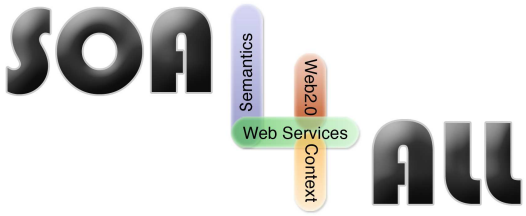


Project Number: **215219**

Project Acronym: **SOA4ALL**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D6.4.1 Specification and First Prototype Of Service Composition and Adaptation Environment

Activity:	Activity 2 - Core Research and Development	
Work Package:	WP 6 - Service Construction	
Due Date:		M12
Submission Date:		06/03/2008
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		ATOS
Revision:		1.7
Author(s):	Rafael González-Cabero Freddy Lecue Matteo Vila Sven Abels (reviewer) Francoise Baude (reviewer)	ATOS UNIMAN TXT TIE INRIA

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	5/12/2008	First Version	Rafael González-Cabero (ATOS)
1.1	29/1/2009	Template Generator theoretical description and architecture, Updates	Matteo Vila (TXT), Rafael González-Cabero(ATOS)
1.2	3/2/2009	Semantic Link Design Operador and Optimizer Sections	Freddy Lecue (UNIMAN)
1.3	6/2/2009	Process Template Generator updates	Matteo Vila (TXT)
1.4	14/2/2009	General review	Rafael González-Cabero(ATOS)
1.5	25/2/2009	Comments from the reviewer	Sven Abels (TIE)
1.6	3/3/2009	Comments from the reviewer	Francoise Baude (INRIA)
1.7	4/3/2009	Some revisions	Matteo Villa (TXT)
Final	9/3/2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Definition
BPM	Business Process Modelling
BT	British Telecom
BT 21CN	21st British Telecom Century Network
C2C	Commerce to Commerce
COP	Constraint Optimization Problem
CSP	Constraint Satisfaction Problem
D	Deliverable
DSB	Distributed Service Bus
EC	European Commission
eCommerce	Electronic Commerce
eShop	Electronic Shop
GA	Genetic Algorithm
ID	Identificator
IP	Integer lineal Programming
IT	Information Technology
MAS	Multi Agent Systems
NF	Non-Functional
NP	Nondeterministic Polynomial
ProM	Process Mining
QoS	Quality of Service
SOA	Service-Oriented Architecture
UML	Unified Modelling Language
URI	Uniform Resource Identifier
WP	Work Package
DAG	Direct Acyclic Graph

Executive summary

One of the main objectives of SOA4All is to facilitate to a wide spectrum of users the adoption of service-oriented computing, so that they can benefit from the advantages that such paradigm brings. Web services are becoming the de facto standard for the implementation of service-oriented architectures, specially in the case distributed enterprise computing systems, as they enable collaborative business processes and ease their construction by their recombination. Nevertheless, service orientation and hence the usage of Web services to provide business functionality is mostly kept within enterprises boundaries.

In this work package, we address this problem from one concrete perspective of the service lifecycle, service construction as the composition of business processes. Most 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**. We will build an environment **that will transform the aforementioned lightweight processes in to complex services orchestrations transparently in a context-dependant manner**. In this document, ***D6.4.1 Specification and First Prototype Of Service Composition and Adaptation Environment*****,** we address precisely this transparent transformation from a user-oriented lightweight representation to a more complex one, by means of advanced composition techniques. As described in the description of work, this deliverable includes a draft of the functional requirements that we have to consider in order to design the software, as well as design models in standard notations such as UML. Nevertheless, in our humble opinion, solely these specifications without the description of the background theory that leads us to that design, and its associated algorithms, is meaningless. Therefore, we will also make special emphasis in what we refer as its theoretical grounding. This theoretical grounding will include advances in three main topics: an scalable approach to service composition using parametric design; a novel approach to service composition to optimize services using genetic algorithms; and finally, as parametric design and design in general will rely heavily on a set of abstract process templates, a generator of such abstract templates from past executions.

1. Introduction

SOA4All motto is to enable the SOA revolution on a worldwide scale. One of the main difficulties to overcome this enterprise is to empower non-technical users to do simple IT modelling and development work in the area of service construction and composition. This involves elements of BPM to describe the processes, which bring together constituent services. However, the use of existing BPM solutions requires high expertise in both business and IT. Moreover, the composed processes should be tailored to a specific context (e.g. user characteristics, geographical location, environmental details, etc.) both at design-time and at run-time on a per case basis.

In the context of this task, ***Task 6.4 Context-aware Service Composition and Adaptation***, we aim to aid end-users to tackle with these difficulties by **Creating a scalable system for the flexible and ad-hoc creation and adaptation of complex services at design time**. We should transform the aforementioned lightweight processes in to optimized complex services orchestrations; or already existing complex services should be adapted to a specific use. These activities should be heavily influenced by the context in which they will be carried out.

The system for the flexible and ad-hoc creation and adaptation of complex services will make use of available generic process templates, the environmental context information, and user needs (expressed using the lightweight modelling language). Thus, it will be adaptive in the sense that the composition system will be able to tailor itself to the needs of a particular user in a particular context. We will base this activity on the knowledge-intensive configuration of instances of generic parameterized process templates. The inputs for this composition tool will be the user requirements (a lightweight BPM process, see T6.3 and its specification in *D6.3.1. First Specification Of Lightweight Process Modelling Language*) and context. The user requirements will determine the skeleton of the service template (formed by a set of pre-designed reusable templates). We will use context and the current state of affairs as requirements and constraints that determine the parametric design process that will instantiate this newly formed service template. Both this parametric design process, and the initial creation of the skeleton, will be enabled and enhanced by means of the semantic descriptions of service templates; and a set of different case-based heuristics. To ensure the application of parametric design in a controlled and safe fashion, we will access the variability of generic process composition templates for the target class of problems, and identify classes of generic problem-solving templates which comprise a suitable basis for adaptive service composition. We will then investigate the impact of context on the variability dimension.

We will also try to enhance the composition and adaptation processes in two different innovative ways.

- We will develop a module to achieve **optimal service compositions**, mainly by means of the non-functional (henceforth NF) quality of services (e.g., price, robustness, response time, reliability). In this direction the NF quality of service is useful to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.
- We will develop an automatic **template generator**, that will generate new process templates using data coming from past service executions. We will dynamically compose service templates applying machine-learning techniques, and being able also to order them in a hierarchical fashion. This module will be very useful in many complex organisational situations, where the effort required for specifying a process schema is too high. Later on, end-users will have the opportunity of adapting these automatically generated processes.

Purpose and Scope

The goal of this deliverable is twofold:

- From an engineering perspective, the goal of this deliverable is **to provide a specification and a design for the first prototype of the Service Composition and Adaptation Environment**.
- From a research perspective **to provide a clear description of the advanced techniques and algorithms that we will use**. That includes a novel approach to AI parametric-design techniques using a multi agent approach to configure and adapt services processes; an innovative composition algorithm for optimal services, based on the formal description of the semantics of both functional and non-functional properties of services; and finally, an automatic template process generator, that is able to generate abstract process templates and their hierarchy form past executions.

Structure of the document

We structure the deliverable into the following sections:

- **Requirements**. In this section, we will enumerate the requirements applicable to the service composition and adaptation environment. They will be a resume of those that verse about service composition and adaptation of the catalogue of requirements that we defined in the deliverable *D6.1.1 State of the Art Report And Requirements For Service Construction*. That includes requirements identified in the use cases; and those posed by SOA4All challenges as a whole.
- **Theoretical Grounding**. In this section, we will describe the theory and our research approach to service composition and adaptation, covering an extension to the classic approach to design, advanced algorithms to achieve optimized compositions of services, and the automatic extraction of process templates from previous logs.
- **Design of the 1st Prototype**. Finally, in this section we are going to provide a specification of the software artefacts that compose the first implementation of the prototype of the Service Composition Framework.

© SOA4All consortium

2. Requirements

In this section, we will enumerate the requirements applicable to the service composition and adaptation environment that we have extracted from the necessities identified in the use cases; and those posed by SOA4All challenges as a whole[1]. In this section we will summarize the requirements that we consider for the presented work.

2.1 Requirements from the Use Cases

2.1.1 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"**[1]. 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. This requires establishing new communication mechanisms between service providers and administration offices. Let us now enumerate the main requirements that we have identified as relevant to this concrete task:

- 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, and *Back-office*, very detailed knowledge of selected processes
- The lightweight processes and services are connected via user defined control flow (incl. rule-based logic)
- Processes should be reusable. Further, a process should be identifiable as a building block that can be recombined to more complex processes.
- We must take in to account when composing processes the nature of each component of processes. They can be:
    - 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

[1] The catalogue of requirements for this work package are contained in the deliverable *D6.1.1 State of the Art Report And Requirements For Service Construction*

automatic information sources (services output or current context).

- Regarding the abstraction level that we might expect Processes should be described with the enough abstraction and freedom to allow:
    - transparent deployment on demand
    - multiple and parallel running instances per process

2.1.2 W21C BT Infrastructure

Web21C 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 (e.g. Ribbit services); but also third parties will provide others (e.g. facebook, amazon, etc.). 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 identify different requirements from each of the scenarios that we have defined for this use case that we believe relevant.

- 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 be based on different criteria, namely functionally based, non-functional based (e.g. QoS), user goal based, and context-based.
    - We should define formally the lightweight model operational semantics in order to automate tasks such as the suggestions for compatible services in Web service compositions.
    - Compositions should be easily extendable.
- 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 C2C Service eCommerce

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, it will enable the inclusion of additional third party services via a service broker.

The requirements we believe that more closely related with service composition are:

- Complex services need to be constructed based on the connection of simple services.
- The end users can build their eShop using ready-to-use process templates such as manage products, categories, stock, payment and delivery options and services.

2.2 Requirements from SOA4All

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 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. We presented those principles at a very high level since we can address them from various points of view, using various technologies. The commitment to these principles poses specific challenges and requirements that will affect directly to this work. Let us enumerate them.

- **We should support 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 hard to predict that in the midterm as more advanced mobile devices and more efficient wireless infrastructures appear, facilitating 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. This near-term situation 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; or to be readily enlarged to cope with new workload on the fly (i.e. should be elastic).

3. Theoretical Grounding

In this section we will describe the theory and therefore the motivation of the design of the composition software environment. Let us introduce the overall process, and relate it with the different sections that we will include hereafter.

As we have previously stated, the main motivation of our work is to facilitate to non-expert users the composition and adaptation of processes, by means of a lightweight BPMN processes.

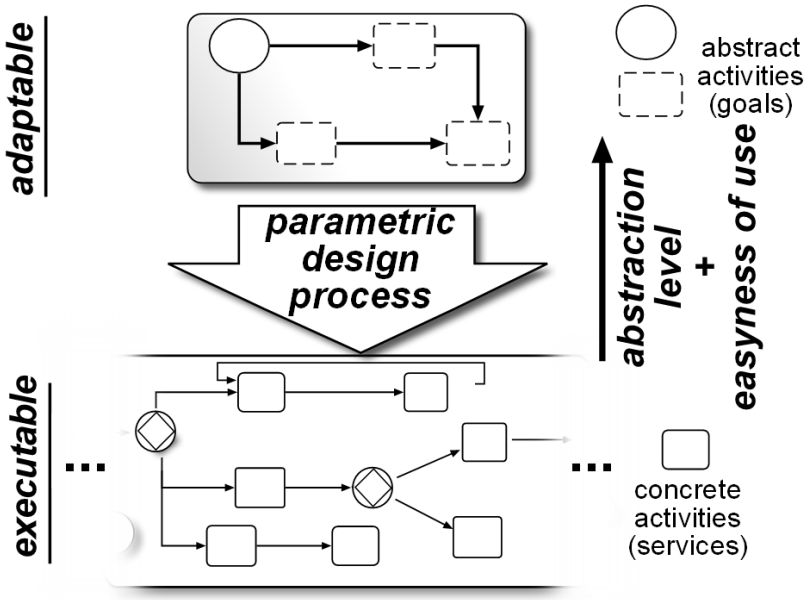


Figure 1 From abstract processes to concrete executable processes[2].

Our approach consists in letting users handle an abstract and easy to use processes representation, that one the on hand will ease enormously its usage; and on the other will be flexible enough to allow its later context-dependent customization. The user will handle coarse-grained goals, and the system will carry out intelligently the transformation in to concrete and executable process, that the system will tailor to the concrete execution context and the current state of affairs.

We will realize this transformation process using a knowledge-intensive configuration process, more precisely a parametric design process. As we shall see, in order to increase the scalability of this process, we will extend the classical approach to this synthesis task by using an opportunistic approach, based on blackboard-based multi-agent system. Nevertheless, the hearth of this system, as in classical approaches, is a set of reusable process templates that capture patterns of functionality of the system. Knowledge-experts are mean to create these templates, but we will provide an automatic template generator system that will ease experts' task, and in some cases, will allow end-users to obtain their own templates based on previous processes enactments.

Finally, when the system has generated an executable process we will perform, upon user request, an optimization process on it, based on the concrete services NF properties.

[2] Please, note that we express both abstract and executable processes using the same lightweight BPM language, defined in the deliverable *D6.3.1. First Specification Of Lightweight Process Modelling Language*. The difference resides mainly in the nature of its activities, and the complexity of their control flow (see Figure 1).

In the following sections we will describe all these topics, what is a parametric design, how we plan to implement it, how we plan to optimize its generated executable processes, and last but not least, how we plan to aim experts and end-users in the complicated task of the creation of process templates by means of the process template generator.

3.1 Parametric Design based Composition and Adaptation

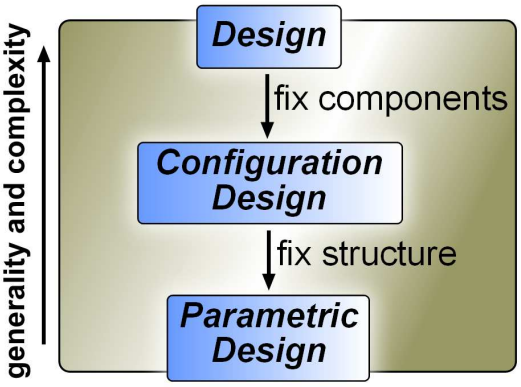
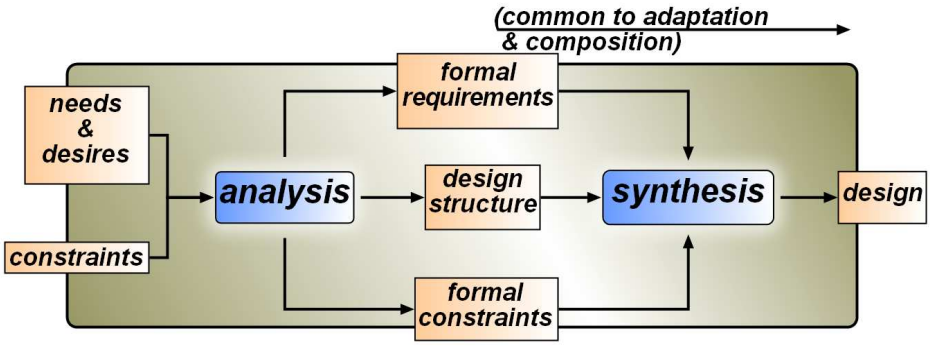


Figure 2 Relationship among the Design, Configuration Design and Parametric Design tasks.

As we have stated, the core of processes transformation will be perform by means of a parametric design procedure. Parametric design is a simplification of the design as configuration problem (as shown in Figure 2), which is in itself a simplified form of design where a set of pre-defined components[3] is a priori selected and combined till it satisfies a set of requirements and obeys a set of constraints [2][3]. Nevertheless, configuration-based design assumes a free arrangement of the components that compose the design. Parametric design refines configuration design, assuming the existence of functional solution templates that guide the design process, decreasing greatly the complexity of the design process. Parametric design restricts thus the space of possible designs, assuming that the targeted artifact can be expressed in the form of parameterized functional templates.

