Grant Agreement N° 215483

Title:	*Comprehensive overview of the state of the art on service-based systems*
Authors:	*Vasilios Andrikopoulos (Tilburg), Alea Fairchild (Tilburg), Willem-Jan van den Heuvel (Tilburg), Raman Kazhamiakin (FBK), Philipp Leitner (TUW), Andreas Metzger (UniDue), Zsolt Nemeth (SZTAKI), Elisabetta di Nitto (POLIMI), Mike P. Papazoglou (Tilburg), Barbara Pernici (POLIMI), Branimir Wetzstein (USTUTT)*
Editors:	*Vasilios Andrikopoulos, Willem-Jan van den Heuvel, and Mike P. Papazoglou (Tilburg)*
Reviewers:	*Marina Bitsaki (UOC), Manuel Carro (UPM), Schahram Dustdar (TUW)*
Identifier:	*Deliverable # CD-IA-1.1.1*
Type:	*Deliverable*
Version:	*1*
Date:	*25 September 2008*
Status:	*Final*
Class:	*External*

Management Summary

This deliverable describes the state-of-the-art in service-based systems in the form of a Knowledge Model (KM) for S-Cube, explaining its purpose and its individual components. It also identifies previous approaches from related EU projects and international activities that have resulted in the definition of a large body of concepts relating to software services research. These approaches are scrutinized, adapted and reused to the extend possible as part of the S-Cube KM. In addition, it summarizes and cross-correlates the major research findings of the state-of-the-art deliverables in S-Cube, and shows how they contribute towards building an initial version of the KM. Finally, it describes the connection of the S-Cube KM to a number of knowledge sources and knowledge-intensive activities within S-Cube and its usage by both internal and external users.

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

Members of the S-CUBE consortium:

University of Duisburg-Essen	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany

Published S-CUBE documents

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

PO-JRA-1.1.1: State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge

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

PO-JRA-1.3.1: Survey of quality related aspects relevant for SBAs

PO-JRA-2.1.1: State-of-the-art survey on Business Process Modelling and Management

PO-JRA-2.2.1: Overview of the state of the art in composition and coordination of services

Table of Contents

Table of illustrations

List of acronyms

BPEL	Business Process Execution Language
BPM	Business Process Management
BPMS	Business Process Management Suite
INTEROP	Interoperability Research for Networked Enterprises Applications and Software
KM	Knowledge Model
KPI	Key Performance Indicators
NESSI	Networked European Software & Services Initiative
NEXOF-RA	NEXOF Reference Architecture
QoS	Quality of Service
SBA	Service-Based Application
SeCSE	Service Centric System Engineering
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SSAI&E	Software Services Application Integration and Engineering
UML	Unified Modelling Language

1 Introduction

The *Knowledge Model (KM)*, which is part of Obj-1 of S-Cube's key objectives, is developed within work package WP-IA-1.1. The purpose of the knowledge model is "to synthesize and integrate diversified knowledge", thereby fulfilling the mission of innovation and integration of research agendas of disparate research groups in software services.

This work package will produce several deliverables during the project life span. The first deliverable (CD-IA-1.1.1) aims to develop an initial knowledge base of key terms used in the S-Cube research areas. It is a first attempt to cross-correlate terminology from different domains and to synthesize terms into an initial cohesive KM.

It combines research output from the first internal deliverables of the *Joint Research Activities (JRAs)*, with industry knowledge and existing EU project outcomes. Figure 1 illustrates the positioning of the KM activity in relation to the other activities of the project. More specifically:

- The KM interacts with the functional SBA layers of S-Cube (business process management, service composition and coordination, and service infrastructure) by using the JRA-1 and -2 activities as input for its development. More importantly, it furthermore allows for a common understanding and positioning of these activities into a holistic and consistent framework that is required for achieving integration and coordination within the project. This framework also permeates the other activities of the project, by acting as a foundation on which they can develop and interact with each other.
- The analysis and construction of the KM is performed in coordination with the integration framework for SBAs that is developed in IA-3 and which integrates, aligns, and coordinates the results of the JRAs.
- By providing links to knowledge models from other EU projects it facilitates the synthesizing and harmonization of research on SBAs across communities and initiatives.

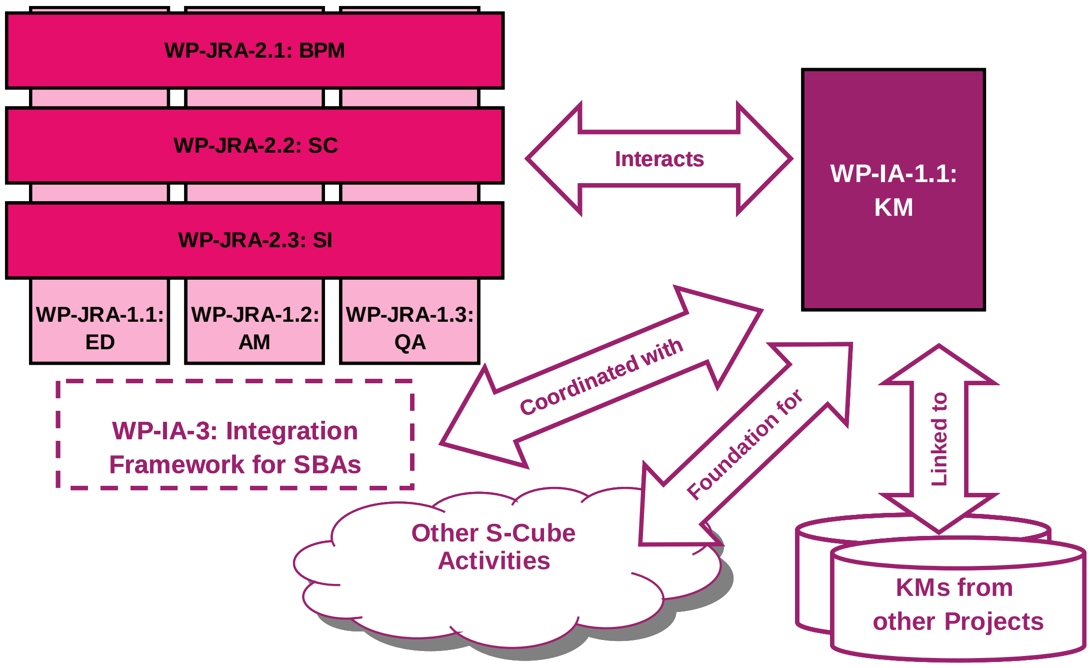


Figure 1: Integration of KM and other activities

This document contains a brief description of the purpose of this deliverable, a description of state-of-the-art JRA deliverables and of the terms extracted from these deliverables that have been incorporated in the initial version of the KM, its association with related EU and other activities (within the project), and further reports on the initial version and evolution of the KM. Additionally, in Appendix A we list all the initial key terms and their respective definitions and their interrelationships. Appendix B describes the initial list of expertise and competences found among partners and external experts.

2 Purpose of the Deliverable

The purpose of this deliverable is to provide a comprehensive overview of the state of the art on service-based systems (in the form of *Service-Based Applications – SBAs*), including an overview of the associations of the related research domains. In particular, the results from S-Cube's JRA-1 and JRA-2 activities are integrated and consolidated in this report. An additional objective of this report is to highlight the current and future structure of the S-Cube KM.

The S-Cube Knowledge Model will strive to have the form of a free, open-content "living" encyclopedia that will collect, cross-correlate and summarize services-related terminology and knowledge. This will help users navigate through a large body of knowledge related to all aspects of service-oriented research, associated methodologies, and support environments.

The initial Knowledge Model that is captured in this deliverable will be evaluated against KPIs, e.g., that each research area (e.g., BPM) is appropriately typified in terms of a limited number of key knowledge items (10 to 20).

3 Summary of the State of the Art JRA deliverables

In the following we summarize the first round of deliverables of the S-Cube research work packages (WP-JRA-*) that constitute the JRA activities and interrelate key terms and concepts from domains corresponding to the WP-JRA-* work packages and to each other. Figure 2 illustrates the relationships between the JRA work packages.

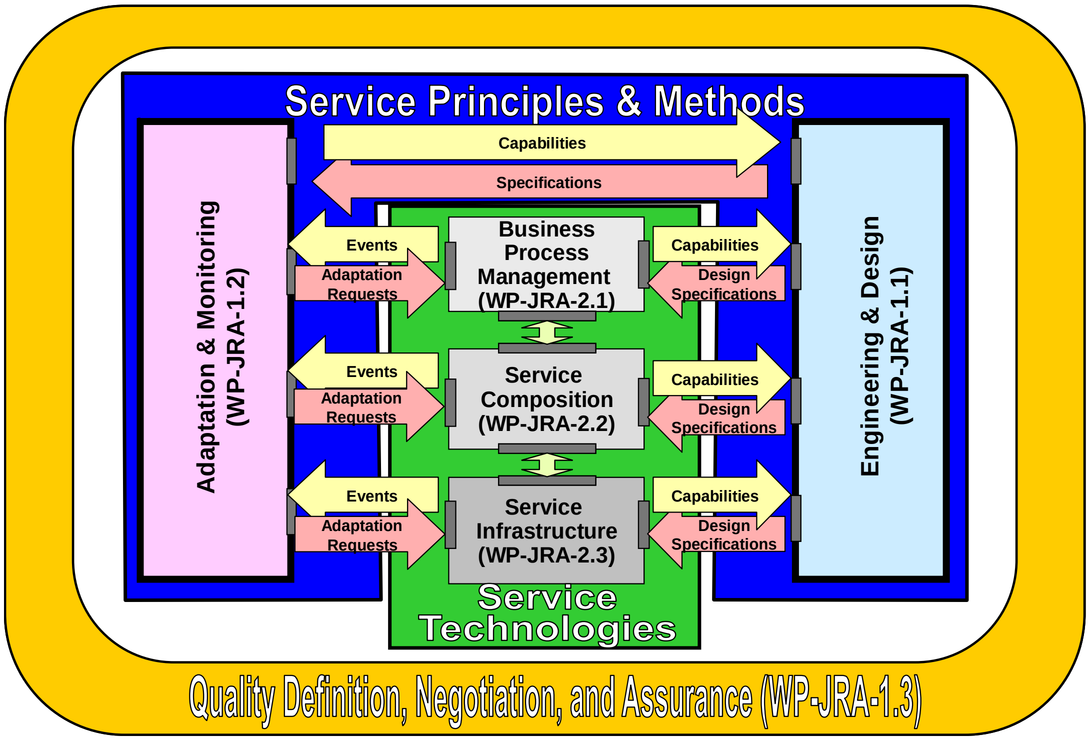

Service Principles & Methods
Capabilities
Specifications
Adaptation & Monitoring (WP-JRA-1.2)
Engineering & Design (WP-JRA-1.1)
Events
Adaptation Requests
Business Process Management (WP-JRA-2.1)
Service Composition (WP-JRA-2.2)
Service Infrastructure (WP-JRA-2.3)
Capabilities
Design Specifications
Service Technologies
Quality Definition, Negotiation, and Assurance (WP-JRA-1.3)

Figure 2: S-Cube Research Framework

Italicized and bold-typed words (e.g. ***term***) in the context of the JRA descriptions that follow in this section denote concepts that have been incorporated in the initial version of the KM (see section 5) which can be found in Appendix A.

3.1 *JRA-1: Objective and Coverage*

The objective of JRA-1 is to jointly develop the next generation of engineering and adaptation methodologies which, by combining different competences, take a holistic view and empower service composers, services providers as well as the European Citizens to compose and adjust service-based systems.

3.1.1 WP-JRA-1.1: Engineering and Design

Engineering ***Service-Based Applications (SBAs)*** is quite different from any other software engineering activity. These applications are built by gluing together some possibly already existing ***services*** that can be built and operated by third parties with which we may decide to establish a Service Level Agreement.

The possibility to reuse in SBAs existing services without even caring about the details needed to operate them is very attractive in principle. However, it changes the way applications are built and are operated. All services used to compose an application are not anymore under the direct control of the application developer and provider. This means that they could be modified or even temporarily or permanently dismissed without notice. Even what a service provider could perceive as an improvement of the service functionality could result in the impossibility for an SBA of using this service anymore. Think for instance at a service that is offering images at a certain resolution. Increasing the resolution is not necessarily a plus if the application that is using this service has to display the images on a PDA. This fact leads to the need for being able to replace a service with another at runtime in order to make the application resilient to any change happening on the side of the service.

Another interesting and promising aspect of Service-Based Applications is their ability to adapt to different execution contexts (we call these ***Adaptable Service-Based Applications***). Again, this requires the selection and possibly (re)placement of services at runtime.

