Project Number:	**215219**
Project Acronym:	**SOA4All**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D6.1.1. State of the Art Report and Requirements for Service Construction

Activity:	Activity 2 - Core Research and Development Activities
Work Package:	WP 6 - Service Construction
Due Date:	M6
Submission Date:	28/08/2008 Resubmission 06/03/2009
Start Date of Project:	01/03/2008
Duration of Project:	36 Months
Organisation Responsible of Deliverable:	ATOS
Revision:	3.4
Author(s):	Juergen Vogel, Lai Xu, Florian Schnabel SAP Reto Krummenacher UIBK Sven Abels TIE Rafael Gonzalez ATOS Adrian Mos, Freddy Lecue INRIA Nikolay Mehandjiev UNIMAN Luchesar Cekov SIRMA Gianluca Ripa CEFRIEL Matteo Vila TXT
Reviewers(s):	Francoise Baude INRIA Dumitru Roman, Elena Simperl UIBK Carlos Pedrinaci OU

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
1.0	22/07/2008	First integration version	Lai Xu (SAP) Sven Abels (TIE) Luchesar Cekov (SIRMA) Nikolay Mehandjiev (UniMan) Adrian Mos (INRIA) Reto Krummenacher (UIBK) Gianluca Ripa (CEFRIEL) Matteo Villa (TXT) Rafael Gonzalez (ATOS)
1.1	28/07/2008	Context extended,	Lai Xu (SAP) Nikolay Mehandjiev (UniMan) Gianluca Ripa (CEFRIEL) Reto Krummenacher (UIBK)
1.2	29/07/2008	References Context extended	Matteo Villa (TXT) Lai Xu (SAP)
1.3	31/07/2008	Updated Sections 3 and 7	Adrian Mos (INRIA)
2.0	31/07/2008	Updated Section 5 Context extended	Rafael Gonzalez (ATOS) Lai Xu (SAP)
2.1	24/08/2008	Changes according to review	Florian Schnabel (SAP)
		Review	Carlos Pedrinaci (OU)
2.2	18/08/2008	Review	Francoise Baude (INRIA)
3.0	27/11/2008	Update with reviewers comments	Rafael González (ATOS) Florian Schnabel (SAP) Sven Abels (TIE) Freddy Lecue (UniMan)

3.1	5/12/2008	Review	Florian Schnabel (SAP) Rafael González (ATOS)
3.2	6/2/2009	Review	Rafael González (ATOS)
3.3	23/2/2009	Update from Internal Reviewer (Francoise Baude, INRIA)	Rafael González (ATOS)
3.4	5/3/2009	Update from Internal Reviewers (Dumitru Roman, Elena Simperl (STI)	Rafael González (ATOS)
Final	9/3/2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

Glossary of Acronyms

Acronym	Definition
ACID	Atomicity Consistency Isolation Durability
AD	Activity Diagram
AI	Artificial Intelligence
API	Application Programming Interface
B2B	Business-to-Business
BPEL	Business Process Executable Language
BPM	Business Process Management
BPML	Business Process Modelling Language
BPMN	Business Process Modelling Notation
BPMS	Business Process Management System
CRM	Customer relationship management
DAG	Directed Acyclic Graph
ebXML	Electronic Business using eXtensible Markup Language
EC	European Commission
EPC	Event-driven Process Chains
ER	Entity-Relation
ERP	Enterprise Resource Planning
GOLOG	alGOl in LOGic
HTN	Hierarchical Task Network
IT	Information Technology
ITIL	the IT Infrastructure Library
MAPE	Monitoring, Analysis, Planning and Execution
PCM	Propose-Critique-Modify
PHP	PHP Hypertext Preprocessor
PPM	Polymorphic Process Model
PPM	Polymorphic Process Model
QoS	Quality of Service
SAP	Systeme Anwendungen und Produkte
SCA	Service Component Architecture (SCA)
SCM	Supply chain management
SCM	Supply chain management
SDK	Software Development Kit
SOA	Service-Oriented Architecture
STS	State Transition System
UML	Unified Modelling Language
WfM	Workflow Management
WP	Work Package
WSCI	Web Service Choreography Interface
XPDL	XML Process Definition Language

List of Figures

List of Tables

Executive summary

Web services are becoming the de facto standard for the implementation of distributed enterprise computing systems nowadays, as they enable collaborative business processes and ease their construction by their recombination. So far, enterprises have kept service orientation and hence the usage of Web services to provide business functionality mostly within their boundaries. However, the adoption of Web services should overcome enterprise boarders, to open their business functionality to a larger community in order to enable cross-organizational business processes.

One of the obstacles that we identify is that most of the research on Business Process Management (BPM hereafter) mainly addresses sophisticated and highly formalised process descriptions. Processes are usually specified once, instantiated very often, highly repetitive and are characterised by a certain degree of temporal stability. In order to achieve the adoption of these technologies by the mainstream of users there is still a great need of enabling non-technical users to describe their to-be processes in a lightweight manner. **Lightweight means simple to use and having an abstract way to represent composite services and processes**. In other words it means to provide a user interface and tools to easily construct, deploy and execute the services and processes as well as the underlying composition model and its representation in a specific language.

However, a lightweight BPM process modelling language needs enough expression power at the same time. To reduce the inherent complexity of BPM orchestration, we will create an automatic system for the flexible and ad-hoc composition of services. Thus, we will transform the aforementioned lightweight processes in to complex services orchestrations in a context-dependant manner.

Finally, we have to execute the newly created services. We will develop a context-aware execution infrastructure, adaptive to environmental changes; and flexible enough to allow its context-dependent self-reconfiguration. It also should exhibit some failure recovery behaviour, since the user is not aware of the complex process that is being executed and more importantly, it will not be able to handle any problem since we assume a short level of technical expertise.

This deliverable ***D6.1.1 State of the Art Report and Requirements for Service Construction*** presents an evaluation of the most relevant methods related with the afore described process, that is, techniques and tools used for the tasks of BPM language representation, automatic service construction, and the business process execution. We will present **a survey of requirements that we identify for each of the use case related work packages** on the one hand; and from the research **and technical challenges posed by SOA4All** on the other. This document will also contain **an unbiased comparison of the most recent and relevant technologies in the key research areas related to service construction in SOA4All**; summarizing the existing gaps as a list of unfulfilled requirements that will be used as an input to the technical tasks of this work package.

1. Introduction

Service-oriented computing is emerging as promising paradigm for enabling the flexible interconnection of autonomously developed and operated business applications within and across organizational boundaries [1]. But so far, most enterprises have gained lots of experience in deploying service-oriented business applications only for internal consumption. SOA4All aims to empower ordinary end users rather than enterprise IT experts. In the context of this work package is to bring service composition outside company walls; and it is easy to figure that mainstream adoption of these techniques involves the inclusion of non-experts in the service composition process. According to [2] there are three different viewpoints on service composition[1]

- ***Choreography*** (also called global model in Web Service Choreography Interface (WSCI) and multiparty collaboration in ebXML): this viewpoint captures collaborative processes involving multiple services where the interactions between these services are seen from a global perspective.
- ***Behavioural interface*** (also called abstract process in BPEL and collaboration protocol profile in ebXML): this viewpoint captures the behavioural dependencies of the interactions in which a given service can engage.
- ***Orchestration*** (also called executable process in BPEL): this viewpoint deals with the description of the interactions in which a given service can engage with other services, as well as the internal steps between these interactions (e.g., data transformations)

We will focus in third perspective of service composition; the orchestration of services, which is the one that uses executable process, from our perspective, a **meaningful business process will present the end user's idea of the business goals he plans to achieve**. It describes process steps, or captures the interactions between a process and its environment.

Henceforth we see that **there exist fundamental link between BPM and SOA**. On the one hand, emerging BPM techniques rely on SOA as a paradigm for managing resources (especially software ones), describing process steps, and capturing the interactions between a process and its environment. On the other hand, a service may serve as an entry point to an underlying business process, thereby inducing an inherent relation between the service model and the process model. Services may also engage in interactions with other services in the context of collaborative business processes.

Hitherto we have not mentioned the main impediment in the adoption of BPM solutions by non-expert users. Most research, tools and mechanisms developed on **BPM mainly addresses sophisticated and highly formalised process descriptions**. Processes are usually specified once, instantiated very often, highly repetitive and are characterised by a certain degree of temporal stability. We summarize these qualities with the term "heavyweight". In contrast, **there is a great need for enabling non-technical users to describe their to-be processes in a lightweight manner**.

Lightweight means simple to use and having an abstract and coarse-grained way to represent composite services. It aims at enabling non-technical users in describing their

[1] There are different classifications on service composition. Some of them view it from only choreography and orchestration aspects. Some of them view it from interface behaviour, provider behaviour, choreography, and orchestration aspects.

to-be processes in an environment suiting their needs. Lightweight BPM brings flexibility and affordability to the orchestration layer. Nevertheless, a Lightweight BPM language needs enough expression power at the same time.This would allow the untrained users to compose choreographies to serve their own needs. Lightweight does not only imply less coding and user friendliness. It also means lower design and deployment costs/time for applications of sufficient complexity. Lightweight maintains tradeoffs between agility and long-lasting usage. A lightweight system should include a certain degree of formal design as well as meet the service quality level that users require.

Providing Lightweight BPM design through a web-based platform enables widespread collaboration in process design[2]. A web-based platform implies some sort of universally available process repository. A process design tool should as such be a pure web platform as well. Moreover, the platform for lightweight BPM supports not only just execution of collaborative processes, but also collaborative process design.

Nevertheless, once the user creates the Lightweight BPM specification of a process it is easy to realize that there is a gap between the complexity and formalization-level of this model and those of the actual BPM execution environments. Therefore, **our user-oriented processes models must be transparently and automatically translated in to more complex and complete ones; adapted to the particular needs of a particular context**; and hence configured on a per case basis.

Finally, there is a need of a **context-aware execution infrastructure able to execute those complex processes** that will be adaptive to environmental changes; and flexible enough to allow its context-dependent self-reconfiguration. The execution engine should exhibit some failure-recovery behaviour, since the user is not aware of the complex process that the system is executing. Therefore, the user will not be able to handle any problem since, apart that we assume a low level of technical expertise, she will not be able to know what is happening.

1.1 Purpose and scope of this deliverable

The goal of this document is twofold. Firstly, this deliverable **comprises a first survey of requirements that we identify for each of the use case related work packages** on the one hand; **and from the research and technical challenges posed by SOA4All** on the other. This document will also contain **an unbiased comparison of the most recent and relevant technologies in the key research areas related to service construction in SOA4All**; in an attempt of identifying and summarizing the existing gaps as a list of unfulfilled requirements.

1.2 Structure of the document

This deliverable describes the state of the art and requirements related to service construction. The aim of the service construction work package in SOA4All is to provide techniques and tools that adapt and generate complex services and processes from simpler lightweight processes; and the context-dependant and adaptive execution of these processes. Non-technical users should be able to construct services by just using lightweight

[2] The task of creating Lightweight BMP development interface will be carried out in *WP2 SOA4All Studio*, nevertheless the graphical model for processes and its intended semantics will be developed in this work package

process design technologies. More precisely, this document will be composed of the following sections.

- **A first survey of requirements** which will be identified in the use case work packages firstly, and those posed by SOA4All challenges[3].
- **An unbiased comparison of the most recent and relevant technologies in the key research areas related to service construction in SOA4All**; namely BPM and executable languages description, adaptation and composition of Web services (along with process mining), and process execution. For each of these areas we will include:
  - ***A brief description of the most outstanding and recent approaches in the area*.**
  - ***An enumeration of the more relevant requirements for each area*.** We will include further information of the requirements included in the survey to contextualize them in the correspondent area. Note that these requirements do not substitute those contained in the general survey of requirement, but put in context or extend some of them.
  - ***A comparison of the functionalities provided by each of the described approaches*.** We will focus on those functionalities more closely related with the requirements that are listed at the beginning of each area section.
  - ***In each section we finally summarize the existing gaps in a list of unfulfilled requirements*,** which will be the main outcome of this deliverable, since they represent users' needs that are not yet addressed by existing technologies. This list of unfulfilled requirements will serve as a starting point for the work to be done in the work package.

