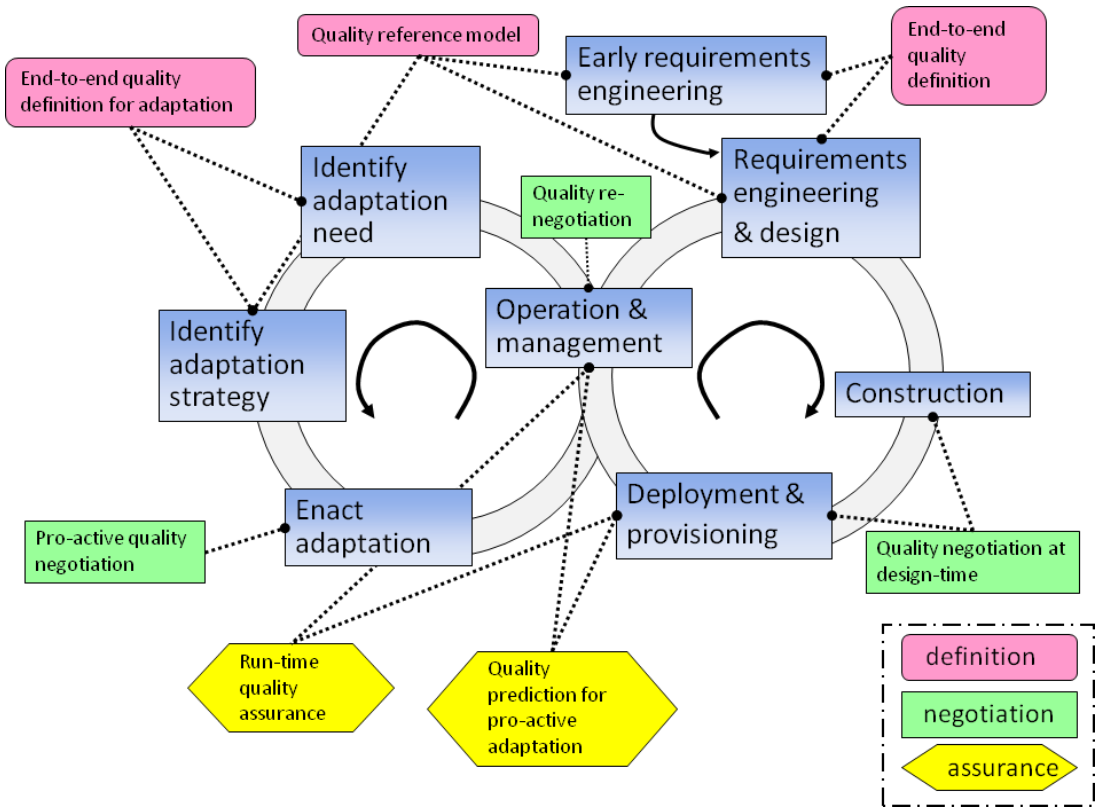

Figure 3.2: Reference life-cycle: contributions of JRA-WP-1.3

figure distinguishes the contributions related to quality definition (rounded boxes), quality negotiation (boxes), and to quality assurance (hexagons).

During *early requirements engineering*, the expected quality of the SBA is expressed in terms of high-level requirements. During this process, the quality reference model can be exploited in order to understand the different kinds of quality attributes relevant for a service-based application. In addition, if needed for the application domain, additional quality attributes can be introduced. These quality attributes should be used to extend the quality definition language, which will be used throughout the remaining stages in the life-cycle (see below).

During *requirements engineering*, the requirements to the SBA are detailed and quantified as far as possible by exploiting the quality definition language. Based on these detailed requirements, the requirements and specification for the constituent services are determined and expressed using the quality definition language. In addition, this step requires an understanding of the dependencies between the different quality attributes across layers and along the network of service providers and consumers. The quality definition language is key to defining the contract terms and thus needs to be understood and shared by the contracting parties. It should be noted that parts of these activities, especially defining a quality reference model and quality definition language, can occur outside the life-cycle of an individual

SBA (e.g., in the case that a quality definition language for a dedicated application domain is established by a multitude of stakeholders).

The *construction and realization* of a SBA may involve the selection of the contract partners (the service providers) among a set of potential partners, and the negotiation of the contract terms (e.g., SLAs) between the selected partner and the service consumer. To this end, techniques that are able to automatically establish contracts on quality attributes between service requestors (consumers) and providers provide support during the construction. Again, the quality definition language supports expressing those contracts.

More and more, service providers are selected during the *deployment* of the SBA or even during the *operation*. Thus, automated quality negotiation becomes important during the deployment and provisioning phase. In addition, already during deployment the quality of the running system (or parts thereof) can be assessed using run-time quality assurance techniques and a different deployment strategy could be chosen in case of problems. As an example, if the service of a chosen service provider does not respond, a different service provider could be chosen. At this stage, techniques similar to pro-active adaptation can thus be exploited. The quality of the running system is assessed against the expected quality, which is expressed using the quality definition language.

As mentioned above, service selection, negotiation and binding more and more occur during run-time, in the *operation and management* phase. Thus automatic negotiation and run-time quality assurance become key activities during operation. Run-time quality assurance includes monitoring, but also more advanced quality assurance techniques, which check the satisfaction of the quality contracts negotiated between a service provider and the service requestor. Again, those contracts are expressed using the quality definition language. When the application is running, SBAs need to timely respond to changes implied by the highly dynamic and flexible contexts of future SBAs and to promptly compensate for deviations in functionality or quality. Thus, anticipating adaptation needs and thereby enabling the pro-active self-adaptation of SBAs becomes important.

