



Project Number: **215219**
 Project Acronym: **SOA4ALL**
 Project Title: **Service Oriented Architectures for All**
 Instrument: **Integrated Project**
 Thematic Priority: **Information and Communication Technologies**

D6.5.4 Evaluation of Service Construction

Activity 2:	Core Research and Development	
Work Package 6:	Service Construction	
Due Date:	28/02/2011	
Submission Date:	31/03/2011	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	ATOS	
Revision:	1.0	
Author(s):	Yosu Gorroñoigoitia, Freddy Lecue, Giovanni di Matteo, Juergen Vogel, Patrick Un, Florian Schnabel, Gianluca Ripa, Teodoro De Giorgio	
Reviewers:		

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	25/11/2010	ToC	Yosu Gorroñoigoitia (ATOS)
0.2	20/12/2010	Final ToC, Initial contributions	Yosu Gorroñoigoitia (ATOS), Gianluca Ripa (CEFRIEL), Nikolay Mehandjiev (UNIMAN)
0.3	10/02/2011	First main contributions to section 3. Complete section 2.	Yosu Gorroñoigoitia (ATOS), Giovanni di Matteo (TXT) Gianluca Ripa, Teodoro De Giorgio (CEFRIEL) Patrick Un (SAP) Freddy Lecue (UNIMAN)
0.4	16/02/2011	Additional contributions to section 3.	Giovanni di Matteo (TXT) Freddy Lecue (UNIMAN) Yosu Gorroñoigoitia (ATOS) Patrick Un (SAP)
0.5	07/03/2011	Additional contributions to section 3.	Gianluca Ripa, Teodoro De Giorgio (CEFRIEL), Freddy Lecue (UNIMAN), Yosu Gorroñoigoitia (ATOS)
0.6	25/03/2011	Final contributions to section 3. Annex B	Gianluca Ripa, Teodoro De Giorgio (CEFRIEL), Yosu Gorroñoigoitia (ATOS)
0.7	29/03/2011	Final contributions to section 3. Conclusions	Juergen Vogel, Florian Schnabel (SAP), Yosu Gorroñoigoitia (ATOS)
1.0	31/03/2011	Final Editing	Yosu Gorroñoigoitia (ATOS)

Table of Contents

VERSION HISTORY	2
LIST OF FIGURES	4
EXECUTIVE SUMMARY	7
1. INTRODUCTION	8
1.1 PURPOSE AND SCOPE	9
1.2 STRUCTURE OF THE DOCUMENT	10
2. EVALUATION CRITERIA FOR SERVICE CONSTRUCTION	11
3. EVALUATION OF LIGHTWEIGHT PROCESS MODELLING AND ADAPTIVE PROCESS EXECUTION	12
3.1 PROCESS MODELLING WITH LPML	12
3.1.1 <i>Completeness, Expressiveness, Executability, and Correctness</i>	14
3.1.2 <i>Flexibility, Composability, and Interoperability</i>	16
3.1.3 <i>Discoverability and Reusability</i>	16
3.1.4 <i>Usability for End Users</i>	17
3.2 ASSISTED PROCESS MODELLING	22
3.3 MODELLING BY KNOWLEDGE INTENSIVE REUSABILITY	27
3.4 CONTEXT-AWARENESS PROCESS ADAPTATION	31
3.5 PROCESS OPTIMIZATION	33
3.5.1 <i>Context of Experimentation</i>	34
3.5.2 <i>Benefits of Combining Quality Criteria</i>	35
3.5.3 <i>Evolution of Satisfaction Constraints</i>	36
3.5.4 <i>Evolution of the Composition Quality</i>	36
3.5.5 <i>Towards Large Scale based Compositions</i>	37
3.5.6 <i>Decoupling GA Process and DL Reasoning</i>	38
3.5.7 <i>Comparing IP and GA-Based Approaches</i>	39
3.5.8 <i>Convergence of GA-Based Approaches</i>	41
3.6 PROCESS DEPLOYMENT ADAPTATION	41
3.7 EXECUTION AUTONOMOUS CAPABILITIES.	43
3.8 HYBRID PROCESS SUPPORT	45
4. CONCLUSIONS	49
5. ANNEX A	50
6. ANNEX B	51
7. REFERENCES	59

List of Figures

Figure 1 Response times for DTC methods	26
Figure 2 Response-times for DTC resolveProcess method with the KB scale.....	27
Figure 3 Costs of Data Integration.....	35
Figure 4 Evolution of Satisfaction Constraints - Static versus Dynamic Fitness.....	36
Figure 5 Evolution of the Composition Quality.....	37
Figure 6 DL and GA Processes in our Approach.....	38
Figure 7 DL Computation Costs along GA Generations.	39
Figure 8 Costs for Reaching 95% of the Global Optimum.	40
Figure 9 Costs for Reaching 99% of the Global Optimum.	40
Figure 10 SOAP and REST Service support in the EE.....	42
Figure 11 Distribution of API protocols and styles. Source: [31].	46
Figure 12 Possible data formats.Source:[31].....	46
Figure 13 Most popular protocols, styles and dataformats	46
Figure 14 Percentage of new APIs with JSON support [28].....	47
Figure 15 SOAP_REST_WeatherForecast Process.....	52
Figure 16 LILO Process	53
Figure 17 Notification processBPEL Fragment.....	55
Figure 18 Notification process, BPEL Fragment simplified	55
Figure 19 WP8 Scenario	57
Figure 20 WP9 Scenario	58

List of Tables

Table 1: Favourite modelling languages or tools of respondents.....	21
Table 2 Overview of Computation Costs.	37
Table 3 Large Scale Compositions.....	38
Table 4 GA-based Approaches (Population size of 200).....	41
Table 5 What EE supports	47
Table 6 SOA4All scenarios from the EE point of view	48
Table 7 Service Construction Evaluation Criteria	50
Table 8 Processes and evaluation criteria.....	51

Glossary of Acronyms

Acronym	Definition
M	Milestone
DOW	Description of work
WP	Work package
SOTA	State of the art
BPM	Business process modelling
DTCE	Design Time Composition Environment
EE	Execution Environment
LPML	Lightweight Process Modelling Language
DTC	Design Time Composer
API	Application Programming Interface
BPEL	Business Process Execution Language
RDF/S	Resource Description Framework/Schema
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
YAWL	Yet Another Workflow Language
SPARQL	SPARQL Protocol and RDF Query Language
BPMN	Business process modelling notation
EAI	Enterprise Application Integration
WSDL	Web Services Description Language
WSMO	Web Service Modelling Ontology
IT	Information Technology
SAP BPX	SAP BUSINESS PROCESS EXPERT
DSB	Distribute Service Bus
SOA	Service Oriented Architecture
DMA	Design Modification Agent
WSML	Web Service Modelling Language
POSM	Procedure Oriented Service Model
KB	Knowledge Base
SLO	Semantic Link Operator
MSM	Minimum Service Model

P2P	Peer to peer
DAA	Design Analysis Agent
I/O	Input/Output
TG	Template Generator
URL	Uniform Resource Locator
PE	Process Editor
NP	Non-Deterministic Polynomial-time
QoS	Quality of Service
GA	Genetic Algorithms
IP	Integer Programming
DL	Description Logic
LCS	Least Common Subsumer
LILO	Lifting Lowering
XML	Extensible Markup Language
JSON	Javascript Object Notation
URI	Uniform Resource Identifier
XSLT	Extensible Stylesheet Language Transformations

Executive summary

This document contains a detailed description of the technical evaluation of the SOA4All Service Construction results: Lightweight Process Modelling Language, Design Time Composition Suite and Execution Environment. This technical evaluation complements other usability, fit-for-purpose and technical evaluations conducted in the context of other use case and technical work packages that may have evaluated similar functionality under other criteria. In the document, we discuss possible evaluation criteria (SOA4All design principles, general requirements, specific requirements, similar scientific approaches, etc) that we use in order to evaluate SOA4All Service Construction features. Addressing those criteria, we evaluate the main functionality of SOA4All Service Construction using heuristics (based on our experiences during the tools development and testing phases, supporting the enactment of some case studies scenarios), questionnaire-based expert survey and experimental-based tests. Evaluation results are collected and commented in the document. When possible, further improvements are proposed to improve the SOA4All Service Construction features.