[3] The requirements from SOA4All will follow the principles and challenges that we have thoroughly described in the deliverable D1.1.1 *Design Principles for a Service Web*.

2. Catalogue of Requirements

In this section, we will present a catalogue of the requirements that we have identified as relevant for this work package. They are related both with the use cases and with the challenges that the SOA4All global scale SOA vision poses. Let us describe them profoundly in the following sections.

2.1 Requirements from the Use Cases

2.1.1 WP7 Requirements: End-user Integrated Enterprise Service Delivery Platform

WP7 *End-User Integrated Enterprise Service Delivery Platform* use case focuses in the EU Services Directive that targets at facilitating and harmonizing the provisioning of services within the EU. "Service" in this context means all sorts of economic services and includes consulting, construction, maintenance, advertising, tourism, etc. The Directive's vision is "**to make progress towards a genuine Internal Market in Services so that, in the largest sector of the European economy, both businesses and consumers can take full advantage of the opportunities it presents"** [3]. By supporting the development of a truly integrated Internal Market in Services, the Directive will help realize the considerable potential in terms of economic growth and job creation of the services sector in Europe. For this reason, the Services Directive is a central element of the renewed Lisbon Strategy for growth and jobs. Moreover, by providing for administrative simplification, it also supports the better regulation agenda. Besides simplifying, accelerating and unifying the administrative processes of all member states, a special focus is on providing services between multiple administrations within a single state or between different states. This requires establishing new communication mechanisms between service providers and administration offices. A major element to accomplish the simplification of administrative procedures from the constituent's point of view is to install so-called single points of contact at the administrations that are responsible for guiding the constituent throughout the entire process and to provide as much help and information as is necessary.

Let us now enumerate the main requirements that we have identified as relevant to the work package

- The models and tools should support a range of different users with different roles and skills. In the concrete context of this use case we will differentiate
  - Front office: high-level knowledge of all processes
  - Back-office: very detailed knowledge of selected processes
- Processes should have associated meta-data properties that allow process search using properties such as name, author, category, data objects, user comments, etc.
- Processes should be reusable. Furthermore, a process should be identifiable as a building block that can be recombined to more complex processes.
- The process model should provide a graphical representation of processes. It will make them
  - Easy to create
  - Easy to share
- The steps in the processes should enable the representation of:
  - Concrete services (classes of services or services instances)
  - Services templates, similar to above mentioned, but with some information left intentionally unspecified.

   - Goals, which are functional descriptions of the objective that the user wants to achieve with a service invocation.
- Regarding the parameters involved in the description of process the use case requires that:
   - Dynamic input parameters may be output parameters of preceding services or context-dependent parameters.
   - The definition of static input parameters should be also possible.
   - Input parameters may be provided by the user (via browser-based UI) or by automatic information sources (services output or current context).
- Processes should have precise semantics in order to build tools that interpret them automatically, detect errors, resolve conflicts and prevent duplicates.
- Processes should be described with the enough abstraction and freedom to allow:
   - transparent deployment on demand
   - multiple and parallel running instances per process
- Regarding the execution of services
   - Processes should be executed transparently, the end user should not be aware of its internals.
   - Since the processes may involve human intervention, users should be notified if input required

2.1.2 WP8 Requirements: W21C BT Infrastructure (Telecommunications Sector-focused Case Study)

Web21C[4] is the name currently given to the platform over which BT will provide next generation services on top of its all IP-based 21st Century Network (BT 21CN). BT will provide some of these services but also third parties will provide others. **Web21C is central to BT's transformation from a traditional telecommunications company to a converged software and services business. Web21C will allow third parties to use BT's network as a platform for delivery of their services,** for which BT get revenue. These are not typically other network competitors, but a new breed of partner - software companies, developers and content providers.

Currently Web21C comprises of a set of Web services, and software development kits (SDKs) that provide external access to a number of BT capabilities, such as making a voice call and sending an SMS text message.

In the following, we will indentify different requirements from each of the scenarios that we have defined for this use case. From the ***Web21c Telco application design*** scenario (casual-user side) we identify the following:

- The representation, tools and techniques that we will develop to compose services should envisage that different communities might generate compositions, which can be either internal or external to the telecommunication company.
- Services compositions should not only be based on functionality and should take in to

[4] http://web21c.bt.com/

account different criteria, namely non-functional properties (e.g. QoS), and context.

- The lightweight process model (and the overall service construction environment) should be easy to use, lowering thus the barrier in entry in using the composition. That includes:
    - o Users don not need to have any programming experience (using SDK via Java, C# or PHP API will cause problems with users).
    - o The system should support users without knowledge of specific ontology language (no interaction, edition in their naive form).
    - o The system should support users without knowledge of Service description language such as WSMO, WSDL.
- We should define formally the lightweight model operational semantics in order to automatize tasks such as the suggestions for compatible services in web service compositions.
- Compositions should be easily extendable. The compositions will have to be saved, stored, and shared with other users of the community.
- It should be possible to annotate computed Web service compositions by the web2.0 community.
- The description of services as processes should take in to account that users should be able to classify these compositions through simple categorisation (e.g., Taxonomies of Topics) for ease-of-discovery. We should also index compositions by means of keywords, and semantic signatures.

From the ***Business Reseller scenario***, we identify the following requirements

- The lightweight process model should allow the definition of fault-handling situations, and provide constructs to report errors.
- The lightweight process model should also contain information about the QoS, context criteria, etc… in order to be used in a ranking process later on.
- Apart from a non-determinist first, the composition system should allow the possibility of computing the optimal Web service composition. This optimization process will be based on non-functional and functional parameters, context for a given goals.

2.1.3 WP9 Requirements: C2C Service eCommerce

The C2C Service eCommerce use case will be entirely focused on **providing an easy way for end users to use third party services offered through the framework, enabling them to build eCommerce applications in order to market and sell their own products**, such as photos or furniture or by providing their own innovative services built from a mash-up of existing service offers. End customers are able to use various SOA4All-enhanced tools offered through this framework to build their own end customer applications. While people may use the SOA4All results to build generic applications, the eCommerce framework will provide eCommerce specific functionality and will itself also use the SOA4All services for achieving this. For example, it will provide typical Web Shop functionalities such as a shopping cart feature and an access to payment providers using the SOA4All service orchestration and communication facilities. More precisely, WP9 will provide services for different eCommerce areas such as advertisement, marketing, distribution and payment, based on existing partner products and services. In addition, the inclusion of additional third party services via a service broker will be enabled.

The requirements for this work package of this deliverable are:

- Complex services need to be constructed based on the connection of simple services.
- Those composed services need to be orchestrated. Once the consumer has connected services, he needs to be able to launch the execution process.
- We should provide ready-to-use templates that represent commonly used processes. Users should be able to use this processes as off-the-shelf solutions.

2.2 SOA4All General Requirements

As we have presented, SOA4All will be highly steered by the requirements from use cases, Nevertheless we must also take in to account the requirements that arise from the general project objectives, the research challenges that targeting a service Web architecture poses.

In the deliverable *D1.1.1 Design Principles for a Service Web*, we present the principles and rationale behind a service Web architecture; along with a outlining of how these principles will provide the means and methods for an internet-scale deployment and adoption of SOA infrastructures. These principles are those described by the SOA paradigm, combined with the principles underlying Web, the Autonomic computing initiative, and the Semantic Web. The commitment to these principles poses specific challenges and requirements that will affect directly to this work. Let us enumerate them.

- ***Supporting both machine and human-based computation*.** In several scenarios, Web 2.0 and human computing approaches, together with their underlying social consensus-building mechanisms, have proven the potential of combining human-based services with services provided via automated reasoning. Services operated by humans can be introduced to solving tasks that otherwise remain computationally infeasible. The transparent provisioning of services abstracting over whether the 'engine' is a human or machine will significantly increase the overall quality of services available to the end-user.
- ***Dynamicity and adaptability*.** Services can appear, change, or disappear; we envisage a great services churning rate. Thus, it should be possible to control the life cycle of services and to handle their dynamicity by offering proper mechanisms that enable the adaptation of those systems that exploit these dynamic services. Adaptation usually is concerned with the possibility of replacing on the fly a service with a similar one that we should identify and select during the execution of the system.
- ***Scalability*.** SOA4All main objective is to provide a framework and an infrastructure that help to realize a world where billions of parties are exposing and consuming services via advanced Web technology. We are still far of this scenario, since SOA is largely an in-house enterprise-specific solution. However, it is not to predict that in the midterm as mobile devices and more efficient wireless communications facilitate ubiquitous computing; and as optical and broadband communication infrastructures expand, we expect the number of Web services to grow exponentially in the next few years. In near-term scenario imposes great scalability requirements on the overall SOA4All infrastructure. Therefore the composition, adaptation and execution framework should be either able to handle growing amounts of work in a graceful manner (definition of scalability given in [4]); or to be readily enlarged to cope with new workload on the fly (i.e. should be elastic).

3. General Architecture and Relationship with other SOA4All Components

Let us briefly describe the components of the Service Construction environment, so that we can put in context the analysis of the state of art that we are going to perform. Note that each of these components will be described and specified with great detail in forthcoming deliverables (which will be the outcomes of the tasks represented in Figure 1).

Users will use the user interface component to specify their required composite services and processes (part of the SOA4All Studio). Nevertheless, we need to define a graph-oriented lightweight process modelling language that will be used as specification language. To improve usability pre-designed and user-designed process templates are stored in the semantic service & template repository.

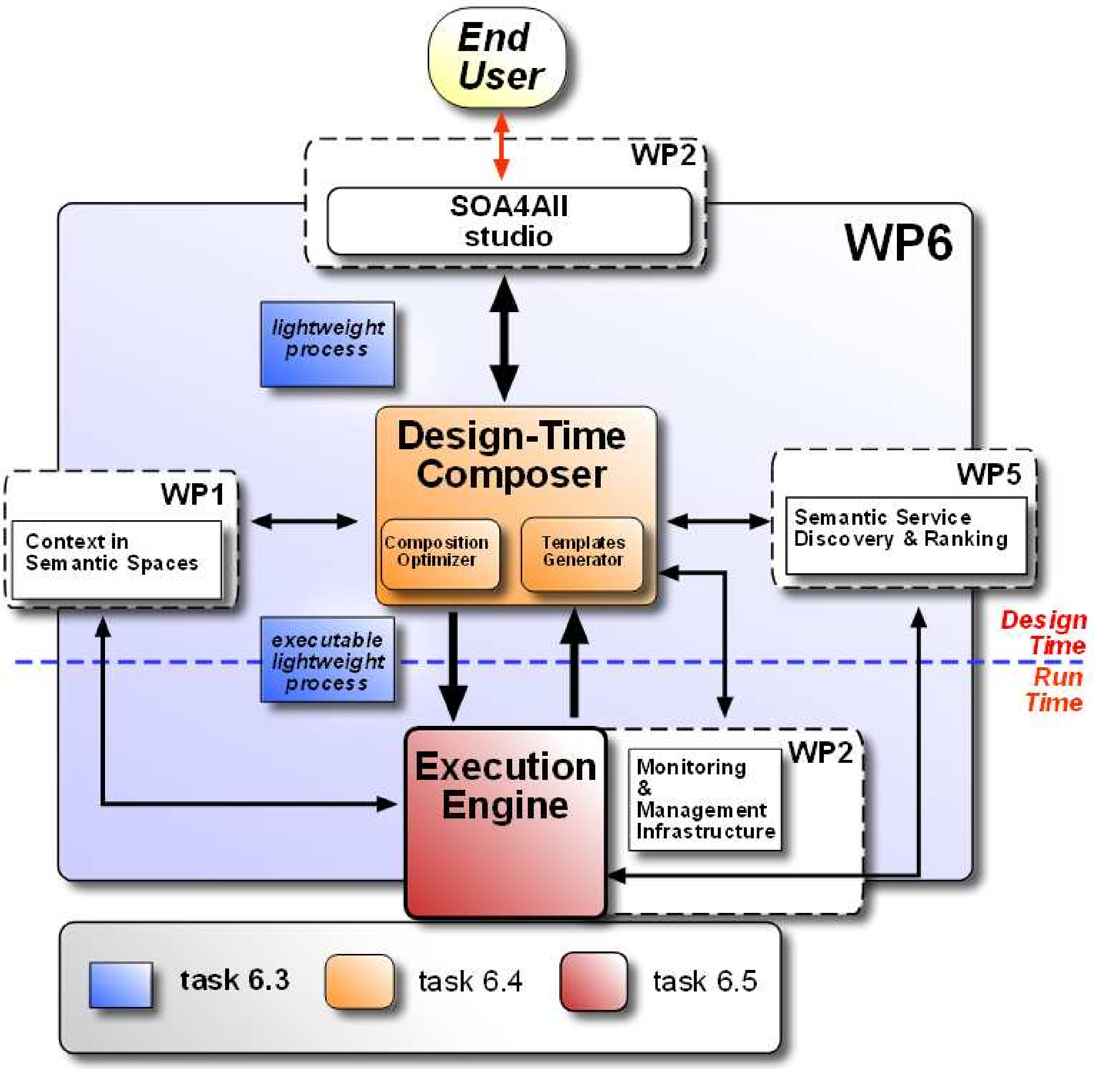

Figure 1: Service Construction Framework at a glimpse.

