# SOA4All: An Innovative Integrated Approach to Services Composition[1]

Freddy Lecue
The University of Manchester
*Booth Street East*
*Manchester, UK*

Yosu Gorronogoitia, Rafael Gonzalez,
Mateusz Radzimski
*ATOS Research*
*Albarracín, 25*
*E-28037 Madrid, Spain*

Matteo Villa
TXT e-Solutions SpA
*Via Frigia 27*
*I-20126 Milan, Italy*

## Abstract

*Automated web service composition has been tackled from different directions and to different purposes. In addition, most of the approaches address the composition problem with underspecified requirements, returning compositions models that do not necessarily satisfy and fulfill end-users objectives. Satisfying the latter objectives is a difficult problem, especially from scratch, which requires stronger requirements and a further step of integration with service-based components in order to make service oriented computing and service composition a reality. In this work, we address this issue by presenting an innovative and integrated approach to service composition which consists in i) an automatic template process generator, that is able to generate abstract process templates and their hierarchy form past executions; ii) a novel and scalable approach to AI parametric-design techniques using a multi agent approach to configure and adapt services processes, heavily relying on the latter set of abstract process templates; iii) an optimization process that maximize the overall quality of final compositions.*

## 1. Introduction

A key to the success of a distributed business service model stands in the possibility of reusing the wealth of existing services, their descriptions, some past experience of their use on order to elaborate new ones by composition, providing novel and complex functionalities, even with better overall qualities (e.g., in term of Quality of Services QoS). However, given the requirements for a new service, the task of identifying the existing ones useful to its realization, combining them, and then optimize its result is extremely demanding and time-consuming for a human operator. In addition, composing services from scratch is a complex task which may fail to satisfy the end-users' objectives. Therefore, the development of support tools in this respect is necessary to enact a distributed service view. Ideally, such tools should be able to automatically generate an optimal composed service, starting not only just from properly formulated user requirements (such as preferences and constraints, overall goal) but also with use of predefined template (or schema generally speaking) based composition in order to assist the end-user. However, up to now, the search for such a Holy Grail has been unsuccessful. This is mainly due to the fact that most of composition approaches e.g., [TraP04] make composition requirements and goals underspecified and even not considered in some cases, so considering compositions from scratch, which is very difficult to get. Towards such issues some template based composition approaches [SirHP02] have been proposed in the literature, then coming up with some pre-existing and predefined composition templates ready to be instantiate. However, the latter templates are required before starting the composition process. In addition, their results are not optimized at all. Consequently, automated computation of optimal composition with support of on-the-fly compositions templates is not considered in literature. This means that, in order to obtain an end-to-end architecture that generates compositions templates, instantiate them with concrete service and then optimize them, we need to chain and integrate different, phase-specific techniques.

Since the selection of initial template by then end-users has high impact on the relevance of the final result, phases high in the chain must effectively provide appropriate specifications of compositions templates so that phases lower in the chain can focus on concretize them in an optimal way. This calls for a careful adaptation and integration. Our contribution in this work is an architecture designed and realized within the context of the SOA4All project (*http://www.soa4all.eu/*), performing end-to-end template generation and optimal composition by seamlessly integrating a template generator, a service composer and optimizer. This integration poses serious challenges, in particular at the interface of the template generator and composer, where a high level composition specification (or template) must be picked from the available ones to assist the end-user in the composition and optimization process. For the

[1] Foundation Project: Supported by European Commission VII Framework IP Project Soa4All.

individual components, we utilize appropriate state-of-the-art techniques, except for composition where we provide a novel, advanced and flexible technique, able to divide a complex modeling problem into many domain-specific sub-problems that can be tackled by multi-agents with specialized knowledge.

The current instantiation of our architecture adopts two well-known languages, WSMO [VitKVF08] and LPML[2], to express requirements and service workflows respectively. This makes our architecture immediately usable in conjunction with standard web services engines such as Active BPEL, as witnessed by our experiments on an E-Commerce scenario.