1. Introduction

This document describes the evaluation of the main results of the SOA4All Service Construction work package along with the project lifetime until the M30 milestone. The main technical results of this work package are:

- Lightweight Process Execution Language specification and its reference implementation.
- Context-aware Service Composition and Adaptation framework (aka Design Time Composition Environment)
- Adaptive Service Compositions Execution framework (aka Execution Environment).

According to the DOW this deliverable will **validate the results of the light-weight and adaptive composition techniques in terms of applicability and suitability as well as other criteria**. Therefore, this deliverable will:

- Define the evaluation scope and objectives.
- Identify and define the evaluation criteria that it is used to assess the Service Construction results.
- Use these evaluation criteria to assess the most relevant technical results of the Service Construction as a whole, collecting and analysing the evaluation results.
- Provide a detail analysis and explanation of the evaluation results, proposing improvements for future work that overcome the non-positive aspects of the evaluation.

After some internal discussion, the Service Construction team decided to adopt the following principles for the evaluation:

- The Service Construction Suite is evaluated globally as a comprehensive entity, but not for each individual component separately. That is, we perform a holistic evaluation, based on the most remarkable functionality. The reason is that end-users perceive Service Construction features, rather than available components. Moreover, different components are involved together to offer some of these functionalities, that is, they are not offered just by a isolated component. Nonetheless, when necessary, we assess concrete Service Construction components.
- We consider different source for evaluation criteria. As a result of such analysis, the evaluation criteria are classified by importance and applicability. The functionality of the Service Construction platform is evaluated according to some concrete evaluation criteria within that classification depending on which ones suit better for the evaluation objectives, that is, addressing the SOA4All main principles and objectives.

1.1 Purpose and Scope

SOA4All Service Construction was built along with the project lifetime to achieve some scientific and technical objectives and support the use case requirements that challenged SOA4All. Additionally, SOA4All defined some principles that constrained the research and development of the SOA4All platform. After the development cycle, we are required to assess the SOA4All results in order to ensure that we have addressed the main objectives and requirements and observed the principles. Deviations of the SOA4All platform results from the objectives and requested requirements must be identified and measured, and corrective actions or improvements proposed in order to minimize the user experience and the successful exploitation of the SOA4All results.

In the early stage of the project, the SOA4All jointly evaluation work was defined, see [12]. This work defined the SOA4All evaluation process what is jointly accomplished by some technical work packages¹ and the case studies work packages. Moreover, this document identified the main SOA4All objectives and their potential evaluators: end users, use cases and technology, and, based on those beneficiaries, the sort of possible evaluations, respectively: usability, fit-for-purpose and technical. Besides, the document depicts the internal SOA4All evaluation model, which identifies:

- Providers of requirements/metrics: WP7-WP9 use cases
- Providers of technology: WP2-WP6
- Technical Evaluators: WP1, WP2
- Technical providers: WP3, WP5, WP6
- Fit-for-purpose evaluators: WP7-WP9 use cases
- Usability evaluators: end-users.

In this SOA4All evaluation model, the technical evaluation of WP3-WP6 results is done by WP1 and WP2, about the criteria that are relevant to WP1 SOA4All infrastructure and WP2 Studio requested functionality. Use cases WP7-WP9 do the fit-for-purpose evaluation of WP3-WP6 outcomes.

In this context, the internal WP6 Service Construction evaluation should complement the afore-mentioned overall evaluation schema. In this sense, WP6 evaluation is restricted as a technical evaluation of the Service Construction features, attending to internal (to WP6) scientific and technical requirements as well as the main SOA4All principles identified in its main 4 pillars: Semantics, Web 2.0, Web principles and Context.

SOA4All Service Construction scientific, technical and functional criteria, as well as SOA4All principles are identified after a survey from different sources. The results of the assessment are analysed and further ideas for improvement suggested as future work. The main SOA4All Service Construction features that are evaluated are:

- Lightweightness, expressivity and executability of process models.
- Easiness on process modelling, including assisted features
- Modelling by knowledge intensive reusability
- Context-awareness process adaptation

¹Concretely WP1 for SOA4All platform and WP2 for SOA4All Studio.

-
- Process Optimization
 - Process deployment adaptation
 - Autonomous capabilities for process execution
 - Hybrid process execution support.

These features has been selected since they are the most relevant Service Construction functionalities implemented that differentiate SOA4All from other related projects, such as Super and SeCSE.

1.2 Structure of the document

The rest of this document is organized as follows. Section 2 surveys possible evaluation criteria inspecting different sources, evaluates and classifies them according to the relevance and importance for SOA4All Service Construction. Section 3 evaluates the main SOA4All Service Construction features according to some of those criteria when apply, describes the results and proposed future work and improvements. Section 4 outlines the main evaluation results. Annex A collects the most relevant evaluation criteria for Service Construction Suite. Annex B describes all the processes generated during the project in order to test and evaluate the EE characteristics at different stages of the EE implementation.

2. Evaluation Criteria for Service Construction

This section describes the evaluation criteria used to assess the SOA4All Service Construction and classifies them according to their relevance and importance. Functional, non-functional criteria and other criteria based on comparison against the SOTA (including related projects) are identified from different sources and described.

The survey for sources of Service Construction Evaluation criteria collected some relevant documents where from we could extract some relevant criteria:

SOA4All design principles described in [2] collects most relevant and motivating SOA4All principles: service-orientation, web, autonomic computing and formal semantic principles. It additionally summarise the main challenges for SOA4All. Most of those principles and challenges are quite relevant for SOA4All Service Construction and therefore used as baseline criteria for evaluation of Service Construction. As example, we remark some service-orientation principles, such as reusability, discoverability and composability; web principles such as distributed, openness, human-centric, autonomic computing principles, such as self-healing, self-optimization; formal semantic principles such as ontology-based, centrality of mediation, problem solving.

General SOA4All requirements collected in [3] and [4], section 2, such as machine and human-based computation, dynamicity and adaptability, scalability. This document also summarizes the requirements for Service Construction, split into BPM and executable languages, service composition and adaptation, and service execution, in sections 4, 5 and 6 respectively. Since this document also describes the current State of the Art on Service Construction, we can use it in order to compare our Service Construction outcomes with the research initiatives described within, in case they try to solve similar challenges and address common target users. A particular case of evaluation based on the comparison with regards to the State of the Art consists on the collation of SOA4All Service Construction results with the results obtained by foregoing projects SOA4All was inspired for, when they address similar challenges and target users, otherwise, they are not comparable. This could be possible for some SOA4All Service Construction features when compared to SUPER and SeCSE results.

SOA4All Service Construction principles and assumptions are collected in [4], sections 3 and 4. Main design principles are lightweightness and context-awareness.

Other SOA4All Service Construction principles and requirements are collected in [6], [9],[10]. They describe the requirements for the Design Time Composition Environment and the Execution Environment.

Finally, SOA4All, as a NESSI Strategic Project, has been actively contributing to NEXOF-RA Conceptual Model and Reference Architecture. The architectural patterns submitted to NEXOF-RA have been evaluated according to the NEXOF-RA Quality Model for NEXOF-RA Pattern Designing described in [1]. This model describes measurable and observable qualities of a software system, which are susceptible to be used to assess any software system, in particular SOA4All Service Construction. Table in Annex A summarise the most relevant evaluation criteria we will use within this evaluation.

3. Evaluation of Lightweight Process Modelling and Adaptive Process Execution

This section evaluates the SOA4All Lightweight process modelling and execution approach, based on LPML and SOA4All Design Time Composition Environment (DTCE) and Execution Environment (EE). The evaluation is done on per functionality/concern basis, involving either LPML and/or some DTCE and EE tools in the following Sections.

The evaluation is conducted by comparing WP6 Service Construction results against the evaluation criteria that apply for each main functionality or concern. Evaluation results are analysed, and based on those results some corrective/amendment actions are suggested as future work.