As described in [4], parametric design can be divided in two main phases, namely analysis and synthesis. The synthesis phase is common for any problem that we may address, since its starting point is a common formalized design model. The analysis phase, to the contrary, involves the gathering of design constraints, needs and desires which by nature are dependent of the problem to solve. Let us depict these concepts in detail in the following sections.



[3] The set of components considered for a design as configuration task receives is denoted as technology. We remit the reader to [5] for a clear description in all this matters.

Figure 3 Analysis and Synthesis phases (adapted from [4]).

3.1.1 Analysis

In the analysis phase we transform the needs, desires of the user and specific constraints of domain into a formal and complete set of requirements and constraints [4]. In this analysis process the composition system use of knowledge of the domain and external information sources. Fortunately in the context of service composition and service adaptation, the initial requirements are precise and very well structured, and therefore the analysis phase losses great part of its importance. On the one hand this process is not as difficult as in other domains where parametric design is being use (e.g. architecture, 3D objects modeling, etc). On the other hand, the process can be heavily automated, since in other fields analysis might involve a lot of human intervention and a knowledge elicitation process.

3.1.1.1 Analysis for Composition

In the context of the composition activity, the highly structured inputs of are:

- **An abstract lightweight process definition**. The input for the analysis for composition activity is an abstract lightweight process. This is one of the key concepts of our approach, in order to be able to customize services to a certain context, user-preference or situation; we need certain initial degree of freedom, admitting thus certain level of abstraction in our processes. Note that this degree of freedom is directly proportional to the easiness of use, since the user is not forced to specify the process in high detail.
- **A fixed set of user constraints and preferences**. They express the concrete preferences or restrictions that the user may pose in a concrete moment. The user can have a clear idea of some high-level requirement (e.g. price, time of execution of the service, security-related issues, etc.)
- **Contextual information**. The context of the environment, paying special attention to the context of the end-user, must be taken in to account in the analysis phase.

3.1.1.2 Analysis for Adaptation

In the analysis for adaptation there are a couple of subtle differences in the inputs:

- **A lightweight process definition.** As in the case of composition the input contains also a lightweight process definition. In this case, it does not have to be an abstract one, since adaptation can take as a starting point more concrete process descriptions than composition.
- **A non-monotonic requirements and preferences**. A set of non-monotonic requirements obtained in a negotiation process with the external agent that interacts with the service adaptation module.
- **The result of the previous iteration of the adaptation process.**

With this inputs as a starting point a first draft of the design model is created, and as such it is offered to the external agent. The agent then can add/modify/remove its requirements, in order to influence in the next iteration the adaptation process. Let us clarify the adaptation process with a sequence of schematic pictures.

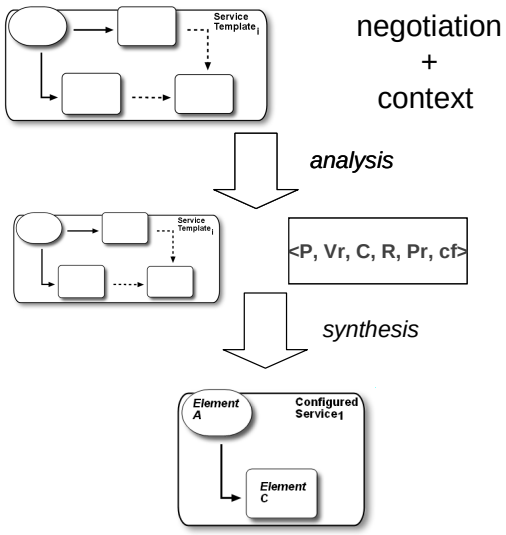


Figure 4 Adaptation process 1st iteration.

In the first iteration we obtain a preliminary design, that can be used as an input to the next iteration (as portrayed in Figure 5). Contrary to the case of composition, the user is aware of the result of the parametric design process.

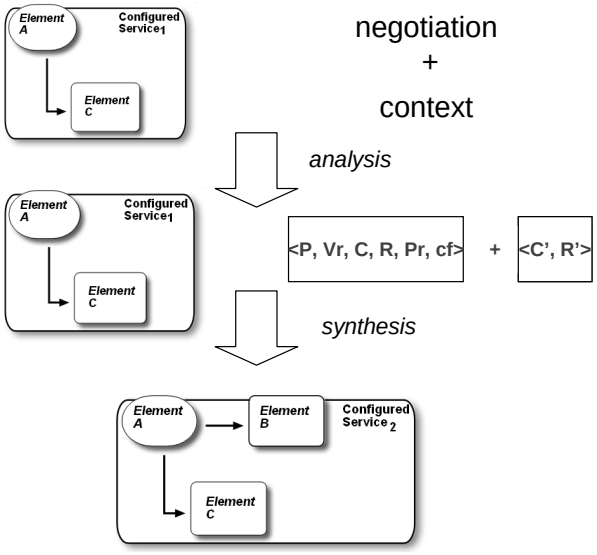


Figure 5 Adaptation Process 2nd iteration.

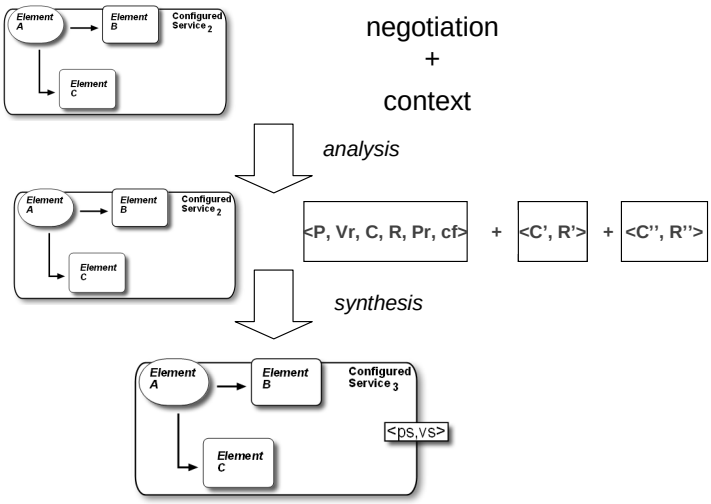


Figure 6 Adaptation Process final iteration.

3.1.2 Synthesis

The synthesis phase of parametric design can be characterized as search in a space of subassemblies [6]. We can consider it a search problem in a large space of artifacts of a subset of artifacts that satisfy multiple constraints. From the myriad of possible objects that can be generated, only an utterly small number of objects will be viable, they are needles in a huge haystack. Moreover, these designs are usually not by far optimal solutions. In Figure 7 we represent this search through the design space. From an initial design (depicted as a red circle in Figure 7 that we refer design plan as we will describe later on) we have to navigate thorough the space of possible intermediate design structures till we reach a design that we consider a final solution (the possible final solutions will also be described later on)

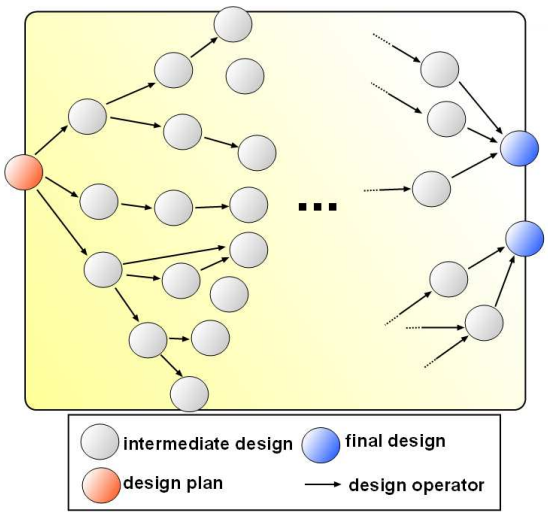


Figure 7 Parametric design synthesis as a search (figure adapted from [7]).

Transitions between states are realized by applying design operators (represented as arrows in *Figure 7*) to design models. Initially it is not required, but ideally, a design operator should drive the search towards a design model that is closer to the targeted final state (i.e. solution design model, either optimal or suboptimal, as we shall see).

In order to guide the navigation the synthesis phase of the parametric design process, the automatic designer makes use of knowledge about the world and of specific design knowledge. As depicted in the main use of specific domain knowledge is to relate possible

violations on the requirements and constraints of the design with concrete design decisions.

3.1.2.1 Characterization and formalization of design models

As we have already stated in the beginning of the description of parametric design, parametric design knowledge-intensive configuration process where designs have their structure fixed. Let us describe the design model of parametric design, which of course will include the fixed structure.

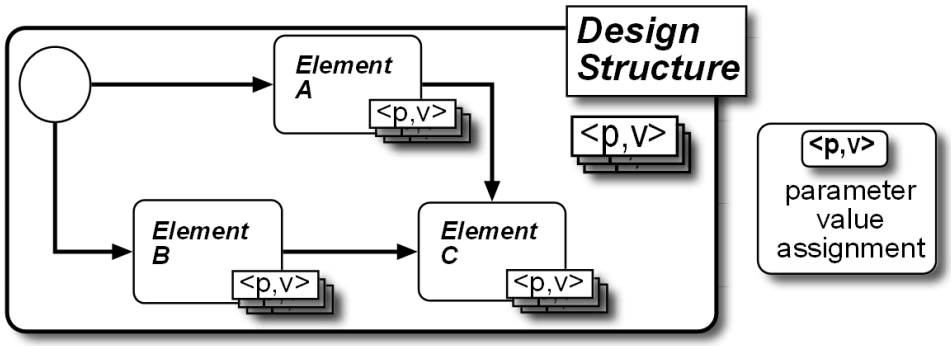


Figure 8 The fixed Design Structure.

We formalize a parametric design model as follows:

<P, Vr, A, C, R, DS, Pr, Cf>

- **Parameters and Value Ranges.** The set ***P*** represents the set of parameters that can be attached both to:
  - **Elements**. Each of the elements that belong to the design structure may have their own parameters.
  - **Whole structures**. The whole design structure can also be parameterized.
- **Value Ranges**. The set ***Vr*** ($Vr=\{V_1,.....,V_N\}$) represents the possible value ranges that each of the parameters where each $V_i = \{v_{i1},....., v_{iM}\}$.
- **Assignment Set.** The assignment set ***A*** is a set of tuples $\{(p_i, v_{ij})\}$ that represent the values associated with each of the parameters, both those of the design structure and those of the elements that compose the design structure. In the case that the parameter p_{ij} represents an atomic goal, we assign it a service that can solve that goal (details about how we represent such things are given in the deliverable D6.3.1).
- **Constraints and requirements**. The sets ***C***($=\{c_1 \ldots c_N\}$) and ***R***($=\{r_1 \ldots r_N\}$) represent the sets of constraints and requirements that formalize the admissibility and suitability of a design. Constraints indicate limitations on what counts as a possible design solution [4]. Requirements are also constraints and, as was described in [5], the difference between requirements and constraints is conceptual rather than formal. Requirements typically have a "positive" connotation, in the sense that they describe the desired properties that the solution must satisfy; whereas constraints have a "negative" connotation, in the sense that they limit the space of admissible designs, by expressing the applicable technological or physical constraints.
- **Design Structure**. As we have stated, parametric design fixes the structure of the design to be configured (a service in our case). Thus our problem of parametric design should also include the fixed process structure DS, which parameters should belong to the set *P*. We show a graphical representation of a design structure in Figure 8.

- **Preferences and Global Cost Function.** The set $Pr(=\{pr_1..pr_N\})$ is the set of preferences. Preferences describe knowledge that allows us given two design models, D_i and D_j, to make a judgement about which of the two is the more suitable (in accordance with some criterion. The Global Cost Function (cf) is closely related with the set of preferences, and provides a global cost criterion for ordering solution designs. As described in [7] we define a cost criteria is for each preference. The global cost function can be derived thus the combination of these preference-specific cost criteria. The global cost function $Cf = F(cf_1(pr_1),...,cf_N(pr_N))$, where $cf_i(pr_i)$ is the cost function associated with preference pr_i.

3.1.2.2 The taxonomy of possible designs

In Figure 9 we show graphically the taxonomy of possible design models. Depending on the characteristics of a given design model *D*, formalized as we have defined in the previous section, D can be classified as follows

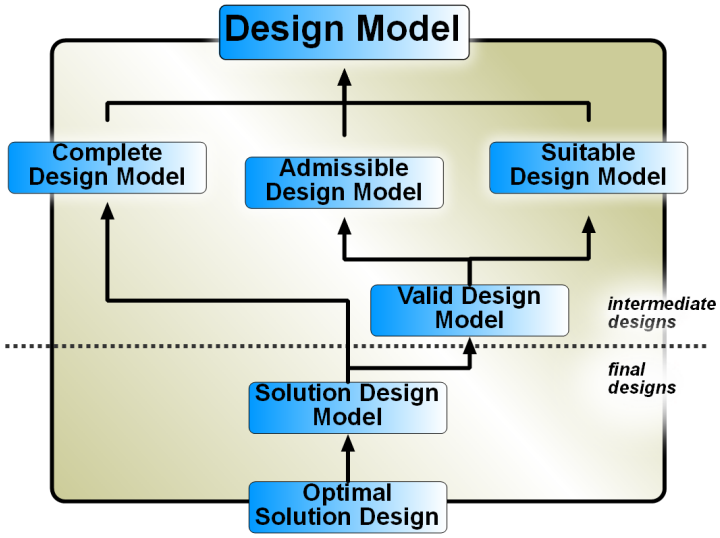


Figure 9 Taxonomy of possible designs models.

- D is a **complete design model** if each parameter, say p_i, in the parameters set *P* has a valid value assigned in *D*. In the case of the parameters that represent services (i.e. goals) means that each of the activities in the structure process has a service associated with it.

$$Complete(D) \equiv \forall p_i \in P \exists v_i \ \left| (p_i, v_i) \in A \wedge v_i \in V_i \right.$$

- D is an **admissible design model** if the design model does not violate any constraint in the set *C* in the design model *D*.

$$Admissible(D) \equiv \forall c_i \in C \ \left| \neg c_i(D) \right.$$

- D is a **suitable design model** is if satisfies all applicable requirements in the set R. $C(=\{c_1 \ldots c_N\})$ and $R(=\{r_1 \ldots r_N\})$.

$$Suitable(D) \equiv \forall r_i \in R \ \left| r_i(D) \right.$$

- D is a **valid design model** is if it is both an admissible and suitable model.

$$Valid(D) \equiv Admissible(D) \wedge Suitable(D)$$

- D is a **solution design model**, and therefore a solution to a parametric design problem, if it is complete and valid.

$$Solution(D) \equiv Valid(D) \wedge Complete(D)$$

- An finally, a design model D is an **optimal solution design model** if there not exist any other solution model with a smaller global cost than the cost of D. More formally:

$$OptimalSolution_{CF} \equiv \forall D' \left| Solution(D') \rightarrow CF(D) < CF(D') \right.$$

Note that we write $OptimalSolution_{CF}$ since it depends on the cost function that we consider.

3.1.3 A Multi-agent Approach to the Parametric Design Synthesis Phase

[8] describes that the occurrence of some combination of a set of problem characteristics can serve as a good indication of the appropriateness of the blackboard approach. In our opinion the synthesis phase of parametric design is a perfect candidate to be approached using a blackboard model, since it posses many of the afore mentioned characteristics, such as a large solution space, the need of the developing of various lines of reasoning, the need of incremental reasoning, opportunistic control, multiple reasoning techniques, etc.

Blackboard architectures where firstly introduced in the Hearsay-II speech understanding system [9]. Blackboard models formalise the metaphor of a group of experts working on a problem and communicating ideas using a blackboard. The blackboard becomes a repository of information, which is globally accessible and records the problem specific information available from each expert. The flow of information between the blackboard and each expert is bidirectional, since participating experts both contribute modifying the blackboard and obtain information from the blackboard. Moreover, the blackboard not only con, but also the controlflow since it also coordinates and synchronizes the participants. In Figure 10 we depict this approach, where multiple agents observe the same blackboard, and each of them select different design models to change, each

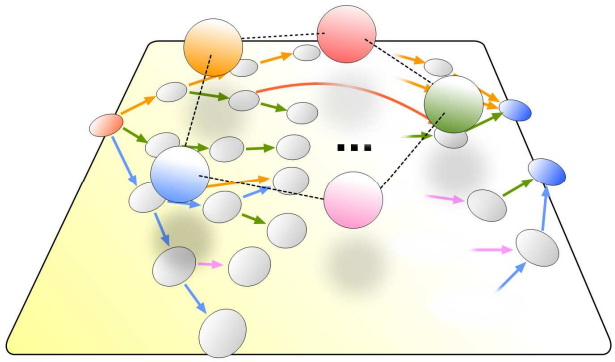


Figure 10 Multi-agent Approach to Parametric Design.