Once created and stored, in order to be used this lightweight processes have to be translated in to more complex processes that can be interpreted by an execution engine in an effective fashion. We will create a **scalable design-time composer for the flexible and ad-hoc creation and adaptation of complex services at design time**. The aforementioned lightweight processes will be transparently transformed in to optimized complex services orchestrations; or already existing complex services processes could be adapted to a specific use. These activities will be heavily influenced by the context in which they will be carried out.

Finally, regarding the runtime phase of service construction, the outcome of this work package will be the **execution engine**. It will execute complex processes that represent orchestration of services. This execution **will be adaptive to environmental changes; and flexible enough to allow its context-dependent self-reconfiguration.** This engine will consider also context during execution as well.

4. BPM and Executable Languages

4.1 Introduction

In this section, we will analyze different languages and approaches to BPM modelling, relating them with the set of requirements that we have already identified. We will start enumerating the requirements that we consider related to the descriptive and executable BPM languages. We then list and introduce the current approaches to BPM languages. After that, we will summarize the common features and differences between these languages as well as provide a list of the missing features. The missing features have to be addressed in the context of SOA4All.

4.2 Requirements for SOA4ALL

From the requirements that we have identified in the section 2.1, let us enumerate those that are more related with the modelling and description of processes in SOA4All.

One of the objectives of SOA4All is to open the world of service and process composition to the non-technical user. Depending on the user's skills and knowledge, we should allow the user to model its services and processes on different levels of complexity. Hence, SOA4All seeks to integrate all kinds of experts and non-experts. Graphical modelling elements will facilitate the process composition. Furthermore, every modelling element will contain metadata allowing for the easy integration into other processes.

The proposed SOA4All approach will use a template-based service and process composition approach where the templates are represented by lightweight service and process models. This approach seems to fit the problem class of mapping a kind of natural language to a diagrammatic representation. However, the complexity of service interactions cannot be fully represented by a simple representation. Examples of the mentioned complexity are the issues of state and transaction control.

Our envisioned approach is to abstract these complex aspects and hide them from users. However understanding the hidden aspects is often crucial for understanding the overall behaviour of the system. Mehandjiev and Bottaci [5] have proposed the use of assumption descriptions that are explicit textual statements of how missing aspects are implemented by the system. However the mainstream approach seems to rely on choosing appropriate metaphors for the representation.

The problem with the metaphor-based solution is that some of this "common sense" knowledge seems to depend on annotations, the background and other characteristics of the target user group. We would therefore need to proceed with context-driven techniques like user profiling techniques and systematic, user-aware design of our process and service composition representations. This can be done by using formative and summative evaluation techniques as appropriate.

The service and process models will be used to create representations meeting both the business and technical needs of users and their use cases. Achieving the objectives of SOA4All requires the delivery of a service and process composition interface, the SOA4All process editor, considering the skills and tasks of our target users. The long-term aim of SOA4All is to open up process modelling to everyone, yet at first instance our target users are those found in the SOA4ll case studies (WP7-9).

The case studies are still in the stage of initial definition, yet the following characteristics of our end users are clear:

- Most target users will have professional background

- Some target users will not be professional software developers and would not have received significant training in programming nor system design
- Most target users will be experts in the tasks and processes they are trying to support by using SOA4All

4.3 Overview of current approaches

In this section we will start describing the academic initiatives to process description. Then we will introduce industrial approaches to business process modelling by using standards, e.g. the Business Process Modelling Notation (BPMN), the Business Process Execution Language (BPEL), the Unified Modelling Language (UML), the Object Process Methodology (OPM), and the Web Services Choreography Description Language (WS-CDL).

4.3.1 Academic Languages and Systems

Whilst a large number of visual representations for service composition and interaction have been proposed (e.g. [7]), only a few of them have been evaluated in terms of usability and cognitive effectiveness. For example, Lets Dance [6] has been created using the framework of Cognitive Dimensions as a set of guiding principles, but iterative testing and enhancement have not been documented in the related references.

Vitabal WS [8] is a version of an earlier visual language tuned to the needs of web service composition. It has been evaluated using the cognitive dimensions framework, yet it targets experienced web service developers and hence would have different characteristics from the service composition representations to be developed by SOA4All.

Yet Another Workflow Language (YAWL) [16] is a business process modelling language founded on the workflow patterns [17]. YAWL has a formal semantics specified in form of a label transition system, allowing it to exploit fundamental properties from Petri nets[5]. It has also a graphic syntax which supports higher-level modelling activities. The YAWL language is implemented in the YAWL system [18], an open-source reference implementation of a workflow engine. It has an associated editor which allows process specification to be created and modified as well as an operational environment, of which the workflow engine is a part, together with facilities such as a worklist handler that supports user interaction with the engine during process execution, a Web services integration module and a graphical forms manager.

4.3.2 BPMN

Business Process Modelling Notation (BPMN) [15] is a business process modelling language. The main purpose of BPMN models is to facilitate communication between domain analysts as well as strategic decision-making based on techniques such as cost analysis, scenario analysis and simulation ([13], [14]). BPMN models are also used as a basis for specifying software system requirements and providing input to software development projects. It was developed by Business Process Management Initiative (BPMI) and is currently maintained by the Object Management Group (OMG).

The process of modelling in BPMN is performed with a small set of graphical elements, particularly: flow objects, connecting objects, swimlanes, and artefacts, which enables the stakeholders to construct a simple business process diagram (BPD). A Business Process Diagram defines a pattern as a series of activities connected by sequence flow. The

[5] newYAWL is based on Coloured Petri Net (Russell et al., 2007 [95])

graphical elements include the following:

- Flow Objects: events, activities, gateways.
- Connecting Objects: sequence flow, message flow, association.
- Swimlanes: pool, lane.
- Artifacts: data objects, group, annotation.

The BPMN diagrams convey a wide range of information to different viewers and effectively models end to end business processes. Examples of the business processes that can be modelled using BPNM include: high-level private process activities, as-is or old business process, to-be or new business process, detailed private business process, and two or more detailed private business processes interacting

BPMN is supported by more than 30 tools (see www.bpmn.org). One of them is the Eclipse SOA Tools Platform which offers a fully featured open-source BPMN editor available from download at http://www.eclipse.org/stp/bpmn/

4.3.3 BPEL

WS-BPEL (short for BPEL) is a Web service composition language originally proposed by BEA, IBM and Microsoft. In July 2002, the first version of BPEL was published [12]. Subsequently, SAP and Siebel joined the effort and the second version of BPEL [11] was published in May 2003. There are also other versions of BPEL: Websphere Integration Developer version 6.0 (informally WebSphere BPEL), Oracle BPEL and so on. The latest version of BPEL has been described by [10]. Many major vendors of business solutions have joined the Web Services Business Process Execution Language Technical Committee (WSBPEL TC), including Adobe, Hewlett-Packard, NEC, Oracle and Sun.

BPEL is also known as a Web services flow language, Web service execution language, Web service composition language, Web service orchestration language and web-enabled workflow language. Web services composition languages such as BPEL is build directly on top of Web Service Description Language (WSDL). BPEL can provide and/or use one or more WSDL services. A web service composition language can glue composed services together into a process model. BPEL provides the means to specify such a process model. An important difference between WSDL and a Web service composition language is revealed when considering the states. WSDL is in essence stateless while Web service composition languages such as BPEL records states for processes.

BPEL combines the features of a block structure language inherited from XLANG with those for directed graphs originating from WSFL [9]. The language should support the modelling of two types of processes, executable and abstract processes. An abstract, (not executable) process is a business protocol, specifying the message exchange behaviour between different parties without revealing the internal behaviour for any one of them. An executable process specifies the execution order between a number of activities constituting the process, the partners involved in the process, the messages exchanged between these partners, and the fault and exception handling specifying the behaviour in case of errors and exceptions.

There are additional other **BPEL related platforms**, namely the Eclipse STP BPMN Diagram Editor, the Orchestra Fully Open source, the ActiveBPEL, the BPEL Resource Guide, the Service Interaction Patterns (with BPMN diagrams that match BPEL code samples), the

Open Source BPMS (Eclipse and Apache-based), the Apache ODE[6], Open source BPEL server, the Netbeans Enterprise Pack, and the BPEL for Windows Workflow Foundation.

Finally, widespread transformation tools address the **Transformation from BPMN to BPEL**[7]

- BPMN2BPEL: A tool for translating process models represented in BPMN into process definitions represented in BPEL:
  The distribution of the BPMN2BPEL tool does not include a graphical editor for BPMN modelling. However, a separate tool[8] is available and can export BPMN models defined in the SOA Tools Platform into the format required by BPMN2BPEL.
- SPM2BPEL (superseded by BPMN2BPEL): A tool for translating process models represented in the SPM (Standard Process Models) language into process definitions in BPEL (Business Process Execution Language). The SPM language corresponds to a core subset of UML Activity Diagrams, BPMN and XPDL composed of activity nodes, AND-splits, AND-joins, XOR-splits, XOR-joins, initial and final nodes, and edges connecting these nodes. A quick manual explains the usage of the translation tool as well as the XML format used for SPM-process definitions.

4.3.4 Unified/Universal Modelling Language (UML)

UML is a standardized visual modelling language used to create an abstract model of a system, that we can also use for modelling business processes. The Object Management Group (OMG) defines UML components and their intended semantics, but we can easily extend it through the customisation of profiles and stereotypes.

A system's model is composed of a set of diagrams (graphical representations) and a semantic backplane. The UML diagrams represent three different views of a system model:

- Functional requirements view (e.g. Use case diagrams)
- Static structural view (e.g. class diagrams and composite structure diagrams)
- Dynamic behaviour view (e.g. sequence diagrams and activity diagrams)

4.3.5 Web Services Choreography Description Language (WS-CDL)

WS-CDL [108] s an XML-based non-executable language which represents global business processes. It is aimed to describe peer to peer collaborations of multiple business participants that interact together to achieve a common business goal [2]. The chorography specification is devoted to compose interoperable, peer-to-peer collaborations between participants regardless of their platform or programming model used by the implementation of the hosting environment.

The WS-CDL model is composed of the multiple entities, namely a Role Type, Relationship Type and Participant Type, an Information Type, Variable and Token, a Choreography, a Channel Type, a Work Unit, Actitivies and Ordering Structures, an Interaction Activity, and Semantics.

4.3.6 SCA with a Process Definition Pattern

The Service Component Architecture is a set of specifications for architectural descriptions

[6] See http://ode.apache.org

[7] See http://www.bpm.fit.qut.edu.au/projects/babel/tools/

[8]http://oomethod.dsic.upv.es/labs/index.php?option=com_content&task=view&id=14&Itemid=35

and runtime execution of SOA systems. Although SCA covers a wide array of aspects for assembling, configuring and executing services and service compositions, we choose to focus on reduced subset of the standard, which is relevant in the context of this section. Thus we consider only the assembly part of the SCA specification and we ignore the rest, most notably the runtime and technological aspects. These other aspects become relevant when generating executable processes which we cover in the section 6 of this deliverable.