3.1 Process Modelling with LPML

The Lightweight Process Modeling Language (LPML) is devised to provide a lightweight methodology and a set of process modeling vocabularies to aid users in their modeling tasks. LPML is the SOA4All language for process modeling and is used in the entire project for process representation, annotation, instantiation, persistence and for facilitating process optimization and execution. Non-experienced end users are able to compose services via connecting the activities and defining their control-flows and dataflow using visual LPML conceptualization elements of the SOA4All Composer (see [43]) within the SOA4All Studio, thereby creating their business processes using the canonical LPML vocabulary on the language level. These models are enhanced by the WP6 components such as the Design Time Composer (DTC) and Optimizer (T6.4). Finally, the enhanced process models are transformed into executable processes and executed by the Execution Engine (T6.5). The final specification of LPML is defined in [44].

During the SOA4All project, LPML has undergone an iterative specification process. The original design described in [4] was based on a detailed state-of-the-art analysis that allowed us to select a number of concepts of other process modelling formalisms with due attention to the correctness and the right balance between the expressiveness of the language and the resulting complexity of the executable process. The initial design has been modified and extended in [5] and [44], respectively. All updates of LPML are motivated by feedback gathered from the technical requirements of the different SOA4All work packages (including the use cases) and on feedback from different usability studies (see section below).

In summary, the LPML implements the following design principles [33]:

- **Abstraction layer:** The abstraction layer comprises the elements *Process*, *ProcessElement*, *Activity*, *Flow*, and *Gateway*. These elements are directly manipulated by the user and might be visualized through graphical symbols.
- **Semantic annotations:** Semantic annotations support the automated discovery, selection, binding, composition, and execution of services. They are based on an ontology and might be attached to any process element including the process itself. A reasoner resolves the semantic annotations. For some process elements, such as *Process*, *Activity*, *Gateway*, *Flow* and *Parameter*, the annotations are mandatory. The user has to provide these annotations comprising information about functional classification, non-

functional properties, preconditions, postconditions, metadata, conditions, requirements, constraints, selection criteria, replacement condition, and contextual information.

- **Context-awareness:** The context defines the environment a process artefact is used in. The LPML uses context information to discover, select, bind, and compose services through the automated fulfilment of properties through context information, to align and adjust the naming of activities through references to ontology-based business entities, and to adjust the process structure through the context references of the process elements.
- **Patterns and templates:** The support for patterns and templates is achieved through indicating the affiliation of process elements to patterns and templates.
- **Activity instantiation:** The activities in the abstract process model have to be instantiated by a goal and by a service for the process execution. Through the separation of the activity description and the instantiation, binding might be done at various stages. The LPML further provides elements for defining service selection criteria and replacement conditions.
- **Data flow connectors:** The data flow connectors perform a data mapping both on syntactic and semantic level. The more complex data manipulations are covered by special connectors, such as merge, split, filter, or count connectors.

The process modelling lifecycle for creating executable processes has been described in [44] and can be summarized as follows [33]:

- **Design Process:** This design process starts with the process modelling performed by the business user. The further steps comprise the generation of semantic annotations, the mapping of goals to activities, the service discovery, the service selection, and the service composition. The result of this design process is the executable process.
- **API:** The API manages the exchange of LPML code. This API abstracts and hides the complexities of the LPML elements and their concrete serialization formats to the programmer. In the context of the LPML, the API supports serialization into an extended form of BPEL but may also be used by 3rd party tools.
- **SOA4All Composer (see [43]):** The Composer is the user interface allowing for the creation, manipulation, and execution of LPML models. It is implemented according to the principles of RIAs. In order to implement the abstraction principle of the LPML, a drawing area and a data area are provided. The drawing area is dedicated to the abstract, graphical LPML layer and implements modelling support functionality, such as hiding gateways. The Composer supports gathering, applying, and visualizing contextual information. Further, favourite lists and wizards are provided supporting the user in modelling and providing information step by step. The data flow modelling is facilitated through support on the specification of data flow connections and according connectors.
- **DTCE:** DTCE enables the flexible and dynamic creation, instantiation, and adaptation of processes at design time. In the context of the LPML, this means binding goals and services to activities, binding services to goals, resolving activities to process patterns

and templates, checking and supporting the matching of activities and services on semantic and syntactic layer, and creating data flow connectors.

- Execution Engine: To execute the LPML processes a special execution engine is needed. The LPML leaves open whether to create an engine executing directly the LPML models or to transform the LPML models into a standard executable process language, such as BPEL, and to be executed on an existing engine. The new functionalities for usage in the context of the LPML comprise the handling of execution-relevant, semantic information, the runtime adaptation and dynamic binding, the dynamic replacement of services, and the handling of user preferences and context.

In the remainder of this section, we analyse and evaluate LPML with respect to the following criteria: (1) Completeness, Expressiveness, Executability, and Correctness, (2) Flexibility, Composability, and Interoperability, (3) Discoverability and Reusability, and (4) Usability for End Users. Please note that some of the results presented are taken from the dissertation thesis of Florian Schnabel that is not yet published [33].

3.1.1 Completeness, Expressiveness, Executability, and Correctness

The concept of lightweightsness of the LPML is realized by adopting the set of elements of more expressive modelling language that guarantee a correct process model: for pragmatic reasons, constructs that would rely on the modeller's expertise to ensure the correctness of a process model are not suitable for LPML. Obviously, beside possible incurred consistency, too expressive modelling elements may also induce complexity into the process model that would be unnecessary. Since LPML has a close alignment to the process construction tools, the design of LPML has been mainly focusing on the usability aspect of it and prevents deadlocks, conflicts of resources, irresolvable race conditions, or endless loops within process models by design. Consequently, the decision has been to retain the following main elements (see [44] for a detailed description):

- *Activity*, including start and end events to represent process tasks
- *Reusable* elements such as process templates that capture existing modelling knowledge
- *Flows* for modelling control and data flows
- Gateways: including OR and AND split and merge gateways
- Conditions including logical expression and semantic language such as RDF
- Mechanism for dynamic *dataflow* handling using SPARQL

The *Activity* element is *adopted* in LPML since it represents the basic unit to perform a task. In such an element, a single unit of task is carried out. The next step follows our introduction of a finite number of iterative processing of task in the *Activity* element. For the convenience of usability, the countable iteration using a finite number of steps to perform a repetitive task (i.e. a for-loop) has been added. Furthermore, due to practical concerns of data set input containing a finite set of data elements, we have augmented the *Activity* with a while-loop, where the termination condition can be set using semantic expression or simple using the

fact to halt the iteration when all data elements in the finite set are processed. For practical reason such as decidability of the condition expression, we have decided use the lightweight syntax of RDF(S) to refer to concepts in a taxonomy.

In the *Activity* element, we have adopted binding concrete services using two methods: by providing a concrete service endpoint to bind it at design time and by suggesting a list of existing and functionally equivalent service candidates for manual selection by the modeller. The *Activity* relies on the *Binding* element to perform these transparent operations. Moreover, different service transport and invocation paradigms such as SOAP and RESTful services are supported transparently. The benefit in this approach lies in the limited amount of technical requirement and assumption on the actual process modeller. Users who may not be familiar with service annotation and semantic service discovery mechanism are supported in the process construction as described in the following sections.

Flow elements that we have adopted are sequences and parallel control flows that are either induced by an OR-split gateway which is synchronized mandatorily by a corresponding OR-merge gateway after the concurrent execution flows rejoin; or they are derived from an AND-split gateway which is mandatorily synchronized by an AND-merge. In fact, we have enforced in the user interaction through the process construction in the SOA4All Composer that a split of either types must be synchronized by a subsequent merge. The reason is that non-synchronized splits would result in an incorrect process model. In more formal representation of a process model, for instance, YAWL or workflow networks, there are other types of splits and merges as well as some other synchronization elements, e.g. a discriminator. Though they can provide more expressive form to model more complex process flows and concurrency conceptually, in practical use, the correctness is not guaranteed by the formalism of the language alone; the process modeller has to take care of the correctness issue, deal with possibly hidden inconsistencies in the process model using such expressive elements.

