Grant Agreement N° 215483

| | |
|---|---|
| *Title:* | *State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge* |
| *Authors:* | *Vasillios Andrikopoulos (Tilburg) , Piergiorgio Bertoli (FBK), Silvia Bindelli (Polimi), Elisabetta Di Nitto (Polimi), Andreas Gehlert (UniDue), Lola Germanovich (City), Raman Kazhamiakin (FBK), Angela Kounkou (City), Barbara Pernici (Polimi), Pierluigi Plebani (Polimi), Thorsten Weyer (UniDue)* |
| *Editor:* | *Elisabetta Di Nitto (Polimi)* |
| *Reviewers:* | *Schahram Dustdar (TUW), Mohand-Said Hacid (UCBL), Frank Leymann (USTUTT), Ita Richardson (Lero-UL), Shingo Takada (Keyo University, Japan)* |
| *Identifier:* | *Deliverable # PO-JRA-1.1.1* |
| *Type:* | *Deliverable* |
| *Version:* | *1* |
| *Date:* | *14 July 2008* |
| *Status:* | *Final* |
| *Class:* | *External* |

## Management Summary

This deliverable surveys the state of the art in all areas related to the engineering of service-based applications with particular focus on all aspects related to adaptivity. Moreover, it provides an overview of various aspects concerning the way human beings interact with each other and with the computerized systems. This second aspect, even if not directly related to the Service-Oriented Computing context is being analyzed in order to understand if it could be a good source of inspiration for new challenges and new issues for service-based applications. The deliverable provides some initial thoughts about these challenges and issues. Further analysis and research will be developed in the following of the S-Cube project.

*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/

# The S-CUBE Deliverable Series

## Vision and Objectives of S-CUBE

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.
- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with today's limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-CUBE materials are available from URL: http://www.s-cube-network.eu/

# Table of Contents

# Table of illustrations

# List of acronyms

BPEL: Business Process Execution Language
BPM: Business Process Management
CBD: Component-Based Development
CCTT: Collaborative ConcurTaskTree, an extension to CTT to support the modeling of collaborative systems
CTT: ConcurTaskTree, a task modeling technique
CTTE: ConcurTaskTree Environment, tool to support the development of task models following the CTT notation
EA: Enterprise Application
ERP: Enterprise Resource Planning
HCI: Human-Computing Interaction
HTA: Hierarchical Task Analysis, a task modeling technique
KAOS: Knowledge Acquisition in Automated Specification
OOAD: Object-Oriented Analysis and Design
OWL: Web Ontology Language
QoS: Quality of Service
RE: Requirements Engineering
SBA: Service-Based Applications
SDM: Strategic Dependency Model
SeCSE: Service Centric System Engineering
SLA: Service Level Agreement
SOA: Service-Oriented Architectures
SOAP: Simple Object Access Protocol
SOC: Service-Oriented Computing
SOMA: Service-Oriented Modelling and Architecture
WSDL: Web Services Description Language
WSCF: Web Services Contract First

# Introduction

## *1.1 Context and objectives*

Engineering service-based applications is quite different from any other software engineering activity. As we will clarify in the next section, 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 creation of the glue used to compose the various services is usually achieved by exploiting a workflow-based approach [16] and, lately, a specific workflow language called BPEL [17].
The possibility to reuse in service-based applications 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 a service-based application 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. Again, this requires the selection and possibly (re)placement of services at runtime.

While a number of specific techniques have been developed to support so called dynamic discovery, selection, and binding of services, a set of engineering methods aiming at developing service-based application ready to self-adapt at runtime is still missing. An engineering 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.

This report surveys the state of the art in all areas related to the engineering of service-based applications with particular focus on all aspects related to adaptivity. Also, it overviews various aspects concerning the way human beings interact with each other and with the computerized systems. These aspects are often incorporated in the Human –Computing Interaction (HCI) field. We analyze these because HCI aspects have always been very important for the definition of the proper tools and approaches that enable the development and engineering of software systems. The deliverable provides some initial thoughts about how these HCI aspects are relevant to service-based applications engineering. Further analysis and research will be developed in the following of the S-Cube project.

The rest of this paper is structured as follows:

- Section 1.2 provides some useful terminology
- Section 2 focuses on the issues related to the engineering of service-based applications. In particular:
    - Section 2.3 surveys the classical approaches to support service-based applications
    - Section 2.4 focuses on the requirement engineering phase
    - Section 2.5 focuses on the design phase. Here, in particular, we focus on those approaches that are suitable to adaptive service-based applications

  - Section 2.6 summarizes the aspects concerned with monitoring and quality assurance and their relationships with the life cycle of service-based applications.
  - Section 2.7 focuses on the techniques that support the life cycle of service-based applications at runtime.
- Finally, Section 3 focuses on the experiences gathered in the field of Human Computing Interaction (HCI) and on their relevance within the context of engineering service-based applications.

This deliverable does not focus on the engineering of atomic services. In future activities we will investigate if this aspect can influence the development of service-based applications and how. Indeed, the deliverable does not address the issues related to monitoring since they are specifically addressed by PO-JRA-1.2.1 and PO-JRA-1.3.1.
Self-adaptation is also addressed in PO-JRA-1.2.1. While in that deliverable the focus in on reviewing the various techniques that address some aspects of adaptation in order to evaluate similarities and overlapping between them, here we try to understand the role of such techniques within the life cycle of service-based application and the corresponding development and operation processes. In the next steps we will focus on merging the knowledge that has been acquired within JRA-1.2 with ours in order to achieve a common view both on the specific techniques (that are the main aim of JRA-1.2) and on the engineering methods (that are the main objective of JRA-1.1).

## *1.2 Terminology*

In the following we propose some definitions that are important for the following of this report. These definitions have been derived by studying the ones offered by various sources [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [18].

### 1.2.1 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 behavioural 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 behavioural 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.

A **web service** is 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.

In [14] and [15] some important characteristics of services are presented. Two important principles are the **standardization of their service contracts** -- which define the way they can be accessed and used -- and the level of **abstraction** of such contracts that should be independent of any information that is not essential to enable the access and usage of the service.
Services can be characterized by different degrees of **reusability**, **discoverability**, and **composability**, depending respectively on how easy it is to reuse the service in different deployed architectures, to discover the service, and to compose it within complex architectures.
Another interesting feature of services is the degree of **interoperability**, which consists in making services less dependent from one another, with the aim of making each service more open to the invocation from different service consumers. This principle is highly connected with **loose coupling**,

whose aim is to reduce the dependencies within and outside service boundaries. Loose coupling can indeed allow for fostering interoperability. As an example of “internal” reduction of coupling, services should achieve a good degree of **implementation neutrality**, that is, they should be as independent as possible from the implementation.
Conversely, a positive feature of services is to what extent they are able to **cooperate with each other** to achieve computation.
One more key feature of services is their **autonomy**, with respect mainly to their execution environment.
Another feature of services it is interesting to evaluate is the way they require to be configured: **flexible configurability** is indeed a relevant characteristic. A service should also be **persistent** enough for the detection and solution of possible problems, and for engendering some trust in its behaviour.
Moreover, the approaches to service-based architecture should model components and their interactions at a **coarse granularity**, as to reduce the dependencies and the messages exchanged among SOA’s participants.
Finally, here we are considering two more features of services: **heterogeneity** of services and **always-on semantics**. The second one consists of the continuous availability of a service, without the need to create and destroy it every time it is used.

### 1.2.2 (Adaptive) Service-based Application

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.

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).

### 1.2.3 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.
Context is relevant to many activities concerned with the development and operation of service-based applications. In particular, it is relevant for the understanding, definition, and interpretation of the requirements for the planned system. Its changes are important drivers for the adaptation of a service-based application, and the information that it holds are needed for the application of specific adaptation techniques.

# 2 Engineering service-based applications

As we will discuss in detail in Section 2.3, main companies such as IBM have participated to the definition of approaches to support engineering of service-based applications. In these efforts the focus is mainly on defining methods that allow analysts and designers to map business processes into services. When implemented in software, services can be built internally to the company, possibly reusing other services or wrapping legacy code, or they can be outsourced from well known companies. There are various aspects that are peculiar of service-based applications where compared to normal software. In particular, SLA definition and monitoring becomes very important in all cases services are outsourced. Also, the possibility to consider the services that implement processes as reusable assets within the company is an important issue. Figure 1 shows the life cycle of service-based applications. It has been derived by slightly refining the one presented in [3].

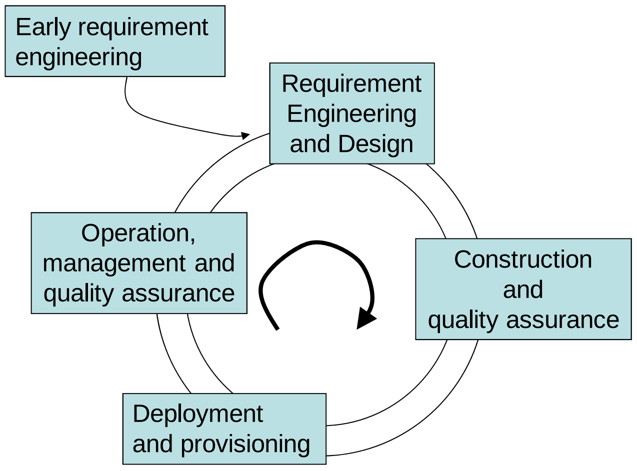

Early requirement engineering
Requirement Engineering and Design
Construction and quality assurance
Deployment and provisioning
Operation, management and quality assurance

**Figure 1. Service-based applications life cycle.**

The detailed presentation of the figure is given in Section 2.3.1 where we describe Papazoglou's approach in detail. Here we want to stress the fact that while in the classical software life cycles the main focus is on the pre-deployment phases, in the figure, operation, management and quality assurance are highlighted to stress the importance of keeping under control the characteristics of service-based applications at runtime. Also, the iterative characteristic of the life cycle, where there is a possible transition from the execution to analysis and design, is shown.

So far we have discussed about those service-based applications that are developed to fulfill the business needs of a specific enterprise. The number of issues to be faced grows up when we look at less structured application domains. An example is ambient intelligence where there is the assumption that an application enters in contact with unknown services and exploits them for a short period of time, moving to different services when this is needed, for instance, because of a context change. Another example, still in the business domain, is the spontaneous formation of networks of enterprises that start cooperating to achieve a common goal. In both these cases the environment is highly dynamic and the relationships among the various services that compose an application are not long lasting nor regulated by predefined SLAs. In these settings, the *adaptability* becomes a key feature of the SBA as it provides a way for the application to continuously change itself in order to satisfy new requirements and to fit new situations dictated by the environment.

The goal of the adaptation is, therefore, to accommodate the application to various changes that occur at different phases of the life cycle of the SBA. We can identify a set of concepts that are relevant for the description of the adaptation.

The *adaptation reason* defines why the system has to be changed. Following the classification proposed in [118], we can distinguish the following types of adaptation:
correction/repair - to handle faults and managed problems reported during the execution of an SBA instance or discovered in the previous executions.

- Customization - to modify certain aspects of the system in order to fit to the situation, in which the application currently operates. This corresponds to the adaptation to the change in the application context, where context may have very general meaning, ranging from the set of available services and the properties, to the characteristics of the middleware or device, to user constraint and preferences, and to the parameters of environment (location, time, etc).

- Optimization - to improve certain (usually quality-of-service) issues and characteristics of the SBA.
- Prevention - to prevent and avoid future faults or undesirable situations in the execution of the system.

The *adaptation requirements* identify the aspects of the SBA model that are subject to change, and what the expected outcome of the adaptation process is. In these regards, we can speak about the functional SBA layer affected by the adaptation (i.e., service infrastructure issues, composition, or business process model); objective of the adaptation (i.e., certain QoS characteristics, behavioral model, user interface); whether only a particular application instance is changed or the whole application model, etc.

Accordingly, the *adaptation strategies or policies* identify the ways to achieve given adaptation requirements. Depending on the type of requirements, the adaptation strategy may speak about re-execution and backtracking, (re-)binding, reconfiguration, (re-)negotiation, re-planning, etc.

## 2.1 Adaptability requirements and SBA life-cycle

In contrast to classical SBAs, the distinguishing feature of the adaptable applications is the ability to accommodate and manage various changes at runtime. This, in turn, requires the capability to *observe* the changes relevant to the application execution, and the capability to *enact* the corresponding adaptation strategy. Figure 2 shows how the life cycle of service-based applications changes when adaptability comes into place. Not only applications can undergo the transition between the runtime operation and the analysis and design phases in order to be continuously improved and updated, but they also have intrinsic mechanisms that, during runtime, continuously and automatically a) detect new problems, changes, and requirements for adaptation, b) identify possible adaptation strategies, and c) enact them. These three steps are shown in the left hand side of the figure and lead to the deployment and provisioning of the modified application. The observation of the changes in the environment is obtained through monitoring (that is part of the management activities typically performed during execution). This is the trigger for the iteration of the adaptation cycle, whose effect is to inject changes directly into the application being operated and managed. This idea is also shared by autonomic computing [174] where a similar cycle is called MAPE (Monitor, Analyze, Plan, Execute).

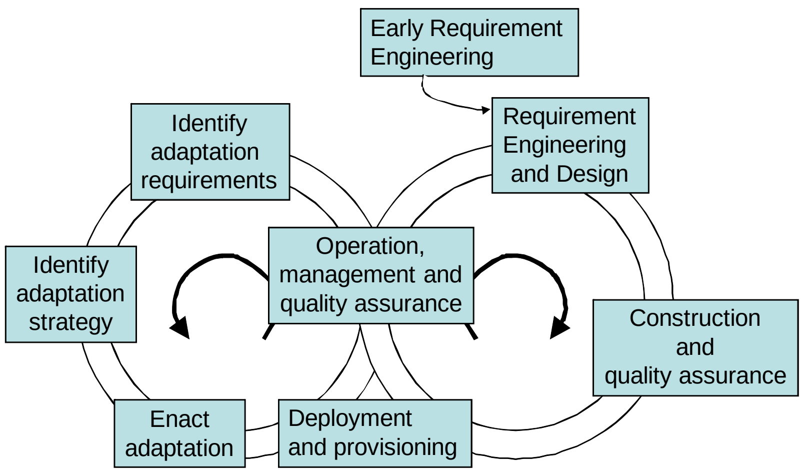

Early Requirement Engineering
Requirement Engineering and Design
Identify adaptation requirements
Identify adaptation strategy
Operation, management and quality assurance
Construction and quality assurance
Enact adaptation
Deployment and provisioning

**Figure 2. The adaptable service-based applications life cycle.**

In order to address the adaptability requirements in the provision and execution of the service-based systems, the operational platform, where the application is being deployed and executed, should extend the standard functionalities with the facilities that support the relevant phases of the life-cycle of the adaptable SBA.

Depending on the type of adaptation addressed by the platform, the set of extensions amounts to

- Monitoring, i.e., the components of the overall architecture that aim at observing the execution of the application and its environment in order to identify the relevant events and changes. This may include run-time monitoring of the execution and of the context, diagnosis of the detected faults in order to identify the origin of the problem, prediction of future problem and faults on the basis of current situation and historical information.
- Discovery, i.e., the components of the architecture that allow for identifying new candidate services to be used by the application. Depending on the requirements, such functionality may include various analysis techniques in order check the conformance of the candidate to the predefined requirements or to select the best option among appropriate ones.
- Negotiation used to provide a new set of requirements and constraints on the candidate services and their characteristics when the previous one could not be met.
- Adaptation, i.e., all the tools and methods available for enactment of the adaptation strategies, such as re-configuration, re-binding, re-execution, re-planning, etc.

## *2.2 Challenges for Engineering Adaptable SBA*

The adaptability requirements pose new challenges on the process of engineering of adaptable service-based applications. In particular, the engineering process should provide a comprehensive support for the adaptation-oriented design and specification, and the principles and approaches for integrating the adaptation-specific activities into the service-oriented architectures and platforms.

In the scope of the adaptation-oriented design novel modeling and specification approaches 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 adaptation strategies is 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.

In the scope of engineering service-oriented platforms a wide range of approaches and techniques concentrate on the ways to incorporate the adaptation-specific functionalities. The goal there is to develop principles and architectures for integrating different adaptation operations, such as monitoring/diagnosis, discovery/binding, and re-configuration/re-planning, into the overall SBA execution and management environment.

### 2.2.1 Approach adopted to survey the various contributions

In the following sections we survey the state of the art concerned with the various aspects that are related to engineering service-based applications. For each approach we will investigate the following aspects:

- its goals;
- the reason why we consider it important;
- the kind of contribution, methodological or technological;
- the context in which it is applicable;
- the key ideas behind the approach;
- its foundation theories;
- if it has been validated and how, i.e., through case studies, controlled experiments, arguments, simulation, analytical proof, action research.

Some of the approaches that we will consider have not been specifically developed for service-based applications. For these we will also explain why they are interesting in the context of the S-Cube project.

## *2.3 State of the art on engineering classical service-based applications*

For the purpose of the state of the art of engineering classical service-based applications, we only take into account approaches that deal either with the development life cycle or with the whole SBA lifecycle. SBAs can be perceived to be services themselves, and in this sense the methodologies discussed below can be equally applied to services and SBAs. Therefore, in the remainder of this section, the two terms will be used interchangeably.
Furthermore, most of these methodologies are generic enough to embody most of the characteristics illustrated in section 1.2.1. But at the same time, because of the difference in emphasis on different aspects of service lifecycle, they inherently promote some of these characteristics above others.
Apart from the works discussed below there are a number of commercial service lifecycle management methodologies, like that offered by ZapThink, CBDI forum, HP (Systinet), etc. that are usually parts of proprietary products. For that reason they are not included in the following discussion.

### 2.3.1 Web services development lifecycle (SLDC)

| **Synopsis** | |
|---|---|
| **Goal** | Service-oriented design and development, or web services development lifecycle (SLDC) is an iterative and continuous approach for the development, implementation, deployment, and maintenance of solutions as assemblies of services that are amenable to analysis, change, and evolution. |
| **Importance** | The methodology provides a series of models, best practices, standards, reference architectures, and run-time environments that are needed to provide guidance and support through all SOA development, deployment, and production phases. |
| **Kind of contribution** | Methodology |
| **Applicable to** | Artifacts: Services, Web services, and SBAs.<br>Functional layers: BPM, Service composition, Service infrastructure |
| **Phase where it applies** | All phases of the service development and operational life cycle, distinguishing them in eight distinct main phases and a preparatory one, encompassing planning, analysis and design, construction and testing, provisioning, deployment, and execution and monitoring of services. |
| **Key Ideas:** | Service lifecycle is depending on key SOA principles of service coupling, |

| | |
|---|---|
| | cohesion, and granularity. |
| **Foundations** | SLDC aims to incorporate the advantages of earlier successful development methodologies such as the Rational Unified Process (RUP), Component-Based Development (CBD), and Business Process Management (BPM), building on the valuable lessons learned by them and introducing new concepts, patterns, and practices. |
| **Validation** | - |

The methodology has been briefly mentioned at the beginning of Section 2. It emphasizes the characteristics of service coupling, and cooperation in computation in the form of service cohesion. Coupling refers to the degree of interdependence between services (as the basis for SBAs) – and the goal is to minimize it as possible in order to create self-contained SBAs without knowledge or relying on other SBAs. Cohesion is the degree of functional relativity of operations within an SBA – ideally, all the services and service operations offered by the SBA have to be strongly and genuinely related to another. Coupling and cohesion are inextricably associated with service granularity; that is the scope of the functionality exposed by the services of the SBA.
The phases of the methodology (figure 1) can be summarized by the following: Planning (phase 0) determines the feasibility, nature, and scope of the SBA within the bounds of an enterprise in order to achieve a service technology "fit" for its context. Analysis (phase 1) identifies the requirements for new applications, reviewing the business goals and objectives that drive the service development. Design (phase 2) requires modeling and defining of all the major service component interfaces available, considering both functional and non-functional characteristics. It produces a line of services to be developed from scratch (possibly incorporating pre-existing components) and a line of services to be assembled out of existing ones. Construction (phase 3) includes the definition of the SBA interface, the development of the implementation, and the definition of the implementation description. Testing (phase 4) looks for latent implementation failures in the services of the SBAs as they are integrated, configured, and run on different platforms. Provisioning (phase 5) calls for decisions on governance, certification, enrollment, auditing, metering, billing, and operation managing issues that define the behavior of the SBA in run-time. Deployment (phase 6) includes the publication of the service interface and service implementation definition. Execution (phase 7) deals with the execution of the SBA; Monitoring (phase 8) concerns with the management and monitoring of the services in order to gain a clear view of their behavior in their operational environment and address any possible deviations.

### 2.3.2 Rational Unified Process (RUP) for SOA

| **Synopsis** | |
|---|---|
| **Goal** | To provide development teams with a complete, open, modular, and proven environment for business-driven development of flexible SOA solutions [182]. |
| **Importance** | The methodology provides a foundation for business-driven development for SOA, supporting code-centric, visual programming, and model-driven style of development. |
| **Kind of contribution** | Methodology |
| **Applicable to** | Artifacts: Services, Web services, SBAs<br>Functional layers: Service composition, Service infrastructure |
| **Phase where it applies** | Takes into consideration all the development lifecycle in 4 phases (inception, elaboration, construction, and transition) and 9 workflows – group of activities (business modeling, requirements analysis & design, implementation, testing, deployment, configuration and change management, project management, and environment). |
| **Key Ideas** | Application of the RUP software development methodology in the SOA environment |

| | |
|---|---|
| **Foundations** | RUP, IBM Method (used by Global Services), UML, SOMA |
| **Validation** | Used in a number of projects. |

The rational unified process (RUP) aims to support the analysis and design of iterative software systems. As pointed out in [184], it adheres to the principles of OOAD and CBD at its foundation, and therefore it does not lend itself readily for adaptation to SOA principles. Nevertheless, RUP provides support for both the bottom-up and top-down development approaches, acknowledging existing elements and identifying architectural elements like components. Several of its milestones can therefore be appropriately adjusted and fitted in the context of SOA solutions [182], [183].
The methodology is comprised of:

- best practices for software engineering from a wide range of disciplines, including iterative development, requirement management, use of the component-based architectural paradigm, visual modeling of software, and continuous verification of quality
- process delivery tools that allow for integration with other s/w development tools and accessibility to developers
- configuration tools to meet the needs of different projects
- process authoring tools to extend or modify the methodology itself when necessary
- an existing community and marketplace of developers and solutions

By building on CBD, RUP for SOA lends itself for the creation of SBAs that exhibit the characteristics of standardization, autonomy, and coarse granularity.

### 2.3.3 Service-Oriented Modeling and Architecture (SOMA)

| **Synopsis** | |
|---|---|
| **Goal** | Method to support analysis and design of service-based applications |
| **Importance** | It has been proposed by IBM that is one of the main industrial players in this context. It is one of the few methods that focuses on engineering service-based applications starting from an understanding of the business goals and the processes in place in an enterprise |
| **Kind of contribution** | Methodology |
| **Applicable to** | Artifacts: Services, Web Services, SBAs.<br>Functional layers: BPM, Service composition, Service infrastructure |
| **Phase where it applies** | Requirement analysis and design phases. |
| **Key Ideas** | Structures analysis and design phases in three main steps, service identification, specification, and realization. In all steps there is a strong emphasis on the need for identifying and reusing the assets existing in the enterprise |
| **Foundations** | Processes, main concepts from service-based applications |
| **Validation** | IBM is using it in concrete cases |

SOMA is a method developed by IBM [187] to guide in the modeling (analysis and design) of SOAs. With respect to normal software development methods it introduces new and innovative techniques to specifically address SOA needs. In particular, it enables creation of SBAs by creating continuity between the business intent and the specific technical implementation.

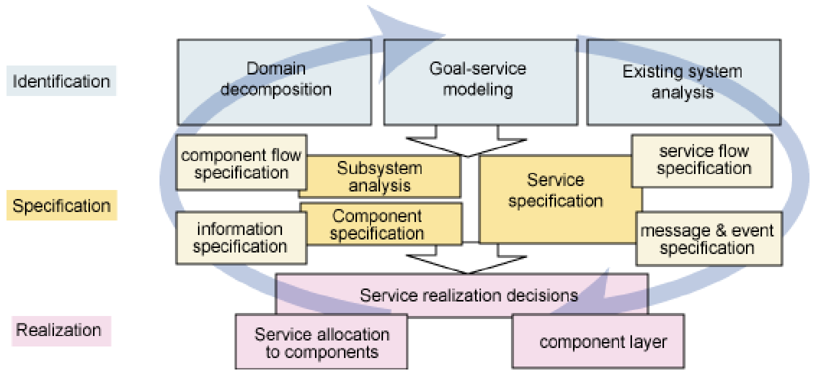


**Figure 3. SOMA Activities (from [187]).**

Analysis and modeling performed during SOMA is technology- and product-agnostic, but establishes a context for making technology and product specific decisions in later phases of the lifecycle. SOMA activities are structured in three phases (see figure 3):

1. The service identification phase aims at identifying services that are relevant to support the business processes of the enterprise. This phase can be performed following three approaches. In the top-down approach the business domain is decomposed into its functional areas and subsystems, and each process is decomposed into sub-processes and high-level business use cases. These use cases are often good candidates for identifying those business services exposed at the edge of the enterprise. In the bottom-up approach existing systems are analyzed and selected as candidates for the implementation of the underlying service functionality that supports the business process. APIs, transactions, and modules are analyzed and leveraged from legacy and packaged applications. In some cases, componentization of the legacy systems is needed to re-modularize the existing assets for supporting the services functionality. In the meet-in-the-middle approach goal-service modeling is used to validate and discover other services. In this activity, services are tied to goals and sub-goals, key performance indicators, and metrics.

2. The service specification phase aims at classifying services, analyze subsystems, and specify the details of the components that implement the services. In particular, services are classified into a service hierarchy, reflecting the composite or fractal nature of services that can be composed of finer-grained components and services. Such a classification helps in determining composition and layering, and in avoiding the proliferation of many services performing the same functionality. Subsystem analysis specifies the interdependencies and flow between the subsystems identified during domain analysis. Use cases become exposed services on the subsystem interface.

3. The service realization phase aims at assigning services and the components that realize them to the resources that are available. The software that realizes a given service must be selected or custom built. It could also be obtained from legacy systems by properly wrapping them. Other options include integration, transformation, subscription and outsourcing of parts of the functionality using Web services. Other realization decisions for services other than business functionality include: security, management and monitoring of services.

### 2.3.4 SOA Analysis and Design/Decision Modeling (SOAD)

| **Synopsis** | |
|---|---|
| **Goal** | To complement existing SOA analysis and design methods such as SOMA with techniques, architectural knowledge, and innovative tool support required during service realization. |

| | |
|---|---|
| **Importance** | The methodology describes a way to incorporate the historical context of a service realizing process (i.e. the decisions made for previous projects) with the best practices of the industry (expressed as patterns) in order to leverage the analysis and design decision making. |
| **Kind of contribution** | Methodology (micro-methodogy as referred in [186]) |
| **Applicable to** | Artifacts: Services, Web services, and SBAs.<br>Functional layers: BPM, Service composition, Service infrastructure |
| **Phase where it applies** | Requirement analysis and design phases. |
| **Key Ideas** | Utilization of architectural patterns and decisions, encoding of existing architectural knowledge to support design decision support. |
| **Foundations** | Methodologies and ideas from the fields of Enterprise Applications (EA), Object-Oriented Analysis and Design (OOAD) - especially RUP, and Business Process Management (BPM). |
| **Validation** | Applied to existing projects [185] which led to refactoring and additions to the methodology. Tool support and SOA decision tree that support it are also available . |

The SOAD project **Error! Hyperlink reference not valid.** aims to give SOA practitioners concrete, tangible service modeling and realization advice in a comprehensive and consumable way.
There are two central metaphors in SOAD: the architectural patterns and the architectural decisions. Architectural patterns describe proven solutions to recurring problem. Architectural decisions so far have mainly been used to document designs; in SOAD, their usage context is extended and made into an integral element of the design process [184]. The SOAD project is in the process of creating a reusable SOA Decision Model, which is structured according to Model-Driven Architecture (MDA) principles.
The methodology builds on the codification of existing knowledge in the form of architectural patterns and decision models to identify systematically the required decisions, give domain-specific pattern selection advice, and provide traceability from platform-independent patterns to platform-specific decisions [186].

### 2.3.5 ASTRO

| **Synopsis** | |
|---|---|
| **Goal** | To develop tools supporting the evolution and adaptation of distributed business processes during their lifecycle, from design to run-time. |
| **Importance** | The methodology addresses one of the major challenges for industry-wide adoption of Web services: the automated composition of distributed business processes, i.e. the development of technology, methods and tools that support an effective, flexible, reliable, easy-to-use, low-cost, and time-efficient composition of electronic distributed business processes. |
| **Kind of contribution** | Methodology |
| **Applicable to** | Artifacts: SBAs.<br>Functional layers: BPM, Service composition, Service infrastructure |
| **Phase where it applies** | Service-based orchestration life-cycle |
| **Key Ideas** | Structured requirement specification language creation, composition of services that realize complex and stateful protocols, incremental refinement of service specifications and code, and integration of composition, verification and monitoring. |
| **Foundations** | Service composition, iterative software development |

| | |
|---|---|
| **Validation** | - |

The ASTRO methodology for the design and development of service-based systems provides a set of integrated techniques and tools that cover the full life-cycle of (service) orchestrations. Namely, the methodology allows to automatically synthesize a new service that provide novel functionalities by orchestrating existing third-part services, on the basis of semantically univocal requirements that declaratively define the expected behavior of the code to be synthesized (from [188][189]). On top of this, the methodology provides verification functionalities that allow to automatically check the behavior of services, both at run-time [190] and off-line [191][192], so that it is possible to re-design services to either improve them, or fix possible faults.
The methodology covers the whole life-cycle of service-based systems, starting from their requirements specification, going through their realization and deployment, and finally to their execution, possibly looping back to re-design, triggered by the monitoring of unexpected/unwanted run-time behaviors.
While several approaches exist to support the composition and verification of services in specific settings (e.g. constraining to compose query-and-response services), the ASTRO methodology has several distinctive aspects:

- it relies on a structured requirement specification language that clearly splits logical and data-oriented aspects;
- it allows the composition of services that realize complex, stateful protocols, by orchestrating services which are as well multi-stage and stateful;
- it provides the ability to incrementally refine service specifications and the correspondent synthesized code;
- it integrates composition, verification and monitoring in a uniform framework, supporting the whole life-cycle of services.

Due to these aspects, ASTRO facilitates standardization, composability, reusability, and interoperability.
While the ASTRO methodology has been first developed with the aim to support the design and maintenance of services at the level of processes (expressed e.g. in languages such as BPEL), it is easily applicable to the modeling of business processes as well. Indeed, the current instantiations of the methodology are supported by powerful synthesis techniques and tools that can accommodate languages such as BPMN.
It has to be remarked that, at a first stage, the methodology had been developed to cope with the need of designing novel service composition directly from requirements; at a second stage, it has been extended to allow an iterative development approach, making it possible to proceed to incremental refinements of (initially raw) behavioral requirements.
Such enhancement is actually a first step towards the support for the development of adaptive services, where adaptation (in the requirements and in the resulting orchestration) can be triggered by evolving run-time conditions; this is one of the long-term targets of the methodology.

### 2.3.6 BEA Services Lifecycle (BEA)

| **Synopsis** | |
|---|---|
| **Goal** | To provide a detailed process for services management through the service lifecycle, from inception through to retirement or repurposing of the services. |
| **Importance** | The methodology takes an organizational-level viewpoint of the services lifecycle, introducing stakeholders, established best practices, and overall service governance to the services design, development and execution processes. |
| **Kind of contribution** | Methodology |

| | |
|---|---|
| **Applicable to** | Artifacts: Services, Web services, SBAs<br>Functional layers: BPM, Service composition, Service infrastructure |
| **Phase where it applies** | BEA covers the phases of the services lifecycle in three stages: requirements & analysis, design & development, and IT operations. |
| **Key Ideas** | Integrating service lifecycle management with service governance. |
| **Foundations** | BEA reference architecture [181] |
| **Validation** | - |

For each stage of the services lifecycle, the BEA methodology identifies [180] the actors (stakeholders) involved, the tools to be used, the expected artifacts (in terms of deliverables), the key considerations to be addressed, the recommended processes, and the best practices & requirements. The entire lifecycle is underlined by and built around the notion of service governance, i.e., the set of processes, tools, and organizational structures that is essential to deliver the SOA promise. Among the primary responsibilities of the SOA governance function are the publication of SOA standards and best practices for the community, the actual definition and execution of processes in the organization, the custody of the shared services across the enterprise or Line of Business (LOB), the propagation of standards and best practices across the organization, and the advertising of the SOA achievements within the organization.
Since this methodology focuses on the governance of services as the basis of SBAs, promoting the visibility in service lifecycle across the organization, it inherently contributes towards interoperability, discoverability, and to a lesser extent standardization of services.

### 2.3.7 Models and Tools for SOA Governance

| **Synopsis** | |
|---|---|
| **Goal** | Supporting services governance |
| **Importance** | Developing a generic model to represent service metadata, and providing tools using this model to manage service-based architectures |
| **Kind of contribution** | Method and Tools |
| **Applicable to** | Any kind of service |
| **Phase where it applies** | Various phases both design time and runtime |
| **Key Ideas** | Representing service metadata through an extensible model which highlights service dependencies, service proposals (for service specification), supporting service evolution and reuse through a browser tool |
| **Foundations** | Service-based architectures concepts |
| **Validation** | - |

By governance, [193] refers to the set of actions required for exercising control over a service-based system. These activities include change management, quality assurance, and standards compliance. The paper presents an approach for SOA Governance that deals with many of the phases of a service architecture lifecycle, including specification, deployment and evolution. This approach is particularly addressed to the financial context, but the presented concepts are claimed by the authors to be easily adaptable to many other contexts. The authors present a typical service lifecycle, describing the various stages of the process and the roles involved in each of them. Then, on the basis of this description, they introduce a *model* for the representation of the service metadata, that is, the information about the services deployed in the system. Eight different concepts are represented in the model, including service implementation, proposals, policies regulating the use of resources (access levels, provided quality), and so on. Proposals contain information about proposed client, service or

library and status information indicating whether the proposal has been accepted or not. Moreover, dependencies between services are explicitly represented by proper concepts in the model.
The paper also describes two tools, a Service Repository Console and a Service Browser, developed within the Eclipse platform. These tools allow for the description of services, their proposals and dependencies, and for browsing a service repository and analyzing service dependencies. Thus, for instance, the Repository Console can be used by developers to implement a new service, starting from an existing proposal or from scratch. During the development, the model of a service continuously evolves, exhibiting the characteristics of reusability, discoverability, and interoperability.

### 2.3.8 Summary of the presented approaches

The previous paragraphs discuss different approaches to non-adaptive service-based application engineering. All of the works presented above are based on the knowledge accumulated over the years in the software and systems engineering fields, and attempt to apply the best practices and theories from them to service engineering. Each of the approaches focuses on a different aspect of the service lifecycle (including that of the service development) and promotes in this way different service characteristics: SLDC provides a foundational framework to discuss service lifefycle issues, together with guidelines on how to manage the different aspects of the lifecycle. The BEA and RUP methodologies come from the software industry and as such focus on robust development and governance issues respectively. The IBM-originating methodologies of SOMA and SOAD address mainly the analysis and design phases, using a wide-scope approach in bridging requirements and design issues that takes into account previous decisions on that issues. ASTRO project emphasizes the compositional and iterative aspect of service development and lifecycle. Finally, the approach of Derler et al. discusses how service governance permeates the service lifecycle is presented.

## *2.4 Requirements Engineering for Service-Based Applications*

The aim of this section is to review requirements engineering approaches for service based applications. *Requirements Engineering* (RE) is one of the activities of the software development lifecycle, which deals with the identification and documentation of the purpose of the system to be [19]. To achieve this documentation, the requirements engineering process is usually split in two sub activities. *Early requirements engineering* aims to identify the system's stakeholders, their needs, concerns and goals in order to understand the problem the new system should solve. *Late requirements engineering* aims to identify the system's functions to understand the system to be itself [20].
A *stakeholder* in RE is any person or group who affects the system to be or who is affected by the system to be [21]. A stakeholder's *goal* is an objective, the system to be should achieve. "A goal under responsibility of a single agent in the software-to-be becomes a *requirement* whereas a goal under responsibility of a single agent of the environment of the software-to-be becomes an *assumption*." [22].
The section is organised in three parts. In section 2.4.1 we review RE approaches designed for the development of traditional software systems. In section 2.4.2 we review requirements engineering approaches, which are specifically designed for service-based applications. Finally, section 2.4.3 introduces a framework to classify existing RE approaches for SBAs, classifies these approaches and outlines further research directions.

### 2.4.1 Traditional Approaches to analyse and model requirements and context

The aim of this section is to briefly review requirements engineering approaches, which were designed for traditional software systems. The purpose of this section is twofold: First, many of the traditional RE approaches were further developed to meet the needs of requirements engineering processes of service based applications. Consequently, the approaches outlined here, are the basis for their extensions in the service domain. Second, the traditional approaches provide requirements engineering techniques for modelling and analysing the system’s context. These context-specific RE techniques have not yet been extensively discussed in the service domain.

#### *2.4.1.1 Structural Analysis*

The structural analysis belongs to the earliest approach to model system requirements. These approaches were put forward in the 1970ies and include the structured analysis [23][24][25] and the structured systems analysis [26]. These early approaches were extended among others by McManim and Palmer (essential system analysis [27]), by Youran (modern structured analysis [28]) and by Ashworth and Goodland (structured systems analysis and design method [29]).
The main focus of system analysis approaches is to analyse existing information systems to serve as a basis for their adaptation and/or extension or for the development of a new system. The approaches concentrate on a functional decomposition of the system by using data flow diagrams on different granularity levels. These data flow diagrams contain data flows, system functions, data storages and terminators, which represent the system boundary. Functions and data flows as well as data storages can be further described by mini specifications and data dictionaries.
Although structural analysis approaches have been successfully used in formulating requirements for software systems, their capabilities to model the system’s context are limited. Context in this approaches are only considered as terminators, e. g. entities which interact with the system. This interaction is explicated by data flows between the terminator and the system’s functions. In DeMarco’s Structured Analyses technique the context diagram is used to models these context aspects. This context diagram contains only terminators and a single function representing the software system as well as data flows between these terminators and the software system [25].