Apart from these characteristics of blackboard based systems, in the context of our work, in our concrete case they have two characteristics that we find of great importance :

- ***Divide et impera* principle enabler**. The blackboard model is a direct application of the *divide et impera* principle. The rationale underlying such a principle is to decompose complex problems into more manageable and presumably simpler ones. We can obtain solutions to a problem by bringing together solutions to smaller subproblems easier to solve. This is a well-known, widely applied, and often successful approach to dealing with complex problems that need to be solved in a scalable fashion. In future implementations of our composition and adaptation system, the reuse of previously solved subproblems will be a necessity.

- **Emergent behaviour enabler**. [10] present this advantage of blackboard-based MAS. The set of agents that share the blackboard intelligently coordinate a response to the request of the creation of an artefact, which emerges as a result of shared interests of agents (cf. entities, institutions) within a network of complementary expertise.

3.1.3.1 Agents Taxonomy

We shall describe now the initial taxonomy of agents that we consider for our MAS. In this section, we will describe how each of these collaborative agents decide which their actions, which will be. In Figure 11 we show the initial taxonomy of agents that we define in this first prototype.

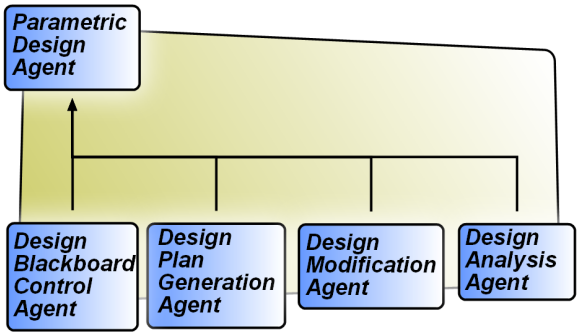


Figure 11 Taxonomy of the agents involved in the parametric design process.

3.1.3.2 Blackboard Control Agent

This type of agent is the one that initializes the blackboard when a synthesis request is received in the composition system. Its main roles are:

- Constitute and initialize the blackboard. Initially the blackboard will just contain the initial synthesis request.
- Advertise the newly created blackboard to possibly interested parties. That includes Design plan generators agents, Design model modification agents, and Design analysis agents.
- Recollect possible solutions. If one of the design analysis agents notifies that the synthesis process has resulted in a valid result, the agent sends the result back to the requester composition system.

Each of the cited types of agents will be described in the following subsections, and we will illustrate this process more precisely using UML diagrams in the design of the 1st prototype section.

3.1.3.3 Design Plan Generators

In few words, design plan generator agents are those responsible of generating the initial design plan using both the initial inputs to the parametric design process and the knowledge about the domain and context of the environment. A design plan specifies a sequence of design actions to take for producing a design or part of a design. Design commitments made by a design plan are abstract; as depicted in [6], choices are not made at the level of primitive objects but at the intermediate level of design abstractions. In our case, this choices are made in the level of goals (either atomic or composite), but they should not define concrete service invocations, this kind of assignment should be further refined in the parametric design process.

As in the case of In our concrete scenario, the design planner using the assumptions about the context contained in the adapter, should select which is the template that fits better in the actual context. In Figure 12 we show how a composite goal is related with different processes templates. Once a user request is placed in the initial blackboard, it contains a high-level process that contains, for example the $Goal_i$, the set of requirements (preferences and constraints of the user). The design plan generator, using the assumptions and applicability conditions contained in the adapters ($adapter_{ij}$ and $adapter_{ik}$) and the plus the actual context, chooses the most appropriate process template to replace $Goal_i$.

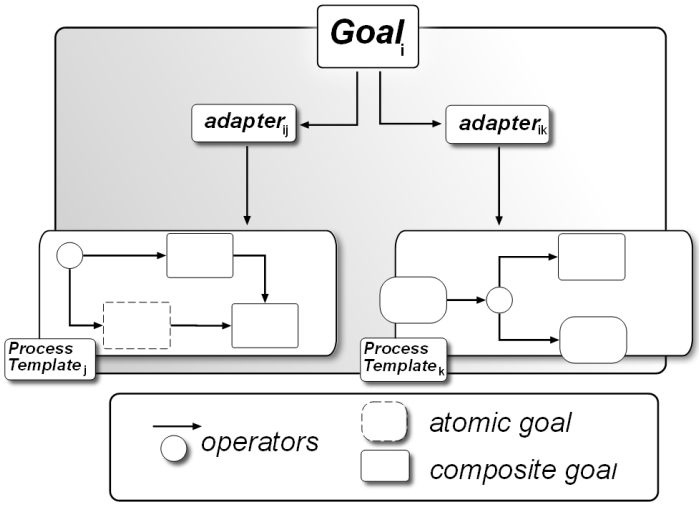


Figure 12 The relationship between Goals and Process Templates.

3.1.3.4 Design Modification Agents

Design Modification Agents are always wait patiently until in some of the blackboards that they are aware of appears a design model that they can modify. The behaviour of all the design modification agents in the multi-agent parametric design problem is determined by same task for determining the design model to change.

The effectors that Design Modification Agent posses are design operator. As we have described, when certain circumstances hold, they apply one design operator to a given design, obtaining one or several new designs. Design operators are transitions between design models [7]. We portrayed them in Figure 7 as links between the spheres that represent the design models. A design operator thus is an action taken by a design transformation agent that from a given design model obtains a new one by altering some of its parameters. This generated design hopefully lead us to a design model closer to a valid state, but we cannot assure it beforehand.

The behaviour of each of the design modification agents is a modification of the one described depicted in [7] for parametric design. We will depict this process using sequence diagrams in the 1st Prototype Design section.

In the first prototype, we will use just two design operators, namely the semantic link design operator and the goal decomposer design operator. As the design-time composition environment evolves, we will continue plugging in new ones. Let us describe the short yet only initial catalogue of design operators that we plan to use.

3.1.3.4.1 Semantic Link Design Operator

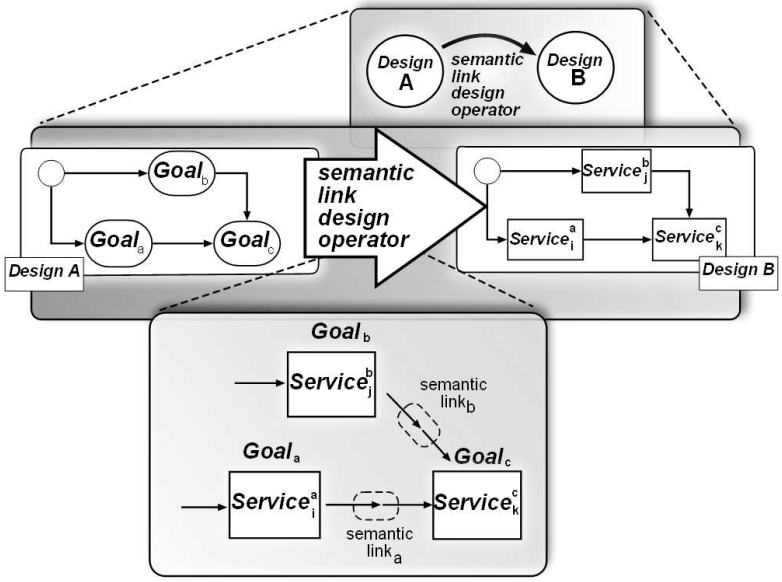


Figure 13 The Semantic Link Design Operator.

The Semantic Link Design Operator, as shown in *Figure 13*, takes as input a process template which actions are fixed atomic goals (we refer to them as goals); and binds those atomic goals to fixed services using the semantic link based algorithm described in [11]. Therefore, services s are assigned to achieve a goal G in case

- s satisfies the latter goals
- some semantic links between s and previous/next goals in the composition exists.

Input and output parameters of services are associated with concepts in a common ontology, or terminology T , where the WSMO-Lite capability [12] and SAWSDL[13] is used to describe them (through semantic annotations). According to this level of description Web service composition consists in retrieving some semantic links [11] noted $sl_{i,j}$ (Figure 14) i.e.,

$$sl_{i,j} := \left\langle s_i, Sim_T(Out\_s_i, In\_s_j), s_j \right\rangle \qquad (1)$$

between output parameters $Out\_s_i \in T$ and input parameters $In\_s_j \in T$ of other services s_j. Thereby s_i and s_j are partially linked according to a matching function Sim_T. Given a terminology T, [14] and [15] value the range of the latter function along five matching types: Exact i.e., $Out\_s_i \equiv In\_s_j$, PlugIn i.e., $Out\_s_i \subset In\_s_j$, Subsume i.e., $Out\_s_i \supset In\_s_j$, Intersection i.e., $\neg(Out\_s_i \cap In\_s_j \subseteq \perp)$, Disjoint i.e., $Out\_s_i \cap In\_s_j \subseteq \perp$.

Example (Matching Type): Suppose $sl_{1,2}$ (like in *Figure 14*) be a semantic link between two services s_1 and s_2 such that the output parameter NetworkConnection of s_1 is (semantic) linked to the input parameter SlowNetworkConnection of s_2. According to Figure 17 this link is valued by a Subsume matching type since $\text{NetworkConnection} \supset \text{SlowNetworkConnection}$.

The matching function Sim_T enables, at design time, finding some levels of semantic compatibilities (i.e., Exact, PlugIn, Subsume, Intersection) and incompatibilities (i.e., Disjoint) among independently defined service descriptions.

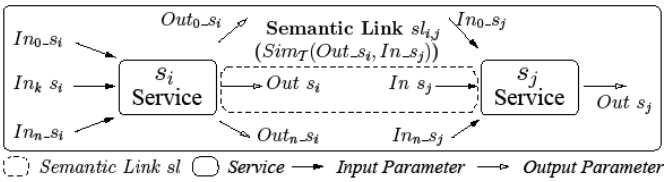


Figure 14 Semantic Link $sl_{i,j}$ between service s_i and s_j

Therefore, this design operator transforms an incomplete design in to a complete one, mainly by considering and automating data flow in Web service composition.

3.1.3.4.2 Goals Decomposer Design Operator

This design operator is quite similar to the design plan generators that we have already described. They substitute composite goals of the design structure. Therefore, they follow a similar approach, using goals, adapters and process templates to transform abstract processes into more concrete ones. Note that the final endeavour of the synthesis phase is to transform the initial design structure formed by abstract elements in to a process described only in terms of concrete activities (i.e. it does not contain any composite goal). This design operator seeks precisely that, to use context information to make user design a little more concrete, closer to be executable (and likely more complex).

3.1.4 Design Analysis Agents

Design analysis agents, once a new design modification agent has generated a new design model inside the blackboard, analyze where the design fits in the taxonomy of possible designs presented in Figure 9. In case that the design model the agent finds a solution, the design analysis agent notifies this fact to the design blackboard control agent.

The idea is that as we manage a knowledge-level representation of designs models, and as [8] blackboard systems allow multiple lines of reasoning and reasoning techniques of representations and logical representation and reasoning mechanisms; we can have several type of analysis agents that perform their tests in different ways (e.g. using model checking techniques [16], constraint processing [17], etc.)

3.2 Optimizing Semantic Web Service Composition

In this section, we focus on optimizing Web service composition at functional level. In such a level, we consider a Web service composition where a set of services is composed to achieve a goal based on the semantic similarities between input and output parameters as indicators of service functionality.

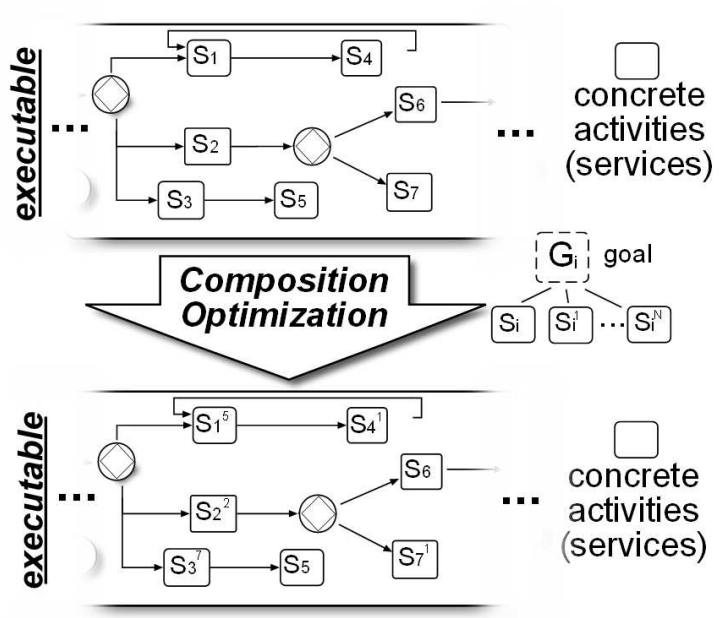


Figure 15 Process Optimization recombining services of lightweight process.

To this end, we consider the template based compositions (provided in the previous section) where their underlying goals (also known as tasks or patterns) can be assigned by different services. The motivation of the composition optimizer is as following: “retrieving a service assigned to each goal of the template based composition such that the i) similarity between output and input parameters (also known as the data flow in Web service composition) and ii) non functional quality of services is maximal”.

To measure semantic similarity, we use the concept of (functional) semantic link [11], defined as a semantic connection between an output and an input parameter of two services. Web service compositions could thus be estimated and ranked not only along well known non functional parameters such as Quality of Services (QoS) [18] but also along the dimension of semantic similarity as indicator of functional fit [19]. In this section, we propose to unify both types of criteria in an innovative and extensible model allowing us to estimate and optimise the quality of service compositions.

Maximizing the quality of service composition using this model is essentially a multi-objective optimization problem with constraints on quality of services and semantic links, which is known to be NP-hard [20]. Most approaches in the literature addressing optimization in Web service composition use Integer linear Programming (IP) e.g., [21]. However, IP approaches have been shown to have poor scalability in terms of time taken to compute optimal compositions when the size of the initial set of services grows. Such a case can arise in the future Semantic Web, where a large number of semantic services will be accessible globally. This is the vision of SOA4All. Rapid computation of optimal compositions is especially important for interactive systems providing service composition facilities for end users, where long delays may be unacceptable. In this section, we demonstrate that the optimisation problem can be automated in a more scalable manner using Genetic Algorithms (GAs), and propose an approach to tackle QoS-aware semantic Web service composition.

In the section 4.3.2 we briefly review i) semantic links, ii) their common descriptions and iii) the Web service composition model. Section 4.3.3 introduces the quality criteria for QoS-aware semantic Web service composition. Section 4.4.4 details the GA-based evolutionary approach, including the strategies of the crossover, mutation and fitness function. Finally, section 4.4.5 draws some conclusions.

3.2.1 Background

In this section, we describe how we use semantic links to model Web service composition. In addition, we remind the definition of Common Description in semantic links.

3.2.1.1 Common Description of a Semantic Link

Besides computing the matching type of a semantic link, [22] suggest to computing a finer level of information i.e., the Extra and Common Descriptions between Out_s_i and In_s_j of a semantic link $sl_{i,j}$. They adapt the definition of syntactic difference [23]for comparing *ALE* DL descriptions and then obtaining a compact representation. The Extra Description $In\_s_j \setminus Out\_s_i$:

$$In\_s_j \setminus Out\_s_i := \min\{E \mid E \cap Out\_s_i \equiv In\_s_j \cap Out\_s_i\} \qquad (2)$$

Refers to information required by In_s_j but not provided by Out_s_i to ensure a correct data flow between s_i and s_j. The Common Description of Out_s_i and In_s_j, defining as their Least Common Subsumer [24] lcs, refers to information required by In_s_j and provided by Out_s_i. by Out_s_i. In case $Out\_s_i \cap \neg In\_s_j \subseteq \bot$, $In\_s_j \setminus Out\_s_i$ is replaced by its more general form i.e., $In\_s_j \setminus lcs(In\_s_j, Out\_s_i)$.

$$NetworkConnection \equiv \forall netPro.Provider \sqcap \forall netSpeed.Speed$$
$$SlowNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.Adsl1M$$
$$Adsl1M \equiv Speed \sqcap \forall mBytes.1M$$

Figure 16. Sample of an ALE Domain Ontology T.