In the phase about the *identification of the adaptation needs*, once a deviation of the SBA or a change of its context has been observed, it must be decided whether and how to change the underlying SBA in order to remove the difference between the actual situation and the expected one. In this case, it can be helpful to express those adaptation needs by using the concepts of the quality definition language. In addition, and more importantly, identifying conflicting adaptation needs across the different domain layers can be supported by exploiting the quality reference model. Finally, the identification of adaptation needs can be faulty and thus the correctness of the adaptation needs to be assured at this stage.

In order to satisfy the adaptation needs, *adaptation strategies are defined and chosen*, which prescribe the possible ways to achieve the expected situation expressed in the adaptation needs. Like in the other stages, during which adaptation is a key concern, assuring the correctness of the adaptation becomes relevant.

In order to *enact the adaptation*, the adaptation strategies might call for selecting different service providers or might require selecting new service providers for newly introduced service invocations (e.g., when the business processes are adapted). In such a case, pro-active negotiation can ensure that the adaptation will be executed in a timely fashion and that the relevant contracts are established before the adapted application is deployed.

3.3.3 Contribution to the Logical Run-Time Architecture

This work-package contributes to the run-time architecture in two ways (see also Figure 3.6). Firstly, this WP will provide service negotiators (e.g., autonomous components provided as core services) that perform the negotiation process on behalf of the service consumers (requestors) and providers — see the “negotiation engine” in Figure 2.4. Secondly, this WP will contribute run-time quality assurance techniques and tools (realized in the form of core services — see “run-time QA engine” in Figure 2.4). Those include, besides others, techniques and tools for online testing and run-time model analysis.

3.3.4 Contribution to the Logical Design Environment

With respect to the design environment, the work-package will provide principles for defining, negotiating and assuring quality. However, at this stage the WP does not plan to provide actual design tools (e.g. quality modeling tools).

3.3.5 Relations with Other WPs

Several relationships between the objectives addressed in this work-package and the objectives addressed in other S-Cube research work-packages exist.

The quality reference model and the quality definition language will serve as input for requirements elicitation / engineering techniques and processes in WP-JRA-1.1, and monitoring techniques and mechanisms in WP-JRA-1.2. The quality prediction techniques will provide essential input to enable proactive adaptation in WP-JRA-1.2. Novel quality assurance techniques (like online testing) provide requirements for mechanisms at the Service Composition and Coordination (WP-JRA-2.2) and at the Service Infrastructure layers (WP-JRA-2.3). As an example, the definition of dedicated test interfaces for services could be such a mechanism. Run-time quality assurance techniques can be exploited as ”monitoring” techniques in WP-JRA-1.2 to trigger adaptation.

Codified user models and context knowledge from WP-JRA-1.1 will provide machine-accessible knowledge which can be exploited by the automated (and proactive) negotiation techniques in WP-JRA-1.3. Novel Web 2.0 technologies (e.g., reputation systems and search engines) from WP-JRA-2.3 provide concepts and mechanisms that can serve as input for quality prediction techniques. Quality attributes relevant fro the layers addressed by WP-JRA-2.1, WP-JRA-2.2 and WP-JRA-2.3 provide input to the quality reference model of WP-JRA-1.3. Local quality assurance mechanisms and techniques from WP-JRA-2.1, WP-JRA-2.3 and WP-JRA-2.3 (e.g., verification of service compositions) constitute input for consolidated and integrated quality assurance techniques. Design models (e.g., describing the behavior of the service-based application) from WP-JRA.1.1 provide input for model analysis techniques at run-time.

Jointly between WP-JRA-1.2 and WP-JRA-1.3, pro-active adaptation concepts and techniques will be addressed.

3.4 Business Process Management (WP-JRA-2.1)

The objectives of this work-package are twofold. Firstly, to develop concepts, mechanisms and techniques for analysis, rationalisation and modelling (design) of end-to-end processes in ASNs. Analysis includes not only the design-time elicitation of functional requirements and performance metrics for end-to-end processes, but also involves mining execution trails of choreographies to recover information about the run-time behaviour of processes and transactions.

Secondly, this work-package aims to develop monitoring, measurement and adaptation concepts, mechanisms and techniques for evolving processes and protocols within ASNs. The second research objective addresses run-time behaviour of business processes, and is particularly oriented towards developing and validating concepts, mechanisms and techniques for monitoring the execution of choreographies, measuring progress and performance of these processes against performance metrics, and pro-actively adapting them before process anomalies or errors occur.

3.4.1 Contribution to the Conceptual Research Framework

The business process management work-package will contribute to modelling, analysing and rationalising, designing, realizing, monitoring and managing end-to-end processes in agile service networks.

3.4.2 Contribution to the Reference Life-Cycle

Figure 3.3 illustrates the contributions of this work-package to the S-Cube life-cycle.