Concerning the data flow, we have designed an straightforward way to enable the transparent handling of data flow between subsequent process *Activities* (in sequence) or in concurrent execution paths. The mechanism uses essentially the possibility of expressing dataflow conditions and constraints as RDF(S) based syntax that can be queried using standard SPARQL query language. Since the essential issue about dataflow is more about providing a consistent way to handle the transformation of input/output data to and from *Activities*, i.e. service messages which represent the input/output of the actual unit of tasks need to be transformed between the syntactic message space and the semantic space in RDF; we have decided that the most efficient way to handle dataflow operation, e.g. merging, extraction and filtering of certain data in the data stream of messages, is to defer the actual operations to the semantic level where the constituent elements are represented as RDF instances in the semantic space. Using dynamic SPARQL queries, the backend component is able to perform the necessary transformation on the instances such as described before the resulting semantic graph is transformed into its syntactic representation of the service messages in XML. This way the process modellers are assisted with a set of dataflow operations that are predefined and correspond to some SPARQL syntax, to alleviate them from complex definitions and to reduce the learning effort to model dataflow efficiently.

To summarize, the syntactic and semantic correctness of LPML process models is guaranteed through the LPML metamodel and the design rules described above. The two

abstraction layers (graphical and canonical layer) of LPML introduced in [5] are not different models that need to be transformed but have the same underlying process model (the canonical format).

The SOA4All uses cases have provided a series of scenarios that have been modelled to validate LPML in its different design iterations, including the final design of LPML described in [44]. Among other practical language usage, we have chosen scenarios for instance to model business registration and KPI-based process model for conducting user survey. These and other scenarios in the public sector have been used to show practical application of LPML to create correct process models together with the SOA4All tools. Thereby, we have observed that removal of very expressive elements in LPML (compared to BPMN), actually does not significantly limit the possibility of models that can be created in the real-world application domain. It is sometimes necessary to have alternative structure of a process that can be different than a functionally equivalent process modelled with BPMN; however we have not observed conflicts of the model when using LPML to create process that conforms to the WP7 scenarios at hand. In an extended sense, the lightweightsness of LPML is realized in a higher rate to create correct, deadlock-free processes which are still expressive and complete for its intended purpose, i.e. with regard to satisfying the underlying business requirements and constraints of the business case or scenario at hand.

A formal evaluation concerning the completeness and expressiveness using the Bunge-Wand-Weber (BWW) models [40] can be found in [44] with the result that 20 out of the 31 core BWW constructs do not have a direct LPML representation but can either be mapped to existing LPML elements or are not highly relevant for the targeted user groups and use cases. A second analysis in [44] concerns the coverage of LPML with respect to a set of 20 control flow patterns from workflow systems [41] and six communication patterns from Enterprise Application Integration (EAI) systems [42]. The results show that most of the patterns can be supported through the LPML. For the missing ones, it is quite unlikely that business user have the skills to model those patterns.

3.1.2 Flexibility, Composability, and Interoperability

LPML allows the composition of web services into executable process models. It supports both RESTful and WSDL-based services with the prerequisite that these services are annotated with MicroWSMO (see [45]) and WSMO-Lite (see [46]), respectively, following the SOA4All semantic service model. This service model provides an abstraction of services that is independent of the actual service implementation technology. Therefore, LPML even allows the mixture of RESTful and WSDL-based services, increasing the interoperability support in general for Web services (see [2]). In addition to Web services, LPML also supports human task activities that are to be executed by humans and that are integrated seamlessly into the data flow and the control flow of a process (see [44], [47], and [48]).

As described in the [44], LPML can be transformed into other process modelling languages such as BPMN and is therefore interoperable with existing BPM standards.

3.1.3 Discoverability and Reusability

LPML process models that have been created using the SOA4All composer can be stored in an online repository and therefore become accessible for all authenticated users (see [49]).

In the LPML User Survey described in Section 3.1.4.2, a survey question about the desire to share services and processes in a repository resulted in an average value of 3.28 on a 5-point Likert scale. Further, examples of successful business process modelling stimulate users to model their processes (3.78). Survey respondents trust other users to model processes and parts to be reused (3.78). In addition, users think it's useful to search for existing processes and parts to be reused before starting modelling (4.13). Only few respondents (average of 2.13) think, that there's no use of searching for existing process models. Currently, the SOA4All process repository manages name, location, creation and modification dates of a model and can be explored using the file browser integrated into SOA4All composer. In particular, this allows users to create and manage large process modelling libraries that can not only be accessed by the original authors but by larger communities. Therefore, reusing one another's process models (that could be either complete models ready to be executed or mere building blocks) is naturally supported.

Process models that have been prepared for execution in the SOA4All Execution Engine (see [44] for a description of the process modelling lifecycle) are exposed as WSDL-based Web services and as such can be annotated with a (WSMO-Lite) semantic service description following the SOA4All approach described in [46] and [50]. Once annotated, process models become discoverable and searchable for the SOA4All Discovery Engine as described in [51]. In principle, the LPML language elements of a process model could also be exploited directly for discovery purposes, for instance when searching for all process models, which use a certain web service. However, in SOA4All this possibility has not been explored due to resource limitations.

3.1.4 Usability for End Users

The target users addressed by the LPML are end-users. They might be differentiated according to their IT-skills. The following types have been identified [33]:

1. IT Experts: Users that have a significant IT education or significant experience in developing and using software. IT experts have experience in service programming and usage as well.
2. Business Users: Users who have strong business knowledge in other domains than IT. These users use IT systems to support their work and achieve their work objectives. Concerning IT, they have computational needs but limited IT knowledge and no interest in becoming an IT professional.
3. Casual Business Users: These users are neither expected to be IT experts nor experts of another business domain. Even if they are experts in any way they would use the LPML for purposes that are not related to their main line of work. These users might be seen as the lowest common denominator profile.

3.1.4.1 Focus Group Discussions

As has been reported previously in [12] and [52], we have conducted in total three focus group discussions at the Centre for Service Research at the Manchester Business School, involving overall 35 participants where 85% considered themselves as not being expert in software or service development. These discussions were held in an early stage of the SOA4All project and their aim was (1) to gather feedback about the general idea of Web

service composition by end-users and (2) to discuss different possibilities for the modelling of processes in order to collect requirements for the design of LPML and the service composition tools. In the first part, participants reported via a pre-defined questionnaire that they find service composition by end-users to be very useful (mean 4.44 on a five-point Likert scale ranging from one = “disagree” to five = “agree”), to be very efficient (mean 4.12), and potentially easy to achieve (mean 3.12). There was also a strong agreement among participants that different ways of encouraging and supporting service composition by end-users are required to be successful in organizations: participants agreed that successful examples (mean 4.69), training courses (mean 4.38), quality standards and tests (mean 4.31), extrinsic motivational measures (mean 4.15) could encourage end-users to involve in service composition. In the second part, participants were asked to draw a service composition in their own style without any constraints on the way process elements are connected. The analysis of these process models revealed that some participants favoured a control flow oriented representation, while others choose a data flow oriented one. Overall, the control flow approach was received to be easier to understand (mean 4.82) than the data flow representation (mean 3.45), easier to use (4.0 vs 3.45), and more effective (3.64 vs. 3.36). As a consequence, we have decided that the graphical representation of LPML and the process modelling in the SOA4All composer follow the control flow notation (see D2.6.2). Interestingly, all participants used a rather higher level of abstraction in their models, for instance, when processing data sets, it was taken as granted that a process activity should iterate implicitly over each element of the data set rather than having to model an explicit loop for that task. As a consequence from this insight, we have integrated the support for loops into the final version of LPML (see [44]) and defined the modelling of loops over data sets in the SOA4All composer as suggested (see [49]).

Lastly, we presented and discussed our vision of assisted process modelling in the focus group workshops, which uses templates to minimize the effort and technical complexity, abstracts from specific services, and automates as many composition tasks as possible. This vision was rated highest among the participants (overall rating of 4.45 versus 4.0 for control flow and 3.55 for data flow). As a consequence, WP6 designed a process modelling lifecycle that offers as much tool support as possible (see [44] and following sections).

3.1.4.2 LPML User Survey

In order to evaluate the usefulness of lightweight business process modelling and in order to validate the main LPML ideas, we conducted a user survey [33]. The 35 participants surveyed are from SAP, SAP Research, SAP customers in the public sector, University of St. Gallen, City of Winterthur (Switzerland), City of Muenster (Germany), and the SAP BPX Community. They all have to fulfil BPM-related tasks in their work environment and can therefore be classified as domain experts for the large part.