Example: (Extra & Common Description)

Suppose $sl_{1,2}$ in previous Example. On the one hand the Extra Description missing in NetworkConnection to be used by the input parameter SlowNetworkConnection is defined by $\text{SlowNetworkConnection} \setminus \text{NetworkConnection}$ i.e., $\forall netSpeed.Adsl1M$. On the other hand the Common Description is defined by $\text{lcs}(\text{SlowNetworkConnection}, \text{NetworkConnection})$ i.e., NetworkConnection.

3.2.1.2 Modelling Semantic Web Service Composition along Template based Web service composition

Here we remind how semantic links are represented in the template based Web service composition. The process model of Web service composition and its semantic links is specified by a state chart [25]. Its states refer to services whereas we label its transitions with semantic links. In addition some basic composition constructs such as sequence, conditional branching (i.e., OR-Branching), structured loops, concurrent threads (i.e., AND-Branching), and inter-thread synchronization can be found. To simplify the presentation, we initially assume that all considered state charts are acyclic and consists of only sequences, OR-Branching and AND-Branching.

Example (Process Model of a Web Service Composition):

Suppose a composition (Figure 17) extending Example 1 with six more services $s_{i,1\le i\le 8}$, eight more semantic links $sl_{i,j}$. Its process model consists of sequences, OR-, AND-Branching.

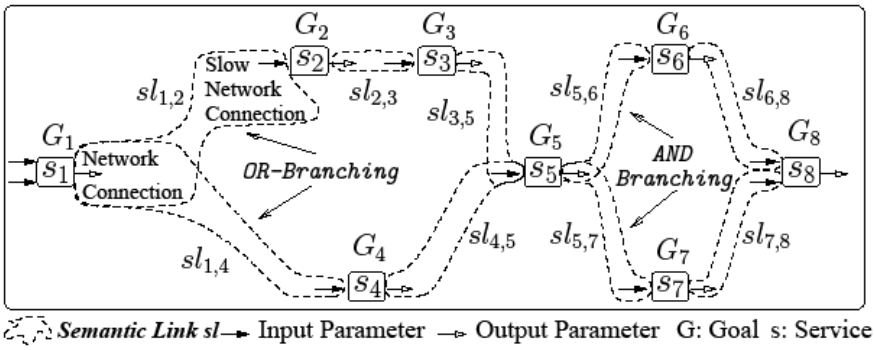


Figure 17 A (Concrete) Web Service Composition.

The previous example illustrates a composition wherein goals G_i and abstract semantic link $sl_{i,j}^{A}$ have been respectively concretized by one of their n candidate services (e.g., s_i) and n^2 candidate links (e.g., $sl_{i,j}$). Indeed some services with common functionality, preconditions and effects although different input, output parameters and quality (e.g., QoS) can be selected to perform a target goal G_i and obtaining a concrete composition. Such a selection will have a direct impact on semantic links involved in the concrete composition.

In the following, we address the issue of selecting and composing a large and changing collection of services. We will make the choice of services at composition time, based on both quality of i) services and ii) their semantic links.

3.2.2 Quality Model

First of all we present a quality criterion to value semantic links. Then we suggest extending it with the non functional QoS to estimate both quality levels of any compositions.

Quality of Semantic Links

We consider two generic quality criteria for semantic links $sl_{i,j}$ defined by $\langle s_i, Sim_T(Out\_s_i, In\_s_j), s_j \rangle$:

- **Common Description rate** $q_{cd}(sl_{i,j}) \in (0,1]$ is defined by:

$$q_{cd}(sl_{i,j}) := \frac{|\, lcs(Out\_s_i, In\_s_j)\,|}{|\, In\_s_j \setminus Out\_s_i\,| + |\, lcs(Out\_s_i, In\_s_j)\,|} \qquad (3)$$

q_{cd} estimates the rate of descriptions which is well specified for ensuring a correct data flow between s_i and s_j.

In more details |.| refers to the size of *ALE* concept descriptions ([26] p.17) i.e., |T|, |⊥|, |A|, |¬A|, |∃r| are 1; |C⊓ D| := |C|+|D|;|∀r.C| and |∃r.C| are 1 + |C|. For instance |Adsl1M| is 3 in Figure 17.

- **Matching Quality** q_m of a semantic link $sl_{i,j}$ is a value in (0; 1] defined by $Sim_T(Out\_s_i, In\_s_j)$ i.e., either 1 (Exact), $\frac{3}{4}$ (PlugIn), $\frac{1}{2}$ (Subsume) and

$\frac{1}{4}$(Intersection). Contrary to q_{cd}, q_m does not estimate similarity between the parameters of semantic links but gives a general overview (discretized values) of their semantic relationships.

In case we consider $Out\_s_i \cap In\_s_j$ to be not satisfiable, it is straightforward to extend and adapt our quality model by i) computing contraction [27] between Out_s_i and In_s_j, and ii) valuing the Disjoint matching type. Thus, the two quality criteria can be update in consequence.

Given the above quality criteria, the quality vector of a semantic link $sl_{i,j}$ is defined as follows:

$$q(sl_{i,j}) := (q_{cd}(sl_{i,j}), q_m(sl_{i,j})) \quad (4)$$

Example. (Quality of Semantic Links)

Let $s_2^{'}$ be another candidate service for G_2 in Figure 17 with NetworkConnection as an input. The link $sl_{1,2}^{'}$ between s_1 and $s_2^{'}$ is better than $sl_{1,2}^{'}$ since $q(sl_{1,2}^{'}) > q(sl_{1,2})$.

In case s_i and s_j are related by more than one link, the value of each criterion is retrieved by computing their average.

3.2.2.1 QoS-Extended Quality of Semantic Link

Here we extend the latter quality model by also exploiting the NF properties of services (also known as QoS attributes [28]) involved in each semantic link. We simplify the presentation by considering only:

- **Execution Price** $q_{pr}(s_i) \in R^+$ of service s_i i.e., the fee requested by the service provider for invoking it.
- **Response Time** $q_t(s_i) \in R^+$ of service s_i i.e., the expected delay between the request and result moments.

The latter values of execution price and response time are given by service providers or third parties.

We will define a quality vector of a service s_i as follows:

$$q(s_i) := (q_{pr}(s_i), q_t(s_i)) \quad (5)$$

Thus a QoS-extended quality vector of a semantic link $sl_{i,j}$:

$$\overset{*}{q}(sl_{i,j}) := (q(s_i), q(sl_{i,j}), q(s_j)) \quad (6)$$

Given an abstract link between goals G_i, G_j, one may select the link with the best matching quality, common description rate, the cheapest and fastest services, or may be a compromise between the four by means of (6). Moreover, the selection could be influenced by predefining some constraints e.g., a service response time lower than a given value.

Example (QoS-Extended Quality of Semantic Link)

Suppose G_2 and its two candidate services s_2 and $s_2^{'}$ wherein $q(s_2^{'}) < q(s_2)$. According to the latter example , $s_2^{'}$ should be preferred regarding the quality of its semantic link with s_1,

whereas s_2 should be preferred regarding its QoS. So what about the best candidate for $sl^A_{1,2}$ regarding both criteria: $\overset{*}{q}$?

3.2.2.2 Quality of Composition

Here we describe how to compute the quality of any concrete composition (Figure 19), given the quality of its underlying services and semantic links (here s and sl sl stand respectively for s_i and $sl_{i,j}$). The approach for computing semantic quality of such a composition c is adapted from the application-driven heuristics of [19], while the computation of its NF QoS is similar to [29].

Common Description rate Q_{cd} of both a sequential and AND-Branching composition c is defined as the average of its semantic links' common description rate $q_{cd}(sl)$. The common description rate of an OR-Branching composition is a sum of $q_{cd}(sl)$ weighted by p_{sl} i.e., the probability that semantic link sl be chosen at run time. The composition designer initializes such probabilities; and may eventually update them considering the information obtained by monitoring the workflow executions.

- **Execution Price** Q_{pr} of a sequential and AND-Branching composition c is a sum of every service's execution price $q_{pr}(s)$. The execution price of an OR-Branching composition c is defined in the same way as $Q_{cd}(c)$, by changing $q_{cd}(c)$ by $q_{pr}(s)$.
- Details for computing **Matching Quality** $Q_m(c)$ and **Response Time** $Q_t(c)$ can be found in Figure 18.

Using the above aggregation rules, the quality vector of any concrete composition can be defined by (7). Contrary to QoS criteria $Q_{l,l\in\{t,pr\}}$, the higher $Q_{l,l\in\{cd,m\}}$ the higher its l^{th} quality for semantic criterion.

$$Q(c) := (Q_{cd}(c), Q_m(c), Q_t(c), Q_{pr}(c)) \quad (7)$$

Definitions (4), (5), (6) as well as (7) can be extended by considering further quality of semantic links and services.

Composition Construct	Quality Criterion			
	Semantic		Non Functional	
	Q_{cd}	Q_m	Q_t	Q_{pr}
Sequential/ AND- Branching	$\frac{1}{\lvert sl\rvert}\sum_{sl} q_{cd}(sl)$	$\prod_{sl} q_m(sl)$	$\frac{\sum_s q_t(s)}{\max_s q_t(s)}$	$\sum_s q_{pr}(s)$
OR-Branching	$\sum_{sl} q_{cd}(sl).p_{sl}$	$\sum_{sl} q_m(sl).p_{sl}$	$\sum_s q_t(s).p_s$	$\sum_s q_{pr}(s).p_s$

Figure 18 Quality Aggregation Rules for Service Composition.

3.2.3 A Genetic Algorithm Based Optimization

The optimization problem i.e., determining the best set of services of a composition with respect to some quality constraints, is NP-hard. In case the number of goals and candidate services are respectively n and m, the naive approach considers an exhaustive search of the optimal composition among all the m^n concrete compositions. Since such an approach is impractical for large-scale composition, we address this issue by presenting a GA-based approach [30] which i) supports constraints not only on QoS but also on quality of semantic links and ii) requires the set of selected services as a solution to maximize a given objective. In this section, compositions will refer to their concrete form.

3.2.3.1 GA Parameters for Optimizing Composition

By applying a GA-based approach the optimal solution (represented by its genotype) is determined by simulating the evolution of an initial population (through generation) until survival of best-fitted individuals (here compositions) satisfying some constraints. The survivors are obtained by crossover, mutation, selection of compositions from previous generations. Details of GA parameterization follow:

- **Genotype**: it is defined by an array of integer. The number of items is equal to the number goals involved in the composition. Each item, in turn, contains an index to an array of candidate services matching that goal. Each composition, as a potential solution of the optimization problem, can be encoded using this genotype.

Example. (Genotype and Composition)

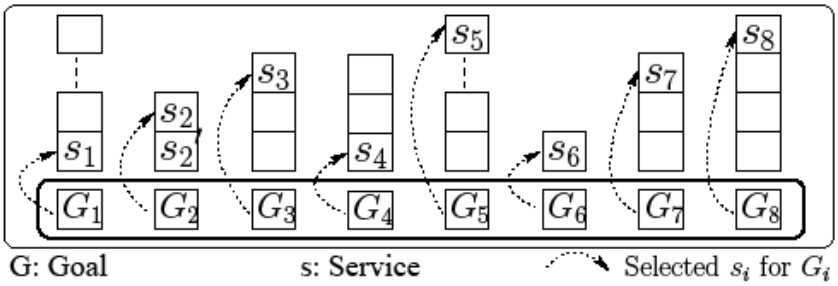


Figure 19 Genotype Encoding for Service Composition.

- **Initial Population**: it consists of an initial set of compositions (characterized by their genotypes) wherein services are randomly selected.
- **Global and Local Constraints**. They have to be met by compositions c e.g., $Q_{cd}(c) > 0.8$.
- **Fitness Function**: this function is required to quantify the "quality" of any composition c. Such a function f needs to maximize semantic quality attributes, while minimizing the QoS attributes of c:

$$f(c) = \frac{\omega_{cd}\overline{Q_{cd}}(c) + \omega_m\overline{Q_m}(c)}{\omega_{pr}\overline{Q_{pr}}(c) + \omega_t\overline{Q_t}(c)} \quad (8)$$

where $\overline{Q}_{l\in\{pr,t,cd,m\}}$ refer to Q_l normalized in the interval $[0,1]$. $\omega_l \in [0,1]$ is the weight assigned to the

l^{th} quality criterion and $\sum_{l\in\{pr,t,cd,m\}}\omega_l = 1$. In this way preferences on quality of the desired compositions can be done by simply adjusting ω_l e.g., the Common Description rate could be weighted higher.

In addition f must drive the evolution towards constraint satisfaction. To this end compositions that do not meet the constraints are penalized by extending (8) with respect to (9).

$$f'(c) = f(c) - \omega_{pe}\sum_{l\in\{pr,t,cd,m\}}\left(\frac{\Delta\overline{Q_l}(c)}{\overline{Q_l^{\max}}(c) + \overline{Q_l^{\min}}(c)}\right)^2 \quad (9)$$

where $\overline{Q_l^{\max}}$ and $\overline{Q_l^{\min}}$ are respectively the maximum and minimal value of the l^{th} quality constraint, ω_{pe} weights the penalty factor and $\Delta\overline{Q_l}$ is defined by:

$$\Delta\overline{Q_l} = \begin{cases} \overline{Q_l} - \overline{Q_l^{\max}} if\ \overline{Q_l} > \overline{Q_l^{\max}} \\ 0 if\ \overline{Q^{\min}}_l \le \overline{Q_l} \le \overline{Q_l^{\max}} \\ \overline{Q^{\min}}_l - \overline{Q_l} if\ \overline{Q^{\min}}_l > \overline{Q_l} \end{cases} \quad (10)$$

- **Operators on Genotypes**. They define authorized alterations on genotypes not only to ensure evolution of compositions' population along generations but also to prevent convergence to local optimum. We use i) composition mutation i.e., randomly selection of a goal (i.e., a position in the genotype) and its service, ii) the standard two-points crossover i.e., randomly combination of two compositions, iii) selection of compositions which is fitness-based i.e., compositions disobeying constraints are selected proportionally from previous generations.
- **Stopping Criterion**. It enables to stop the evolution of a population. First of all we iterate until the constraints are met (i.e., $\Delta Q_l = 0 \forall l \in \{pr, t, cd, m\}$) within a maximum number of generations. Once the latter constraints are satisfied we iterate until the best fitness composition remains unchanged for a given number of generations.

3.2.3.2 Executing GA for Optimizing Composition

Given these parameters, the execution of the GA consists in i) defining the initial population (as a set of compositions), and computing the fitness function (evaluation criterion) of each composition, ii) evolving the population by applying mutation and crossover of compositions (Goals with only one candidate service are disregarded), iii) selecting compositions, iv) evaluating compositions of the population, and v) back to step (ii) if the stopping criterion is not satisfied.

In case no solution exists, users may relax constraints of the optimization problem. Instead, fuzzy logic could be used to address the imprecision in specifying quality constraints, estimating quality values and expressing composition quality.

3.2.4 Conclusion

In this section we studied how to optimize Web service composition, given a template based Web service composition. To this end we focused on QoS-aware semantic Web service composition. This approach has been directed to meet the main challenges facing this problem i.e., how to effectively compute optimal compositions of QoS-aware Web services by considering their semantic links. First of all we have presented an innovative and extensible model to evaluate quality of i) Web services (QoS), ii) their semantic links, and iii) their compositions. In regards to the latter criteria, the problem is formalized as an optimization problem with multiple constraints. Since one of our main concerns is about optimization of large-scale Web service compositions (i.e., many services can achieve a same functionality), we suggested to follow a GA-based approach which is faster than applying IP.