While a number of specific techniques have been developed to support dynamic ***service discovery***, selection, and ***service binding***, a set of engineering methods aiming at developing service-based applications ready to perform ***self-adaptation*** at runtime is still missing. A method of this kind should support the designer in the definition of the application-dependent logic of a service composition as well as of the policies that are actuated when runtime changes are needed. Such policies aim at supporting the evolution of Service-Based Applications still keeping them under control. More in detail, a ***design for adaptation*** approach should address the following problems:

- How to identify and represent relevant changes or situations that the target system should adapt to. This amounts to the classification and description of the dynamic aspects of the target system (such as ***context***-related aspects, available services and their metrics, classes and parameters of failures), and the relevant values and ranges (e.g., acceptable bounds of QoS parameters, values of context parameters, etc).
- How to drive the modification of the application when the necessity to adapt is detected. In this case the specification of ***adaptation requirements and objectives*** and ***adaptation strategies*** should be addressed. The corresponding notations and languages should enable the integrator to describe the desired situations (e.g., “good” state of the system in case of ***self-healing*** systems) or even adaptation actions (e.g., re-execution or ***re-binding*** command) at high level of abstraction.

Services and SBAs can require interaction with human users taking part in the business process enabled by the service, or imparting human intelligence to the relevant services (e.g. the Amazon Mechanical Turk ***Web service***). This interaction is currently supported in initiatives such as BPEL4People, an extension to the BPEL language that defines human tasks through Web Service Human Task (WS-Human Task) and describes them as activities with WS-BPEL Extension for People. We explore whether the ***Human-Computer Interaction (HCI)*** literature can further contribute to the integration of human actors in SBAs by providing richer, more contextual descriptions and models than those currently available. We further examine whether an explicit integration of current HCI knowledge into approaches to the engineering of SBAs would support their development.

WP-JRA-1.1 shows many relationships with the other S-Cube work packages since it gathers techniques and approaches from them and tries to harmonize them into an engineering method. More specifically:

- WP-JRA-1.2 offers the *adaptation approaches* that will be incorporated as part of the engineering method.
- WP-JRA-1.3 offers support for the *quality assurance aspects*. These are of paramount importance for any software engineering method and are particularly critical when we consider the development of SBAs where component services are out of our control and therefore require the definition of SLAs as well as of mechanisms that support both their pre-execution evaluation and their runtime monitoring.
- JRA-2 activities offer features and approaches at the various layers of the SBA stack, ranging from BPM to the execution infrastructure. All of these will have to be considered in the definition of the engineering method for adaptable SBAs. Moreover, the HCI findings, as well as the methodological aspects that will be identified in WP-JRA-1.1 will provide hints and suggestions to the other work packages for the selection of the approaches that best suit the S-Cube engineering vision.

3.1.2 WP-JRA-1.2: Adaptation and Monitoring

Modern Service-Based Applications are required to operate and evolve in highly dynamic environments, being able to adequately react to various changes in these environments. This makes ***adaptation***, i.e., the process of modifying an SBA in order to satisfy new requirements and to fit new situations dictated by the environment on the basis of adaptation strategies designed by the system integrator, one of the key aspects of SBAs functionality. The range of possible changes that require application of adaptation includes changes in the infrastructural layer of the application, where the quality of service changes or the service becomes unavailable; changes of the (hybrid) application context and location, where the SBA should be able to replace one service by another possibly having other properties and parameters; changes of the user types, preferences, and constraints that require application customization and personalization as a means to adapt the application behavior to a particular user; changes in the functionalities provided by the component services that require modifying the way in which services are composed and coordinated; changes in the ways the SBA is being used and managed by its consumers, which should lead to changes in the application requirements and the business processes implementing them.

Depending on the type of the changes in SBA and its environment, adaptation may have different forms. In particular,

- ***optimization*** is the modification of an application to make some aspects of it work more efficient or use fewer resources;
- ***recovery (repair)*** is restoring an application after failing to perform one or more of its functions to fully satisfactory execution by any means other than replacement of the entire application;
- ***QoS-based adaptation*** refers to changes in quality-of-service parameters of an SBA;
- ***evolution*** is a long-term history of continuous modification of SBA after its deployment;
- ***mediation*** is an activity in which a neutral third party, the mediator, assists two or more parties in order to help them achieve an agreement on a matter of common interest.

In order to detect critical changes, adaptation strongly relies on the presence of monitoring mechanisms and facilities. With ***monitoring*** we mean a process of collecting and reporting relevant information about the execution and evolution of SBA. Such information, namely ***monitoring events***, represents evolution of SBA and changes in the environment. These events define the "What?" dimension of the monitoring process: they are used to indicate whether the SBA is executed and evolves in a normal mode, whether there are some deviations or even violations of the desired or expected functionality. ***Monitoring mechanisms*** are the tools and facilities for continuous observing and detecting relevant monitoring events; they identify the "How?" dimension of the monitoring process.

As monitoring events occur at different functional layers of SBAs, different monitoring types exist:

- For the BPM domain (WP-JRA-2.1), ***Business Activity Monitoring*** provides near real-time monitoring of ***business activities***, measurement of ***Key Performance Indicators (KPIs)***, their presentation in dashboards, and automatic and proactive notification in case of deviations.
- For the service composition domain (WP-JRA-2.2), ***Monitoring in Service Compositions*** refers to checking whether certain predefined properties over the composition model are satisfied when the composition is executed.
- For the Service Infrastructure domain (WP-JRA-2.3), ***Monitoring in Grid*** refers to scalable high performance monitoring on a large distributed computational Grid. It aims to tackle monitoring of generic middleware services and application-specific information and data transfer. There is also a need for the cross-cutting monitoring techniques and methodologies for SBAs.

The events identified during the monitoring process carry the information about potential changes that the system and / or the underlying platform should perform in order to adapt to a new situation. The relation between the monitoring events and the changes of SBA are defined by the ***adaptation requirements and objectives***: requirements and needs that SBA should achieve in reaction to critical changes and events. This may include the need for the fault recovery, QoS properties optimization, requirement to mediate service interfaces, etc.

The adaptation requirements are realized through ***adaptation strategies***. Adaptation strategies describe the ways to achieve the requirements given the ***adaptation mechanisms***, i.e., tools provided by the underlying platform (independently) in different functional layers of SBA.

The adaptation and monitoring research domain is strongly related with other research areas. Engineering and Design domain (WP-JRA-1.1) and provides approaches and notations for defining:

- monitoring specifications (*monitoring languages*), which describe SBA execution properties, context, events, and changes;
- engineering principles for monitoring (*design for monitoring*), where monitoring architectures and realization patterns are identified;
- adaptation specifications (*adaptation languages*), which describe the adaptation strategies, application variability models;
- engineering principles for adaptation (*design for adaptation*) with the corresponding architectures and patters.

Domains of realization mechanisms from different functional SBA layers, in turn, provide the corresponding infrastructures, tools and techniques. This include, in particular,

- *Monitoring infrastructure*, such as various logging mechanisms;
- *Monitoring tools and techniques*, such as process/data mining approaches;
- *Adaptation infrastructure*, e.g., automated service discovery and binding frameworks;
- *Adaptation tools and techniques*, e.g., automated service composition approaches.

3.1.3 WP-JRA-1.3: Quality Definition, Negotiation and Assurance

Quality dimensions (a.k.a. ***quality attributes, quality parameters, or quality characteristics***) express "non-functional" capabilities or requirements of SBAs. By grouping a set of relevant quality dimensions, a service can be defined in terms of its quality characteristics, which state how "well" the service works.

Quality related aspects relevant for SBAs cover a broad field of research, including work on *Quality of Service (QoS)* description and modeling, QoS negotiation, as well as quality assurance:

QoS description and modeling: We argue that obtaining a holistic taxonomy of QoS aspects in service-based applications represents a very challenging task. Relevant quality aspects may differ (i) according to the services' application domain, e.g., domain specific quality for B2B applications is different than quality of services intended for retail customers, and (ii) on the basis of the type of services involved in the system, e.g., specifying quality for grid services in a scientific computing application is different than describing quality for application level (or business level) services in a virtual travel agency scenario. Therefore, we first introduced a high level taxonomy of ***QoS dimensions***, built on the basis of previous work on the quality of service components, and we then focused on the review of meta-models that can be used to describe service QoS in different scenarios and on the languages proposed, both in academia and industry, to describe instances of such meta-models.

For what concerns QoS modeling, our survey has uncovered the lack of a well established and standard QoS model for services. In addition, most of the research approaches do not offer a rich, extensible, and semantic QoS model. The differences between these formalisms limit the fulfillment of the vision of automated and precise QoS-based service matchmaking and selection and QoS-aware service composition. Hence, we argue that a first research direction concerns the development of a standard QoS model, attempting to describe every relevant aspect of QoS for services, including metrics, units, measurement functions and directives, constraints, value types, etc. In addition, this QoS model should encompass a rich set of domain-dependent and global quality dimensions and should be extensible so as to allow the addition of new quality dimensions when it is needed (e.g., for a new application domain). Last but not least, this standard QoS model should be semantically enriched in order to be machine-processable and machine interpretable.

QoS negotiation: QoS contracts are parts of Service Level Agreements (SLAs), which deal with statements about the ***QoS levels*** on which the service requestor and the providers reach an agreement. This survey does not focus on other aspects of the contracts, i.e., parties' identification, legal obligations, or contract non-fulfillment penalties, which are also aspects covered in SLAs or general contracts. For what concerns contract establishment, we focus on ***QoS negotiation*** as the primarily means to automatically establish contracts on QoS between service requestors and providers. The review of relevant literature in the field of QoS negotiation in service-based applications shows high dependability on the classification of QoS aspects (see discussion above). In particular, we discern between three main different approaches to QoS negotiation in service-based applications. First, we review relevant work in the field of application level QoS dimensions negotiation. In this case, QoS negotiation is usually performed adopting state of the art techniques mostly drawn from the agent-based computing literature. Besides the need to establish QoS contracts that can be monitored at runtime, QoS negotiation often becomes a means to efficiently select services on the basis of non-functional requirements. Secondly, trust and security of service access and usage – which are among QoS aspects that could be negotiated at application level – show peculiar characteristics. While, in fact, performance related or domain specific QoS negotiation represents a classical multi-attribute negotiation problem, trust and privacy negotiation requires a different paradigm, oriented toward the managed and secure disclosing of credentials among parties that provide and use a service. Third, QoS negotiation and resource allocation represents a fundamental issue in grid services management. While the description of QoS aspects in grid services is not critical, the focus, in this last case, is on negotiation protocols to achieve efficient and manageable resource allocation.

We identify two main streams for research on service QoS negotiation. First, we underline the issue of automated SLA establishment in service compositions. The review shows that most of the current work in this field concerns the negotiation between a service consumer and a service provider or the set of providers of functionally equivalent services. Proposals for managing complex 1-to-*N* negotiation with services involved in the same service composition are still at their infancy and need further development. Second, research efforts should be devoted to the analysis of innovative negotiation strategies explicitly tailored to the requirements of service-based applications.

Quality assurance: To assure the desired quality of a service-based application, two complementary strategies can be employed: constructive and analytical quality assurance. Where the goal of ***constructive quality assurance*** is to prevent the introduction of faults (or defects) while the artifacts are created (in the sense of 'correctness by construction'), the goal of ***analytical quality assurance*** is to uncover faults in the artifacts after they have been created. Three major classes of approaches for analytical quality assurance in service-based applications exist: (i) ***Testing***, the goal of which is to (systematically) execute services or service-based applications with predefined inputs in order to uncover failures, (ii) ***Monitoring***, which observes services or service-based applications as well as their context during their current execution, (iii) ***Static Analysis***, the aim of which is to systematically examine (without execution) an artifact (e.g., a service specification) in order to determine certain properties or to ascertain that some predefined properties are met.

Due to the fact that many development decisions can only be taken during run-time (e.g., deciding on which of the services to actually bind to the application), in future research there will be a strong need for quality assurance techniques that can be applied while the service-based application is in operation. Currently, typically monitoring techniques are employed for assuring the quality of an application during its operation. The problem with monitoring is that it only checks the current execution. It does not allow to pro-actively uncover faults which are introduced, e.g. due to a change in the application, if they are not leading to a failure in the current execution. One important requirement of those "on-line" techniques, however, is that their overhead and costs should not become so high that they become unpractical for this reason. Finally, as (self-)adaptation of service-based applications becomes an essential characteristic, there is a strong need to assure that the adaptation of a service-based application behaves as expected. This requires specific testing and analysis techniques to verify the adaptation behaviour.

In a dynamic business scenario, the contract life-cycle should be automated as much as possible, in order to allow organizations to dynamically change service providers (business partners) or to re-negotiate SLAs. That requires that QoS aspects need to be checked during the operation, e.g., by monitoring the ***QoS characteristics***, in order to determine whether the new service provider meets the desired QoS or whether there is a need for re-negotiating the SLAs. As the survey in PO-JRA-1.3.1 has revealed, there are only few and isolated research contributions on assuring QoS aspects. There is thus a strong need for novel techniques and methods that address QoS characteristics in a comprehensive and end-to-end fashion across all layers of a service-based application. In addition, approaches that consider the context of a service-based application and its impact on QoS are needed in order to pave the way towards context-aware service-based application.