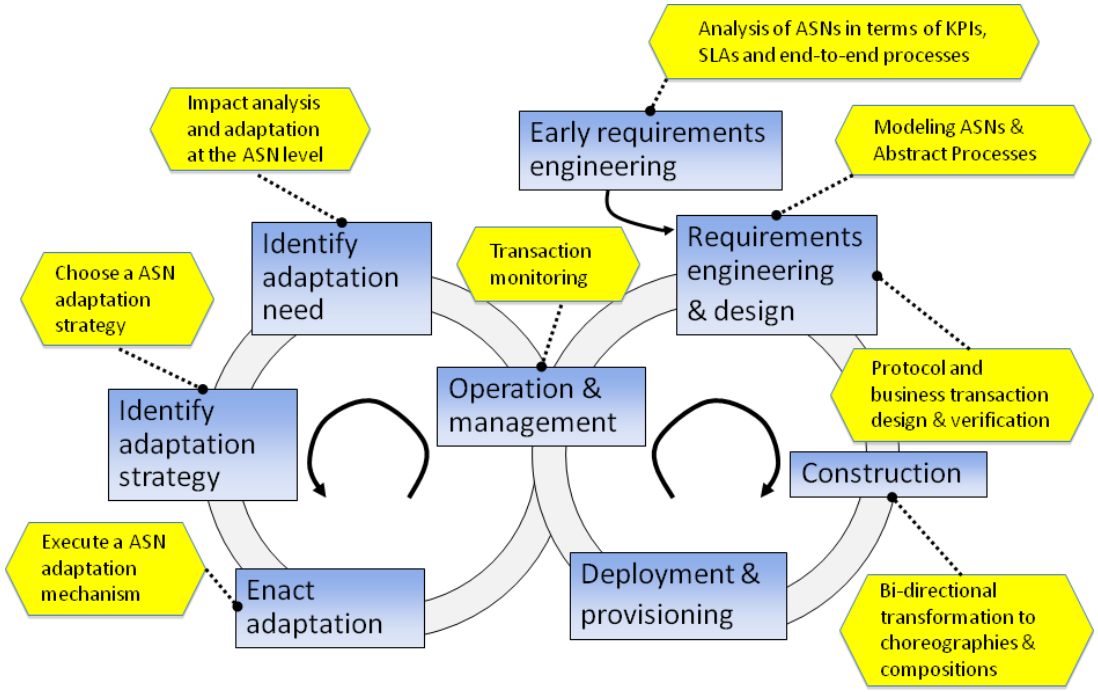

Figure 3.3: Reference life-cycle: contributions of JRA-WP-2.1

In particular, the SBA activity *Early Requirements Engineering* is comprised of BPM-related activities for analysing and rationalising agile service networks in terms of their functional behaviour, non-functional characteristics and process performance. In particular, during this step early requirements about end-to-end processes in the agile service network are elicited, correlated, and analysed.

Once initial requirements have been collected, they are further analysed and rationalised on the basis of models capturing both the structural characteristics and dynamic behaviour of the agile service network not only in terms of operational semantics, but also of business process performance. For this purpose, implementation-agnostic process models are derived from network models, and process performance metrics and key performance indicators are described. Next, and on the basis of these models, the ASN is further designed refining them into business protocols that govern interactions between network partners, some of which in fact may be developed as business transactions.

The prototypical construction phase in the SBA life-cycle includes BPM-related activities for transforming abstract process models into executable choreographies and orchestrations. It is important to note that the transformation of abstract into executable choreographies will occur at the interface between JRA-2.1 and JRA-2.2.

During execution, the business transaction monitor will collect, log and correlate process events emitted by transactional and non-transactional end-to-end business processes. Moreover, the transaction monitor will measure progress of an end-to-end process.

At this point it is worthwhile to reiterate the relation between end-to-end processes, protocols and transactions (see PO-JRA-2.1.1). An end-to-end business process behaves according to protocols (choreography), capturing the information and exchange requirements between trading partners (business conversations), identifying the timing, sequence and purpose of each business collaboration and information exchange. Some of the protocols can be realized as a business transaction that captures an atomic business process describing a trading interaction between possibly multiple parties that strive to accomplish an explicitly shared business objective as stipulated in a SLA. If a business transaction completes suc-

cessfully then each participant will have made consistent state changes, which, in aggregate, reflect the desired outcome of the multi-party business interaction. Process events relate to specific steps in the business transaction such as a step creating a purchase order, or a process event signaling the completion of a payment. In particular, the business transaction monitor will track process performance in terms of volumes and velocity of associated business transactions, as specified in SLAs.

Aside from monitoring and measuring business transactions, the transaction monitor is endowed with mechanisms that allow it to examine and optimise process performance, visualising potential performance anomalies and errors in end-to-end processes to end-users before they actually occur. Clearly, the business transaction monitor will draw upon process events emitted by process engines involved in the execution of a particular end-to-end process, pointing towards another close relation with JRA-2.2 research activities.

Adaptation requirements for end-to-end processes and associated business transactions may not only emerge from monitored execution of process instances, but also from changes in objectives of individual network partners, strategic reorientation of the network, and from changes that are exogenous to the network "per se", e.g., changing legislations. A full-blown analysis of adaptation requirements is part of the BPM variant to SBA life-cycle activity "identify adaptation needs". An essential component of this analysis entails impact analysis to understand how changes in partner- or network-level KPIs, multi-party SLAs, and/or business protocols and associated transactions propagates throughout the network, impacting the way in which partners may interact.

There does not exist a one-size-fits-all approach to accommodate adaptation requirements. Instead the "identify adaptation strategy" activity in the SBA reference life-cycle model, encompasses a broad arsenal of adaptation strategies to cater for specific adaptation needs. For example, adaptation of a partner level KPI (e.g., delivery time) poses completely different requirements regarding an adaptation strategy, than changing a network level KPI (e.g., end-to-end process cycle time) that may in fact involve renegotiation of SLAs or substitution of specific network partners.

Selection and subsequent execution of adaptation in the context of BPM of agile service networks is envisaged as a highly interactive process, involving several stakeholders, including business process analysts, process integrators and managers.

3.4.3 Contribution to the Logical Run-Time Architecture

This work-package contributes to the run-time architecture in the following ways (see also Figure 3.6). First, it will provide the business transaction monitor that allows to monitor and measure progress of end-to-end processes against SLAs and KPIs. In addition, the business transaction monitor will be endowed with facilities that accommodate continuous process optimisation of agile service networks, as well as pro-active change management. For this purpose, it will at least draw upon the following engines: monitoring, negotiation, adaptation and discovery engine.