3.3 Templates Generation

In the context of Task 6.4 we are developing a lightweight process modelling language, which will enable users to design process models and reusable templates. Despite of this, in several complex industrial situations it is not possible to define an "a priori" template for a process, either due to the complexity of the real situation or to the high effort required to formalise such template.

Thus, the problem is to understand what is the typical workflow followed by the various

activities, in order to formalise their sequence in a composed schema

3.3.1 Current approaches and their limitations

Process mining techniques are aiming at abstracting from past task and service instances to induct a new schema, previously unknown, describing all of them. Most of existing state-of-the-art approaches are devoted to identify a single process formalisation, often resulting in particularly complicated schemas and not very accurate especially in cases of processes made by several activities and complex behavioural rules. The resulting complexity is due to the need of deriving a single schema able to explain every event recorded in the logs.

The resulting schema, even if formally complete and adequate to support a process execution, turns out to provide little help to solve the initial problem, that is to let end-users understand what the hidden process schema. This is especially true if such tools are to be used at a managerial level, in order to derive a business-oriented process schema, at a higher level of abstraction.

3.3.2 The Template Generator

In order to overcome such limitations, we plan develop a Template Generator tool which taking service execution logs as inputs is able to generate an hierarchy of process schemas (at different level of complexity/completeness) and a taxonomy of possible process templates (at different level of abstraction), able to support end-users in the selection of the most suitable one.

Such tool will exploit state-of-the-art Process mining, Process Abstraction and Clustering techniques in an innovative way, in order to present the end-users with the most suitable templates representations and let them choose the one that most fits their needs. The selected one, described with the SOA4All lightweight language (task T6.3) can be further validated, adapted or refined by end-users thanks to the SOA4All Process editor developed in T2.6.

In this way, the effort and the complexity for creating new processes can be lowered, and made more accessible to all kind of users. In this sense, such technologies *contribute to the "4ALL"* objectives of the project .

This approach perfectly fits and complements the typical process management cycle, made of a design, execution and verification phase. The following picture shows the role of the Template generator component within the context of SOA4All:

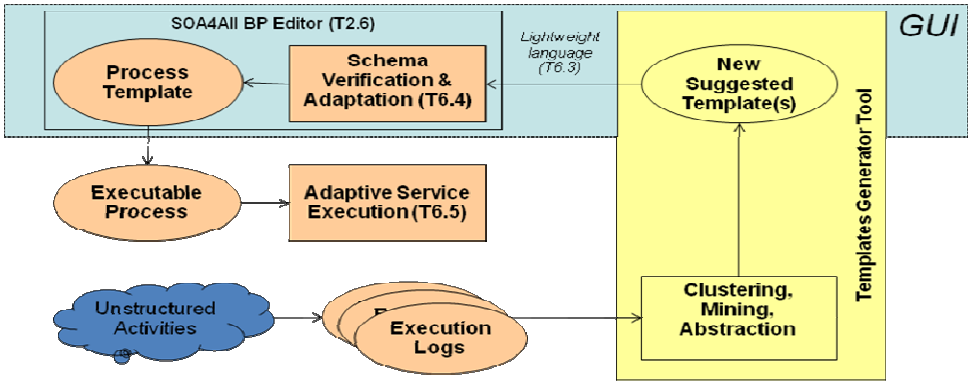


Figure 20: Template Generator typical use-case.

More in detail, the Process Generator will offer the following features:

- It will generate a hierarchy of process schemas at different level of complexity/completeness, and a taxonomy of possible process templates (at different level of abstraction

- It will support end-users to graphically analyse such hierarchy and taxonomy
- It will be able to analyse service execution logs gathered by the Service Analysis tools developed within T2.3, thus taking into account contextual information
- It will be able to generate process templates based on the SOA4All Light-weight language (as defined into T6.3) thus making them directly available for editing in the SOA4All Process editor
- It will be based on the popular open-source ProM framework[4], featuring state-of-the-art workflow discovery techniques, which will be complemented with plug-ins featuring innovative hierarchy mining and process abstraction and clustering algorithms
- We will integrate it in the SOA4All Studio, the user will be able to access the generated processes and adapt them according to her needs.

It should be remarked that this tool is not aiming at advancing research in mining, clustering or abstraction techniques, which already have a good degree of maturity, but rather on an optimal way of using such techniques.

3.3.3 The template generation process

In the following section, we depict the process that we carry out in order to generate process templates from logs of past execution of services. The process (depicted in Figure 21) is composed of the following steps:

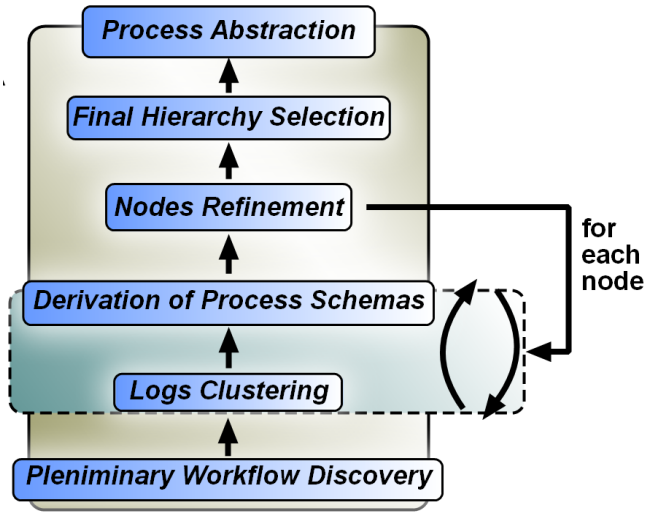


Figure 21 The Workflow of the Process Template Generation Process.

1. ***Preliminary Workflow Discovery*.** As the first step, we generate a preliminary schema for the initial set of logs. In this phase, we will use state-of-the-art *Workflow discovery* techniques, as the ones that are available in literature and available in the ProM framework [31], such as Alpha, Multi-phase, Genetic, Social Network. Anyway, we should stress that such algorithms are not able to induce hierarchical process models.

[4] http://prom.win.tue.nl/tools/prom/

2. **Logs Clustering thanks to Discriminant Rule Extraction methods**. The preliminary schema generated in this way is based on the whole set of logs. In order to detect and to separate meaningful execution scenarios into meaningful set (cluster), we will exploit Discriminant Rule Extraction and Log Clustering techniques (see [32] and [33]). A Discriminant Rule is a process feature about flows within a schema. We can consider them as a global hidden process constraint, which is detectable in the logs. They allow discovering different structural variants of a process. We can find such rules thanks to specific feature mining algorithms. Finally, in order to partition the initial log set into clusters, we project logs in the multi-dimension space induced by such set of discriminant rules, and we apply a numeric data-clustering algorithm on it.
3. **Derivation of Process Schemas for the Clusters.** Once we have a new set of logs arranged into cluster, each of which can be interpreted as a possible different execution of the same process – in this way we are reducing the degree of complexity in the schema, but also the degree of completeness of the schema. Once again, such log clusters can be modelled with a specific workflow schema, using the same Process Mining techniques described in bullet 1) (i.e. ProM tool).
4. **Nodes Refinement.** We can iterate this process and repeat steps 2 and 3 for each node we wish to refine, in order to obtain a final hierarchy of workflow schemas (*Hierarchy Discovery).*The idea is to let the end-user choose the maximum number of iterations.
5. **Final Hierarchy Selection**. We present to the end-user the final hierarchy of schemas: leaves will constitute a disjoint set, which represents the initial log set in a more accurate and expressive way rather than the preliminary schema (root). Indeed, we lose Completeness vs the initial schema, which was aiming at describing all logs. At this point, is up to the user to select the most suitable schema, based on the number of possible situations (i.e. different possible executions) he wants to take into account and based on the complexity of the schema.
6. **Process Abstraction**. So far, we have performed operations in order to achieve a good compromise between complexity and completeness. In order to further improve our process representations and make them fully available for analysis and business planning purposes, we apply a process abstraction methodology, in order to re-structure the knowledge embedded in the various schemas of the hierarchy in a taxonomy of schemas at different level of abstraction. The resulting taxonomy is a tree where leaves describe real process instances and higher-level nodes represent an abstract view on heterogeneous process instances schemas. The approach is to start bottom-up from the hierarchy created at bullet 5), and to modify each non-leaf node, in order to make it an abstraction of the schemas associated to its children. The abstraction technique replaces groups of homogeneous activities with a single, abstract activity (thanks to “is-a” and “part-of” relationships). Finally, end-users will be able to navigate such taxonomy and to visualise the various schemas, based on the abstraction degree they need

The following picture shows in a more detailed way the whole process: from the initial set of logs, the schema complexity is reduced by logs clustering into disjoint set (thus increasing the number of possible schemas) – as shown in the “Process Single schema” box in the picture. Then, abstraction techniques allow to produce a taxonomy of schemas at different level of abstractions (upper part of the picture):

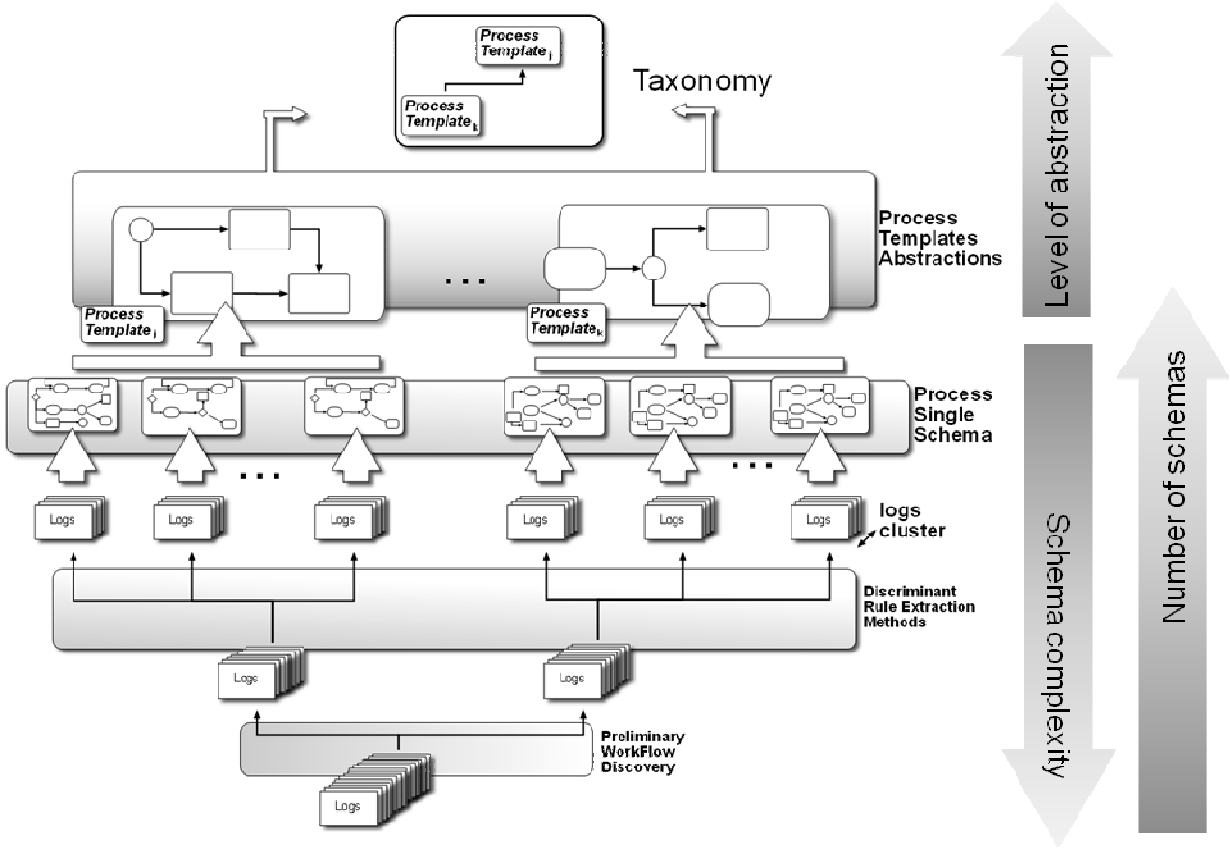


Figure 22 Graphical representation of the template generation process.

4. Design of the 1st Prototype

In this section, we are going to be more concrete than in the previous one devoted to the theoretical grounding. We are going to provide a specification of the software artefacts that compose the first implementation of the prototype of the Service Composition Framework.

In the following table, Table 1, we depict the roadmap of the software that will be following. It specifies the delivery dates of the software associated with this task; and what will be the extend of this software.

Table 1 Task 6.4 Software Roadmap.

Milestone	M12 1st Prot.	M18 2nd Prot.	M24 3rd Prot	M30 (Final Version)
Design-Time Composer	Basic(Standalone & Composition)	Basic(Integrated & Adaptation)	Advanced	Matured
Composition Optimizer		Basic(Integrated)	Advanced	Matured
Templates Generator		Basic(Integrated)	Advanced	Matured

Let us know describe the design of the 1st Prototype of the software, which is the basic version of the design-time composer. We will follow a classic approach to describe the design of the environment. On the one hand, we will sketch its ***structural view***, a picture of the Service Composition Framework, describing the components that constitute it, even those that will not be delivered for the 1st Prototype[5]. After that, we will depict the ***behavioural view*** of the components that we will deliver in the first prototype, how these software modules will interact to achieve the necessary functionality to cope with the requirements listed in the requirements section.

4.1 Structural View of the 1st Prototype

4.1.1 Ubication within SOA4All

In Figure 23 we depict the overall picture of WP6 and where we situate it inside the overall SOA4All architecture. Let us briefly describe the components of the Service Construction environment, in order to situate the software that we are going to describe in this deliverable. 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 we will use as specification language. To improve usability pre-designed and user-designed process templates are stored in the semantic service & template repository.

Once created and stored, in order to be usable and interpretable these lightweight processes have to be translated in to more complex processes that can be interpreted by an execution 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 system will transparently transform the aforementioned lightweight processes in to optimized complex

[5] Note that these descriptions are tentative, should be considered a first draft.

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.

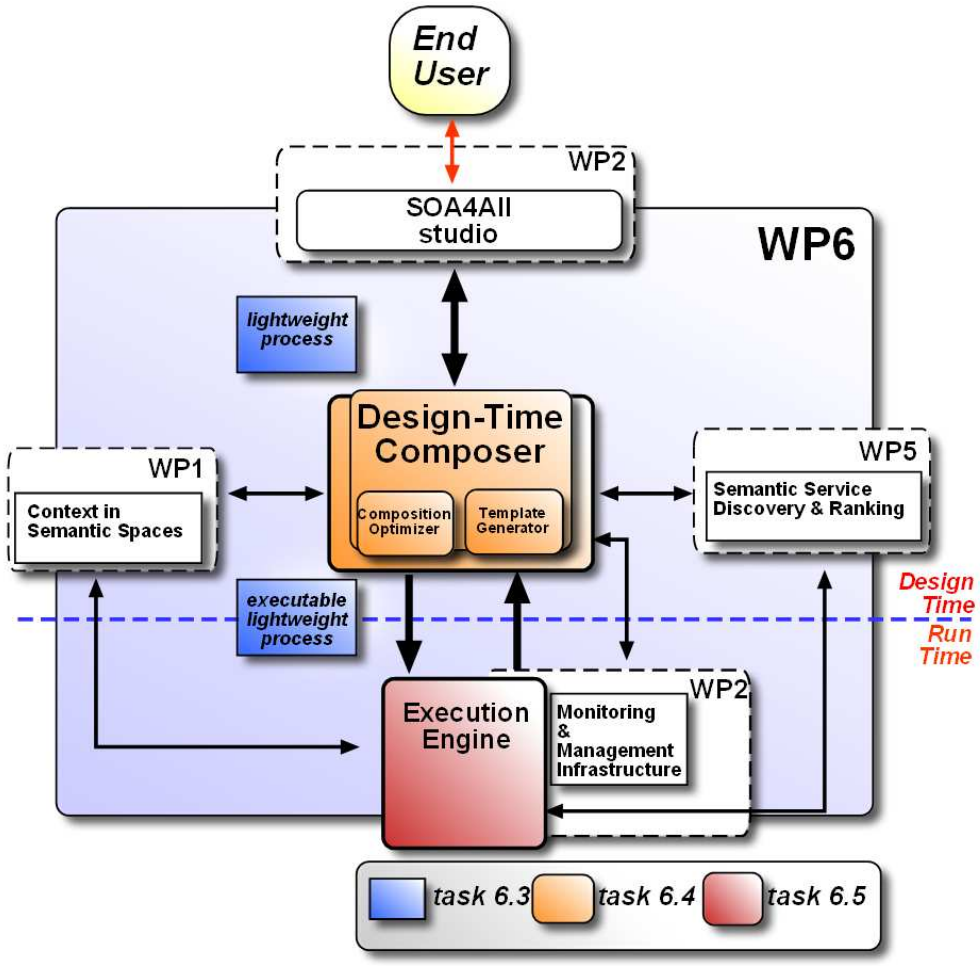


Figure 23 WP6 Overall Picture and its place inside SOA4All.