The relationships of quality aspects for service-based applications with other S-Cube JRA areas are the following:

- *WP-JRA-1.1:* While WP-JRA-1.3 focuses on analytical quality assurance techniques and methods, constructive quality assurance approaches are covered in WP-JRA-1.1 (specifically, process models and design methods).
- *WP-JRA-1.2:* Monitoring (which is one aspect in WP-JRA-1.2) is one important means of quality assurance. One goal of WP-JRA-1.3 is to understand the dependencies and synergies between monitoring and other analytical quality assurance techniques and methods. Thus, WP-JRA-1.3 also considers monitoring but with this specific goal in mind.
- *WP-JRA-2.1 and WP-JRA-2.2:* For what concerns the verification of service compositions, in WP-JRA-1.3 work which is more related with non-functional properties (QoS) is relevant, while other approaches which lean more towards checking correctness of service compositions are examined in WP-JRA-2.1 and WP-JRA-2.2.
- *WP-JRA-2.3: WP*-JRA-1.3 will provide a formal framework and a taxonomy as a basis for developing quality monitoring, negotiation, and assurance mechanisms on the infrastructural level in WP-JRA-2.3.

3.2 JRA-2: Objective and Coverage

The objective of JRA-2 is to jointly design and develop realisation mechanism for the next generation of service-based systems which support the engineering and adaptation at the business process, the service composition and the infrastructure layers.

3.2.1 WP-JRA-2.1: Business Process Management

Business Process Management (BPM) has recently emerged as both a management principle and a suite of software technologies focusing on management of the lifecycle of a business process ranging from business goals reflected in the definition of business processes, to the deployment, execution, measurement, analysis, change, and redeployment of these business processes. A prime constituent of BPM entails a ***business process*** that is defined as a process used to achieve a well-defined business outcome and is completed according to a set of procedures. A business process may span organizations and may typically involve both people and systems.

A ***workflow*** is a technology for realizing of inter- and intra-enterprise (business) process. Workflow constructs make it possible to implement business process aspects like logical decision points, sequential as wells as parallel work routs, as well as managing of exceptional situations.

Business processes are governed and constrained by ***business rules*** that define the business terms and facts (structural assertions) as well as the constraints underlying the business behavior (action assertions). Business rules represent core ***business policies***. Business policies capture the nature of an enterprise's business model and define the conditions that must be met in order to move to the next stage of the process.

A ***value chain*** is the largest possible business process in an organization. The value chain is decomposed into a set of core business processes and support processes necessary to produce a service, product or product line. These core business processes are subdivided into ***activities***. The near real-time monitoring of business activities, the measurement of ***Key Performance Indicators (KPIs)***, their presentation in dashboards, and the automatic and proactive notification in case of deviations constitutes the notion of ***Business Activity Monitoring (BAM)***.

In environments involving business collaborations, business processes are increasingly complex and integrated both within internal corporate business functions and across the external supply chain. In such environments there is a clear need for advanced business applications to coordinate multiple services into a multi-step ***business transaction***. This requires that several Web service operations or processes attain transactional properties reflecting business semantics, which are to be treated as a single logical (atomic) unit of work that can be performed as part of a business transaction.

Business processes and business transactions communicate by employing business protocols. A ***business protocol*** specifies the possible message exchange sequences (conversations) that are supported by the service to achieve a business goal. Business protocols are not executable, but protocols can be specified using BPEL (or any of the many other formalisms developed for this purpose) defining in a reusable manner the way to process the workflow specific data.

Business Process Management Suites (BPMS), which provide an integrated set of tools to model, design, simulate and deploy business processes and process- or transaction based applications, delivering greater degrees of process management delivery, include the following building blocks:

1. ***Business Process Modeling***: Process models are needed to help business managers and analysts understand actual processes and enable them, by visualization and simulation, to propose improvements. In particular, business process modeling relates to design methodologies (WP-JRA-1.1).
2. ***Business Process Integration***: Connecting the process elements so that they can seamlessly exchange information to achieve business goals. For applications this means using APIs and messaging. For people this means creating a workspace on the desktop or fulfilling their part of the process. Business process integration relates to service and process segments synthesized from distributed geographic locations as described in WP-JRA-2.2 (Service Composition).
3. ***Business Process Execution***: Once the design and modeling exercise is accomplished, the process is deployed and executed within a BPM execution engine. The BPM execution engine executes process instances by delegating work to humans and automated applications as specified in the process model. The execution environment employs composition languages such as BPEL (WP-JRA-2.2) and relies on an appropriate service infrastructure (WP-JRA-2.3).
4. ***Business Process Analysis, Monitoring and Auditing***: This involves providing graphical administrative tools that illustrate processes that are in progress, processes that are completed, and integrate business metrics and key performance indicators with process descriptions. Audit trails and process history/reporting information is automatically maintained and available for further use. Business process activities are logged and monitored as described in WP-JRA-1.2.
5. ***Business Process Measurement***: Managing processes first requires aggregating process data in business-oriented metrics such as key performance indicators and balanced scorecards. If the process is "out of bounds" or SLAs are not being met, the next step is to recalibrate it by

reconfiguring resources or modifying business rules – dynamically and "on the fly." Business process activities are measured according to KPI as described in WP-JRA-1.2 and WP-JRA-1.3.

6. ***Business Process Optimization***: Optimization means process improvement, which should be an ongoing activity. This item involves optimizing process flows of all sizes, crossing any application, company boundary and connects process design and process maintenance.

The next-generation of service-enabled BPM will serve as a means of developing mission-critical applications based on strategic technology capable of creating and executing cross-enterprise collaborative business processes and business-aware transactions, so that organizations can deploy, monitor, and continuously update cross-enterprise functions within a mixed environment of people, content, and systems. Such collaborative, complex end-to-end service interactions give raise to the concept of ***Agile Service Networks***.

3.2.2 WP-JRA-2.2: Service Composition

WP-JRA-2.2 covers the service composition domain. In Service Oriented Computing, services are often described as autonomous software components that can be described, published, and discovered in a platform-neutral and interoperable way. They perform functions ranging from simple atomic requests to executing complex business processes. An important property of service orientation is the possibility to combine existing services to create service compositions.

Service composition allows defining more complex applications by reusing existing services at increasing levels of abstraction. One can distinguish between several service composition models. ***Service orchestration*** creates a composite service by describing how it interacts with existing services, including the business logic and execution semantics of these interactions. The so created service orchestration is again exposed as a service and can be orchestrated by other services in a recursive manner. In the context of Web services, WS-BPEL is the standard language for representing Web service orchestrations. While a service orchestration specifies the interactions with services and the business logic from the point of view of a single partner, ***service choreography*** focuses on describing the publicly visible message exchanges between several partner Web services. In addition to orchestration and choreography, ***service coordination***, and ***service wiring***, can also be considered as types of service compositions. All of these composition model types have different purposes, but can often be combined together.

For supporting the lifecycle of service compositions several aspects have to be addressed. *Synthesis* of service compositions deals with creation of service compositions which can happen both at design-time and run-time. In this context, *model-driven*, *QoS-aware*, and *automated service composition* are three relevant research subdomains. After the creation of a service composition, *verification* techniques are needed for verifying the composition against certain properties, such as whether it is deadlock-free. After deployment of a service composition to the corresponding middleware, which for service orchestrations is typically a process engine in combination with a service bus, the composition is *executed*. At runtime, the composition can be *adapted* by for example rebinding other services, if a predefined service fails. Finally, *monitoring* of service compositions is performed either for run-time verification or to measure performance metrics of service compositions.

The S-Cube deliverable PO-JRA-2.2.1 "Overview of the State of the Art in Composition and Coordination of Services" presents the state-of-the-art in the service composition domain. It is structured as follows:

- *Service composition models:* The first part of the report deals with service composition models. It presents and compares approaches to service orchestration, choreography, coordination and wiring. In addition, semantic WS composition approaches are discussed.

- *Service composition approaches focusing on synthesis:* In the second part of the report, three service composition synthesis approaches are presented: (i) ***Model-driven service composition*** copes with generating service composition models from more abstract models. The approaches deal predominantly with the transformation of abstract business process models to executable orchestration models, such as executable BPEL processes. (ii) ***Automated service composition*** aims at selecting services and creating a service composition based on an abstract goal without human intervention. In this context, existing approaches based on workflow techniques and AI planning are presented and compared. (iii) Finally, ***QoS-based service composition*** attempts to create a service composition that adheres to local and global QoS constraints.

- *Verification of service compositions:* The third chapter presents verification techniques for service composition. Most techniques are based on model checking, many of them focusing on BPEL processes checking properties such as safety and liveness.

In the final part of the report research challenges in the service composition domain are identified.

The relevance of the research domain to the other five JRA areas:

- Service composition focuses mechanisms supporting the lifecycle of service composition which are related to techniques from WP-JRA-1.1 and which will be integrated into the overall methodology.
- WP-JRA-2.2 provides techniques and mechanisms for the service composition layer, which are supported by principles, techniques and methodologies for monitoring and adaptation of SBAs on all three layers (WP-JRA-1.2).
- Concerning the QoS aspect, JRA-2.2 focuses on QoS-aware service composition, which is based on guaranteeing local and global quality constraints in service compositions and builds on WP-JRA-1.3. This work package focuses on specification, verification and negotiation of QoS and SLAs, and quality assurance for SBAs.
- WP-JRA-2.2 provides the groundwork for WP-JRA-2.1 to deal with the transformation of service networks and business process models to service compositions.
- Service composition relies on a service infrastructure (WP-JRA-2.3), which is situated in the layer below service compositions, and relates to such issues as service discovery, dynamic binding and invocation. This layer provides middleware and functionalities used by service compositions.

3.2.3 WP-JRA-2.3: Service Infrastructure

The property commonly referred as ***self-**** is a collection of one or more reflexive properties expressing the ability of changing some aspects of the working behaviour of a computing entity. In most cases self-* can be translated to some of ***self-configuration***, ***self-optimization***, ***self-healing***, ***self-protection*** but there are various further self-* properties. The aim of ***autonomic*** computing is to incorporate most of the self-* properties into computing systems.

The driving force behind developing self-* functionalities is identified as the complexity of integrating large-scale heterogeneous computing systems into a single one. The most notable trends nowadays in this direction are ***Grid*** and mobile computing. Accordingly, the self-* chapter is organised around these two computing platforms. The design space of self-* services is extensive, hence, instead of a taxonomy-like presentation, some representative cases are chosen to introduce and demonstrate various aspects of self-* issues, their use cases and the best practices. More specifically:

- Some high-level models that are governing self-management are introduced. They are usually based on some analogy to emulate self-* behavior found in nature.
- Issues related to self-optimisation and self-healing is introduced in a numerical simulation example in grid computing.
- Autonomic brokering plays a crucial role in establishing self-* behavior in Grid computing. Grids must be instrumented with flexible, decentralized decision making capabilities, whereas

clients need a robust distributed computing platform that allows them to discover, acquire, federate and manage the capabilities necessary to execute their decisions.

- Dynamic self-deployment of services that is a novel and unique technique. It is an example for supporting different phases of service lifecycles by self-* capabilities, bootstrapping in the certain example.
- Dynamic adaptation, further self-optimisation, self-healing and self-configuration issues are introduced in a mobile environment via multimedia and transactional examples.

Software systems built on top of Service-Oriented Architectures (SOA) use a triangle of the three operations "publish", "find" and "bind" in order to decouple the participants in the system.

The problem of "finding" services is usually referred to as ***Service Discovery***. Service Discovery is defined as the act of locating a machine-understandable description of a Web Service that may have been previously unknown and that meets certain criteria. Traditionally, SOA-based systems rely on centralized discovery mechanisms, such as centralized ***service registry*** standards (e.g., UDDI or the ebXML registry, ebXMLRR) or centralized indices. By contrast, there are many research approaches that rely on decentralized storage, mainly to get away from the single point of failure problem that centralized registries pose, and to increase scalability. Such approaches include distributed UDDI clouds, P2P-based service registries (e.g., PWSD or the P2P registry proposed by Schmidt and Parashar). Distributed approaches also include agent-based solutions, such as DASD (DAML Agents for Service Discovery). Another interesting research question in Service Discovery is how queries can be formulated, i.e., what retrieval mechanisms are used. The usual approach here is to use keyword matching, using results from the Information Retrieval community. A variant of keyword-based searching is the Vector Space Search Engine presented by Platzer and Dustdar. More advanced than these approaches is signature-based matching. These approaches use the (WSDL) interfaces of Web services to inform the search. Prominent examples include Woogle, SPRanker, WSQBE and the Service Discovery Framework presented by Zisman et al. Semantics-based approaches use Semantic Web Services technologies for Service Discovery, such as OWL-S (or its predecessor DAML-S), SAWSDL or WSMO. Context-based approaches, which include the query context (such as location or user preferences) in the discovery process can be seen as an extension to the other matchmaking approaches (rather than a replacement). Much work in this area has been carried out by Zisman, Spanoudakis et al. Another orthogonal topic is QoS-based service discovery, i.e., finding services that comply to certain non-functional constraints. Lately, languages are being proposed to enhance registries with QoS data, or to model QoS (QML). Work in the area of QoS-based WS matchmaking has been carried out by Kritikos and Plexousakis (OWL-Q).