There may be several approaches to represent a business process in an architectural diagram. An approach that seems to some extend accepted by the community is to generate on SCA composite for one business process. Inside this composite, one can find the services corresponding to the steps defined by the process. In addition to the components for these services, an extra component must provide the orchestration capabilities and offers the composite functionality. Therefore, this component will be promoted to the composite level and it will have references (dependencies) to the actual services needed by the process.

4.4 Comparison

The following table summarises and compares the main business and process modelling and executable languages:

Table 1: Comparison of current approaches

Measure	BPMN	UML	BPML	WS-CDL	SCA
Representation	Graphical (Visual notation of 2d symbols)	Graphical	Textual (text based notation)	Textual	Graphical
XML based language	No	No	Yes	Yes	Yes
Domain	Specific purpose	General purpose	Specific purpose	Specific purpose	Specific purpose
Language	Non-executable	Non-executable	Executable	Non-executable, Not an implementation language	Executable
Approach			Orchestration	Choreography	Orchestration
Extensibility			Yes	Yes	Yes
Semantics	Lack precise semantics		Precise execution semantics		

4.5 List of missing unfulfilled requirements

Existing BPM languages are driven by the characteristics of service technology and the needs of software system professionals. They are usually validated for representational power [20] using ontological frameworks such as the Bunge-Wand-Weber representation model [21]. These evaluations are objective in that they do not take into account the needs of non-technical users. One notable exception is Recker [19], which takes into account differences in stakeholder needs, yet this evaluation is still based on ontological completeness criteria rather than cognitive characteristics and needs of the users.

In SOA4All, we will combine representational evaluation of the lightweight process modelling language with evaluations using annotations and context-driven principals. We thus validate our claims for user-friendliness of the representations and tools developed.

The following main stages encapsulate the process SOA4All can follow to create effective and usable representations and environments to be used in user-driven service composition:

- Identify our end users and the use cases they will be involved
- Identify the IT skills and knowledge of our users.
- Evaluate existing languages, methodologies and tools
- Define graphical and textual representations as well as service interaction techniques
- Validate the effectiveness of representations and service interaction techniques
- Define the transformation of the graphical and textual process models into a format that SOA4All tools can execute.

We will capture user requirements using a set of techniques that involve:

- Interviews and Observation (Initial part)
- Questionnaires and Brainstorming sessions (Middle part)
- Use cases, Scenarios, and Prototyping (Late part)

5. Service Composition, Service Adaptation

5.1 Introduction

First, we want to remark that in this section, when we refer to service composition and service adaptation we are actually referring to (semi) automatic service composition and adaptation, not manual industrially adopted approaches. A BPEL editor is a valid and useful service composition tool, but it requires intensive human intervention.

Nowadays, services automate and execute many of the business process related tasks are being automated and executed by services, which must be adapted to particular needs, and hence configured on a per case basis. Furthermore, various independent services have to be interlinked and composed in order to satisfy the various global business goals. As we want to automate the configuration, and even worse the composition of services is a hard problem, particularly when moving towards the heterogeneous, unreliable and open Web. Furthermore, composite services are likely to be much more multifaceted, as serially executable services, or have to be re-configured and adapted for various specific uses at a higher level of complexity. Nowadays, composed services are mostly configured by means of pre/post-condition-style reasoning, resulting in planning-style approaches to service configuration ([51], [43]), which basically means that the compositions have to be done from scratch each time the composite service is applied. In other words, planning-based composition applies semantic descriptions of atomic services in terms of functionality and interfaces, and the descriptions of the required composite service (the overall goal) to construct a “plan” that determines how the individual atomic services are integrated in order to obtain a solution to the overall objective.

However, in the context of business process and business-to-business coordination we can observe that a company’s legal obligations or the available technologies often force enterprises to organize their computer systems in very similar ways. Therefore, in the given context, system vendors as well as consultants typically offer generic reference process models with their solutions. These are typically defined on a conceptual level, and help to understand how business processes are supported by the particular systems. The benefits of such reference models compared to modelling of business processes from scratch are reduced modelling costs and an increasing use of proven or common practices.

Therefore, in the context of the SOA4All project we aim at more lightweight approaches to service configuration and service composition and use so-called templates to do so. Similarly to the work of ten Teije et al [47], we aim for a knowledge-driven approach. Problem-Solving Methods, in combination with specialized knowledge allow for reasonably solvable problems, even so the composition challenge is observably hard (or even unsolvable) in general. In SOA4All, we propose to describe workflows (or processes) analogously to the generic process models and thus complex composite services with pre-defined and fixed process templates. Templates are skeletons that describe the general pattern of a composite service, but that leave enough freedom to configure and adapt the composition to the needs of specific executions [116]. In other words, templates are predefined “plans” with gaps, and the composition is an instantiation of a plan, rather than a reconstruction from scratch. Process templates are specific to particular goals or problems (e.g. reporting, recruitment, call-center processes). On a lower level of abstraction, but with higher granularity, they apply service templates to describe and classify atomic services. These templates allow the composition engine to configure the inputs and outputs of each individual service separately, in order to optimize the functionality and performance of the composite service.

Whereas in the traditional sense the main state of the art approaches perform planning, the SOA4All approach will rather be based on the principle of service configuration (i.e. reasoning with specialized knowledge in a narrow domain). While a planner is by definition domain independent, and is thus thought to work on any set of (atomic) services, given simply their descriptions, a composition engine à la SOA4All rather exploits specific knowledge about the services it is currently dealing with. This does of course not imply that template-based solutions to service composition are not applicable in the general sense, but only that the available knowledge must be sound and complete with respect to the application domain; there are differences to the composition task in the context of reporting compared to e.g. recruitment. This is the reason why configuration design is seen to be a more knowledge-driven approach than planning.

Consequently, service configuration in the sense of [47] and SOA4All can be modelled as parametric design, in which the parameters, i.e. the gaps, of the template have to be instantiated with appropriate input values in the context of service templates, or the matching atomic services, in the context of process templates. During the configuration process, the composition engine then exploit detailed knowledge about the template and the available services in order to establish the desired composite service. This approach, as [48] shows, is based on well-established work from Knowledge Engineering, and results obtained there in the 90's. Knowledge Engineering has extensively studied the notion of reusable components for knowledge-based systems, in particular reusable problem-solving methods ([48]; [24]; [96]).

Vendors of enterprise or workflow systems as well as consultants typically offer generic reference process models such as process templates, together with their solutions. Typical examples for such business processes are purchasing, reporting, recruitment, CRM, payroll, or call-center processes. These are typically defined on a conceptual level and help understand how business processes are supported by the particular systems. The benefits of reusing process templates compared to modelling business processes from scratch are widely accepted ([53]; [54]; [55]). The systematic adoption of process templates can reduce the modelling costs and increase the use of proven or common practices.

Secondary business processes will rarely be organized in exactly the same way among users. Instead, minor or sometimes even major adaptations are required to tailor the process to the individual customers. To support these different environments, larger enterprise systems like SAP WebFlow [56] often offer more than one way to execute a business process. The selection of the used variant must be made only during the implementation of the process/system.

However, process templates in commercial use tend to be captured in natural language (e.g. ITIL), or in existing general modelling languages such as EPCs, BPML, Protos, SAP WebFlow, etc. These languages do not provide any dedicated support for the different process variants.

The idea of using, and very importantly reusing, predefined templates for service configuration and composition is thus not new. It moreover also appears in other work and fields: the notion of "generic procedures" in [37], the instantiation of predefined BPEL process models [35], and the coordination patterns from [50] are similar examples that provide further indications for successful enrolment of configuration-based service composition. Further related activities also exist in the parallel programming community, where so-called skeletons (i.e. templates) are applied to box up useful patterns of parallel computation and interaction that can be applied across implementations and analysis procedures. The skeletons only provide the structures, but not the details of the computations and interactions, and their instantiation and composition (reuse of ready-made components) is up to the developer.

There are existing business reference models (e.g., the XML Common Business Library, xCBL, RosettaNet or SAP' reference models). Business functions can be derived on their basis. These reference models are important resources of our process templates. The problem is how to organize them. End-users should be able to seamlessly discover and select services on the basis of their requirements and context. User should be allowed to use them in a computer-aided fashion; and these processes should be (semi) automatically adapted or integrated into the whole system.

Process mining can be seen as a way to abstract from past task instances (i.e. logs generated by ERP, CRM, SCM, WfM systems), to induct a new schema, previously unknown, describing all of them or to refine certain models. The schema will contain information such as process and activities structure, durations, resources, constraints and business policies

This methodology can help in many complex organisational situations, where the effort required to explicitly design a process schema (i.e. via graphical tools) is too high, or it is not even known the real schema of the underlying processes. Process mining helps to reduce such effort, by providing automatically a schema that can be then further refined and optimised by end-users in order to match the real structure of the process.

In this way, Process Mining is a powerful additional feature to be added to the “Process Template Generator” component to be developed within WP6.

In summary, the idea of template-based service compositions is to first - at the level of the templates - decompose the overall functionality desired into sub-functionalities that are simpler to process and that thus ease the finding of an optimized service orchestration, which in turn leads to a solution to the initially complex business problem. More technically spoken, SOA4All suggests to statically decomposing a user goal into a sequence of sub-goals through templates, instead of requesting at runtime the discovery of a set of atomic sub-services and combining them properly to achieve the goal. During runtime the compositions is hence simplified to searching for particular atomic services with given characteristics out of a service pool, and no longer relying on querying and browsing arbitrary sets of services. Moreover, the templates inherently provide a rough idea about the service interface and functionality (at the level of service templates) or the sequence of services (at the level of process templates). This eases the work of the user and developers significantly – or more drastically said, that allow SOA4All users to understand the composition process at a much higher level of abstraction; i.e. without digging into the technical problems of service composition.

5.2 Requirements for SOA4ALL

From the set of requirements that were described in the catalogue at the beginning of this document (section 2.1), we extract the following requirements that we believe that are more relevant to the automatic service composition and adaptation field.

- ***Usability by non-modelling experts*.** The process of configuring a process template to the demands of an organization requires users to have a thorough understanding of both the domain and the modelling language which is an unrealistic assumption.
- ***Executability*.** The compositions generated by our tool should be executable. Current process templates are designed as conceptual models in order to facilitate discussions in the design time of a BPM system. As such, there are intuitive suggestions rather than a precise and concise specification of an enactment. In SOA4All, we try to build a bridge between process templates and service templates for eventually executing the BPM.
- ***Efficiency***. The generation of a complex process template from the requirements of the user should be done as diligently as possible. The task of composing processes is transparent to the user. Moreover, end users are not aware of the complexity of the underneath composition process, and therefore they expect their result instantly. We need to bust efficiency; and we believe that the key enablers for efficiency achievement

will be the use of the extra knowledge about the services and its domain of appliance. This is why we plan to use of preconfigured heuristics and templates in order to reduce the search space when composing services.

In summary, SOA4All should seek methods for automatically transforming business process models in configured processes that enact composite services. These templates are then instantiated by use of knowledge intensive problem solving methods, which allows both, the reusability of pre-configurations of known tasks, and dynamic and adaptive composition on a per execution basis.

5.3 Overview of current approaches

5.3.1 Current approaches for process template configuration

A process template combines a family of similar process models. It is designed in a generic manner and is intended to be configured to fit the requirements of specific users and applications. In this subsection, we review current techniques for representing configurable process models.