#### *2.4.1.2 Domain Analysis and Modelling*

Neighbors coined the term domain analysis in [30]. Currently the term is used in two different meanings. First, in the software product line community the terms domain analysis and domain modelling is used to describe the essential and the variability of a set of *software systems*. Second, in the domain specific modelling community the term is used to describe the (problem) *context* of a given software system.
According to the first meaning, domain analysis and modelling are essential aspects of the development of software product lines. The purpose of this approach is to describe a set of software systems regarding their commonalities and differences. The result of this requirements engineering activity is the documentation of the set of software systems with special emphasis on the documentation of the variability of the software product line [31]. FODA (Feature Oriented Domain Analysis) is one approach to address this problem [32].
According to the second meaning domain modelling can be regarded as context modelling (e.g. [29]). In this sense, the purpose of domain modelling is to document the problem domain to facilitate problem solving in this domain. Domain modelling disregards any system aspect and wherewith concentrates only on the context in which the latter system is embedded. Typical modelling techniques used in domain modelling include entity relationship models [33] and class diagrams from object-oriented approaches such as the Object Modelling Technique [34], the Object Oriented Analysis [35] and the Unified Modelling Language [36].
The concept of domain modelling has been refined in the domain specific visual modelling domain (DSVL). Researchers of this domain found that general purpose modelling grammars such as the UML are not equally well suited for all domains [37]. Consequently, new modelling grammars were put forward, which address the specific characteristics of a given domain [38]. These domain specific modelling grammars use domain specific constructs with precise semantics, which is grounded in the domain language vocabulary. A domain in this sense describes a class of real world proportions (e. g. embedded systems, banking information systems and air traffic control systems).

#### *2.4.1.3 Goal Modelling*

Goal modelling approaches were developed in the early 1990ies based on the observation that most existing techniques were not useful to support the early requirements engineering phases (e. g., [39][40][41]). These early phases are devoted to analyse the desires, visions and believes (intentions) of the stakeholders as a starting point for the in-depth analysis of the detailed requirements. The

analysis of these intentions is necessary to solve conflicts early in the development cycle and to validate the latter requirements against these intentions. A major aspect of all goal-oriented approaches is that they disregard any implementation aspects of the latter system.
The basic element of goal modelling is And/Or trees, which were initially developed by Simon [42]. The most prominent goal-oriented approaches include the i*-framework [40][43] based on the goal requirements language [44] and the KAOS (Knowledge Acquisition in Automated Specification [45]) approach.
The i*-framework is a framework for the identification, analysis and documentation of stakeholder (called actors in i*) goals. The dependencies between the stakeholders and between the stakeholder and the system are analysed in the strategic dependency model (SDM). The strategic rationale model – an extension to the SDM – documents the rationales of the stakeholders and their goals.
The KAOS approach also rests on the analysis of the stakeholder (called agents in KAOS) and their goals. The KAOS-object model describes these stakeholders and their goals as a tree. Responsibilities are assigned to these stakeholders, which are modelled in the KAOS-responsibility model. Two stakeholder types are distinguished in KAOS: human and software agents. Whereas human agents always belong to the system's context, software-agents may belong either to the system in focus or to its context.
As goal-oriented approaches such as i* and KAOS analyse the stakeholders and their intentions, they also address a specific aspect of the system's context. Knowledge about these stakeholders and their intentions are important context information for the development of software systems (e. g., [46][47]). The explication of the stakeholder's goals is important to validate system's requirements, to resolve the conflicts of the stakeholders and to achieve an agreement of the requirements [22].

#### *2.4.1.4 Scenario Modelling*

Goal-oriented approaches have been successfully applied in the early phases of the systems development process. However, a gap between the intentions of the stakeholders and the precise requirements of the system remained. Scenario-based approaches have been put forward to bridge this gap (e. g. [48][49][50][51]). *Scenarios* describe sequences of events, which lead to a defined result. For instance, a scenario may document a concrete interaction of a stakeholder with the system. Scenarios can be used at different levels of precision, detail and granularity [52]. Furthermore, scenarios can be used to document unwanted usage scenarios of the system [53].
A combination of goal-oriented and scenario-based approaches has proved beneficial in practice to elicit requirements [54]. Abstract intentional stakeholder information (e. g. goals and soft-goals) are refined with concrete scenarios. This refinement implies that the scenario either supports or objects the goal, with which it is associated [55][56][57].
Use cases are special types of scenarios. A use case describes a specific functionality of the system to be as interaction between the stakeholders and the system (interaction scenario). There are two approaches to use case modelling: use case diagrams and textual descriptions of use cases. Jacobsen proposed use case diagrams in [35], which were later integrated in the Unified Modelling Language (UML [36]) and the Unified Process [58]. Whereas use case diagrams show only coarse-grained information about the use case, textual use case descriptions are used to capture more information, e. g. quality aspects, actors, exception scenarios etc. [50][59][60]. These textual descriptions are usually structured with templates.
Rolland et al. distinguish between scenario types, namely system internal scenarios, system interaction scenarios, organisational context scenarios and organisational requirements scenarios [52]. Consequently, only the latter three scenario types are interesting for modelling the system's context. Use case diagrams for instance describe interaction scenarios and therefore document contextual information regarding the stakeholders and other systems, which interact with the system under development. However, use case diagrams are usually limited to the stakeholders, which directly interact with the system. This limitation can be overcome by completing use case diagrams with textual descriptions (use case templates), which specify the interaction between the stakeholders and the systems more precisely. In addition, these textual descriptions may also cover external events on which the system has to react. These external events represent another class of important context information.

Many approaches were put forward to extend the expressiveness of use case diagrams. Bourdeau et al. for instance propose to use use case diagrams with UML collaboration diagrams to describe the interaction within the system and between the system and its stakeholders more precisely [61]. Other extensions include extensions to use case templates [62], techniques to elicit the system boundaries and its stakeholders [63] and extensions to model virtual actors to represent sources of system inference [64].

#### *2.4.1.5 Problem Frames*

Jackson proposed in [65][66] his "world-and-the-machine" model to structure the system development phases according to two different views. The goal of this separation was to facilitate a construction of the software system, which is adequate to the problem at hand (Figure 4).

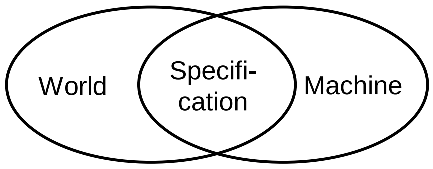


**Figure 4: World-and-the-machine model**

In this model the requirements of the system to be are derived from the context (world). These requirements represent a problem. The problem's solution is realised as a system (machine). The interface between the world and the machine is the specification of the system, e. g. its functional and qualitative characteristics.
However, as Swartout and Balzer argue, the detailed elicitation of requirements is impossible when some elements of the system (the solution to the problem) are not already understood [67][68]. Consequently, Halls et al. proposed to combine the problems frames approach with requirements co-desgin approaches [69]. This approach includes an iterative process of problem analysis and problem solving. This process is combined with the problem frames approach by using these problem frames for the refinement of an initial solution.

#### *2.4.1.6 Viewpoints*

Viewpoints approaches (e. g., [70][71][72][73]) were proposed upon the believe that the requirements of a system cannot be elicited and documented in a single view. Consequently, the requirements analysis was split according to different viewpoints. Each *viewpoint* represents a specific view on the system and usually corresponds to a certain stakeholder group. The separation of the problem space into different views allows to reduce the complexity of the problem and to concentrate on particular aspects of the problem. Consequently, These viewpoints allow to elicit requirements more completely and to facilitate their analysis and validation [73].
After the analysis of the requirements according to these viewpoints the partial models are integrated to gain an overall picture of the system to be. This integration is an essential aspect when working with viewpoints [74]. The integration of the different viewpoints is necessary, because the viewpoints are not necessarily disjoint. This means, that some information may be visible in more than one viewpoint. Consequently, the integration of all viewpoints fosters the internal consistency of the system's requirements (e. g., [74]).

#### *2.4.1.7 Context Facets*

Another approach to reduce the complexity of the context analysis is to split the context into different facets. Mylopoulos et al. for instance propose to use four facets (called worlds in [75]). The subject facet describes the domain in which the system is embedded, e. g. the organisational environment. The system facet describes at different abstraction levels what the system does. These levels may include functional and non functional requirements as well as designs and implementations. The usage facet describes how the system is used by whom. The development world focuses on the influence of the software development process and including the project management on the software systems.
Jarke and Pohl put forward slightly different facets. These facets are based on the observation that a software system needs to represents objects of the environment (object facet similar to Mylopoulos et

al.'s subject facet), that the system is embedded into a technical environment (system factet) and that the system is used by some stakeholders and/or other systems (usage facet identical to Mylopoulos et al.'s usage facet). The development facet (similar to Mylopoulos et al.'s development facet) covers the observation that the context of the software development process influences the requirements of a system [76]. For instance, in a software development project it may be necessary to remove existing requirements due to resource problems.
Another extension of the four facets by Jarke and Pohl was put forward by Pohl [77] to foster a more complete requirements elicitation. The author proposes to distinguish not only facets but also four context aspects. These context aspects include requirements sources (stakeholders, documents, other systems), context objects (persons, material and immaterial objects) and properties and relations of these context objects.

#### *2.4.1.8 Applying Traditional RE Approaches for SBAs*

The traditional RE methods and techniques has been applied to the service domain. As evident from section 2.4.2, there are two trends of this application: First, the fragments (e. g. models) of the traditional RE approaches are used as input for a service-specific RE technique. For instance, the SeCSE approach uses use cases to search for services and to refine these use cases by analysing the search results (section 2.4.2.1). Second, traditional approaches are extended and refined to meet the RE characteristics of service based applications. To this end, goal modelling techniques such as I* have been refined to model quality of services, service compositions and to engineer SBAs (section 2.4.2.2). The Map approach, usually applied to model scenarios, was extended to model service compositions and orchestrations (section 2.4.2.3).
In addition to these actual usages of traditional RE approaches in the service domain, the approaches to context analysis and modelling could be used to model SBAs, which adapt to the system's context.

### 2.4.2 RE Approaches for SBAs

In this section we investigate RE approaches, which are specifically designed for service-based systems. We organise this review according to three research streams: First, in section 2.4.2.1 we discuss the requirements engineering approach developed in the SeCSE project. Second, we review the approaches, which apply Tropos – a requirements engineering approach – to the service domain (section 2.4.2.2). Third, we review the application of the Map modelling grammar to the service domain in section 2.4.2.3.

#### *2.4.2.1 The SeCSE Approach*

| **Synopsis** | |
|---|---|
| **Goal** | Aligning and refining requirements to the available services |
| **Importance** | Facilitating the elicitation of more complete requirements, facilitate using existing services. |
| **Kind of Contribution** | Method + Tool |
| **Applicable to** | Service Discovery |
| **Phase where it applies** | Requirements Engineering |
| **Key Ides** | Create an initial requirements specification, transform this specification into a registry query and use the query results to refine the specification. |
| **Foundations** | Requirements specification languages, Extended Service Registries, Natural Language Processing (handling synonyms and homonyms with WordNet) |
| **Validation** | Action Research + Case Study |

The Service Centric Systems Engineering (SeCSE) approach to requirements engineering of SBAs is based on the idea that service provision and the requirements of a company should be aligned that is a company should make the best out of the current service provision. To do so the approach includes an incremental refinement of an initial set of requirements. This refinement is achieved by exploiting existing service descriptions and by using them to refine the requirements specification
The RE process in the SeCSE approach starts with an initial set of requirements specification (Figure 5). This specification is than automatically converted into a query, which is sent to a service registry. The registry processes this query and returns the services which fit the service query to the requirements engineer. After the services were found, their service descriptions are analysed. Upon this result the requirements engineer has three options: First, the requirements engineer may decide to incorporate the service descriptions into the initial requirements specification and thereby extend and/or refine it. Second, the requirements engineer may decide to reformulate the requirements specification to retrieve a broader range of services for his/her query especially in the case that no services are discovered for the initial requirements specification. The third option is to end this process. This option will especially used when a service query based on a refined requirements specification delivers the same service than the one executed for the initial requirements specification.

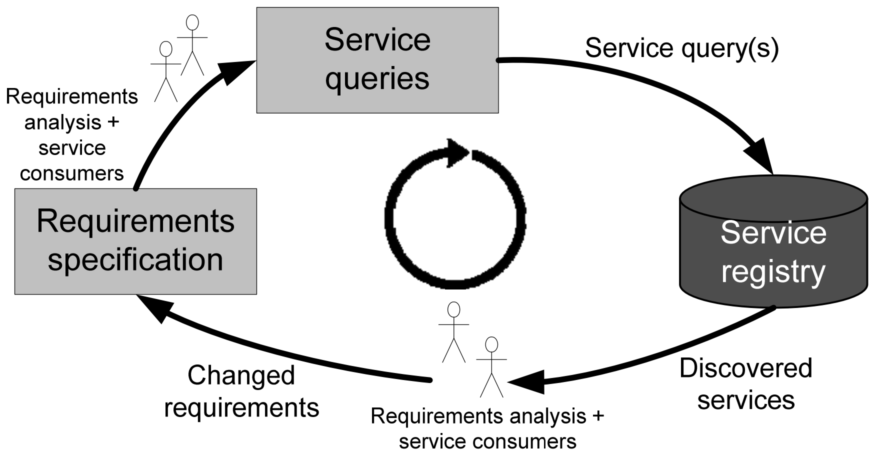

Service queries
Service query(s)
Requirements analysis + service consumers
Requirements specification
Service registry
Changed requirements
Requirements analysis + service consumers
Discovered services

**Figure 5: RE Process proposed in the SeCSE-Project [78]**

In addition to the alignment of the provided services to the business needs, the authors argue, that their approach facilitates four different forms of creative thinking including analogical matching by transferring knowledge from one domain to another, pattern matching by providing means to locate service which fulfil the business'es needs, random retrieval by discovering random services and constraint removal by step-wise relaxing constraints to being able to find a broader range of services [78], p.111.
One challenge in this approach is the automatic transformation of the requirements specification into a query suited for a service registry. The challenge includes resolving conflicts, which are very similar to the conflicts analysed in the model comparison literature. Especially, the approach has to deal with naming conflicts, e. g. homonym and synonym conflicts. These conflicts arise because the natural language used in the requirements specification and in the service description is differently used by different people (e.g., [79], p. 653). In addition, the transformation needs to handle abstraction conflicts, e. g. to detect when a service description and the requirements specification addresses the same problem on different granularity levels.
Especially the naming conflicts were resolved in two ways: First, the transformation engine, which creates the service query, uses WordNet [80] for word disambiguation. This disambiguation resolves synonym conflicts by expanding the query with synonyms and by resolving homonym conflicts on the basis of the definitions (senses) provided in WordNet [81], p.148. The second approach includes a auto-completion feature, which helps the requirements engineer when formulating the requirements specification [82], p.64. While word disambiguation resolves naming conflicts after their occurrence,

auto-completion limits the use of natural language expressions and therefore, reduces naming conflicts in the first place.
The approach was validated by using use cases to specify the requirements. This validation included two rounds: In the first round, researchers and industry experts worked together with the tool developed in the SeCSE project to discover requirements for an in-car route planner (action research). Industry experts ranked requirements formulated prior to the service query higher than those resulting from theses queries. However, the experts ranked the requirements found from service discovery higher than those initially elicited. This indicates that these new requirements would have not been found with traditional RE techniques [81], p.155, [83], p.15. In addition, Zachos et al. concluded that the approach is capable to discover relevant services based on use-case descriptions and that the resulting services could be used to refine the initial requirements.
A second analysis was carried out by two independent practitioners who used the tools provided by the SeCSE project to find services and to refine requirements for a travelling agent's time and expenses accounting system (case study). The practitioners found that over half of the services discovered by the search engine were incorrect [84], p.175. In addition, the practitioners had difficulties to map the requirements to the available services since the requirements were coarser grained then the services provided (abstraction conflict). In addition it was difficult to refine the requirements based on the service query results, which was mainly due to the tool implementation [84], p.177. On the positive side, the practitioners found in this case studies new requirements, which were not previously elicited and improved pre-existing requirements [84], p.177.

#### *2.4.2.2 The Tropos Approach*

The Tropos approach was developed to support the early systems development phases such as early and late requirements engineering, architectural design and late design (e. g., [20][85]). Previous approaches to requirements engineering apply concepts, methods und tools mainly used in the design and programming domains. On the contrary, Tropos uses RE approaches such as goal and scenario modelling to describe a system specification [20], p.365.
Tropos uses a similar approach than i* [43] to elicit early and late requirements. In line with the i* approach, the main actors, their goals, soft-goals, plans and resources are identified. These elements comprise the actor model (strategic dependency model in i*). *Actors* in this sense refer to organisational units, roles, positions or other systems. *Goals* represent the actor's strategic interests. Two types of goals are distinguished in Tropos – hard-goals and soft-goals. In contrast to hard-goals, soft-goals do not have a "… clear-cut definition and/or criteria for deciding whether they are satisfied or not." [85], p.207. *Plans* represent activities carried out by some actor to satisfy goals or to satisfice soft-goals. Resources are informational or physical entities which are consumed or produced during the execution of a plan [85], p.206.
The goal model (strategic rationale model in i*) is used to derive late requirements in Tropos. Goal models describe an insight view of a single actor. This insight view reveals the structure of the actor's goals (goal decomposition), plans and resources. It is used to evaluate different alternatives of satisfying a certain goal. Tropos defines three types of links: decomposition, contribution and means-end links. *Decomposition links* are used to decompose goals using an And/Or structure. An And decompositions means that all sub-goals of the decompositions must be satisfied to satisfy the root goal. In an Or decomposition it is sufficient that at least one sub-goal is satisfy to also satisfy the root goal. *Contributions links* connect soft-goals with soft-goals, hard-goals or plans. They represent the degree to which the soft-goal, hard-goal or plan contributes to satisficing the soft-goal. *Means-end links* connect plans with plans, soft-goals or hard-goals. They represent that the plan (end) implements realises the plan, soft-goal or hard-goal (mean) [85], p.208.
The system's architecture is derived in three steps: Firstly, new actors are defined, which are responsible to execute the system's plans. Secondly, capabilities, which are needed to fulfil the system's goals and to execute its plans, are defined. Lastly, new actors are defined and the capabilities are assigned to these actors [85], p.214.
The detailed design is derived by attaching scenarios to the strategic rational models. More, activity diagrams are used to visualise specific interaction scenarios and state charts are used to specify the internal processes of an actor. These scenarios are complemented by capability and plan diagrams.

In the service domain, Tropos has been applied for designing web service [86], for reasoning about quality of service aspects, for deriving service requirements and to verify web services according to an initial set of requirements. These approaches are described below.

**Quality of Service Requirements**

| **Synopsis** | |
|---|---|
| **Goal** | Find services, which best fulfil given goals (top down), effects of using service on goal fulfilment |
| **Importance** | Services depend on each other to fulfil some goal; alternative services can be used to fulfil some goal |
| **Kind of Contribution** | Technique |
| **Applicable to** | Services |
| **Phase where it applies** | Requirements Engineering |
| **Key Ides** | Relate service to Tropos' plans; apply reasoning mechanisms to goal models |
| **Foundations** | Tropos |
| **Validation** | None |

Aiello and Giorgini explore quality properties of services (quality of service, QoS). According to their arguments, QoS is important because of three different reasons: First, different services may work together to jointly fulfil some task. This collaboration requires that each service is aware of the quality of the cooperating services. Second, service may compete with each other. In this case, the service consumer needs to know the quality of service to decide, which service to use. Third, quality of service aspects are also important for service providers, as they may provide the same functionality with different qualities (e. g. different response times, [87]), p.21.
To model quality aspects of services, Aiello and Giorgini apply Tropos' goal modelling techniques to the service domain [87]. In particular the authors use the goal model. In addition to the goal model specified in Tropos the author use qualitative and quantitative contribution links. In qualitative contribution links means that a plan contributes strongly positive, positively, negatively and strongly negative to a soft goal. A quantitative contribution link is used to express for instance that a plan satisfices a soft-goal by 60%. (cf. Figure 6).

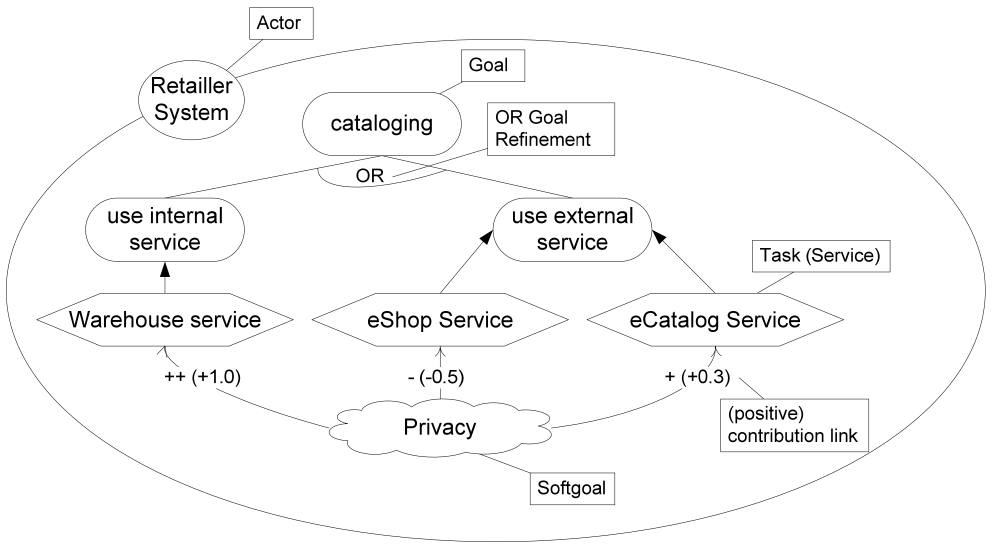


**Figure 6: Strategic Rational Model (example taken from [87])**

The basic idea behind the application of Tropos for the service domain is the mapping of the Tropos plan to a service. This mapping means that a given service fulfils a certain task completely. The quality of a service can then be represented by soft-goals connected via contribution links to this service (cf. [88] for similar arguments).
Reasoning in a Tropos goal models can be applied bottom up or top down [87]: First, one can ask how a given set of services influence goal achievement, that is, how can the decision to use a set of services be propagated towards the root of the goal hierarchy (bottom up; forward reasoning). Second, backward reasoning (top down) allows to find a set of services, which fulfil the given goal structure best while having minimal costs. The algorithm implementing this forward and backward reasoning is described in detail in [89]. Consequently, the approach provides useful insights into how the usage of services (and their related quality of service aspects) influences the goal achievement.
Since Tropos was not especially constructed to support requirements engineering for service based systems, the approach by Aiello and Giorgini is applicable to any Tropos project. However, in the service domain, the approach can be used to reason about quality of service aspects (represented by soft-goals). Consequently, the approach can be used for describing quality requirements for services from a consumer's perspective. The proposed approach has not been validated in the service domain. However, the performance of the underlying algorithm for forward and backward reasoning was validated in experiments [89].

**Developing SBAs**

| **Synopsis** | |
|---|---|
| **Goal** | Developing Service Based Applications with Tropos |
| **Importance** | |
| **Kind of Contribution** | Method |
| **Applicable to** | Engineering Service Based Applications |
| **Phase where it applies** | Requirements Engineering and SBA design |
| **Key Ides** | Align tasks in the strategic rational model with services. |
| **Foundations** | Tropos |

| | |
|---|---|
| **Validation** | None |

Penserini et al. proposed in [90] is an extension to the Tropos process model to engineer service based applications. The approach relies on assigning service consumers to actors, which interact, with the SBA and service providers to actors, which represent the system. Each actor in turn has an ability (a plan) to execute an assigned task and thereby to satisfy a goal and/or to satisfice soft-goal. This fulfilment relationship is modelled in Tropos as contribution links between tasks and goals or soft-goals [90].
On this basis a *capability* is any relation between a goal and a plan including the plan's contribution links and a type. This capability is used to describe services. The capability's type can either be independent, dependent on some actor or unknown. Independent capabilities are provided by the SBA itself. Dependent capabilities are provided by the actor on which they depend. This actor is always outside the SBA and therefore, the SBA consumes these capabilities. Capabilities with the type unknown are not yet assigned to any actor. These capabilities can either be assigned to an existing actor to a new actor or be further decomposed [90].
The designer chooses competing sets of capabilities according to the SBA consumer's initial service needs. These *initial service needs* are a set of functions assigning soft-goals to one goal. In principal this selection of capabilities can also be done at runtime.
The authors build upon the usual Tropos' design process. Upon this process Penserini et al. propose to further analyse the capabilities. The list of all capabilities is built using all goal-plan dependencies for each of the SBA's goals. To these goal-task dependencies the contribution links of the tasks and their respective soft-goals are added. Then the designer needs to decide about the type of each capability (independent, dependent, unknown). The designer can either assign the capability to the SBA, to an existing external actor, to a new external actor or to further decompose the capability's goal and to repeat the actor-assignment recursively. After the strategic rational model is complete and all capabilities are properly assigned, the designer chooses the capabilities, which fulfil the initial service needs best.
Although the approach allows to systematically model service capabilities it does not allow to model the execution order of these services, e. g. the workflow [90]. This deficiency is complemented by the approach described in the next section.

### Deriving Service Compositions from Goal Models

| **Synopsis** | |
|---|---|
| **Goal** | Deriving service compositions by refining Tropos models. |
| **Importance** | The approach assures that the service compositions formulated in BPEL4WS are in line with the requirements formulated in Tropos models. |
| **Kind of Contribution** | Technique |
| **Applicable to** | Service Composition |
| **Phase where it applies** | Requirements Engineering and composed service design |
| **Key Ides** | Enrich formal Tropos models with BPEL4WS code |
| **Foundations** | Tropos and formal Tropos |
| **Validation** | None |

The central idea of the approach outlined by Pistore et al. is to refine requirements expressed in Tropos models into workflow models (expressed in BPEL4WS), which can be executed on a virtual machine

[91]. Since BPEL4WS can be used to describe service compositions, the approach by Pistore et al. is used to derive composed services, which comply with the requirements specified in the early phases of a Tropos project.
To achieve the definition of a service composition, the initial tasks in the goal model are step-wise refined until they can be used to define a workflow. After this refinement, the model is complemented by using the formal Tropos-specification [92]. Formal Tropos allows specifying the elements in the goal model in an object-oriented manner. The outer layer of this specification is similar to a class definition in an object-oriented programming language. The inner layer is used to define constraints of the refined model element throughout its lifetime [92].
Once the elements of the goal model are refined using formal Tropos, the workflow can be generated as BPEL4WS specification. “For instance, it is possible to automatically generate the definition of messages, ports and services for the business domains …” [91]. This definition of the messages, ports and services has to be complemented by defining the “body” of the workflow. This refinement needs to be aligned to the plan, to which this workflow is mapped.
After the definition of the workflows, it can be formally ensured that the workflow specification actually corresponds to the requirements described in the goal model. To achieve this formal verification it is necessary to define constraints on the model and its formal counterparts. These constraints can be formulated using existence operators (keyword possibility) and all operators (keyword assertion, [91]). These formal verification techniques can also be used to check an evolved workflow model according to the requirements formulated in Tropos’ goal model.

### *2.4.2.3 The Map Approach*

| **Synopsis** | |
|---|---|
| **Goal** | Goal-Driven elicitation of services and their composition & orchestration. |
| **Importance** | The approach assures that the business requirements are closely aligned with the services provided. |
| **Kind of Contribution** | Technique |
| **Applicable to** | Service discovery, service composition, service orchestration |
| **Phase where it applies** | Requirements Engineering |
| **Key Ides** | Relate Map strategies to services. |
| **Foundations** | Map modelling grammar |
| **Validation** | Multiple Action Research Projects, no service-specific validation. |

The Map approach provides a process modelling grammar, which allows to model process structure rather than control flows. Such process structures allows to dynamically derive concrete business processes from the Map models and, thus, to address the changing business processes of a company. Consequently, the approach also enables a dynamic process construction by deriving any process model fitting the process structure on the fly [93]. This approach shares many similarities with the Semantic Object Modelling method (SOM) described by Ferstl et al. [94].
The concept of a Map was introduced by Rolland et al. in [95]. A Map is a directed graph containing intentions *I* as nodes and strategies *S* as edges (Figure 7). This graph describes the structure of a process leading from *START* to *STOP*, which represent to specific intentions. A strategy is an edge from an initial intention $I_1$ to a target intention $I_2$. It represents one way to achieve intention $I_2$. The triple <$I_1$, $I_2$, $S$> is called a section of a Map. Sections with identical intentions but different strategies are called multi thread strategies (strategies $S_2$ and $S_3$ in Figure 7; [95]). When an intention $I_2$ can be

reached from another intention $I_1$ by using different strategies, this is called a multi path (strategies $S_1$, $S_2$, $S_4$ and strategies $S_5$, $S_6$). Generally, a map from its START to its END is a multi path.

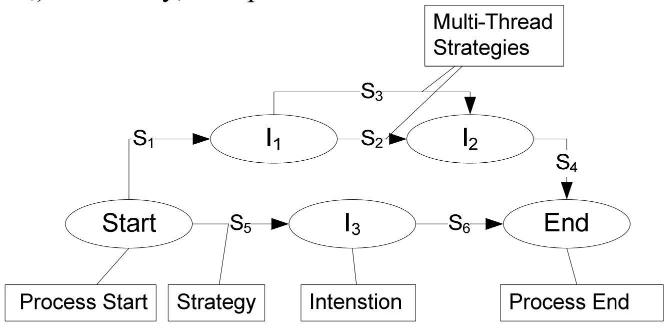


**Figure 7: Example of a Map model**

The Map modelling grammar is enhanced with guidelines. These guidelines allow the requirements engineer to understand how an intention can be achieved (intention achievement guidelines), how to select between different strategies (strategy selection guidelines) and how to select among different intentions (intention selection guidelines, [95]).

The Map approach has been successfully used in the method engineering domain to describe method engineering processes (e. g. [96][97][98]) and to align organisational needs to Enterprise Resource Planning (ERP) functionality (e. g. [93][99]). The latter application is also interesting for SBAs, because one of the requirements engineer's task is to align the business processes to the services available. The fitness relationship between the organisational needs and the ERP system functionality is achieved by assigning each section of the map to an ERP functionality (e. g. to an ERP component) so that the interface of this functionality is the target of this section [99]. Using a map in this adds another facet to it. While the original facet describes intentions and strategies the new facet describes application systems [93].

The authors argue that assigning ERP systems functionality to a Map facilitates understanding between stakeholders, enables the discussion of using alternative business strategies and ERP solutions by discussing multi-threads and multi-paths in the Map and to fully exploit the technology available by using paths in the Map, which have not been considered relevant before [99].

More recently the authors of the Map approach applied the principle uncovered in the ERP research to the service domain. Initially this has been achieved by exchanging ERP functionality for service. Rolland and Kaabi argue that the behaviour of a service can be described by the pre- and post-conditions associated with the initial and target intentions of the section [100]. The authors argue that modelling services with a Map leads to better understanding of the business processes and its related services among the stakeholders. In addition, they argue that service discovery should be based on business goals (expressed as intentions) rather than on the basis of IT functionality [100].

In another paper Kaabi et al. [101] apply the Map approach to derive service compositions and service orchestrations. This approach rests again on the idea of relating the Map's intentions to the services available. In their paper the authors propose a five step process model to derive service compositions and orchestrations [101]: The first step is the identification of services by modelling a Map model and by interpreting strategies as services (more precise as service needs). The second step is the identification of service providers (labelled key playing actors), which may provide service for the identified strategies. In the third step legacy systems are identified, which can be converted into a service. In step four these services are distributed among the service providers and the fifth step is deriving the orchestration (Figure 8).

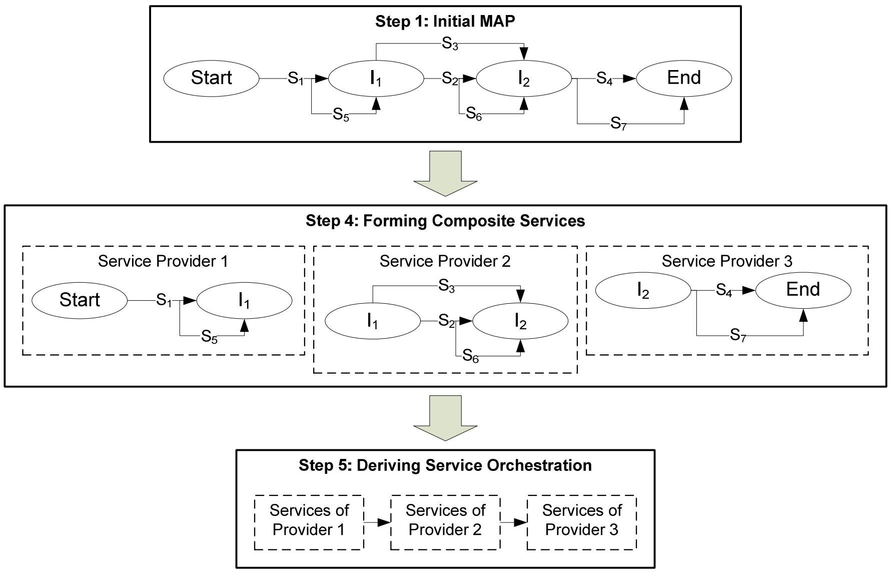


**Figure 8: Example of the process model described in [101].**

The decision of the service distribution among the different service providers results in splitting the initial Map model (produced in step 1) into sub-models for each provider (step 4). Each provider in turn provides a composed service including all strategies and intentions from his/her Map model [101]. Each service provider can use the multi-path and multi-thread topologies to dynamically use the service, which best fits the runtime situation. Upon this service composition the service orchestration can be derived based on the dependencies between the distributed Map models.

The authors argue that their approach minimises the gap between the requirements and service provision, facilitate changes, which can be easily introduced in the Map models and may be propagated immediately to the service based application.

The Map has been successfully used by the authors in multiple action research projects (see also the references provided in [93][99][102]). An evaluation of the approach in the service domain and especially for the interesting parts of service composition and orchestration was not undertaken so far.

### 2.4.3 RE for SBAs

Eck and Wieringa were among the first who analysed the impact of SBAs on RE processes and tools. The authors argue in [103] that the rational for using service oriented architecture from a business viewpoint is the economies of scale. Since the service provision of an enterprise are bundled at the *service provider* (usually the IT department), the service provider gains detailed insights in how to offer and operate services. These insights can be used to reduce the costs of service provision. In this view the goal of the service provider is the "… availability of a service infrastructure." [103].

The goals of the consumers in a service-oriented architecture include the support of their business processes by software systems. A *Business processes* in this sense is a set of activities and their dependencies, which produce a value for some customer [104]. In a service oriented architecture the business processes are not supported by traditional software but by services offered by some provider. Since service providers and service consumers have inherently different goals, requirements engineering approaches for SBAs should reflect this distinction (Figure 9). Consequently, we distinguish between RE for service providers and RE for service consumers as follows [103]: RE for service consumers is the continuous task of aligning the available services to the company's requirements. This task includes not only the identification of missing services and to specify their requirements but also the identification of service, which closely match these requirements. In the latter case the requirement engineer's task includes the adaptation of the requirements and possibly the business processes according to the offered services.

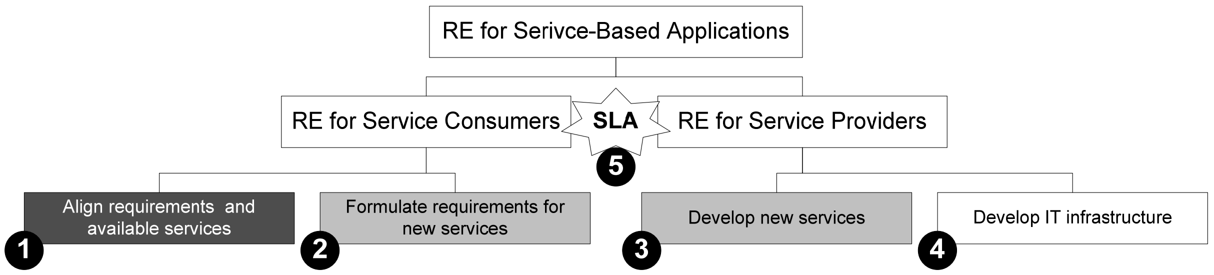

RE for Serivce-Based Applications
RE for Service Consumers
SLA
RE for Service Providers
5
Align requirements and available services
1
Formulate requirements for new services
2
Develop new services
3
Develop IT infrastructure
4

**Figure 9: RE Perspectives for SBAs**

The tasks of the requirements engineer at the provider side are twofold: First, the requirements engineer is responsible for designing the IT infrastructure. Second, the requirements engineer is also responsible to develop new services (in close cooperation with service clients), which can than be offered to service consumers.
The consumer and provider side requirements engineering processes are coupled with the service level agreement. Consequently, a service level agreement is the result of an RE process whenever new requirements are formulated by service consumers or new services are developed by service providers [105][106]. In addition, Lichtenstein et al. point out that, when the provider's infrastructure concerns are disregarded, this leads to reduced performance of the service oriented architecture as the whole because providers may compensate exaggerate service provision expectation by reducing the actual service provision [106]. The immediate consequence of this statement is that the service level agreement (SLA) must be carefully negotiated and agreed between service consumers and service providers.
Although the importance of SLAs for the requirements engineering has been recognised, Trienekens et al. [105] found a number of problems with current SLA specifications including unclear and incomplete service specifications, insufficient cost management and too technical SLA documents. Consequently, RE approaches for service based applications should especially concentrate on how to document and negotiate SLAs.
Consequently, requirements engineering of SBAs differ in three ways from traditional RE: First, RE approaches for SBAs include the continuous task of aligning the consumers' business processes to the available services. Second, on the provider side, RE approaches need to address how to design an infrastructure for the service provider. Third, when requirements for new services are raised by the consumer or new services are developed by a provider the task of the requirements engineer is to guide the creation and negotiation of the SLA.
The SeCSE approach clearly deals with the alignment of available services and the requirements (1). However, the support for adjusting requirements in accordance to the service provision is limited. This support is realised only by the refinement of the requirements specification according to the search results of the service query. In addition, the respective validation showed that this refinement is difficult to achieve.
Based on the given validation, the SeCSE approach is valuable for discovering and refining existing requirements and for aligning existing services to business processes. Consequently, the approach clearly contributes to the requirements engineering for service consumers (❶ in Figure 9). The Tropos approaches can be used to align existing services to requirements (❶ in Figure 9) and to derive requirements for new services (❷ in Figure 9). In addition, especially the proposal, which was put forward by Pistore et al. in [73] can also be used to develop new services (❸ in Figure 9). The Map approach was specifically designed to align requirements expressed as business processes to existing IT functionality (❶ in Figure 9). The approach could also be used to support the reverse direction. In this sense the requirements engineer uses the Map constructed of the provided services to derive business processes. The Map approach also supports formulating requirements for new services (❷ in Figure 9). However, this support is limited to a service description in form of a Map section.
From this investigation it can be concluded, that none of the investigated approaches deals with the development of an IT infrastructure for service providers (❹in Figure 9). In addition, no approach explicitly addresses the development and negotiation of service level agreements between service provider and consumer (❺ in Figure 9). Both aspects are subject to further research.

Given the characteristics of SBAs outlined in Section 1.2, the RE approaches require that services are reusable, discoverable, composable and interoperable. Since the SeCSE approach aims to discover services for a given set of requirements, it assumes that the services can be re-used and can be discovered. In addition, the authors assume that the services found in this process can work collaboratively, which requires service interoperability. The approaches by Pistore et al. as well as the Map approach assume that the services can also be composed. The remaining characteristics of SBAs were not found in current RE approaches. Consequently, it is subject to further research to find whether they can be exploited as well.