3.4.4 Contribution to the Logical Design Environment

With respect to the design environment, the work-package will provide the concepts, languages and mechanisms needed for the analysis, optimisation and design of end-to-end processes and associated business protocols and business transactions. In addition, support for protocol verification will be donated to the design environment.

3.4.5 Relations with Other WPs

This work-package converges research efforts from virtually all other research work-packages for the purpose of defining, developing and continuously optimising end-to-end processes in agile service networks. In this section, we will outline the most important logical linkages, considering "vertical" con-

nections between this WP and the other two JRA-2 work-packages first, after which we will outline how work this WP is "horizontally" correlated with those in JRA-1.

Firstly, BPM logically sits on top of the WP-JRA-2.1 Service Composition layer. In particular, it defines abstract choreographies in terms of protocols models and transaction models, and states network- and partner-level KPIs and SLAs, that serve as the basis for PPMs and service-composition-level SLAs. In particular, the BPM work-package will work closely together with WP-JRA-2.2 in developing ontological models that cross-correlate KPIs, SLAs and PPMs, as well as transformation models for mapping ASN models into abstract choreographies, and abstract choreographies into executable choreographies and orchestrations. The other way around, BPM will rely on WP-JRA-2.1 for its process event model. In order to design and execute ASNs the BPM layer will not only draw upon the service composition layer, but also "directly" on the infrastructural layer that will provide discovery mechanisms as well as execution concepts and containers for atomic services, which should behave according to some SLA.

Analysis, rationalisation and design of agile service networks, and its constituents will be co-developed with WP-JRA-1.1. In particular, BPM may benefit from life-cycle models, and design principles and patterns developed in this work-package. The transaction monitor that makes up the heart-and-soul of the BPM work-package will be developed and tested in close collaboration with WP-JRA-1.2. In particular, the guidelines for monitoring and adaptation as well as management protocols will leverage the business transaction monitor. Lastly, transformation tools that are developed in this work-package may provide the baseline supporting refinement of ASN models into choreographies and orchestrations.

Lastly, WP-JRA-2.1 will draw upon the principles, methods and techniques for quality assurance. In this way, the BPM work-package will leverage definition and redefinition of SLAs as well as network-level KPIs, while performance optimisation will benefit from principles, methods and techniques for quality assurance. Vice-versa, the quality attributes relevant for these SLAs and KPIs will be taken into account in WP-JRA-1.3 for devising the end-to-end quality definition language.

3.5 Adaptable Coordinated Service Compositions (WP-JRA-2.2)

This work-package focuses on service compositions considering them with respect to different aspects. Service compositions are the service-based realizations of the business processes that applications must support, and, therefore, reside in the middle layer of SBAs. The main goal of this work-package is to establish the foundation of QoS-aware adaptable service compositions. These compositions adapt as reaction to changes in the QoS properties of SBAs. Such properties are specified on all three domain layers, which means that the service compositions and their adaptation are influenced by all these properties in a combination. The work in this work-package hence relies on inputs and requirements from both BPM and Service Infrastructure layers and addresses them throughout the whole life-cycle of service compositions, including modelling and verification, execution, monitoring and adaptation.

More precisely, the work-package will focus on the following objectives. First, it will elaborate on the formal models and languages for service compositions that are extended to explicitly model and reflect the QoS characteristics of the composition. This will also include the mechanisms for extracting the composition QoS information from the KPI models defined in BPM layer and from the service QoS defined in SI layer (and vice-versa). Second, the work-package will work on providing mechanisms for monitoring and analysis of QoS-aware service compositions (including monitoring of composition performance metrics and the dependency analysis between KPI, PPM, and QoS properties of SBAs). Third, the mechanisms for service composition adaptation will be investigated. The latter will include, in particular, mechanisms for composition fragmentation/defragmentation and for proactive composition adaptation.