To capture variability, an approach has been explored in ([60]; [57]) that annotates model fragments of process templates with boolean conditions and that removes those fragments whenever the conditions evaluate to false. In [60] the authors extended UML Activity Diagrams (ADs) and Business Process Modelling Notation (BPMN) diagrams with stereotypes to accommodate variability points. A variability point is aligned to a feature and is evaluated with respect to a feature configuration (e.g. to activate/deactivate model elements). In [57] UML ADs are annotated using presence conditions and meta-expressions, which are then linked to elements of a feature diagram. Presence conditions indicate if the model element they refer to should be present in the model. Meta-expressions are used to compute attributes of model elements (e.g. name, return type). The approach only supports simple mapping of features to standard variability mechanisms provided by UML.

Another approach to capture variability in process templates is represented by Configurable EPCs (C-EPCs) [59]. C-EPCs extend the Event-driven Process Chains (EPC) notation by identifying a set of configurable nodes in the model, to which alternatives are assigned to restrict their behaviour. Once all the configurable nodes are assigned with an alternative that complies with given requirements, the C-EPC is transformed into a syntactically correct EPC.

A questionnaire-driven configuration of reference process models has been proposed by La Rosa et al. [58]. The approach is based on the representation of choices and their dependencies. Facts represent answers; the questions are lined to variation point in reference models. Questions are expressed in natural language and can be answered by domain experts without extensive knowledge of the underlying reference model. The user thus does not have to directly deal with the reference model anymore. The major assumption is that questions have a finite or discretized domain of possible answers. This assumption provides the models to be efficiently analyzed in order to avoid the user from entering conflicting responses to subsequent questions. The toolset has also been implemented and can be downloaded from[9].

5.3.2 Automatic Service Composition Technologies

In this section we give a brief description of each of the approaches of the state of the art that we found to be relevant to the work that we are going to carry out in the composition of services in the context of SOA4All. We have considered several categories.

[9] http://www.processconfiguration.com/download.html

One of the first approaches to Web services composition was carried out proposing a **Golog adaptation**. Golog is a programming language whose semantics is decomposed via macro-expansion into sentences of the situation calculus [37]. The situation calculus was defined by [36] as a first order language. It was created to represent precisely dynamically changing worlds. Thus it is action-centric. A set of actions over an implicitly defined model of the world is referred to as situation. [37] considered Web services as Golog complex actions. They introduced new generic procedures in to Golog so that these actions became easier to use, generic, customizable, and that were usable by a variety of users under varying conditions. The process of composing services is achieved by relaxing the order of execution of actions. Their implementation, which relies on the Golog interpreter permits the interleaved execution of additional actions between two consecutive actions in order to satisfy the preconditions of the former action.

Another approach for service composition is rule-based composition.

SWORD [38] is a developer toolkit for building composite Web services using rule-based plan generation. SWORD uses the Entity-Relation (ER) model to specify the inputs and outputs of these services. Services are associated and thus described using actions, which are modelled by its preconditions and postconditions. SWORD uses a classical rule-based expert system in order to compose the service. The rules are of the form

IF precondition THEN do some action that implies postcondition

SWORD considers the composition of services as a process of forward rule chaining, which input is the initial state of affairs and the output is the composite service (embodied as the final state after the appliance of the selected rules actions to the initial state).

5.3.2.1 Planning Based Composition

Planning-based approaches to Web services composition exploit the similarity between a process of service composition and the process of creating a plan of the classic AI planning problem.

SHOP2 [45] is a Hierarchical Task Network (HTN) planner. HTN based planning [46] uses knowledge about the networks of dependencies among the actions of a given domain. Authors refer to such actions as primitive tasks; compound tasks, which can be seen as composed of a set of simpler tasks; or goal tasks. Compound and goal tasks both require a sequence of primitive actions to be performed; however, goal tasks are specified in terms of conditions that have to be made true, while compound tasks can only be specified in terms of other tasks via the task network.

The main assumption made in ([45], [51]) is that the afore presented type of HTN complex actions is very similar to the concept of composite process decomposition in the OWL-S process ontology. Thus they translate semantic Web services described in OWL-S. Each atomic process with effects is translated as a SHOP2 operator that simulates the effects of the world-altering Web service. OWL-S atomic processes with output are encoded as SHOP2 operators whose precondition includes a call to the information-providing Web service. Finally for each OWL- S Simple or Composite Process one or more SHOP2 methods are created. These methods will specify the decomposition of an HTN task that represents the simple or composite process.

[49] propose the use of a Model Based Planner (**MBP)** [23] for Web service composition. MBP is based on planning as a model checking approach, and performs an experimental evaluation. According to Traverso and Pistore, MDB planners deal with the nondeterministic behaviour of Web services, the partial observability of their internal status, and with complex and changeling environments. They translate OWL-S process models in to a MBP planner understandable domain, and this planner takes into account these issues. The generated

plans are later on translated into executable processes, such as BPEL4WS programs.

The input to the service composer thus becomes the composition goal (a set of conditions over the expected result service expressed in the goal specification language EAGLE [27]; and the planning domain (which is the set of available Semantic Web services formalized as STSs). MBP uses a Symbolic Model Checking planning technique (described with detail in [29]). Planning is done by searching through a finite state automaton whose sets of states are represented symbolically as propositional formulae (that in our case represent the formalization of all the possible services compositions). Search through the state space is performed as a set of logical transformations over propositional formulae. All these formulas are represented as Reduced Ordered Binary Decision Diagrams [25] that allow their compact representation and their effective manipulation.

This approach demonstrates also how the explicit and formal description of services performed at the high level of OWL-S process models is orders of magnitudes more efficient than the one applied at the low level of executable processes, thus demonstrating experimentally a practical advantage of the Semantic Web approach to Web services.

5.3.2.2 Workflow related composition solutions

Apart from the techniques presented so far, more related with the AI field, in the workflow field there have been several initiatives [44] that we believe are valuable input for the work to be done in this work package. As presented in [40] there are two types of workflow composition, as we can consider both static and dynamic workflow generation. The static one means that the requester should build an abstract process model before the composition takes place. The abstract process model includes a set of tasks and their data dependencies; but the selection and binding of Web service that is finally invoked is done automatically by the composition environment. On the other hand, the dynamic composition both creates process model and selects atomic services automatically.

eFLOW [52] is a platform for specifying, enacting, and monitoring composite eServices. Composite services are modelled as business processes, enacted by a service process engine. eFlow provides a number of features that support service process specification and management, including a powerful yet simple service composition language, events and exception handling, ACID service-level transactions, security management, and monitoring tools. eFLOW performs static composition of workflows, the automation to bind the nodes with concrete services include dynamic service discovery, multiservice nodes, and generic nodes, which permit the multiple, and parallel invocation of instances of the same type of service.

eFLOW also allows modifications applied to a single, running process instance, being able to carry out two different types of modifications, which are process schema modification and on-the-fly process instance change.

Pegasus [28] is an approach to composing Web services for which users sketch a high-level outline of a composition and a mapping system automatically transforms and optimizes these high-level specifications based on the underlying execution [33]. Pegasus combines both static and dynamic workflow generation. The static workflow generation corresponds to the aforementioned mapping process from an abstract to a target architecture. The dynamic workflow generation is based upon the MAPE functional decomposition [32] which partitions adaptive functionality into four areas, Monitoring, Analysis, Planning and Execution. The MAPE functional decomposition is a useful framework for systematic development of adaptive systems.

The execution of a workflow enactment is monitored in order to detect relevant events at the assigned resources. In the adaptation strategy described in this paper, an executing workflow instance is monitored for the relevant events at the assigned resources. An analysis of these events is carried out in order to determine if a reconfiguration of the workflow is necessary. If this is the case, a planning process regenerates the service, updates the information available to Pegasus, and reruns the workflow on Pegasus. The revised plan for the work that remains to be done is compared with the current plan, and the new plan is adopted if it is predicted to give an improved overall response time. Changes to the workflow execution proposed by the planning process are implemented in an execution step that removes and replaces the executing workflow.

As a classical AI planning system, it receives as input the representation of the current state of its environment, a declarative representation of a goal state, and a library of operators that the planner can use to change the state. The planning process in Pegasus considers as goals the desired data products and the operators are the set of application components. Each operator has a description of the states in which it can be applicable, called preconditions, and a concise description of the changes to the state that will take place, called effects. The Pegasus planning system searches for a valid, partially ordered set of operators that will transform the current state into one that satisfies the goal. Each operator's parameters include the host where the component is to run, while the preconditions include constraints on feasible hosts and data dependencies on required input files. The plan returned corresponds to an executable workflow, which includes the assignment of components to specific resources that provide the requested data product when executed.

Polymorphic Process Model (PPM) [42] also combines both types of workflow composition, the static and dynamic ones. The dynamic composition of services in PPM is based upon the use of service-based processes. In such processes, services are encoded as state machine that specify the possible states of a service and its transitions. Transitions are caused by service activities invocations or internal service transitions. In the setting, the dynamic service composition is enabled by the reasoning based on state machine. Static composition is supported by the use of reference process-based multi-enterprise processes. Those processes are formed by abstract subprocesses that have functionality description but lack implementation (i.e. they are bound at runtime).

5.3.2.3 Configuration Based Composition

After describing the approaches that use classical AI techniques, or that consider service composition as planning we explore a new way of thinking about services composition in a knowledge-intensive fashion, which is Configuration Based Composition**.** To the contrary to planning-based approaches that start their composition from scratch, [50] describe a complex Web service as a fixed template that must be configured for each specific use. The process of composing a Web service is considered thus as a parametric design over the parameters of a fixed template. This configuration process is a knowledge-intensive task, since detailed knowledge about the template, its intended domain are used to obtain the required composite Web service. This process of configuring a service is based on the work made by ten Teije and colleagues in the configuration of Problem-Solving Methods [47], which roots in already existing reasoning method for parametric design, the Propose-Critique-Modify (PCM) [24].

5.3.3 Process Mining Technologies

Data mining techniques are usually strictly related to process mining problems: in fact, supposing a certain number of tasks within a process, the number of possible different executions is an exponential number (while not all of them have the same probability to be actually executed). Here is where data-mining technology helps.

[26], while working in the area of software engineering processes, analysed three different methods for process discovery: neural networks, Markovian models, and an algorithmic approach. Their work allows to generate explicit process models, and to measure the actual gap between process model and actual observed behaviour

[22] introduced the idea of extending process mining to workflow management, by analysing the events recorded in a log and by identifying constraints. Their approach is based on an algorithmic approach, by enumerating tasks instances and applying a folding procedure

E. M. Gold in [30] shows that the problem of finding a state-machine compatible with a set of recorded data is NP-hard. This problem has an analogy with the process mining problem, even if an important difference is that process mining needs to take into account concurrent tasks.

Another important research stream is the one linked with the capability of identifying uncertain variables (either due to low frequency of their observation either due to incomplete datasets). Silve, Zhang and Shanahan [44] propose a probabilistic workflow model and a learning algorithm that is capable to compute in a polynomial time. This model is based on directed acyclic graphs (DAG) where each node represents a task and the arrows represent dependency relationships, so that given the predecessor tasks, a task execution is independent from the other ones. The learning algorithm builds such graph by connecting the various tasks (nodes) on the basis of their joint instances in a log file.

Finally, other approaches of [31] are based on clustering techniques on data contained in log files. The solution is based on dynamic Bayesian networks or Petri stochastic networks. More in particular, factorial Hidden Markov Models are suitable for modelling parallel tasks.

The major limitations of these mining approaches when applied to workflow modelling are:

- process instances have a precise start and conclusion time
- there are synchronization constraints
- process past history is defined in a deterministic way
- a hidden state (i.e. a task not observed) is associated to just one observable symbol (the effect of that task in the process)

5.4 Comparison

Table 2: Comparison of template management and process generation approaches

Name	Based on configurable processes	Workflow based composition	Automatic composition	Template based approach	Based on process mining
Complexity of cases	**Simple cases**	**Simple cases**	**Complex cases**	**Complex cases**	**Complex cases**
Constraints	**Highly constrained**	**Highly constrained**	**Constrained by assumptions**	**High quality of domain specification required**	**Start and conclusion time, synchronization**
Use of parametrical templates	**X**	**X**	**X**	**X**	**X**

Service binding	**-**	**At runtime**	**At runtime**	**At runtime**	**At runtime**
Template management	**-**	**At design time**	**At runtime**	**At runtime**	**At runtime**
Assumptions	**-**	**-**	**X**	**-**	**-**
Conditional and iterative behaviours	**-**	**-**	**-**	**X**	**X**

X = yes, O = Partly Covered, - = Not Covered

Approaches based on the use of configurable processes, either using conventional software engineering techniques or AI techniques are quite useful for very simple cases, or those where a human can be involved in the whole design process. Nevertheless, they are quite constrained by its nature. They just allow the parameterization of certain aspects of the process, which may lead to very static solution.