## *2.5 Design for Adaptation*

The traditional design and engineering approaches are not enough to enable the development of highly flexible service-based applications that are dynamically assembled or dynamically reconfigured in order to accommodate themselves to critical events and situations,. The application development process should incorporate the facilities to meet the adaptation requirements from very early phases to the application execution. This requires languages and modelling approaches that are capable of representing adaptation-specific aspects, i.e., how the application will react to the relevant conditions and events, how it should adapt itself to new situations.

This dynamism may be caused by wide range of factors. First, the service-based applications rely on using services that are not under the control of the service integrator. Consequently, the services and their parameters are subject of unpredictable changes: services may disappear; their signatures and behaviour may change, as well as the quality-of-service characteristics. Such changes may trigger the faults and errors in the execution of the application, may lead to bottlenecks in resource allocations. In the former case, the system should be able to recover from the failure situation and return to a normal execution (self-healing systems [107]). In the latter case, reconfiguration actions are needed in order to better satisfy the QoS requirements [108].

Second, service-based applications are often designed with a goal to address wide range of service consumers, having their own preferences, constraints, service-level agreements and requirements. That is the application should dynamically adapt to the variations in these properties (customization or personalization).

Third, the dynamism in the application execution is often subject to changes in the application context. Contextual information may refer to the properties of the device (e.g., memory available and physical dimensions) or its environment (e.g., location, time, and settings) in mobile systems, relevant business-level information, etc.

Finally, the dynamism may refer not only to the execution of a single application instance (e.g., a particular business process instance), but to the *evolution* of the application as a whole, when certain relevant "trends" affect the model of the whole process.

Another important factor for the classification of the dynamic aspects of the execution refers to the placement of the changes and events in the system specification. The addressed problem may refer to variations at different functional SBA layer, namely *business process management layer*, *service composition layer*, or *service infrastructure layer*. In the first case, the adaptation deals with business events and parameters, and may require the modification in the business process model. In case of composition, the events and changes may reflect, e.g., the variation of the behavioural specifications, thus requiring re-composition or creation of adapters. In case of infrastructure, the changes may correspond to the modification of QoS parameters, entailing the re-configuration and re-binding of the involved services. The changes may be also classified as *internal* (e.g., failure of the invocation of involved service) or *external* (e.g., contextual changes), *local* (referred only to the application itself) or *global* (when the changes have effects on the other related actors).

The characteristics of services and service-based systems on the one hand facilitate the design and development of adaptation mechanisms, but on the other hand pose new challenges and adaptation problems. Indeed, services are intrinsically *heterogeneous* and *autonomous*; they are not under the control of the service integrator. This requires addressing specific types of changes and extensive use of run-time service monitoring techniques. On the other side, services and service definition standards are designed in a way to ensure *high level of abstraction* and *coarse granularity*, *implementation neutrality*, *loose coupling*, and *reuse*. Such characteristics dramatically simplify *discovery* and *composition* of services, thus providing a basis for adaptation and re-configuration in service-based applications.

### 2.5.1 Adaptation Specification

From the design perspective, the most important aspect of the development of adaptable service-based applications is the way the adaptation mechanisms are specified and configured. In a general form, the adaptation specification considers two relevant aspects: when the application adaptation should take place, and how the adaptation should be performed. The first aspect characterizes the variable part of the application and/or its context, and amounts to the description of relevant events, situations, modifiable system parameters and their possible values. The second aspect is used to drive the adaptation management, and may define the goals of the adaptation, the strategies to achieve the goals, the actions to be performed, and instructions for making decisions and selections.

#### *2.5.1.1 Specifying application variability*

The representation of the moment, when the adaptation should take place, may be given either implicitly or explicitly. In case of *implicit* specification, the relevant information is not defined by the designer, but is in certain sense “predefined” in the platform and can not be changed, without modification of the adaptation mechanism. This is the case, for instance, for a wide range of “abstraction” approaches, where at design-time the application (service composition) does not define the concrete services to be involved, but refer only to the abstract interfaces or functional requirements. At deployment- or run-time the platform accesses the service registry and then selects and composes the candidate services. Here the point of adaptation (deployment time) is predefined. Implicit modelling is typical for many optimization approaches, where the reconfiguration actions are triggered upon detecting certain changes in the resource allocation.

In case of *explicit* specification, the adaptation point is chosen and prescribed by the designer. There are many approaches towards definition of such information. First, the point may be associate to a certain control point in the behavioral model of the application: functional block (e.g., scope in BPEL or structured activity in business process model) or execution of a basic activity. Second, it can be associated to a certain event, such as reception of a message or a timeout, start of the compensation or exception processing, etc. Third, such a moment may describe certain state of the application and its environment. In this case the specification is given as a complex condition over a set of relevant parameters, both external and internal. Finally, the point may be associated to more complex events, such as violation of a complex behavioral requirement. In these latter cases the property describing such a situation may be represented in some specific language, such as temporal logics, first-order logics, event or situation calculi, etc.

#### *2.5.1.2 Specifying adaptation activities*

As in the case of the application variability, the instructions regarding the adaptation may be defined implicitly or explicitly. In case of *implicit* definition, the application designer has no influence on the adaptation process; the adaptation strategies and actions are decided and executed by the system according to some predefined schemata. In abstraction-based approaches, for instance, the decision on which services to select and how to compose them is often driven by some predefined utility function, and the approach is trying to achieve the best possible value. Also in the correction approaches the recovery strategies are often predefined, and are out of control of the designer.

The explicit specification of the adaptation activities may be given in different ways. We will classify them into action-based, goal-based, and approaches based on explicit variability.

*Goal-based* approaches describe the adaptation activities in a higher-level form, where certain objectives to be reached by the system are defined, leaving the system or the middleware to determine the concrete actions required to achieve those objectives. While the decisions and choice of actions are left to the platform, the designer (or the user) has an ability to guide the adaptation process with a set of requirements. These approaches are close to many works in AI, where in some multi-agent systems, agents are often described as being autonomous, goal-oriented and having social abilities to communicate with other agents. The cooperation between individual agents converges and tends to achieve the global application goal.

Since the set of concrete actions and options is not known at design-time, the applicability of the implicit and goal-based approaches relies on the possibility of the corresponding mechanisms to discover potential options and make decisions at the run-time. This, however, depends on the following factors. First, on the expressiveness and completeness of the service descriptions, which are relevant for the discovery, and often requires very rich multi-dimensional specifications (e.g., business domain description, functional and non-functional properties, semantics, etc.). The corresponding languages and notations constitute the specification support for adaptation on the side of service provider. Second, it depends on the goal (requirements) specifications that drive the selections and decisions process. The notations for expressing these requirements constitute the specification support on the side of service consumer.

*Action-based* approaches are widely used in different areas, such as distributed systems, active databases and expert systems. The specification is defined as a situation-action rule, which exactly specifies what to do in certain situations or upon occurrence of a certain event. The situation part corresponds to the specification of variation, while the second part prescribes concrete adaptation actions to be performed. The action-based approaches differ in the way the instructions and the primitive actions are defined and structured. These actions may include

- Adaptation actions, e.g., re-execute (service invocation), re-bind (service), substitute (service), re-negotiate (SLA), roll-back (to previous stable execution point), re-compose (set of services), halt.
- Management actions, e.g., notify, log.

The approaches based on *explicit variability* target the specification of the adaptation activity as follows. The identified variation point is associated with a set of alternatives (variants) that define different possible implementations of the corresponding application part. In business processes this corresponds, for example, to a nominal sub-process, and a set of potential customized flows (c.f.,[109]). Additionally, the specification may include the guidelines for selection of the variants, e.g., preferences or ranking rules, specification of relations and dependencies between different alternatives, etc.

Following the above classification, we will divide the surveyed approaches into the following groups: *(i)* implicit approaches, where both the adaptation point and the adaptation activities are predefined (or at most parameterized) and the design problem mainly amounts to the service description; *(ii)* action-based approaches; *(iii)* goal-based approaches; *(iv)* approaches based on explicit variability.

### 2.5.2 Implicit Adaptation Approaches

In the implicit adaptation approaches the decisions when the system has to be changed and which actions to perform are predefined by the adaptation framework. This is a typical situation for dynamic service compositions, where the services are selected and composed dynamically upon, e.g., unavailability of some of them. This is also the case for many self-healing systems, where the recovery activities are somehow hard-coded. The role of the design activities in case of implicit approaches is to provide possibly richer and more complete descriptions of the services and compositions in order to support and simplify the decisions made at run-time automatically. In case of dynamic composition, for example, these decisions correspond to the discovery and selection of the candidate services. However, since this kind of approaches mainly concern run-time activities, they will be presented in detail in Section 2.7.5.

### 2.5.3 Goal-based Adaptation Approaches

Goal-based adaptation specification approaches may be seen as an extension of the implicit approaches. Indeed, they do not prescribe the actions that the platform or the application should take in a critical situation, but only (declaratively) define a goal that the platform (or application) should achieve. For instance, in the approaches towards dynamic composition of Semantic Web services the composition goal may be defined as a set of outputs and effects, while the set of concrete services providing these effects and their composition are derived at run-time.

Moreover, very often the main adaptation goal is predefined, and the adaptation specification identifies only some additional requirements and constraints that should guide the adaptation process. These requirements may have form of queries of a certain language, formal contracts (SLA) defining the constraints on functional or non-functional parameters, etc. In particular, in grid computing the optimization problem is driven by the goal of better resource allocation, but the additional specifications defining the required quality levels or the negotiation bounds are provided.

As in the case of implicit approaches, the multi-dimensional description of the application and the involved services is very important. Additionally, in the goal-based approach these descriptions are also shared with the declarative specifications that describe the adaptation goals and requirements.

#### *QoS adaptation in service grids [108]*

| **Synopsis** | |
|---|---|
| **Goal** | Optimization |
| **Importance** | Dynamically changing characteristics of the execution environment (e.g., network traffic or workload) require run-time adaptation of application behavior and QoS properties in order to improve the resource utilization. |
| **Kind of contribution** | Design method and implementation technique |
| **Applicable to** | Grid-based compositions |
| **Phase where it applies** | Design and execution |
| **Key Ideas** | Comprehensive taxonomy of of QoS requirements; ad-hoc optimization algorithm tightly related with the taxonomy |
| **Foundations** | Service-level agreement, QoS, optimization |
| **Validation** | Case study |

In many applications of Grid computing, the resources advertised and traded as services are used (allocated) simultaneously. Dynamically changing characteristics of the execution environment (e.g., network traffic or workload) require run-time adaptation of application behavior and QoS properties in order to optimize the resource utilization. While the main goal of the platform – resource use optimization – in this case is fixed, the adaptation strategies and their realization has to take into account many additional constraints dictated by service level agreements established on the executed applications, as well as by new requests for resources. The specification of these SLAs and requests defines the additional requirements to the adaptation process.

In [108] the authors define an adaptation scheme that enables dynamic optimization of resource allocations based on the QoS parameters in contracts and service requests. In this scheme, the service consumers specify the required QoS properties, which then are negotiated until the service level agreement is established. At run-time the platform may vary the parameters of provided services in order to optimize the resource utilization and to increase the number of service requests. This variation, however, takes into account the requirements specified in the negotiated SLAs.

The authors define the following forms of the QoS requirements and SLA specification: "guaranteed" service, for which the provider commits to deliver exact values of the QoS parameters; the "controlled load" service, for which ranges of the corresponding parameters should be respected; and "best effort" services, for which the consumer does not put any constraints on the QoS properties. Since the resources are associated with costs, the goal of the optimization is to maximize the total cost of the resource utilization, while sill providing quality of service acceptable by the requestors. The proposed adaptation scheme aims at adjusting QoS parameters dynamically, e.g., by re-negotiating SLA values for "controlled load" services, allocating more resources to "best effort" services, when the resources are not used by other classes of services, etc.

While the expressiveness of the SLA specification in this approach is very simple and limited to the definition of service class and the parameters, the proposed adaptation method is different from the implicit adaptation, since the specified information directly affects and guides the corresponding optimization process.

### *Adaptation in Web service composition and execution [110]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction, optimization |
| **Importance** | Supports run-time correction through re-binding or re-composition of services in highly dynamic environments |
| **Kind of contribution** | Method |
| **Applicable to** | Composed Semantic Web services |
| **Phase where it applies** | deployment time, execution |
| **Key Ideas** | Two-stage approach: first, template is generated using Semantic descriptions and functional requirements; second, the template is grounded using concrete services with the goal of optimizing non-functional requirements |
| **Foundations** | Semantic Web servies |
| **Validation** | Experiments |

In [110] the authors address the problem of adaptive service composition states as follows: "given the specifications of a new service, create and execute a workflow that satisfies the functional and non-functional requirements of the service, while being able to continually adapt to dynamic changes in the environment". This problem requires not only to compose and to bind the services in a composition that satisfies the given requirements, but also to continuously monitor the execution and the environment and to dynamically modify the composition when the critical changes occur.

The authors propose the two-staged approach, where first the abstract service composition (template) is defined based on the functional user requirements, and then the abstract composition is instantiated with the dynamic services based on the optimization of quality-of -service metrics. While the second stage of the approach is implicit and similar to the one presented, e.g., in [111], the first one relies on the composition goal specifications from the Semantics Web services domain [112]. Furthermore, at run-time the two approaches may be interleaved. This happens when the platform cannot find a suitable instantiation of currently selected abstract process templates, and the templates should be regenerated from the same user requirements.

The results of the experimental evaluation carried out by the authors demonstrate the appropriate level of efficiency and robustness of the presented approach when handling quite large number of failures and re-compositions.

## *METEOR-S [113][114][115]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction, optimization |
| **Importance** | Supports run-time correction through re-binding services in highly dynamic environments |
| **Kind of contribution** | Method |
| **Applicable to** | Semantic Web services |
| **Phase where it applies** | Design-time, deployment time, execution |
| **Key Ideas** | Separate notations for functional and non-functional requirements that are dynamically satisfied by specific reasoners and coordinated by a special algorithm |
| **Foundations** | Semantic Web services, Markov Decision process |
| **Validation** | Case study |

In [113] [114] [115] the authors address a problem similar to the one presented above, that is, dynamically configuring and executing abstract workflows with a set of available services. The way the processes and requirements are specified is, however, different and relies on a set of specifications defining functional and non-functional constraints on the processes and involved services.

In the presented approach the abstract workflow models are defined by the designer and are represented in BPEL. Additionally to the workflow specification, the designer provides a set of specific requirements that constrain the functional and non-functional properties of the target process instance and the involved component services. In order to specify this information, the process model is equipped with the semantic specifications in OWL (OWL-S for service descriptions) that describe the domain specific knowledge, the functional and non-functional characteristics of the relevant aspects of the process model. The non-functional constraints refer to QoS properties, security or transactional aspects, and are transformed into integer linear programming constraints. Functional constraints define the statements over the data compatibility or control flow, and are represented in [113] using Semantic Web Rule Language (SWRL).

At deployment-time, as well as at run-time when the service failures are detected and the reconfiguration is required, the proposed platform performs service discovery based on the requested service templates and their semantic descriptions, performs quantitative analysis over non-functional properties using linear programming solver, and qualitative analysis over functional properties using a dedicated SWRL reasoner. The run-time execution platform supports also data mediation, and run-time reconfiguration in case of service execution failure.

The implementation adopted in [115] relies on a specific algorithm based on Markov Decision Process, which enables coordinated management of adaptation activities in case of distributed service and process constraints.

## *Service discovery framework [116]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction |
| **Importance** | Permits dynamic identification and replacement of the services that failed their contracts (functional and non-functional) |
| **Kind of contribution** | Method |
| **Applicable to** | Service-based application |
| **Phase where it applies** | Execution |
| **Key Ideas** | Monitor the behavioural and QoS requirements, and use the violation information in order to automatically extract discovery queries for replacing services |

| | |
|---|---|
| **Foundations** | Web service concepts, service discovery |
| **Validation** | Discussion |

In many approaches the adaptation problem refers to dynamic substitution of component services that fail certain requirements and contracts. When this happens, the system should take a decision whether the service should be replaced, and then discover and bind a new service that is supposed to satisfy the requirements. Service discovery is particularly relevant in this context, since the services are selected in such a way that the adaptation requirements are satisfied in the best possible way. In many cases the selection is driven not only by the categorization of the replaced service and/or by the necessity to optimize the quality-of-service characteristics of a system, but additional requirements that the replaced service has failed to satisfy. The adaptation goal is, therefore, to bind to a new service that is compliant with these additional requirements.

In [116] this problem amounts to dynamic substitution of services that fail certain behavioral requirements and constraints. In this approach, the composite application is monitored at run-time, and if the violation of requirements is identified, the platform automatically extracts the additional constraints to the replacing services, and performs service discovery and selection based on those constraints. The requirements to be monitored are related to the behavior of the system or QoS parameters over a service or a composition, and are defined in event calculus in terms of event and fluents. The former correspond to operations performed by the application logic (message reception or emission, assignments, etc), while the latter characterize the state of the application (conditions that hold in some interval). At run-time these requirements are checked by the monitor against the actual executions of the system.

When the violation is detected and the platform decides to replace the failed service, new services are discovered and the candidate is selected. The selection is based on the additional adaptation requirements extracted from the diagnostic information provided by the monitor. This information comprises the structural part regarding the categorization and functionality of the failed service and the behavioral part that defines the set of paths that the execution of the target service should respect. The behavioral part is obtained from the violated requirement and the violation synopsis generated by the monitor using predefined transformation rules. The discovery tool checks the behavioral specification of the candidate services expressed as state machines against the behavioral part of the query and selects the corresponding candidate.

An important feature of the approach is that the adaptation activities are based on the requirements specifications defined by the service integrator at design-time. In this way, the monitoring activity and the dynamic adaptation share the same model of the application.

### *Interleaved planning and execution [117]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization |
| **Importance** | The standard domain-specific business process should be customized in order to satisfy the requirements of a particular customer and the constraints of particular providers |
| **Kind of contribution** | Method |
| **Applicable to** | Service-based business processes |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Special languages for defining customization requirements and specific domain and provider assertions are defined. The requirements are used to automatically adapt the standard business process and re-configure it dynamically |
| **Foundations** | Business process modelling, automated planning |
| **Validation** | Case study |

In [117] the authors address the problem of customizing the standard business process in order to satisfy particular requirements of a given process user. In many domains the business processes accomplishing certain goals are predefined, while the actual components and services participating to the process may vary. The services have different functional and non-functional constraints and assertions that the composed process should satisfy. Furthermore, different user may put additional requirements on the process model and the component services, such as total cost, optional use of certain operations, etc. The execution platform, therefore, should be able to dynamically adapt the standard processes to all these additional requirements and goals by selecting the appropriate component services.

In order to address the problem, the authors present the specification and execution framework, which enables to define the user-specific requests and goals, as well as the service- and domain-specific assertions constraints, and to automatically obtain and execute the customized version of the standard process, which satisfies these requirements. Moreover, the approach interleaves the execution with monitoring of those requirements and assertions, and in case of violations, re-composes the process taking into account the initial user request, the current process state, and the execution environment.

While the approach is conceptually similar to the one on quality-driven service composition, the adaptation goals in this approach deal with the functional properties of underlying process. The authors present two languages, namely XML Service Request Language (XSRL) and XML Service Assertion Language (XSAL), to express the user requests and service assertions respectively. XSRL allows one to express complex functional goals, such as the necessity to achieve certain state (e.g., the flight and hotel are booked in the trip reservation process), to maintain certain condition (total amount should be less than 500$), user preferences (prefer flights to trains), etc. XSAL defines the domain constraints and service-specific assertions in a similar way: simple assertions define reachability requirements, preservations assertions define the invariants that the application should maintain, and the entity assertions define the constraints on the evolution of a business object.

The corresponding algorithms are defined that take the process model, the domain assertions, and the user request as input, perform the selection, composition and execution of different component services continuously monitoring different constraints.

## *Petri-net based reconfiguration [153]*

| **Synopsis** | |
|---|---|
| **Goal** | Enabling run-time reconfiguration of service-based systems |
| **Importance** | Allows for the adaptation of a system configuration to the changes in the context or the requirement of the users |
| **Kind of contribution** | Method |
| **Applicable to** | Web-Services context |
| **Phase where it applies** | Run-time reconfiguration |
| **Key Ideas** | The system is described through a Petri-net showing dependencies among services. The model can change at runtime according to changes in the availability of services or in the requirements of the users. An algorithm to maximize QoS defined as a function of time is also provided. |
| **Foundations** | Service-based architectures concepts, Petri-net models |
| **Validation** | Simulation results are provided to prove correctness of the algorithm |

Another example of run-time management of a SOA configuration can be found in [153].
The approach presented in [153] is based on representing a service configuration through a *model*, and then modifying this model as needed to adapt the configuration to changes in the environment and in the requirements of the users. This approach addresses the web service context, particularly the case of highly dynamical environments.

The system is described through a Petri Net, which represents the dependencies among the services in the configuration. In other words, the Petri Net represents the *places* where the services are mapped, and the arcs in the graph model the relationships among the places.

This model can dynamically evolve, according to changes in the environment. In [153] the authors describe an algorithm which modifies the configuration (and its model) with the aim to provide the highest QoS. Many different parameters are used to evaluate the QoS, and many metrics are considered to measure each parameter. The QoS provided by a service is defined as a function of place and time, and the described algorithm looks for the best configuration.

### *QoS-based reconfiguration [154]*

| **Synopsis** | |
|---|---|
| **Goal** | Enabling run-time reconfiguration on the basis of the provided QoS. |
| **Importance** | It focuses on the real time analysis of QoS provided by a system |
| **Kind of contribution** | Method and tool |
| **Applicable to** | Service-based architectures |
| **Phase where it applies** | Run-time reconfiguration |
| **Key Ideas** | Choosing the configuration with the highest QoS using genetic algorithms: genes are represented by variables concerning service selection and resource allocation. The QoS of each configuration is evaluated, and then a new configuration is generated through mutation. |
| **Foundations** | Service-based architectures concepts, genetic algorithms, Markov Chain Model |
| **Validation** | Some tests have been performed of the tools, considering an Intrusion Detection System as a case study |

This approach particularly focuses on developing methods to provide the highest QoS. Because of this focus, it has been classified among the goal-based approaches. However, it includes something more than usual goal-based approaches: it also defines an algorithm to achieve the goal, which is the highest QoS.

In [154] the authors use an extension of WSDL to express properties about the QoS behaviour of a system. The focus is on obtaining an adaptation of the system configuration through and adaptation of the observed QoS behaviour. The information gathered about the QoS behaviour provided is used to compare the different candidate configurations, using genetic algorithm to find the best one.

## 2.5.4 Action-based Approaches

In contrast to the goal-based approaches, the action-based specifications aim at representing concrete actions to be performed upon certain event, such as a service failure or context change. Due to this nature, the action-based approaches are usually based on procedural notations and languages. This may refer both to special-purpose high-level languages, where the instructions correspond to specific repair or management actions, and to low-level programming languages that define the adaptation steps at the level of the application implementation.

Since the adaptation strategies and their implementations are pre-defined, the action-based approaches do not require additional reasoning or decision-making at run-time, and, therefore, are considerably more efficient. On the other hand, such approaches are less flexible, since the adaptation specification defines the activities corresponding only to a restricted set of scenarios, and can not address all the relevant events and contexts. As a consequence, the action-based approaches are typically used in the self-healing systems, where the critical events correspond to an a priori fixed set of application faults, and the adaptation operations represent system recovery actions.

## *Policy-driven middleware for self-adaptation [118]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction, customization |
| **Importance** | Provides a way to correct and customize the application instance without changing application model |
| **Kind of contribution** | Method |
| **Applicable to** | Service-based Applications |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Policies (ECA rules) for handling specific situations and faults. The action part of the policy represent the adaptation instructions, which are enforced at run-time |
| **Foundations** | Web service concepts, Service-level agreements, QoS |
| **Validation** | Case study |

In [118] the problem of adaptation of Web service compositions in order to accommodate to various business exceptions and faults. In highly dynamic systems the execution environment of the application may change in a way not foreseen at design-time. Very often the system should be customized at run-time and refer to a particular application instance, without changing the whole application model. This problem requires specific approaches to the application design, allowing one to define the actions necessary to correspondingly alter the execution of an application instance.

In order to address the problem the authors propose a policy-based middleware, MASC, where the policies define the customization actions that should be performed in order to react to certain exceptional situation or a fault. The policies are defined in the form of ECA-rules (Event-Condition-Action) that describe the triggering event (application fault, interaction operations, start/end of an application instance), the condition, under which the rule applies (restriction on the application state or history), and the actions to be applied. The policies are represented in a specific language that extends WS-Policy standard in order to deal with monitoring and adaptation activities. The language allows for defining not only adaptation but also monitoring rules and directives. The monitoring rules define the (relevant) information to be observed and map the undesirable situation or condition to a meaningful failure event that will be processed by the adaptation rules.

The approach allows for the specification of two kinds of adaptation action: process-level actions and message-level actions. The former are used to alter business logic of a particular instance, while the latter are mainly used to deal with various low-level faults, such as invocation failure, SLA violation, etc. The possible process-level actions are add, remove, replace, change order of activities, suspend, delay, resume business process. Message-level actions include invocation retries, service substitution, concurrent invocation of several similar services, etc. The adaptation rules may be also assigned priorities to define the order of executions when several rules apply to an event.

The adaptation process is supported by the specific platform that allows for monitoring relevant policy information and events and performs the necessary actions both at message-level (intercepting the messages) and process-level (altering the process engine). The approach and the platform were evaluated on two comprehensive case studies that represent different adaptation problems, namely customization of a process to specific business conditions, and a problem of system recovery in case of message faults and QoS contract violations.

## *Self-healing service compositions [119][120]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction |
| **Importance** | Provides a way to recover from various faults in dynamic environment |
| **Kind of** | Method |

| **contribution** | |
|---|---|
| **Applicable to** | BPEL processes |
| **Phase where it applies** | Design and execution |
| **Key Ideas** | Run-time monitoring of pre- and post-conditions over process activities. Defensive process design to define recovery steps. |
| **Foundations** | Service composition concepts, dynamic binding |
| **Validation** | Discussion |

In [119] [120] the authors address similar problem, that is, how to recover the application execution in case of unpredictable service failures in highly dynamic execution environments. The problem relates to the execution of service compositions, where the component services are identified and bound at deployment-time or at run-time.

In order to cope with this problem, in [119] the authors propose the approach called Defensive Process Design in conjunction with run-time monitoring of process execution. Run-time monitoring allows for timely detection of problematic situations, while Defensive Process Design provides the facilities to define the necessary recovery steps. The monitoring instructions are defined as functional and non-functional assertions (pre- and post-conditions) on the composition activities (defined in BPEL). In order to react to the detected violations, the authors propose three kinds of recovery strategies, namely retry, rebind, and restructure. The corresponding instructions for the retry and rebind actions are introduced directly in the composition specification, while the restructure instructions, defined as rewriting rules on the process flow graphs, are defined separately.

In [120] the proposed approach is further refined and extended. In particular, the monitoring specification is defined using specific notation, Web Service Constraint Language (WSCoL), which provides expressive and extendable facilities for monitoring composition assertions over functional and non-functional properties of the system. In order to define recovery specifications, Web Service Recovery Language (WSReL) is proposed. The main ingredients of the language are the atomic recovery actions and the recovery specifications that declaratively join atomic actions in two complex procedures (steps) and alternatives. The set of actions include actions at service level (retry, rebind, ignore), at process level (substitute, halt, call), and management actions (notify, log). The supporting platform relies on a specific rule engine that manages recovery specifications and activates recovery, and on aspect-oriented techniques to introduce the recovery implementation at the level of the process engine.

### *SH-BPEL [107]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction |
| **Importance** | Provides failure recovery capabilities for BPEL processes in dynamic settings |
| **Kind of contribution** | Method |
| **Applicable to** | BPEL processes |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Extend standard BPEL engine with the self-healing API and facilities to describe, coordinate and implement the fault management activities |
| **Foundations** | Service composition concepts, fault handling |
| **Validation** | Discussion |

In [107] the authors also address the problem of providing self-healing capabilities to the service based systems, tailored, however, to a particular kind of such systems, namely service compositions in BPEL. In this case the authors aim at extending the standard failure recovery and management capabilities of BPEL with additional functionalities that are crucial in open and dynamic settings.

Apart from service-level recovery strategies, such as retry or rebind, the authors identify and describe a wide range of process-specific activities, such as modifying the values of process variables, redoing a process task or an entire part of a process (scope), specifying and executing alternative paths in the process, going back to a "safe point" in the process execution, etc.

The proposed solution defines the extension of the standard BPEL execution environment, namely SH-BPEL, which integrates and supports the necessary recovery facilities. In this solution, the original process specification is pre-processed and extended with the additional instructions and control points, which allow for performing the above actions. These additional recovery actions constitute the management API and can be invoked through a special process management interface that is made available at deployment-time. The underlying architecture provides the necessary tools for detecting critical events and engaging recovery actions invoked through this management interface.

The existence of a well-defined management and recovery API and interface enables various ways to control and define the recovery strategies. First, these strategies and decisions may be pre-defined in the underlying process manager and customized through a corresponding programming interface. Second, the architecture allows for a collaborative environment, where a set of engines running different composed services participating to choreography perform coordinated recovery activities through the corresponding management interfaces. Finally, the platform allows for recovery both at the instance level, when only current process instance is adapted, and class level, when the whole process model is changed.

## *Aspect-oriented adaptation framework [121]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization |
| **Importance** | Resolution of mismatches between functional specifications of the interacting services |
| **Kind of contribution** | Method |
| **Applicable to** | BPEL processes |
| **Phase where it applies** | Design |
| **Key Ideas** | Aspect-oriented methodology for defining adaptation templates is proposed. The template is instantiated with the actual parameters specified by the designer and the adaptation is performed automatically |
| **Foundations** | Aspect-oriented techniques, functional interoperability |
| **Validation** | Case study |

In [121] the authors address an adaptation problem that deals with the resolution of mismatches between functional specifications of the interacting services. The authors present taxonomy of mismatch types that include mismatches between the operation signatures and constraints on the input parameters and protocol mismatches, such as missing or extra message emissions, message split or merge, message ordering, etc.

The presented approach proposes a set of adaptation templates that define the implementation of the composition restructuring. A template consists of two parts – query and advice. Following the aspect-oriented methodology, query is used to identify the location in the specification model, where the adaptation should apply (joinpoint). More precisely, query specification identifies the simple or complex activity, to which the adaptation applies, whether it is applied before, after or around the activity, and a specific condition that defines additional restrictions on the corresponding parameters. Advice characterizes the generic program to be executed when the query conditions are met. As a notation for specifying the advice the BPEL language is used. Such a program defines the low-level transformations, such as receiving a message and storing it in internal procedure in case of ordering mismatch.

During the design phase, the developer is required to identify mismatches, and to provide the relevant information for the queries and advices, that is, instantiate the template. Based on the concrete parameters, the design environment extracts the adaptation specification, which is executed at run-time.

## *Policy-based adaptive services for mobile commerce [122]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization |
| **Importance** | Allows for dynamic customization of the mobile applications when the relevant contextual information is changed. |
| **Kind of contribution** | Method |
| **Applicable to** | Mobile service-based application |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Policy-based framework for modelling, structuring, and enforcing context and context-specific management rules is proposed. At run-time the policies are evaluated and enforced by the platform |
| **Foundations** | Mobile applications, semantic web, policy modelling |
| **Validation** | Case study |

In [122] the authors address a different adaptation problem related to the changes in the context of mobile service-based applications. In mobile applications pervasiveness and ubiquitous availability are essential features; they exploit a wide range of diverse mobile services provided and advertised locally, making the user and application contexts a central concept in the development and provision of mobile service-based systems. The problem of representing and understanding contextual information, as well as the problem of adapting the application to the changes in context remains open.

The proposed approach defines a simple policy-based framework and architecture for designing and providing mobile service-based and context-aware applications. The proposed approach relies on a set of existing methods and techniques from various disciplines that integrated in an intelligent way provide the necessary facilities.

In the presented approach, the context is represented using Resource Description Framework, and then reflected in definition of policies using the concepts of template (generic contextual aspect) and facts (concrete contextual situation). The context in these settings refers to the properties of the device, possible services and their characteristics, user preferences and settings, etc. The policies define the adaptation actions that should be fired when the system occurs in a certain context.

Based on these concepts, the work also defines an iterative methodology for defining contextual information and adaptation policies. Starting from initially collected contextual knowledge, the designer specifies the adaptation policies and extracts intermediate contextual information. The process is iteratively repeated until no new rules can be identified or new contextual specifications can be obtained. The context models and policies are separated into modules and the module pipelines are identified. The process is supported with a visual notation, which shows the initial and intermediate contexts, the final result and the policy processing flow between context modules.

The proposed methodology is supported with the agent-based run-time platform that incorporates the monitoring facilities for observing various contextual properties, the communication middleware for interacting with the services, the policy decision maker that filters the policies to be applied, and the policy enforcement point, where the adaptation actions are defined.

## *Model transformation [128]*

| **Synopsis** |
|---|

| | |
|---|---|
| **Goal** | Enabling adaptive reconfiguration of systems |
| **Importance** | Allows for automatic, adaptive reconfiguration of systems |
| **Kind of contribution** | Method |
| **Applicable to** | Embedded Service-based Systems |
| **Phase where it applies** | Design-time and run-time configuration |
| **Key Ideas** | Buildes a model of the system configuration at design-time, including policies to specify its behavior. Uses rules made up of graphs to manage reconfiguration: when the graph on the Left Hand Side of a rule is met, it is substituted with the graph of its Right Hand Side |
| **Foundations** | Service-based architectures concepts, graph-transformation techniques |
| **Validation** | Not at the time of paper writing (2005) |

Model-based approaches, which are often used at design-time, can also be exploited to obtain automatic run-time adaptation of a system configuration.

An example of this approach is presented in [128]. It is based on building a model of the system at design time, together with an initial configuration of its services. The model is object-oriented, and it includes high level policies that specify the desired behaviour of the system. We are not going deeper in the details about how this model is created. Indeed, here we are interested in how it is transformed. The approach presented in [128] to reconfigure the system at run time is based on *rules*, that are composed by a left-hand-side and a right-and-side, which both are graphs of the same kind of the one representing the model. When a part of the system model meeting the left-hand-side of a rule is met, then it is substituted with the corresponding right-hand-side. These simple rules can be easily modify by users as needed, with the same tool allowing for the building of a model of the system. Moreover, the policies described at design-time are deployed on a structure made of services which enables the run-time adaptation of the system.
When the paper was written (2005), its authors were developing a prototype implementing the described meta-model, so there were no actual proof of its validity. Nonetheless, a graphical tool allowing for the building of the model of a system exists. Such a tool also implements the algorithm used for the model transformation.

### *SCENE: integrating WS-binder to achieve dynamic binding and adaptation [130]*

| **Synopsis** | |
|---|---|
| **Goal** | Providing a composition language and a runtime execution environment for service compositions |
| **Importance** | Allows for dynamic binding re-binding, and negotiation |
| **Kind of contribution** | Technique |
| **Applicable to** | Web services |
| **Phase where it applies** | Design-time definition of policies and runtime binding, re-binding, and negotiation |
| **Key Ideas** | To offer a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding, re-binding, negotiation, and self-reconfiguration operations. |
| **Foundations** | Web service technologies concepts |
| **Validation** | Case studies concerning automotive and telecom domains |

SCENE [130] offers a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding and re-binding self-reconfiguration operations. A SCENE composition is enacted by a runtime platform composed by a *BPEL engine* executing the composition

logic, an open source rule engine, *Drools*, responsible for running the rules associated to the composition, *WS-Binder* (see Section 2.7.3.4) that is in charge of executing dynamic binding and re-binding, and by a Negotiation component that can be used to automatically negotiate SLAs with component services when needed [134].
In a later work (see [131]), the SCENE framework has been extended through the integration of a module enabling the resolution of mismatches between the interfaces and protocols of invoked services. In the paper a set of possible mismatches is defined, together with a list of available adaptation strategies, that can be combined in scripts through a language. The adaptation script specifies the differences between the primary concrete service selected for binding, which is defined at design time, and the other available concrete services that can be candidate for dynamic binding. Some case studies have been developed to test this approach. These tests have shown that some overhead is introduced, with respect to the execution of plain BPEL process.

#### *WS-Diamond [132][133]*

| **Synopsis** | |
|---|---|
| **Goal** | Design and execution of self-healing services |
| **Importance** | Self-healing is based on diagnosis of faults and repair planning, repair execution is based on additional functionalities on top of BPEL engine. At design time self-healability properties are proven. |
| **Kind of contribution** | Models + Tools + execution environment |
| **Applicable To** | Web services |
| **Phase where it applies** | Runtime + design-time: Self-healability (diagnosticability+repairability) evaluation. |
| **Key-ideas** | WS-Diamond allows the identification of causes of errors (faults) in service compositions. Plans are generated to perform repair actions on the process, so that unforeseen failures can also be repaired. Repair actions include substitution, compensation, retry. A management interface is provided to support repair. |
| **Foundation** | Process modelling, diagnosis, planning |
| **Validation** | Action Research + Case Study |

The WS-Diamond EU project [132][133] (http://wsdiamond.di.unito.it) developed an execution environment and design tools to design self-healing service compositions. Self-healing is based on the diagnosis of faults from symptoms during execution (failures or unexpected messages or quality levels) and applying a set of repair actions including compensation of operations, retry, substitution of services. Fault identification is based on diagnosis techniques and the focus has been mainly of data faults and QoS faults. Repair plan generation is based on planning techniques. Both diagnosis and plan generation are based on model-based techniques. A management interface to support the execution of repair actions, considering also interaction with stateful service, has been designed and realized (see SH-BPEL).
Methods and tools to assess the self-healability of processes have been proposed, including repairability evaluation, diagnosability, temporal conformance checkers.
A methodology for managing information quality in self-healing web services has been developed, focusing on the choice of the more appropriate repair actions based on multiple actors and requirements for repair of service compositions.

### 2.5.5 Explicit Variability

Explicit variability approaches allow one to define relevant application changes and the way the system should react to it, but to precisely identify a particular moment in the execution, where the change happens, and to represent all the relevant variants of behaviour, applicable in such cases. In order to provide explicit definition of the execution location (variation point), where the variation takes place, the behavioural specification of the application is used. The variation point is equipped with a set of alternative behaviours (variants) that may be applied under certain conditions and in particular cases.

The explicit variability approaches are widely used in business process modelling and software product line engineering [123], where variability models provide the basis for application customization, flexibility and re-use. In business processes variation often refers to the single tasks or sub-processes, while in SPL this may also correspond to components, their interfaces and implementations. Analogous classification for the service-based application is proposed in [124], where variability types may be defined at different SOA layers, namely business process layer, unit service layer (or service composition layer), service interface layer, and service component layer.