The scope of ***Dynamic Binding*** is somewhat different to Service Binding: Dynamic Binding assumes that a service has already been discovered, and now has to be connected to. An early approach to dynamic binding has been developed within the SeCSE project (WS-Binder). Similarly scoped was the JOpera framework, that also incorporated some dynamic binding ideas. Recently, the VRESCo project proposed a new infrastructure for service-oriented computing, that also assumes that dynamic binding is the foundation on which loosely-coupled service-based systems are built. A topic closely related to Dynamic Binding is ***Dynamic Invocation***. Dynamic Invocation considers that it is not so easy to dynamically invoke recently discovered, previously unknown Web services. Apache WSIF is the well-established service framework used for Dynamic Invocation. However, recent toolkits such as DAIOS advance the concepts of Dynamic Invocation to include topics such as RESTful Web services and ***service mediation*** between incompatible services.

Seemingly, self-* services and service registry, discovery and binding are very loosely coupled but in an operational service based infrastructure they are related and may rely on each other. Most self-* functionalities, like self-healing and self-optimization for instance, obviously need information about the available services that can be obtained by service discovery. Self-deployment is both relying on the information of service registries and maintaining them. Also, version management is essential in the presence of self-deployment, from a practical point of view they are belonging to the same scope of problems. Self-* brokering in grids, that can be a potential solution for optimisation, fault tolerance and adaptation, is strongly related to service discovery mechanisms. On the other hand, in certain cases service registry and discovery techniques may require some degree of self-* behavior to ensure fault tolerance or context awareness.

Service infrastructures are the technical foundation on which research within the other JRAs is based:

- Adaptation and monitoring of services (WP-JRA-1.2) demands for a strong infrastructural background (e.g., service monitoring is only feasible in a well-defined end-to-end service environment). Additionally, service adaptation has some clear requirements towards discovery of alternative services.
- The same is also true for adaptable service compositions as developed within WP-JRA-2.2. Adapting compositions demands for a well-defined service infrastructure and mature service discovery mechanisms. Furthermore, WP-JRA-1.1 develops methodologies and techniques which are used to implement service infrastructures themselves.
- Monitoring is also an essential functionality for establishing self-* services. Also, self-* and adaptability has much in common. Monitoring and adaptability are covered in depth in depth in WP-JRA-1.2.
- The life-cycle of services (that is also relevant for autonomic services) is introduced in more details in WP-JRA-1.1.

4 Relation of S-Cube's Knowledge Model to other EU Projects and Initiatives

In the following we shall describe the relationship of S-Cube's Knowledge Model in connection to several European projects and initiatives, including NESSI's NEXOF-RA, BEinGRID, SeCSE, and INTEROP. This section also briefly relates S-Cube's KM to the OASIS Reference Model for SOAs. It must be noted that several related terms from these models have been reused and adapted for the purposes of S-Cube's Knowledge Model. They have also influenced the structure of the S-Cube KM development approach.

4.1 NESSI and NEXOF-RA

One of the primary means for S-Cube to establish an intense and long-lasting collaboration with industry (cf. Obj-5) will be through participating in the European Technology Platform NESSI (Networked European Software & Services Initiative). NESSI (www.nessi-europe.eu) aims at providing a unified view for European research in Services Architectures and Software Infrastructures. NESSI currently has 22 partners and over 200 members from major European ICT companies and research institutions.

S-Cube's relation to NESSI is two-fold:

- The research agenda of S-Cube and the strategic research agenda (SRA) of NESSI (see Figure 3) will be coordinated in order to maximise synergy effects between long-term research challenges and shorter term research challenges.
- Key individuals of NESSI serve on S-Cube's Industrial Advisory Board (IAB) and thereby support the relationship with NESSI, e.g., the NESSI standardisation committee.

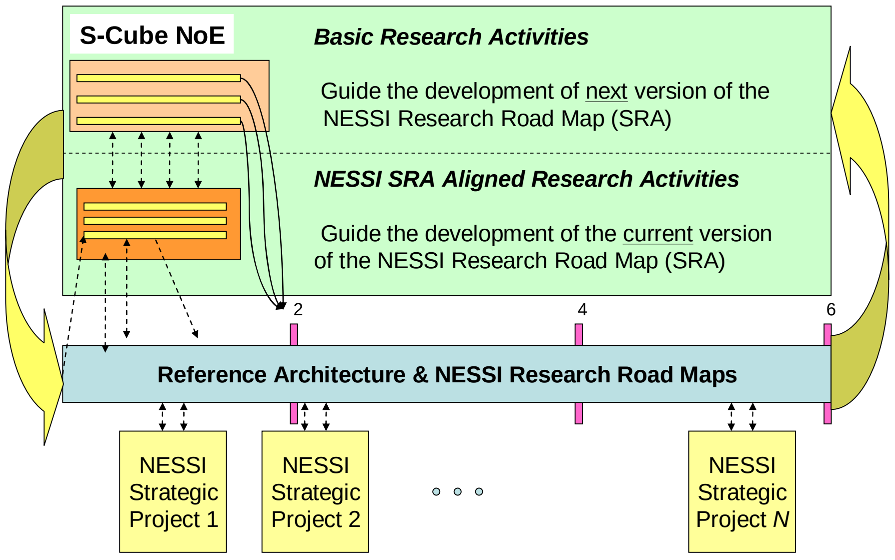

S-Cube NoE
Basic Research Activities
Guide the development of next version of the NESSI Research Road Map (SRA)
NESSI SRA Aligned Research Activities
Guide the development of the current version of the NESSI Research Road Map (SRA)
2
4
6
Reference Architecture & NESSI Research Road Maps
NESSI Strategic Project 1
NESSI Strategic Project 2
NESSI Strategic Project N

Figure 3: Overview of Relation of S-Cube to NESSI

The NEXOF-RA (NEXOF Reference Architecture) project, funded under objective 1.2 in FP7, is NESSI's first step in the process of building a generic open platform for creating and delivering applications enabling the creation of service based ecosystems where service providers and third parties can easily collaborate. NEXOF-RA main results will be the Reference Architecture for NEXOF, a proof of concept to validate this architecture and a roadmap for the adoption of NEXOF as a whole.

The following differences between the NEXOF-RA Glossary and the S-Cube Knowledge Model have been identified:

- The NEXOF-RA Glossary has more restricted focus regarding terms, focusing on architectural and infrastructural issues (e.g. platforms), while the S-Cube Knowledge Model is broader in focus, encompassing for example BPM and HCI concepts;
- The NEXOF-RA follows a dictionary (glossary) approach whereas the S-Cube Knowledge Model is based on an encyclopaedic (knowledge model) approach, which provides users with mental cues to navigate through a vast space of knowledge and terms from different domains, e.g., software engineering, business process management, grid computing and so on.

NEXOF-RA Glossary contains key definitions with respect to the NEXOF-RA reference model (conceptual model of the architecture) and the NEXOF-RA reference architecture (concrete specification). Thus, aligning the work on the NEXOF-RA Glossary and the S-Cube Knowledge Model will also foster progress for what concerns architectural issues.

4.2 BEinGRID and Gridipedia

The repository of Gridipedia, a part of the BEinGRID (Business Experiments in Grid) IST project, is populated with Grid software components and solutions that are designed to meet common business requirements. Information on these components can be found in the Technical Solutions section of their Web portal (www.gridipedia.eu). Released components can be downloaded via the Component Access page on their portal.

Other contents include:

- Information on how the components relate to business needs.
- Design patterns providing solutions to common Grid problems.

- An explanation of what the Grid is, including a classification of Grids and a glossary.
- Information on how Grid technology is being applied to business today, including:
    - Details of companies that are investing in or developing Grid technology.
    - The emerging business models.
    - Case studies investigating the applicability of Grid technology to business.
    - Legal issues related to Grid computing.
- Information about the leading Grid middleware.

In comparison, the S-Cube approach will have a more dynamic, web- and encyclopedia-based approach that is broader in content coverage, focusing on semantic associations between concepts, approaches and methodologies.

4.3 SeCSE and it's conceptual model

The Service Centric Systems Engineering (SeCSE) conceptual model was defined in order to encourage the usage of a common terminology and promote a common understanding on the concepts, entities, processes and facts involved in the various activities of the SeCSE project.

Since its first version released at the end of the first year of the project (2007), the model has represented a reference for all partners collaborating to the activities of the project. In addition to that, the model has shown to be suitable for a number of purposes and contexts beyond those of the SeCSE project. As an example, the model is being used to classify and compare different languages and technologies available for the composition of services. The model has also been adopted by the European Commission for classifying the different research initiatives funded by EC in the area of the Service Oriented Systems and as conceptual model in a number of other research projects on SOA, such as the Plastic research project (Providing dependabLe and Adaptive Service Technology for pervasive Information and Communication), a project funded under grant number 026955 by the IST Program of the European Commission as a STREP.

A key difference is that SeCSE has a limited scope and its conceptual model is of static nature based on UML diagrams, while S-Cube aims for dynamic cross-correlation of service-related knowledge.

4.4 INTEROP and KMap

The INTEROP Knowledge Map (KMap) aimed at drawing a picture of the status of research in interoperability and to maintain this picture for future (see Figure 4).

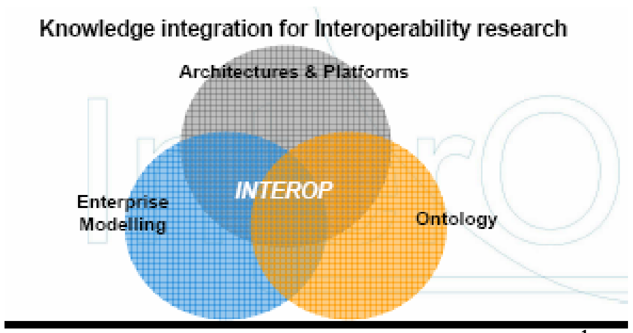

Knowledge integration for Interoperability research
Architectures & Platforms
Enterprise Modelling
INTEROP
Ontology

Figure 4: Knowledge Intergration with INTEROP[1]

[1] Presentation Sept 2006 - INTEROP: an original approach to solve Enterprise Interoperability problems by combining Ontology, Enterprise Modelling and IT", from Guy DOUMEINGTS, University Bordeaux 1

The first main objective of the KMap, as of the S-Cube KM as well, was to support a periodic diagnostic of the extent of research collaboration and coordination among INTEROP partners. Their work went further to include a repository for papers that were used to index terms for KMap. A similar approach will be exercised by the S-Cube KM.

Several artifacts were produced by the INTEROP project that are of interest with respect to KM development for the S-Cube project. More specifically it contains the following components:

- *Terms, relationships and definitions extraction process from texts*: This process is supported by tooling that aims to analyze textual data (research papers, web pages, and other documents) for extracting glossary data. The extraction itself identifies potential interoperability terms, definitions of these terms and existing relationships among terms (mainly kind-of relationships), based on statistical and lexical analysis of these texts. This would be very interesting for S-Cube documentation, if the right parameters can be programmed in, and it can search multiple formats of data, including flat files.
- *Glossary (terms) voting system*: This system aims to allow interoperability experts to collaboratively select terms to be included in an interoperability glossary by voting on their relevance.
- *Definition validation tool*: This tool aims to allow interoperability experts to collaboratively select, define and refine terms definitions extracted from reference interoperability texts (see extraction process described above) in order to be included in an interoperability glossary. This appears to be too exclusive, not inclusive, for use in S-Cube.
- *Taxonomy edition and visualization tool*: This tool aims to graphically display and edit a taxonomy (or glossary). This includes, for instance, functionalities for taxonomy reorganization by moving sub-parts of the taxonomy to more appropriate places. Interesting for S-Cube, but it depends going forward on what tool the repository sits in, as the current repository may not be able to include this kind of tool.
- *Interoperability Explicit Knowledge Repository:* The objective of this repository is to allow the storage of relevant knowledge about interoperability. This knowledge will take the form of papers, publications and journals, research methodologies, literature reviews, tutorials, etc. These elements must be classified with some taxonomy of interoperability and be searchable. This is the intention of S-Cube's KM within its own software services research area as it matures.
- *KMap system:* This system aims at describing the competencies, results and collaboration of INTEROP researchers in the domain of interoperability. All data of the KMap are classified according to an interoperability "classification framework" derived from the INTEROP Glossary. The tool will allow performing a diagnostic of the current and past research on interoperability by providing support for queries (such as "identify research domains in which insufficient research effort is devoted or in which there is insufficient collaboration").
- Protégé KMap prototype (WP1): This prototype was developed in order to start collecting KMap data while the KMap system is being developed. The data collected through this tool will be imported in the KMap system when it will be put in operation. This is a must-have going forward for S-Cube.