4.1.2 WP6 Task 6.4 Core Components Description

In the following subsections we are going to describe, each of the components which belong to the task 6.4, which are pictured Figure 24. We depict the software modules, and the initial interfaces that are going to be exposed[6].

[6] Note that as this is ongoing work, some of which will not be delivered until month 18 and there can be slightly changes in the interfaces.

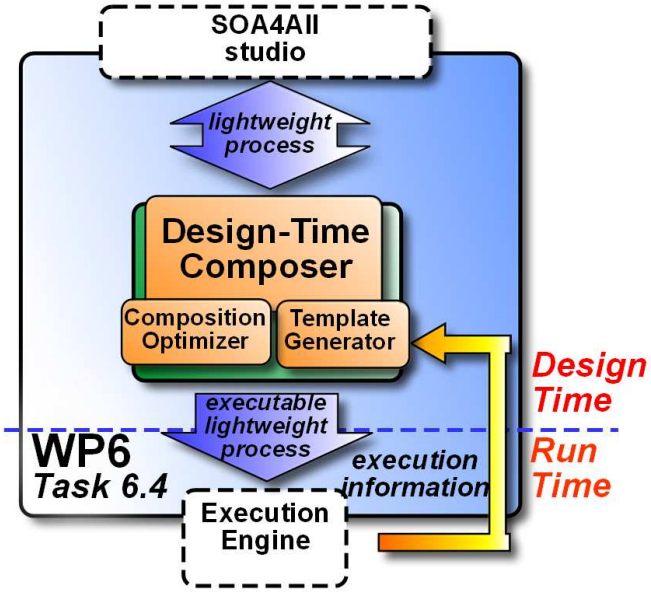


Figure 24 Core components of Task 6.4.

4.1.2.1 Design-Time Composer

- **Name**: Design-Time Composer
- **Description**: The design-time composer component will carry out two closely related activities, namely process adaptation and composition, offering thus two external interfaces, the Service Composer and the Service Adapter.
    - The Service Composer perspective is a scalable system for the flexible and ad hoc creation of complex services, the environmental context information, and user needs (expressed using the lightweight modelling language).
    - The Service Adapter is a subsystem for service context-based adaptation at design-time. The first and basic usage of this tool is the adaptation of services according to context (e.g. personal preferences, business rules, etc.), but also more advanced dynamic adaptation procedures. Mechanisms such as incremental revealing of services descriptions imply that not all the service characteristics have to be revealed at once, but require a reciprocal knowledge, trust and a negotiation process between parties.

  Both activities are closely related, since as we have already described, both composition and adaptation will share the same core of functionalities that are provided by a parametric design engine which will use a catalogue of generic knowledge-level service templates and context-dependant configuration.

- **Inputs**: In both cases the inputs are the same, the difference resides in the nature of the interaction.
    - *A Lightweight Process*. In the case of composition, an abstract process expressed in terms of user's goals.
    - *A set of user requirements*. The user especially in the case of adaptation might also provide additional requirements about the process to generate.
- **Outputs**: The outputs both in the case of composition and adaptation are:
    - *A Lightweight Process*. The result of the composition/adaptation process is a lightweight process, in the case of the composition process it should be an executable one.
    - *A set of assumptions*. Whilst composing the processes, several assumptions

could be made about the context, inputs and outputs of the service, etc. They are as important as the process in itself, and therefore should be forwarded to the user.

- **Interfaces exposed**: This component offers two external interfaces, Service Composer and Service Adapter. For this 1st Prototype only the Service Composer will be exposed.

4.1.2.2 Composition Optimizer

- **Name**: Composition Optimizer
- **Description**: The composition optimizer considers an innovative and extensible quality criteria model by coupling non-functional quality of service and semantics of the executable lightweight process. On the one hand, the non-functional criteria of Web services are valued by means of Quality of Services (e.g. execution price, response time, reliability, availability), while on the other semantics is valued along the semantic links (i.e. data flow in an executable lightweight process) between Web services. The latter criterion requires the WSML reasoning framework to i) give an estimation of semantic matching between functional output and input parameters of services and ii) estimate robustness issues (through a non-standard Description Logics inference) in data flow of any executable lightweight process. In regards to the latter criteria the problem is formalized as a Constraint Satisfaction Problem (CSP) with i) multiple constraints and ii) a function to optimize. Towards such an issue we model an optimization problem COP (Constraints Optimization Problem), adapted from CSP. Since one of our main concerns is about optimization of large-scale executable lightweight process (i.e., many services can achieve a same goal or functionality), we suggested to follow a Genetic Algorithm-based approach which is faster than applying Integer Linear Programming.
- **Inputs**: A set of services (in a repository), their functional qualities, a template based Web service composition where tasks are semantically described, a set of constraint to met (in term of Quality of composition: aggregation of Quality of services and semantic links). We also require a reasoning engine to compute semantic similarities between output and input parameters of services.
- **Outputs**: A concrete composition i.e, all tasks of the template based composition are achieved by a unique service. Such a composition is the most optimal composition that met the end users constraints.
- **Interfaces exposed**:
  - o ConcreteComposition optimizeComposition(SetOfService, TemplateComposition, Constraints)

4.1.2.3 Templates Generator

- **Name**: Template Generator
- **Description**: The Template Generator will be able to analyse service execution logs and to generate an hierarchy of process schemas (at different level of complexity/completeness) and a taxonomy of possible process templates (at different level of abstraction), in order to support end-users in the selection of the most suitable one. Such a tool will exploit state-of-the-art process mining, process abstraction and clustering techniques in an innovative way, in order to present the end-users with the most suitable templates representations and let them choose the one that most fits their needs. The selected template (described with the SOA4All light-weight process language - D6.3.1) can be further validated, adapted
- **Inputs**: Past processes and/or services execution logs
- **Outputs**: A new process template, defined in the SOA4All light-weight language (as

defined into T6.3)

- **Interfaces exposed**:
  - GetProcessSchema (log_set, mining_algo_id, store_location): discover a schema expressed in the lightweight BPM language for a given set of logs, using a given mining algorithm, and store it in the semantic space at a given store_location. Returns a schema ID
  - ClusterLogs (log_set, rule_extraction_algo_id, clustering_algo_id, store_location) : cluser logs based on a given discriminant_rule and clustering algorithms. Returns number of logs created (n), and a set of cluster IDs
  - GetClusterByID (cluster_id) : return a log_cluser given its ID
  - Abstract_schema (schema_id, abstraction_algo_id, store_location): performs a process schema abstraction. Return a new schema ID
  - GetSchemaByID (schema_id): returns a schema for a given ID (expressed in light-weight language)
  - DeleteSchemaByID (schema_id): deletes a schema from the Semantic Space
  - DeleteClusterByID (cluster_id): deletes a cluster given its ID from the Semantic Space
- **Interaction with internal components**: Retrieve service execution logs from the DSB monitoring collector component developed in T2.3.
- **Interaction with external components:** The output process is delivered to T6.4 Design Time Composition platform and/or T2.6 Business Process Editor. End-users will be able to further verify/edit/change the output process

4.2 Behavioural View for the 1st Prototype

The use cases of the overall Context-aware Service Composition and Adaptation Environment are depicted in Figure 25

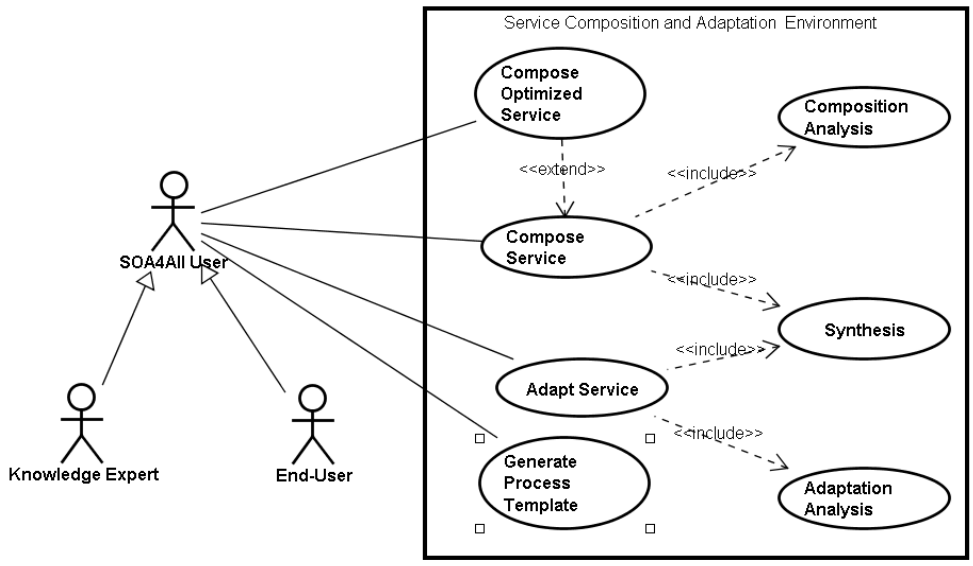


Figure 25 Design-Time Composer Use Cases Diagram.

According with the software roadmap (see Table 1)we will cover the Compose Service use case, which includes the Composition Analysis and the Synthesis subcases. Their textual description can be found in the correspondent subsections related to these activities in the theoretical background introduction. Let us know specify using UML sequence diagrams how

the system should behave.

4.2.1 Compose service

As described in Figure 3 parametric design is decomposed in two secuential phases in the case of process composition. Therefore, we consider this use case as divided in two phases, namely Composition Analysis and Synthesis.

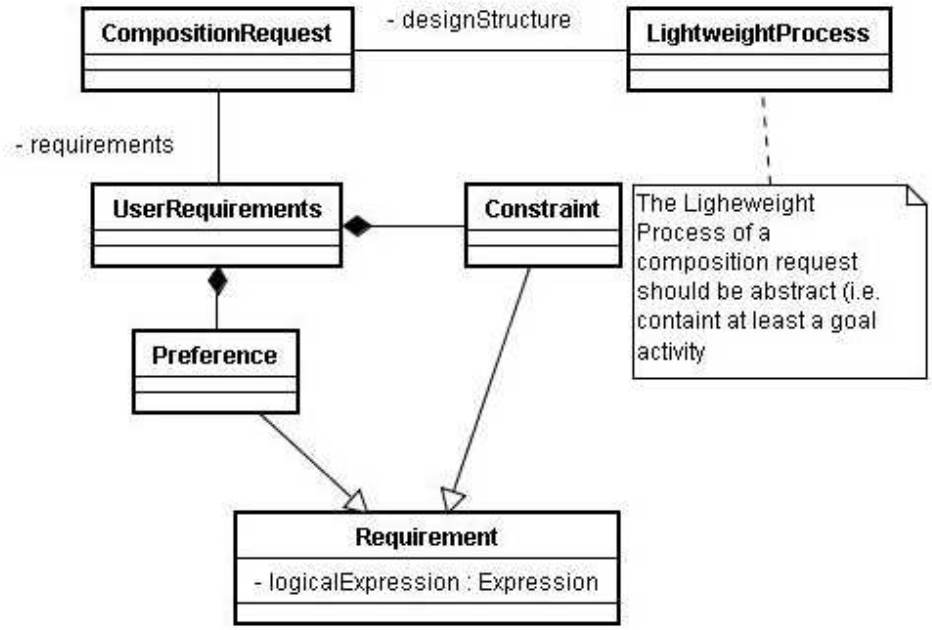


Figure 26 Classes involved in a composition request.

4.2.1.1 Composition Analysis

The following diagram depicts how the different components and agents interact in the synthesis phase of the parametric design phase. The initial structure that is stored in the blackboard is specified in the class diagram presented in Figure 26

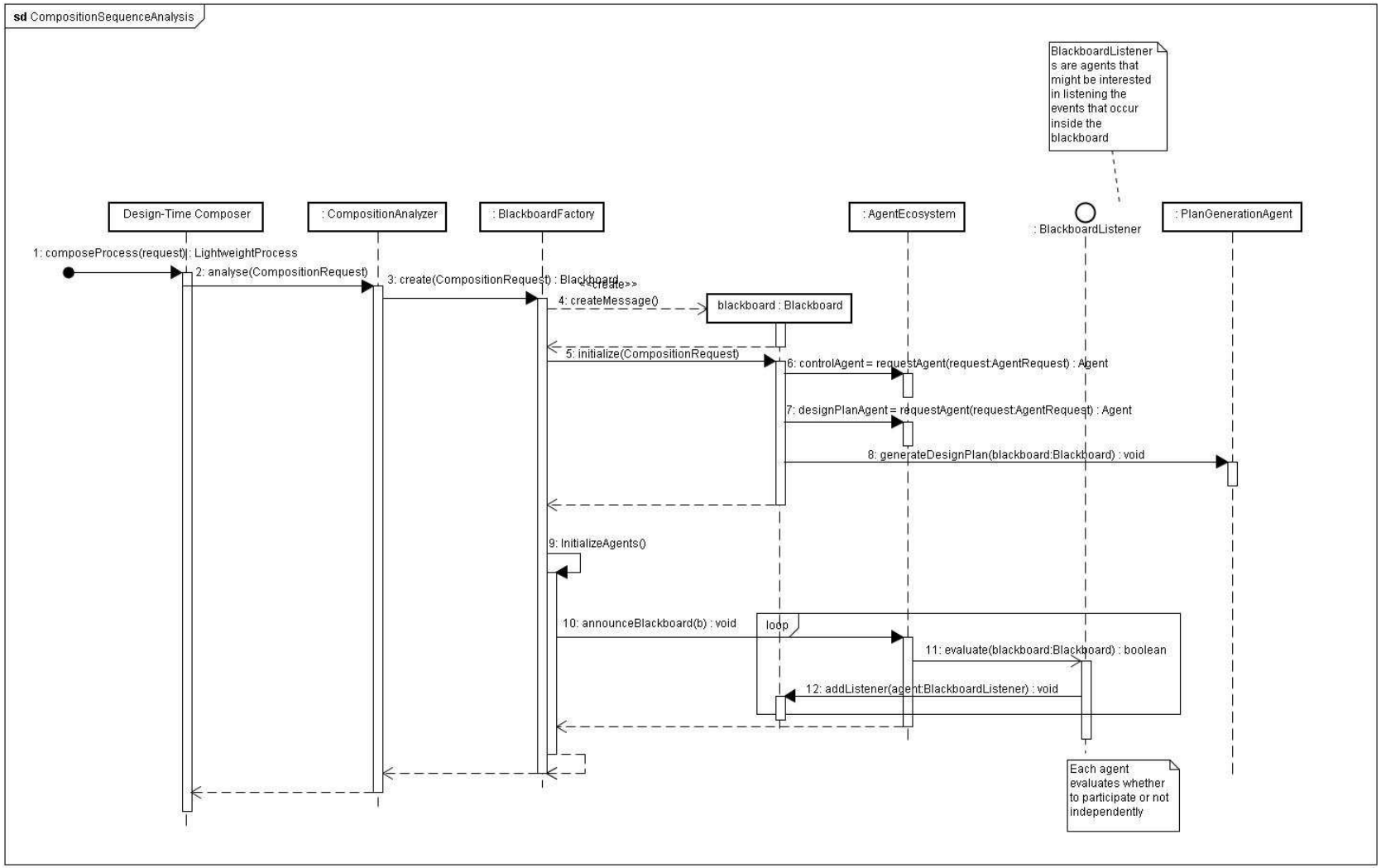


Figure 27 Analysis for Composition Sequence Diagram.