Due to its nature, the role of monitoring activity within the adaptation approaches based on explicit variability is different from goal-based and action-based approaches. Since the variation point is already defined explicitly, there is no need to monitor specific conditions or assertions. Instead, the monitoring activity is used to obtain some relevant context information that may be required to choose one alternative or another. In this setting, the specification of context and contextual conditions is still important.

As well as in the case of action-based approach, the explicit variability approaches are rather efficient, since the only overhead refers to the evaluation of the selection criteria for a particular alternative, and the latter is already defined.

### *Identifying possible types of changes [161]*

| **Synopsis** | |
|---|---|
| **Goal** | Dealing with service evolution |
| **Importance** | Classifying possible changes and proposing approaches for dealing with them |
| **Kind of contribution** | Method |
| **Applicable to** | Services in general |
| **Phase where it applies** | Run-time evolution management |
| **Key Ideas** | It classifies service changes on the basis of functional criteria and of their effects, proposes some approaches for dealing with the different kinds of changes, sketches the change-oriented service lifecycle, and discusses issues of service consistency on the basis of an abstract service definition model |
| **Foundations** | Service-based architecture concepts |
| **Validation** | -- |

[161] presents a theoretical approach for dealing with the service evolution. It categorizes the types of changes that occur in services based on functional criteria in:

- *structural*, that focus on service types, messages, interfaces, and operations
- *business protocol*, that affect the structure and ordering of the messages that a service and its clients exchange to achieve a business goal
- *policy induced*, requiring changes to policy assertions and constraints
- *operational behaviour*, that concentrate on the effects and side effects of changing service operations.

Furthermore, depending on the nature of their effects, service changes can be distinguished in:

- *shallow changes*, with localized effects to either service or at most its clients
- *deep changes*, i.e., cascading changes that extend beyond the clients of a service and possibly to the entire supply chain.

Structural and business protocol changes are usually shallow, whereas policy induced and operational behaviour changes are typically deep changes and require a change-oriented service life cycle to

accommodate them. This life cycle must provide the foundation for the management of business process end-to-end change, and take into consideration both functional and non-functional aspects. The paper introduces some key approaches for each type of change that can be used as the basis for future research in service evolution, and sketches the characteristics of the change-oriented life cycle required for deep changes.
One of these approaches is further elaborated in [162], where the issue of structural changes and evolution is discussed. For this purpose, they develop an abstract service definition model that comprises of generic concepts and inter-relationships between them, and abstracts away from the terminology and syntactical nuance of the wide-spread service definition standards. Based on that, they discuss the management of multiple active versions of a service, and the issues of service consistency (well-formedness of service specification) and service conformance (persistence of the service execution result).

## *DySOA [125]*

| **Synopsis** | |
|---|---|
| **Goal** | Correction, customization |
| **Importance** | Allows for managing and guaranteeing the QoS parameters of the application in dynamic environment |
| **Kind of contribution** | Method |
| **Applicable to** | Service compositions |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Explicit representation of the different variants of the critical quality metrics, and the corresponding implementation of the application functionality. The relations and dependencies between the variants may be defined |
| **Foundations** | Service-level agreements, QoS, variability modelling |
| **Validation** | Case study |

In [125] the problem of reconfiguring the application in order to react to the critical changes in QoS metrics is addressed. Due to high dynamism of the application environment, irregularities of the network, and multiplicity of user with their own service-level agreements, these metrics may degrade, and the application should self-adapt in order to be able to provide the expected quality. As in many similar approaches, the adaptation actions correspond to selection and binding to a new service or to changes in the composition specification structure.

The approach adopted in this work, however, relies on a completely different design method. In their work, the relevant adaptation concepts become first-class entities, and the variation of the application is modeled and represented explicitly. The approach is based on explicit modeling of different variants corresponding to the variation points, and on defining various constraints that drive the selection of one alternative or another. The variation model consists of the following elements. *Variation point* defines a particular element of the specification where the selection may apply. *Variant* defines the behavioral or functional alternative to be applied in the variation point (e.g., process fragment or a concrete component service). The actual code/specification of the variant is defined in its *realization* definition. The variation model may contain *intrinsic variation constraints* that restrict the selection of a particular variant, or *extrinsic variation constraints* define mutual dependencies between various choices potentially at different variation points. The latter capability is particularly important, since in many cases there exist control and data dependencies between components, which restrict possible variations of the whole application.

The proposed DySOA architecture provides run-time support for the application adaptation. The QoS metrics of the application are continuously monitored and evaluated. When the certain violations are detected, the reconfiguration unit takes care of analyzing and selecting possible variants in the corresponding points, such that all the variation constraints are met and, moreover, the QoS optimality is ensured.

### *PROVOP [126]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization |
| **Importance** | Allows for managing the specific process contexts and situation through dynamic application re-configuration |
| **Kind of contribution** | Method |
| **Applicable to** | SOA-based business processes |
| **Phase where it applies** | Design, execution |
| **Key Ideas** | Explicit modelling of the application variants through modelling deviation from the nominal case. The definition of variant dependencies is also possible |
| **Foundations** | Business process modelling, process variability |
| **Validation** | Case study |

In [126] the authors focus on the design and management of process variants. The variants are necessary in order to better reflect the specific issues of the process context. The management of variants becomes a complex and error-prone procedure, when the complexity of process models and the number of variants grows.

In [126] the PROVOP (PROcess Variants by OPtions) approach for managing large collections of process variants. The basic idea is to keep the variants in the one model. For this purpose, the basic (or the most common case) process is defined, and its variants are represented by the set of change operations that allow the migration of the basic case model into a specific variant model. The transformation operations are defined as action templates, where actions are *insert*, *delete* or move *process* fragment, and *modify* process attributes. Additionally to the option definition, the PROVOP approach allows for specifying constraints on their usage. The constraints include *dependency*, *mutual exclusion*, *execution order constraints*, and *hierarchy*. In order to associate the process variants to the process context, the latter is defined explicitly using special context variables and the rules that define their relations and evolution.

The supporting run-time environment provides the following functionalities. First, the relevant variants are selected and filtered according to the contextual information. Second, the option constraints are evaluated in order to further restrict the possible options. Finally, the selected options are applied and executed in a process engine. Due to the fact that the context variables may change dynamically, the platform also aims at providing support for run-time migration from one option to another.

### *Dynamic workflow adaptation [127][109]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization |
| **Importance** | Allows for addressing process variability in dynamic process context |
| **Kind of contribution** | Method |
| **Applicable to** | Business processes |
| **Phase where it applies** | Design-time, execution |
| **Key Ideas** | The language for defining process adaptation activities as the set of instructions for defining deviation from the nominal process model |
| **Foundations** | Business process modelling, dynamic transformation |
| **Validation** | Case study |

Relatively different problem is addressed in [127][109]. While the works address the variability of the business process models, the main focus of the approach is on the problems related to dynamic transformation of one process to another. On the one side, this requires specific approaches for the definition of the process transformation actions and instructions. On the other side, in many cases the transformation should be applied to the already running process instances, potentially leading to unpredictable problems and situations.

Apart from modeling process variants, the approaches presented in [127][109] define the notations and formalisms for describing and implementing transformation actions. These actions define the instructions for changing both control flow and data flow model. In [127] the list of possible actions includes add/remove variable, insert task, add/remove successor of a task, change connection type, or change transition condition. In [109] the actions also include changes in the order of the operations, serialization, task deletion, etc.

In these regards the proposed approaches may be considered hybrid, as they include both the explicit variability modeling and the definition of adaptation actions (process model transformations) to be applied when the change is required.

The presented works also propose solutions for dynamic transformation of already running process instances. The solutions are based on the formalization of the transformation activities and the transformation correctness conditions. These formalizations rely on Petri-Net based models, and allow for automated analysis of the corresponding conditions in order to dynamically check the possibility to perform the transformation in a particular case.

### *Semantic constraints for adaptation [172][173]*

| **Synopsis** | |
|---|---|
| **Goal** | Customization, correction |
| **Importance** | Verifying and enforcing consistency and correctness constraints in dynamic adaptation |
| **Kind of contribution** | Method |
| **Applicable to** | Business processes, organizational models |
| **Phase where it applies** | Design-time, execution |
| **Key Ideas** | Formal model for representing organizational models and business process models, for representing changes in these models, and for representing and (semi-automatically) enforcing semantic consistency and correctness constraints on these changes. |
| **Foundations** | Business process modelling, organization modelling, correctness and consistency, dynamic transformation |
| **Validation** | Case study |

Dynamic transformation of enterprise models is also addressed in [172][173]. Both approaches propose a formal framework for representing both the underlying dynamic models and changes in them. Differently to previous approaches, these works also come up with semantic constraints on the changes in these models, and with notion of correctness and consistency of the changes with respect to those constraints. The underlying adaptation frameworks take into account verification and enforcement of the constraints when the changes are performed.
In [172] the underlying models correspond to organization models, where one models the structure of the organizations in terms of roles, persons and their relations, as well as the access rules and privileges of these roles. Semantic constraints have a form of correctness conditions of these rules with respect to the mode. The authors provided a way to perform the semi-automated adaptation of access rules in response to changes made on the organizational model.
In [173] business process modeling is addressed. Similar to the approach of [127], the authors formally define the language and semantics of changes over business process model. Additionally, semantic constraints are introduces in the form of dependency and mutual exclusion of applicable changes. Using these models, a framework for automated verification of process changes with respect to the constraints is proposed.

### 2.5.6 Discussion

The approaches discussed above represent different ways of designing adaptable service-based applications. Below we provide a comparison of these approaches, discuss the possible gaps and overlaps, and try to identify the research challenges that should be addressed by the service science.

The comparison was based on the set of criteria constituting the taxonomy presented in the initial part of the section, namely the adaptation goal (i.e., correction, customization, optimization or prevention); the kind of change that triggers the adaptation (i.e., which characteristics of the application model change); the functional layer, where change takes place (i.e., business process management layer, service composition layer, or service infrastructure layer); whether the change is internal (e.g., faults) or external (e.g., context, requirements) to the application model; the way the change is represented (implicitly or explicitly), and the way the adaptation is modeled (implicit, goal-based, action-based, and explicit variability). The result of the comparison is represented in Table 1.

While the range of existing adaptation approaches is considerably wider than that presented in this section, the number of different design methods is not; the works presented here cover many of them. The state of the art in the research on adaptation of service-based applications covers different functional layers and different types of adaptation.

However, there many research challenges that should be incorporated in the research agenda of the project as a whole and of the partners of the consortium. These challenges dictated not only by the comparative novelty of the adaptation problem in the service area, but also by the high level of fragmentation of the research activities of different institutions.

First, the proposed approaches address only particular aspects of the SBA adaptation. They target only a specific kind of application changes, or a specific functional layer of SBA. The problems of how to identify and model cross-layer dependencies as well as how to design cross-cutting adaptation strategies remain open.

Second, the set of addressed changes and the adaptation solutions is rather restricted; the proposed approaches are not flexible enough to accommodate to a wide range of scenarios or use cases. Most of the approaches deal with QoS changes and the related adaptation strategies, such as service replaceability, or with a particular set of application faults and the recovery actions. Accordingly, the kinds of goals, as well as the adaptation actions are rather simple. While on the one hand this allows for simple and efficient adaptation implementations, on the other hand this prevents creation of comprehensive and multidimensional adaptation methodologies. This requires new languages and techniques that would enable more complex and expressive adaptation capabilities, with respect to those currently presented in the literature. In particular, there is a necessity to provide notations that interleave the goal-oriented and action-oriented specification that would give more freedom to the platform in identification and definition of the necessary adaptation strategies in the application production mode.

In these regards, an important problem is how to integrate various constraints on different aspects of the application functionality with the adaptation technique. On the one hand, there is a need to express real-world domain knowledge and properties, while on the other hand still have to be manageable by the underlying adaptation realization. An important issue is also a possibility to incorporate different types of constraints (e.g., behavioral, QoS), and hence to come up with hybrid analyzers and solvers for those constraints within the adaptation framework.

Third, there is a need to target not only the adaptation types, where the goal is to modify the system in reaction to the changes that already happened, but to adapt the system before these changes take place, i.e., preventive adaptation. This, however, requires novel techniques and methods for the application diagnosis in order to foresee the possible changes and the potential consequences of them.

Finally, in the settings, where the role of hybrid and highly dynamic open systems continuously growth, the adaptation design should target the problem of modelling various aspects of the application context and its evolution. Currently, only few adaptation proposals deal with the context-aware methods and techniques, restricting the applicability of the existing approaches to the new requirements and needs.

| **Approach** | **Adaptation goal** | **Type of change** | **Functional layer** | **Internality** | **Change representation** | **Adaptation representation** |
|---|---|---|---|---|---|---|
| QoS adaptation in grids | Optimization | QoS | SI | Internal | Explicit | Goal-based |
| Adaptation in WS composition | Correction, optimization | QoS, service failures | SI | Internal | Explicit | Goal-based |
| METEOR-S | Correction, optimization | QoS, service failures | SI | Internal | Explicit | Goal-based |
| Discovery framework | Correction | Func. and non-func. requirements violations | SI, SC | Internal | Explicit | Goal-based |
| Interleaved planning and execution | Customization, correction | Service failures, assertion violations | SI | Internal | Implicit, explicit | Goal-based |
| PetriNet-based reconfiguration | Correction, optimization | Fault, requirements | SI | Internal, External | Explicit | Goal-based |
| QoS-based reconfiguration | Optimization | Context | SI | External | Implicit | Goal-based |
| Policy-driven middleware | Customization, correction | Faults or specific application events | SI, SC | Internal | Explicit | Action-based |
| Self-healing compositions | Correction | Assertion violations | SI, SC | Internal | Explicit | Action-based |
| SH-BPEL | Correction | Failures | SI, SC | Internal | Explicit | Action-based |
| AO adaptation | Customization | Functional and behavioral mismatches | SC | Internal | Explicit | Action-based |
| Policy-based adaptation for mobile apps | Customization | Contextual properties | SI | External | Explicit | Action-based |
| Model transformation | Correction | behavior | SC | Internal | Explicit | Action-based |
| SCENE | Correction, customization | Mismatches | SC | Internal | Explicit | Action-based |
| WS-Diamond | Corrections | Faults | SI, SC | Internal | Implicit | Action-based |
| Change identfication | Evolution | Structural and behavioral changes | BPM, SC, SI | Internal, external | Explicit | Explicit variability |
| DySOA | Correction, customization | QoS | SI | Internal | Explicit | Explicit variability |
| PROVOP | Customization | Process context changes | BPM | External | Explicit | Explicit variability |
| Workflow adaptation | Customization | Process context changes | BPM | External | Explicit | Explicit variability |
| Semantic constraints for adaptation | Customization, correction | Process / organization model changes | BPM | External | Explicit | Explicit variability |

**Table 1 Comparison of the design approaches for SBA adaptation**

## *2.6 Quality assurance for service-based systems*

The aim of this section is to give an overview of testing SBAs and related quality assurance approaches. Since these approaches are at the heart of deliverable PO-JRA-1.3.1 "Survey of quality related aspects relevant for service-based applications" [175] we provide a summary of this deliverable here. For further information about this topic please refer to the before-mentioned document.
The authors of PO-JRA-1.3.1 distinguish three classes of quality assurance approaches:

1. *Static Analysis*: In a narrower sense, static analysis "… is the systematic examination of program structure for the purpose of showing that certain properties are true, regardless of the execution path the program may take. " [176]. In a broader sense static analysis can be extended to documents at all stages of the software life cycle, e. g. it can be used to analyse requirements documents such as. goal models and scenarios and design documents such as workflow models and BPEL specifications. Static analysis techniques include for instance. model checking approaches, data flow analysis, symbolic execution and type checking. Since static analysis techniques can be applied at any life cycle stage they complement testing and monitoring approaches described below.
2. *Testing*: "… testing entails executing a program and examining the results produced." [176]. Testing a software system or a SBA requires test data, which are fed into the system. The resulting outputs are than compared to the expected outputs. An error (or defect) results if the actual outputs do not fit the expected outputs. In SBAs these defects are due to services or to service compositions, e. g. a wrong sequence of service requests in a BPEL specification.
3. *Monitoring*: The purpose of monitoring in the software engineering domain is to "… determine whether the current execution [of the software] preserves specified properties; thus, monitoring can be used to provide additional defence against catastrophic failure …" [177]. In SOAs monitoring can be used to observe the status of SBAs – as in traditional software engineering – and of services. Monitoring of services may lead to an adaptation of the SBA, e. g. when one ore more services are not available. The current state of the art of monitoring is described in detail in the deliverable [178].

In [175] the authors distinguish between the design and the operation of a SBA: These phases match the life cycle model depicted in Figure 2 as the design represents the right and the operation represents the left cycle. To see how well the life cycle is covered by quality assurance approaches we counted the respective papers. We found that 21 papers address static analysis in the design phase while the same type of approach was only addressed by 12 papers in the operation's phase. In line with our definition of testing, 30 testing papers addressed the design of SBAs. Interestingly there were also 8 papers which apply testing in the operation phase. The approaches resemble online testing approaches and testing QoS characteristics at runtime. In addition, 36 monitoring papers were examined. Figure 10 shows the correspondence between the life cycle model and the associated testing approaches.

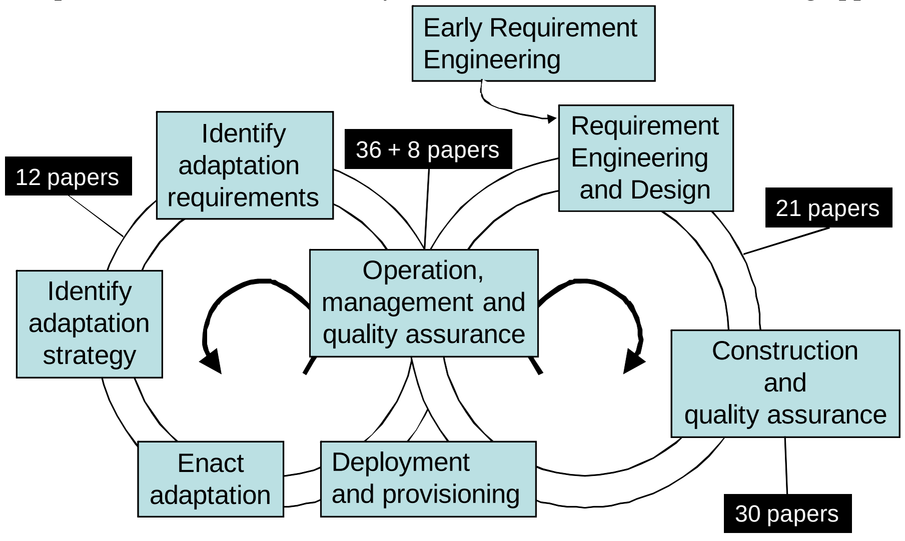

Early Requirement Engineering
Requirement Engineering and Design
Identify adaptation requirements
36 + 8 papers
12 papers
21 papers
Identify adaptation strategy
Operation, management and quality assurance
Construction and quality assurance
Enact adaptation
Deployment and provisioning
30 papers

**Figure 10: Quality Assurance Approaches in the SBA's Life Cycle**

There are a number of quality assurance approaches described in [175] which do not fit in the life cycle model. The approaches include post-mortem monitoring approaches (12 papers), combined testing and monitoring approaches (17 papers) and two papers addressing the whole life cycle of a SBA.
In sum, there is already a rich body of quality assurance knowledge, which was either developed particularly for SBAs or was adapted from traditional software engineering. The challenge is to combine the results of these approaches with engineering principles, techniques and methods, e. g. to achieve an automated adaptation of SBAs due to monitoring results or to closely align requirements with current service provision.

## *2.7 Mechanisms for deployment, operation, and run-time evolution of service-based systems*

Many approaches have been proposed to implement the various phases of deployment, runtime, and evolution of a SBA. This chapter will present the ones we have identified as most promising. We are approaching the survey going through the different phases that concern the life-cycle of a service after its design and development.
We start then from the deployment of the system, and go through the various steps to the reconfiguration of running systems.

We will focus our attention mainly on dynamic approaches, i.e., those approaches that are able to control and evolve a SBA while it is running.

### 2.7.1 Deployment of services and service-based applications

By *deployment* we mean the process of concretely associate services to devices in a real-world system, and the set of choices that must be made to achieve this goal.

*Dynamic deployment*, in particular, is related to the application of such a process in a dynamic context, where changing conditions in the environment must be taken into consideration, together with changes in the requirement, QoS, and other aspects. Dynamic deployment is particularly important in a service-based context where new services or new versions of the same services need to be deployed without stopping or interfering with the normal execution of the others. Thus these approaches are particularly focusing on the autonomy of services from the context where they are deployed. Some of them, concentrating on the possibility of dynamically deploying services, are also dealing with the degree of reusability of services, and how flexibly they can be configured. The main goals of these approaches are indeed both to provide a high level of QoS and to enable dynamic deployment.

A deployment infrastructure should offer the following elements: ways to describe the services to deploy (what) and where to deploy them, a strategy for deployment, and an infrastructure for executing the deployment strategy. In [138] the authors have classified some approaches for describing strategies in four main classes: manual, script-, language-, and model-based approaches. Here we take a broader view and we present approaches that fit all ingredients that we have mentioned above. Moreover, we focus not only on deployment of traditional web services, but also of grid and ad-hoc network services.

#### *2.7.1.1 Approaches to describe the services to be deployed and the resources where to deploy them*

*Using ontologies to describe devices*

| Synopsis | |
|---|---|
| Goal: | Provide a mean to describe devices to achieve dynamic deployment |
| Importance: | Provides a flexible method to fulfil an important ingredient of deployment |
| Kind of contribution: | Technique and implementation in the router context |
| Applicable to | Concretely applied to distributed routers, likely applicable to generic distributed contexts |
| Phase where it applies | Run-time deployment |
| Key Ideas: | Devices (and services) are described by means of ontologies, which also allow for the description of relationships among entities. Profiles of devices can be dynamically retrieved by means of semantic query languages. The matchmaking is performed through hard-coded algorithms. |
| Foundations: | Network architectures, semantic web services, ontologies |
| Validation: | Not provided |

[139] is focusing on the deployment of services on top of routers in a communication network. This approach is interesting since it uses ontologies to provide a rich description of the devices and of the services to be deployed. Ontologies permit to express dependencies among entities.
The infrastructure supporting deployment offers a default approach to distribute the services through the routers. A rule-based language can be used to modify the deployment strategy by exploiting the information in the ontology.

*Gathering information about resources in ad-hoc mobile network*

| Synopsis | |
|---|---|
| Goal: | Providing a strategy for the deployment in ad-hoc mobile networks |
| Importance: | Offering an approach supporting the gathering of device profiles in heterogeneous, mobile (but still known) environments |
| Kind of contribution: | Method |
| Applicable to | Deployment of service-based applications onto mobile, ad-hoc networks |
| Phase where it applies | Run-time Deployment |
| Key Ideas: | Divides a known network of mobile devices into groups with a leader, which collects the profiles of the devices in its group. A component connected to the wired networks maintains the information about the group leaders and forwards the queries according to the relevance of each group. |
| Foundations: | Service-based application concepts, wireless network architectures |
| Validation: | Evaluation of the performance of the prototype in terms of number of required messages |

Deployment in mobile ad-hoc networks must deal with an highly dynamical environment, where the available resources typically have limited capabilities and are often unreliable. In such an environment the need for efficient, flexible, and adaptable deployment strategies arises. To face these needs, in [140] the authors present a protocol for the peer-to-peer deployment of services. The application developer must specify the requirements and the service binaries, while the system allows for an efficient and dynamic deployment of cooperative services, onto heterogeneous devices. In this case there is the need for collecting information about the devices before deciding to install some services on them. The proposed approach is completely distributed. A wireless network is divided into groups, each having its own leader. The group leader knowledge about its vicinity is exploited to avoid blind flooding through the network. The deployment process is treated, in this approach, as a mapping function, having as input the services to be deployed and the devices where they can be deployed. The requirements of services cooperating in a large application must be provided, especially in the case of services that should be dynamically mapped on new nodes to cooperate in a new application. The devices must be described as well. Their profile mainly consists of dynamic attributes, changing from time to time. To retrieve a device where to deploy a certain resource, authors use a retrieval algorithm taking into account the QoS provided, the adaptivity, and other variables.

#### *2.7.1.2 QoS-aware approach for deployment strategy*

| Synopsis | |
|---|---|
| Goal: | Providing mathematical support to choose the service configuration with the highest QoS |
| Importance: | Providing a general purpose mathematical support in the definition of service deployment strategies |
| Kind of contribution: | Analytical Method |
| Applicable to | Grid Systems |
| Phase where it applies | Design time |
| Key Ideas: | A set of candidate configuration meeting the requirements is identified, the deployment of each service in a set on different *capsules* is considered, with the aim of maximizing the provided QoS at a capsule-level and then globally |
| Foundations: | Grid architectures, optimization functions, mathematical deviation models |
| Validation: | Proved through an application of the approach to a use case |

The work proposed in [141] can be used in the definition of a strategy for selecting the devices on which to deploy some services. The target of the approach is the Grid context. The aim is to define a mapping function between the services belonging to a service composition and the available grid resources with the objective of maximizing some specified QoS requirements.
This approach works as follows. Firstly, a set of candidate service compositions satisfying the functional requirements of a service is created. Then, they evaluate the mapping alternatives for each service composition. The objective of each service deployed onto a so called capsule, i.e., an executable unit including a number of resources, is to maximize the provided QoS, then maximizing the QoS at application level. They also provide a way of measuring quality, and a quality deviation model, which indicates how to minimize the distance between the aspired and the actual quality level. Summarizing, the output of this approach is a mathematical framework providing a way to find a configuration that provides the nearest QoS to the aspired one. In [141] they also provide a use case proving the validity of the approach.

#### *2.7.1.3 Approaches for describing information and strategies for deployment*

*Language-based approaches*

| Synopsis | |
|---|---|
| Goal: | Provide a language for the specification of deployment strategies |
| Importance: | Provides an easy-to-use mean for the specification of run-time deployment strategies |

| | |
|---|---|
| Kind of contribution: | Technique and Tool (Language) |
| Applicable to | Grid context |
| Phase where it applies | Run-time deployment |
| Key Ideas: | Expressing the deployment strategy through a java-like language, where services are language constructs and the deployment is achieved through proper statements and can be regulated according to certain policies |
| Foundations: | Grid architectures, programming languages definition |
| Validation: | Evaluation of the experiments performed on use case (NGB – NAS Grid Benchmark) |

In language-based approaches the service deployment process is specified through an ad-hoc language, requiring no configuration and allowing for on demand deployment, thus reducing the waste of resources.
Here we are presenting Abacus, which is an example of language-based approach to deployment. Abacus is a service-oriented, java-like programming language, presented in [143]. The context addressed by Abacus is, once again, the grid context. In Abacus a service is represented through a language construct, and the declaration of a service is similar to a class declaration in java. Each service is represented by a language construct and is stored in a virtual space, where it is identified thanks to a UUID which makes the service always available, also when it moves from a site to another. This space is managed by a compiler and a run-time system.
Thanks to the functions provided by the language, Abacus allows for on-demand deployment, and it is possible to achieve dynamic deployment according to user-defined conditions. Indeed, Abacus allows for different deployment strategies, and through the language it is possible to specify when a given service should be deployed.
The approach introduces some time overhead that can be reduced through the improvement in the Abacus substrate, i.e., the AVM (Abacus Virtual Machine).
Abacus reduces the number of steps required by the development and deployment of grid applications.

*Script-based Approach*
Script-based approaches, as explained in [144], simply consist in writing scripts that define which service should be deployed on which resource. Therefore, they are very easy to use, however, they only are suited for small scale applications. In [144] authors present a script-based configuration approach for the Unix environment, based on SSH and developed in Python. Their solution assumes the existence of a service pool server, acting as a server repository.
Another script-based approach is implemented by Nixes 0[145], a platform which suites well the context where it has been deployed (PlanetLab), but it doesn't scale well to other applications.
Both implementations consist of concrete tools.

#### *2.7.1.4 Infrastructures for executing deployment strategies*

| Synopsis | |
|---|---|
| Goal: | Providing an infrastructure to enable dynamic deployment reacting to the changes in the demand |
| Importance: | Allows for on-demand resource provisioning |
| Kind of contribution: | Method and prototype |
| Applicable to | Grid context |
| Phase where it applies | Run-time deployment |
| Key Ideas: | A broker receives the requests and forwards them to the resources where the requested service is deployed. When the demand increases, a service can be |

| | |
|---|---|
| | deployed on a new resources, according to policies specified by the service provider. |
| Foundations: | Grid services and architectures based on grids |
| Validation: | Proved in experiments testing the prototype |

In [142] the authors propose an approach consisting in performing the deployment only when it is needed. The context to which this approach has been applied is the context of Grid. The main goal is to allow the provider of a service to adjust the resources held by the service according to the demand. The presented architecture mainly consists of two components: a UFS (Universal Factory Service) component, which should be deployed onto each resource in the Grid and which allows the service providers to dynamically deploy their services, and Door, a service broker, that manages the requests for services, and the deployment of services onto new resources when needed, according to given policies. Door both forwards the service requests to the resource where the service is deployed, and deploys the service on new resources dynamically in reaction to the amount of demand, according to given policies that state when a service should be deployed on a new resource and when a resource should be released. When Door receives a request from a service user, it asks the resource where the service is deployed to create a new instance of the service, or uses UFS for the deployment of the service onto a new resource. This way there is no need to modify the configuration of the deployed service.
The developers of this approach have performed some tests on a grid testbed. These tests show that when the service demand increases, the number of resources where the service is deployed increases too, thus demonstrating that the implemented framework behaves correctly.

### 2.7.2 Run-time management of Services

The approaches described within this section focus only on a part of the main features characterizing services, i.e., those which are more related to dynamism and flexibility, such as service autonomy, service reusability, loose coupling and flexible configurability. Moreover, the first approach presented here focuses on enabling service interoperability.

*The WSCF framework to support Web Service-based Application*

| Synopsis | |
|---|---|
| Goal: | To provide a framework supporting cooperation among different COTS (Commercial Off-The-Shelf) middleware through web service containers |
| Importance: | The framework also allows for dynamic and adaptive management of services |
| Kind of contribution: | Method and Tool |
| Applicable to | Enterprise middleware (CORBA, J2EE) to enable cooperation among different enterprises |
| Phase where it applies | From the development of services on |
| Key Ideas: | Wrapping COTS components in web services enabling transparency towards service users; providing engines that enable automatic, run-time management, which can be regulated through customizable policies. |
| Foundations: | Web services concepts, conventional middleware architectures, Semantic Ontologies |
| Validation: | Tests performed on a prototype implementation |

The WSCF Framework [156] aims at enabling cooperative service-based applications. It is based on wrapping the Commercial Off-The-Shelf (COTS) components into Web Service Beans, which are then exposed through a standard interface. The main part of the architecture is a *Web Service Container*, which provides a layer of abstraction allowing for the exposure of COTS components as Web Services. COTS components belong to the most common distributed middleware, such as

CORBA and J2EE. The objective of WSCF is indeed to enable cooperation among different middlewares. Thus WSCF provides transparency at the level of service provider, of the communication protocol, of time (to achieve scalability) and of adaptation. The complete architecture includes the Web Service Container and various application, supporting the running of services on a SOAP bus. Precisely, the architecture is made up of three layers: a *Resource Layer* providing access to the components, a *Web Service Container*, and a *Service Layer* guaranteeing adaptation transparency. The most interesting layer is, as said, the *Web Service Container*. It includes a *configuration engine*, which permits to customize the policies in charge of managing the service adaptation, a *service monitoring engine*, providing always up-to-date information about service availability, and a *service adaptation engine*, which provides functionalities to manage the Web Service Beans (which wrap the COTS components) and reacts to the information gathered by the monitoring service about the availability of services. This way WSCF grants an adaptive, run-time management of services.
Apart from supporting lifecycle management, WSCF also provides applications that support services for building cooperative applications. For the discovery functionalities, for example, the framework relies on a dynamic infrastructure, Stratus, which permits to consider also non-functional parameters (such as behaviour and quality) in the service selection. Moreover, Stratus defines a Web Service semantic description language, QWSDL, which describes services on the basis of the provided QoS. QWSDL introduces a shared ontology to avoid heterogeneity in service description, and supports rich description logic for reasoning. WSCF also offers applications supporting Web Services composition through a service called StarWebFlow.
This framework has been validated by observing the performance of an implementing prototype.

*Mule, an Enterprise Service Bus providing lifecycle management of services*

| Synopsis | |
|---|---|
| Goal: | To provide a framework that supports the lifecycle of various objects and their exposition as services as well as their communication through various different channels |
| Importance: | It the open source best known Enterprise Service Bus |
| Kind of contribution: | Tool |
| Applicable to | Services |
| Phase where it applies | From the deployment of services on |
| Key Ideas: | Wrapping COTS components in web services enabling transparency towards service users; providing engines that enable automatic, run-time management, which can be regulated through customizable policies. |
| Foundations: | Web services concepts, conventional middleware architectures |
| Validation: | Used in various cases by various different users |

Mule [158] is a flexible Enterprise Service Bus, supporting the encapsulation of many kinds of component within services and communication among services over a wide range of communication channels. It makes no assumptions about the structure of the messages or about the interfaces exposed by the services. It is adaptive to its surrounding technologies rather than prescriptive. All the interactions among components are managed by Mule in a transparent way. Moreover, one of the functionalities provided by Mule is to allow for a customisable management of the lifecycle of the deployed objects.

### 2.7.3 Approaches for run-time Binding and Re-Binding

Another aspect particularly challenging in the run-time management of service-based architectures is *binding*, where by *binding* we mean the process of associating a service request to a service offer. Here we are particularly focusing on the run-time management of binding, and on *re-binding*, i.e., the dynamic reconfiguration of binding in order to face changes in the environment or in the requirements of the users. Thus, this section describes approaches focusing mainly on service composability. The approach described in 2.7.3.1 also provides a way to improve coarse granularity.

#### *2.7.3.1 Automatic Service Binding: specification and matching of offers and requests*

| Synopsis | |
|---|---|
| Goal: | Providing a semantically-rich request language allowing for dynamic binding |
| Importance: | Aims at enabling dynamic adaptation and focuses on a particular, not very explored subject, i.e., the definition of requests |
| Kind of contribution: | Technique |
| Applicable to | Service-based applications |
| Phase where it applies | Run-time binding |
| Key Ideas: | A precise definition of the request language enables automatic binding. The defined language is based on fuzzy sets of objects representing services, a matching algorithm based on membership functions, and the possibility for the requestor to specify some matching preferences. |
| Foundations: | Programming language, service-based architectures concepts |
| Validation: | Tests performed about a more recent implementation, within DIANE |

In [146] an approach aiming at the automation of Dynamic Service Binding is proposed. Dynamic binding allows for the dynamic selection of the service provider, granting more robustness and efficiency. Such an automatic mechanism can only be enabled by a detailed, machine-readable description of the offered services, a detailed description of the service request, and an automatic match between offers and requests. This approach consists in the development of a language for the specification of the requests, and a corresponding, ontology based, language for the description of service offers. The request is integrated with a complete semantics for the specification of preferences, and starting from such a specification a matcher to test an offer against a request is generated. The matcher can then use the set of preferences to rank the offers. The matching process is then completely controlled by the requestor.
The language for the specification of the requests is based on fuzzy objects. Request sets contain *Service* objects, representing suitable services for the requestor. To automatize the binding, the authors of the paper introduce fuzzy sets of objects: to avoid problems due to multiple matching services or the absence of any service meeting the request, the matching of the offered services with the requested ones are regulated through a membership function, instead of a crispy one.
Summarizing, this approach introduces a definition of a language for service requests specification which also includes the possibility to specify some preference about how the matching should be performed. The precise definition of such a language allows for an automatic binding.
On the side of offers, the authors focus on building a semantic description of the offered services.
No implementation of this approach exists at the time [146] was written, but a description of further works by the same authors can be found in later papers. In [147], for instance, a language (DSD – DIANE Service Description) for an ontology-based specification of services is presented. In this paper they also provide an approach which integrates service composition within a matchmaker. This way, it is possible to fulfil the requests for those services requiring multiple connected effects. The matchmaker integrating the composition approach has been implemented and tested. From their tests the approach results to be scalable enough to be used in real-world cases for “on the fly” matchmaking.

#### *2.7.3.2 Adaptive Location-aware Service Binding*

| Synopsis | |
|---|---|
| Goal: | Defining an approach for the automatic, location-aware, adaptive binding |
| Importance: | Offers a mean for the automatic re-binding when a service is found to be no more available |
| Kind of contribution: | Method and implementation (Atlas) |
| Applicable to | Service-based applications on mobile devices |
| Phase where it applies | Run-time binding |
| Key Ideas: | Atlas keeps on sampling the position of the user of a mobile device; when the user enters a new region where a given service is no longer available, the middleware automatically performs a re-binding to a new (similar) service |
| Foundations: | Service-based applications concepts |
| Validation: | Small-scale controlled tests |

A binding is adaptive when it is able to change itself as needed, according to changes in the state of the environment or in the requirements of the users. The adaptivity of binding is, obviously, particularly important in the case of highly dynamical environments, such as the context of mobile network devices, where location-based applications are widely used (for instance, in the case smart phones). In [148], the authors present a middleware to enable adaptive binding in such a dynamic context. The presented implementation is *Atlas*. Atlas continuously tests the position of the user of mobile devices, and, when it finds out that a given service is no longer available (for instance, because the user enters a different region), it autonomously builds up a binding to a new service. Atlas looks transparently for a new service matching the application requests, thanks to the fact it continuously monitors the validity of a service. The middleware includes adapters that translate the interface of services offered by the provider into an Atlas interface, thus hiding the service protocols to the application developer.
Atlas reacts to service failures according to two policies: a *failure determination* policy and a *failover* policy, both hard-coded within the framework.
An implementation of this middleware exists, and it has been tested. In the version presented in [148], in which a new binding is only built when a user enters a different region, the introduced overhead is quite low, but it could increase if the validity of services is checked at every location update.