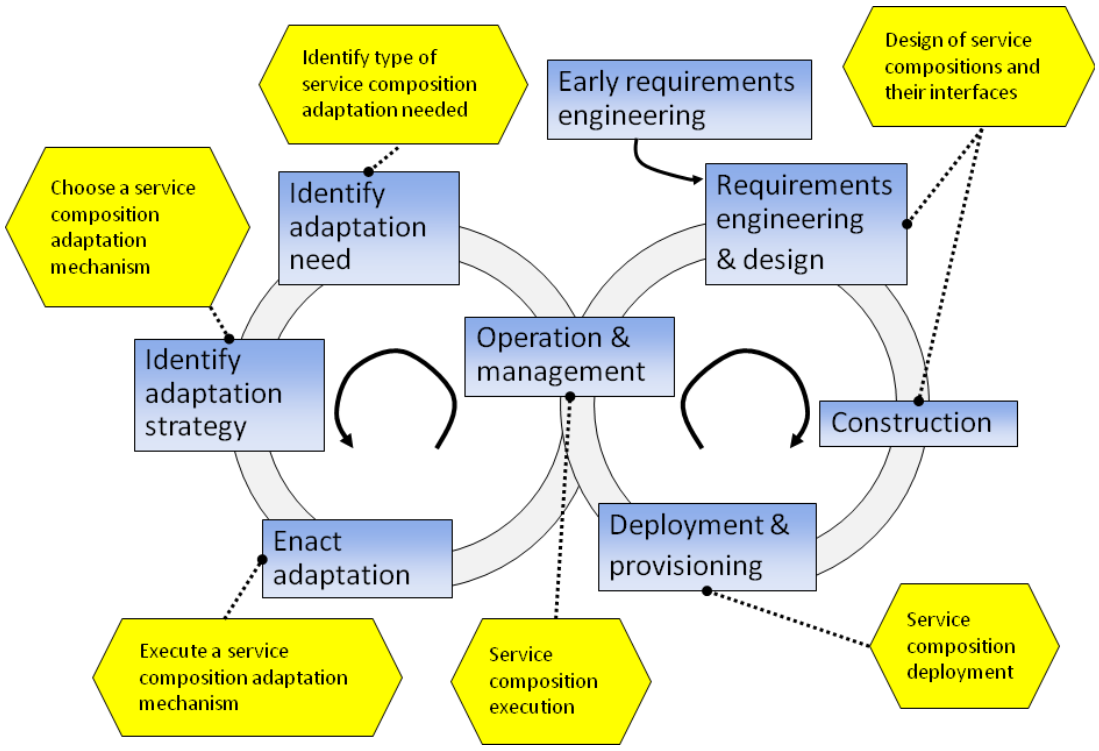

Figure 3.4: Reference life-cycle: contributions of JRA-WP-2.2

3.5.1 Contribution to the Conceptual Research Framework

As it follows from the description of the work-package objectives, the contribution of this work-package to the conceptual framework will mainly correspond to the service composition domain layer, providing the foundations for the realization of adaptable coordinated service compositions. As the work-package targets also the QoS aspects of those compositions, the WP will also contribute to the cross-cutting issues on "Quality Definition, Negotiation and Assurance" with the models of and the mechanisms for QoS properties of compositions. By providing the mechanisms for composition adaptation and monitoring, this work-package also contributes to the cross-cutting concern "Adaptation and Monitoring"

3.5.2 Contribution to the Reference Life-Cycle

Figure 3.4 illustrates the contributions of this work-package to the S-Cube life-cycle.

During the phases of *Requirements Engineering and Design* and *Construction* the service compositions are defined and created. This is performed either automatically (e.g., using automated service composition approaches), generated from another representation of the composition (e.g., applying model-driven methodologies) or developed from scratch by a service composition developer. As a result, an executable representation of the service composition as well as its interfaces are provided (e.g., in the form of BPEL and WSDL specifications). The actual mechanisms for creating such representations may also correspond to the combination of existing and novel approaches for creation of executable service compositions based on business-level modelling notations (e.g., BPMN). It may also include the transformation of a business-level modelling notation to the composition specification language.

During the *Deployment* phase the composition is configured and is made available for execution on an execution environment, such as BPEL engine. Depending on a particular execution platform, the configuration, deployment and enactment steps may be different and require specific mechanisms and tools.

During the *Operation and Management* phase the service composition is being executed. This phase

include also the monitoring of the compositions to track their status and the utilisation of underlying data and resources (services). The monitored information may be exploited for the purpose of run-time or "post-mortem" composition analysis, also using the historical information about previous composition executions. The analysis may involve process mining, fault patterns detection, resource consumption and throughput analysis.

The results of the analysis of monitored data corresponds to the *Identification of Adaptation Needs* phase, as it may trigger adaptation decisions towards modifications of the composition or its parts. Indeed, the results of the analysis can be used for the purpose of improving/optimising the quality of the service composition with respect to various criteria. Based on these results, different *Adaptation Strategies* can be identified. In particular, recommendation for modification of service composition in general may take place, thus requiring composition re-engineering, or the adaptation may concern only a specific instance (e.g., a concrete customer order, a specific item processing). Based on the strategy, specific composition *Adaptation Mechanisms* are identified and are enforced. The latter may range from the composition re-design, to run-time composition modification using appropriate techniques.

3.5.3 Contribution to the Logical Run-Time Architecture

This work-package contributes to the run-time architecture in the following ways (see also Figure 3.4. First, it provides the specific mechanisms for enactment and execution of adaptable and QoS-aware service compositions within the corresponding composition engines. Second, it contributes the composition-specific mechanisms and facilities for the monitoring and adaptation engines of the architecture.

3.5.4 Contribution to the Logical Design Environment

With respect to the design environment, the models, languages, and mechanisms provided by this work-package aim to support the definition, specification, and deployment of executable service compositions and of the corresponding process performance metrics, as well as of the composition-specific monitoring properties and the adaptation strategies and rules. In order to automate the process of derivation of such specification, the corresponding transformation solutions will contribute to the design environment (e.g., BPMN to BPEL and KPI to PPM transformations and vice-versa). Indeed, such models are subject of accurate analysis, and therefore the WP will contribute with the verification approaches that also address the QoS-specific information of service compositions.

3.5.5 Relations with Other WPs

The research activities performed by the work-package will be carried out in a strong collaboration with the other research WPs of the project.

With respect to the technology work-packages, the relations with the BPM work-package and the SI work-package are motivated by the overall SBA engineering approaches, including the top-down and bottom-up methodologies. In the former case, the abstract business process, choreography, and transaction models provide the input for the definition and realization of service compositions. The corresponding KPI and SLA models are necessary to derive the PPMs of service compositions. Analogously, the QoS characteristics, the information of service registries and execution platforms, which is available and is subject of research in SI work-package, provides an input to the bottom-up design of service compositions. With respect to the cross-cutting work-packages, the relations are described as follows. First, the service composition WP will provide isolated the languages and supporting techniques for the design and construction of service compositions to the "Engineering and Design" work-package, including languages for QoS-aware adaptable compositions, transformation techniques, and analysis techniques. Second, it will provide the isolated, composition-specific adaptation and monitoring mechanisms to be exploited in "Adaptation and Monitoring" WP for integrated, cross-layer adaptation and monitoring principles. Third, in collaboration with other technology WPs, the composition work-package will contribute

to the development of cross-layer orthogonal KPIs, SLAs, and QoS models based on the quality model investigated in WP-JRA-1.3.