In the analysis phase of the parametric design, we also include the creation of the initial design model using a design plan generator. We will explain the generateDesignPlan method in a separate sequence diagram, where we will depict the steps taken to achieve its creation (Figure 28).

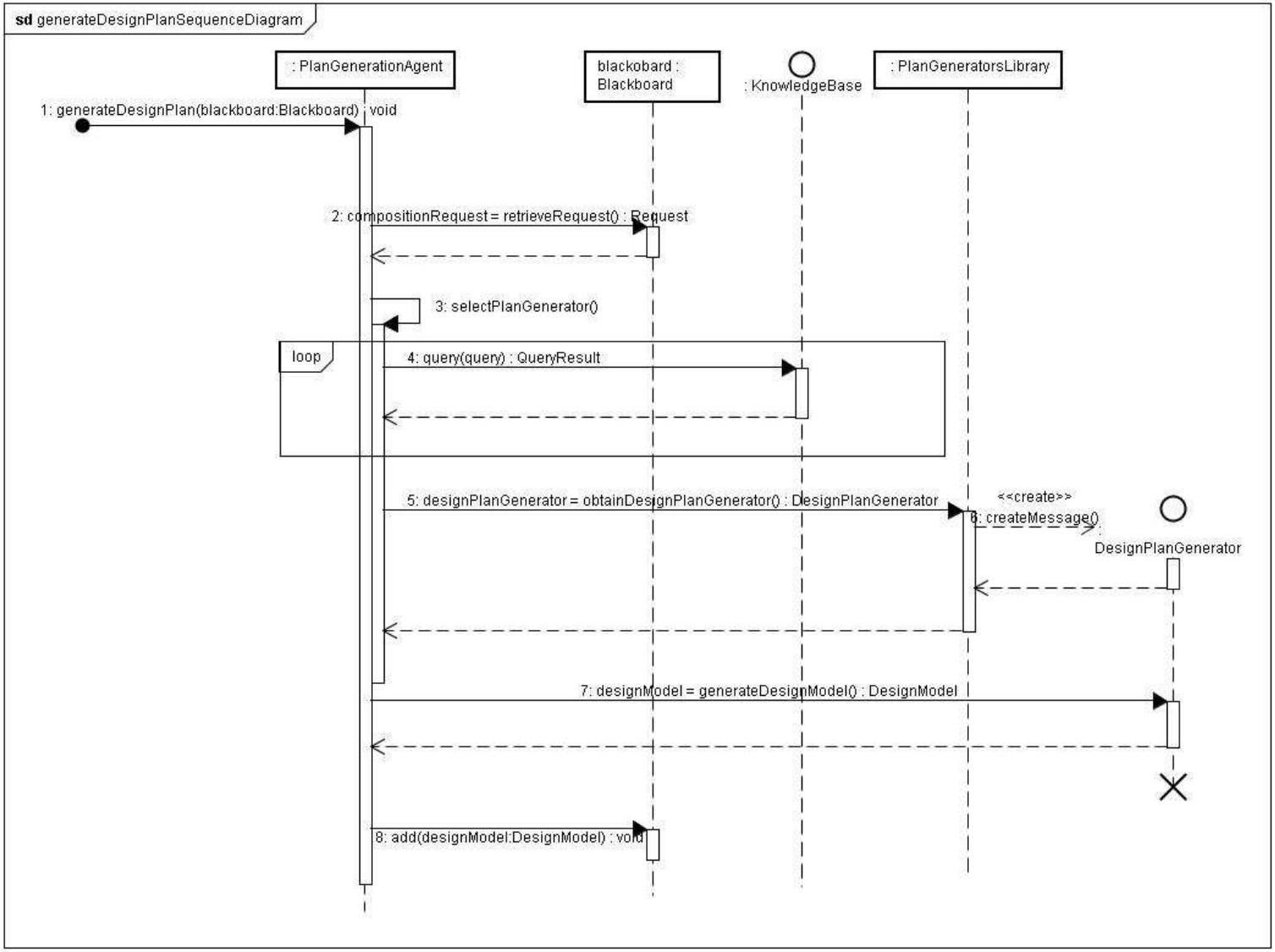


Figure 28 Generation of the Design Plan Sequence Diagram.

The process is rather simple. The agent gets the request from the blackboard, checks its knowledge base that stores both its operational rules, knowledge about the domain and the current state of affairs; and depending on this information chooses one of the available design plan generators from the set of available catalogue. After that, the agent invokes that generator, to obtain an initial design model (which corresponds with the red sphere that appeared in Figure 7).

4.2.1.2 Synthesis

Once we have the initial design model, as described previously, the synthesis phase commences. In Figure 29 we depict the agents involved in this phase (all but the PlanGenerationAgent, that as we have already stated is used in the analysis phase).

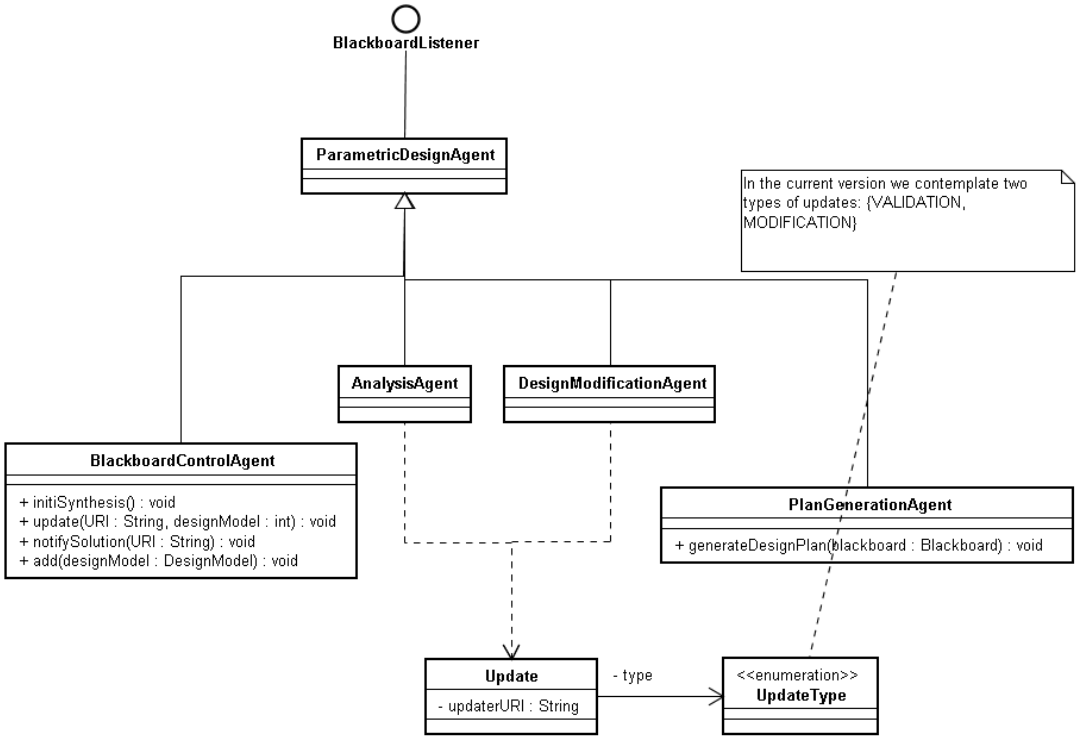


Figure 29 Agents involved in the composition process class diagram.

These agents will carry out the process of synthesis in a collaborative fashion. Let us describe the behaviour of each of them in the following diagrams, each of them will act independently, but the final result will be a design model that can be considered a solution.

4.2.1.2.1 Blackboard Controller Agent Behaviour

In Figure 30 we depict the basic interaction of the blackboard controller agent. We define two basic cases

- The initial case is the moment where the design model that originates the synthesis phase is introduced in the blackboard. The control agent sends this design model to all the interested agents, by searching in its knowledge base.
- The other case, which is the habitual case, corresponds with the case where a new design model has been created in the blackboard on the basis of an already existing one (update(update:Update, oldDesignModel, newDesignModel):void)). The blackboard stores the new design, and relates the initial design model and the new one, tagging this relationship with the URI of the design operator originally used in the transformation (creating thus the DAG depicted in the Figure 7 when we described the parametric design as search abstraction). The blackboard control agent then carries out its duty, informing of the new design to the interested agents, consulting again its knowledge base.

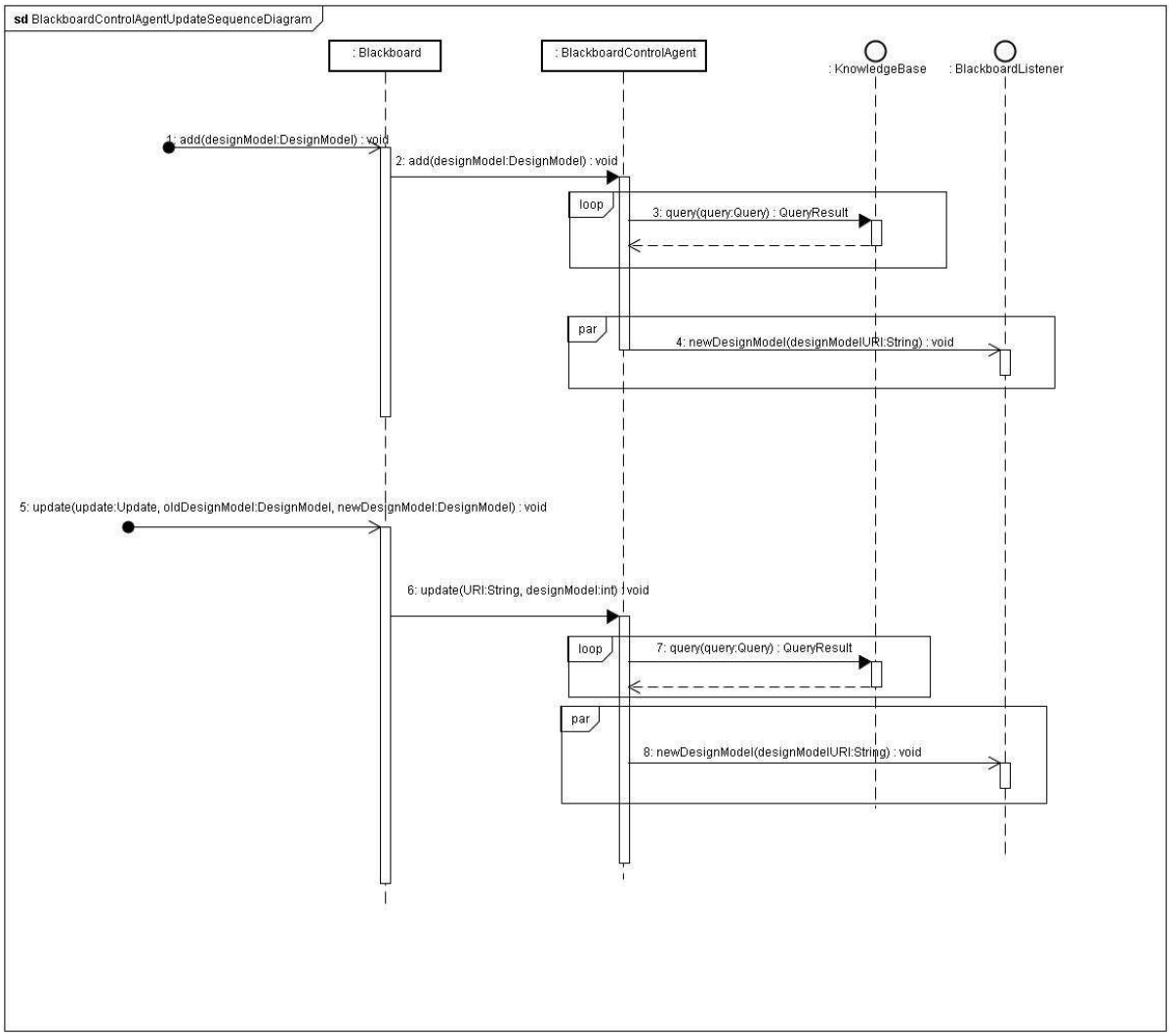


Figure 30 The Blackboard Control Agent Sequence Diagram.

4.2.1.2.2 Analysis Agent Behaviour

The analysis agent, once it receives a new design model, retrieves the model from the blackboard, validates it, and depending on the result of this activity:

- The agent can consider this design model a solution to the problem. It notifies to blackboard control agent that a solution has been found (using void:notifySolution(URI) as shown in the diagram on Figure 31).
- Else, it tags the solution with the validation result, so that other agents can use it in the future to solve the problems that the design model might have. Note that in order to make other agents aware of the addition to this information to the design model; we should communicate to peers that observe the blackboard that the information about this design model has been updated.

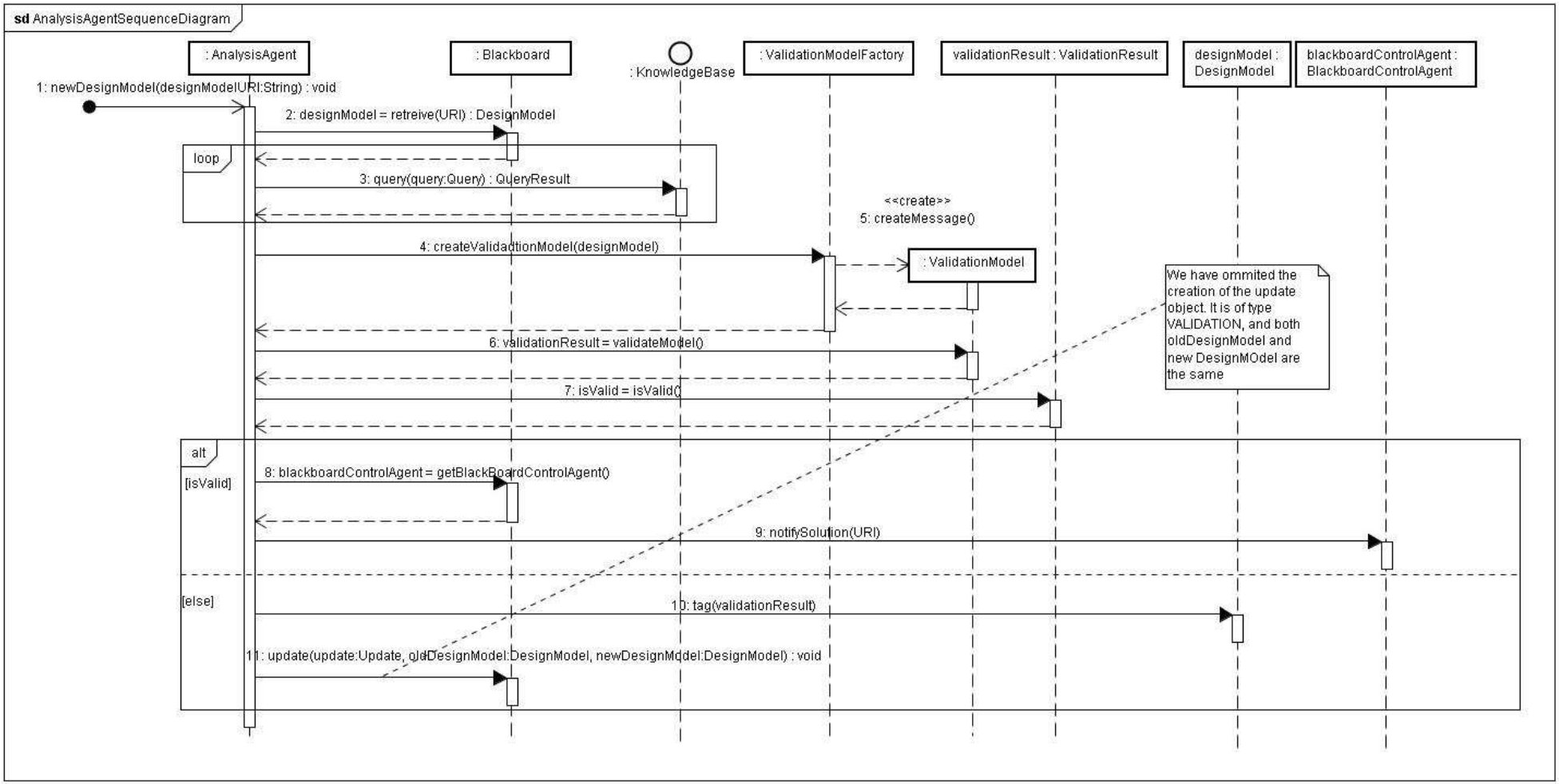


Figure 31 Analysis Agent Sequence Diagram.