#### *2.7.3.3 PAWS: a framework allowing for runtime adaptation of web-services*

| Synopsis | |
|---|---|
| Goal: | Service selection and process optimization with run-time adaptation, focusing on QoS, mediation, and self-healing |
| Importance: | Proposing a framework to separate design time and allowing for run-time adaptation |
| Kind of contribution: | Models + Tools |
| Applicable to | Web-service processes |
| Phase where it applies | Design-time composition and run-time binding and re-binding |
| Key Ideas: | Designing a flexible process at design time, with a selection of candidate services to be invoked, with prepared interfaces for mediation purposes, and agreed QoS. At run time, the concrete services are selected based on the process execution context and QoS optimization, mediation executed, and self-healing applied in case of failure |

| | |
|---|---|
| Foundations: | Semantically enriched Service Registry, annotated processes, process QoS optimization, QoS agreement, self-healing engine |
| Validation: | Action Research + Case Study |

The PAWS framework [160] (see http://black.elet.polimi.it/paws) proposes an interplay between design-time and run-time activities. Starting from an abstract process definition, in the design time a selection of candidate services is performed using a semantically enhanced registry, defining mapping information to be used for mediation at run time, and negotiating QoS levels with potentially participating services. At run time, concrete services are selected, based on QoS global constraints and QoS optimization techniques, and services are invoked through a mediation engine to semantically transform input and output messages.
The run-time activities are managed by three modules: a *Process Optimizer*, a *Self-healing module* and a *Mediation engine*. The *Process Optimizer* is in charge of guaranteeing both local and global QoS, according to the constraints required by the user. The *Self-healing module* enables adaptation, performing semi-automatic actions in reaction to failures. The recovery could imply service reinvocation or substitution. If the recovery requires to substitute the running service, a new service is selected, among the candidates. Finally, the *mediation engine*, which is set up at design-time, redirects the invocations of the deployed process to the selected services.
Two proofs-of-concept have been implemented for this framework, proving the reduction of the design effort, and suggesting to focus on the improvement of performances.

#### *2.7.3.4 WS-Binder, a framework to enable dynamic binding*

| Synopsis | |
|---|---|
| Goal: | Providing a framework for run-time service discovery and binding |
| Importance: | Allowing for automatic, run-time rebinding and run-time, context-aware services selection |
| Kind of contribution: | Method and tool |
| Applicable to | Web Services context |
| Phase where it applies | Just before composition execution , run-time discover, run-time binding |
| Key Ideas: | The tool is a binding framework offering many binding mechanisms and policies to regulate binding. Run-time binding occurs when a service is no longer available or does not provide the estimated QoS. It is possible to select services at run-time. |
| Foundations: | Web services concepts and basic technologies |
| Validation: | The tool has been applied to a concrete case study and evaluated |

*WS Binder* [157] is a framework that enables dynamic binding of service compositions. The framework offers the possibility of performing different kinds of bindings, and to write customized policies that regulate the binding itself. Among the capabilities offered by the framework are:

- A *global binding* function, that finds the sub-optimal set of bindings through the exploitation of Genetic Algorithms. This approach is run just before the execution of the composition starts.
- A *run-time local binding*, that allows for the selection of the services actually available during execution, and for basing the selection on context information. This approach can be seen as complementary to the previous one.
- A *run-time workflow slice re-binding*, which occurs when a service is found to be no longer available or the provided QoS violates the constraints of the requestor. If this is the case, the execution is stopped, and restarted after a new global binding has been performed.

The binding can be customized by setting some preferences, like QoS objectives, or the inclusion/exclusion of some services. The framework also offers an interface that monitors the execution of the system, and allows for its management. At run-time, if an actual binding has not been previously specified or if a failure occurs, the framework will select the slice of the workflow to be re-bound, on the basis of the specified preferences.
The framework has been applied to a concrete case study belonging to the tourism context, and it has been evaluated paying attention particularly to the response time.

### 2.7.4 Approaches to support run-time Discovery

By *discovery* we mean the process of locating the services providing the required functionalities.

Many approaches to discovery have been proposed along years.
Some of them concern design time while others runtime. Here we are only considering run-time approaches. Some are based on information about the functional and non functional requirements to be fulfilled, some others on the architectural structure of the SBA, and others include the use of semantic descriptions of non functional aspects of services and requirements. However, they all are focusing on discoverability of services. Moreover, they also deal with coarse granularity and implementation neutrality.

*A discovery approach based on the behaviours of services*

| Synopsis | |
|---|---|
| Goal: | Enable dynamic discovery based on the behaviour of services |
| Importance: | Provides a method for representing web services and for run-time, dynamic discovery |
| Kind of contribution: | Technique |
| Applicable to | Web services context |
| Phase where it applies | Run-time discovery |
| Key Ideas: | Representing each service through an automaton, representing requests as sets of the automata representing the required services; the discovery consists in a search among the automata. |
| Foundations: | Service-based systems concepts, finite state automata, web services standards |
| Validation: | Not provided |

In [149] the authors introduce an approach enabling dynamic discovery through a peer-to-peer framework. To perform the matching the proposed approach takes into consideration, together with the functionalities provided by a service, also its *behaviour*. Moreover, the system includes a reputation model to rank the discovered services. Being this a peer-to-peer architecture, the typical limits of a centralized system (such as scalabilty and single point of failure) are overcome.
The context addressed by this approach is that of web services. A web service is modelled as a triple consisting of *implementation*, *service* and *requests*, where the *implementation* is considered to be the BPEL/DAML-S description of the service, and is represented as a finite automaton; the *service* is represented by a finite automaton describing the interactions observed by its user; the *requests* are the set of automata including the services required by the considered one for its execution.
Each execution of a web service is considered as made up of a number of interactions, and it correspond to a path on the *service* automaton, from the start state to a final state.
The discovery process consists of a search among the services through their automata, that is, their behaviour. Similarly, new services are published according to their *service* automaton.
The implementation of the approach as a peer-to-peer structure relies on Chord [150], which implements a distributed hash table. The overall system only works properly if a global dictionary for the deployed services exists. Such a common dictionary could for instance be achieved thanks to a DAML-S ontology.

*A discovery approach for Semantic Web Services based on ontologies*

| Synopsis | |
|---|---|
| Goal: | Providing a Discovery Tool based on semantics for Semantic Web Services |
| Importance: | Offers a way of achieving run-time discovery based on semantics |
| Kind of contribution: | Method and Tool |
| Applicable to | Semantic Web Services |
| Phase where it applies | Run-time discovery |
| Key Ideas: | Performing discovery through a three phase process: starting with a keyword based matching on ontology concepts, using pre- and post- conditions for a further, logical matching, and providing ranked results |
| Foundations: | Semantic Web Services concepts, Ontologies, Logic Programming |
| Validation: | Evaluation of the performance of the developed prototype |

An example of this approach is presented in [151], where they describe the discovery engine developed within the INFRAWEBS project. The context this approach refers to is the Semantic Web Services one. The described engine receives as input the WSML description of the goal, and provides a list of Semantic Web Services matching this goal.
The Discovery process presented in [151] is made up of three steps:

- *Pre-filtering* step, consisting of traditional text-based matching algorithms;
- *Logical matching* step, based on pre- and post- conditions of both the goal and the candidate services selected in the previous step;
- *Finalizing* step, when the selection of services is improved thanks to QoS data gathered during previous executions.

The *pre-filtering* phase can be based on keyword matching on ontology concepts. Particularly, the matching is based on *axioms*, that must be proper formulated to ensure the correct use of semantic web services.
The logical matching, instead, is performed thanks to the use of Prolog. Both ontologies and goals are converted to Prolog, and then a proper algorithm, developed in Prolog, allows for the matching between the clauses of the goals and the clauses of the web services. For each web service selected through pre-filtering an attempt of matching with the goal is made. Then, a ranked list of results is provided to the user. The ranking is based on some QoS aspects, including the number of clauses ignored on the service and on the goal side.
An implementation of the described algorithm has been developed, and its performance have been evaluated, but since no benchmark of discovery approaches exists it is not possible to make a concrete comparison with the performance of other approaches.

#### 2.7.4.1 A discovery approach for supporting Web service run-time substitution

| Synopsis | |
|---|---|
| Goal | Semantic-based Web service discovery based on similarity |
| Importance | It returns a ranked set of services having a WSDL description similar to the requested one. |
| Kind of contribution | Methods and Tool |
| Applicable To | Services described by WSDL documents |
| Phase where it | Design-time and run-time |

| | |
|---|---|
| applies | |
| Key-ideas | Similarity functions are defined to evaluate service API similarity in view of substitutability. Similarity is based on structural and semantic evaluation. The similarity among WSDL quantifies the effort for re-coding the client-side when a substitution is required. QoS requirements satisfaction criteria are also provided. |
| Foundation | service similarity, QoS evaluation |
| Validation | Action Research + Case Study |

The approach described in [164] enables Web service discovery by comparing the related WSDL descriptions. Such an evaluation reflects how much will be the effort in case a Web service should be substituted with another one in terms of re-coding the client-side. The approach takes into account the relationships between the main elements composing a WSDL description (i.e., portType, operation, message, and part).
The underlying algorithm combines both semantic and syntactic aspects of the Web service that can be derived from a WSDL description. The semantic aspects are related to the goal of the Web service and correspond to the names used for the operations and parameters. Instead, the syntactic aspects can state the compliance between the input and output structures and the adopted data types. The algorithm assumes that, as usually occurs, the WSDL specification of the Web service interface is (semi-)automatically generated by a tool starting from a software module, such as a Java class. This implies that the resulting description will probably reflect the naming conventions usually adopted by developers.
The approach relies on a domain specific ontology where terms usually adopted in a given scenario are organized according to semantic relationships such as synonymy (two words have same meaning), antinomy (two words have opposite meanings), homonymy (a word with more meanings). Moreover, a general purpose ontology, e.g., WordNet, supports the matching between terms that are not included in the domain specific ontology.
The evaluation allows ranking available services for substitution, to minimize missing functionalities, incomplete information, and parameter transformations.

#### *2.7.4.2 Context-aware Runtime Web Service Discovery*

| Synopsis | |
|---|---|
| Goal: | Allowing for context-aware runtime discovery of services |
| Importance: | Defines context broadly and uses it to perform runtime discovery |
| Kind of contribution: | Method + tool (available as a service) |
| Applicable to | Web services |
| Phase where it applies | Run-time services management |
| Key Ideas: | The runtime matching is based on structural, behavioural, and context conditions, where the context includes, apart from the common concepts of context, some dynamically changing information. The context operations are described and accessed through ontologies. |
| Foundations: | Graph analysis algorithms concepts, web services concepts, web service technologies |
| Validation: | -- |

In [163] the authors present a framework supporting the description and the execution of context-based discovery queries. The platform allows for the runtime evaluation of structural, behavioral and context conditions for discovery and provides a query language enabling the specification of complex queries.
At runtime, the platform accepts queries composed by a BPEL description of the workflow in which the service should be deployed, the WSDL describing the required service, and a description of the context conditions that the service should satisfy. The platform represents through a state machine the

part of the workflow interacting with the required service, and the WSDL is translated into an operation type graph. Then the registry is contacted to find matching services. The matching process passes through three stages: the first one concerns a structural matching, the second one a behavioural one, while the third one the context conditions are evaluated. The *context* concept here is broader than the common one, since it also includes some other dynamically changing information, such as the information concerning the provided QoS. Moreover, context operations are grouped into web services, and then available through remote invokation. The semantic categories of context operations are described by ontologies.

### 2.7.5 Approaches to support the run-time evolution of service-based applications

Many approaches supporting dynamic adaptation of SOA have been explored; they allow for dealing with changes in the state of the architecture, in the context in which services are executed, as well as changes in the requirements of the users, adapting the configuration of the system to meet new needs.

During the last years many studies have focused on how a *semantic* approach can help in achieving a high degree of adaptation in the reconfiguration of SOA. Together with these recent interests, also more “traditional” model-based approaches are cited here, as well as approaches that particularly focus on the maximization of the provided QoS. With reference to the features of services, these approaches deal meanly with service reusability, composability, and flexible configurability.

As defined in 2.5.4, in these approaches the actions to be performed on the system are predefined by the adaptation framework.

#### *Quality driven service composition [111][165]*

| Synopsis | |
|---|---|
| Goal: | Optimization |
| Importance: | Supports dynamic composition of services with the goal of optimal valuation of service qualities |
| Kind of contribution: | Technique |
| Applicable to | Service compositions |
| Phase where it applies | Deployment |
| Key Ideas: | Multi-dimensional optimization of quality service metrics |
| Validation: | Case study, experiments |

In these works the authors propose an implicit approach towards dynamic service composition based on multi-dimensional optimization of quality of service metrics. While the used techniques are different, the conceptual methodology is similar. In the approaches the composed process (e.g., in BPEL) is designed as a workflow composing elementary tasks. At run-time a concrete elementary service is selected to perform a particular task from a community of services that provide the same functionality, but have different quality characteristics. The description of the services, therefore, should include not only functional aspect, but also non-functional properties that are required in the selection process. The authors identify different sets of the relevant quality properties, such as price, duration, reputation, reliability, availability, and define the corresponding aggregation functions for each of them. The predefined goal of the approach is, therefore, at run-time optimize the values of these functions. Since this model is multi-dimensional, the weights should be provided in order to define the global criteria. These weights may be predefined, or set by the end user (as a set of preferences).

### *Reputation-based dynamic maintenance of composed services [166]*

| Synopsis | |
|---|---|
| Goal: | Correction |
| Importance: | Supports maintenance of dynamic service compositions, when the component services fail or become defective |
| Kind of contribution: | technique |
| Applicable to | BPEL processes |
| Phase where it applies | Execution |
| Key Ideas: | Pro-active reputation information propagation |
| Validation: | Discussion |

The approach targets the problem of maintaining dynamic service compositions, when the component services fail or become defective. The composition is designed as a BPEL process. However, the run-time criterion for selecting the best possible service is different. First, the decision is made for each invocation of the component services. Second, the decision is driven by the only factor, which is service reputation. However, the approach adopts a proactive approach, where the processes proactively provide the reputation information about the usage of a service. If the service invocation was successful, the reputation is positive, while in case of failure the value degrades. This approach allows improving the quality of selection. However, the system integrator has no way to control or alter such selection and adaptation process. Interestingly, this technique does not require extra description of the component services needed to drive adaptation strategy.

### *P2P architecture for self-healing processes [167]*

| Synopsis | |
|---|---|
| Goal: | Correction |
| Importance: | Supports to dynamic look-up and replacement of elementary services that failed during the execution of the process |
| Kind of contribution: | Technique |
| Applicable to | BPEL processes |
| Phase where it applies | Execution |
| Key Ideas: | Peer-to-peer resource management for publishing and discovery and binding of the necessary services |
| Validation: | Discussion |

In [167] the authors address the problem of self-healing execution of BPEL processes. The problem amount to dynamic look-up and replacement of elementary services that failed during the execution of the process. At the implementation level, this amounts to replacing a service invocation with a loop containing candidate look-up and invocation, until successful result is obtained. The most critical element of the approach is the service lookup, and the authors propose to exploit a peer-to-peer Resource Management Framework for this purpose instead of centralized repository. In this framework data elements are published, and it is possible to subscribe for changes in their definitions. The authors proposed a resource format to embed the WSDL service descriptions into this framework for robust look-up and binding.

### *Generation of adapters for service integration[168][169][170]*

| Synopsis | |
|---|---|
| Goal: | Customization |

| | |
|---|---|
| Importance: | During the design and development time not all the ways, in which the services interact, are foreseen. The approaches allow for resolving mismatches between the provided and the required functionalities. |
| Kind of contribution: | Techniques and methods |
| Applicable to | Service-Based Application |
| Phase where it applies | Composition development and provisioning |
| Key Ideas: | Automatic generation of mediators based on predefined requirements (e.g., deadlock freeness) or semi-automated methodologies for identifying and modelling instructions and procedures for adapting the specification (transformation templates or commands). |
| Validation: | Case study |

Several works target the adaptation techniques to overcome various types of mismatches that may occur among services developed by different parties. Indeed, during the design and development time not all the ways, in which the services interact, are foreseen. The mismatches between the provided and the required functionalities should be resolved dynamically.

A common approach to target different types of mismatches is to automatically create certain mediators that make the services correctly interact. Depending on the level of the service specification, this may amount to signature-based adaptation (syntactic properties of the exchanged messages), ontology-based adaptation (exchanged data represent different concepts), or behavior-based adaptation (differences in behavioral specification). In order to perform all these kinds of adaptation, the service descriptions should provide the corresponding models at the different levels of Web service stack. In particular, behavior-based adaptation, generation of adapters [168][169][170], requires that the participating service descriptions are equipped with the interaction protocol the service implements. Such a protocol may be defined, for instance, as an abstract BPEL process.

While the approach presented in [168] aims at automated generation of an adapter that guarantees the non-locking interaction of the services, the approaches of [169][170] transfer the problem to the system integrator. In [169] the authors propose taxonomy of different behavioral mismatches and a set of parametric behavioral patterns that may resolve the mismatch. The corresponding pattern is instantiated when the mismatch is detected and proposes it to the application integrator as a possible adaptation strategy. In [170] the authors propose an algebraic model of six transformation operators and the corresponding visual notation that permits, given a pair of required and provided interfaces, construct the necessary adaptation. Based on this construction, a mediation engine performs the necessary run-time actions for processing and managing the messages and service invocations.

*Reconfiguration Approach based on local knowledge*

| Synopsis | |
|---|---|
| Goal: | Enabling automatic, run-time reconfiguration of service-based systems |
| Importance: | Allows for run-time adaptation of the system configuration according to changes in the context |
| Kind of contribution: | Technique and tool |
| Applicable to | Web Services context |
| Phase where it applies | Run-time reconfiguration |
| Key Ideas: | The properties of a complex system are defined starting from the local knowledge, defined as the knowledge about its immediate structure. Local knowledge is used to reconfigure the structure of the system when a change in the context is found, and is propagated upward when needed |
| Foundations: | Web services concepts, compositionality concepts |

| | |
|---|---|
| Validation: | Not provided |

To allow for an adaptable reconfiguration, that is, to allow for the automatic reconfiguration of a system according to the changes in the context, the system should be *context-aware*, i.e., it should be able to identify the changes in the environment, and choose the best reaction. With this aim, in [152], the authors propose an approach based on semantic properties, which are used to describe a service and its current status. Thanks to these properties it is then possible to change the overall configuration of the system, according to the changes in the local requirements.
This approach relies on the compositionality concept to provide a definition of *local knowledge* as knowledge about the "immediate" structure of a complex system, the components of such a system and the dependencies among the components belonging to the same layer in the architecture. Thus, on the basis of the local requirements of each layer and the properties provided by the web services, the properties of a complex system can be defined locally, and then propagated upward as needed, when some reconfiguration is required.
This approach is used in the web services context. A concrete prototype implementation exists, but no validation seems to have been provided.

*Reconfiguration approach based on SDO (Super Distributed Objects)*

| Synopsis | |
|---|---|
| Goal: | Providing an approach for run-time reconfiguration of services |
| Importance: | Supports the continuous provision of services |
| Kind of contribution: | Technique and example prototype |
| Applicable to | Service-based architectures |
| Phase where it applies | Run-time reconfiguration |
| Key Ideas: | Real-world resources are modelled as SDO (Objects). Requests for a service are accomplished by a selection of SDO. When a change in the context is encountered, the system automatically reconfigure SDOs without interrupting the provision of the service. |
| Foundations: | Service-based architectures concepts |
| Validation: | Not provided at the time the paper was written (2004) |

An approach similar to the previous one (although not strictly model-based) is presented in [129], where the authors propose an architecture based on *modules*, called Super Distributed Object, which model resources and objects. According to this approach, the reconfiguration of the system is achieved through the re-combination of these SDO in order to fulfil the requests for a service.

*MAIS project*

| Synopsis | |
|---|---|
| Goal | Requirements elicitation and context-aware web applications development |
| Importance | Focusing on context modelling and automatic generation of web services |
| Kind of contribution | Method and tool |
| Applicable To | multichannel, mobile, context-aware service-based applications. |
| Phase where it applies | from requirements elicitation to service composition |
| Key-ideas | The MAIS methodology concerns the stages from requirements elicitation to service composition. Code generation functionalities are provided based on WebML. The methodology is based on UMMAIS, Unified Methodology for Multichannel Adaptive Information Systems. Flexible service design is supported. |
| Foundation | use cases, WebML, context modelling, flexible service modelling with QoS and |

| | |
|---|---|
| | context parameters |
| Validation | Action Research + Case Study |

In [135][137] the authors present a methodology for designing context-aware service-based applications, described in the UMMAIS work [136] and in the MAIS book, focuses on context-aware service compositions developed within the MAIS project (www.mais-project.it). Context elements include personalization, location, role modelling and user profiling. Context changes are considered as well. Service compositions in flexible services are designed. A special emphasis is put in the design of front-end services, which are based on back-end flexible service compositions.

*Adaptation in Semantic Web Services*

| Synopsis | |
|---|---|
| Goal | Providing a tool for web services protocol mediation |
| Importance | Allowing for the interaction between two services relying on different protocols |
| Kind of contribution | Method and tool |
| Applicable To | Web services |
| Phase where it applies | Run-time binding |
| Key-ideas | Protocol mediation allowing for the automatic adaptation of the service requester behaviour meeting the constraints of the provider’s interface, by abstracting from the existing interface description. A shared ontology is used to understand the semantics of the domain actions. |
| Foundation | Web services concepts, ontology concepts |
| Validation | -- |

[171] focuses on web services protocol mediation by providing a framework that allows for the interaction between two services despite the difference of the protocols they rely on. Mediation allows automatic adaptation of a service requester behaviour while meeting the interface constraints of the service provider. This is done by abstracting from the existing interface descriptions before services interact and instantiating the right actions when contracting a service. To be possible, a framework managing these levels of abstraction was provided. It consists of defining abstract primitives used by services during their interaction that will be mapped to concrete primitives that represent the real actions in terms of messages exchange between the two communicating parties. Monitoring is however necessary for ensuring the correct use of the abstract primitives that must follow the constraints defined for the service in a low level description language. It could be stated in a state chart expression. An ontology of shared concepts (of the business domain for example) where each concept used in the protocols is defined is also necessary for understanding the semantics of the domain actions which in addition have to be exploitable by the machine.

### 2.7.6 Approaches to support on-the-fly composition of Services

Among the approaches facing the run-time management of services, some approaches particularly focus on providing the possibility to perform on-the-fly composition of services. On-the-fly composition allows for composing services according to the current state of the context at the moment the service is actually required.
This approach is particularly useful in highly dynamic contexts. Once again, the approach focuses on composability of services.

*Agent-based Service Composition*

| Synopsis | |
|---|---|
| Goal: | Enable dynamic service composition to enable continuous provision of media services in wireless environments |
| Importance: | Offers a method to enable dynamic, on-the-fly composition of services |
| Kind of contribution: | Method |
| Applicable to | Wireless context, with media services |
| Phase where it applies | Lifecycle management of services |
| Key Ideas: | The service broker is made up of a set of five different types of agents; the composition is made up by the proper agent after the request of the user, according to the availability of the services; information about the context is used to verify the availability of a service at a certain time. |
| Foundations: | Mobile agents foundations, service-based architecture concepts |
| Validation: | Development of a service for the real-time multimedia stream playback to a wireless client |

Here we are presenting an approach described in [159]. This approach is based on the exploitation of agent-based concepts, and allows for dynamic, on-the-fly composition of services. The aim is to enable service-based multimedia applications for wireless clients.
The system implements a decentralized discovery approach through a network of five different types of agents. These agents cooperate with each other to enable dynamic service composition, dealing with service availability. Indeed, when the request for a service arises, a Planning Agent plans the composition of services required to accomplish the request. Then, the composition of component services is performed on the basis of the present availability of the different concrete services. If a selected Services is not available, another “equivalent” service is selected. Moreover, a different kind of *context* is associated with each type of service considered (Composite Service, Component Service and service instance). Such a context contains information about the service it is associated to. For instance, the context associated with each Component Service contains information about the number of running instances of that service, and their status.
To validate this approach, the authors have built a real-time multimedia stream playback to a wireless client. To provide a proof-of-concept, the idea is to build an implementation based on Jini.

### 2.7.7 Summary of the approaches and their relevance to the design of adaptable service-based applications

Within this section we have provided an overview of the approaches dealing with the run-time part of a service life cycle. Particularly, we have chosen to present approaches allowing for the achievement of adaptation at run-time. Let us now point out how these approaches can be usefully employed to support the development and management of service-based systems.
With reference to figure 2, these approaches can be exploited to perform the stages included in the left-side cycle. Some of them, the one presented in 2.6.1, deal with the deployment stage, but with reference to the *dynamic* deployment, rather than the one which comes after design.
In section 2.6.2 and 2.6.6 some approaches are presented allowing for the management of services during their execution time. These approaches can offer support especially for the “Operation, management and Quality Assurance” phase in figure 2. However, “quality assurance” is an issue taken into consideration also within other stages, such as in deployment (see approach about “QoS-aware deployment”). Indeed, QoS provision is a key goal of many adaptive approaches, especially those concerning dynamic binding (see PAWS), dynamic configuration and deployment.
Most of the described approaches mainly deal with the last three stages of the run-time cycle in figure 2, and many of them particularly focus on designing adaptation strategies and on the ways of enacting adaptation. For instance, considering approaches dealing with dynamic binding, PAWS concentrates on the way of enacting adaptivity, while WS-binder also takes into account policies and strategies to achieve adaptive binding.

Similar considerations can be made about the other phases we have described, such as *discovery*, *service management* and *configuration*. About the latter, for instance, the approach described in the "model transformation" approach particularly focus on the strategies, and how to actually apply them to achieve run-time adaptation.
Finally, some of the described approaches, instead of concentrating on the actual achievement of an adaptive behaviour, focus on general purpose considerations. For instance, the QoS approach to dynamic deployment described in 2.7.1.2 presents a generic mathematical framework to obtain configurations providing the highest QoS in grids.

# 3 State of the art on HCI knowledge for service-based applications engineering

Human-Computer Interaction (HCI) has been defined as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [194]. It is the study of the interaction between humans (both as individual users or groups of users collaborating) and computers, here taken in the broadest sense and hence including computerized devices and large scale computer systems as well as stand-alone computers. HCI aims to improve the interaction between humans and computers by making the latter both more usable and useful for the users. It proposes various theories, methodologies and techniques in the design and evaluation of computer systems (e.g. for interface design or information architecture) in order to achieve its aim, and also researches novel human-computer interaction paradigms.

As well as being accessed and used by other services or applications, 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 [195]). This interaction is supported in BPEL by BPEL4People, an extension to the language that defines human tasks through Web Service Human Task (WS-Human Task) and describes them as activities with WS-BPEL Extension for People [196]. This section explores whether HCI can further contribute to the integration of human actors in SBAs by providing richer, more contextual descriptions and models than those currently available. It further examines whether an explicit integration of current HCI knowledge into approaches to the engineering of SBAs would support their development.

## *3.1 Task Modeling*

Approaches to the engineering of SBAs (e.g. SOMA or SLDC) comprise phases for the analysis of applications functionalities and business processes in order to inform their implementation as services. The human element however is not specifically taken into account, and HCI task modeling techniques may contribute to the output of these phases by focusing on the tasks and the human users rather than on the system in place and the processes to be. They may as well support the definition of appropriate level of granularity, and functional service cohesion for SBAs by helping to clearly scope and define tasks for their later implementation as service operations.

### 3.1.1 Overview

A task model describe structured sets of activities [197] - often in interaction with a system influenced by its contextual environment - that the user has to perform to attain goals [198]. In cognitive psychology, task models are a means for formally describing human problem solving behavior [199]. Tasks models enable designers to represent and manipulate a formal abstraction of the activities that ought to be performed in order to reach user goals; the information necessary to their generation is usually obtained from documentation and observation. Typical characteristics include their "hierarchical structure, the task decomposition and sometimes the temporal relationships between elements" [200]. Their attributes and related information usually consist of general information (task's identifier and extended name, category and type, frequency of use, informal annotations, possible preconditions...), the interface or domain application objects (name and class) that have to be manipulated to perform it, and the estimated performance time [201].

Task models may have various applications at different stages of the software life cycle: they help understanding how people currently work [202] so they can function as a requirements elicitation device and indicate how activities should be performed in an application being designed. They allow for the software design to be described more formally, analysed in terms of usability, and be better communicated to people other than the analysts [201], [202], [203]. Wilson et al. [204] also propose such models to be used as a core for the task-centered design of user interfaces [205]: they can inform the design of user interfaces (more particularly the choice of appropriate dialogue elements for the individual atomic activities) and the analysis of the cognitive complexity of existing interfaces.

### 3.1.2 Approaches and Techniques

A number of task modeling techniques have particular angles or focus that would not be suitable to model tasks for their later implementation as SBAs' operations; those will not be described here.

**HTA (Hierarchical Task Analysis)**

| **Synopsis** | |
|---|---|
| **Goal**: | Describe structured sets of activities |
| **Importance** | Permits the representation of tasks leading to user goal(s) |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Analysis, Design |
| **Key Ideas** | Hierarchically decomposes tasks, represents constraints on them using plans |
| **Foundations** | Task decomposition |
| **Validation** | Commercial projects |

HTA represents the relationships between tasks and subtasks by a hierarchy of operations (what people must do within a system to attain a goal) and plans (the necessary conditions to undertake these operations). They can be represented in either tabular or diagrammatical form (see Fig. 1 for an example) and are one of the most commonly used modeling technique although it may be both time and effort consuming to obtain some of the necessary information or to evaluate conflicting comments [206]. The layout of the tasks and subtasks does not represent a temporal relationship - only the plans express constraint as to the order in which tasks and sub-tasks can be carried out - and so a HTA diagram is suitable to represent asynchronous processes. Any level of granularity may be represented and tasks that will not be decomposed further within the diagram are recognisable by the horizontal line drawn under them - it would be possible to model interoperability between several services cooperating for the execution of a process using this notation.

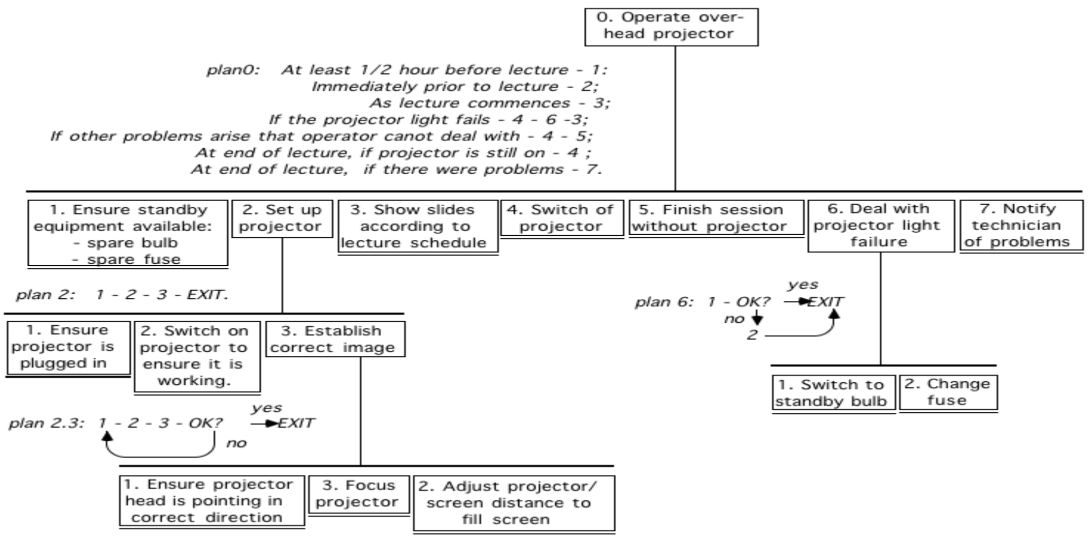

0. Operate over-head projector
plan0: At least 1/2 hour before lecture - 1:
Immediately prior to lecture - 2;
As lecture commences - 3;
If the projector light fails - 4 - 6 -3;
If other problems arise that operator canot deal with - 4 - 5;
At end of lecture, if projector is still on - 4 ;
At end of lecture, if there were problems - 7.
1. Ensure standby equipment available: - spare bulb - spare fuse
2. Set up projector
3. Show slides according to lecture schedule
4. Switch of projector
5. Finish session without projector
6. Deal with projector light failure
7. Notify technician of problems
plan 2: 1 - 2 - 3 - EXIT.
1. Ensure projector is plugged in
2. Switch on projector to ensure it is working.
3. Establish correct image
plan 2.3: 1 - 2 - 3 - OK? yes EXIT no
1. Ensure projector head is pointing in correct direction
3. Focus projector
2. Adjust projector/ screen distance to fill screen
plan 6: 1 - OK? yes EXIT no 2
1. Switch to standby bulb
2. Change fuse

**Fig.1: HTA for operating an overhead projector [207]**

**CTT (ConcurTaskTree)**

| **Synopsis** | |
|---|---|
| **Goal**: | Describe structured sets of activities |
| **Importance** | Permits the representation of tasks leading to user goal(s) |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Analysis, Design |
| **Key Ideas** | Hierarchically decomposes tasks, represents choice, information exchange and temporal relationships between them |
| **Foundations** | Task decomposition, LOTOS (Language of Temporal Ordering Specifications), |
| **Validation** | Case study |

CTT is a widely used notation as it contains a rich set of operators. It can be supported by a tool, CTTE (ConcurTaskTrees Environment) which facilitates the creation, visualization and sharing of task models. Proposed by Paterno ([208]), CTT can represent different task types (user, application, interaction or abstract) as nodes in a task tree; it shows the relationships (including choice, exchange of information and temporality) between the tasks at a same hierarchical level using operators (see Fig. 2 for example).
Sinnig et al [202] have implemented an extension to CTT that may be used for the modeling of tasks whose execution is distributed and relies on cooperation and interaction with other tasks (including human ones); additionally CTT has already been used with success to "obtain the operations that implement the requirements of a web application in a Service Oriented Architecture" [209].

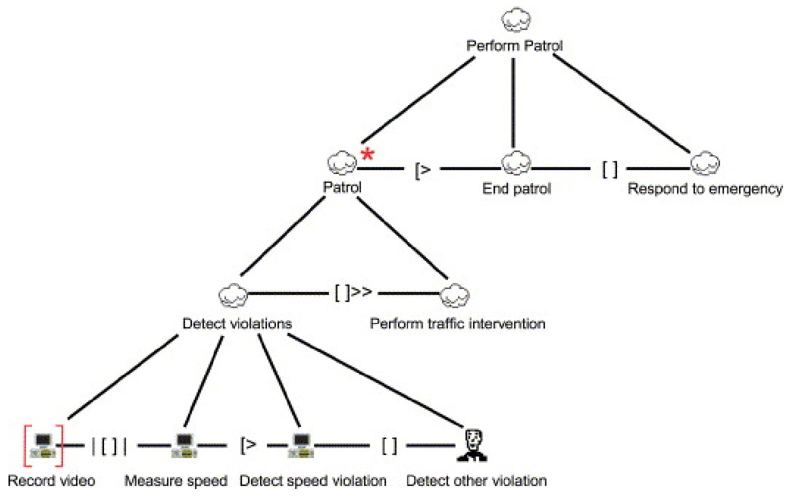


**Fig. 2: CTT for accessing student data (in CTTE)**

It may also be interesting to note that the repetitive modeling of the same tasks in case of service reuse in different SBAs could be addressed by the use of Task Patterns - context-specific, problem-centered task fragments which can be reused for building task models - as described in [210].

### 3.1.3 Discussion

Current service-oriented approaches rely primarily on business process models and notations such as BPEL to indicate the process-oriented context in which a service needs to be invoked; however process models normally lack other important information about the actor who is performing the process and his actions. Task models have semantically richer models and notations and may provide more context-specific information to inform SBAs' modeling: their specific semantics permit different possible uses of task models in service-centric systems; for example tasks and operations in HTA can map to specific service capabilities, thus enabling finer-grain service discovery and selection. Likewise, the distinction between abstract, interaction, application and user tasks in CTT can be mapped to different service types, thus informing more effective service composition, especially if finer-grain services are available.

## *3.2 User Modeling*

Services are supplied by providers for use by other entities; as specified in the OASIS definition "the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider". In order for systems and humans to successfully cooperate it is important to take into account some of the qualities and characteristics of people. For each human user, a service should behave as if it were designed with their specific needs, abilities and expectations taken into account in order to allow for a seamless and efficient interaction. It is a feat that is difficult to achieve considering there is no one typical user of a system, but many different ones who have changing requirements as their context of use and their level of experience evolve.
User modeling is an essential technique that can support the personalization of services for adaptation to a user by analyzing and representing the user's characteristics and attitudes.

### 3.2.1 Overview

According to Carmichael et al [211], the term user model encompasses both the notion of the idiosyncratic model developed by an individual user, and the idealized model which the designer considers an appropriate one for users to develop.

The definition we will employ here however will be that of [212], with user models considered as "models that systems have of users that reside inside a computational environment". Here the models are a system's representation of user characteristics, and focus on acquiring and representing the user's preferences, goals, and level of knowledge. In keeping with Stary [213], they contain all the properties of an individual user that are necessary to adapt a computer system to this user's needs, with the adaptation being oriented towards the experiences, the behavior during interaction, the goals, and the further needs of users.
The key characteristics for user models are usually categorized as personal characteristics, which are relatively stable over time, or system-dependent characteristics (such as goals, motivation and expectations) which are variable depending on the system [214]. The personal characteristics can be further decomposed into general characteristics (e.g. age, gender, personality) and the user's previously acquired knowledge and abilities (such as experience or cognitive abilities). Shifroni et al [215] add that the user model should also contain aspects of the immediate context of the user while Frias – Martinez [216] comment that among all human factors, gender differences, levels of expertise and cognitive styles have been recognized as especially relevant to users' interaction with the Web.
Rich in [217] identifies three dimensions for the models in which they can either be:

- canonical **(**model of a single user) or a collection of individual user models,
- explicit (i.e. user specified) or implicit (system-inferred),
- long-term (built from relatively long term facts e.g.) or short-term.

## 3.2.2 Approaches and Techniques

User model data can be implicit and obtained through observation (automatic user modeling) or explicit and directly collected by involving the user (Cooperative user modeling) [218]. Approaches to user modeling that may be applicable to the engineering of SBAs include Group modeling and Distributed User Modeling.

**Distributed User Modeling:**

| **Synopsis** | |
|---|---|
| **Goal** | Represent user's characteristics |
| **Importance** | User characteristics inform the design and adaptation of systems |
| **Kind of contribution** | Methodology |
| **Applicable to** | SBAs |
| **Phase where it applies** | Design, Run-time |
| **Key Ideas** | Use a client-server architecture to centrally maintain user model fragments |
| **Foundations** | Ubiquitous computing |
| **Validation** | Academic projects |

User Modeling Servers with a client-server architecture allow the user model information to be imported and maintained in a central repository from which it is concurrently accessible to applications. Rather than monolithic user models, distributed user model fragments [219] are used that can be maintained by several agents and are often quite shallow, inconsistent with one another and reflect not only characteristics of the users, but also some of the social relationships among them [211].
The wide range and increasing availability of software services means that rich user models combining information from varied sources could be developed using this approach to inform the engineering of SBAs

**Group modeling :**

| **Synopsis** | |
|---|---|
| **Goal** | Represent user's characteristics |
| **Importance** | User characteristics inform the design and adaptation of systems |
| **Kind of contribution** | Methodology |
| **Applicable to** | SBAs |
| **Phase where it applies** | Design, Run-time |
| **Key Ideas** | Use stereotypes to categorize users and infer additional characteristics |
| **Foundations** | Expert Systems |
| **Validation** | Academic projects |

User Modeling Shell (UMS) systems act as separate user modeling components that group users using stereotypes from which they infer their characteristics; they are domain-independent and hence respect SBAs' loose coupling and reusability.
User stereotypes here refer to a descriptive enumeration of a set of traits that often occur together [217]. The necessary information to build them can be gathered among others through the observation of the users interacting with the system or the analysis of their interaction history. A stereotype is described by:

- the user experience with respect to the task,
- System experience, related to the user's general technological skills,
- The task motivation - the attitude of the user towards the task.
- the user's experience and skill in using state-of-the-art HCI devices (all from [213]).