The remainder of this paper is organized as follows. In Section 2, we use a reference "E-Commerce" scenario to illustrate our approach. In Section 3, we describe in detail our architecture, its modules implementing the *Template Generator*, the *Design Time Composer* and the *Composition Optimizer*. Section 4 presents details about the prototype implementation. Section 5 briefly comments on related work. Finally, Section 6 draws some conclusions and talks about possible future directions.

$$ProductData \equiv \forall hasID.ProductID \sqcap \forall hasName.ProductName \sqcap \forall hasURI.URI$$
$$ProductURI \equiv \forall hasURI.URI$$
$$Product \equiv \forall hasID.ProductID \sqcap \forall hasName.ProductName \sqcap \forall hasURI.URI \sqcap \forall hasPrice.ProductPrice$$
$$EuopeanProduct \equiv Product \sqcap \forall hasProvider.Europe$$
$$ProductIDList \equiv \forall hasId.ProductId$$
$$ProductID \sqsubset \top, ProductPrice \sqsubset \top, Europe \sqsubset \top$$

**Figure 1 Sample of an AL Ontology T**

## 2. A Motivating Use Case

The scenario takes place in the E-Commerce domain. It is about utilizing the cloud to expose services and product information from sellers but allowing resellers to expose product offers to multiple platforms and to utilize various distribution channels. All of sellers are from different companies which are all producing different products from the textile industry reaching from footwear to T-Shirts. They are responsible for the E-Commerce part of their companies.

On the one hand the sellers want to increase their sales by offering (semantic) web services, allowing resellers to retrieve a product list and to order a specific product. Each of these services has their own non functional quality criteria (here we only consider the price and response time for illustration purpose). For instance, here is a list of semantic web services exposed by a *Footwear provider*:

- *getFootwearProductIDList* that returns the list of its product Identities i.e., *ProductIDList.* The price "*pr"* and response time "*t*" values are respectively: $5 and 25ms.
- *getFootwearProductData* that returns a list of information *ProductData (e.g., Name, Origin)* given a product identity *ProductID. (pr: $1, t: 150ms).*
- *getFootwearProductPrice* that returns the *ProductPrice* of a product identity *ProductID. (pr: $0.5, t: 50ms).*
- *getFootwearProductURI* that returns the *ProductURI* of a product identity *ProductID. (pr: $3, t: 5000ms).*
- *classifyEUFootwearProductPriceData* that classifies a *EuropeanProduct* in categories of price. *(pr: $2, t: 1000ms).*
- *classifyFootwearProductPriceData* that classifies a *Product* in categories of price. *(pr: $1, t: 200ms).*

wherein all input and output services parameters refer to concept of a domain ontology (Figure 1 shows a portion of the DL based ontology in *AL* - Attributive language with Atomic negation, Concept intersection, Universal restrictions and Limited existential quantification - we use in this scenario).

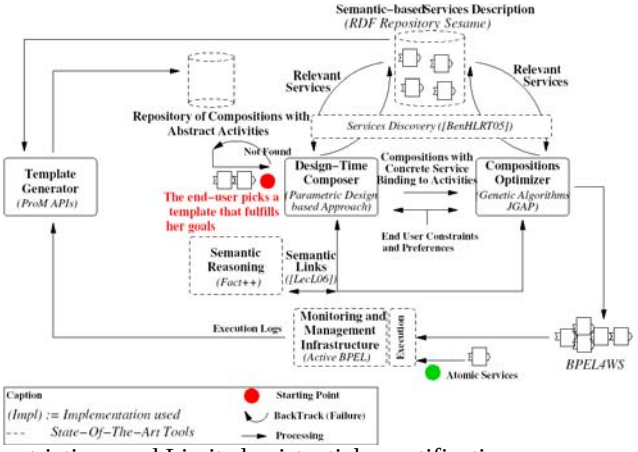


**Figure 2 Core Architecture**

On the other hand the resellers, without any technical background in web service and semantic web technologies, want to generate some side income by reusing and composing the latter services, for instance by exposing their composition on web 2.0 sites. However, automating composition from scratch is far from trivial. In addition, the overall quality of connected services can be not appropriate for business purpose.

We address this scenario by presenting an integrated architecture (Figure 2) that generates on-the-fly compositions templates, concretize one of them (depending on the user selection – see "Starting Point" of Figure 2) in an optimal way with relevant services. By using our architecture, any reseller as any end-user will take profit of traced of past services execution in order to generate abstract compositions, then concrete compositions and finally compositions as optimal as

[2] This language is the composition language supported by SOA4All the EU project that funds this work.

possible in order to deal with the resellers' business perspectives. For instance this can be achieved by minimizing their prices and ensuring seamless connections between services as well.

## 3. Architecture

### 3.1 Templates Generation

Designing complex processes, composed of several services invocation, can be a difficult task for end-users. On the other hand, the availability of pre-defined process template can ease this task. Anyway, how to define such templates may also be a difficult or expensive task, especially in those situations where either an a-priori model is unknown or the effort to create the model is too complex. Process Mining techniques [CooW95, VanHV02] aim at automatically discovering a process model, based on data gathered during its past executions (logs). Most of existing state-of-the-art approaches is devoted to identify a single process formalisation, often resulting in particularly complicated templates and not very accurate (single template for all possible executions). The resulting template, even if formally complete and adequate to support a process execution, turns out to provide little help to let end-users understand what the hidden process template.