Workflow based composition approaches go one-step further. They use also parametrical templates, which later on at runtime are bound to concrete services. Nevertheless, since the management of templates is placed at design time, conceptually for us they are very similar to the reconfigurable processes approach afore described. Some of these approaches, like Pegasus [31] include dynamic composition, in the sense that they can change the structure of the workflow. Nevertheless, this reconfiguration is usually quite simple and based on non-functional runtime based properties such as efficiency.

In the case of automatic composition of services, most of current approaches make the analogy of compositions as AI planning. A set operators define the planning domain, which are a parameterized representation of the transitions available in the domain. The problem with those approaches is that they usually make strong assumptions about the domain, which make them simplistic. For example, they do not usually cope with non-determinism and partial observability. Moreover, these approaches do not generate conditional and iterative behaviours. The only planning- based approach that takes this things into account is the work on model checking based planning carried out by [52].

The approach made by ten Teije has a main advantage over the exposed AI planning-based techniques. The intensive use that this technique does of domain knowledge combined for configuring already existing templates, makes the process generation more computationally feasible than the generic planning approach. Its main disadvantage with respect to the AI planning based approaches is precisely the need of a high quality and complex description of the domain of the services to be composed. As the authors recognize in [50] the costs of acquiring this knowledge may well be prohibitive in a web-service scenario.

The generation of process templates by means of process mining is a very promising field. In contrast with the other approaches that we presented so far, it is very useful when we cannot obtain by other means a formal description of the processes, situation that is quite common. Other approaches need a heavy knowledge acquisition process for formalizing the processes before the composition is made.

5.5 List of missing unfulfilled requirements

Using the comparation of the above section, we have now a clearer idea about the requirements that the exposed approaches does not fulfil. Let us briefly enumerate the possible discrepancies that we have encountered (emphasized in italic style in the table).

Table 3: List of unfulfilled requirements

Name	Covered by current approaches	To be covered by WP6?
Usability by non-modelling experts: The process of configuring a process template to the demands of an organization requires users to have a thorough understanding of both the domain and the modelling language the process template has been constructed in. It is an unrealistic assumption. There is not a clear separation of roles between an expert using the system and a user that probably will not be able to create complex processes. Indeed, the approaches afore exposed need an expert in the whole lifecycle of the service, both for its design and for running it.	**-**	**X**
Configurability for process modelling language. Existing general modelling languages have limitations for expressing dedicated supports for the different process variants. The SOA4All template management should also provide decision support for the actual selection of an alternative. The function, such as highlight difference between similar process templates should be included. The SOA4All platform should guide the users as to what might be a recommended configuration given the user's environment	**-**	**X**
Executability. current process templates are designed as conceptual models in order to facilitate discussions in the design time of a BPM system. As such, there are intuitive suggestions rather than a precise specification of requirements. In SOA4All, we try to build a bridge between process templates and service templates for eventually executing the BPM. Most of the approaches described and analysed above do not generate executable processes. It they do, the outcome is usually only restricted to a subset of the target language.	**O**	**X**
Efficiency. The generation of a complex process template from the requirements of the user should be done as diligently as possible. We believe that the key enablers for efficiency achievement will be the use of the extra knowledge about the services and its domain of appliance; and the use of preconfigured heuristics and templates in order to reduce the search space when composing services. Most of the approaches presented in the state of the art are slow, especially those that use AI planning as its composition mechanism. AI planning is a complex task mainly because it does not make extensive use the knowledge of the domain of the service, as we consider it knowledge independent. Moreover, each planning process starts from scratch each time a service needs to be composed.	**-**	**X**

X = yes, O = Partly Covered, - = Not Covered

6. Service Execution

This section addresses how services and processes can be executed. First we define requirements related to the service and process execution. The second subsection will describe current approaches before the third subsection will give a comparison of these approaches. Note that some of the content of this section, such as the description of WS-BPEL, might at a first glimpse seem redundant with the modelling section, where we already described it. The difference resides in the perspective, in this section we focus in the execution of services; in that section we focused in how to model complex services (i.e. run-time vs design-time).

As we have stated, in this section we will focus on one of the most promising current approaches to the execution of services orchestration, the Web Services Business Process Execution Language OASIS Standard WS-BPEL 2.0 [73], WS-BPEL (or BPEL for short) to execute an orchestration of services. Hence WS-BPEL is a language for specifying executable business processes based on Web Services. BPEL processes rely on Web Service interfaces exclusively.

Besides the execution using a process representation in WS-BPEL there exist other approaches to make processes executable. These approaches will also be described in section 6.2.

6.1 Requirements for SOA4ALL

6.1.1 Selection of requirements from Use Cases

From the requirements that we have identified in the section 2.1, let us enumerate those that are more related with service execution in SOA4All. There are a lot of functional and not functional requirements, explicitly claimed in use cases descriptions that need to be considered during process execution:

- Support for manually and automatically adapting specific services or entire processes to different consumption settings.
- Support services that are executed either by humans (e.g., “sign contract”) or computers (e.g., “calculate taxes”)
- Support services that are freely available on the Internet as well as enterprise services
    - provide different security, authentication and authorization mechanisms
    - support different interface descriptions
    - support handling of heavyweight enterprise services
- Registry Integration: While at design time users can narrow down the list of available services for a particular task used in a process template based on their descriptions, at runtime the SOA4All execution infrastructure must provide functionality for dynamic selection of execution targets based on environmental and contextual conditions. Based on previously defined sets of possible target services for each step, the process engine must be able to switch dynamically the invocation target in order to provide the optimal execution flow. Therefore, SOA4All must integrate with and use information from runtime service registries (such as those based on UDDI).
- Managing Quality of Service (QoS) and Service Level Agreements (SLAs) in composed services
- Trust and Security

  - Security: as usual, eCommerce applications deal with sensitive data - both payment and customer details are submitted to an application, which has been directly generated by an end user of the chillydomains platform. Thus, either additional services have to be added to the applications, which deal with authorization and payment issues, or services need to be wrapped or extended by dedicated security functionalities.
  - Privacy and trust model: In a similar vein, trust and privacy issues are important for all relevant actors. Specifically in the case of third party service providers, the end user has to be certain that both his and his customer's data will be dealt with accordingly.
- Scalability: the chillydomains Web hosting platform has a fast growing user basis (currently there are more than 12.000 customers; this user basis is growing by 7% per month). In addition, there is a potentially large number of applicable services for the creation of arbitrary eCommerce applications. While we expect to constrain the customer's option at first – only a defined set of services in combination with the current products will be available – this can grow quickly in future.
- Intuitive composition of services: As described before, chillydomains customers are generally not IT experts and will expect tools that are convenient to use. These tools should produce workable solutions without a steep learning curve or the need for additional programming and configuration.

6.1.2 Other non functional requirements

Another non functional requirement not explicitly claimed in the use cases description is adaptability. According to Baresi, Di Nitto and Ghezzi [74], "The need for software that can continuously evolve in an open world is reaching unprecedented levels. Existing approaches to software development can't cope with these new challenges. Consequently, we must explore new research directions. The more we move toward dynamic and heterogeneous systems, and the more we stress their self-healing and self-adapting capabilities, the more we need new approaches to develop these applications and new ways to structure and program them."

6.2 Overview of current approaches

In this section we will focus on the most promising approaches about the management of the run-time phase of a service composition. In particular we will show some adaptation mechanisms that allow a service composition to evolve and adapt itself during the execution. In addition we present an approach for generating and refining execution artifacts from design-time elements, positioned at the border between design and execution. By itself it does not deal with the actual execution; however it provides important mechanisms for using design information to obtain execution and deployment data, which can be used during execution.

6.2.1 Stable approaches executing standard BPEL

The ActiveBPEL Community Edition Engine [76] is an open source implementation of a BPEL engine and its development environment.

The **Active BPEL designer** [76] is an Eclipse-based development environment that supports the OASIS Web Services Business Process Execution Language v2.0 (WS-BPEL 2.0) standard and the BPEL4People and WS-Human Task specifications. Active Endpoints is an author of both the WS-Human Task and BPEL4People specifications.

In addition to fully supporting the WS-BPEL 2.0 specification, ActiveBPEL supports some BPEL extensions:

- Compensation handler and termination handler.
- Query handling including Create XPath and Disable Selection Failure;
- activeBPEL extension activities (Suspend, Break, and Continue) and custom XPath Functions.

PXE[10] is a modular, extensible, embeddable BPEL execution engine implemented in Java. The engine is built around a JMX microkernel and includes a BPEL object model, JMX instrumentation, pluggable persistence, and a set of commandline and Ant tools.

The execution engine is built around a specialized virtual machine that can run BPEL4WS 1.1, WS-BPEL 2.0, and other orchestration languages side-by-side.

Oracle BPEL Process Manager[11] is a BPEL engine that is a member of the Oracle Fusion Middleware family of products. It enables enterprises to orchestrate disparate applications and Web services into business processes. In Oracle BPM a rule engine can be invoked as a Decision Service from a BPEL process. Rules executed by the rule engine can perform dynamic processing of intelligent routing decisions, validation of policies, constraint checks, policy based task assignment, load balancing of tasks execution, and activity monitoring.

BEA AquaLogic BPM Suite is a complete product suite for creating, executing and optimizing business processes. BEA supports BPMN, BPEL, XPDL and provide a process templates repository for capturing process best practices and encouraging reuse across different BPM projects. BEA AquaLogic allows providers of external services used by business processes to dynamically redirect traffic and move services around. Binding information is kept and read from the service registry. This ensures loose coupling between process and service layers and results in increased agility for both business and IT.

Apache ODE (Orchestration Director Engine)[12] executes business processes written following the WS-BPEL standard. It talks to Web services, sending and receiving messages, handling data manipulation and error recovery as described by the process definition. It supports both long and short living process executions.

Apache ODE extends the standard WS-BPEL in several ways:

- Implicit Correlations: a client interacting with the BPEL engine must identify the particular instance with which it intends to interact in all of its communications;
- Activity Failure and Recovery: Apache ODE introduces a new class of error condition called failures, distinct from faults. Failures does not affect the normal flow of the process;
- XPath Extensions: ODE supports for XPath 2.0 and offer a few utility extension functions;
- External Variables: may be records stored in a database, REST resources, etc.;
- Headers Handling: in SOAP, message parts are the SOAP body and the SOAP header; ODE allow access to the header part of a SOAP message.
- RESTful BPEL: ODE extends the invoke activity to handle RESTful Web services and extends receive and onEvent activities to expose RESTful resources.

[10] http://swik.net/FiveSight-PXE/

[11] http://www.oracle.com/technology/products/ias/bpel/index.html

[12] http://ode.apache.org/

6.2.2 Approaches not exclusively related to BPEL

This section describes different approaches in the area of service and process execution that are not exclusively related to BPEL. Besides the following sections an overview of the state of the art can be found in [118].

6.2.2.1 Approaches for dynamic binding and re-binding

An approach aiming at the automation of dynamic service binding [77] consists in the proposal of a language for the specification of the requests, and an ontology based language for the description of service offers, which also include 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.

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. Adaptivity is very important in highly dynamical environments, such as the context of mobile network devices, where location-based applications are widely used. In [77], the authors present a middleware, named Atlas, to enable adaptive binding. Atlas continuously tests if a given service is no longer available (for instance, because the user leaves a certain region) and it autonomously binds to a new service. Atlas continuously looks for new services matching the application requests, and it includes adapters that translate the interface of services offered by the provider into an Atlas interface. Thus is transparent 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 [111], a new binding is only built when a user enters a different region.