### 3.2.3 Discussion

User modeling approaches have not so far been considered in methods for developing service-centric systems. Their integration however could provide an important source of context information for the development of SBAs as well as for their discovery, composition and monitoring at run-time. The decoupling of the user models from the applications
The monitoring components of in SBAs could also potentially gather user preferences automatically and either maintain their user models or construct new ones as appropriate.
A possible research direction also emerges here as the exploration of user models in the presence of the service-centric systems that users work with, this in keeping with Chin ([220]) as he states that user models cannot and should not be separated from the software system that uses them.

## *3.3 User Interface: Automatic Generation and Portlets*

User interfaces are the most important point of contact with a system for a user; they are of a crucial importance to the efficiency, usefulness, and overall experience in interacting with the system. It is hence paramount for those services that require a user interface to get the design right.
The costs of providing high quality user interfaces for different platforms and different users accessing various combinations of services however would be high, both economically and in terms of the complexity of the operation; Automatic User Interface Generation is an area of research that yields promises to address these issues.

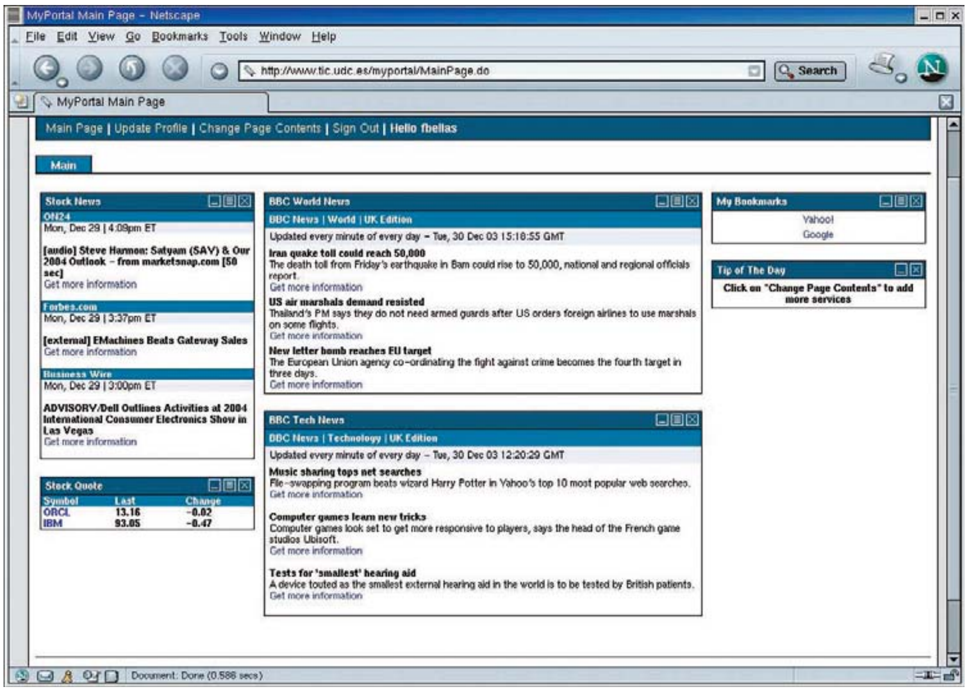
### 3.3.1 Overview

The user interface (UI) can be considered as the visible, physical representation of systems that allow users to interact with them and use the systems' functionalities. They are an abstracted description of the system's behavior, pared down to avoid overloading the user with irrelevant or unnecessarily abundant information [221].

Initiatives for generating graphical user interfaces (GUIs) for web services exist. WSUI (Web Service User Interface) is one such initiative that allows the creation of user interface components for web services. It requires however human operators to write additional code in order to permit the invocation of the WSUIs.
Automatic User Interface generation aims to allow designers to specify user interfaces at a very high level, and have the system provide the details of the implementation [222]. User interface generators often either define rules relevant to the appropriate design conventions for each application, or rely on human designers to add design conventions to interfaces after they have been automatically generated. Related academic projects include SUPPLE, SUPPLE++ and ARNAULD [223], and PUC [224].

Nichols et al [225] researched the viability of automatic interface generation. They demonstrated that an automatic generator applying good HCI practice in its generation had the potential to improve usability by automatically ensuring the new interfaces generated were consistent with user's previous experience. The interfaces were also found to be easier to learn for novice users and had the added benefit of being portable to other devices. The conclusion was that automatic interface generation is now viable, and especially desirable where users will benefit from individualized interfaces or where human designers are constrained by cost and other factors.
Gajos et al [226] illustrated a special case of automatic user interface generation enhancing the usability of a system. They identified motor-impaired users that had to rely on specialized assistive technologies (devices enabling or facilitating task performance by users with disabilities) as direct beneficiaries of this approach - assistive technologies, although useful, bring issues of their own such as cost, complexity, limited availability and a need for ongoing maintenance. Furthermore, it is estimated that only 60% of users who need assistive technology actually use it, and automatically generated ability-based interfaces could help circumvent the need for it by adapting the software to the capabilities of its user rather than the other way round as is the case at the moment.

Another related technology of interest is that of portlets.
A portal is a system for collecting, tailoring and displaying different kind of data onto a single screen for a user, and portlets are basic components of portals. A portlet represents an interactive web mini application and is deployed on portal server (see Fig. 3). It renders markup fragments that the portal can integrate into a page, and may represent the user interface of part or the whole of an application. Web services exposing part of an application's business logic are often defined to be integrated into portals. However all portals wishing to integrate them must re-implement the user interface; a suggested solution is to provide the service as a remote portlet returning HTML markup fragments rather than plain data [227]. Interactive web services (unlike data-oriented ones) are presentation-oriented and can carry fully rendered markup to be included within a portal page; they usually offer one or more portlets and WSRP – Web Services for Remote portlets [228] – permits the use of remote portlets as if they were locally accessible.


MyPortal Main Page - Netscape
File Edit View Go Bookmarks Tools Window Help
Search
MyPortal Main Page
Main Page | Update Profile | Change Page Contents | Sign Out | Hello fbellas
Main

http://www.tic.udc.es/myportal/MainPage.do

**Stock News**

ON24
Mon, Dec 29 | 4:09pm ET

**[audio] Steve Harmon: Satyam (SAV) & Our 2004 Outlook – from marketsnap.com [50 sec]**
Get more information

Forbes.com
Mon, Dec 29 | 3:37pm ET

**[external] EMachines Beats Gateway Sales**
Get more information

Business Wire
Mon, Dec 29 | 3:00pm ET

**ADVISORY/Dell Outlines Activities at 2004 International Consumer Electronics Show in Las Vegas**
Get more information

**Stock Quote**

| Symbol | Last | Change |
|---|---|---|
| ORCL | 13.16 | -0.02 |
| IBM | 93.05 | -0.47 |

**BBC World News**

BBC News | World | UK Edition
Updated every minute of every day – Tue, 30 Dec 03 15:16:55 GMT

**Iran quake toll could reach 50,000**
The death toll from Friday's earthquake in Bam could rise to 50,000, national and regional officials report.
Get more information

**US air marshals demand resisted**
Thailand's PM says they do not need armed guards after US orders foreign airlines to use marshals on some flights.
Get more information

**New letter bomb reaches EU target**
The European Union agency co-ordinating the fight against crime becomes the fourth target in three days.
Get more information

**BBC Tech News**

BBC News | Technology | UK Edition
Updated every minute of every day – Tue, 30 Dec 03 12:20:29 GMT

**Music sharing tops net searches**
File-swapping program beats wizard Harry Potter in Yahoo's top 10 most popular web searches.
Get more information

**Computer games learn new tricks**
Computer games look set to get more responsive to players, says the head of the French game studios Ubisoft.
Get more information

**Tests for 'smallest' hearing aid**
A device touted as the smallest external hearing aid in the world is to be tested by British patients.
Get more information

**My Bookmarks**

Yahoo!
Google

**Tip of The Day**

**Click on "Change Page Contents" to add more services**

Document: Done (0.588 secs)

**Fig.3: Sample portlet based portal [227]**

## 3.3.2 Approaches and Techniques

There exist general HCI principles, guidelines and industry standards (e.g. [229], [230]) that provide support and guidance in the development of user interfaces at different levels of abstraction. Their usefulness and necessity is widely accepted in the engineering of software systems in general.
In the specific case of automated generation of user interfaces for web services however, the use of Smart Templates is of particular interest..

**Smart templates:**

| **Synopsis** | |
|---|---|
| **Goal**: | Permit the automatic generation of user interfaces |
| **Importance** | Decreases UI development overheads without compromising principles of good HCI design |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Design, Run-time |
| **Key Ideas** | Predefine parameterized templates for later application to automatic interface generators |
| **Foundations** | HCI interaction design, Automatic interface generation |
| **Validation** | Academic project |

Nichols et al [231] propose to improve the quality of automatically generated user interfaces and make it comparable to that of hand-designed ones by using and applying HCI design conventions.
Smart templates, the technique they describe, uses parameterized templates (defined in advance by template writers who specify the set of states and commands to be included) to allow automatic interface generators to create interfaces that are usable and consistent with other user interfaces on the same device.

### 3.3.3 Discussion

Despite significant advances, automatically generating user interfaces is still a challenging task that suffers from different problems. Technical difficulties mean that there are limitations on the kind of interfaces that can be produced as well as a degree of unpredictability. The generated interfaces can also sometimes be considered not as good as those created with conventional programming techniques [222].
The key challenge for service-centric systems is to deliver in an automated manner user interfaces that are at least as usable and useful as those developed by experienced engineers and analysts. New semantics for discovering and composing portlets are needed to deliver usable interfaces, both at design-time and run-time; one possible approach would be to seek to emulate experienced human designer as is done with the Smart Templates technique.

## *3.4 User Error Analysis*

User error is a common feature of interactions between users and computer-based systems. While better interaction design can avoid some errors occurring too frequently, all user errors cannot be eradicated. Available services and current service-centric systems do not address user (or other forms of non-software) error effectively, which suggests 3 major research challenges for the development, but also the running of service-centric systems:

- How to design and publish software services that can handle and respond to user errors?
- How to compose and orchestrate software services that can handle and respond to user errors?
- How can the discovery, composition and orchestration reduce the range and frequencies of errors that occur when users work with service-centric systems?

### 3.4.1 Overview

HCI's outlook on human error have evolved with time. They were initially considered as the causes of accidents that originated from people's fallacious assessments and decisions; however modern error theory now regards human errors as the symptoms of trouble deeper inside a system. It seeks to understand and address how operators' erroneous actions connect to their tools, tasks and environment (Dekk, as cited in [232]).
Many definitions for errors have been proposed (Senders and Moray [233], Woods et al [232], Isaac et al cited in [234]); the definition adopted here is that of Reason [235], of error as a generic term to encompass all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and when these failures cannot be attributed to the intervention of some chance agency.

Norman (as cited in [232]) differentiates between two types of errors: slips and mistakes, to which Reason [197] proposes to add a third one: lapses.
Norman defines slips as action errors or errors of execution triggered by a person’s organized knowledge, memories, and experiences that resulted from the combination of environmental triggers and schemas. Lapses are primarily memory errors and are less observable than slips, occurring when a person becomes distracted in the performance of a task.
Mistakes are explained as errors of thoughts in which a person’s cognitive activities lead to actions or decisions that are contrary to what was intended. Rasmussen (cited in [194]) further expands the cognitive aspect of errors and proposes 3 types of operator performance and associated errors:

- skill-based performance errors which, like Norman’s slips, are largely errors of execution
- rule-based performance errors resulting from a human inability to correctly assess a situation – either through lacking the necessary information or applying the wrong rule to it
- Knowledge-based performance errors, the product of shortcomings in operator knowledge or limitations in his ability to apply existing knowledge to new situations.

All of these types of user error are possible during interaction with service-centric systems, and future methods, technologies and services need to have capabilities to handle such errors.
It is however important to note that the occurrence of errors is not necessarily non-constructive. Strauch [232] for instance comments that some situations such as learning new skills call for errors: people sometimes require feedback to realize they have committed errors. This view is shared by Senders & Moray [233] and Van Dyck et al [236] who agree that zero-error policies underestimate or ignore the potentially positive value of errors. Opportunities hence exist for services that support error-based learning during interaction. The lack of formal training or help resources in using some services, especially those embedded in transactions, is another argument to allow for "learning by doing".

Reliable error detection is paramount to the handling of errors in systems since in order to recover from an error it is necessary to become aware of its occurrence in the first place. This awareness can be triggered by many events such as a mismatch between the expected and the observed outcome. Bagnara et al (cited in [234] have proposed a taxonomy of error identification which, crossed with taxonomies of services types and their possible user interactions, may guide error handling in service-based applications. The extent to which service-centric systems can support human user's error detection is undetermined and the scope of this challenge will be discussed later in this section.

### 3.4.2 Approaches and Techniques

It may be of interest to note that when considered from an HCI point of view, human error analysis is influenced by two main perspectives: psychological – "the error is in the head" – or systemic, where the error is "in the world" and hence linked to tasks, tools and the work environment (Dekker, [237]); Wiegmann et al's viewpoints on error (in [238]) further refine the psychological and systemic perspectives.

Human error analysis and detection in HCI is often informed by using methods stemming from fields such as Safety Critical Systems Analysis such as Probabilistic Risk Assessment (PRA) (e.g. event and fault trees to identify potential areas of risk) and Human Reliability Analysis (HRA) (to quantify the likelihood of human error occurring in given situations). More specific HCI approaches can however be used for human error analysis and handling.

**THEA: Technique for Human Error Assessment Early in Design**

| **Synopsis** | |
|---|---|
| **Goal**: | Anticipate human interaction failures before a system becomes operational |
| **Importance** | Interaction design contributes to reducing the likelihood of human errors occurring |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Planning, Analysis, Design |
| **Key Ideas** | Potential interface problems can be discovered and handled early in the design process |
| **Foundations** | Human Reliability Assessment, Cognitive Walkthrough |
| **Validation** | Commercial project |

THEA takes into account the context of use in its human error assessment; it is meant for use early in system's development cycle to drive their interaction design while anticipating interactions failure.
It captures usage context using scenarios, structures and interprets the information they yield using decomposition techniques, identifies areas of design where cognitive failures may occur and assesses their possible impact on the system (Pocock et al [239], illustrated in fig. 4).

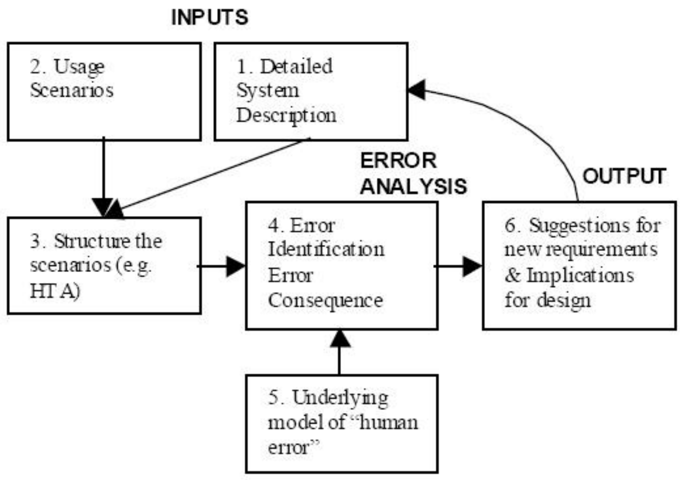

INPUTS
2. Usage Scenarios
1. Detailed System Description
ERROR ANALYSIS
OUTPUT
3. Structure the scenarios (e.g. HTA)
4. Error Identification Error Consequence
6. Suggestions for new requirements & Implications for design
5. Underlying model of "human error"

**Fig. 4: The THEA process [239]**

**Guidelines for handling human errors:**
Rizzo et al [239] notably have produced the following set of guidelines for managing errors in user-centered design:

- Make the action more perceptible – improve the match between actions and their outcomes
- Use Multi-sensory feedback
- Display message at high level but with specific content – improve the understandability and appropriateness of error messages for users
- Provide an activity log
- Allow comparisons between related outputs
- Make results available to user evaluation and allow user control on the display format
- Provide results explanations

### 3.4.3 Discussion

The diversity of users that may potentially use services (and their accordingly various levels of IT literacy) means that all efforts should be made to ensure that the design and operation of services is such as to minimize the occurrence of errors, to improve error resilience and error concealment where necessary, and to ensure that, if they do occur, they will be handled properly. Models and notations for reasoning about services need to handle error semantics, and service capabilities to handle errors must be documented. Integrating information about error likelihood and prevention in software services might be a challenge that an appropriate framework in the engineering of services could contribute to addressing.

## *3.5 Personalization*

Personalizing services would enable tem to better serve the users by anticipating needs, making the interaction efficient and satisfying for both parties, and building a relationship that encourages the services reuse.

### 3.5.1 Overview

Personalization involves a process of gathering user-information during interaction with the user, which is then used to deliver appropriate, tailored content and services to the user. The aim is to improve the user's experience of a service [241].
The personalization of service-centric systems can occur at two basic levels: that of the individual service and the service orchestration. At the level of the individual service, different services can be invoked for different users, and different parameter values can be input to the same service. At the level of service orchestration, user access data can be used to inform the orchestration of services and

personalize it to users' preferences. For the personalization to occur, the services must be context – sensitive (with the user context here being an aggregation of the user's location, previous activities and preferences, [242]) and adapting accordingly. Much research has been conducted, looking into delivering adaptive web content (e.g. the AVANTI project [243]), as well as the issues and challenges associated with it [244], [245], [246].

### 3.5.2 Approaches and Techniques

Two main approaches may be of interest in the personalization of service-centric systems delivered over the web: one based on adaptive hypermedia techniques, the other on filtering and recommendations techniques.

**Adaptive hypermedia**

| **Synopsis** | |
|---|---|
| **Goal**: | Adapt the system to the user |
| **Importance** | Adaptation supports context-appropriate delivery of services |
| **Kind of contribution** | Methodology |
| **Applicable to** | SBAs |
| **Phase where it applies** | Composition, Run-time |
| **Key Ideas** | User models inform the adaption of systems to the users themselves |
| **Foundations** | |
| **Validation** | Commercial project |

This approach aims for the system to be adapted to the user in such a way that the information presentation is understandable to the user, fits his needs and constraints (e.g. platform used, environment), and the user is steered towards relevant information. User Models are extensively used to guide the adaptation, which may be content-based or access-based (a user's past interaction with the service may be used here) although a combination of both is possible.

**Filtering and Recommendation - based approach**

| **Synopsis** | |
|---|---|
| **Goal**: | Adapt the system to the user |
| **Importance** | Adaptation supports context-appropriate delivery of services |
| **Kind of contribution** | Methodology |
| **Applicable to** | SBAs |
| **Phase where it applies** | Composition, Run-time |
| **Key Ideas** | User models inform adaptation of systems to the user themselves and to other users |
| **Foundations** | Information filtering |
| **Validation** | Commercial project |

This approach is based on the interests, preferences, likes, dislikes, and goals a user has; the information is mostly stored in a modeling server [245]. Resources are recommended according to features extracted from a resource content, or to ratings of a user or learner of similar profile [246]. The recommendations are usually based on collaborative filtering - [247] reported that this mechanism was able to provide content recommendations to individual users within a multi-user environment with a high level of user satisfaction, and without the need for user authentication or profile creation. Inductive logic programming [248] has also been used with good results, as has benefit theory (during a search, the search engine asks for a few questions which help characterize the user with respect to the search item). The prediction based on benefit was reported to be distinctively more accurate than collaborative filtering, but the challenging part of this solution would be in coming up with sets of questions that are accurate but small enough to help characterize users without making them feel overwhelmed.

### 3.5.3 Discussion

How to provide intelligent services that leave the user in control? The challenges are to discover how much intelligence has to provided, for what purpose, in which situations, how to sense changes of context and to adapt to other cultures and languages while addressing related privacy concerns, laws and industry self- regulations that may be in effect [249].
The success of service-centric systems will be affected by their ability to resolve fundamental challenges that existing technologies don't sufficiently address; personalization is one of these challenges. If web services can be successfully personalized to accommodate every user context and overcome cultural and language barriers, they can become a truly global phenomenon [250].

Research and practice on user personalization in HCI provides important evidence with which to design service-based adaptation. Approaches taking into account users' past interactions and their user profile can inform service-based adaptation if monitoring identifies which users access which services.

## *3.6 End User Customization*

As with personalization, the fact that the same services could potentially be used by millions of different consumers makes it crucial that services are customizable to accommodate the different requirements of those users.

### 3.6.1 Overview

Customization refers to the design and creation of content that meets a customer's specific needs; customized content is not delivered dynamically (e.g., personalization), but is created by an author for a specific requirement and is static until changed [251].
The customization of web-based applications is often considered a designer skill rather than an end-user need. However, there is an ongoing shift to end-user-centered technology, and even users with poor or no skill in web-based languages may feel the need to customize web applications according to their preferences. Accommodating the user's needs by building technology that would help users help themselves would ensure the users can stay in control and are able to personalize the application in a way that satisfies, mitigates, or enhances their various emotional states and needs [252].
Although web authoring environments have an increasing number of features, the challenge of providing end-users with the ability to easily customize entire Web applications still remains unsolved [253].

### 3.6.2 Approaches and Techniques

Existing approaches to customization are either mainly component-based, or reuse end-user customization.

**Component-based approach**

| **Synopsis** | |
|---|---|
| **Goal**: | Allow users to select and move content units on web pages |
| **Importance** | Enables users to customize the content and layout of web pages |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |

| | |
|---|---|
| **Phase where it applies** | Run-time |
| **Key Ideas** | Record user's customization settings to automatically apply them to the same or similar web pages on subsequent visits |
| **Foundations** | - |
| **Validation** | Commercial project |

This approach to end-user customization has been previously used on the desktop environment to enable the personalization of Web pages. Web pages such as Google Personalized Homepage (see fig. 5) allow users to choose content units to be displayed in the page, and to select the areas of the page in which these units should appear. Support for such technique is limited to a few sites however and the customization options offered are restricted. Users also need to be familiar with the different interfaces provided by each site [254].

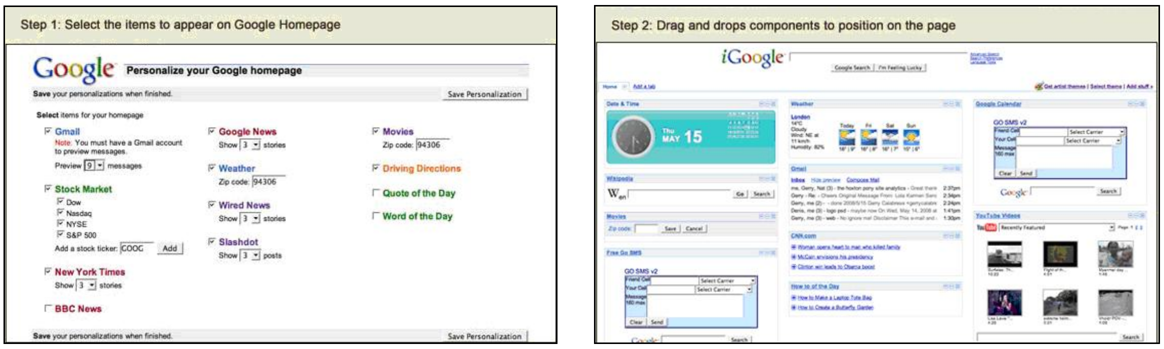
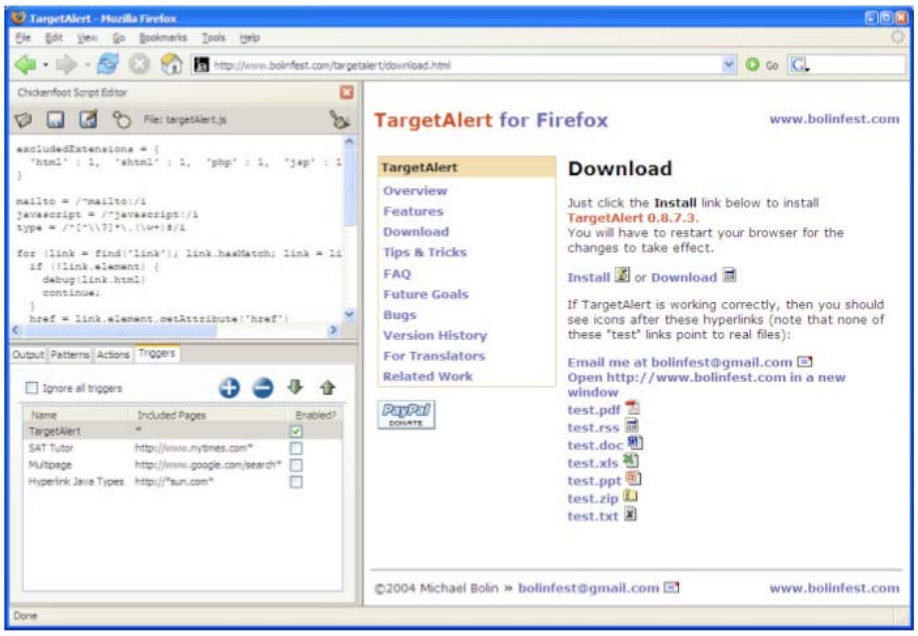


*Figure 5: Google Customization Feature*

The FLEXIBEANS component model and the FREEVOLVE tailoring platform are based on this approach and embody concepts and software developed by Stiemerling [255][256], Won [257], and Hinken [258].
Variants of this approach are taken by a number of systems, including Chickenfoot [259], Greasemonkey [260] and Platypus [261]. Chickenfoot and Greasemonkey provide programming interfaces for the desktop through which users specify actions to be performed on page content, such as removing elements. The development environment runs inside a browser and allows the user to customize the page currently being viewed (see fig. 6). Platypus adds a What You See Is What You Get (WYSIWYG) interface to Greasemonkey, allowing desktop users to perform customization graphically. However, none of these systems employ algorithms to ensure customizations are long-lived, nor are able to automatically reuse customizations on similar pages.


TargetAlert - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.bolinfest.com/targetalert/download.html
Go
Chickenfoot Script Editor
File: targetAlert.js
Output Patterns Actions Triggers
Done

```
excludedExtensions = {
  'html' : 1,  'shtml' : 1,  'php' : 1,  'jsp' : 1
}

mailto = /^mailto:/i
javascript = /^javascript:/i
type = /^[^\\?]*\.(\w+)$/i

for (link = find('link'); link.hasMatch; link = li
  if (!link.element) {
    debug(link.html)
    continue;
  }
  href = link.element.getAttribute('href')
```

☐ Ignore all triggers

| Name | Included Pages | Enabled? |
|---|---|---|
| TargetAlert | * | ✓ |
| SAT Tutor | http://www.nytimes.com* | ☐ |
| Multpage | http://www.google.com/search* | ☐ |
| Hyperlink Java Types | http://*sun.com* | ☐ |

TargetAlert for Firefox www.bolinfest.com

TargetAlert
Overview
Features
Download
Tips & Tricks
FAQ
Future Goals
Bugs
Version History
For Translators
Related Work

PayPal DONATE

Download

Just click the **Install** link below to install TargetAlert 0.8.7.3.
You will have to restart your browser for the changes to take effect.

Install or Download

If TargetAlert is working correctly, then you should see icons after these hyperlinks (note that none of these "test" links point to real files):

Email me at bolinfest@gmail.com
Open http://www.bolinfest.com in a new window
test.pdf
test.rss
test.doc
test.xls
test.ppt
test.zip
test.txt

©2004 Michael Bolin » bolinfest@gmail.com www.bolinfest.com

Fig. 6: Chickenfoot development environment [259].

It is worth mentioning that no matter how well a tool is designed for end users, they will use it in unexpected ways and so logs and the tools associated with them are often relied upon to "tell the story" by showing usage patterns, change, and unusual events [262].
It has been suggested that unpredictable autonomous interface adaptations can easily reduce a system's usability as well as require a large amount of user support. However research shows that the negative aspects of adaptive user interfaces can be overcome without actually improving the user's mental model of the adaptive system [263].

**REUC (Reusable End-User Customization)**

| **Synopsis** | |
|---|---|
| **Goal**: | Record and store users' customization and preferences |
| **Importance** | Permits the reuse of customization settings |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Run-time |
| **Key Ideas** | Record user's customization settings to automatically apply them to the same or similar web pages on subsequent visits |
| **Foundations** | - |
| **Validation** | Academic project |

The Reusable End User Customization technique allows end users to adapt the layout of Web pages by removing, resizing and moving page elements. It records the user's customizations and automatically reapplies them on subsequent visits to the same page or to other, similar pages, on the same Web site. PageTailor, a prototype running on Windows PDAs developed by Bila et al [264], uses REUC. Use testing experiments conducted with PageTailor demonstrated that the technique was viable, with users able to successfully customize complex web pages from large real world sites such as Amazon, MSN, eBay and Flickr and, for a considerable majority, to reapply their customizations for at least a month and in some cases up to a year [264].

### 3.6.3 Discussion

Empirical as well as design-oriented research has identified a number of challenging areas in developing and implementing customizable systems [265], [266], [267], [268], [269], [270]; one of the main issues is support for customizing services to meet users' preferences and technical skills [257]. Explicit solutions that have the potential to deliver success include dialogs that support customization (shown to work in [271]) and end user service composition approach that reduces the composition complexity and difficulty from the end user perspective [272]. The latter enables users to select and customize the services, with the composition occurring only at the presentation layer - an approach that respects the loose coupling desirable in service-centric systems.

It is necessary to represent and reason about the complexity of both services and service-centric systems and to take into account the explicit user competencies needed to achieve the customization. A possible research direction on the topic may be the exploration and articulation of possible global qualities of service-centric systems that are needed to support successful end-user customization from an HCI perspective. This could provide a baseline for exploring and supporting end-user customization of service-centric systems further.

## *3.7 Accessibility*

Generally speaking, the accessibility of services is paramount to their use - either as stand-alone services or as part of service-based application - and their value to businesses and individual users. Between 11 and 15% of the EEC population are said to have some form of disability [273], and in Europe alone it is estimated that by 2050, the number of people over 60 will have doubled to 40% of the total population or 60% of the working age population [274]. Furthermore the diversification of the available interaction platforms (portable, wearable equipment, kiosks…) and the potential accessibility issues each of them brings means that the range of the population which may gradually be confronted with accessibility problems extends beyond disabled and elderly users to potentially include all people.

With SBAs potentially serving millions of people around the world (e.g. Amazon, eBay or CRM services) and within different corporate settings (and hence in accordingly diverse organizational cultures), economic, legal and social motivations as well as ethical and moral ones [275] call for the engineering of services to strive for their accessibility to the wider number.

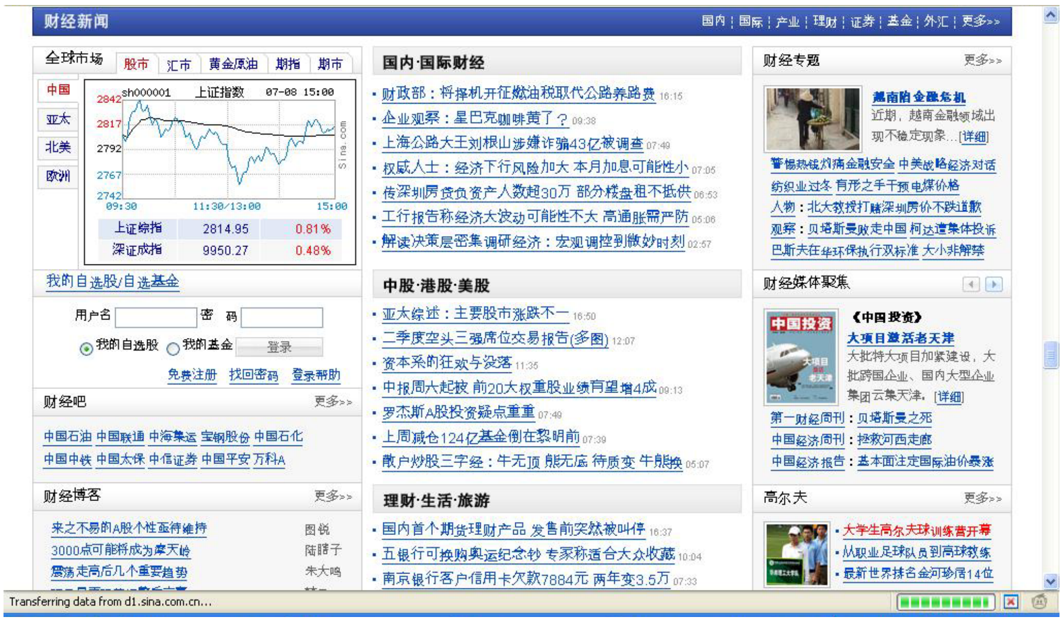
### 3.7.1 Overview

Accessibility has been defined as referring to the ability for individuals with diverse capacities, preferences and context of use, to use a product, a service or an environment – but not necessarily with the same degree of usability for all ([276], [277]).

It is one of the major challenges for designers and developers to guarantee universal inclusion, and it requires among others compliance with accessibility guidelines which, although commonly accepted, are not always obvious to identify and correctly apply - especially since guidelines are updated or newly proposed as the research on accessibility progresses [278].

Accessibility is often linked to disability issues and the adjustments needed for disabled users to use software; an extensive body of research exists in HCI that, although initially intended for the engineering of software systems in general, is applicable for services. Cultural differences however as well as age, context of use and experience all greatly affect the use of a system, and it is felt this aspect of accessibility is particularly relevant to service-centric systems. Culture here relates to the attitudes and behaviors of a group that are relatively stable over time, and also to the group united by these commonalities [279]. The only related, service-specific standards and provisions found were the Web Service Internationalization Usage Scenarios that are aimed at web services designers wishing to add international capabilities in their services [280].
Although the use of English is dominant on the web, particularly for business purposes, language for example is an aspect of culture that ought to be provided for in order to maximize the possibility of

services' seamless composition and interoperability with other services using different character sets (e.g. Fig. 7). Issues such as the rendering of data where restrictions in the usage of characters present a significant inconvenience for certain language groups ([281]) also ought to be addressed.


财经新闻

国内 ¦ 国际 ¦ 产业 ¦ 理财 ¦ 证券 ¦ 基金 ¦ 外汇 ¦ 更多>>

全球市场 股市 汇市 黄金原油 期指 期市

中国 亚太 北美 欧洲

sh000001 上证指数 07-08 15:00

| | | |
|---|---|---|
| 上证综指 | 2814.95 | 0.81% |
| 深证成指 | 9950.27 | 0.48% |

我的自选股/自选基金

用户名 密 码

[✓ 我的自选股] 我的基金 登录

免费注册 找回密码 登录帮助

财经吧 更多>>

中国石油 中国联通 中海集运 宝钢股份 中国石化
中国中铁 中国太保 中信证券 中国平安 万科A

财经博客 更多>>

| | |
|---|---|
| 来之不易的A股个性亟待维持 | 图锐 |
| 3000点可能将成为摩天岭 | 陆睛子 |
| 震荡走高后几个重要趋势 | 朱大鸣 |

国内·国际财经

- 财政部：将择机开征燃油税取代公路养路费 16:15
- 企业观察：星巴克咖啡黄了？ 09:38
- 上海公路大王刘根山涉嫌诈骗43亿被调查 07:49
- 权威人士：经济下行风险加大 本月加息可能性小 07:05
- 传深圳房贷负资产人数超30万 部分楼盘租不抵供 06:53
- 工行报告称经济大波动可能性不大 高通胀需严防 05:06
- 解读决策层密集调研经济：宏观调控到微妙时刻 02:57

中股·港股·美股

- 亚太综述：主要股市涨跌不一 16:50
- 二季度空头三强席位交易报告(多图) 12:07
- 资本系的狂欢与没落 11:35
- 中报周六起披 前20大权重股业绩有望增4成 09:13
- 罗杰斯A股投资疑点重重 07:49
- 上周减仓124亿基金倒在黎明前 07:39
- 散户炒股三字经：牛无顶 熊无底 待质变 牛熊换 05:07

理财·生活·旅游

- 国内首个期货理财产品 发售前突然被叫停 16:37
- 五银行可换购奥运纪念钞 专家称适合大众收藏 10:04
- 南京银行客户信用卡欠款7884元 两年变3.5万 07:33

财经专题 更多>>

越南陷金融危机
近期，越南金融领域出现不稳定现象…[详细]

警惕热钱灼痛金融安全 中美战略经济对话
纺织业过冬 有形之手干预电煤价格
人物：北大教授打赌深圳房价不跌道歉
观察：贝塔斯曼败走中国 柯达遭集体投诉
巴斯夫在华环保执行双标准 大小非解禁

财经媒体聚集

《中国投资》
大项目激活老天津
大批特大项目加紧建设，大批跨国企业、国内大型企业集团云集天津。[详细]

第一财经周刊：贝塔斯曼之死
中国经济周刊：拯救河西走廊
中国经济报告：基本面注定国际油价暴涨

高尔夫 更多>>

- 大学生高尔夫球训练营开幕
- 从职业足球队员到高球教练
- 最新世界排名金河珍居14位

Transferring data from d1.sina.com.cn...

**Fig. 7: sina.com portal**

### 3.7.2 Approaches and Techniques

There are no structured methods for accessible design in HCI; developers rely on principles and guidelines to steer system development.

Accessibility guidelines of particular interest for web services and SBAs include:

- Web content accessibility guidelines, with recommendations for making web content accessible to people with disabilities [281]
- Accessibility guidelines for specific technologies: XML (see [282])
- Accessibility guidelines for web applications with dynamic content and advanced user interface controls (see [283])

Additionally, a mention of HCI accessibility evaluation techniques may be of interest for services – possibly in the testing phase of SBAs engineering.
Several methods are available, most of them derived from usability investigation methods, among which (in [284], [285], [286], [287], [288], [289]):

- Comprehensive Accessibility Evaluation, Conformance Tests and Heuristic Evaluation. They all rely on practitioners evaluating systems to assess conformance to the appropriate standards and guidelines.
- Screening Techniques, which are mainly used to evaluate accessibility for disability and are conducted using assistive technologies.
- Design Walkthroughs, Heuristic Walkthroughs and User Testing that all rely on evaluating the system or early prototypes of it using pre-defined scenarios of use.

Automated evaluation is also possible through the use of built-in Hard-Coded Tests [290], Web Page Annotation [291], [292], [293], Statistics-Based Algorithms [294] or Extensible Rulesets [295], [296], [297], [298].

It was demonstrated that the guidelines embedded in some of these automated tools guarantee improved usability and accessibility of the sites more than experienced web designers who rely on their own knowledge [299].