In order to brief the survey participants about LPM an introductory video and slideset has been provided on the start page of the survey. Following, participants had to fill in a questionnaire covering the research questions “How might business users be enabled to model executable processes in a lightweight way?” and “What are the design principles for artefacts supporting the business user in process modelling and executing?” from different angles.

An introducing question about the respondents’ understanding of the general process modelling lifecycle revealed a good understanding (average value of 3.97 on a 5-point Likert scale from 1 = “Strongly disagree” to 5 = “Strongly agree”), revealing that the lifecycle is

consistent.

A question whether LPML is a real innovation revealed an average value of 3.66, whether the solution would be potentially used for business tasks 3.45, whether in a non-business context 3.07, and whether people would share services and processes an average value of 3.28. Furthermore, the question of whether business users would benefit from LPML as a BPM solution revealed an average value of 3.78.

The target users of the LPML are business users. Hence, in order to identify potential users, the survey questioned the BPM experience of the respondents. In particular, the answers of the respondents with an economic background are of interest. The five respondents with an economic background are experienced in – the number in brackets indicate the amount of respondents - Petri Nets (0), UML Activity Diagrams (1), BPMN (3), WS-BPEL (0), YAWL (0), Flow charts (3), and EPC (0). The average value of whether the Composer is easy to use from a business user perspective is 3.40, which is almost the same as for all participants (3.52). The value of whether they understand easily the graphical process representation is 3.40 for business users compared to 4.00 for all respondents, of whether they don't want to be informed about technical details is 3.50 for business users compared to 3.38 for all respondents, and whether they miss important information is 2.40 for business users compared to 2.78 for all respondents.

The question about whether the graphical symbols of LPML are clearly and intuitively understandable revealed an average value of 4.00. A question about users preferring graphical models to textual models resulted in an average value of 3.91. Finally, the question about whether users missed important information in the graphical model showed an average value of 2.78.

Users have also been asked whether they prefer to indicate a service category rather than a concrete service. The average result value is 3.70. However, the trust that search tools will find the best fitting service is low with an average value of 2.87. The business users show more trust with an average value of 3.20. The average result to the question about the preference to select a concrete service by the user himself is 3.74 and for business users 3.40.

In terms of data connectors, people desire the Composer to handle lists of data entries. The according average value is 3.77. The average value of respondents preferring to define data manipulation activities is 3.43. This shows that the respondents prefer to have explicit activities allowing for the handling of complex data flow operations. As well the respondents prefer to use predefined data operators. The according average value is 3.78. Lastly most of the users prefer not to specify a data mapping (average value of 3.39).

The average value of respondents preferring to have no explicit gateways is 3.23 and preferring to draw multiple outgoing or incoming connections for an activity is 3.26. This legitimates the decision not to explicitly represent gateways in the graphical abstraction of LPML.

3.1.4.3 Process Modelling Usability Studies

In a later stage of the SOA4All project, as soon as the SOA4All composer and the WP6 tools were available as stable versions, we conducted several usability studies around service composition. While the focus of these evaluations was more on any usability issues with the SOA4All tools, the graphical representation of LPML as defined in [53] and the process

modelling lifecycle as described in [44] have also been evaluated in parallel.

We carried out a task-based evaluation to test the usability of the wizard-assisted modeling method described in [43], [47] and [14] and measured the performance of users while carrying out typical tasks. For this, we analyzed the number and type of errors users made and recorded the time spent to perform these tasks. In total, we evaluated 12 participants with the result that a wizard, which guides the user through difficult process modeling steps in the SOA4All composer, can significantly reduce the task completion time and the number of errors for complex process models (see detailed report in [14]).

Another evaluation carried out by WP7 investigated the concept of assisted process modelling by automated optimal service selection (i.e., SOA4All composer and WP6 Optimizer). As reported in [14], the participants of the usability study achieved a significantly better process quality and lower task completion time when the WP6 Optimizer automatically selected the optimal services for their specific requirements compared to a manual selection. In addition to demonstrating the importance of assisted process modelling, this also shows that the higher level of abstraction achieved by being able to semantically describe service properties and requirements is a clear advantage of the SOA4All service composition method.

The overall process modelling lifecycle has also been evaluated by a large usability study of WP7 (see detailed report in [14]). This user study took the form of a contextual inquiry in which we collected data about how users interact with the annotation tool (WSMO-Lite Editor), the consumption platform, and the SOA4All Composer in natural situations. Participants were given an end-to-end WP7 scenario description and instructed to develop a service-based application covering three aspects of the development lifecycle: annotation, consumption, and modeling. The modelling tasks focused on modelling the process of surveying citizens, binding services to the right activities in the process model, adding semantic descriptions to the activities, and resolving and optimising a complete process model. The average rating of different aspects of the Composer was higher than those of the other SOA4All studio tools. Users agreed that process modelling is a rewarding task (mean 4.2), which could be attributed to their prior experience with process modelling using SAP NetWeaver. The assisted modelling feature and the automatic optimization were highly rated (mean 4.0 and 4.0, respectively) since they reduce efforts and save time. Also, multiple participants commented positively on the simple graphical representation of LPML and the intuitive definition of control flow and data flow.

Please note that additional evaluation studies involving process modelling are currently ongoing. Results are not yet available at the time of writing this deliverable, but will be reported in [56], [14], [54], and [55].

3.1.4.4 Comparison of LPML with BPMN

The Business Process Modelling Notation (BPMN) [34] is one of the most common languages for process modelling. In the following, we give a short analysis of the usability of BPMN compared to LPML [33].

The main purpose of BPMN models is to facilitate the communication between domain analysts and the strategic decision-making [35][36]. BPMN models are also used as a basis for specifying software system requirements and providing input to software development projects. The modelling procedure in BPMN is performed with a small set of graphical elements, in particular, flow objects, connecting objects, swim lanes, and artefacts enabling

the stakeholders to construct a Business Process Diagram. BPMN itself is not executable. To execute the process models the BPMN models have to be transformed into an executable language.

An analysis of BPMN models revealed that only 20% of its vocabulary is regularly used [37]. Another study by Recker revealed that 36% of respondents only use the core BPMN set to create their process models, that 37% use an extended set of BPMN symbols, and that the remaining 27% use the whole bunch of symbols and expressiveness [38]. These studies clearly revealed the gap between the BPMN surface and the expressiveness in the backend. While BPMN has only the three shapes activity, gateway, and event on the graphical modelling level, the expressiveness and precision for execution purposes allows for a myriad of subtypes of each. The subtypes are distinguished by detailed graphical aspects, such as border style, the symbols inside, and the placement in the diagram. Hence, although the surface seems to be rather simply usable it is complex [39].

Recker further states that a formal education for BPMN is required. Currently only about 14% of the BPMN modellers took part in a training [38]. In addition, in his work, a statistics is provided indicating the uselessness of certain symbols. Furthermore, the event concept is criticised due to too much different types that are difficult to understand from a user perspective [36][38]. Although BPMN has been created to close the gap between describing and executing processes, in practice the models often lack the execution focus [39]. Recker states in his study that about half of the users model processes for documentation purposes [38]. To document and describe processes with a language dedicated for execution overstrains business users. And BPMN does not natively support translating the process documentation into execution aspects.

Modelling language or tool	Percentage of respondents feeling the language or tool highly usable		Percentage of respondents expecting a high training effort	
	All users	Business users	All users	Business users
Flow charts in Office Software, e.g. in MS Power Point	52%	80%	4%	0%
Flow charts in MS Visio	35%	40%	17%	0%
EPC and ARIS	17%	20%	22%	20%
BPMN	43%	60%	35%	20%
WS-BPEL	17%	Not applicable	30%	Not applicable
BPM Suite (e.g. SAP NetWeaver, IBM Rational, etc.)	22%	20%	22%	20%

Table 1: Favourite modelling languages or tools of respondents

In the LPML User Survey described above, we also gathered feedback on existing modelling languages and tools (see Table 1) [33]. Especially the evaluation of BPMN is interesting for the thesis at hand. 20% of business users heard about it and used it. The business user respondents feel it highly usable (60%), don't think that high training effort is required (20%), and none of them faces difficulties. In contrast, only 45% of IT-experts feel BPMN highly usable, 45% think that the language requires a high training effort. The interpretation of the differences is that business users see BPMN as a graphical modelling tool that is easy to use. However, the IT people see BPMN as a graphical modelling tool that has to be enhanced by execution information. Since this requires much more effort, the IT people feel BPMN less usable and expect more training effort to use the language effectively and efficiently. As well, no one of the business users heard about and used BPEL and YAWL that are executable languages. Again, this shows that business users don't think at process execution when modelling processes, as in BPMN for example.