Several approaches have been undertaken to solve these kinds of problems. The Template Generator adopts the methodology proposed by [GreGP05], aiming at discovering not a single template, but rather an hierarchy of possible templates, at different abstraction levels, where leaves represent a disjoint set of possible executions, and higher level present a more abstract view, as shown in Figure 3.

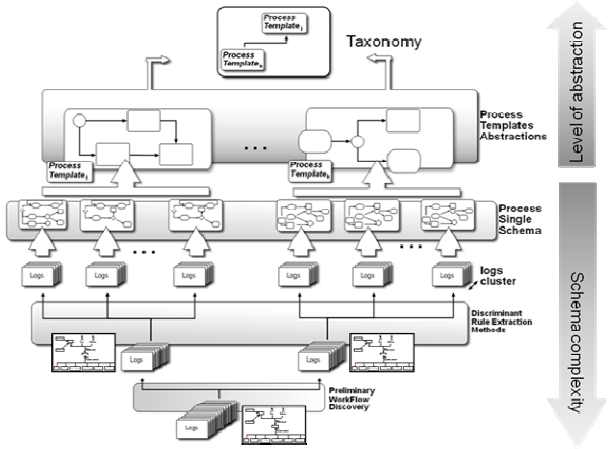


**Figure 3 Mining Approach of the Template Generator**

The Template Generator will then present users with such graphical representations, and let them the freedom to choose the template that best fits their goals.

While this approach is found to be very useful in intra-company scenarios, where process executions logs are typically generated by enterprise legacy systems, new challenges emerge when trying to adopt such technologies to address the Internet of services, due to the very open nature of the Web. First of all, it should be noticed that the problem of deriving relevant and useful process templates out of their services execution logs is still relevant: in fact users could be actually following some process while invoking services in a logical sequence, or they may be actually participating in some collaborative process with other Internet users, without even being aware of this fact, or without having the process structure formalized somewhere.

The first challenge encountered is linked with the difficulty for a user to understand the meaning of the services forming the proposed templates: it is obvious that service names (operations) don't necessarily reveal what's a service is about in an intuitive way, and in any case this may be subject to different interpretations. This situation would not happen in intra-company environments, where a common taxonomy is usually well established.

To address this challenge, the Template Generator exploits semantic technologies, including semantic elements associated with services in the templates presented to the user, so that a better understanding of the service scope could be achieved. It should be noticed that the Template Generator doesn't require the availability of semantic annotations on services, but it can exploit them whenever these are available.

The second challenge arises when trying to decide upon which logs should be analyzed as input, due to the expected large amount of users and services invoked over the Internet, and due to the fact that these belong to different contexts: just taking all logs together would result in non-sense templates.

Due to such reasons it is necessary to follow a contextual-driven filtering approach to input logs (Figure 4), so that only contextually coherent logs will be processed together by the Template Generator component. Unfortunately this problem doesn't have a simple solution: it is not possible to define an a-priori fixed template of such context, as it should describe potentially any situation, and it would eventually turn to be of little help in specific situations. According to a "context framework", where contextual information is structured along a number of aspects or dimensions, applications have to play an active role in indicating the modelling resource used for capturing contextual information as well as directing the way this information is kept and managed. The Template Generator will then

allow users to dynamically configure and customize such framework based on more specific "vertical" scenarios, in a graphical and intuitive way. The approach is summarized in the following figure:

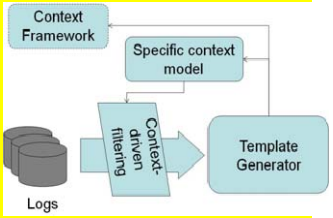


**Figure 4 Context-Driven Logs Filtering**

The output of the Template Generator is the template selected by the end-users. As explained, such template can include one or more abstract activities, and include semantic descriptions associated with each process activities. The tool allows exporting such template to other formats, so that end-users have the possibility to change or refine it in some graphical editor.