4.2.1.2.3 Design Modification Agent Behaviour

The following sequence diagram depicts the behaviour of a design modification agent. Basically it receives a notification of a change in a design model, which in this prototype, as we are depicting, can have two different meanings:

- A design modification agent has changed the design model by means of a design operator.
- An analysis agent has analyzed the design model. In such case, the design modification agent might use this additional information to modify the model more precisely.

For the design modification agent, all the updates are similar. First, the agent must set the its design focus. Although it seems logical that the agent acts upon the newly created design model, it is not mandatory. The agent can chose to modify any other design model that is present in the blackboard.

Once the agent has decided which design model to modify, the agent must choose the action to do upon this design model. The actions that an agent can carry upon design models in the case of design modification agents are design operators. The agent reasons using its knowledge base and in terms of the design model; and decides which design operator to apply.

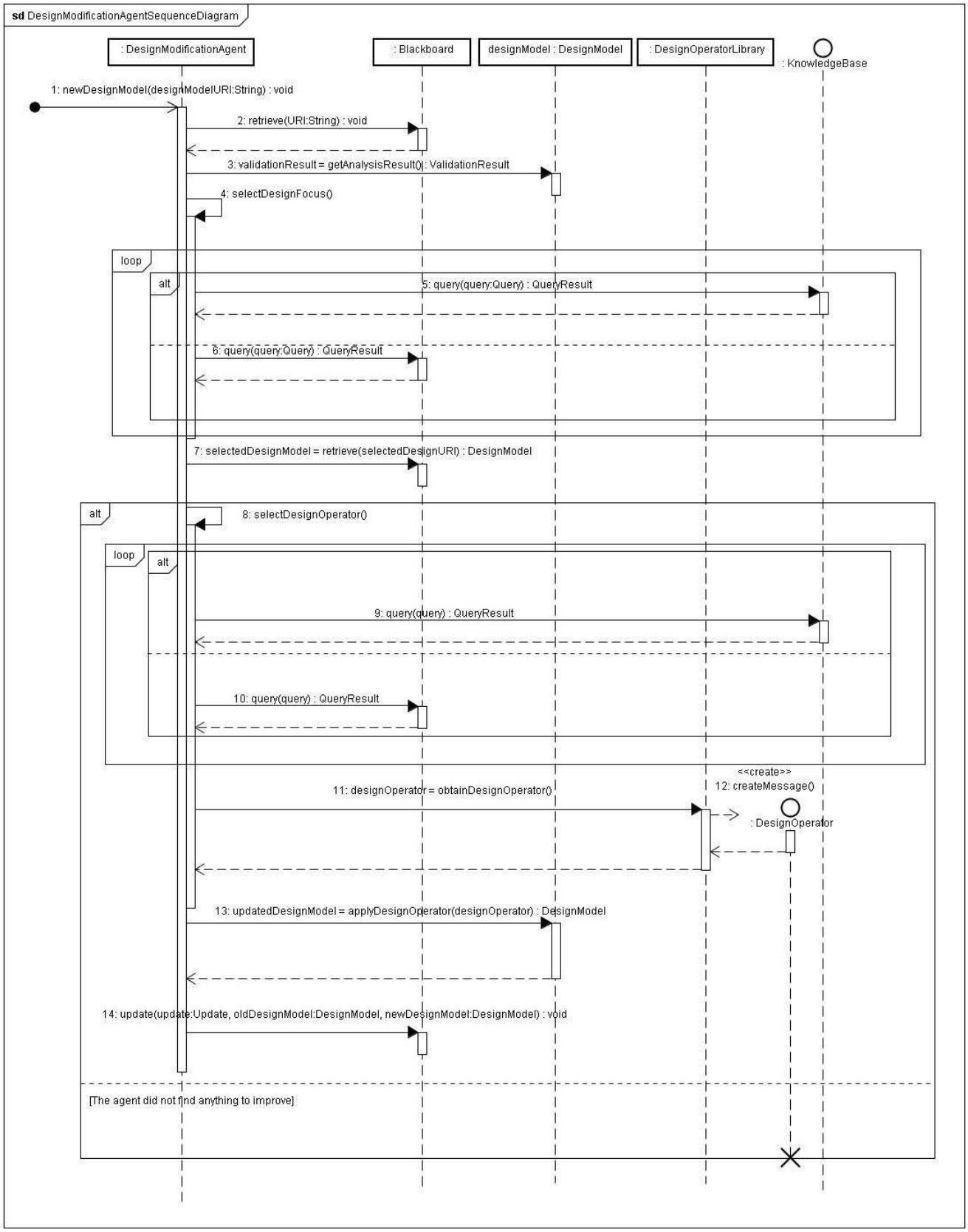


Figure 32 Design Modification Agent Sequence Diagram.

Finally, we must emphasise on the characteristics of the opportunistic approach. The agent,

when receiving the notification, after consulting to its knowledge base might do something; or simply ignore the design model update notification, as we have portrait in the sequence diagram.

5. Conclusions

In the context of WP6 of the SOA4All project the term service construction mainly refers to model complex services in a lightweight mannerThis 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 modeling 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 deliverable, we have addressed the transformation from the user-friendly lightweight representation of processes in to complex and concrete executable processes. We have precisely depicted the algorithms and formal approaches that we will use in order to address this problem, and how to make it easier for non-experts to build the process templates that the composition algorithm will use, and how to make the generated processes more efficient.

Finally, we have sketched the first draft of the design of the Service Composition Framework, which we will use as a blueprint to implement the 1st software prototype. We have included an overall structural view of the whole Service Composition Framework; and the behavioural description of the software modules that will be delivered in the 1st Prototype of the framework.

6. References

[1] European Commission: Handbook on Implementation of the Services Directive, Commission of the European Communities Internal Market and Services DG, Brussels, 2007.

[2] Mittal, S. & Frayman, F. (1989). *Towards a generic model of configuration tasks*. In Proceedings of the 11th IJCAI, pages 1395–1401, San Mateo, CA, Morgan Kaufman.

[3] Yu, B. and MacCallum, K. (1995). *Modelling of Product Configuration Design and Management by Using* Product Structure Knowledge. Int. Workshop on Knowledge Intensive CAD, Finland, 1995.

[4] Wielinga B. J., Akkermans J. M., Schreiber A. Th., *A Formal Analysis of Parametric Design Problem Solving*, In Proceedings of the 9th Banff Knowledge Acquisition Workshop (KAW-95)

[5] Wielinga B., Schreiber G., Configuration-Design Problem Solving, IEEE Expert: Intelligent Systems and Their Applications, v.12 n.2, p.49-56, March 1997.

[6] Chandrasekaran, B. 1990. *Design problem solving: a task analysis*. AI Mag. 11, 4 (Oct. 1990), 59-71.

[7] Motta E., (1999) *Reusable Components For Knowledge Modelling Case Studies In Parametric Design Problem Solving*,IOS Press (Netherlands)

[8] Pedrinaci, C. (2005) *Knowledge-Based Reasoning Over The Web*, PhD. Dissertation, Universidad País Vasco San Sebastián, Noviembre 2005

[9] Erman L. D., Hayes-Roth F., Lesser V. R., and Reddy D. R.. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. Computing Surveys, 12(2):213–253, June 1980.

[10] Carpenter M., Mehandjiev N., Stalker, I.D.: *Flexible Behaviours for Emergent Process Interoperability*. WETICE 2006: 249-254

[11] Lécué F., and Léger A., (2006), 'A formal model for semantic Web service composition', in ISWC 2006, pp. 385–398,

[12] Vitvar T., Kopecký J., Viskova J., Fensel D. (2008) WSMO-Lite Annotations for Web Services. ESWC 2008: 674-689.

[13] Kopecky J,, Vitvar T., Bournez C., and Farrell J.. Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing,11(6):60–67, 2007.

[14] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara, (2002) Semantic matching of Web services capabilities. In ISWC, pages 333–347

[15] L. Li and I. Horrocks. A software framework for matchmaking based on semantic Web technology. In WWW, pages 331–339, 2003.

[16] Clarke E. M., (2000) Model Checking MIT Press (4 Feb 2000) ISBN: 978-0262032704

[17] Dechter R.(2003) Constraint Processing, Morgan Kaufmann, ISBN: 978-1558608900

[18] Canfora G., Di Penta M., Esposito R., and Villani M. L.. An approach for qos-aware service composition based on genetic algorithms. In GECCO, pages 1069–1075, 2005.

[19] Lécué F., Delteil A., and Léger A.. Optimizing causal link based Web service composition. In ECAI, pages 45–49, 2008.

[20] Papadimtriou C. H. and Steiglitz K.. Combinatorial Optimization: Algorithms and

Complexity. Prentice-Hall, 1982.

[21] Zeng L., Benatallah B., Dumas M., Kalagnanam J., and Sheng Q. Z.. (2003) *Quality driven Web services composition*. In WWW, pages 411–421, 2003

[22] Lécué F. and Delteil A..(2007) *Making the difference in semantic Web service composition.* In AAAI, pages 1383–1388, 2007.

[23] Brandt S., Kusters R., and Turhan A.. Approximation and difference in description logics. In KR, pages 203–214, 2002.

[24] Baader F., Sertkaya B., and Turhan A..(2004) *Computing the least common subsumer w.r.t. a background terminology*. In DL, 2004.

[25] Harel D. and Naamad A.. (1996) *The statemate semantics of statecharts*. ACM Transactions on Software Engineering Methodologies, 5(4):293–333, 1996.

[26] Küsters R., Non-Standard Inferences in Description Logics, volume 2100 of Lecture Notes in Computer Science, Springer, 2001.

[27] Colucci S., Di Noia T., Di Sciascio E., Donini F M., and Mongiello M. Concept abduction and contraction in description logics. In DL, 2003.

[28] O'Sullivan J., Edmond D., and ter Hofstede A. H. M.. (2002) What's in a service? Distributed and Parallel Databases, 12(2/3):117–133, 2002.

[29] Cardoso J., Sheth A. P., Miller J. A., Arnold J., and Kochut K. (2004) *Quality of service for workflows and Web service processes*. J. Web Sem., 1(3):281–308, 2004.

[30] Goldberg D. E.. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[31] Weijters A.J.M.M., van der Aalst W.M.P., van Dongen B., Günther C., Mans R., Alves de Medeiros A.K., Rozinat A., Song M., and Verbeek E.,In M. Dastani and E. de Jong, editors, Proceedings of the 19th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC), 2007.

[32] G. Greco, A. Guzzo, and L. Pontieri. Mining hierarchies of models: From abstract views to concrete specifications. In *Proc. 3rd Intl. Conf. on Business Process Management (Bprocess mining'05)*, pages 32--47, 2005

[33] J. A. Hartigan and M. A. Wong. A K-Means Clustering Algorithm. *Applied Statistics,* 28(1): 100–108, 1979.

Annex A. The Process Template Generator of SOA4All compared to the Process Miner of SUPER

In order to understand the differences / similarities with the approach followed by project Super-IP, we refer to Super Deliverable D6.5 – "Semantic Process Mining Prototype":

"The SUPER methodology, like the usual BPM lifecycle, contains four phases: *Semantic business process modelling, semantic business process configuration, semantic business process execution* and *semantic business process analysis"*

As mentioned, these are the typical phases that are present in every BPM approach. SOA4All is also focussing on these 4 phases. So, the differences should be sought in how such phases are taken in into account. More in detail, if we stick to the objectives of the "Templates Generator Tool", we need to compare it with the "Semantic Process Mining" prototype developed into Super IP.

In the same deliverable (D6.5) we read:

"The Semantic Process Mining prototype aims at providing for a semantic analysis of process instances that are executed within the SUPER framework. This prototype is a subcomponent of the Business Process Management (BPM) Analysis Tools of the SUPER architecture.

The feedback provided by the semantic process mining prototype will aid the (re-)design of processes in the semantic business process modelling phase and their (re-)configuration during the semantic business process configuration phase."

More in detail, Super Semantic Process Mining is composed by a set of plug-ins for ProM. The most relevant one for our comparative analysis is the following:

"Plugin #3. Semantic Control-Flow Mining – Supports the mining of process models with different levels of abstractions. The abstraction is provided based on the concepts in a log"

From all this, we see a **similarity in the objectives**: the discovery of a previously hidden process schema. The **differences** are in the strategies chosen by the two approaches:

- **Super** IP is focussing on a semantic-based approach:

  "Current discovery, conformance and extension process mining techniques are already quite powerful and mature. However, the analysis they provide is purely syntactic. In other words, these mining techniques are unable to reason over the concepts behind the labels in the log, thus the actual semantics behind these labels remain in the head of the business analyst that has to interpret them. Therefore, within SUPER we are developing process mining techniques that ***make use of this semantic Perspective****"*

  The assumption of Super is to: "…make use of the ontological annotations in logs/models to develop more robust process mining techniques that analyze the logs/models at the concept level"

As such it requires that logs are generated by Super-IP execution tools.

- Our approach for SOA4All is more generic, as it does not assume the availability of semantic annotations on service logs

Another difference can be found on how the two approaches can present the user with different levels of abstractions on the discovered process:

- In **Super**, "*more compact models can be generated when instances of different task subconcepts are all mapped to a common task superconcept".* This is achieved thanks to a *"log filter that would allow the end user to pre-process the log to set the desired level of abstraction and, afterwards, use one of the existing control-flow mining algorithms.".* Again the approach is based on the availability of ontologies: *"the plug-in that supports the filtering of the log based on the concepts is the Ontology Abstraction Filter."*
- Our approach is based on a recursive use of discriminant rule extraction and logs clustering algorithms, which generates first an hierarchy of possible process schemas (where leaves constitute a disjoint set representing the initial log set in a more accurate and expressive way rather than the root schema)

As a summary we include the following table with the main differences:

Table 2 Comparative analysis Super-IP vs SOA4All Template Generator.

	Super-IP:	**Template Generator SOA4All:**
Main Purpose	**Process Analysis:** The Semantic Process Mining prototype aims at providing for a **semantic** analysis of process instances that are executed within the SUPER framework	**Process & Service Construction:** 1. Present end-users with an understandable taxonomy of process schemas at different abstraction level, starting from unstructured activity logs. 2. Allow end-users to use such schemas as input for service construction within the SOA4All framework
Advance Core Research in Mining techniques ?	No	No
Limitation to just a single process schema ?	No, thanks to a log filter: *"that would allow the end user to pre-process the log to set the desired level of abstraction and, afterwards, use one of the existing control-flow mining algorithms*", and based on an *"Ontology Abstraction Filter*"	No
Requires Semantic Annotation on Logs required ?	Yes	No
Derivation of an hierarchy of	No	Yes, thanks to recursive use of discriminant rule extraction and logs

schemas		clustering algorithms
Generation of a taxonomy at different abstraction levels	No (only separate models ad different level of abstractions, not arranged into a taxonomy)	Yes, by applying taxonomy discovery techniques on the hierarchy built
Graphical Navigation in the taxonomy	No	Yes
Compatibility with a light-weight BP language	No	Yes
Compatibility with SOA4All studio	No	Yes
Technological Implementation	New plug-ins for ProM	Existing plug-ins for ProM Additional hierarchy & taxonomy building algorithms Clustering algorithms