In comparison with BPMN, LPML uses a much smaller set of process modelling elements as described in Section 3.1.1: based on the insights gained from the BPMN studies and the requirements from the SOA4All use cases, we decided for a minimum set of these modelling elements. At the same time, through the model itself and the modelling rules enforced (see Section 3.1.1), we ensured that the process models defined by the end-users via the SOA4All Composer can be transformed automatically into executable processes.

3.2 Assisted Process Modelling

SOA4All Service Construction offers some assisting features to the process modelling; most of them provided by the Process Editor and the DTC and Optimizer APIs. The easiness of process modelling (from the end user perspective) is supported by the LPML languages itself, which has been evaluated in previous section, and the assisted modelling tools. Process Editor is evaluated in [14], this section focuses on the evaluation of the modelling assisted features provided by DTC, while optimization features are evaluated in section 3.5. DTC follows the **machine and human based computation**² principle. The process modelling is a complicate task that DTC cannot resolve it automatically, even starting from a scratch process model. In the experiments, DTC only resolves completely a process model for which we have prepared suitable semantic knowledge (service and process template descriptions) and described properly the activities of the given model. In the other cases, DTC exhausts the design space without reaching a solution model, what is computational costly, when DTC is requested to resolve the whole process model. A more fruitful approach happens when the burden of designing a complex process model is shared between humans and computer based services. This hybrid approach happens when human modellers creates the process model assisted by computer services such as the DTC. To this end, DTC provides a fine grain API that allows solving the complete process model or just some parts of it, such as resolving a single activity, binding an activity, creating the data flow, etc. The modeller can validate or reject the partial changes that DTC introduces in the

²Machine and human based computation principle is defined in [2], page 37, and in [6], page 12.

process model after every API invocation.

Scalability³ is an important evaluation criterion for any SOA4All Service Construction supporting service, such as DTC, which should be elastic enough to attend an increasing number of requests for process modelling. However, DTC was not specifically designed to support elasticity within its internal design, but it leveraged on the SOA4All distributed infrastructure in order to scale up to the required computational power. That is, DTC, as any other SOA4All platform services plugged to the DSB, can scale up thanks to several mechanisms. Firstly, DTC is **stateless**⁴ compliant, whereby every DTC invocation (even subsequent ones from the same client) are disconnected each other. Secondly, the stateless property enables the service container to instantiate several instances of DTC and deal out incoming requests among the instances. Thanks to this feature, the DSB (as any other SOA middleware) can instantiate several DTC instances on demand, distributing among them the incoming requests. In this way, DTC can manage an increasing number of requests on peaks of demand, while DTC resources are disposed of on idle periods. The behaviour of DTC about the elasticity of domain specific knowledge is described in next section.

Efficiency⁵ is a hard requirement, especially when tackling with semantic-based reasoning. Assisted modelling support given by DTC is efficient enough when working at activity level and considering the semantic reasoning efficiency⁶. A different problem arises when the modeller request DTC to resolve the complete process, especially when DTC cannot find out a complete solution without exhausting the whole tree of process models within the solution space. In these cases, DTC returns an incomplete process model within minutes, what is not acceptable for Web based applications, as the SOA4All Process Editor⁷. Increasing the computational power that host DTC only dims the problem. On the contrary, we should assume that DTC should not exhaust the design space, but returning the best-found process model solution within a given maximum time. However, that feature has not being implemented within the current prototype, although its implementation is relative simple.

In next paragraphs, we analyse the **performance** of a single DTC service based on experimental evaluation. The experiments were run in laptop Core 2 Duo T7250 2Ghz, 3 Gb RAM. DTC was configured with two Design Modification Agents (DMA):

- WSML-DMA: it embeds a WSML2Reasoner⁸ and an in-memory knowledge base containing a preconfigured number of domain specific service and process descriptions (PSOM). This agent can resolve LPML process activities replacing them with process templates or binding them to matching services, using semantic inference.
- SLO-DMA: it resolves the data flow connectors between LPML process activities.

³ Scalability is defined in [3], page 15 and [6], page 12

⁴ Stateless is defined in [2], page 23

⁵ Efficiency is defined in [2], page 27

⁶ DTC resolves most of the SOA4All case study activities within few seconds.

⁷ Process Editor is the application client that invokes DTC.

⁸ WSML2Reasoner performance was evaluated in [15].

Since we are evaluating DTC performance alone, we did not configure DTC to use another DMA that searches for service matching in iServe using the Service Discovery (SD)⁹ service, but only relying on its own registered KB of service and process descriptions. We evaluate three DTC operations: resolveProcess, resolveActivity and bindActivity. ResolveProcess is more computational demanding, since it tries to complete the whole process model (including data flow), while the other two are restricted to a single activity.

First run of experiments evaluates DTC performance in resolving the process models created in the context of eGovernment use case. In this case, we use real input process models, process templates, service and process descriptions and domain specific knowledge bases. However, the number of service and process descriptions in the KB is fixed and very small (around fifty). As input models, we use four different draft process models. The three first models correspond to the business registration process. They differ on the global annotations (requirements, preferences and context), resulting on DTC different process models returned. The number of activities of those input models range from five to seven. The number of activities for the completely resolved models range from 10 to 20.

The fourth model corresponds to an email survey process. Next table summarises the results:

⁹ Service Discovery performance is evaluated in [16].

Method	Model	Execution Time	Number of runs
resolveProcess	Model1	24.82	1
	Model2	30.00	1
	Model3	29.17	1
	Model4	58.04	1
resolveActivity	Model1	0.24+/-0.03	6
	Model2	0.20+/-0.05	6
	Model3	0.19+/-0.03	6
	Model4	0.16+/-0.04	3
bindActivity	Model1	0.09+/-0.07	19
	Model2	0.10+/-0.08	16
	Model3	0.1+/-0.08	16
	Model4	0.17+/-0.05	9

Table 2 Response times for DTC method execution in the eGovernment scenario

DTC resolves eGovernment process models within an acceptable time considering a typical expected response time for Web services¹⁰. In case of activity-level methods (resolveActivity, bindActivity), DTC responds within milliseconds. When the number of experiments is greater than one, we also display the standard deviation for informative purposes, but it lacks of statistical validity since the number of runs is small in most of the experiments. Unfortunately, we lack of additional real use case models to experiment with, in order to improve the statistical confidence.

However those experiments cannot provide valid estimations about the performance of DTC under more demanding resources, since: i) the KB managed by DTC in case studies is small, ii) in case of resolving the whole process, the solution space is not exhaustively traversed since a complete solution is found before.

In other to evaluate DTC performance managing a larger KB, we have programmatically prepared the following experimental set. We have created programmatically a KB containing a pre-determined number of service and process descriptions. Each service/process

¹⁰ Google App Engine requests any deployed application to response every request within a maximum of 30 seconds.

description contains randomly selected annotations from the eGovernment ontology: one functional classification annotation, one input and output annotation, one non-functional annotation. We have also created programmatically a repository of a predefined number of process templates and input models (LPML models), containing a configured number of tasks, each randomly described in a similar way we described the KB of service/process descriptions. Additionally, every process model contains random requirements and preferences annotations, taken from the same eGovernment domain ontologies. The size of this KB, that is, the number of services and process descriptions can be pre-configured. This allows us to create KB with different sizes.