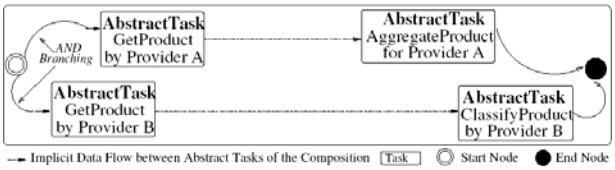


**Figure 5 Composition generated by the Template Generator and selected by the End-User**

*Example (Generated Template Composition):*
*Figure 5 illustrates a template composition computed by the Template generator, and then selected by the end-user. Such a template simply satisfies the following goal: "aggregates and classifies products of different providers".*

## 3.2 Parametric Design Based Composition

Our approach is driven along some modeling principles: i) iterative, incremental, easy to use, semi-assisted compositions modeling, ii) coarse-grained goal based activity-centric description of compositions models, as opposed to the service-centric SOA composition approach, iii) semantically annotated activity descriptions (goals), iv) intensive reuse and customization of preexisting domain specific process templates and fragments, v) context-aware model composition and adaptation. In this picture, the composer assists iteratively human modelers to complete the composition model, by fulfilling required model gaps left by human modelers, with information extracted from different knowledge sources.

The composer accepts an incomplete composition model, which comprises a set of activities, logically linked by a draft workflow, coarse-grained described by their requirements (goals) and expressed as a set of annotations that reference semantic concepts defined within a shared ontology. The composer returns a more elaborated composition model which has resolved some of its information gaps: activities are bound to concrete WS, or expanded with compositions templates or fragments, data flow is populated with connectors mediating between Input/Output parameters, semantic compatibility between subsequent the latter parameters is checked, etc. The composition approach increases the level of concreteness of composition models; closer to executable as opposed to abstract and adaptable (Figure 6).

This adaptable concretization transformation is implemented using a knowledge-intensive configuration process, more precisely "*a parametric design procedure*" [WieAS95]. In order to increase the scalability of this procedure, we extend the classical approach to its synthesis task by using an opportunistic approach, based on "*blackboard-based multi-agent system*" [ErmHLR80]. Multi-agent architecture allows also for extra flexibility with regard to management of knowledgebase used by the composer, by allowing for hot-plugging or updating knowledge while the composer is operating. This can be either adding new ontologies, services and templates capabilities or registering completely new agents as well.

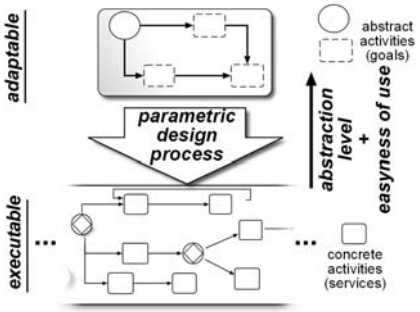


**Figure 6 Adaptable Concretization of Compositions**

### 3.2.1 Parametric Design Procedure

From a high level design model of a composition given by an end-user, the composer converts the composition modeling problem in a parametric design problem [Mot99]. In such a mapping, the parameter set is mapped to the set of unbound activities, whose values range is either the range of available services or process templates. The potential design model solutions are requested to satisfy some constraints, requirements and preferences parameters (used to express limitations and desires on the requested functionality). Then they are all classified along properties of completeness, admissibility and suitability [Mot99], but not in term of overall quality (e.g., QoS).

3.2.2 Blackboard-based Multi-Agent System

The composer implements the synthesis phase of parametric design procedure with a blackboard-based multi-agent system. Autonomous and specialized agents share a common backboard upon which they post new design models, which are modified versions of previously posted ones, after applying changes driven by some specific knowledge.

Some agents, named as Design Modification Agents (DMA), are specialized to introduce changes in the models, while other agents, named as Design Analysis Agents validate those changes. All of the DMAs semantically match the composition activities with a service or a template by considering semantic satisfiability of the (semantic) intersection of their functional descriptions (i.e., defined by input, output, functional classification, requirement, and preference). Preconditions, effects imply logical expressions we don't support, event we can not expect end-users to express such Preconditions/Effects logical expressions.

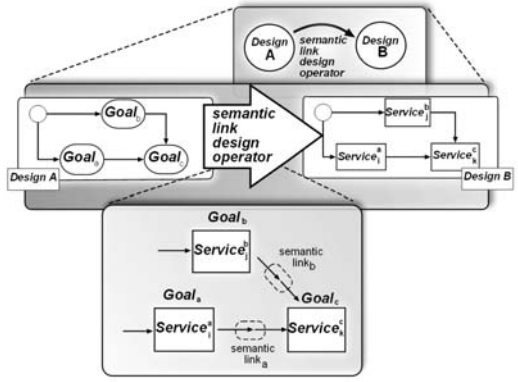


**Figure 7 The Semantic Link Design Operator**

In our approach, four DMAs have been designed:

- A Domain Specific Language [MerHS05] DMA, which exploits descriptions of services and process templates according to concrete compositions requirements. A possible domain language used to describe available knowledge about known services and composition templates are shown below, where the functional classifications simply refer to a concept in a hierarchy of domains (see WSMO-Lite definition [VitKVF08]).

```
templateORservice_def URL
    functionalClassification <URI> [1..*] /** for service and template */
    input <URI> [0..*] /** for service and template */
    output <URI> [0..*] /** for service and template */
    definition <URL>    /** only for service*/
    operation <name>  /** only for service*/
    global requirement <URI> [0..*] /** for service and template */
    global constraint <URI>[0..*] /** for service and template */
def_ templateORservice
```

- A Service Discovery [BenHLRT05] DMA, which binds activities to concrete services. The discovery process acts on functional classification, input, output, preconditions and effects parameters. The result of candidate services is filtered based on the matching quality of the input and output parameters of their operations.
- A Semantic service description DMA, which exploits domain specific descriptions of services and goals, according to concrete composition requirements. In contrast to Service Discovery DMA, this DMA is enhancing design process by taking into account additional information to narrow the target of candidate services: domain specific knowledge, global annotations and context of compositions.
- A Semantic Link Design Operator [LecL06]. DMA (Figure 7), that establishes dataflow in the models. In other words this DMA checks the semantic compatibility between the input and output parameters of connected compositions bound activities and creates suitable connections, constituting the composition data flow. The latter compatibility is valued by different standard matching type between semantic descriptions of output and input parameters of services, by order of semantic quality: Exact (i.e., same concept to describe output and input parameters), PlugIn (i.e., the output parameter is more specific than the input parameter), Subsume (i.e., the output parameter is more general than the input parameter) [PaoKPS02] and Intersection (i.e., some properties are in common) [LiH03]. Such a DMA acts on models that are already complete in terms of parametric design process which means that all services are bound and model contains all necessary information to start dataflow mapping.

The aim of the blackboard is to control and coordinates the autonomous and independent agents.
At the beginning of the procedure, the blackboard is seed with an initial design plan, whose assignment set is empty. At the end of the procedure none or several found solutions are returned (completed assignment set). The procedure finalizes either when the requested number of solutions are found or when the DMAs can not post new design models.

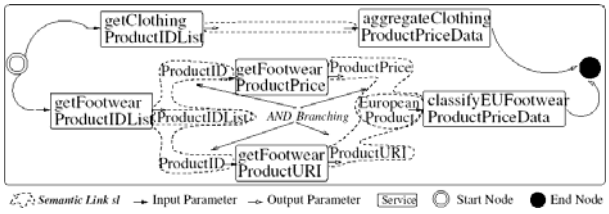


**Figure 8 Composition returned by the Composer**

*Example (Final Non Optimal Composition):*
*Figure 8 depicts the composition computed by the composer. Such a composition consists in services connected by semantic links. For instance, according to Figure 1 the semantic link between both services getFootwearProductURI and getFootwearProductPrice with classifyFootwearEUProductPriceData is valued by a Subsume matching type. Indeed, this link is valued by a*

*Subsume matching type since EuropeanProduct is subsumed (or more specific than) by the conjunction of ProductURI and ProductPrice.*

## 3.3 Optimizing Service Composition

Since the design-time composer aims at only computing compositions that satisfy a specific goal and does not consider global optimization of a composition, a step of optimization is required (cf. Composition Optimizer in Figure 2). Mostly due to performance optimization, context adaptation or specific user preferences and constraints, it is necessary to optimize the completed compositions. While the input to the composer is generally a rather goal-heavy process specification, the optimizer only accepts complete process models for which it seeks a better global cost function (in term of functional and non functional qualities of services). The optimizer transparently transforms compositions into their optimal versions by replacing service bindings and modifying the dataflow but without changing the workflow (i.e., its structure – Figure 9).