3.6 Self-* Service Infrastructure and Service Discovery Support (WP-JRA-2.3)

The main goal of this work-package is to provide autonomic behaviour for services, and to maintain the autonomic behaviour of interoperable, cooperative services in a seamless and effective way.

The research objectives of this work-package are structured in three threads:

- *Service Discovery*. Current discovery mechanisms are not well prepared to deal with the huge number of Internet-scale service ecosystems. Novel discovery mechanisms must be able to deal with millions (or even billions) of services. Additionally, these discovery mechanisms need to consider new constraints, which are not prevalent today, such as Quality of Experience requirements and expectations (feedback) of users, geographical constraints, pricing and contractual issues, or invocability (not every service can be invoked and used by every client).

- *Service Registry*. With the advent of the Internet-scale service ecosystems a number of new challenges for the next generation of registries will arise. In such systems, fault tolerance and scalability of registries is of prominent importance. Autonomic registries need to be able to form loose federations, which are able to work 24/7, in spite of heavy load or faults. Additionally, a richer set of metadata (data describing services) is needed for services in such ecosystems, in order to capture novel aspects such as self-adaptation, user feedback evaluation, or Internet-scale process discovery.

- *Runtime Environment*. We will investigate an infrastructure that is able to adapt autonomously and dynamically to changing conditions. Such adaptation should be supported by past experience (learning), should be able to take into consideration a complex set of conditions and their correlations, to act proactively to avoid problems before they can occur, and to and have a long lasting, stabilizing effect.

3.6.1 Contribution to the Conceptual Research Framework

In the conceptual framework, this work-package will contribute to service infrastructure in the context of adaptation and self-adaptation of services as well as on discovery and registries.

3.6.2 Contribution to the Reference Life-Cycle

Figure 3.5 illustrates the contributions of this work-package to the S-Cube life-cycle. In particular, the reference life-cycle is extended with artifacts and actions related to the infrastructure. In addition to the artifacts related to adaptation, some infrastructure specific tools and actions are depicted.

Concerning adaptation, self-healing actions as well as reactive and proactive adaptations are enacted by adaptation engines included in the run-time infrastructure (see also the infrastructure view). Run-time monitoring has to be supported by monitors, some of which are included in the infrastructure (such as platform specific monitors for example).

In order to support deployment, deployment tools have to be provided. These tools use brokering and resource allocations techniques.

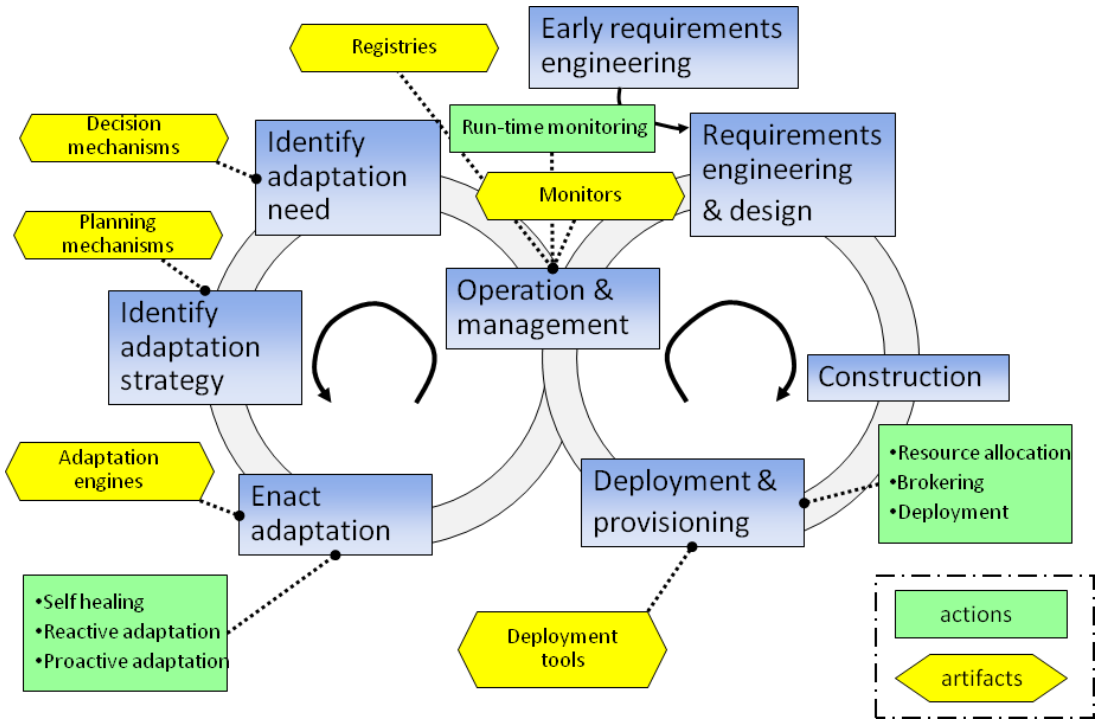

Figure 3.5: Reference life-cycle: contributions of JRA-WP-2.3

3.6.3 Contribution to the Logical Run-Time Architecture

This work-package contributes to the run-time architecture (see also Figure 3.6) by providing realization mechanisms for the adaptation engines, containers for adaptable and self-adaptable services as well as discovery and registry infrastructure.

3.6.4 Contribution to the Logical Design Environment

With respect to the design environment, this work-package will provide research on the run-time support for configuration, deployment and adaptation.