In the **PAWS framework** [78] starting from an abstract process definition, a selection of candidate services is performed using a semantically enhanced registry. The registry stores mapping information to be used for mediation at run time, and negotiating QoS levels with candidate service providers. 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 is composed of three modules:

- A Process Optimizer: in charge of guaranteeing both local and global QoS, according to the constraints required by the user;
- A Self-healing module: performing semi-automatic actions in reaction to failures;
- A Mediation engine: set up at design-time, redirects the invocations of the deployed process to the selected services.

Another approach aiming at the automation of adaptive dynamic service binding is **WS Binder** [61]. WS Binder is a framework that enables dynamic binding of service compositions. The framework offers the possibility of performing dynamic binding writing customized set of rules that govern the binding itself. With specific set of rules WS Binder is able to provide different binding mechanisms:

- Pre–execution workflow global binding, which aims at determining the (sub) optimal set of bindings that satisfies functional and non-functional constraints through 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 influenced by setting up user 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.

WS Binder offers a management interface that allows to display the following data:

- The paths followed while executing composite service
- The set of bindings determined both at pre-execution and at run-time.
- The new bindings in case a re-binding action has been performed.
- Graphs visualizing QoS values before execution (estimated), before re-binding (i.e., the value that has caused the re-binding), after re-binding and when the execution is complete.
- Information about the availability of services used during the execution.

All execution data are obtained through a monitoring mechanism that is not part of WS Binder but interacts with it.

6.2.2.2 Approaches for service interface adaptation

SCENE ([85], [86]) 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 6.2.2.1) 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 [81].

The SCENE framework has been extended (see [80]) 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.

6.2.2.3 Approaches for run-time service composition evolution

The **WS-Diamond** EU project ([82]; [83]) developed a self-healing service composition execution environment and design tool. Self-healing is realized by the diagnosis of faults and applying a set of repair actions (i.e. compensation of operations, retrying and substitution of services). Faults are identified through diagnosis techniques while 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. Methods and tools to assess the self-healability of processes have been proposed, including reparability 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.

In [84] a **reputation based approach** is described. It targets the problem of a dynamic service composition that become unavailable when the component services fail or become defective. The decision of the run-time criterion for selecting the best possible service is made for each invocation of the component services and the decision is driven by the only factor, which is service reputation. However, the approach proactively provides 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. This technique does not require extra description of the component services needed to drive adaptation strategy.

In [85] and [86] the authors propose an 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 provides 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).

6.3 Comparison

Table 4: Comparison of qualities

Name	Support services that are executed either by humans or computers	Support services freely available on the Internet as well as enterprise services	Managing QoS and SLAs	Trust and Security	Scalability	Ease of use
Stable approaches executing standard BPEL	O	O	O	O	-	-
Approaches for dynamic binding and re-binding	-	X	O	-	-	-
Approaches for service interface adaptation	-	X	-	O	-	-
Approaches for run-time service composition evolution	-	X	X	O	-	-
Platform transformation based approaches	X	X	X	X	X	O

X = yes, O = partly covered, - = Not Covered

6.4 List of missing unfulfilled requirements

Table 5: List of unfulfilled requirements

Name	Covered by current approaches	To be covered by WP6?
Support services that are executed either by humans or computers	O	X
Support services freely available on the Internet as well as enterprise services	X	X
Managing QoS and SLAs	O	O
Trust and Security	O	O
Scalability	-	X

Ease of use	**-**	**X**

X = yes, O = Partly Covered, - = Not Covered

7. Conclusions

In the context of WP6 of the SOA4All project the term service construction mainly refers to model and execute composite services and processes in a lightweight manner. This should enable the non-technical end user to build new services and processes according to its specific needs. Hence the technologies used within the SOA4All project should allow for seamlessly integrating every kind of service, providing them on a generic user interface and making them usable for non-technical experts.

Since in the area of business process modelling most research addresses sophisticated and highly formalised process descriptions we focus on the need to enable non-technical users to describe their to-be processes in a lightweight manner. As stated in the deliverable the term “lightweight” means simple to use and having an abstract way to represent composite services and processes.

In this document we first have defined the relevant terms. We then have reviewed different research topics that are relevant for the service construction aspects of the SOA4All project. These areas concern the user interface, the business process modelling and execution languages, template management and process generation, verification and evaluation, and service and process execution. For each identified research area, general requirements for the SOA4All project have been discussed and the current approaches reviewed and compared. In a further step we have mapped the functionalities provided by the current approaches to the requirements that were listed at the beginning of each section. Proceeding like this we were able to identify the users’ needs that are not yet addressed by existing technologies. Finally in each section we have summarized the existing gaps in a list of unfulfilled requirements. These lists of unfulfilled requirements can be seen as the main outcome of this deliverable.

The next steps in this work package are, on the basis of the presented state of the art and collection of requirements, to design a framework for the lightweight construction of services. The following WP6 deliverables (i.e. *D6.3.1. Specification Of Lightweight, Context-aware Process Modelling Language*, *D6.4.1 Specification and First Prototype Of Service Composition and Adaptation Environment*, and *D6.5.1. Specification and First Prototype of Composition Framework*) will take all the gaps found in the state of the art as a starting point; and they will compose all together a first design draft of the lightweight service composition environment.

8. References

1. Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2003). Web services: Concepts, architectures and applications. Springer Verlag.
2. Barros, A., Dumas, M., and Oaks, P. (2005). A Critical Overview of the Web Services. Choreography Description Language. BPTrends March.
3. European Commission: Handbook on Implementation of the Services Directive, Commission of the European Communities Internal Market and Services DG, Brussels, 2007.
4. Bondi A. B., Characteristics of scalability and their impact on performance, Proceedings of the 2nd international workshop on Software and performance, Ottawa, Ontario, Canada, 2000, ISBN 1-58113-195-X, pages 195 – 203
5. N. Mehandjiev and L. Bottaci. User-enhanceability for organizational information systems through visual programming. In P.Constantopoulos et al editors, Advanced Information Systems Engineering: 8th International Conference, CAiSE'96, Lecture Notes in Computer Science No 1080, pages 432-456. Springer-Verlag, 1996.
6. Johannes Maria Zaha, Alistair P. Barros, Marlon Dumas, Arthur H. M. ter Hofstede: Let's Dance: A Language for Service Behavior Modelling. OTM Conferences (1) 2006: 145-162
7. Martinez, A., Patino-Martinez, M., Jimenez-Peris, R., and Perez-Sorrosal, F. 2005. ZenFlow: A Visual Web Service Composition Tool for BPEL4WS. In Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (September 20 - 24, 2005). VLHCC. IEEE Computer Society, Washington, DC, 181-188. DOI= http://dx.doi.org/10.1109/VLHCC.2005.74
8. Li, Karen Na-Liu. Visual Languages for Event Integration Specification. PhD thesis, University of Auckland, Department of Computer Science.2008
9. van der Aalst, W.M.P. (2003). Don't go with the flow: Web Services composition standards exposed, IEEE Intelligent, February 2003
10. Arkin, A. Askary, S. Bloch, B. Curbera, F. Goland, Y. Kartha, N. Liu, C.K. Mehta, V. Thatte, S. Yendluri, P. Yiu, A. and Alvesa, A. (2005) Web services business process execution language, version 2.0, December 2005
11. Andrews, T. Curbera, F. Dholakia, H. Goland, Y. Klein, J. Leymann, F. Liu, K. Roller, D. Smith, D. Thatte, S. Trickovic, I. and Weerawarana. S. (2003) Business process execution language for Web services, version 1.1, May 2003.
12. Curbera, F. Goland, Y. Klein, J. Leymann, F. Roller, D. Thatte, S. and Weerawarana. S. (2002) Business process execution language for Web services, version 1.0, July 2002.
13. Recker, J. C.; Indulska, M. Rosemann, M. and Green, P. (2005) Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. In Campbell, Bruce and Underwood, Jim and Bunker, Deborah, Eds. Proceedings 16th Australasian Conference on Information Systems, Sydney, Australia.
14. Becker, J.; Kugeler, M.; Rosemann, M. (2003) Process Management. A Guide for the Design of Business Processes. Springer-Verlag.
15. White S.A. (2004) Business Process Modelling Notation (BPMN) Version 1.0. Business Process Management Initiative, BPMI.org.
16. van der Aalst, W.M.P. and ter Hofstede, A.H.M. (2005) YAWL: Yet Another Workflow

Language. Information Systems, 30(4):245-275, 2005.

17. van der Aalst, W.M.P.; ter Hofstede, A.H.M.; Kiepuszewski, B. and Barros, A.P.(2003) Workflow Patterns. Distributed and Parallel Databases, 14(3):5-51, 2003.
18. van der Aalst, W.M.P. Aldred, L. Dumas, M. and ter Hofstede, A.H.M. (2004) Design and Implementation of the YAWL system. Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE04), Riga, Latvia, June 2004. Springer-Verlag.
19. Jan Recker, Marta Indulska and Peter Green. (2007) Extending Representational Analysis: BPMN User and Developer Perspectives. In Business Process Management. Springer. DOI - 10.1007/978-3-540-75183-0_28
20. Terje Wahl, Guttorm Sindre, (2005). An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In Proceedings of the Workshop on Evaluating Modelling Methods for Systems Analysis and Design (EMMSAD'05), held in conjunctiun with the 17th Conference on Advanced Information Systems (CAiSE'05), Porto.
21. Wand, Y., Weber, R. (1993). On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. Journal of Information Systems 3, 217–237
22. Agrawal R., Gunopulos D., Leymann F., Mining process models from workflow logs, in: Proceedings of the Sixth International Conference on Extending Database Technology, pp. 469–483, 1998
23. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. MBP: a Model Based Planner. In Proceeding of ICAI-2001 workshop on Planning under Uncertainty and Incomplete Information, pages 93–97, Seattle, USA, August 2001.
24. Brown, D., B. Chandrasekaran: Design problem solving: knowledge structures and control strategies. Research notes in Artificial Intelligence (1989)
25. Bryant R. E., Graph-Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, C-35(8):677–691, August 1986.
26. Cook J.E., Wolf A.L., Software process validation: quantitatively measuring the correspondence of a process to a model, ACM Transactions on Software Engineering and Methodology 8 (2) (1999) 147–176
27. Dal Lago U., Pistore M., and Traverso P.. Planning with a Language for Extended Goals. In Proc.AAAI'02, 2002.
28. Deelman E., Singh G., Su M., Blythe J., Gil Y., Kesselman C., Mehta G., Vahi K., G. Berriman B., Good J., Laity A., Jacob J. C., Katz, D. S.. "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems" .Scientific Programming Journal, Vol 13(3), 2005, Pages 219-237
29. Giunchiglia F., and Traverso P., Planning as Model Checking. In Proc. 5th European Conference on Planning (ECP'99), pages 1–20, September 1999.
30. Gold E.M., Complexity of automation identification from given data, Information and Control 37 (3) (1978) 302–320.
31. Greco G., Guzzo A., Pontieri L., Sacca D.. Mining expressive process models by clustering workflow traces. Proc. of the 8th PAKDD, 2004.
32. Kephart J., Chess D., The Vision of Autonomic Computing.IEEE Computer, 36(1):41–50, 2003.
33. Lee K., Paton N. W., Sakellariou R., Deelman E., Fernandes A. A. A., Mehta G., (2008)

Adaptive Workflow Processing and Execution in Pegasus 3rd International Workshop on Workflow Management and Applications in Grid Environments (WaGe08), in Proceedings of the Third International Conference on Grid and Pervasive Computing Symposia/Workshops, Pages 99-106, ISBN 978-0-7695-3177-9, May 25-28 2008, Kunming, China.

34. Levesque H., Reiter R., Lespérance Y., Lin F., and Scherl R,. GOLOG: A logic programming language for dynamic domains, 1997, Journal of Logic Programming, Vol. 31 pp. 59-84.

35. Mandell, D., McIlraith, S.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web service Interoperation. Int'l Semantic Web Conference, 2003: 227–241.