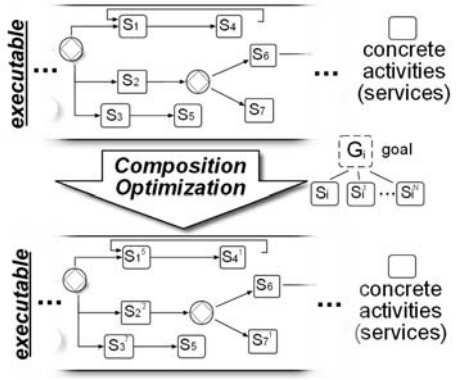


**Figure 9 Process Optimization Recombining Services**

The optimizer receives a complete process model and tries to replace current bound services with other web services which make better global cost function (e.g., better global performance). The cost function of our optimization approach uses a combination of functional and non-functional considerations. The functional perspective comprises a set of metrics related to how well the functionalities of the constituent services fit together. The semantic quality is such a core metric, measuring the degree of semantic similarity between the outputs produced by constituent services and the inputs required by their peers. Such a quality is one of the measures of the overall functional quality for the composition, indicating the "goodness of fit" between the functionalities of the constituent services. To measure the degree of semantic similarity, we use the concept of semantic link (used by the composer), and so a semantic reasoning component which evaluate subsumption of concept-based parameters. Web service compositions could thus be optimized and ranked using not only nonfunctional parameters such as the well-known QoS [CanPEV05], but also using semantic quality as a core indicator of functional quality [LecDL08]. In the same way as [LecN09], we suggest to unify both types of criteria, allowing us to estimate and optimize the quality of service compositions.

In addition rules for aggregating quality values of services and semantic links in the composition are required. The approach for computing semantic quality of compositions is adapted from the application-driven heuristics of [LecDL08], while the computation of its non functional QoS is similar to [CarSMAK04]. For instance, the overall price of a sequence-based composition of services is valued by their average whereas its response time is valued by their sum.

Optimizing the quality of semantic links between services and QoS aims at not only consider non functional properties such as price or availability rate but also the quality of semantic fit along non-trivial data flow, where the information required and provided by services does not match perfectly in every dataflow, using semantic-based description of services. By addressing non trivial data flow in composition, we aimed at limiting the costs of (semantic heterogeneity) data integration between services by considering appropriate quality of semantic links.

Optimizing the quality of service composition using this model is essentially a multi-objective optimization problem [ArdP07] with constraints and preferences on the quality of services and their semantic links. Such a problem is known to be NP-hard [PapS82], thus putting into question the practical applicability of such optimization for compositions of realistic scale. To speed up the application of the proposed optimization model in a context of realistic scale, we adapt the approach based-GAs (Genetic Algorithms) of [CanDEV04] in order to consider non functional properties of services and non qualities of semantic links between services. In addition we extend the latter model by revisiting the fitness function in order to avoid local optimal solution (i.e. compositions disobeying constraints are considered).

Roughly speaking, the execution of the GA consists in the following steps: i) defining the initial population (as a set of concrete compositions), and computing the cost function (used as an evaluation criterion) for each composition; ii) evolving the population by applying mutation and crossover of compositions (this step simply binds new services in the composition by discovery close service in the service repository); iii) selecting compositions; iv) evaluating compositions of the population; v) and back to step (ii) if the stopping criterion is not satisfied in the GA evolution. In case no solution exists, users are requested to relax constraints of the optimization problem in order to compute alternative

solutions still providing a reasonable quality of composition (at both QoS and semantic levels).

*Example (Optimal Composition):*
*Suppose de composition returned by the composer in Figure 8. The optimization process changed its service binding by discovering services with better non functional qualities and also with higher semantic quality of the links between services (Figure 10). For instance, the services getFootwearProductData and classifyFootwearProductPriceData have been preferred since they maximize the cost function of the optimizer process. On the one hand their price and response time are the lowest. On the other the semantic quality of both links between the latter services are better (in term of matching type). Indeed, the connection between the conjuction of the output ProductData of getFootwearProductData and the output ProductPrice of getFootwearProductPrice and the input Product of classifyFootwearProductPriceData is of an exact matching type.*

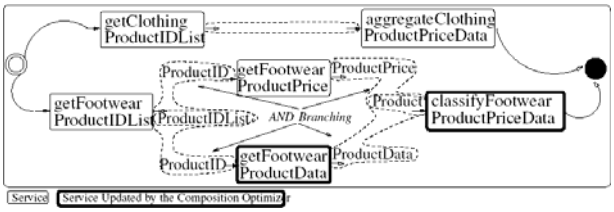


**Figure 10 Composition returned by the Optimizer**

In case of conflicts e.g., the value of the non functional quality is the best for a first composition but worse regarding the semantic quality, we compare a weighted average (with a weight of 1/2) of their normalized values.

## 4. Prototype Implementation

The Template Generator has been implemented as a Widget front-end, developed with GWT (http://code.google.com/webtoolkit/), interfacing with a server-side component, which is also integrating the ProM APIs (http://prom.win.tue.nl/tools/prom/) and the mining plug-in (http://staff.icar.cnr.it/wfmining/tools.htm).