We have measured the computation time spent, for different KB, sizes for three DTC methods: *bindActivity* and *resolveActivity* (which work locally on a given process activity) and *resolveProcess* (which works at process level, for each activity in every model posted by DTC onto its local blackboard). For functional details on DTC methods, see [7]. We do not explicitly evaluate herein a fourth DTC method, *generateDataflow*, since essentially it uses the Optimizer core, known as Semantic Link Operator (SLO), which is evaluated in section 3.5. KB sizes range from 10^3 to $2 \cdot 10^4$. *bindActivity* and *resolveActivity* methods are executed 50 times and *resolveProcess* 10 times. Next picture shows DTC experimental performance results:

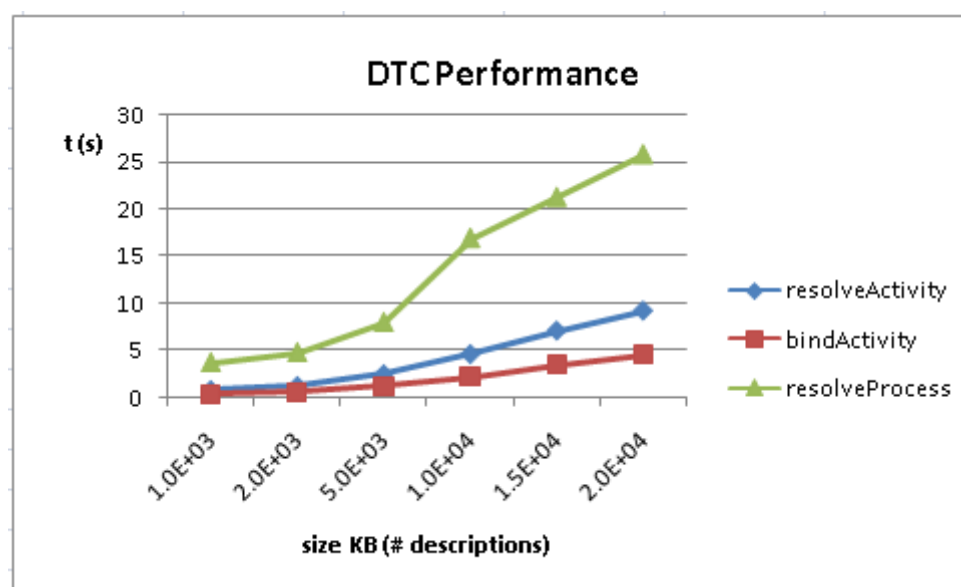


Figure 1 Response times for DTC methods

DTC methods answer within [0, 30] seconds for KB sizes ranging [10^3 , $2 \cdot 10^4$], which are response times compatible with Web applications. *ResolveProcess* method response strongly depends on the number of design models posted by DTC onto its blackboard, and on whether DTC found a complete design model with data flow generated (in this case, the SLO service is also invoked). In case the solution space is completely traversed, *resolveProcess* execution time can get extremely high (some minutes, depending on the solution space size). Guard conditions can be programmed in DTC in order to guarantee a reasonable processing time, relaxing the completeness of the returned solution: Future work

to improve DTC will limit the number of posted design models or just returned the best-found model solution within a pre-configurable processing time.

In the picture, we have limited to three the number of bindings per activities and the number of selected service candidates. For this parameterization, DTC does not need to traverse the model space and returns a best-found model solution within an acceptable response-time (less than a minute). However, the response-time of resolveProcess method is highly fluctuating (see next figure), since, as said before, the response time of these methods strongly depends on the number of design models posted and analysed by DTC before returning a solution model.

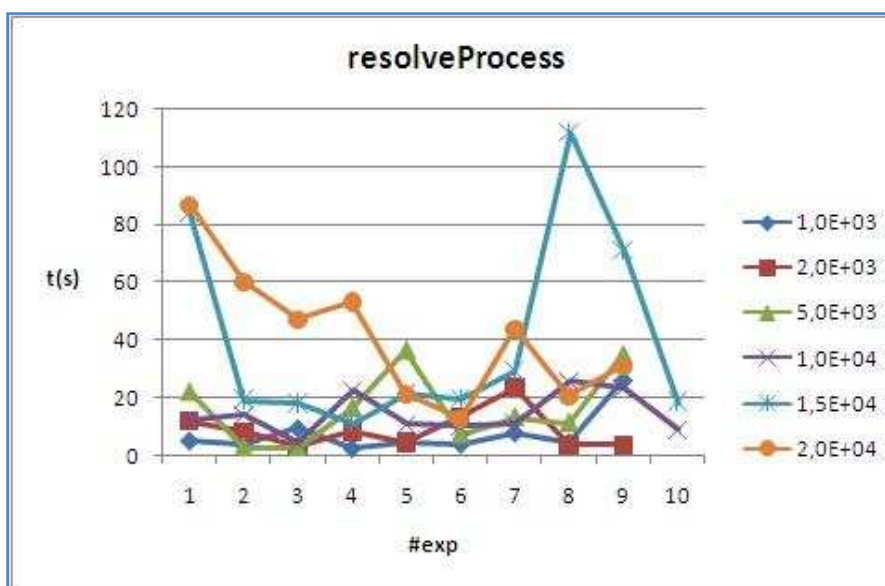


Figure 2 Response-times for DTC resolveProcess method with the KB scale.

In any case, performance of a single DTC service instance, running in an average laptop is reasonable to cope with a relative large ($2 \cdot 10^4$ service/process descriptions) KB and give an answer, in most of the cases within a typical Web application response time.

3.3 Modelling by knowledge intensive reusability

This section evaluates the SOA4All service construction ability to assist process modelling by applying knowledge intensive techniques that reuse existing process fragments or templates.

Template-based composition¹¹ is the key criterion for SOA4All process construction driving this section. It is based on the availability of specific repositories¹² of process fragments or

¹¹Template-based composition is defined in [3], section 4.2.

¹² Some template repositories can be domain specific while others contains general-purpose templates.

templates and their descriptions for further re-usage in other process modelling tasks. According to this principle, recently created process models (and some fragments) can be reused in future modelling tasks¹³.

We have faced with some limitations of DTC to support for template-based composition. The main limitation concerns with the fact that a process fragment or template is, in principle, not executable, as it is. Therefore, we cannot deploy it within the SOA4All Execution Environment to get its WSDL based description. This lack of syntactic description impedes SOA4All end-users to use the SOA4All Studio Provisioning Platform editors to create the counterpart semantic descriptions for process fragments or templates that DTC requires to match them to other activities. Furthermore, since process fragments or templates are not usually executable, we cannot describe them, even manually, using the MSM so their descriptions are stored within the SOA4All iServe repository. Therefore, we require an additional Process Template Model (emulating the MSM) and a Process Template repository (storing their semantic descriptions). Additionally, this requires an editor (resembling the SOA4All Studio Editors) to assist modellers to create these descriptions. These editing features can be reasonably integrated within the SOA4All Studio Process Editor, since the process fragment, once loaded in the Process Editor, already contains most of the semantic annotation required to create its description.

Due to those limitations, the current trade-off adopted in SOA4All service construction has been to rely on hand-crafted descriptions of process fragments or templates that are registered within the DTC knowledge based. Those descriptions use a lightly modified version of MSM and we describe them in WSML-Flight.

DTC component as such is completely **reusable**¹⁴, that is, is completely domain independent concerning the knowledge intensive reusability feature. That implies that as far as there is a domain specific repository of process template models (and their semantic descriptions), registered within DTC, DTC can be used to resolve process models within that domain. In other words, we can parameterize DTC to resolve any domain specific process models, without requiring changes in its code, just access to the appropriate domain knowledge.

DTC is compliant with the **composability**¹⁵ principle. The specific DTC design, based on a multi-agent blackboard approach, allows it to split a complex process-modelling problem into several smaller ones that DTC resolves separately by its specific agents. Indeed, current DTC implementation splits up work-flow from data-flow resolution using separate agents. Fortunately, DTC hides its multi-agent architecture to its users (i.e. Process Editor) that only observe a unique DTC service.

The internal DTC architecture is potentially **distributed**¹⁶, since it is based on a multi-agent paradigm. However, current DTC implementation only works within a single JVM, whereby

¹³ For instance, reusing them as draft initial models that are extended, improved or just modified to fit into another purpose, or inserting them as process fragments within existing activities of a given process model.