36. McCarthy, J., Hayes, P. J. “Some philosophical problems from the standpoint of artificial intelligence”, 1969 In Meltzer, B. and Michie, D., editors. Machine Intelligence 4, pages 463502. Edinburgh University Press. Can be found at http://wwwformal.stanford.edu/jmc/mcchay69.html.

37. McIlraith S. and Son T. Adapting Golog for composition of semantic Web services, 2002, Proc 8th International Conference on Principles of Knowledge Representation and Reasoning.

38. Ponnekanti S. and Fox A. “SWORD: A Developer Toolkit for Web Service Composition”, 2002, In Proceedings of the 11th International World Wide Web Conference 2002, WWW02.

39. OWL Services Coalition (2004), OWL-S 1.1 Release: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.0/owl-s.pdf.

40. Rao J., Su X., A Survey of Automated Web Service Composition Methods, 2005, Semantic Web Services and Web Process Composition, Vol. 3387/2005 (2005), pp. 43-54.

41. Sacerdoti E.. The Nonlinear Nature of Plans. In Proceedings of the International Joint Conference on Arti□cial Intelligence IJCAI-75, pages 206–214, 1975.

42. Schuster H., Georgakopoulos D., Cichocki A., Baker D.. Modelling and composing service-based and reference process-based multi-enterprise processes. In Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

43. Sheshagiri, M., desJardins, M., Finin, T.: A planner for composing service described in DAML-S. Workshop on Planning for Web Services at ICAPS 2003.

44. Silva, R., Zhang, J., and Shanahan, J. G., Probabilistic workflow mining. In Proceeding of the Eleventh ACM SIGKDD international Conference on Knowledge Discovery in Data Mining (Chicago, Illinois, USA, August 21 - 24, 2005). KDD '05. ACM Press, New York, NY, 275-284, 2005.

45. Sirin E., Parsia B., Wu D., Hendler J., Nau D., “HTN planning for Web Service composition using SHOP2”, 2004, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1, No. 4. (October 2004), pp. 377-396.

46. Tate A.,. Generating Project Networks. 1977, In Proceedings of the International Joint Conference on Arti□cial Intelligence IJCAI-77, pages 888–893,.

47. ten Teije, A., van Harmelen, F.,Wielinga, B, Configuration of Web Services as Parametric Design, Lecture Notes in Artificial Intelligence Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management EKAW'04

48. ten Teije, A., van Harmelen, F., Schreiber, G.,Wielinga, B., Construction of problem-solving methods as parametric design. Int'l Journal of Human Computer Studies, Special Issue on Problem-Solving Methods 49(4), 1998: 363-389.

49. Traverso P. and Pistore M. Automated Composition of Semantic Web Services into Executable Processes. In 3rd International Semantic Web Conference, November 2004.

50. van Splunter, S., Sabou, M., Brazier, F., Richards, D.: Configuring Web service, using structurings and techniques from agent configuration. IEEE/WIC Int'l Conference on Web Intelligence, 2003.

51. Wu, D., Sirin, E., Parsia, B., Hendler, J., Nau, D.: Automatic Web Services Composition Using SHOP2. Workshop on Planning for Web Services at ICAPS 2003.

52. Casati F., Ilnicki S., LiJie J. Krishnamoorthy V., Shan M., Adaptive and Dynamic Service Composition in eFlow, Conference on Advanced Information Systems Engineering, 2002

53. Curran, T., Keller, G., and Ladd. A.SAP R/3 Business Blueprint: Understanding the Business Process Reference Model.(1998) Prentice Hall, Upper Saddle River, NJ, USA.

54. Fettke, P., and Loos, P. (2003) Classification of Reference Models – a Methodology and its Application. Information Systems and e-Business Management, 1(1):35–53.

55. Fettke, P., Loos, P., and Zwicker, J. (2006) Business Process Reference Models: Survey and Classification. In C. Bussler and A. Haller, editors, Business Process Management Workshops, volume 3812 of Lecture Notes in Computer Science, pages 469–483, Springer Verlag.

56. Rickayzen, A., Dart, J., Brennecke, C., and Schneider, M. (2002) Practical Workflow for SAP –Effective Business Processes using SAP's WebFlow Engine. Galileo Press.

57. Czarnecki, K.; Antkiewicz, M. (2005). Mapping Features to Models: A Template Approach Based on Superimposed Variants. In Gluck R. and Lowry M.R., editors, GPCE, volume 3676 of Lecture Notes in Computer Science, pages 422-437. Springer.

58. La Rosa, M.; Lux, J.; Seidel, S.; Dumas M. and ter Hofstede, A.H.M. (2007) Questionnaire-driven Configuration of Reference Process Models. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Trondheim, Norway. LNCS Vol. 4495, pp. 424–438, Springer-Verlag.

59. Rosemann, M. and van der Aalst W.M.P. (2007). A Configurable Reference Modelling Language. Information Systems, 32(1): 1-23, 2007.

60. Schnieders, A.; Puhlmann, F. (2006) Variability Mechanisms in E-Business Process Families. In W. Abramowicz, H. Mayr (Eds.): 9th International Conference on Business Information Systems (BIS 2006), pages: 583-601.

61. 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

62. G Canfora, M. Di Penta, "SOA: Testing and Self-Checking", WS-MATE 2006

63. L. Baresi and S. Guinea. Towards dynamic monitoring of WS-BPEL processes. In Benatallah et al. 64, pages 269–282.

64. B. Benatallah, F. Casati, and P. Traverso, editors. Service-Oriented Computing - ICSOC 2005, Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005, Proceedings, volume 3826 of Lecture Notes in Computer Science. Springer, 2005.

65. Axel Martens. Analysis and Re-Engineering of Web Services. Proceedings of the 6th International Conference on Enterprise Information Systems, pages 419-426, 2004
66. Narayanan, S., McIlraith, S., 2002. Simulation, Verification and Automated Composition of Web Services http://www.icsi.berkeley.edu/~snarayan/nar-mci-www11.pdf
67. L. Baresi, R. Heckel, S. Thöne, and D. Varrò. Modelling and Analysis of Architectural Styles Based on Graph Transformation. Proceedings of the 6th Workshop on Component-Based Software Engineering, co-located with ICSE 2003Verification
68. Howard Foster, Sebastián Uchitel, Jeff Magee, Jeff Kramer. Model-based Verification of Web Service Compositions. Proceedings of the 18th IEEE International Conference on Automated Software Engineering, IEEE CS, pages152-163, 2003.
69. Xiang Fu, Tevfik Bultan and Jianwen Su. Analysis of interacting BPEL Web services. Proceedings of the 13th international conference on World Wide Web, ACM, pages 621-630, 2004
70. A1.D3.4 Testing method definition- Technical Report – SECSE FP6 EU Project
71. Wynn, Moe T. and Verbeek, H.M.W. and van der Aalst, Wil M. and ter Hofstede, Arthur H. and Edmond, David (2007) Business Process Verification - Finally a Reality!. Business Process Management Journal
72. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Information Systems , 30(4):245-275, 2005.
73. OASIS Web Services Business Process Execution Language (WSBPEL) TC: http://www.oasis-open.org/committees/wsbpel/
74. Luciano Baresi, Elisabetta Di Nitto, Carlo Ghezzi: Towards Open-World Software: Issue and Challenges. SEW 2006: 249-252.
75. BEA AquaLogic® BPM: http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/aqualogic/albpm/
76. The ActiveBPEL Community Edition Engine: http://www.activevos.com/community-open-source.php
77. 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
78. 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 – web site: http://black.elet.polimi.it/paws
79. 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
80. 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
81. 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
82. WS-Diamond Team, “WS-DIAMOND: an approach to Web Services – DIAgnosability, MONitoring and Diagnosis”, Proc. E-Challenges Conference, The Hague, October 2007

83. C. Cappiello, B. Pernici, “A Methodology for Information Quality Management in Self-healing Web Services”, International Conference on Information Quality, Boston, Nov. 2006
84. 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
85. 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
86. 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
87. Gero Decker, Oliver Kopp, Frank Leymann, Mathias Weske, "BPEL4Chor: Extending BPEL for Modelling Choreographies," icws, pp. 296-303, IEEE International Conference on Web Services (ICWS 2007), 2007
88. Nick Russell, Arthur H.M. ter Hofstede, and Wil M.P. van der Aalst. newYAWL: Specifying a Workflow Reference Language using Coloured Petri Nets. ,Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, October 2007.
89. E. Motta, "Reusable Components for Knowledge Modelling", in Frontiers in Artificial Intelligence and Applications Vol. 53, IOS Press, 1999
90. Roman, D., Keller, U., Lausen, H., de Brujin, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. 2005. Web Service Modelling Ontology. Applied Ontology 1, 77–106.
91. WSMO. Web Service Modelling Ontology working group (WSMO). http://www.wsmo.org/.
92. OWL-S. Web Service Ontology Language working group (OWL-S). http://www.daml.org/services/owl-s/.
93. Haller, A., Cimpian, E., Mocan, A., Oren, E., and Bussler, C. 2005. WSMX - a semantic service oriented architecture. In Proceedings of the 3rd IEEE International Conference on Web Services. Orlando, FL.
94. Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., and Pedrinaci, C. 2006. IRS-III: A broker for semantic Web services based applications. In Proceedings of the 5th Semantic Web Conference. Athens, GA.
95. Cabral, L., Domingue, J., Motta, E., Payne, T. and Hakimpour, F. (2004). Approaches to Semantic Web Services: An Overview and Comparisons. In proceedings of the First Euro-pean Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece. LNCS 3053
96. Garofalakis, J., Panagis, Y., Sakkopoulos, E., and Tsakalidis, A. 2006. Contemporary Web Service Discovery Mechanisms. Journal of Web Engineering 5, 3, 265–290.
97. Bianchini, D., De Antonellis, V., Pernici, B., and Plebani, P. 2006. Ontology-based methodology for e-service discovery. Inform. Syst. 31, 4-5, 361–380.
98. Bernstein, A. and Klein, M. 2002. Towards high-precision service retrieval. In Proceedings of the 1st International Semantic Web Conference. Sardinia, Italy.
99. Stroulia, E. and Wang, Y. 2003. Semantic structure matching for assessing Web service similarity. In Proceedings of the 1st International Conference on Service Oriented Computing. Trento, Italy.

100. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Mecella, M. 2005. Automatic service composition based on behavioral descriptions. Int. J. Coop. Inf. Syst. 14, 4, 333–376.

101. Benatallah, B., Sheng, Q. Z., and Dumas, M. 2003. The Self-Serv environment for Web services composition. IEEE Internet Comput. 7, 1, 40–48.

102. Benatallah, B., Dumas, M., and Sheng, Q. Z. 2005. Facilitating the rapid development and scalable orchestration of composite Web services. Distrib. Parallel. Dat. 17, 1, 5–37.

103. Ardagna, D. and Pernici, B. 2007. Adaptive service composition in flexible processes. IEEE Trans. Softw. Engin. 33, 6, 369–384.

104. Yu, T., Y., Z., and Lin, K.-J. 2007. Efficient algorithms for Web services selection with end-to-end quality constraints. ACM Transactions on the Web 1, 1, 1–26.

105. Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. 2004. QoS-Aware middleware for Web services composition. IEEE Trans. Softw. Engin. 30, 5, 311–327.

106. Cardoso, J. and Sheth, A. 2003. Semantic E-Workflow composition. J. Intell. Inf. Syst. 21, 3, 191–225.

107. Agarwal, R., Verma, K., Miller, J., and Milnor, W. 2004. Constraint driven Web service composition in METEOR-S. In Proceedings of the 1st IEEE International Conference on Services Computing. Shanghai, China.

108. W3C: Web Services Choreography Description Language Version 1.0, 2005, http://www.w3.org/TR/ws-cdl-10/

109. SCA Service Component Architecture. Assembly Model Specification, 2006, www.osoa.org/download/attachments/35/SCA_AssemblyModel_V096.pdf

110. B. Benatallah, M. Dumas, M.-C. Fauvet, and F. Rabhi. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi, editors, Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, UK, Nov 2002.

111. 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

112. PO-JRA-1.1.1 - State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge – http://www.s-cube-network.eu