The composer core has been implemented using Spring framework IoC container [JohHARS07], endowing the composer of beam configuration, management and observable pattern blackboard notifications for blackboard. All agents, including the blackboard control agent have been implemented as Spring beans. The design analysis agents have been implemented using JBoss Drools rule engine (http://www.jboss.org/drools/). A specific drools rule language has been implemented to check the parametric design taxonomy of design models. The Domain Specific Language DMA also uses JBoss Drools rule engine. The semantic service description DMA is using wsmo-api, wsmo4j (http://wsmo4j.sourceforge.net/). For legacy WSMO support it is also using WSMO-discovery module. The service discovery DMA consumes a WSMO based service, exposed as a WSDL WS, using JAX-WS (https://jax-ws.dev.java.net/). The semantic link DMA consumes a semantic link engine, exposed as a WSDL service, using JAX-WS as well. The latter engine is internally using the Fact++ Reasoner [TsaH06] through an interpreter of WSML descriptions. Finally, the composer exposes the result as a BPEL4SWS service, understandable by the optimizer.

In the same way as the composer, the optimizer use Fact++ on top of the semantic links. The GA is implemented in Java, extending a GPL library JGAP (http://jgap.sourceforge.net/). The optimal compositions are computed by using an elitist GA where the best 2 compositions were kept alive across generations, with a crossover probability of 0.7, a mutation probability of 0.1, a population of 200 compositions. We consider a simple stopping criterion i.e., up to 400 generations.

## 5. Related Work

### 5.1 Template Generator

Data mining techniques are usually strictly related to process mining problems: in fact, supposing a certain number of tasks within a process, the number of possible different executions is an exponential number (while not all of them have the same probability to be actually executed). Here is where data-mining technology helps.
[CooW99], working in the area of software engineering processes, analysed three different methods for process discovery: neural networks, Markovian models, and algorithmic approach. Their work allows to generate explicit process models, and to measure the actual gap between process model and actual observed behaviour. [AgrFL98] introduced the idea of extending process mining to workflow management, by analyzing the events recorded in a log and by identifying constraints. Their approach is based on an algorithmic approach, by enumerating tasks instances and applying a folding procedure. [Gol78] shows that the problem of finding a state-machine compatible with a set of recorded data is NP-hard. This problem has an analogy with the process mining problem; even if an important difference is that process mining needs to take into account concurrent tasks.

### 5.2 Design Time Composition

There is intense research on the automated synthesis of services compositions. Some approaches exploit heavyweight semantic descriptions of WS: [SirHP02] exploits DAML service descriptions to derive semi-automatically service compositions, [HakSCDM05] proposes a WSMO orchestration complementary model implemented in IRS-III.
A parametric design method applied to SOA composition is found in [HarW04], where they applied parametric design configuration over a fixed composition template.
Other automatic service composition approaches uses AI techniques. [TraP04] converts the problem of composing a fixed number of services described by OWL-S models into an AI planning problem. A complete survey of proposed techniques for synthesis of service compositions can be found in [MarP09].
In general existing approaches requires heavyweight service descriptions, fixed composition templates that aggregates single services, but do not consider the reuse of pre-existing process templates.

### 5.3 Composition Optimization

Proposals to address composition optimization center on stochastic approaches [CarSMAK04], Constraint Programming [HasMI06] and Integer linear Programming (IP) [ZenBNDKC04], [LecDL08], with the latter considered showing most promise. However, IP approaches have been shown to have poor scalability in terms of time taken to compute optimal compositions when the number of available services grows. The optimization problem can be also modeled as a knapsack problem [YuL05], wherein [ArpZAM05] performed dynamic programming to solve it. Unfortunately the previous QoS-aware service composition approaches consider only links valued by Exact matching types, hence no semantic quality of compositions. Here we apply [Lec09] by using GAs to achieve optimization in web service composition, and by i) using semantic links to consider data flow in composition, ii) considering not only QoS but also semantic quality (and constraints) of composition,

### 5.4 Integration

As confirmed by the previous study on the related work, many works address template generation, service composition and optimization, but their integration, which is far from trivial, is not studied. As motivated by our E-Commerce scenario, the service providers could benefits such an integration by using our end-to-end architecture, easing composition and their optimization.

## 6. Conclusion

We presented and tested an architecture that achieves fully end-to-end composition and optimization in a realistic setting, where a process of compositions templates is required before starting the approach. The architecture combines advanced techniques for Template Generation, flexible Composition, and Optimization, and therefore significantly departs both from architectures which only focus on one of these tree components or compositions approach starting from scratch.

A further direction of work stands in a fuller integration of semantic-based mining (and reasoning) within the Template Generator, to support the extraction of more relevant abstract process, useful for composition and then optimization processes. In addition, we will further investigate on the adaptation of our architecture to contextual information: context-driven logs filtering for the template generator, and context-based composition and optimization. Finally, we expect large scale experiments with thousands of users and services.