3.6.5 Relations with Other WPs

The WP-JRA-2.3 work-package provides discovery and registries concepts used by to WP-JRA-2.2. It is also related to WP-JRA-2.1 because it takes into account multilevel adaptation.

It enacts adaptation defined by WP-JRA-1.1 using the guidelines of WP-JRA-1.2 in order to contribute to achieving the end-to-end quality as defined, negotiated and assured by the techniques and methods in WP-JRA-1.3

3.7 Concluding Remarks

In this chapter, we have described the integration of the S-Cube research work-packages within the IRF by describing the contributions that these work-packages provide to the four IRF views introduced in Chapter 2. We note that this description of the contribution is just the first step towards the integration of the research work-packages: it has been performed to validate the consistency of the IRF with respect to the visions and expected results of the WPs, and it is reported in this deliverable to give evidence of this consistency.

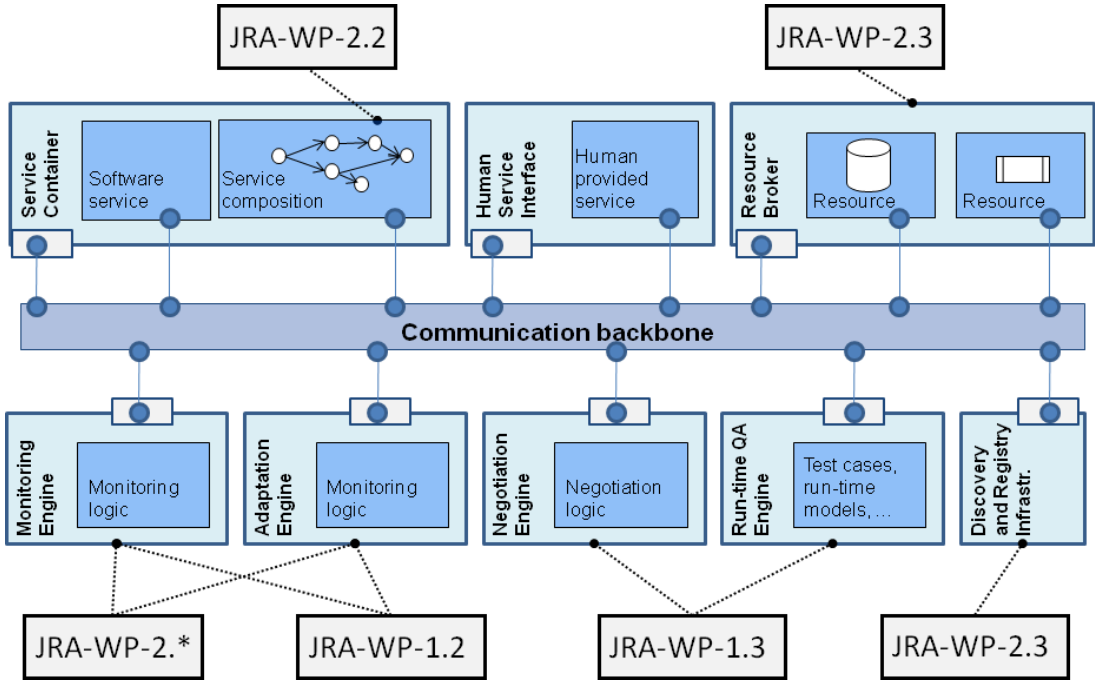

Figure 3.6: Contributions of the WPs to the run-time architecture

A stronger, consolidated integration of the research work-packages within the IRF is the objective of the next activities of the IA-3. In the next chapter, we will comment on the strategies that we have defined and that we are implementing for achieving this stronger integration.

Chapter 4

Evolution of the Integrated Research Framework

In the previous chapters, we have provided an initial definition of the S-Cube IRF, and we have shown how the research efforts undertaken by the different work-packages contribute to this framework. The work on the IRF is far from finished. Actually, what we provided in Chapter 2 is just the baseline for this framework, which needs to be refined and consolidated in order to become a first complete version of the IRF. In addition to this, we remark that the IRF cannot be a static object, as it has to evolve and grow throughout the whole life of S-Cube.

For these reasons, in this chapter we provide a description of the next steps that we will undertake for consolidating the definition of the IRF, as well as of the strategies that we foresee for its management and update.

In the following we discuss the three main lines of work that will be undertaken during the next year according to the S-Cube Description of Work [1], and we describe how we intend to achieve them.

4.1 First Complete Definition of the IRF

The purpose of this activity is to refine and consolidate the IRF baseline in order to produce a first complete version of the IRF. The outcomes of this consolidation activity will be reported in Deliverable CD-IA-3.1.2 “First Version of the Integration Framework”, due at Month 21.

The refinement of the IRF will be achieved by jointly refining the different views that we introduced in this document. More precisely:

- The *conceptual research framework* will be extended with the research challenges identified by the different work-packages, and with their (already achieved or foreseen) results. More precisely, the framework will be refined into a hierarchy, where the components of the framework depicted in Figure 2.1 constitute the higher level; the research challenges addressed by the different work-packages define the intermediate level of the hierarchy, and the results of the work-packages define the lower level. These results will include principles, methodologies, techniques, languages, notations, mechanisms, technologies and any other contribution that the research will identify and investigate.

- The *reference life-cycle* will be enriched by associating to each phase the relevant research results. In particular, the life-cycle will be annotated with the artifacts, the activities, the users and the roles that are relevant for the different phases in the life and evolution of a SBA. The outcome will be a consolidation and extension of the work reported in Chapter 3, in the sections on the contributions of the WPs to the life-cycle.