### 3.7.3 Discussion

A growing number of countries have introduced legislations which addresses the need for websites and other forms of communication to be accessible to people with disabilities; they also enforce the more general requirement that people with disabilities not be discriminated against (e.g. [300], [301], the Disability Discrimination Act in the UK). For compliance with these laws an in order to support their discovery and reusability, services as well as their descriptions and their specifications should be accessible to the widest number of people.
There is scope for research to be extended to discover and articulate global accessibility requirements for service-centric systems – possibly supported by a framework – and to examine related issues that are not currently addressed by existing accessibility guidelines.

## *3.8 Social Networks*

With the facilitation and increase of user-generated and distributed content online, the way we communicate and share information has been revolutionized. Social networks have very much been part as well as a product of this revolution, securing millions of users that have tightly incorporated these sites into their online activities. It is believed that HCI research can contribute to understanding and harnessing some of the potential of this relatively new phenomenon for the design of services and SBAs.

### 3.8.1 Overview

Social network sites (SNS) are defined as web-based services that allow individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections and those made by others within the system - with the nature and nomenclature of these connections varying from site to site [302].
Most SNS are either profile-centric or "passion-centric" (connecting people based on shared interests), and they have greatly changed the way we communicate and share information. They increasingly rely on the use of standard Internet-enabled Web services to support their collaborative activities [303] and allow users to interact, including podcasting and streaming video services, social bookmarking, folksonomies (collaborative tagging), specialized search features and social search engines [304]. Some SNS also allow users to add modules themselves – more services offering additional functionalities.

The interest of social networks for the engineering of SBAs is two-fold. First, services are more and more extensively used in social network sites, a trend that shows no sign of waning. There is an opportunity to learn from the specificities of this mode of use and of distribution in the online world, and to research the particular interaction with humans that occurs there. This knowledge could be applied to the engineering of services - especially if shifts from the social aspects of SNS to more business-oriented ones come to materialize – and their distribution in the online world using such networks as a medium. Additionally, an understanding of the workings of such networks could potentially support the discoverability of services by exploring the possibilities for recommendations of web services and SBAs within the network.

### 3.8.2 Approaches and Techniques

**Social Network Analysis**

| **Synopsis** |
|---|

| | |
|---|---|
| **Goal**: | Analyze social networks |
| **Importance** | Permits the description and visualization of social networks |
| **Kind of contribution** | Technique |
| **Applicable to** | SBAs |
| **Phase where it applies** | Run-time |
| **Key Ideas** | Describing networks' nodes and relationships non-discursively permit the identification and quantification of their properties |
| **Foundations** | Social Psychology, Graph Theory, Statistics, Matrix Algebra |
| **Validation** | Academic project |

SNA is a technique for describing and visualizing networks through the analysis of relational data and the identification of their structural properties. People or groups in a network are represented as nodes, and links between them show their relationships with other nodes. SNA can either be ego-centered (focusing on the individual, with a random sample of the network population involved) or can focus on the whole of the network. The data is inputted in matrices recording the presence or absence of relationships between the subjects, their type and even their weight where appropriate.

**Social Simulation**

| **Synopsis** | |
|---|---|
| **Goal**: | Simulate social phenomenon |
| **Importance** | Permits the representation of a social phenomenon in order to understand and explore it |
| **Kind of contribution** | Method |
| **Applicable to** | SBAs |
| **Phase where it applies** | Run-time |
| **Key Ideas** | Monitoring and testing a social simulation yields information that is applicable to the target social phenomenon |
| **Foundations** | Social network analysis, Agent based modeling |
| **Validation** | Academic project |

Simulations are used to mimic networks and test hypothesizes on them (usually by creating and running behavioral rules). Several online tools can be used to create simulations, including NetLogo [305] which was used to create the graphs shown in Fig. 8.

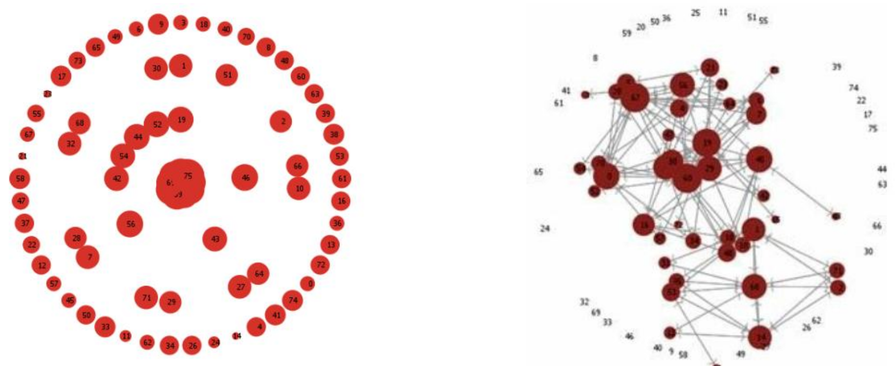


**Fig. 8: Social simulation graphs generated with NetLogo [306]**

### 3.8.3 Discussion

A mass growth in web services is predicted which, according to estimates from market intelligence agency IDC, will reach $9.1 billion by the end of 2008 [307]. One interesting characteristic of this trend is that these web services will not just be non-interactive information processing services, but will support multi-media content (pictures, video and audio) developed for a wide-range of applications including leisure. In this sense social computing is not only an application for the human-computer aspects of service-centric computing, but also a driver for the development and evolution of such systems and the research underpinning them. This correlates to significant upward trends in service outsourcing (Gartner) and national strategic investments for longer-term development of web services economy [308].

Rating and voting services that have proven effective in building trust in on-line communities - such as feedback services support for online auction systems (e.g. eBay) and recommender services that elicit recommended services from friends and colleagues - give examples indicating that similar types of infrastructure services could be developed and/or employed more widely in service-centric systems. More generally however, SNS which are based on the use of a relatively small number of application services can potentially provide valuable lessons learned for the wider dissemination and uptake of service-centric systems.

## *3.9 Security*

An increase in the availability and accessibility of computerized tools, coupled with the expansion of their applications to include entertainment and convenience services (internet shopping, banking), has led to an increase and a diversification of information technology users at various levels of computer literacy, including the elderly and children.

Concerns for online security have been growing following increases in the report of business security breaches and an heightened awareness of the existence and implications of security violations such as phishing and identity theft. With individuals apprehending the possibility that inadvertent information disclosure online could create a threat offline [309], it is important to address the security of services to reassure users and enable them to take advantage of the available technology while keeping themselves and their data safe.

### 3.9.1 Overview

Security has been defined as the protection of both a computer system and its data against unauthorized access or alteration [310], [311].

Security in relation to services will be considered from the angle of the security violation categories reported by Saltzer and Schroeder [312]: unauthorized information release, information modification, and denial of use. "Unauthorized" here refers to the events occurring contrary to the desire of the person who controls the information or the constraints supposedly enforced by the system even though the intruder may be an otherwise legitimate user of the computer.

There has been an increase in the number of web services necessitating authentication in order to access personalized versions [313]; several standards and security solutions have been proposed and implemented in parallel to allow for secure identification of the user to the service provider.

The human aspect of security however has not always been considered or properly taken into account despite the abundance of literature suggesting that human vulnerabilities are a major source of security breaches. Indeed human behavior is often the source of vulnerabilities in the system for access or alteration, either to the system or its data, by attackers. Schneier [314] is often quoted in this context as he asserted that '... security is only as good as its weakest link, and people are the weakest link in the chain.'

One difficulty for users is the need to manage several digital identities (sets of claims about the user and his characteristics that the service provider recognizes [313]) to access various personalized services. Remembering account details and their associated passwords often cause the use of very similar or identical sign-in details from account to account, the choice of weak passwords that are very

vulnerable to compromise (either through dictionary attacks or social engineering), and the writing of passwords in plain sight as a memory aide.
Security standards such as SAML (Security Assertion Markup Language) or the integration of OpenID with web services [315] aim to address this issue by providing a Single Sign On (SSO) for web services. This approach however is not immune to password acquisition attacks (through HTTP redirects or social engineering for instance) exploiting users' lack of knowledge or computing skills.

Another OASIS standard, Web Services Security (WSS or WS-Security) provides mechanisms to guarantee message integrity and confidentiality by using digital signatures and encryption. It is to be used within other non service-specific security models such as Kerberos or PKI [313]. Usability problems and possible ensuing security breaches have however been reported among users of PKI-enabled software. They are due to the complexity of PKI, the transfer of responsibility onto the user, flawed users assumptions about the technology and a lack of feedback [316]. Similar issues related to the complexity of encryption technology and its general misunderstanding by users were also researched by Whitten et al [317] and could apply to WS-Trust as well, a standard that necessitates the creation and validation of digital tokens for security.
Another technology, described by Vassilev [313], is a hardware solution using a network smart card as a secure solution for personal brokerage of web service access. This introduces issues of its own, such as the proposition that the layout of the virtual PIN pad used for authentication be randomized – hence addressing the threat of screen-monitoring software, but also increasing the likelihood of user error while typing in their PIN.

The facts reported above argue for the consideration of the human element and the inclusion of principles of HCI research in securing computer systems. Indeed, considering that security mechanisms are only effective when used correctly, it is paramount to identify and address the causes of risky user behavior to design effective security systems.
Sasse et al [318] have established that unintentional users violations of security happen when users fail to comply with the behavior required by a secure system. They argue that there are two reasons for this occurrence: the users are unable to behave as required (due to unreasonable demands on their physical or cognitive abilities, impatience or diverted attention, their individual perceptions and attitudes or a failure to recognize their responsibilities) or the users do not want to behave in the way required, not because it is too difficult, but because it is awkward to them. In either case, the reasons are clearly linked to poor usability of the system leading either to user error or to user resistance to the recommended mode of operation.
Considering that previous research has suggested basic security needs (privacy, anonymity, authentication, integrity of data … [314], [319]), it is proposed that for systems to be actually and not only theoretically secure these tenets be examined in the light of appropriate HCI considerations to ensure user compliance.

### 3.9.2 Approaches and Techniques

Much of the HCI research on security has focused on the usability of system's security tools and interfaces for the user. Approaches to security have relied on guidelines and ad hoc methods drawn from experience and applicable to software engineering in general. One proposed method for the engineering of secure systems, AEGIS, borrows from HCI contextual design and usability evaluation and so will be described in this section.

**Principles for Secure Interaction Design:**
Yee has proposed guidelines aimed at both minimizing the risk of undesired events (by controlling authorization) and promoting good communication between the user and the system so that the user's intents can be accurately and naturally conveyed to the system. The guidelines, as presented in [320], are:

- Path of least resistance: match the most comfortable way to do tasks with the least granting of authority.

- Active authorization: grant authority to others in accordance with user actions indicating consent.
- Revocability: offer the user ways to reduce others' authority to access the user's resources.
- Visibility: maintain accurate awareness of others' authority as relevant to user decisions.
- Self-awareness: maintain accurate awareness of the user's own authority to access resources.
- Trusted path: protect the user's channels to agents that manipulate authority on the user's behalf.
- Expressiveness: enable the user to express safe security policies in terms that fit the user's task.
- Relevant boundaries: draw distinctions among objects and actions along boundaries relevant to the task.
- Identifiability: present objects and actions using distinguishable, truthful appearances.
- Foresight: indicate clearly the consequences of decisions that the user is expected to make.

**Model of Informed Consent:**
Considering that privacy - the ability of an individual (or organization) to decide whether, when, and to whom personal (or organizational) information is released [312] – also affords the user security (or a sense of it), Friedman et al [321] proposed a model of informed consent and recommended that six components be considered:

- Disclosure: providing accurate information about the benefits and harms that might reasonably be expected from the action under consideration
- Comprehension: the user has an accurate interpretation of what is being disclosed
- Voluntariness: the action is not coerced or overly manipulated –the user could resist participation
- Competence: the user possesses the mental, emotional and physical capabilities needed to give informed consent
- Agreement: the user has a reasonably clear (but not necessarily explicit) opportunity that is visible, readily accessible and ongoing to accept or decline to participate.
- Minimal distraction: meeting all above criteria without "unduly diverting the individual from the task at hand."

Saltzer [312] also defined general design principles, some of which may be of interest in the context of software services.

**AEGIS (Appropriate and Effective Guidance in Information Security)**

| **Synopsis** | |
|---|---|
| **Goal**: | Support developers in creating secure systems |
| **Importance** | Integrate and address security and usability concerns into the software development process. |
| **Kind of contribution** | Method |
| **Applicable to** | SBAs |
| **Phase where it applies** | Planning, Design, Testing |
| **Key Ideas** | Integrating security engineering into system's engineering by incorporating principles of HCI contextual design and Risk analysis results in more secure systems |
| **Foundations** | Risk analysis, Spiral model of software development, Contextual design |
| **Validation** | Case study |

AEGIS is a method for creating secure systems that is "based on security requirements identification, asset modeling, risk analysis and context of use" [322]. It is inspired from the Boehm model of software development (see fig. 1) and goes through the phases of Asset Identification, Security Requirements Elicitation, Risk analysis and Security Design.

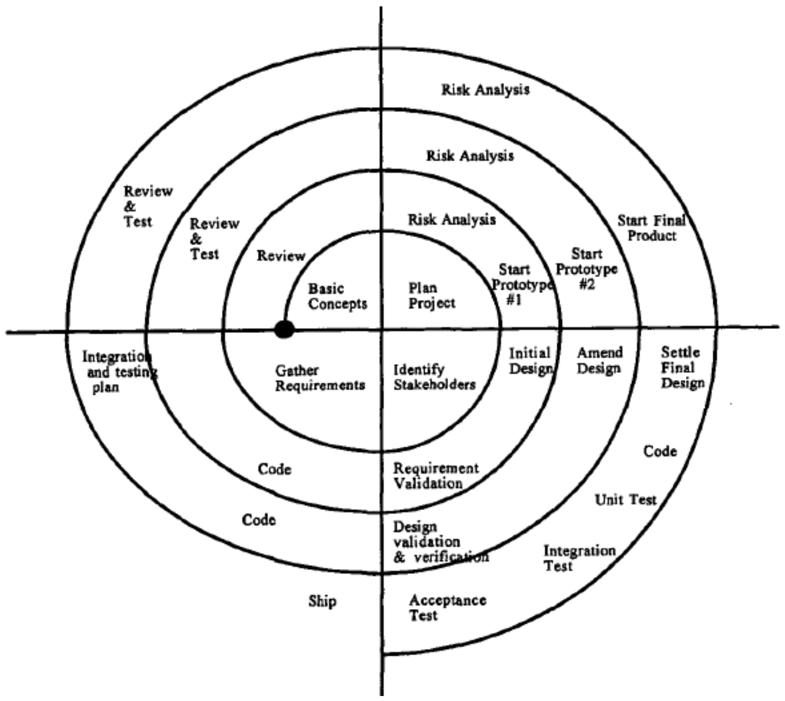


**Fig. 9: AEGIS Spiral model of software development [322]**

### 3.9.3 Discussion

Usability and security are often seen as competing goals. Security experts are inclined to reject proposals for improving usability as more usable mechanisms may introduce a vulnerability in the system or increase risk of attack. For example, changing passwords less frequently means that a compromised password may be used longer [318], [321]. Security systems also typically attempt to introduce barriers to action (such as passwords or other authentication mechanisms) while HCI designers attempt to remove such barriers because they often seem to confuse or frustrate end users [309].
Finally, there is a lack of clear, objective methodology for measuring and comparing the size and intensity of the Internet-mediated threat, and hence a corresponding lack of methodology for measuring the effectiveness of the countermeasures against it [318], [321].

Providing convenient yet secure Web services is one of the challenges of today's IT industry [313]. The use of services is exploding across government and businesses, and suggested possible uses have included the formation of federations of organizations forming a common information infrastructure within which critical data could be circulated without delay [323]. The security of data is hence paramount to the continued uptake of services across organizations, and addressing one of the main threats to that security – the human operator – is one of the requisites to the realization of this vision.

## *3.10 Conclusion*

For many service-centric systems, interaction occurs exclusively with other applications and considerations involving human users are often relegated to second plan when it comes to engineering services and SBAs. Humans however are important stakeholders and active participants at various levels of SBAs' lifecycle, as initiatives such as BPEL4People have recognized.
This section has presented areas of HCI knowledge where it is felt that its integration alongside established SBAs' engineering methodologies will provide rich and informative data on the goals and the context of use of SBAs, this from the perspective of human users which –implicitly or explicitly-

are the ultimate beneficiaries of such systems. In this process, challenges and possible directions for future research on the topic have been uncovered which may provide the bases for an improved design of SBAs interaction with human users.

# 4 References

[1] "Reference Model for Service Oriented Architectures 1.0", OASIS Standards, 12 October 2006

[2] "A5.D9.3 – SeCSE Conceptual Model v4", *Report* delivered on 30/04/08, University of Sannio and CEFRIEL, available at http://secse.eng.it.

[3] M. Papazoglou "Web Services: principles and technology". Ed. Addison-Wesley, July 2007

[4] "W3C Working Group Note 11", http://www.w3.org/TR/ws-gloss/#defs , February 2004

[5] Liguo Yu, "Applying Software Wrapping on Performance Monitoring of Web Services", *INFOCOMP Journal of Compute Science* (ISSN: 1807-4545), vol. 6, no. 3, 2007, pp. 1-6

[6] T. Chaari, F. Laforest, A. Celentano, "Service-oriented context-aware application design", In *First International Workshop on Managing Context Information in Mobile and Pervasive Environments*, Ayia Napa, CYPRUS, 2005

[7] M. Papazoglou, "Service-oriented computing: concepts, characteristics and directions", *4th International Conference on Web Information Systems Engineering* (WISE 2003), 10-12 December 2003, Rome, Pages: 3 - 12

[8] G.Denaro, M. Pezzé, D. Tosi, "SHIWS: a Self-Healing Integrator for Web Services", *Software Engineering Companion*, 2007. ICSE 2007 Companion. 29th International Conference on, Publication Date: 20-26 May 2007

[9] K. Pohl, "Requirements Engineering - Grundlagen, Prinzipien, Techniken", Ed. dpunkt, Heidelberg, 2007.

[10] T. Moran, P. Dourish, (Eds.). (2001). "Introduction to context-aware computing" [*Special issue*]. *Human--Computer Interaction*, 16, 2001

[11] T. Winograd (2001), "Architectures for Context". In *Human-Computer Interaction*, 16 (2) pp. 401-419

[12] G. D. Abowd , A. K. Dey, et al., "Towards a Better Understanding of Context and Context-Awareness Moderators", in LNCS, January 1999.

[13] M. Colombo, E. Di Nitto, M. Di Penta, D. Distante, Maurilio Zuccalà: "Speaking a Common Language: A Conceptual Model for Describing Service-Oriented Systems". In proceedings of the 3rd International Conference on Service Oriented Computing (ICSOC 2005), December 2005, Amsterdam, The Netherlands, pp. 48-60.

[14] Thomas Erl, "SOA Principles of Service Design", Prentice Hall, 2007.

[15] Michael N. Huhns and Munindar P. Singh, "Service-Oriented Computing: Key Concepts and Principles", IEEE Internet Computing, vol. 9, n. 1, pages 75-81, 2005.

[16] "2007 BPM & Workflow Handbook", WFMC, May 2007

[17] "Web Services Business Process Execution Language Version 2.0", OASIS Standards, May 2007

[18] "Wikipedia - Service (economics)", http://en.wikipedia.org/wiki/Service_%28economics%29, read: 11/07/2008

[19] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," in *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)* Limerick, Ireland, 2000, pp. 35-46.

[20] J. Castro, M. Kolp, and J. Mylopoulos, "Towards Requirements-Driven Information Systems Engineering: The Tropos Project," *Information Systems*, vol. 27, pp. 365-389, 2002.

[21] J. Cybulski and P. Sarkar, "Requirements Engineering for Web-Based Information Systems," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Berlin, Heidelberg: Springer, 2005, pp. 327-349.

[22] A. v. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *5th IEEE International Symposium on Requirements Engineering (RE 2001)* Toronto, Canada, 2001, pp. 249-262.

[23] G. M. Weinberg, *Structured Analysis*. New York: Yourdon Press, 1978.

[24] C. Gane and T. Sarson, *Structured Systems Analysis – Tools & Techniques: Improved System Technologies*. New York, 1977.

[25] T. DeMarco, *Structured Analysis and System Specification*. New York: Yourdon Press, 1978.

[26] D. T. Ross and K. E. Schoman, "Structured Analysis for Requirements Definition," *IEEE Transactions on Software Engineering,* vol. 3, pp. 6-15, 1977.

[27] S. M. McMenamin and J. F. Palmer, *Essential Systems Analysis*. New York: Yourdon Press, 1984.

[28] E. Yourdon, *Modern Structured Analysis*. Englewood Cliffs: Prentice Hall, 1989.

[29] C. Ashworth and M. Goodland, *SSADM – A Practical Approach*. Columbus: McGraw Hill, 1990.

[30] J. Neighbors, "Software Construction Using Components," in *Department of Information and Computer Science*. vol. Ph.D. Irvine: University of California, 1981.

[31] K. Pohl, G. Böckle, and F. v. d. Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*. Berlin, Heidelberg, New York: Springer, 2005.

[32] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Pittsburgh, Carnegie-Mellon University, Technical Report 1990.

[33] P. P.-S. Chen, "The Entity Relationship Model -- Toward a Unified View of Data," *ACM Transactions on Database Systems,* vol. 1, pp. 9-36, 1976.

[34] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*. Upper Saddle River: Prentice Hall, 1991.

[35] I. Jacobson, M. Christerson, P. Jonsson, and G. Oevergaard, *Object Oriented Software Engineering – A Use Case Driven Approach*: Addison Wesley, 1992.

[36] OMG, "UML 2.0 Superstructure Specification," Object Management Group 2003.

[37] A. H. M. t. Hofstede and T. F. Verhoef, "On the Feasibility of Situational Method Engineering," *Information Systems,* vol. 22, pp. 401-422, 1997.

[38] J. Evermann and Y. Wand, "Toward Formalizing Domain Modeling Semantics in Language Syntax," *IEEE Transactions on Software Engineering,* vol. 31, pp. 21--37, 2005.

[39] A. v. Lamsweerde, A. Dardenne, B. Delcourt, and F. Dubisy, "The KAOS Project: Knowledge Acquisition in Automated Specification of Software," in *Proceedings of AAAI Spring Symposium Series* Stanford University: American Association for Artificial Intelligence, 1991, pp. 69-82.

[40] E. Yu, "An Organisational Modelling Framework for Multiperspective Information System Design," in *Requirements Engineering 1993 – Selected Papers*: Department of Computer Science, University of Toronto, Toronto, 1993, pp. 66 -86.

[41] A. Dardenne, A. v. Lamsweerde, and S. Fickas, "Goal-Directed Requirements Acquisition," *Science of Computer Programming,* vol. 20, pp. 3-50, 1993.

[42] H. A. Simon, *The Science of the Artificial*. Cambridge: MIT Press, 1981.

[43] E. Yu, "Towards Modeling and Reasoning Support for Early-phase Requirements Engineering," in *Proceedings of the 3rd International Symposium on Requirements Engineering (RE'97)* Los Alamitos: IEEE Computer Society Press, 1997.

[44] U. o. Toronto, "Goal-oriented Requirements Language (GRL): GRL Ontology (www.cs.toronto.edu/km/GRL)." vol. 2006, 2005.

[45] A. v. Lamsweerde, E. Letier, and R. Darimont, "Managing Conflicts in Goal-Driven Requirements Engineering," *IEEE Transactions on Software Engineering,* vol. 24, pp. 908--926, 1998.

[46] A. I. Antón, "Goal-Based Requirements Analysis," in *2nd International Conference on Requirements Engineering (ICRE 1996)* Colorado Springs, CO, USA, 1996, pp. 136-144.

[47] K. Yue, "What Does It Mean to Say that a Specification is Complete?," *4th International Workshop on Software Specification and Design (IWSSD 1987),* pp. 42-49, 1987.

[48] K. M. Benner, M. S. Feather, W. L. Johnson, and L. A. Zorman, "Utilizing Scenar-ios in the Software Development Process," in *IFIP WG 8.1. Working Conference on Information System Development Process* North-Holland, Amsterdam, 1993, pp. 117-134.

[49] J. M. Carroll, "The Scenario Perspective on System Development," in *Scenario-Based Design: Envisioning Work and Technology in System Development*, J. M. Caroll, Ed. New York: Wiley, 1995, pp. 1-17.

[50] A. Cockburn, *Writing Effective Use Cases*. Boston: Addison-Wesley, 2001.

[51] C. Potts, K. Takahashi, and A. I. Antón, "Inquiry-based Requirements Analysis," *IEEE Software,* vol. 11, pp. 21-32, 1994.

[52] C. Rolland, C. B. Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, and P. Heymans, "A Proposal for a Scenario Classification Framework," *Requirements Engineering,* vol. 3, pp. 23-47, 1998.

[53] S. Uchitel, J. Kramer, and J. Magee, "Negative Scenarios for Implied Scenario Elicitation," in *10th ACM SIGSOFT Symposium on Foundations of Software Engineering (SIGSOFT'02/FSE-10)* New York, 2002, pp. 109-118.

[54] C. Rolland, C. Souveyet, and C. B. Achour, "Guiding Goal Modelling Using Scenarios," *IEEE Transactions on Software Engineering,* vol. 24, pp. 1055-1071, 1998.

[55] A. I. Antón and C. Potts, "The Use of Goals to Surface Requirements for Evolving System," in *20th International Conference on Software Engineering (ICSE'98)*, 1998, pp. 157-166.

[56] P. Haumer, K. Pohl, and K. Weidenhaupt, "Requirements Elicitation and Validation with Real World Scenes," *IEEE Transactions on Software Engineering,* vol. 24, pp. 1036-1054, 1998.

[57] A. Sutcliffe, N. Maiden, S. Minocha, and M. Darrel, "Supporting Scenario-based Requirements Engineering," *IEEE Transactions on Software Engineering,* vol. 24, pp. 1072-1088, 1998.

[58] P. Kruchten, *The Rational Unified Process – An Introduction*: Addison-Wesley, 2003.

[59] D. Kulak and E. Guiney, *Use Cases – Requirements in Context*: Addison-Wesley, 2000.

[60] K. Bittner and I. Spence, *Use Case Modeling*. Boston: Addison-Wesley, 2003.

[61] E. Bourdeau, P. Lugagne, and P. Roques, "Hierarchical Context Diagrams with UML – An Experience Report on Satellite Ground System Analysis," in *First International Workshop on The Unified Modeling Language: Beyond the Notation (UML 1998)* Mulhouse, France, 1998, pp. 227-239.

[62] F. Pettersson, M. Ivarsson, and P. Öhman, "Automotive Use Case Standard for Embedded Systems," *ACM SIGSOFT Software Engineering Notes,* vol. 30, pp. 1-6, 2005.

[63] E. Nasr, J. McDermid, and G. Bernat, "Eliciting and Specifying Requirements with Use Cases for Embedded Systems," in *7th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002)* San Diego, CA, USA, 2002, pp. 350-357.

[64] D. D. Zhang, "Use Case Modeling for Real-Time Applications," in *4th International Workshop on Object-Oriented Real-Time Dependable Systems* Santa Barbara, CA, USA, 1999, pp. 56-64.

[65] M. Jackson, "The World and the Machine," in *17th International Conference on Software Engineering (ICSE 1995)* Seattle, Washington, USA, 1995, pp. 283-292.

[66] M. Jackson, *Software Requirements & Specifications: A Lexicon Of Practice, Principlees and Prejudices*. New York: Addison-Wesley, 1995.

[67] W. Swartout and R. Balzer, "On the Inevitable Interwining of Specification and Implementation," *Communications of the ACM,* vol. 25, pp. 438-440.

[68] P. Ward and S. Mellor, *Structured Development of Real-Time Systems – Introduction and Tools* vol. 1. Upper Saddle River: Prentice Hall, 1985.

[69] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti, "Relating Software Requirements and Architectures using Problem Frames," in *10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002)* Essen, Germany, 2002, pp. 137-144.

[70] G. Mullery, "CORE - A Method for Controlled Requirements Specification," in *4th International Conference on Software Engineering (ICSE 1979)* Munich, Germany, 1979.

[71] J. C. S. P. Leite and P. A. Freeman, "Requirements Validation through Viewpoints Resolution," *IEEE Transaction on Software Engineering,* vol. 17, pp. 1253-1269, 1991.

[72] S. Easterbrook and B. Nuseibeh, "Using ViewPoints for Inconsistency Management," *Software Engineering Journal* vol. 11, pp. 31-43, 1996.

[73] G. Kotonya and I. Sommerville, "Requirements Engineering with Viewpoints," *Software Engineering Journal,* vol. 11, pp. 5-18, 1996.

[74] B. Nuseibeh, J. Kramer, and A. Finkelstein, "A Framework for Expressing the Relationships between Multiple Views in Requirements Spcifications," *IEEE Transactions on Software Engineering,* vol. 20, pp. 760-773, 1994.

[75] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos – Representing Knowledge about Information Systems," *ACM Transactions on Information Systems,* vol. 8, pp. 325-362, 1990.

[76] M. Jarke and K. Pohl, "Establishing Visions in Context – Towards a Model of Requirements Processes," in *14th International Conference on Information Systems (ICSE 1993)* Orlando, USA, 1993.

[77] K. Pohl, *Requirements Engineering – Grundlagen, Prinzipien, Techniken*. Heidelberg: Dpunkt, 2007.

[78] S. V. Jones, N. A. M. Maiden, K. Zachos, and X. Zhu, "How Serivce-Centric Systems Change the Requirements Process," in *Proceedings of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2005)* Porto, Portugal, 2005, pp. 105-119.

[79] C. Batini and M. Lenzerini, "A Methodology for Data Schema Integration in the Entity Relationship Model," *IEEE Transactions on Software Engineering,* vol. 10, pp. 650-664, 1984.

[80] G. A. Miller, "WordNet - Princeton University Cognitive Science Laboratory, http://wordnet.princeton.edu/," 2006.

[81] K. Zachos, N. A. M. Maiden, X. Zhu, and S. Jones, "Discovering Web Services to Specify More Complete System Requirements," in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)* Trondheim, Norway, 2007, pp. 142-157.

[82] K. Zachos, X. Zhu, N. Maiden, and S. Jones, "Seamlessly Integrating Service Discovery Into {UML} Requirements Processes," in *Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering (SOSE 2006)* Shanghai, China, 2006, pp. 60-66.

[83] N. Maiden, "Servicing Your Requirements," *IEEE Software,* vol. 23, pp. 14-16, 2006.

[84] K. Zachos and N. Maiden, "Web Services to Improve Requirements Specifications: Does It Help?," in *Proceedings of the 14th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2008)* Montpellier, France, 2008, pp. 168-182.

[85] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," *Autonomous Agents and Multi-Agent Systems,* vol. 8, pp. 203-236, 2004.

[86] D. Lau and J. Mylopoulos, "Designing Web Services with Tropos," in *International Conference on Web Services (ICWS 2004)* San Diego, CA, USA, 2004, pp. 306-313.

[87] M. Aiello and P. Giorgini, "Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities," *UPGRADE: The European Journal for the Informatics Professional,* vol. 5, pp. 20-26, 2004.

[88] S. C. Misra, S. Misra, I. Woungang, and P. Mahanti, "Using Tropos to Model Quality of Service for Designing Distributed Systems," in *International Conference on Advanced Communication Technology (ICACT 2006)* Phoenix Park, Gangwon-Do, Republic of Korea, 2006, pp. 541-546.

[89] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal Reasoning Techniques for Goal Models," in *Journal on Data Semantics* Berlin, Heidelberg: Springer, 2003, pp. 1-20.

[90] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos, "From Stakeholder Needs to Service Requirements," in *2nd International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 2006)* Minneapolis, Minnesota, USA, 2006, pp. 8-17.

[91] M. Pistore, M. Roveri, and P. Busetta, "Requirements-Driven Verification of Web Services," *Electronic Notes in Theoretical Computer Science,* vol. 105, pp. 95--108, 2004.

[92] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso, "Specifying and Analyzing Early Requirements in Tropos," *Requirements Engineering,* vol. 9, pp. 132--150, 2004.

[93] C. Salinesi and C. Rolland, "Fitting Business Models to System Functionality Exploring the Fitness Relationship," in *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)* Klagenfurt, Austria, 2003, pp. 647-664.

[94] O. K. Ferstl, E. J. Sinz, M. Amberg, U. Hagemann, and C. Malischewski, "Tool-Based Business Process Modeling using the SOM Approach," Universität Bamberg, Bamberg 1994.

[95] C. Rolland, N. Prakash, and A. Benjamen, "A Multi-Model View of Process Modelling," *Requirements Engineering,* vol. 4, pp. 169-187, 1999.

[96] J. Ralyté, C. Rolland, and V. Plihon, "Method Enhancement with Scenario Based Techniques," in *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE 1999)*, Heidelberg, Germany, 1999, pp. 103-118.

[97] J. Ralyté and C. Rolland, "An Approach for Method Reengineering," in *Proceedings of the 20th International Conference on Conceptual Modeling (ER 2001)* Yokohama, Japan, 2001, pp. 471-484.

[98] J. Ralyté, C. Rolland, and R. Deneckère, "Towards a Meta-Tool for Change-Centric Method Engineering: A Typology of Generic Operators," in *Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004)* Riga, Latvia, 2004, pp. 202-218.

[99] C. Rolland and N. Prakash, "Bridging the Gap Between Organisational Needs and ERP Functionality," *Requirements Engineering,* vol. 5, pp. 180-193, 2000.

[100] C. Rolland and R.-S. Kaabi, "An Intentional Perspective to Service Modeling and Discovery," in *Proceedings of the 1st IEEE International Workshop on Requirements Engineering For Services (REFS 2007)* Beijing, China, 2007, pp. 455-460.

[101] R. S. Kaabi, C. Souveyet, and C. Rolland, "Eliciting Service Composition in a Goal Driven Manner," in *Proceedings of the 2nd International Conference on Service-Oriented Computing (ICSOC 2004)* New York, NY, USA, 2004, pp. 308-315.

[102] C. Rolland, C. Salinesi, and A. Etien, "Eliciting Gaps in Requirements Change," *Requirements Engineering,* vol. 15, pp. 1-15, 2004.

[103] P. v. Eck and R. Wieringa, "Requirements Engineering for ServiceOriented Computing: A Position Paper," in *Proceedings of the 1st International E-Services Workshop (ICEC 2003), September 30, 2003*, Pittsburgh, PA, USA, 2003, pp. 23-28.

[104] T. Davenport, *Process Innovation: Reengineering Work Through Information Technology*: Ernst & Young, 1993.

[105] J. J. M. Trienekens, J. J. Bouman, and M. v. d. Zwan, "Specification of Service Level Agreements: Problems, Principles and Practices," *Software Quality Journal,* vol. 12, pp. 43-57, 2004.

[106] S. Lichtenstein and A. H. L. Nguyen, "Issues in IT Service-Oriented Requirements Engineering," *Australasian Journal of Information Systems,* vol. 13, pp. 176-191, 2005.

[107] S. Modafferi, E. Mussi, B. Pernici, "SH-BPEL - A Self-Healing plug-in for WS-BPEL Engines", in *Proceedings of MW4SOC Workshop,* 2006.

[108] R. Al-Ali, A. Hafid, O. F. Rana and D. W. Walker, "QoS Adaptation in Service-Oriented Grids", in *Proceedings of Middleware Workshops*, 2003.

[109] M. Reichert and P. Dadam, "ADEPTflex – Supporting Dynamic Changes of Workflows Without Loosing Control", JIIS, vol. 10:2, 1998, pp. 93 – 129.

[110] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, B. Srivastava, "Adaptation in Web Service Composition and Execution", in *Proceedings of International Conference on Web Services (ICWS)*, 2006.

[111] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam and Q. Z. Sheng, "Quality Driven Web Services Composition", in *Proceedings of International Conference on World Wide Web (WWW),* 2003.

[112] V. Agarwal, G. Chafle, K. Dasgupta, N. Karnik, A. Kumar, S. Mittal, and B. Srivastava, "Synthy: A System for End to End Composition of Web Services", *J. Web Semantics*,Vol.3:4, 2005.

[113] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, and Z. Wu, "The METEOR-S Approach for Configuring and Executing Dynamic Web Processes", Technical Report 6-24-05, LSDIS, University of Georgia, USA.

[114] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, and J. Lee, "On Accommodating Inter Service Dependencies in Web Process Flow Composition", in *AAAI Spr. Symp.*, pages 37–43, 2004.

[115] Y. Wu and P. Doshi, "Regret-Based Decentralized Adaptation of Web Processes with Coordination Constraint", in *Proceedings of Service Computing Conference (SCC)*, 2007.

[116] G. Spanoudakis, A. Zisman, A. Kozlenkov, "A Service Discovery Framework for Service Centric Systems", in *Proceedings of Service Computing Conference (SCC)*, 2005.

[117] A. Lazovik, M. Aiello, and M. P. Papazoglou, "Associating assertions with business processes and monitoring their execution", in *Proceedings of International Conference on Service-Oriented Computing (ICSOC)*, 2004.

[118] A. Erradi, P. Maheshwari and V. Tosic, "Policy-Driven Middleware for Self-adaptation of Web Services Compositions", in *Proceedings of Middleware*, 2006.

[119] L. Baresi, C. Ghezzi, and S. Guinea, "Towards Self-healing Service Compositions", in *Proceedings of PRISE*, 2004.

[120] L. Baresi, S. Guinea and L. Pasquale, "Self-Healing BPEL Processes with Dynamo and the JBoss Rule Engine", in *Proceedings of International workshop on Engineering of software services for pervasive environments (satellite of the 6th ESEC/FSE)*, 2007, pp. 11-20.

[121] W. Kongdenfha, R. Saint-Paul, B. Benatallah, and F. Casati, "An Aspect-Oriented Adaptation Framework for Dynamic Component Evolution", in *Proceedings of International Conference on Service-Oriented Computing (ICSOC),* 2006.

[122] E. Rukzio, S. Siorpaes, O. Falke, H. Hussmann, "Policy Based Adaptive Services for Mobile Commerce", in *Proceedings of IEEE International Workshop on Mobile Commerce and Services (WMCS)*, 2005.

[123] P. Clements and L. M. Northrop, "Software Product Lines: Practices and Patterns", Addison-Wesley, 2001.

[124] S. H. Chang and S. D. Kim, "A variability modeling method for adaptable services in service-oriented computing," in *SPLC '07: Proceedings of the 11th International Software Product Line Conference*, 2007, p. 261-268.

[125] Johanneke Siljee, Ivor Bosloper, Jos Nijhuis, and Dieter Hammer. "DySOA: Making Service Systems Self-Adaptive", in *Proceedings of International Conference on Service-Oriented Computing (ICSOC),* 2005.

[126] A. Hallerbach, T. Bauer and M. Reichert, "Managing Process Variants in the Process Lifecycle", in *10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)*, 2008.