S-Cube plans to study the different components of INTEROP and reuse a similar tooling approach wherever possible.

4.5 OASIS and SOA-RM

The goal of the OASIS Reference Model (SOA-RM) was to define the essence of Service Oriented Architecture (SOA) and emerge with a vocabulary and a common understanding of SOA.

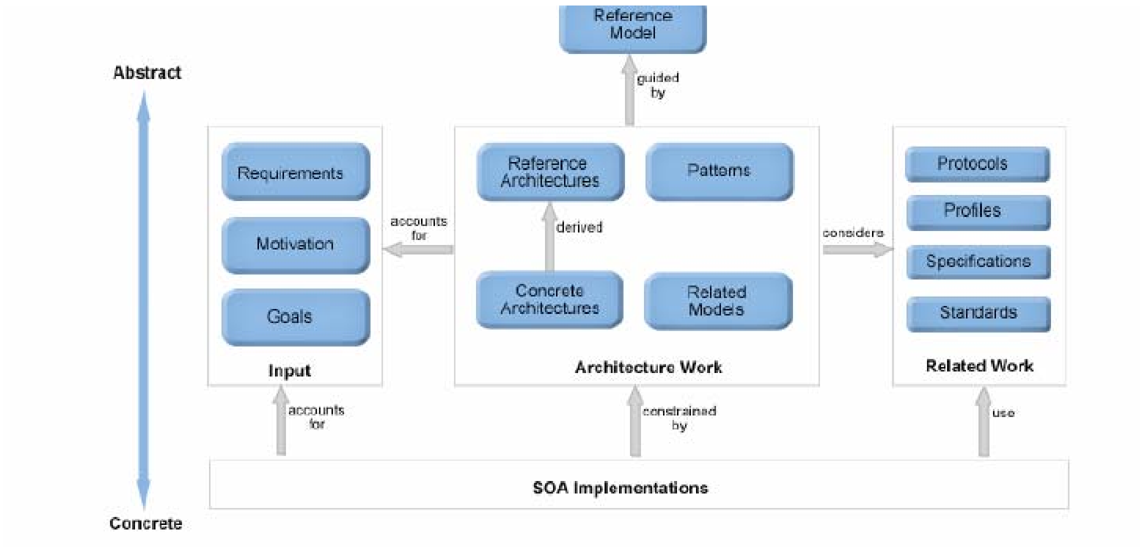


Figure 5: OASIS SOA-RM

The purpose of their reference model (see Figure 5) is to provide a common conceptual framework that can be used consistently across and between different implementations and is of particular use in modeling specific solutions.

In summary, the OASIS model is not as complex and interactive as the S-Cube KM, but it contains several concepts which could be beneficial for inclusion.

5 Evolution of the Knowledge Model

S-Cube's Knowledge Model aims at mapping and synthesizing diverse concepts and knowledge from partners in different research domains in the network. It identifies research gaps, determines the research issues that are of importance for the next generation services technologies, harmonizes research results and, in general, enables the streamlining of the research activities of the six joint research areas and their respective domains.

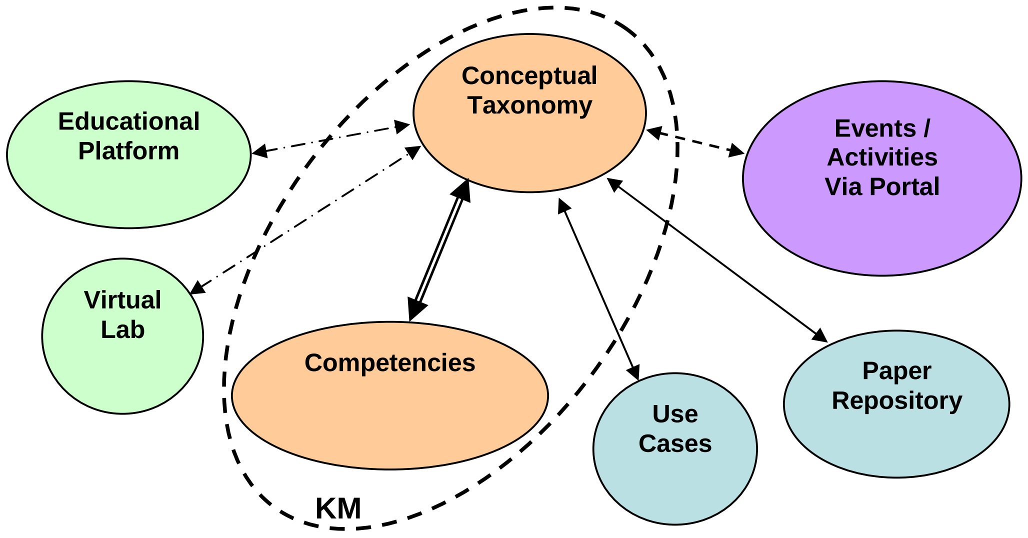


Figure 6: The KM and other knowledge sources and knowledge-intensive activities

The purpose of the KM is to synthesize and integrate diversified knowledge in the form of a free, open-content "living" encyclopedia accessed through the web. This will help users to negotiate a large body of knowledge by providing them with mental cues to navigate through a vast space of knowledge and terms from different domains related to all aspects of service-oriented research and associated methodologies and support environments. The S-Cube KM is a dynamic, interactive encyclopedia-based approach focusing on semantic associations between concepts, approaches and methodologies. Two critical components of the KM can be identified (see Figure 6):

- *Conceptual taxonomy:* containing (possibly overlapping) key terms for each research domain (as identified for example by section 3), together with their definitions and relationships with other terms.
- *Competencies*: representing key contributors and experts relating to research domains in terms of research output (volume and relevance) from both inside and outside the S-Cube consortium.

Figure 6 also describes the relationship of the S-Cube Knowledge Model with other knowledge sources and knowledge-intensive activities. In particular, the S-Cube KM relates to the following knowledge sources in S-Cube:

- *Use Cases*: representative use case scenarios that are useful for the wider community to appreciate the S-Cube approach. For instance, scenarios could be drawn from WP-JRA-2.3 to describe situations where run-time service adaptation is required and illustrate how it is handled.
- *Paper Repository*: a centralized database of software services related research papers that were produced within the context of S-Cube and assembled as part of the community outreach activity (WP-SoE-1.2).

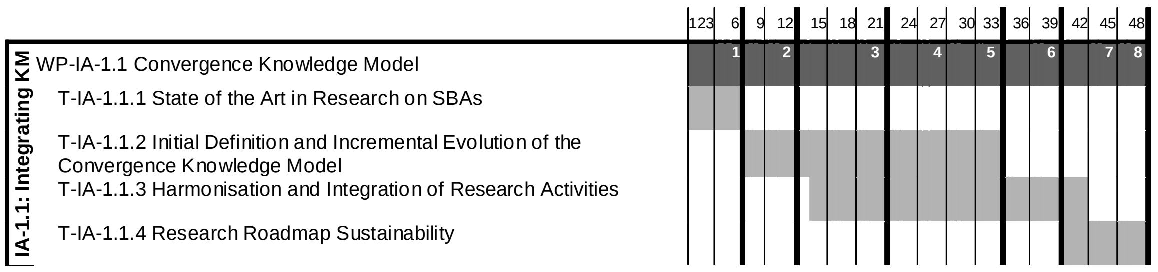
In addition to these sources, the knowledge models produced by the other EU projects discussed in section 4, and especially the ones that are part of the SSAI FP7 collaboration activity (i.e., NEXOF-RA and Gridipedia) can also be valuable sources of knowledge. An investigatory effort in linking terms from the initial version of the KM presented below with these knowledge models has been performed, but due to the scope and time limitations involved, the results of this exercise have been deemed too premature to be discussed here.

Furthermore, the S-Cube KM relates to the following knowledge intensive activities in S-Cube

- *Educational Platform* (the *Virtual Campus* and *Joint PhD programs* activities in WP-SoE-1.1): it is expected that students and professionals will use the KM as a source for course material, its unambiguous definitions, the offered cross-correlation of terms from different disciplines, and have access to case studies and publications relating to specific domains or terms.
- *Virtual Lab* (WP-IA-1.2 output): apart from used as a source of knowledge for the virtual lab, the KM will serve as a basis of integrating different tools and implementations by experimenting with the different use cases and applying the methodologies stipulated in the KM.

Focus in this deliverable is in the establishing of an initial content for the conceptual taxonomy and the competencies, and provide a mapping between them as the stepping stone for establishing the S-Cube KM.

The following chart illustrates the phases of development of the knowledge in S-Cube.



5.1 *Initial Knowledge Model and Competencies*

So far in the early phases of the KM development for the S-Cube project, the emphasis has been on establishing the initial scope and requirements for the KM (e.g. with regards to collecting the initial terms and definitions, and competencies for each domain). This process is supported by a Content Management System (CMS), which is linked to the project's Web portal. In this way, external users as well as project participants will have access to a large body of knowledge relating to services research.

In the first phases of the KM development, an initial set of terms and competencies was established for each of the relevant research domains, represented by respective JRA activities within the S-Cube project. These have been added to the KM section of the S-Cube Web Portal. Research domain terms constitute important concepts within a research domain. Competencies represent key contributors to a research domain in terms of research output (volume and relevance) from both inside and outside the S-Cube consortium.

In line with the goal of this deliverable to establish an initial set of terms and competencies for the different S-Cube research domains, the scope of each domain and the identification of relevant terms and competencies were determined through consultation of the domain experts within the S-Cube consortium. In future development of the KM the possibility of the usage of quantitative metrics will be explored in order to facilitate further elaboration and refinement of the KM.

The methodology adopted for the selection and definition of initial terms and competencies was iterative in nature comprising the following steps:

1. A definition of the research domain was established based on the State of the Art document (SoTA) developed for the corresponding JRA.
2. An initial set of terms to be inserted in the KM was selected by isolating the important concepts within the research domain definition and their relationships.
3. An initial set of Competencies was established for each research domain.

Through iteration of these steps the initial sets of terms and competencies was then further elaborated and refined by contrasting them against the research domain definition. Also, where needed, the research domain definition was adjusted to better reflect the content of the research domain in terms of the identified terms and competencies.

See Appendix A for the actual terms and Appendix B for the competencies list.

5.2 *Consolidation and Exploitation of the Knowledge Model*

The consolidation and exploitation of the KM concerns the identification of *overlaps* and *gaps* within and among the research domain terms. Overlaps constitute terms that are encountered across two or more research domains (including synonyms and homonyms). Gaps represent the lack of common concepts across domains that are interrelated. By identifying such overlaps and gaps the conceptual map will be geared toward more accurately reflecting the relationships between the research domains. This will result in a concise and integrated lexicon for the different research domains, which will promote interoperability among researchers from these domains.

The consolidation of the conceptual map in terms of overlaps will be conducted by identifying the following cases[2]:

1. Enumerating terms that belong to more than one research domain, and have similar definitions.
2. Enumerating terms that belong to more than one research domain, but have incompatible definitions.
3. Enumerating pairs of terms that belong to different research domains with similar definitions.

This procedure requires the manual processing of existing terms from each research domain in relation to each other and will be facilitated through utilization of the current or future infrastructure. The assistance provided by the infrastructure will enable experts from different research domains to effectively communicate with each other in order to address the identified overlaps.

The overlap analysis will also reveal overlapping trends among research domains. Based on these trends, gap analysis will then be applied to pairs of research domain. Such analysis will focus on locating existing terms within a domain to assess whether it is feasible to add a corresponding term to the paired domain in order to resolve the gap.

5.3 Mapping Competencies to Knowledge Model Terms

The mapping between the Competencies to the Knowledge Model constitutes of cross-correlating concepts and terms contained within the KM with available expertise and contributions. The benefit of establishing such mapping is that it provides a connection of key concepts, terms, methodologies and research outputs to experts from the different research domains. Using the KM, distinct expertise and competencies can be linked both within and across the S-Cube domains. The will give a yellow-pages like approach to identify and locate experts and specific knowledge and expertise along with their contributions.

6 Summary and future work

In the previous we have described the Knowledge Model of S-Cube, its purpose and its individual components. We also identified previous approaches from related EU projects and international activities that have resulted in the definition of a large body of terms relating to software services research. These have been scrutinized, adapted and reused to the extend possible as part of the S-Cube KM. In addition, we have summarized and cross-correlated major research findings of the state-of-the-art deliverables in S-Cube, and showed how they contribute towards building the initial version of the KM. Finally, we describe the connection of the KM to a number of knowledge sources and knowledge-intensive activities within S-Cube and its usage by both internal and external users.