- Also the *logical run-time architecture* and the *logical run-time environment* will be extended by associating research results to the different elements that define these two views. In the case of the run-time architecture, this corresponds to identifying those research results that contribute technique, mechanisms, technologies which are relevant for the engines and containers that appear in Figure 2.4. The case of the design environment is similar, but in this case the research results that are relevant for the design of SBAs will be placed in the matrix represented in Figure 2.5. In addition to this, the extension of these two views will also consist in the definition of specific usage patters, i.e., specific ways of combining the different techniques in the run-time architecture, or specific ways of exploiting the techniques in the design environment.

We note that the consolidation work undertaken in this activity, as well as the validation activity that we will describe below, may possibly lead to the introduction of additional views, or to the revision of the views identified in this document, in case different perspectives will become useful for describing the IRF. In this case, these changes in the views will be taken into account in the definition of the IRF. However, we expect that most of the changes will happen within the identified views, which already offer a reasonable set of perspectives to the IRF.

In order to be able to manage the complexity associated to the definition of the IRF, and to its updates and revisions, we plan to move the IRF to the S-Cube portal; that is, the IRF will become a live object that can be explored, analysed, and extended on-line by the S-Cube partners.

4.2 First Validation of the IRF

During the next year, we will perform a first validation of the IRF. This activities aims at validating the IRF using the industrial case studies collected by work-package WP-IA-2.2, and reported in [9]. More precisely, this activity will start with the collection and definition of a set of scenarios based on these industrial case studies (Deliverable PO-IA-3.2.1 “Initial Definition of Validation Scenarios”, due at Month 17). Then, these scenarios will be applied to the IRF, in order to validate it and to identify possible improvements (Deliverable CD-IA-3.2.2 “Results of the First Validation”, due at Month 24).

The validation will be performed by mapping the scenarios on the different views of the IRF and by analysing how the scenarios can be represented in terms the “elements” which define these views. We will analyse in particular:

- Which parts of the conceptual research framework contribute to the scenarios, which research challenges are relevant, and which research results are exploited in the scenarios.
- Which phases of the life-cycle are relevant for the scenarios, which activities and artifacts associated to these phases are useful, and how are they exploited.
- Which elements of the run-time architecture and of the design environment, and which usage patterns of these elements are relevant for the scenarios.

The application of the IRF to the scenarios will also become part of the IRF and will become part of the base on which the research work-packages will perform their activities.

4.3 Analysis of User Patterns and Methodologies

This activity will also start during the second year of the project, even if the first deliverable will be issued only during the third year (Deliverable PO-IA-3.1.4 “Initial Definition of Users Patterns and Methodologies”, due at Month 27).

The activity aims at identifying the various types of users that may be involved in the engineering, in the adaptation, as well an in the usage of SBAs; its purpose is to characterise all these different types of users, in terms of their role and expertise, and to identify methodologies tailored to these users.

This activity will be undertaken in close collaboration with the other two activities. In particular, the usage patterns which will be identified for the elements in the design environment and in the run-time architecture will take into account these various types of users. Also the scenarios used in the validation will help identifying different types of users, and, vice-versa, the scenarios will be chosen to cover as many as those user types as possible.

Chapter 5

Conclusion

The main objective of this deliverable is to describe the baseline definition of the IRF that has been produced as outcome of the investigation of IA-3 during the first 6 months work[1].

This baseline definition is given by providing a set of views corresponding to different perspectives to the IRF. These perspectives have been introduced in order to be able to give a comprehensive overview of the research undertaken by S-Cube, and are essential to clarify the relations among the research efforts and to better understand responsibilities and mutual expectations of the different research work-packages. In the deliverable, we report both the definition of these views and a preliminary mapping of the work-package activities on these view.

In addition to the definition of the baseline of the IRF, this deliverable also defines clear interfaces between the different components of the research framework. These interfaces play a fundamental role in understanding synergies, relations and mutual expectations among the research activities in the different work-packages, as well as to avoid or solve gaps and overlaps.

Finally, the deliverable illustrates the activities that will be undertaken on the IRF during the second year of the project and the strategies that we intend to adopt in these activities.

[1] We recall that Activity IA-3 has started only at Month 7 of the S-Cube project.

Bibliography

[1] S-Cube – Description of Work, 2007. See `http://www.s-cube-network.eu`.

[2] S-Cube Deliverable PO-JRA-1.1.1: State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge, 2008.

[3] S-Cube Deliverable 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, 2008.

[4] S-Cube Deliverable PO-JRA-1.3.1: Survey of Quality Related Aspects Relevant for Service-based Applications, 2008.

[5] S-Cube Deliverable PO-JRA-2.1.1: Survey on Business Process Management, 2008.

[6] S-Cube Deliverable PO-JRA-2.2.1: Overview of the State of the Art in Composition and Coordination of Services, 2008.

[7] S-Cube Deliverable PO-JRA-2.3.1: Use Case Description and State-of-the-Art, 2008.

[8] S-Cube Deliverable CD-IA-1.1.1: Comprehensive overview of the state of the art on service-based systems, 2008.

[9] S-Cube Deliverable CD-IA-2.2.2: Collection of industrial best practices, scenarios and business cases, 2009.

Appendix A

Acronyms

Acronym	Description
ASN	Agile Service Network
A&M	Adaptation and Monitoring
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Modeling Notation
IA	Integration Activity
IRF	Integrated Research Framework
JRA	Joint Research Activity
KPI	Key Performance Indicator
PPM	Process Performance Measure
QoS	Quality of Service
SBA	Service Based Application
SOA	Service Oriented Architecture
SC&C	Service Composition and Coordination
SLA	Service Level Agreement
WP	Work-Package
WSDL	Web Service Description Language