## 7. References


[ArdP07] Danilo Ardagna and Barbara Pernici. Adaptive service composition in flexible processes. IEEE Trans. Software Eng., 33(6):369–384, 2007.

[AgrFL98] R. Agrawal, D. Gunopulos, F. Leymann, Mining process models from workflow logs, in: Proceedings of the Sixth International Conference on Extending Database Technology, 1998, pp. 469–483.

[ArpZAM05] Arpinar, I.B., Zhang, R., Aleman-Meza, B.,Maduko, A.: Ontology-driven web services composition platform. Inf. Syst. E-Business Management 3(2), 175–199 (2005).

[BenHLRT05] Boualem Benatallah, Mohand-Said Hacid, Alain Léger, Christophe Rey, Farouk Toumani: On automating Web services discovery. VLDB J. 14(1): 84-96 (2005)

[CanPEV05] 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.

[CooW99] J.E. Cook, A.L. Wolf, Software process validation: quantitatively measuring the correspondence of a process to a model, ACM Transactions on Software Engineering and Methodology 8 (2) (1999) 147–176

[CooW95] J.E.Cook and A.L.Wolf. Automating process discovery through event-data analysis. In Proc.17th Int. Conf. on Software Engineering (ICSE'95), pages 73–82, 1995.

[ErmHLR80] 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

[Gol78] E.M. Gold, Complexity of automation identification from given data, Information and Control 37 (3) (1978) 302–320.
[GreGP05] G. Greco, A. Guzzo, and L. Pontieri. Mining hierarchies of models: From abstract views to concrete specifications. In BPM, pages 32--47, 2005
[HakSCDM05] Farshad Hakimpour, Denilson Sell, Liliana Cabral, John Domingue, Enrico Motta: Semantic Web Service Composition in IRS-III: The Structured Approach. CEC 2005: 484-487
[HarW04]Teije, A. t. van Harmelen, F. Wielinga, B. Configuration of Web Services as Parametric Design. Lecture Notes in Computer Science. Springer Verlag. ISSU 3257, pages 321-336. 2004
[JohHARS07] R. Johnson, J. Hoeller, A. Arendsen, T. Risberg, C. Sampaleanu. Professional Java Development with the Spring Framework. Wrox. 2007
[LecL06] Lécué F., and Léger A., (2006), 'A formal model for semantic Web service composition', in ISWC 2006, pp. 385–398.
[LecDL08] Lécué F., Delteil A., and Léger A.. Optimizing causal link based Web service composition. In ECAI, pages 45–49, 2008.
[LecN09] Lécué F., Mehandjiev N.: Towards Scalability of Quality Driven Semantic Web Service Composition. ICWS 2009: 469-476
[LiH03] L. Li and I. Horrocks. A software framework for matchmaking based on semantic Web technology. In WWW, pages 331–339, 2003.
[MarP09]A. Marconi, M. Pistore: Synthesis and Composition of Web Services. SFM 2009: 89-157
[MerHS05] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," ACM Computing Survies, vol. 37, no. 4, pp. 316–344, 2005
[Mot99] Motta E., (1999) Reusable Components For Knowledge Modelling Case Studies In Parametric Design Problem Solving,IOS Press (Netherlands)
[NitN08] Nitzsche, J., Norton, B.: Ontology-based data mediation in bpel. In: BPM Workshops. (2008) 523–534
[PaoKPS02] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara, (2002) Semantic matching of Web services capabilities. In ISWC, pages 333–347
[PapS82] Christos H. Papadimtriou and Kenneth Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, 1982.
[SirHP02] E. Sirin, J. Hendler, B. Parsia, Semi-automatic composition of Web services using semantic descriptions, in: Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003, 2002
[TraP04] P. Traverso, M. Pistore: Automated Composition of Semantic Web Services into Executable Processes. International Semantic Web Conference 2004: 380-394
[TsaH06] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System De-scription. In Proc. IJCAR 2006, pages 292{297, Seattle, WA, USA, 2006.
[VanHV02] W.M.P. van der Aalst, A. Hirnschall, and H.M.W. Verbeek. An alternative way to analyze workflow graphs. In Proc. 14th Int. Conf. on Advanced Information Systems Engineering, pages 534–552, 2002.
[VitKVF08] Vitvar T., Kopecký J., Viskova J., Fensel D. (2008) WSMO-Lite Annotations for Web Services. ESWC 2008: 674-689.
[WieAS95] 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)
[YuL05] Yu, T., Lin, K.-J.: Service selection algorithms for composing complex services with multiple qos constraints. In: ICSOC 2005. LNCS, vol. 3826, pp. 130–143.