It must be noted that although in the DoW is mentioned that we should provide around 10 terms per research domain in the JRAs, the first version of the knowledge model actually contains far more entries than originally envisioned.

The next iteration of the KM will focus on developing separate knowledge models for the functional SBA layers, i.e., business process management, service composition and coordination, and service infrastructure. The three separated knowledge maps will contain siloed knowledge that has been assessed, tuned and coordinated for each of the three individual functional SBA layers, and common terminology within each functional layer will have been agreed-upon and defined. The separated knowledge model for the functional layers will be measured against KPIs. The KM produced will be

[2] Notice that this is an ongoing activity that has already commenced in this deliverable.

assessed by ensuring that the knowledge items (concepts) for each functional layer (e.g., service composition) that are elaborated into finer grained knowledge items, are actually based on the key knowledge items that were described in this deliverable.

Appendix A Initial Knowledge Model terms

A.1 WP-JRA-1.1: Engineering & Design

Contributors: Vasilios Andrikopoulos (Tilburg), Andreas Gehlert (UniDue), Angela Kounkou (CITY), Elisabetta di Nitto (POLIMI)

Term	**Definition**	**Relationships**	**Associated Terms**
Adaptable Service-Based Application	An Adaptable Service-Based Application is a service-based application augmented with a run time control loop that monitors and modifies itself on the basis of adaptation strategies designed by the system integrators. Notice that adaptations can be performed either because monitoring has revealed a problem or because the application identifies possible optimizations or because its execution context has changed. The context here may be defined by the set of services available to compose the service-based applications, the parameters and protocols being in place, user preferences, environment characteristics (location, time).	SYN: Adaptable SBA	*Service-Based Application*
Service Discovery	Service Discovery is the process of locating the services providing the required functionalities. Runtime service discovery is an important ingredient for self-adaptation.		*Service*, *Self-Adaptation*
Self-Healing	Self-Healing is the ability of a system or a SBA to repair itself without any external intervention.		*Self-**, *Service-Based Application*

Term	**Definition**	**Relationships**	**Associated Terms**
Service	A Service is the non-material equivalent of a good. A service provision is an economic activity that does not result in ownership, and this is what differentiates it from providing physical goods. Services are explicitly described in a Service Description. This Service Description allows the users to access a service regardless of where and by whom it is actually offered. It specifies the way the service can be accessed together with any behavioral model, constraint, and policy according to which the service must be provided. A service is opaque in that its implementation is typically hidden from the service consumer except for (1) the information and behavioral models exposed through the Service Descriptions and (2) the information required by service consumers to determine whether a given service is appropriate for their needs.		
Web Service	A Web Service is a service provided by a software system that implements a predefined set of standards. It is designed to support interoperable machine-to-machine interaction over a network. It has a service description (called interface) described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.		*Service*
Context	Context refers to the physical and social situation in which a service-based application or a service is embedded. It is defined by any information that can be used to characterize the situation of an entity - be it a person, a place or a physical or computational object - this is because of the way this information is used in interpretation rather than because of its intrinsic properties.		
Human Computer Interaction (HCI)	Human Computer Interaction (HCI) is the study of the interaction between humans and computers (in their broadest sense, including computerized devices and large scale computer systems as well as stand-alone computers). It is concerned with the design, evaluation and implementation of interactive computing systems which it aims to make more usable and useful for human use. [J. Preece, Y. Rogers, D. Benyon, S. Holland, and T. Carey, Human-Computer Interaction, Wokingham:Addison-Wesley, 1994.]	SYN: HCI, Computer Human Interaction (CHI), Human Machine Interaction (HMI)	
Adaptation Requirements and Objectives	The Adaptation Requirements and Objectives identify the aspects of the SBA model that are subject to change, and what the expected outcome of the adaptation process is.		*Adaptation, Service-Based Application*

Term	**Definition**	**Relationships**	**Associated Terms**
Service Binding	Service Binding is the process of associating a service request to a service offer. Binding can happen at design time, deployment time, and runtime.		*Service*
Service-Based Application (SBA)	A Service-Based Application is composed by a number of possibly independent services, available in a network, which perform the desired functionalities of the architecture. Such services could be provided by third parties, not necessarily by the owner of the service-based application. Note that a service-based application shows a profound difference with respect to a component-based application: while the owner of the component-based application also owns and controls its components, the owner of a service-based application does not own, in general, the component services, nor it can control their execution.	SYN: SBA	*Adaptable Service-Based Application*
Adaptation Strategies	Adaptation Strategies are the means through which adaptation is accomplished. Examples of adaptation strategies are re-configuration, re-binding, re-execution, re-planning, etc.		*Adaptation, Service-Based Application, Adaptation Mechanisms, Adaptation Requirements and Objectives*
Design for Adaptation	Design for Adaptation is a design process specifically defined to take adaptation into account. It should incorporate into the system under development all those facilities that enable the possibility to meet the adaptation requirements from very early phases to the application execution.		*Adaptation*
Self-Adaptation	Self-Adaptation is the ability of a system or a SBA to adapt itself without any external intervention.		*Adaptation, Service-Based Application*
Rebinding	Rebinding implies the replacement of a binding with another one. Rebinding can happen at design time, deployment time, and runtime.		*Service, Binding*

A.2 WP-JRA-1.2: Adaptation and Monitoring

Contributors: Raman Kazhamiakin (FBK)

Term	**Definition**	**Relationships**	**Associated Terms**
Business Activity	Business Activity is a part of a business process consisting of a series of activities implemented across workflow systems, ERP systems and legacy applications, possibly across organizational boundaries.		*Business Process*
Evolution	Evolution of Service-Based Application is a long-term history of continuous modification of SBA after its deployment in order to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.		*Service-Based Application, Adaptation*
Key Performance Indicator (KPI)	Key Performance Indicators (KPIs) are financial and non-financial metrics used to help an organization define and measure progress toward organizational goals.	SYN: KPI	*Business Activity Monitoring*
Mediation	Mediation refers to an activity in which a neutral third party, the mediator, assists two or more parties in order to help them achieve an agreement on a matter of common interest.		*Adaptation*
Optimization	Optimization of Service-Based Application is the process of modifying an application to make some aspect of it work more efficiently or use fewer resources.		*Service-Based Application, Adaptation*
Recovery	Recovery is a process of restoring the application after failing to perform one or more of its functions to fully satisfactory execution by any means other than replacement of the entire application.	SYN: Repair	*Adaptation*
Monitoring in Service Compositions	Monitoring in Service Compositions refers to checking whether certain predefined properties over the composition model are satisfied when the composition is executed.		*Monitoring, Service Composition*
Adaptation Requirements and Objectives	Adaptation Requirements and Objectives are the requirements and needs that the Service-Based Application should achieve in reaction of critical changes and events.		*Adaptation, Service-Based Application*
Adaptation Mechanisms	Adaptation Mechanisms are the tools and mechanisms provided by the underlying platform in different Functional Layers of Service-Based Application that allow for implementation of various Adaptation Strategies.		*Adaptation, Service-Based Application, Adaptation Strategies*
Quality of Service-Based Adaptation	Quality of Service-Based Adaptation refers to the adaptation that is performed in order to react to the changes in QoS parameters of a Service-Based Application.	SYN: QoS-Based Adaptation	*Adaptation*
Monitoring in Grid	Monitoring in Grid refers to scalable high performance monitoring on a large distributed computational Grid. It aims to tackle monitoring of generic middleware services and application-specific information and data transfer.		*Monitoring, Grid*

Term	**Definition**	**Relationships**	**Associated Terms**
Monitoring	Monitoring is a process of collecting and reporting relevant information about the execution and evolution of the Service-Based Application.		*Testing, Static Analysis, Service-Based Application*
Adaptation	Adaptation is a process of modifying Service-Based Application in order to satisfy new requirements and to fit new situations dictated by the environment on the basis of Adaptation Strategies designed by the system integrator.		*Service-Based Application, Adaptable Service-Based Application, Adaptation Strategies*
Adaptation Strategies	Adaptation Strategies define the possible ways to achieve Adaptation Requirements and Objectives given the available Adaptation Mechanisms.		*Adaptation, Service-Based Application, Adaptation Mechanisms, Adaptation Requirements and Objectives*
Business Activity Monitoring (BAM)	Business Activity Monitoring (BAM) provides near real-time monitoring of Business Activities, measurement of Key Performance Indicators, their presentation in dashboards, and automatic and proactive notification in case of deviations.	SYN: BAM	*Business Activity, Monitoring, Key Performance Indicator, BPM Software Suite, Business Process, Activity, Workflow*
Monitoring Events	Monitoring Events are the events that deliver the relevant information about the application evolution and changes in the environment.		*Monitoring*
Monitoring Mechanisms	Monitoring Mechanisms are the tools and facilities for continuous observing and detecting relevant Monitoring Events.		*Monitoring, Monitoring Events*

A.3 WP-JRA-1.3: Quality Definition, Negotiation, and Assurance

Contributors: Julia Hielscher (UniDue), Irena Trajkovska (UPM)

Term	**Definition**	**Relationships**	**Associated Terms**
Testing	The goal of Testing is to (systematically) execute services or service-based applications in order to uncover failures. During testing, the service or service-based application which is tested is fed with concrete inputs and the produced outputs are observed. The observed outputs can deviate from the expected outputs with respect to functionality as well as quality of service (e.g., performance or availability). When the observed outputs deviate from the expected outputs, a failure of the service or the service-based application is uncovered. Failures can be caused by faults (or defects) of the test object. Examples for faults are a wrong exit condition for a loop in the software code that implements a service, or a wrong order of the service invocations in a BPEL specification. Finding such faults typically is not part of the testing activities but is the aim of debugging. A special case of testing is profiling. During profiling, a service or a service-based application can be systematically executed in order to determine specific properties. As an example, during profiling the execution times of individual services in a service composition could be measured for ’typical’ or ’extreme’ inputs in order to identify optimization potentials. Testing cannot guarantee the absence of faults, because it is infeasible (except for trivial cases) to test all potential concrete inputs of a service or service-based application. As a consequence, a sub-set of all potential inputs has to be determined for testing. The quality of the tests strongly depends on how well this sub-set has been chosen. Ideally this sub-set should include concrete inputs that are representative for all potential inputs (even those which are not tested) and it should include inputs that – with high probability – uncover failures. However, in cases where choosing such an ideal sub-set typically is infeasible, it is important to employ other quality assurance techniques and methods which complement testing.		*Monitoring, Static Analysis*
Quality of Service Level	Quality of Service (QoS) Level defines the different modes in which a system can be. Depending on, e.g., available resources, a different execution level can be jumped to if continuing the execution in the current one is not possible. The submetamodel defines the abstract classes to represent levels, transitions between them, and when those transitions have to take place.	SYN: QoS Level	*Quality of Service Characteristics, Quality of Service Constraints*

Term	**Definition**	**Relationships**	**Associated Terms**
Quality of Service Dimensions	Quality of Service (QoS) Dimensions is a tool for improving the management aspects of Web service-based architectures. In particular, typical issues in loosely coupled environments management, such as service discovery and selection, composition, and monitoring, raise different issues concerned with negotiation of QoS. Finally, QoS negotiation can be implemented according to different paradigms, such as broker-based architectures and multi-agent systems. Each implementation paradigm introduces specific issues that must be dealt with while tackling the Web service QoS negotiation problem.	SYN: QoS Dimensions	*Quality of Service Negotiation, Declarative Quality of Service Models, Ontological Quality of Service Models*
Constructive Quality Assurance	The goal of Constructive Quality Assurance techniques and methods is to prevent the introduction of faults (or defects) while the artifacts are created. Examples for such techniques include code generation (model-driven development), development guidelines, as well as templates.		*Analytical Quality Assurance*
Analytical Quality Assurance	The goal of Analytical Quality Assurance techniques and methods is to uncover faults in the artifacts after they have been created. Examples for analytical quality assurance techniques are reviews and inspections, formal correctness proofs, testing, as well as monitoring.		*Constructive Quality Assurance, Static Analysis, Testing, Monitoring*
Monitoring	Monitoring observes services or service-based applications during their current execution, i.e. during their actual use or operation. In addition, the context of a service or a service-based application can be monitored. This context can include other systems, the execution platform (hardware, operating systems, etc.) and the physical environment (e.g., sensors or actuators). Monitoring can address different goals. Further, monitoring can be used to enable the context-driven run-time adaptation of a service-based application. Also, monitoring may be used to uncover failures during the current execution of a service or service-based application. In contrast to testing and static analysis, which aim at providing more or less general statements about services or service-based applications, monitoring always provides statements about their current execution (i.e., about current execution traces). Thereby, monitoring can uncover failures which have escaped testing, because the concrete input that lead to the current execution trace might have never been tested. Also, monitoring can uncover faults which have escaped static analysis, because static analysis might have abstracted from a concrete aspect which was relevant during the current execution.		*Testing, Static Analysis, Service-Based Application*