[127] F.Casati, S.Ceri, B.Pernici, and G.Pozzi, "Workflow Evolution", in *Proceedings of International Conference on Conceptual Modeling (ER)*, 96.

[128] S. Illner, A. Pohl, H. Krumm, I. Luck, D. Manka, T. Sparenberg, "Automated runtime management of embedded service systems based on design-time modeling and model transformation", *Industrial Informatics*, 2005. INDIN '05: 2005 *3rd IEEE International Conference on*, 10-12 Aug. 2005, Peth, Australia, pages 134-139

[129] H. Yamamoto, S. Sameshima, K. Kawano, "Service reconfiguration using super distributed objects (SDO) in context-aware service systems", *Software Technologies for Future Embedded and Ubiquitous Systems, 2004. Proceedings. Second IEEE Workshop on*, 11-12 May 2004, Wien, Austria, pages163-165

[130] M. Colombo, E. Di Nitto, M. Mauri, "SCENE: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined Through Rules". ICSOC, 2006, Chicago, USA, 191-202

[131] L. Cavallaro, E. Di Nitto, "An approach to adapt service requests to actual service interfaces" In *proceedings of SEAMS '08* (satellite of ICSE '08), 2008, Leipzig, Germany

[132] WS-Diamond Team, "WS-DIAMOND: an approach to Web Services – DIAgnosability, MONitoring and Diagnosis", Proc. E-Challenges Conference, The Hague, October 2007

[133] C. Cappiello, B. Pernici, "A Methodology for Information Quality Management in Self-healing Web Services", *International Conference on Information Quality*, Boston, Nov. 2006

[134] Elisabetta Di Nitto, Massimiliano Di Penta, Alessio Gambi, Gianluca Ripa, Maria Luisa Villani: Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach. ICSOC 2007: 295-306.

[135] B. Pernici (ed.), Mobile Information Systems: Infrastructure and design for adaptivity and flexibility, Springer, April 2006

[136] Carlo Batini, Davide Bolchini, Stefano Ceri, Maristella Matera, Andrea Maurino, Paolo Paolini: The UM-MAIS Methodology for Multi-channel Adaptive Web Information Systems. World Wide Web 10(4): 349-385 (2007)

[137] C. Cappiello, M. Comuzzi, E. Mussi, B. Pernici. "Context Management for Adaptive Information Systems". *Proceedings of the International Workshop on Context for Web Services* (CWS '05) in conjunction with the 5th International and Interdisciplinary Conference on Modelling and Using Context, July 2005.

[138] V. Talwar, W. Qinyi, C. Pu, Y. Wenchang, J. Gueyoung and D. Milojicic, "Distributed Computing Systems, Comparison of Approaches to Service Deployment", *ICDCS 2005.* In *Proceedings. 25th IEEE International Conference on*, 10 June 2005, pages 543-552

[139] C. Chrysoulas, G. Koumoutsos, S. Denazis, K. Thramboulidis, O. Koufopavlou, "Dynamic Service Deployment using an Ontology based Description of Devices and Services", In *Networking and Services, 2007. ICNS. Third International Conference on*, 19-25 June 2007, Athens, Greece, pages 80-80

[140] A. Avanes, J. Freytag, C.Bornhovd, *Distributed Service Deployment in Mobile Ad-Hoc Networks*, *Mobile and Ubiquitous Systems: Networking & Services, 2007.* MobiQuitous 2007. *Fourth Annual International Conference on*, 6-10 Aug. 2007, Dublin, Ireland, pages 1-8

[141] S.B. Musunoori, F. Eliassen, and V.S.W. Eide, "QoS-driven service configuration in computational grids", In *Grid Computing*, 2005. The 6th IEEE/ACM International Workshop on, 13-14 Nov. 2005, Seattle, Washington, USA.

[142] E. Jae-Wan Jang, W. Jung and J Kim, "A dynamic grid services deployment mechanism for on-demand resource provisioning", In *Cluster Computing and the Grid, 2005.* CCGrid 2005. *IEEE International Symposium on*, 9-12 May 2005, Cardiff, UK, pages 863-870 Vol. 2

[143] Xiaoning Wang, Wei Li, Hong Liu and Zhiwei Xu, "A Language-based Approach to Service Deployment", In *Services Computing*, SCC '06. *IEEE International Conference on*, Sept. 2006, Chicago, USA, pages 69-76

[144] Wonjae Lee, Yuhyeon Bak, Sangmin Woo, Changsoo Kim, Okgi Min and Hak-Young Kim, "Design and implementation of the script-based service deployment solution", in *Advanced Communication Technology, ICACT 2006. The 8th International Conference*, 20-22 Feb. 2006, Phoenix Park, Korea, volume 3, pages 2 pp.

[145] Nixes, ***http://www.aqualab.cs.northwestern.edu/nixes.html***

[146] M. Klein, B. Konig-Ries, "Combining query and preference – an approach to fully automatize dynamic service binding", in *Web Services*, 2004. *Proceedings. IEEE International Conference on*, 6-9 July 2004, San Diego, California, USA, pages 788-791

[147] U. Kuster, B. Konig-Ries, M. Stern, M. Klein, "DIANE - An Integrated Approach to Automated Service Discovery, Matchmaking and Composition", Feb. 2007

[148] A. Cole, S. Duri, J. Munson, J. Murdock, D. Wood, "Adaptive service binding middleware to support mobility", in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, 19-22 May 2003, San Juan, Puerto Rico, pages 369-374

[149] F. Emekci, O.D. Sahin, Divyakant Agrawal and Amr El Abbadi, "A peer-to-peer framework for Web service discovery with ranking", in *Web Services, 2004. Proceedings. IEEE International Conference on*, 6-9 July 2004, San Diego, California, USA, pages 192-199

[150] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan, *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*, *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149—160, 2001, San Diego, CA, USA

[151] L. Kovacs, A. Micsik and P. Pallinger, "Two-phase Semantic Web Service Discovery Method for Finding Intersection Matches using Logic Programming", In SemWS'06: *Workshop on Semantics for Web Services*, 2006, Zurich, Switzerland

[152] S. van Splunter, P.H.G. van Langen, F.M.T. Brazier, "The role of local knowledge in complex Web service reconfiguration", Web Intelligence, 2005. Proceedings. *The 2005 IEEE/WIC/ACM International Conference on*, 19-22 Sept. 2005, Compiegne, France

[153] PengCheng Xiong, Yu Shun Fan, MengChu Zhou, "Petri net-based Approach to QoS-aware Configuration for WS", In: *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on,* 7-10 Oct. 2007, pages 1286-1291, Montreal, Que.

[154] Tong Gao, Hui Ma, I-Ling Yen, Farokh Bastani, Wei-ek Tsai, "Toward QoS analysis of adaptive service-oriented architecture", *Service-Oriented System Engineering, 2005.* SOSE 2005. *IEEE International Workshop*, 20-21 Oct. 2005, Zurich, Switzerland, pages 219-226

[155] D. Karastoyanova, A. Buchmann, "ReFFlow: A Model and generic Approach to Flexibility of Web Service Compositions", iiWAS, 2004, Jakarta, Indonesia

[156] Jianqiang Hu, Changguo Guo, Peng Zou, "WSCF: a framework for Web service-based application supporting environment", *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, 11-15 July 2005, Orlando, Florida, USA

[157] M. Di Penta, R. Esposito, M.L. Villani, R. Codato, M. Colombo and E. Di Nitto, "WS Binder: a framework to enable dynamic binding of composite web services", In SOSE '06: *Proceedings of the 2006 international workshop on Service-oriented software engineering*, 2006, Shanghai, China, pages 74 – 80

[158] Mule, http://mule.mulesource.org/

[159] Hui Wang, Yuhui Zhao, Deguo Yang, Cuirong Wang, and Yuan Gao, "An Agent-Based Services Composition Framework for Ubiquitous Media", *PRIMA 2006, LNAI 4088*, pp. 478–483, 2006, Guilin, China.

[160] D. Ardagna , M. Comuzzi, E. Mussi, B. Pernici, P. Plebani: "PAWS: A Framework for Executing Adaptive Web-Service Processes". *IEEE Software 24(6)*: 39-46, 2007

[161] M. P. Papazoglou "The Challenges of Service Evolution", Keynote address, In *Procs. Advanced Information Systems Engineering Conf.*: CAISE 2008, Springer-Verlag, Lecture Notes in Computer Science, Montpellier, France, June 2008.

[162] V. Andrikopulos, S. Benbernou, M. P. Papazoglou "Managing the Evolution of Service Specications", In *Procs. Advanced Information Systems Engineering Conf.*: CAISE 2008, Springer-Verlag, Lecture Notes in Computer Science, Montpellier, France, June 2008.

[163] G. Spanoudakis, K. Mahbub, A. Zisman, "A Platform for Context Aware Runtime Web Service Discovery", *Web Services, 2007.* ICWS 2007, July 2007, Salt Lake City, Utah, USA, pages 233-240

[164] D. Bianchini, V. De Antonellis, M. Melchiori, B. Pernici, P. Plebani. (2006). "Ontology based methodology for e-service discovery". *Information Systems*. vol. 31(4-5), pp. 361-380 ISSN: 0306-4379.

[165] G. Canfora, M. Di Penta, R. Esposito, and M.L. Vilani, "An Approach for QoSaware Service Composition based on Genetic Algorithms", in Proceedings of GECCO, 2005.

[166] D. Bianculli, R. Jurca, W. Binder, C. Ghezzi and B. Faltings, "Automated Dynamic Maintenance of Composite Services based on Service Reputation", in *Proceedings of International Conference on Service-Oriented Computing (ICSOC)*, 2007.

[167] T. Friese, J.P. Mueller and B. Freisleben, "Self-Healing Execution of Business Processes Based on a Peer-to-Peer Service Architecture", in *Proceedings of International Conference on Service-Oriented Computing (ICSOC)*, 2005.

[168] A. Brogi and R. Popescu, "Automated Generation of BPEL Adapters", in *Proceedings of International Conference Service-Oriented Computing (ICSOC),* 2006, pp. 27 – 39

[169] B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani, “Developing Adapters for Web Services Integration”, in *17th International Conference Advanced Information Systems Engineering, CAiSE’2005*, 2005, pp. 415 – 429.

[170] M. Dumas, M. Spork and K. Wang, “Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation”, in *Proceedings of BPM’06*, 2006, pp. 65 – 80.

[171] S. K. Williams, S. A. Battle, J. E. Cuadrado, “Protocol Mediation for Adaptation in Semantic Web Services”, ESWC, 2006, pp. 635-649

[172] S. Rinderle and M. Reichert, “A Formal Framework for Adaptive Access Control Models”, J. Data Semantics, vol. 9, 2007, pp. 82 – 112.

[173] L. T. Ly, and S. Rinderle and P. Dadam, “Integration and Verification of Semantic Constraints in Adaptive Process Management Systems”, Data Knowl. Eng., 64 (1), 2008, pp. 3 – 23.

[174] White, S.R. and Hanson, J.E. and Whalley, I. and Chess, D.M. and Kephart, J.O., “An architectural approach to autonomic computing”, Autonomic Computing, 2004. Proceedings. International Conference on, 17-18 May 2004, pages 2-9.

[175] S-Cube, "Survey of Quality Related Aspects Relevant for Service-Based Applications," 2008.

[176] L. Osterweil, "Strategic Directions in Software Quality," *ACM Computing Surveys,* vol. 28, pp. 738-750, 1996.

[177] N. Delgado, A. Q. Gates, and S. Roach, "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools," *IEEE Transactions on Software Engineering,* vol. 30, pp. 859-872, 2004.

[178] S-Cube, "State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adpatation of SBAs," 2008.

[179] M. Papazoglou, W.J.A.M. van den Heuvel “Service-oriented design and development methodology”. In: International Journal of Web Engineering and Technology vol.2, nr.4, 2006, pp.412-442.

[180] S. Durvasula et al.: “Introduction to Service Lifecycle. SOA Practitioner’s Guide. Part 3” BEA Systems, **Error! Hyperlink reference not valid.**, August 2007.

[181] S. Durvasula et al.: “SOA Reference Architecture. SOA Practitioner’s Guide. Part 2” BEA Systems, 2007. **Error! Hyperlink reference not valid.**, August 2007

[182] A. Brown et. al.,: “SOA Development Using the IBM Rational Software Development Platform: A Practical Guide”, Rational Software, **Error! Hyperlink reference not valid.**, September 2005.

[183] J. Ganci et al. “Patterns: SOA Foundation Service Creation Scenario”. IBM Redbook, **Error! Hyperlink reference not valid.**, September 2006.

[184] O. Zimmermann, P. Krogdahl, C. Gee, “Elements of Service-Oriented Analysis and Design”. IBM developerWorks article, **Error! Hyperlink reference not valid.**, June 2004.

[185] O. Zimmermann, J. Koehler, F. Leymann, “Architectural Decision Models as Micro-Methodology for Service-Oriented Analysis and Design”. In: Lübke, D. (ed.), Proc. of the Workshop on Software Engineering Methods for Service-oriented Architecture 2007 (SEMSOA 2007), Hannover, Germany, online CEUR-WS.org/Vol-244 (2007)

[186] O. Zimmermann, U. Zdun, T. Gschwind, F. Leymann, “Combining Pattern Languages and Architectural Decision Models into a Comprehensive and Comprehensible Design Method”. In: Garlan D., Woods E., Proceedings of Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA) 2008. IEEE Computer Society, Los Alamitos (2008)

[187] A. Arsanjani, “Service-oriented modeling and architecture”, IBM developerWorks article, http://www.ibm.com/developerworks/library/ws-soa-design1/, November 2004.

[188] A. Marconi, M. Pistore, P. Poccianti, P. Traverso, “Automated Web Service Composition at Work: the Amazon/MPS Case Study”, IEEE International Conference on Web Services (ICWS 2007), 2007, 767-774

[189] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli and P. Traverso, “ASTRO: Supporting Composition and Execution of Web Services”, Third International Conference Service-Oriented Computing - ICSOC 2005, 2005, pages 495-501

[190] F. Barbon, P. Traverso, M. Pistore and M. Trainotti, “Run-Time Monitoring of Instances and Classes of Web Service Compositions”, Proceeding of the International Conference on Web Services (ICWS), 2006, pages 63-71

[191] R. Kazhamiakin and M. Pistore, “Static Verification of Control and Data in Web Service Compositions”, Proceeding of the International Conference on Web Services (ICWS), 2006

[192] R. Kazhamiakin, P. K. Pandya and M. Pistore, “Representation, Verification, and Computation of Timed Properties in Web Service Compositions”, Proceeding of the International Conference on Web Services (ICWS), pages 497-504, 2006

[193] Derler, P., Weinreich, R., Models for SOA governance, in the proceedings of the international conference on trends in enterprise application architecture (TEAA), Springer, LNCS, 2006.

[194] ACM SIGCHI Curricula for Human-Computer Interaction, http://sigchi.org/cdg/cdg2.html#2_1

[195] Amazon mechanical Turk homepage, http://www.mturk.com/mturk/welcome

[196] Adobe Developer Connection, http://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html

[197] J. Preece, Y. Rogers, D. Benyon, S. Holland and T. Carey, *Human-Computer Interaction,*.Addison-Wesley, 1994

[198] F. Paterno, C. Santoro, “Preventing user errors by systematic analysis of deviations from the system task model” in *International Journal of Human-Computer Studies*, Vol. 56, Issue 2, pp. 225-245, 2002

[199] G.D. Abowd, “Using formal methods for the specification of user interfaces” in *Proceedings of the Second Irvine Software Symposium,* pp. 109-130, 1992

[200] P. Palanque and S. Basnyat, “Task Patterns for Taking into Account in an Efficient and Systematic Way Both Standard and Abnormal User Behavior” in *IFIP 13.5 Working Conference on Human Error, Safety and Systems Development*, pp. 109-130, 2004

[201] D. Diaper and N. Stanton, *The Handbook of Task Analysis for Human-computer Interaction,* Lawrence Erlbaum Associates, 2003

[202] D. Sinnig, M. Wurdel, P. Forbrig, P. Chalin and F. Khendek, “Practical Extensions for Task Models” in lecture Notes in Computer Science, Vol. 4849/2007, pp. 42-55, 2007

[203] J. Richardson, T. Ormerod and A. Shepherd, “The role of task analysis in capturing requirements for interface design” in *Interacting with Computers*, Vol. 9, Issue 2, pp. 367-384, 1998

[204] S. Wilson, P. Johnson, C. Kelly, J. Cunningham and P. Markopoulos, “Beyond hacking: a model based approach to user interface design” in *Procedings of HCI'93*, pp. 217-231, 1993

[205] N.K. Tselios, N.M. Avouris, M. Kordaki, “Student Task Modeling in Design and Evaluation of Open Problem-Solving Environments” in *Education and Information Technologies*, Vol. 7, Issue 1, pp. 17-40, 2002

[206] B. Kirwan and L.K. Ainsworth (eds), *A Guide to Task Analysis,* Taylor and Francis, 1992

[207] “Task Analysis Part II: the application of task analysis”, Class notes for PE2, UCLIC, University College London, 2007

[208] F. Paterno, C. Mancini, S. Meniconi, “ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models” in *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*, pp. 362-369, 1997

[209] M. Ruiz, V. Pelechano and O. Pastor, “Designing Web Services for Supporting User Tasks: A Model Driven Approach” in *Lecture Notes in Computer Science*, Vol. 4231/2006, pp. 193 – 202, 2006

[210] D. Sinnig, P. Forbrig and A. Seffah, "Patterns in Model-Based Development" http://wwwswt.informatik.uni-rostock.de/deutsch/Interact/07SinnigForbrigSeffah.pdf

[211] D. J. Carmichael, J. Kay and B. Kummerfeld, "Consistent Modeling of Users, Devices and Sensors in a Ubiquitous Computing Environment", *User Modeling and User-Adapted Interaction*, Vol. 15, Issue 3-4, pp 197-234, 2005

[212] G. Fischer, "User Modeling in Human-Computer Interaction", *User Modeling and User-Adapted Interaction,* Vol. 11, Issue 1-2, pp. 65-86, 2001

[213] C. Stary, "User Diversity and Design Representation: Towards Increased Effectiveness in Design forAll", http://www.ce.uni-linz.ac.at, 2001

[214] A. Granić and J. Nakić "Designing Intelligent Interfaces for e-Learning Systems: The Role of User Individual Characteristics" in *Lecture Notes in Computer Science* (0302-9743) 4556, pp 627-636, 2007

[215] E. Shifroni and B. Shanon, "Interactive User Modeling: An Integrative Explicit-Implicit Approach" in *User Modeling and User-Adapted Interaction 2*, pp 287-330, 1992

[216] E. Frias-Martinez, S. Y. Chen, R. D. Macredie and X. Liu, "The role of human factors in stereotyping behavior and perception of digital library users: a robust clustering approach", *User Modeling and User-Adapted Interaction*, Vol. 17, Number 3 pp. 305-337, 2007

[217] E. Rich, "Users are individuals:- individualizing user models" in *International Journal of Human-Computer Studies*, Vol. 51, Number 2, pp 323-338, 1999

[218] M. Cannataro, A. Cuzzocrea and A. Pugliese, "A Probabilistic Approach to Model Adaptive Hypermedia Systems" in *Lecture Notes in Computer Science*, Vol. 2115/2001, pp.132-141, 2001

[219] A. Lorenz, "Agent-Based Ubiquitous User Modeling" in *Lecture notes in Computer Science*, Vol. 3538/2005, pp 512-514, 2005

[220] D. N. Chin, "Empirical Evaluation of User Models and User-Adapted Systems" in *User Modeling and User-Adapted Interaction*, Vol. 11, Issue 1-2, pp. 181-194, 2001

[221] M. Heymann and A. Degani, "Formal Analysis and Automatic Generation of User Interfaces: Approach, Methodology, and an Algorithm", http://goldberg.berkeley.edu/courses/S06/IEOR-170-S06/docs/Degani07_UI_Gen.pdf , 2007

[222] B. Myers, S. E. Hudson and R. Pausch, "Past, Present, and Future of User Interface Software Tools" in *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 7, Issue 1, pp. 3-28, 2000

[223] K. Gajos, D. Weld, "SUPPLE: Automatically Generating User Interfaces" in *Proceedings of the 9th international conference on Intelligent user interfaces*, pp. 93-100, 2004

[224] J. Nichols, B.A. Myers, M. Higgins, J. Hughes, T.K. Harris, R. Rosenfeld and M. Pignol, "Generating Remote Control Interfaces for Complex Appliances" in *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pp. 161-170, 2002

[225] J. Nichols, D. H. Chau and B. A. Myers, "Demonstrating the Viability of Automatically Generated User Interfaces", *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1283 - 1292, 2007

[226] K. Z. Gajos, J. J. Long and D. S. Weld, "Automatically Generating Custom User Interfaces for Users With Physical Disabilities" in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 243-244, 2006

[227] F. Bellas, I. Paz, A. Pan, O. Díaz, V. Carneiro and F. Cacheda, "An Automatic Approach to Displaying Web Applications as Portlets" in *Lecture Notes in Computer Science,* Vol. 4317/2006, pp. 264-277, 2006

[228] OASIS http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

[229] Aqua Human Interface Guidelines http://interface.free.fr/Archives/AquaHIGuidelines.pdf

[230] Java Look and Feel Design Guidelines http://java.sun.com/products/jlf/ed2/book/

[231] J. Nichols, B. A. Myers and K. Litwack, "Improving Automatic Interface Generation with Smart Templates" in *Proceedings of the 9th international conference on Intelligent user interfaces*, pp. 286-288, 2004

[232] B. Strauch, *Investigating human error: incidents, accidents and complex systems,* Ashgate Publishing Ltd 2004

[233] J. W. Senders and N. P. Moray, *Human error: Cause, Prediction, and Reduction,* Lawrence Erlbaum Associates, 1991

[234] T. Bove, *Development and Validation of a Human Error Management Taxonomy,* PhD. Dissertation, University of Roskilde, 2002

[235] J. Reason, *Human error,* Cambridge University Press, 1998.

[236] C. van Dyck, M. Frese, M. Baer and S. Sonnentag, "Organizational Error Management Culture and Its Impact on Performance: A Two-Study Replication", http://www.olin.wustl.edu/workingpapers/pdf/2006-10-002.pdf , 2005

[237] S. Dekker, *The field guide to human error investigations,*Ashgate Publishing Ltd 2002

[238] D. A. Wiegmann and S. A. Shappell, *A Human error approach to aviation accident analysis,*Ashgate Publishing Ltd 2003

[239] S. Pocock, M. Harrison, P. Wright and P. Johnson, "THEA: A Technique for Human Error Assessment Early in Design", http://homepages.cs.ncl.ac.uk/michael.harrison/papers/int01pub4.pdf , 2001

[240] A. Rizzo, O. Parlangeli, E. Marchigiani and S. Bagnara, "The Management of Human Errors in User-Centered Design" in *SIGCHI Bulletin*, Vol. 8, Number 3, 1996

[241] M. Bonnet, "Personalization of Web Services: Opportunities and Challenges", http://www.ariadne.ac.uk/issue28/personalization/

[242] C. Muldoon, G. O'Hare, D. Phelan, R. Strahan, R. Collier, "ACCESS: An Agent Architecture for Ubiquitous Services Delivery", *Refereed proceedings of the 7th Information Agents*, CIA 2002

[243] J. Fink, A. Kobsa, A.Nill, "Adaptable and Adaptive Information Provision for All Users, Including Disabled and Elderly People", http://www.isr.uci.edu/~kobsa/papers/1998-NRHM-kobsa.pdf , 1998

[244] P. Brusilovsky, A. Kobsa, W. Nejdl, eds. *The Adaptive Web: Methods and Strategies of Web Personalization*, Springer Verlag. 2007

[245] A. Kobsa, "Generic User Modeling System" in *User Modeling and User-Adapted Interaction*, 11(1-2), 49-63, 2001

[246] P. Dolog, N. Henze, W. Nejdl, M. Sintek, "Personalization in Distributed e-Learning Environments" in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters,* 2004

[247] D. Bonnefoy, M. Bouzid, N. Lhuillier, K. Mercer, "'More Like This' or 'Not for Me': Delivering Personalized Recommendations in Multi-user Environments" in *Lecture Notes in Computer Science*, Vol. 4511, 2007

[248] L. Ardissono, M. Maybury, "Special Issue on UserModelling and Personalization for Television" in *User Modeling and User-adapted Interaction*, 14(1), 2004

[249] Y. Wang, and A. Kobsa, “Respecting Users' Individual Privacy Constraints in Web Personalization” in *Proceedings of the 11th International Conference on User Modeling*, Corfu, Greece, 157–166, 2007

[250] T. Vitvar, M. Zaremba, M. Moran, “SESA: Emerging Technology for Service-Centric Environments Software” in *IEEE Spftware,* Vol. 24, Issue 6, pp. 56 – 67, 2007

[251] A. Rockley, P. Kostur, S. Manning, *Managing Enterprise Content*, New Riders, 2003

[252] N. Tractinsky, “Tools over Solutions? Comments on Interacting with Computers special issue on affective computing” in *Interacting with Computers*, pp. 751-757, Vol. 16 , Issue 4, 2004

[253] J. A. Marcias and F. Paterno, “Customization of Web applications through an intelligent environment exploiting logical interface descriptions” in *Interacting with Computers*, Vol. 20, Issue1, 2008

[254] N. Bila, T. Ronda, I. Mohomed, N. Khai, T.E. de Lara, “PageTailor: Reusable End-User Customization for the Mobile Web”, http://www.cs.toronto.edu/~delara/papers/mobisys2007.pdf, 2007

[255] O. Stiemerling, “Supporting Tailorability in Groupware through Component Architectures” in *Proceedings of the ECSCW '97 Workshop on Object Oriented Groupware Platforms*, pp. 53-57, 1997

[256] O. Stiemerling, A.B. Cremers, “The Evolve  project: Component-based Tailorability for CSCW applications” in *AI and Society*, Vol. 14, Number 1, 2000

[257] M. Won, O. Stiemerling, V. Wulf, “Component Based Approach to Tailorable Systems”, http://www.uni-siegen.de/fb5/wirtschaftsinformatik/paper/2006/wonstiemerlingwulf_componentbasedapproachestotailorablesystems.pdf , 2005

[258] R. Hinken, O. Stiemerling, A.B. Cremers, “Distributed Component-Based Tailorability for CSCW Applications” in *Proceedings of the Fourth International Symposium on Autonomous Decentralized*, pp. 345, 1999

[259] M. Bolin, M. Webber, P. Rha, T. Wilson, R.C. Miller, “Automation and customization of rendered Web pages” in *Proceedings of the 18th Symposium on User Interface Software and Technology (UIST’05)*, pp. 23-27, 2005

[260] Greasemonkey http://greasemonkey.mozdev.org/

[261] Platypus, http://platypus.mozdev.org/

[262] U. Manber, A. Patel, J. Robison, “Experience with personalization of Yahoo!” in *Communications of the ACM*, Vol. 43, Issue 8, pp. 35-39, 2000

[263] T.F. Paymans, J. Lindenberg, M. Neerincx, “Usability Trade-offs for Adaptive User Interfaces: Ease of Use and Learnability”, http://www.cactus.tudelft.nl/CactusPublications/D2004.24.pdf, 2004

[264] M. Bila, T. Ronda, I. Mohomed, K.N. Truong, E. de Lara, “PageTailor” in *Proceedings of the 5th international conference on Mobile systems, applications and services*, pp. 16 – 29, 2007

[265] W.E. Mackay, *Users and customizable Software: A Co-Adaptive Phenomenon*, PhD Dissertation, MIT, Boston, 1990

[266] A. MacLean, K. Carter, L. Lövstrand, T. Moran, “User-tailorable Systems: Pressing the Issue with Buttons” in *Proceedings of CHI '90*, pp. 175-182, 1990

[267] B.A. Nardi, *A Small Matter of Programming - Perspectives on end-user computing,* MIT-Press, 1993

[268] R. Oppermann, H. Simm, “Adaptability: User-Initiated Individualization”, in: R. Oppermann, (ed.): *Adaptive User Support – Ergonomic Design of Manually and Automatically Adaptable Software*, Lawrence Erlbaum Associates, 1994

[269] S. Page, T. Johnsgard, U. Albert, C. Allen, “User Customization of a Word Processor” in *Proceedings of CHI '96*, pp. 340-346, 1996

[270] V. Wulf, B. Golombek, “Direct Activation: A Concept to Encourage Tailoring Activities” in *Behavior and Information Technology*, Vol. 20, Issue. 4, pp. 249-263, 2001

[271] D.N. Kalofonos and F.D. Reynolds, “Task-Driven End-User Programming of Smart Spaces Using Mobile Devices”, Nokia Research Center Technical Report (NRC-TR-2006-001), http://research.nokia.com/tr/NRC-TR-2006-001.pdf, 2006.

[272] X. Liu, L. Zhou, G. Huang, H. Mei, “Towards service pool based approach for services discovery and subscription” in *Proceedings of the 16th international conference on World Wide Web*, pp. 1253–1254, 2007

[273] W.J. Mellors, “Statistics on age and disability and in relation to Telecommunications - A significant market”, WGHI - Working Group on Hearing Impairment, http://www.tiresias.org/wghi/stats.htm, 2008

[274] S. Toyne, “Ageing: Europe's growing problem”, http://news.bbc.co.uk/2/hi/business/2248531.stm

[275] I. Klironomos, M. Antona, I. Basdekis, C. Stephanidis, “White Paper: Promoting Design for All and e-Accessibility in Europe” in *Universal Access in the Information Society*, Vol. 5, Issue 1, pp. 105-119, 2006

[276] G.C. Vanderheiden, “Making Software more Accessible for People with Disabilities” in *Computers and the Physically Handicapped*, Issue 47, pp. 2 – 32, 1993

[277] C. Stephanidis, D. Akoumianakis, M. Sfyrakis, A. Paramythis, “Universal accessibility in HCI: Process-oriented design guidelines and tool requirements”, http://www.ui4all.gr/UI4ALL-98/stephanidis1.pdf

[278] J. Abascal, M. Arrue, I. Fajardo, N. Garay, J. Tomás, “The use of guidelines to automatically verify Web accessibility” in *Universal Access in the Information Society*, pp. 71-79, 2004

[279] B. Berendt, A Hotho, D. Mladenic, G. Semeraro, (eds.), *From Web to Social Web: Discovering and Deploying User and Content Profiles,* SpringerVerlag, 2007

[280] Web Service Internationalization Usage Scenarios, http://www.w3.org/TR/ws-i18n-scenarios/

[281] B. Berendt, A. Kralisch, “From World-Wide-Web Mining to Worldwide Webmining: Understanding People’s Diversity for Effective Knowledge Discovery” in *Lecture Notes in Computer Science*, Vol. 4737, pp.102-121, 2007

[282] W3 Web accessibility guidelines http://www.w3.org/TR/WAI-WEBCONTENT/

[283] XML Accessibility guidelines, http://www.w3.org/TR/xag

[284] Accessible Rich Internet Applications http://www.w3.org/TR/wai-aria/

[285] W.D. Gray, M.C. Salzman, “Damaged merchandise: a review of experiments that compare usability evaluation method” in *Human–Computer Interaction*, 13(3), pp.203–261, 1998

[286] ITTATC http://uiaccess.com/accessucd/

[287] J. Nielsen, R.L. Mack (eds), *Usability Inspection Methods*, John Wiley and Sons, 1994

[288] Heuristic evaluation, http://www.useit.com/papers/heuristic/

[289] J. Preece, Y. Rogers, H. Sharp, *Interaction design,* John Wiley and Sons, 2002

[290] UsabilityNet http://www.usabilitynet.org/tools/methods.htm

[291] LIFT http://www.usablenet.com

[292] Bobby service http://bobby.watchfire.com

[293] TAW http://www.tawdis.net

[294] WAVE http://wave.webaim.org

[295] M.Y. Ivory, R.R. Sinha, M.A. Hearst, *Empirically Validated Web Page Design Metrics,* http://webtango.berkeley.edu/papers/chi2001/chi2001.pdf

[296] J. Abascal, M. Arrue, I. Fajardo, N. Garay, J. Tomás, "The use of guidelines to automatically verify Web accessibility" in *Universal Access in the Information Society*, Vol. 3, Issue 1, pp.71-79, 2004

[297] A. Beirekdar, J. Vanderdonckt, M. Noirhomme-Fraiture, "Kwaresmi – Knowledge-based Web Automated Evaluation with REconfigurable guidelineS optimization" in *PreProceedings of 9th International Workshop on Design, Specification, and Verification of Interactive Systems,* 2002.

[298] B. Leporini, F. Paterno, A. Scorcia, "Flexible tool support for accessibility evaluation" in "*Interacting with Computers*", Vol. 18 , Issue 5, pp. 869-890, 2006

[299] S. Kurniawan, P. Zaphiris, "Research-derived web design guidelines for Older People" in *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pp 129 – 135, 2005

[300] M.Y. Ivory, A. Chevalier, "A Study of Automated Web Site Evaluation Tools", ftp://ftp.cs.washington.edu/tr/2002/10/UW-CSE-02-10-01.pdf

[301] Disability Act, http://www.assistireland.ie/index.asp?locID=1609&docID=4819

[302] The Disability Discrimination Act (DDA) 1995, http://www.direct.gov.uk/en/DisabledPeople/RightsAndObligations/DisabilityRights/DG_4001068

[303] D.M. Boyd, N.B. Ellison, "Social network sites: Definition, history, and scholarship", http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html

[304] A. de Moor, W.-J. van den Heuvel, "Web Service Selection in Virtual Communities" in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, Vol. 7, pp 10, 2004

[305] M.N.K. Boulos, S. Wheeler, "The emerging Web 2.0 social software: an enabling suite of sociable technologies" in *Health Info Library Journal*, 24(1), pp.2-23, 2007

[306] NetLogo http://ccl.northwestern.edu/netlogo/

[307] SNA presentation http://www.slideshare.net/jimbbq/social-networks-and-social-simulation-of-3d-online-communities

[308] A.P. Barros, M. Dumas, P. Bruza, "The Move to Web Service Ecosystems", http://www.bpm.fit.qut.edu.au/projects/serveco/documents/12-05-WP-WebServiceEcosystems-Barros-Dumas.pdf

[309] A.P. Barros, M. Dumas, "The Rise of Web Service Ecosystems" in *IT Professional*, Vol. 8, Issue 5, pp.31 – 37, 2006

[310] P. Dourish, R. E. Grinter, J. Delgado de la Flor and M. Joseph, "Security in the wild: user strategies for managing security as an everyday, practical problem" in *Personal and Ubiquitous Computing*, Vol. 8 pp 391–401, 2004

[311] FOLDOC http://foldoc.org/ accessed April 2008

[312] Webopedia http://www.webopedia.com/TERM/s/security.html accessed April 2008

[313] J. H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems", http://www.cs.virginia.edu/~evans/cs551/saltzer/, 1975

[314] A.T. Vassilev, B. du Castel and A.M. Ali, "Personal Brokerage of Web Service Access" in *IEEE Security & Privacy*, Vol. 5, Issue 5 pp 24-31, 2007

[315] B. Schneier, *Secrets and Lies,* John Wiley & Sons, 2000.

[316] OpenID http://wiki.openid.net/REST/SOAP/HTTP_Bindings

[317] T. Straub *Usability Challenges of PKI,* PhD. Dissertation Darmstadt Technical University, 2005

[318] A. Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0" in *Proceedings of the 8th Usenix Security Symposium*, pp. 169–184, 1999

[319] A.M. Sasse, S. Brostoff, S. and D. Weirich, "Transforming the 'Weakest Link': a Human/Computer Interaction Approach to Usable and Effective Security" in *BT Technology Journal*, Vol. 19, Number 3, July 2001

[320] M. Andrews and J.A. Whittaker, "Computer Security" in *IEEE Security & Privacy* , Vol. 2, Issue 5, 2004

[321] K.P. Yee, "Secure Interaction Design", http://people.ischool.berkeley.edu/~ping/sid/ , 2002

[322] B. Friedman, P. Lin and J. K. Miller, "Informed Consent by Design", http://hornbeam.cs.ucl.ac.uk/hcs/teaching/GA10/lec9extra/ch24friedman.pdf , 2005

[323] I. Flechais, *Designing Secure and Usable System,* PhD Dissertation University College London, 2005

## Some relevant definitions

**Actor**: Actors are organisational units, roles, positions or other systems.

**Binding:** By *binding* we mean the process of associating a service request to a service offer.

**Deployment:** By *deployment* we mean the process of concretely associate services to devices in a real-world system, and the set of choices that must be made to achieve this goal.

**Discovery:** By *discovery* we mean the process of locating the services that provide the required functionalities.

**Error**: generic term to encompass all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and where these failures cannot be attributed to the intervention of some chance agency

**Goal**: A stakeholder's goal is an objective, the system to be should achieve. In i* an Tropos goals represent the actor's strategic interests of the future system.

**Hard-Goal**: Any goal, which has clear cut criteria to decide whether the goal is satisfied or not is a hard-goal.

**Map**: A Map is a directed graph containing intentions as nodes and strategies as edges. The graph is used to represent process structures.

**Method:** A *method* provides a prescription for how to perform a collection of activities focusing on how a related set of techniques can be integrated and providing guidance on their use.

**On-the-fly composition:** *On-the-fly composition* is a run-time composition of component services into complex ones, performed according to the current state of the context at the moment the service is actually required.

**Plan**: Plans represent activities carried out by some actor to satisfy goals or to satisfice soft-goals.

**Portal**: system for collecting, tailoring and displaying different kind of data onto a single screen for a user

**Portlet**: basic component of a portal, ) representing an interactive web mini application

**Portletising**: wrapping existing Web applications as portlets

**Replanning:** By replanning we mean the ability of the infrastructure supporting the execution of a service-based application to introduce changes within the structure of the application itself in order to deal with undesirable situations such as the failure of a component service and the unavailability of a suitable replacement.

**Requirement**: A requirement is a goal under the responsibility of the system.

**Requirements Engineering**: Requirements engineering is one activity in the software development lifecycle, which aims to identify, document, agree upon and verify the purpose of the system to be.

**Self-healing systems**: A *self-healing system* is a system able to autonomously react to failures and problems, modifying itself to restore its correct behavior.

**Stakeholder**: A stakeholder in RE is any person or group who affects the system to be or who is affected by the system to be.

**Soft-Goal**: Soft-goals are goals which do not have clear cut criteria to decide whether the goal is satisfied or not. In contrast to hard-goals, soft-goals can only be partially fulfilled (satisficed).

**Stereotype**: Descriptive enumeration of a set of traits that often occur together

**Task model**: description of a structured sets of activities that the user has to perform to attain goals

**Technique:** A *technique* prescribes how to perform a particular activity and if necessary how to describe the product of that activity in a particular notation.

**User model**: models that systems have of users that reside inside a computational environment

**User interface (UI)**: the visible, physical representation of systems that allow users to interact with them and use the systems' functionalities

**Validation:** *Validation* can be performed in various ways: case study, experiments, arguments, simulation, analytical proof, action research (all these terms should be defined).