Term	**Definition**	**Relationships**	**Associated Terms**
Static Analysis	The aim of Static Analysis is to systematically examine an artifact in order to determine certain properties (synthesis) or to ascertain whether some predefined properties are met (verification). Analysis can be applied at several stages in the development cycle, and therefore examples for artifacts which can be subject to analysis include requirement documents, design specifications, interface descriptions, and code. Examples of static analysis include formal techniques and methods, such as data flow analysis, model checking, execution in abstract domains, symbolic execution, type checking, and correctness proofs, which are all usually characterized because they compute properties that are in many cases approximations of the more concrete properties, but which, in this case, are safe, in the sense that the lack of accuracy must not lead to an error for the intended use of the analysis. Informal approaches, such as reviews, walkthroughs, and inspections, are as well examples of static analysis. In contrast to testing (or monitoring), where individual executions of the services or service-based applications are examined, analysis can examine classes of executions. Thus, analysis can lead to more universal statements about the properties of the artifacts than testing (or monitoring). In order to achieve these more universal statements, static analysis – unlike testing or monitoring – does not execute the artifacts which are being examined, since termination (which is theoretically ensured when the system has a finite state space) is usually a necessary condition for a successful analysis. However, systems may have a state space so large (or infinite) as to make traversing it unfeasible. In those cases static analysis resorts to working with safe approximations of the actual system semantics, which makes the system actually under analysis effectively finite, but different from the initial one. Those approximations can be very sophisticated and take the form of, e.g., relations between inputs and outputs which approximate the system behavior in the domain of the analysis. When these approximations capture the properties of interest faithfully enough, then the results, even if not as accurate as they could be, are useful – and correct. Yet, as approximations might abstract away from some relevant concrete details, aspects might be overlooked or simply not be captured faithfully enough. Thus static analysis can complement the other classes of quality assurance techniques and methods but typically will not be enough, if used in isolation, in order to give a complete picture of the whereabouts of the execution of a computational system.		*Testing, Monitoring*
Quality Dimensions	Quality Dimensions express capabilities or requirements. By grouping a set of relevant quality dimensions a service can be defined by its quality that states how the service work	SYN: Quality Attributes, Quality Parameters	

Term	**Definition**	**Relationships**	**Associated Terms**
Quality of Service Characteristics	Quality of Service (QoS) Characteristics is a submetamodel that includes the names (constructors) of the nonfunctional characteristics in the QoS model (e.g., latency, throughput); the dimensions in which each characteristics is measured (e.g.: reliability can be measured in MTBF, time to repair, etc), as well as the direction of order in the domain, its units, associated statistics, etc.; the possibility of grouping several characteristics (e.g., the performance category); the description of the values that quantifiable QoS characteristics can take, and others.	SYN: QoS Characteristics	*Quality of Service Constraints, Quality of Service Level*
Quality of Service Negotiation	Quality of Service (QoS) Negotiation is a tool for improving the management aspects of Web service-Based architectures. In particular, typical issues in loosely coupled environments management, such as service discovery and selection, composition, and monitoring, raise different issues concerned with negotiation of QoS. Finally, QoS negotiation can be implemented according to different paradigms, such as broker-Based architectures and multi-agent systems. Each implementation paradigm introduces specific issues that must be dealt with while tackling the Web service QoS negotiation problem.	SYN: QoS Negotiation	*Quality of Service Dimensions, Declarative Quality of Service Models, Ontological Quality of Service Models*

A.4 WP-JRA-2.1: Business Process Management

Contributors: Vasilios Andrikopoulos (Tilburg), Willem-Jan van den Heuvel (Tilburg)

Term	**Definition**	**Relationships**	**Associated Terms**
Workflow	A Workflow is a technology for realizing inter- and intra-enterprise (business) process. Workflow constructs make it possible to implement business process aspects like logical decision points, sequential as wells as parallel work routs, as well as managing of exceptional situations. This is realized by the means of control flow constructs of a workflow language. The business rules (complex transition conditions) specify in reusable manner the way to process the workflow specific data.		*Process, Business Process, Business Rules*
Business Process	A Business Process is a process used to achieve a well-defined business outcome and is completed according to a set of procedures. The key elements in this definition are that a business process may span organizations and may typically involve both people and systems. A business process includes both automated and manual tasks.		*Process*
Key Performance Indicator (KPI)	A Key Performance Indicator (KPI) is a quantifiable metric that a firm uses to measure performance in terms of meeting its strategic and operational objectives. KPIs provide critical information to the organization for monitoring and predicting business performance in accordance with strategic objectives in a way that compliments financial performance. By measuring and monitoring operational efficiency, employee performance and innovation, customer satisfaction, as well as financial performance, long term strategies can be linked to short term actions. [A. Neely, M Gregory, and K. Platts, "Performance measurement system design: A literature review and research agenda," International Journal of Operations & Production Management, vol. 25, no. 12, pp. 1228—1263, 2005.]	SYN: KPI	*Business Activity Monitoring*
Activity	An Activity is an element that performs a specific function within a process. Activities can be as simple as sending or receiving a message, or as complex as coordinating the execution of other processes and activities.	SYN: Task, Business Activity	*Process*
Business Protocol	A Business Protocol specifies the possible message exchange sequences (conversations) that are supported by the service to achieve a business goal. Business protocols are not executable, but protocols can be specified using BPEL or any of the many other formalisms developed for this purpose.		*Business Process*

Term	**Definition**	**Relationships**	**Associated Terms**
Business Activity Monitoring (BAM)	Business Activity Monitoring (BAM) refers to near real-time monitoring of business activities, measurement of key performance indicators (KPIs), their presentation in dashboards, and automatic and proactive notification in case of deviations. A “business activity” thereby can be implemented as a service orchestration in a BPMS, or, more general, as part of a business process consisting of a series of activities implemented across workflow systems, ERP systems and legacy applications, possibly across organizational boundaries. BAM software gathers information from these applications in form of events, aggregates and analyzes these events to compute KPIs, and reacts to deviations by notifying business users.	SYN: BAM	*Business Activity, Monitoring, Key Performance Indicator, BPM Software Suite, Business Process, Activity, Workflow*
Business Transaction	A Business Transaction is driven by well-defined business tasks and events that directly or indirectly contribute to generating economic value, such as processing and paying an insurance claim, and has also an associated number of parameters that represent security and timing requirements. A business transaction always either succeeds or fails with respect to the business task (function) that initiated it and governs it throughout its execution. If a business transaction completes successfully then each participant will have made consistent state changes, which, in aggregate, reflect the desired outcome of the multi-party business interaction.		*Business Process*
Business Process Integration (BPI)	Business Process Integration (BPI) refers to the ability to define a commonly acceptable business process model that specifies the sequence, hierarchy, events, execution logic and information movement between systems residing in the same enterprise (viz. EAI) or systems residing in multiple interconnected enterprises.	SYN: BPI	*Business Process, Enterprise Application Integration*
Business Policies	Business Policies capture the nature of an enterprise’s business model and define the conditions that must be met in order to move to the next stage of the process.		
Business Rules	Business Rules define the business terms and facts (structural assertions) as well as the constraints underlying the business behavior (action assertions). Business rules represent core business policies. Business rules are represented as compact (declarative) statements about an aspect of the business that can be expressed within an application in unambiguous terms that can be directly related to the business and its collaborators and as such they determine the route of action to be followed [B. von Halle, "Business Rules Applied", J. Wiley & Sons, 2001. R. G. Ross, "Principles of the Business Rule Approach", Addison-Wesley Information Technology Series. Addison-Wesley, 2003.]		*Business Policies*
Value Chain	A Value Chain is the largest possible process in an organization. The value chain is decomposed into a set of core business processes and support processes necessary to produce a service, product or product line. These core business processes are subdivided into activities.		*Process, Activity*

Term	**Definition**	**Relationships**	**Associated Terms**
BPM Software Suite	A BPM Software Suite (BPMS) provides an integrated set of tools to model, design, simulate and deploy business processes and process-based applications, delivering greater degrees of process management delivery. BPMS present a “closed loop” system for achieving business performance improvement, offering a set of integrated tools that support designing, measuring, monitoring, analyzing, optimizing, and continuously improving business processes.	SYN: BPMS, BPM Suite	*Business Process*
Agile Service Network (ASN)	An Agile Service Network (ASN) comprises large numbers of long-running, highly dynamic complex end-to-end service interactions reflecting asynchronous message flows that typically transcend several organizations and span geographical locations. The term complex end-to-end service interaction signifies a succession of automated business processes, which are involved in joint inter-company business conversations and transactions across a federation of cooperating organizations.	SYN: ASN	*Business Process, Business Transaction*
Business Process Modeling	Business Process Modeling provides a shared environment for the capture, design and simulation of business processes by business analysts, managers, architects and other IT professionals. Modern business process modeling tools include business process analysis functionality of capturing, designing, and modifying business processes and their properties, resource requirements, such as definition and selective enforcement of process standards. They also facilitate the expression of business process views at different levels of abstraction depending on authorization, functional responsibility and the level of detail desired.		*Business Process*
Business Process Execution	Business Process Execution refers to the deployment and execution of a business process within a BPM execution engine (usually a part of BPM Software Suite). The BPM execution engine executes process instances by delegating work to humans and automated applications as specified in the process model.		*Business Process, BPMS*
Business Process Analysis, Monitoring and Auditing	Business Process Analysis, Monitoring and Auditing involves providing graphical administrative tools that illustrate processes that are in progress, processes that are completed, and integrate business metrics and key performance indicators with process descriptions. Audit trails and process history/reporting information is automatically maintained and available for further use.		*Business Process*
Business Process Measurement	Business Process Measurement refers to the activity of aggregating process data in business-oriented metrics such as key performance indicators and balanced scorecards, and the optimization of the process by reconfiguring resources or modifying business rules – dynamically and “on the fly.”		*Business Process, Key Performance Indicator*
Business Process Optimization	Business Process Optimization involves optimizing process flows of all sizes, crossing any application, company boundary and connects process design and process maintenance.		*Business Process*

A.5 WP-JRA-2.2: Service Composition

Contributors: Olha Danylevych (USTUTT), Dragan Ivanovic (UPM), Branimir Wetzstein (USTUTT)

Term	**Definition**	**Relationships**	**Associated Terms**
Service Orchestration	Service Orchestration is a form of service composition in which a new service is created by orchestrating several services in a process flow. Orchestrated services can be atomic services, i.e. self-contained entities which do not use other services, or again service orchestrations. The standard language for orchestrating Web services is WS-BPEL.		*Service Composition*
Service Choreography	Service Choreography is a form of service composition in which the interaction protocol between several partner services is defined. The goal is to define the so called public processes of the interacting partners and how they communicate together. For the specification of service choreographies, visual notations can be used such as BPMN and Let's Dance. Other approaches are defined based on an XML-based language such as WS-CDL, or adapt a language for service orchestrations such as BPEL4Chor.		*Service Composition*
Service Composition	Service Composition is a combination of a set of services for achieving a certain purpose. Different service composition types can be distinguished, in particular: service orchestration, service choreography, service wiring, and service coordination.	SYN: Service Aggregation	*Service Orchestration, Service Choreography, Service Wiring, Service Coordination, Semantic Web Service Composition*
Service Coordination	Service Coordination is a form of service composition in which a distributed activity is created by temporarily grouping a set of service instances following a coordination protocol. At the end of the activity a coordinator decides on the outcome of the protocol and disseminates the result to the participating services. WS-Coordination is an example of a specification which supports coordination of Web services.		*Service Composition*
Model-Driven Service Composition	Model-Driven Service Composition is a service composition that generates service orchestrations from a more general or abstract model.		*Service Composition, Service Orchestration*

Term	**Definition**	**Relationships**	**Associated Terms**
Service Wiring	Service Wiring is a composition type in which a set of interacting services is assembled into a package. Such package, called service assembly, can be deployed in a run-time environment, ready to be invoked as a service itself.Services specify provided and requested interfaces in form of operations with inputs and outputs. In order to create executable service assemblies, the requested interfaces of one service are wired to provided interfaces of other services. The services assembled in this way can again be recursively exposed as a service which can be wired and invoked. A service assembly is a deployable artifact, which is deployed to an enterprise service bus. Service Component Architecture is a service wiring technology.	SYN: Service Assembly	*Service Composition, Service Component Architecture*
Automated Service Composition	Automated Service Composition is a family of approaches to service composition that aim at full or partial automation of the composition process, in order to enable handling of higher levels of complexity.		*Service Composition*
Quality of Service-Aware Service Composition	Quality of Service-Aware Service Composition is a form of service composition that is based on and attempts to improve overall Quality of Service (QoS) attributes of service composition,such as execution time, reliability, availability, or cost.	SYN: QoS-Aware Service Composition	*Service Composition*

A.6 WP-JRA-2.3: Service Infrastructure

Contributors: Attila Kertesz (SZTAKI), Zsolt Nemeth (SZTAKI), Martin Treiber (TUW)

Term	**Definition**	**Relationships**	**Associated Terms**
Service Registry	A Service Registry is a repository that contains Web service related meta information (e.g. Web service descriptions).		
Service Discovery	Service Discovery is the process of finding services that match the requirements of the service requestor.		*Service, Self-Adaptation*
Service Mediation	Service Mediation is the process of intercepting and modification of messages that are exchanged between services.		
Dynamic Invocation	Dynamic Invocation is the execution of a service whose interface is first known at run time.		
Grid	A Grid is a fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources (computing power, storage, instruments, data, etc.) in order to generate knowledge.		
*Self-**	Self-* is called one or more properties of the computing system that represent reflexive actions, e.g. self-healing, self-optimising, etc. (Currently there are over 20 various self-* terms)	SYN: Selfware	*Adaptive, Autonomic*
Self-Healing	Self-Healing is called the ability of a computing component to detect, diagnose and repair localized problems resulting from bugs or failures in software and hardware.		*Self-*, Service-Based Application*
Self-Optimization	Self-Optimization is called the ability of a computing component to seek ways to improve its operation, identifying and seizing opportunities to make itself more efficient in performance or cost.	SYN: Self-Optimising	*Self-*, Optimization*
Dynamic Binding	Dynamic Binding refers to the selection of the actual service at run time.		
Autonomic	Autonomic is called an entity of a computing system capable to manage its own operation without human intervention.	SYN: Self-Managing, Self-Governing	
Self-Configuration	Self-Configuration is called the ability of a computing component to configure itself in accordance with high-level policies that specify what is desired not how it is to be accomplished.		*Self-**

Term	**Definition**	**Relationships**	**Associated Terms**
Self-Protection	Self-Protection is called the ability of a computing component to defend the system as a whole against large-scale, correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures; to anticipate problems form early reports and take steps to avoid or mitigate them.	SYN: Self-Protective	*Self-**, *Service-Based Application*

Appendix B Initial list of Competencies

Domain	Expertise	Individual/Contact Person	Institution	Contact info
Engineering and Design	HCI, Requirements Engineering, Service-Centric Systems Engineering	Angela Kounkou	CITY	
	HCI, Requirements Engineering, Service Discovery, Service-Centric Systems Engineering	Neil Maiden	CITY	
	Requirements Engineering, Service Discovery, Service-Centric Systems Engineering	Kos Zachos	CITY	
	Service Oriented Architectures	Mike P. Papazoglou	Tilburg	
	Service Oriented Architectures	Marco Pistore	FBK	
	Service Oriented Architectures	Paolo Traverso	FBK	
	Service Oriented Architectures	Carlo Ghezzi	POLIMI	
	Service Adaptation	Barbara Pernici	POLIMI	
	Service Adaptation	Luciano Baresi	POLIMI	
	Service Adaptation	Elisabetta di Nitto	POLIMI	
	Goal-Based Requirement Engineering Approaches (especially i* and Tropos)	Klaus Pohl	UniDue	
	Scenario-Based Requirement Engineering Approaches	Andreas Gehlert	UniDue	
	Context Modeling & Analysis		UniDue	
	Service Modeling	Ali Arsanjani	IBM Global Services	arsanjan@us.ibm.com
	Service Modeling	Olaf Zimmermann	IBM Zurich Research Lab	olz@zurich.ibm.com
	Service Oriented Architectures	Boualem Benatallah	University of New South Wales	boualem@cse.unsw.edu.au
	Service Oriented Architectures	Thomas Erl	SOA Systems Inc.	
	Service Oriented Architectures	Fabio Casati	University of Trento	casati@disi.unitn.it
	Service Planning	Marco Aiello	Rijksuniversiteit Groningen	aiellom@cs.rug.nl
Adaptation and Monitoring	Diagnosis	Elisabetta di Nitto	POLIMI	
	Self-healing systems	Luciano Baresi	POLIMI	
	Repair	Luciano Baresi	POLIMI	
	Adaptation	Jean-Louis Pazat	INRIA	

Domain	Expertise	Individual/Contact Person	Institution	Contact info
	Adaptation	Maria Grazia Fugini	POLIMI	
	QoS-Based Adaptation	Elisabetta di Nitto	POLIMI	
	Grid Computing	Jean-Louis Pazat	INRIA	
	Monitoring in WS Compositions	Luciano Baresi	POLIMI	
	Monitoring in WS Compositions	Marco Pistore	FBK	
	Monitoring in WS Compositions	George Spanoudakis	CITY	
	Monitoring in WS Compositions	Luciano Baresi	POLIMI	
	Monitoring in WS Compositions	Salima Benbernou	UCBL	
	Process Mining	Mohand-Said Hacid	UCBL	
	Business Activity Monitoring	Branimir Wetzstein	USTUTT	
	Monitoring in Grid	Gabor Kecskemeti	SZTAKI	
	Process Mining	Fabrizio Silvestri	CNR	
	Adaptation	Luca Cavallaro	POLIMI	
	Business Activity Monitoring	Josef Schiefer	Inst. for Software Technol. & Interactive Syst., Vienna	
	Process Mining	W.M.P. van der Aalst	DTM, Eindhoven University of Technology, Eindhoven	
	Monitoring, SLA	Heiko Ludwig	IBM T.J. Watson	
	Monitoring in Grid	Sergio Andreozzi	INFN-CNAF, Bologna	
	Optimization	Amit P. Sheth	University of Georgia	
	Mediation	Boualem Benatallah	CSE, University of New South Wales	
Quality Definition, Negotiation and Assurance		Marco Pistore	FBK	
		Luciano Baresi	POLIMI	
		Raman Kazhamiakin	FBK	
		Carlo Ghezzi	POLIMI	
		Mike P. Papazoglou	Tilburg	
		Klaus Pohl	UniDue	
		Andreas Metzger	UniDue	
		Wei-Tek Tsai	Arizona State University	wei-tek.tsai@asu.edu
		Raymond Paul	Arizona State University	

Domain	Expertise	Individual/Contact Person	Institution	Contact info
		Marianne Winslett	University of Illinois	winslett@uiuc.edu
		Wil M.P. van der Aalst	Eindhoven University of Technology	w.m.p.v.d.aalst@tue.nl
		Xiaoying Bai	Tsinghua University	baixy@tsinghua.edu.cn
		Elisa Bertino	Purdue University	bertino@cs.purdue.edu
Business Process Management	BPM	Frank Leymann	USTUTT	
	Business Processes, Business Process Reusability	Dimka Karastoyanova	USTUTT	
	Business Activity Monitoring, Business Process Execution	Branimir Wetzstein	USTUTT	
	Workflows	Olha Danylevych	USTUTT	
	BPM	Mike P. Papazoglou	Tilburg	
	Business Protocols, Business Processes	Willem-Jan van den Heuvel	Tilburg	
	Business Protocols	Michele Mancioppi	Tilburg	
	Business Protocols	Mohand-Said Hacid	UCBL	
	Value Networks	Christos N. Nikolaou	UOC	
	Value Networks	Marina Bitsaki	UOC	
	Business Protocols	Boualem Benatallah	University of New South Wales	boualem@cse.unsw.edu.au
	Business Protocols	Fabio Casati	University of Trento	casati@disi.unitn.it
	Business Protocols	Farouk Toumani	Blaise Pascale University	ftoumani@isima.fr
	Business Processes	Wil M. P. van der Aalst	Eindhoven University of Technology	w.m.p.v.d.aalst@tue.nl
	Business Processes	Manfred Reichert	University of Twente	m.u.reichert@cs.utwente.nl
	Business Processes	Arthur H. M. ter Hofstede	Queensland University of Technology	a.terhofstede@qut.edu.au
	Value Networks	Verna Allee	Verna Allee Associates	verna@vernaallee.com
	Value Networks	Jaap Gordijn	Vrije Universiteit Amsterdam	gordijn@cs.vu.nl
	Value Networks	Hans Akkermans	Vrije Universiteit Amsterdam	elly@cs.vu.nl
	Business Rules	Shazia Sadiq	University of Queensland	shazia@itee.uq.edu.au
	Business Processes	Heiko Ludwig	IBM's TJ Watson Research Center	hludwig@us.ibm.com
	Business Processes	Thomas H. Davenport	Babson College	tdavenport@babson.edu
	Business Processes	August-Wilhelm Scheer	Institut für Wirtschaftsinformatik	

Domain	Expertise	Individual/Contact Person	Institution	Contact info
Service Composition	Service Orchestration, Service Choreography, Service Wiring, Service Coordination, Semantic WS Composition, Model-Driven Service Composition	Frank Leymann	USTUTT	
	Service Orchestration, Service Choreography, Service Wiring, Service Coordination	Dimka Karastoyanova	USTUTT	
	Service Orchestration, Service Choreography, Service Wiring, Service Coordination	Branimir Wetzstein	USTUTT	
	Service Orchestration, Service Choreography, Service Wiring, Service Coordination, Model-Driven Service Composition	Mike P. Papazoglou	Tilburg	
	Semantic WS Composition	Kyriakos Kritikos	UOC	
	Automated Service Composition	George Baryannis	UOC	
	Automated Service Composition	Paolo Traverso	FBK	
	Automated Service Composition, Verification of Service Compositions	Marco Pistore	FBK	
	Automated Service Composition, Verification of Service Compositions	Raman Kazhamiakin	FBK	
	Service Orchestration, Service Choreography, Service Wiring, Service Coordination, Model-Driven Service Composition, QoS Aware Service Composition	Schahram Dustdar	TUW	
	Model-Driven Service Composition	Philipp Leitner	TUW	
	QoS Aware Service Composition	Florian Rosenberg	TUW	
	QoS Aware Service Composition	Ivona Brandic	TUW	
	QoS Aware Service Composition	Barbara Pernici	POLIMI	
	Verification of Service Compositions	Manuel Carro	UPM	
	Verification of Service Compositions	Carlo Ghezzi	POLIMI	
	Service Orchestration	Gustavo Alonso	Department of Computer Science ETH Zentrum, Zürich	alonso@inf.ethz.ch
	Service Choreography	Alistair Barros	SAP, Brisbane Research Centre	alistair.barros@sap.com
	Service Coordination	Eric Newcomer	IONA	eric.newcomer@iona.com
	Semantic WS Composition	Dieter Fensel	STI Innsbruck	dieter.fensel@sti2.at
	QoS Aware Service Composition	Michael C. Jaeger	Technische Universität Berlin,	mcj@cs.tu-berlin.de

Domain	Expertise	Individual/Contact Person	Institution	Contact info
			Germany	
	QoS Aware Service Composition	Liangzhao Zeng	IBM Thomas J. Watson Research Center	lzeng@us.ibm.com
Service Infrastructure	SOA Registry and Discovery	Philipp Leitner	TUW	
	SOA Registry and Discovery	Florian Rosenberg	TUW	
	SOA Registry and Discovery	Ivona Brandic	TUW	
	SOA Registry and Discovery	Schahram Dustdar	TUW	
	Web Services Registry and Discovery	Franco Maria Nardini	CNR	
	Web Services Registry and Discovery	Gabriele Tolomei	CNR	
	Web Services Registry and Discovery	Fabrizio Silvestri	CNR	
	Web Services Registry and Discovery	Kyriakos Kritikos	UOC	
	Web Services Registry and Discovery	Pierluigi Plebani	POLIMI	
	Dynamic adaptation in grid and mobile environments	Jean-Louis Pazat	INRIA	
	Dynamic adaptation in grid and mobile environments	Francoise Andre	INRIA	
	Self-healing brokering	Attila Kertesz	SZTAKI	
	Self-deployment	Gabor Kecskemeti	SZTAKI	
	Self-*, nature inspired adaptation models	Zsolt Nemeth	SZTAKI	
	IR, web IR	Ricardo Baeza-Yates	Yahoo! research BCN	
	Peer-to-peer	Ophir Frieder	IIT, Chicago and Georgetown University	
	Service discovery, SOA	Boualem Benatallah	University of New South Wales	boualem@cse.unsw.edu.au
	Service discovery, SOA	Fabio Casati	University of Trento	casati@disi.unitn.it
	Service discovery, SOA	Gustavo Alonso		
	Service discovery, SOA	Paco Curbera		
	Grid	Ian Foster	Argonne National Laboratory, University of Chicago	
	Grid	Carl Kesselman	USC, Information Sciences Institute	
	Autonomic computing, self-*	Jeffrey O. Kephart	IBM T.J. Watson Lab	
	Grid Scheduling	Uwe Schwiegelshohn	Dortmund University of Technology	uwe.schwiegelshohn@udo.edu

Domain	**Expertise**	**Individual/Contact Person**	**Institution**	**Contact info**
	Grid Brokering	Ramin Yahyapour	Dortmund University of Technology	ramin.yahyapour@udo.edu
	Grid Workflows	Andreas Hoheisel	Fraunhofer FIRST	andreas.hoheisel@first.fraunhofer.de