¹⁴ Reusability is defined in [2], page 23

¹⁵ Composability principle is defined in [2], page 24

¹⁶ Distributed principle is defined in [2], page 24

we cannot distribute a single DTC service instance into a network of servers. There are two feasible approaches to make DTC distributed within a network. In one of the approaches, a P2P network hosts one blackboard and several registered agents, in different nodes. Agents exchange messages each other using the SOA4All DSB. In this approach, there is only one single DTC entity and a single entry point (i.e. the DTC API). In the other approach, a P2P-like cluster of DTC nodes are arranged, where every node hosts a single DTC instance. A typical DTC client, such as the SOA4All Process Editor, dispatches a request for the DTC to the DSB, which distributes the request to the more appropriate DTC node, based on, for instance, the current payload on the cluster nodes. However, both approaches are different from the operational point of view. In the former approach, the only available DTC instance (which spans across a P2P network) resolves the modelling task. In the later approach, different DTC instances resolve different modelling tasks.

DTC is also compliant with the **openness** principle¹⁷, since DTC can be easily extended either by coding a new DMA/DAA¹⁸ and/or by plug-ing and configuring a new agent that uses its own new domain specific knowledge based of process templates.

DTC is an **ontology based**¹⁹ service, since it extensively uses ontology based knowledge. Even if early prototype of DTC included a rule-based DMA, current DTC prototype exploits semantic based agents²⁰ to resolve the work and data flow of a given process model. Even if automatic composition relies on this ontology-based principle, we observe²¹ some limitations in the DTC implementation since current prototype only supports exact/subsumption matching between the service template that describes the process activity and the service description that represents either the process fragment or the actual service. We should further extend DTC to explore other matching choices such as plug-in and intersection. Otherwise, process modellers are requested to provide precise activity descriptions (i.e. service templates) that exactly match available service and process descriptions. In practice, this is not feasible unless the modeller has a pre-existing knowledge of the available service or process template descriptions²².

DTC satisfies partially the **centrality of mediation**²³ principle. DTC uses ontology mediation to identify the data flow connectors of heterogeneous syntactically mismatches between activity data I/Os. However, SOA4All does not offer mediation support between heterogeneous ontology knowledge bases.

¹⁷Openess principle is defined in [2], page 24.

¹⁸ Design Modification Agent (DMA) and Design Analysis Agent (DAA) are DTC agents that were introduced in [6], [7] and [8]

¹⁹Ontology based principle is defined in [2], page 27

²⁰ Those semantic based agents exploit WSMML based ontological descriptions using WSMML2Reasoner.

²¹ According to the experiments done in the context of the SOA4All case studies.

²² For instance, if the process template or service provider and the new process modeller are the same person or belong to the same team.

²³ Centrality of mediation principle is defined in [2], page 27.

Template Generator (TG) is the SOA4All component that end-user can use to semi-automatically populate the repository of process templates from the post-mortem analysis of process execution logs. Under a user perspective, the main goal of the Template Generator (TG) is to provide a simplified way to derive process templates out of past process execution logs. Several existing tools allow performing such task, but they are usually targeted for expert researchers in the domain of process mining. The most popular tools is indeed PRO-M [17], which is “a framework for the extraction of knowledge about a (business) process from its process execution logs”. The Template Generator exploits the main functionalities of this tool (by integrating its APIs) aiming at enabling non-expert users to perform mining tasks. It is worth to stress that SOA4All technologies have been thought for all kinds of users, as specified in the project acronym, not only for people with a deep skill in data and process mining, as the common users of PRO-M. More in particular, the Template Generator is targeting “process analysts” and “process modellers”.

This big challenge has been faced creating an intermediate layer between PRO-M libraries and the SOA4All user, easing the user interface and hiding some complex details not meaningful and neither probably usable by a person not expert in this domain.

Under a **usability by non-experts** principle²⁴ point of view, after testing Template Generator in the context of the SOA4All use cases, we noticed that the common user needs some time to familiarise with the tool, especially with the algorithm parameters. However, after a small number of tries and iterations over a known set of logs, he is able to achieve interesting and useful results, producing meaningful and exploitable process templates.

Up to current version, Template Generator does not provide detailed and user-friendly explanations of parameters available in the GUI, leaving to the single final user to understand their functionalities through their own knowledge, or mainly, as explained before, through the experience derived from the use of the tool. Some solutions in this direction can be thought as a desirable future improvement, to provide easy and helpful indications under a graphical perspective. Anyhow, we can consider this “training” as acceptable and expected, as the domain in which the tool is located is not trivial and tests to become familiar with it need to be considered necessary.

Additionally, what we have just explained brings to underline how the Template Generator is not an **automatic**²⁵ component. User interaction, user’s choosing skills, user’s domain knowledge are essential elements for a correct production of a useful process template. TG helps the user producing different possible schemas and trying to elicit some knowledge from a set of apparently meaningless execution logs, but the active role of the user is unavoidable.

According the **Ontology-based** principle, to help users understanding the proposed processes, a possible further TG improvement could be the possibility of receiving as input a set of execution logs enriched with semantic annotations. In this way, the user of the TG would benefit of a more meaningful description for the single services composing the process and not just their URL, enhancing the tool usability.

²⁴Usability by non-experts principle is defined in [2], [4], [9].

²⁵Autonomous criteria are defined in [2] and [9].

Moreover, once the user has chosen a possible final process between the ones proposed by the Template Generator, it could be very useful to have some kind of functionality to check whether this schema suits some previous or future execution logs. This comparison could highlight missing paths, un-used branches, bringing to a true testing of the selected template and probably the easiest way to perform a first kind of process optimization.

Under an implementation point of view, TG component is completely **reusable**, as all the business logic behind the graphical interface has been developed and deployed as Web services, without any bound to other modules or parts of SOA4All software. Input logs format is a standard (MXML) and the output is the one foreseen by PRO-M. SOA4All logs import and template storage functions have been implemented upstream and downstream these services, as they can be changed or removed without modifying the core functionalities of the tool.

TG is also compliant with the **openness** principle, as TG algorithms can be easily extended by coding a new plug-in for PRO-M and adding it to the libraries to be included.

3.4 Context-awareness process adaptation

This section evaluates the SOA4All support for process model adaptation based on contextual information. This includes Execution environment adaptation capabilities as described in [9] that support service adaption (interface and data)

Context-awareness in general and context-awareness process adaptation in particular has received low attention by SOA4All, both by the technological providers (WP1-WP6) and the case study consumers (WP7-WP9), even if it was identified as one of the four SOA4All pillars. Apart from the theoretical work done in WP3 to describe an ontology model for context [13], context has only played a role in some concrete examples of process modelling at design time involving WP6 (as context consumers) and WP7 use case (as context providers). That means that other SOA4All features, especially Provisioning and Service Discovery did not provide support to include contextual information in the service description or to perform a context-aware service discovery where contextual information is included as part of the Service Template used as searching criteria in Service Discovery. This lack of support for contextual information in the service descriptions and discovery was not an obstacle to address partially the context-awareness process-modelling feature **at design time**. However, since case studies did not request for this feature at design time, due to most of the relevant contextual information makes sense at runtime, we needed to foreseen meaningful contextual information at design time in any of the case study scenarios. Fortunately, some eGovernment storyboards depicted in WP7 offer scenarios where contextual information at design time is meaningful.

We modelled contextual ontologies using the theoretical framework described in [13] that describes some contextual dimensions for an eGovernment department, whose members participate in the modelling of a Business Registration process model. In particular, we modelled dimensions for preferable payment methods, form reception methods, notification methods, validation methods, language, department locations, currency, etc. In our experiments we were working with an ontology that contained around 35 context dimensions and 40 instances of context entities. Besides, we defined within that ontology contextual instances that we used to populate the contextual information of the actual modeller role in a particular modelling task. Moreover, we manually annotated some process templates

(fragments) and services so they declared to have some sensitivity to some particular contextual dimensions. In other words, we prepared all the ad-hoc contextual knowledge base required to tackle with the context. Since this knowledge is not compatible with current provisioning support in SOA4All Provisioning, we stored these descriptions as knowledge bases registered within the WP6 Design Time Composer (DTC). Additionally, since Service Discovery is not context-aware, we implemented that support in DTC.

We explicitly annotated some business process models in the eGovernment domain with contextual information using the Process Editor (PE). However, PE does not provide support to extract that contextual information implicitly from the modeller profile, for instance, inspecting contextual knowledge bases and obtaining contextual information for her role, department, etc. Therefore, annotations were typed manually using the PE, by selecting concrete contextual instances described on a given eGovernment context ontology, described above. This approach is not intended for average Web end-users but just for evaluation purposes, since requires a deep knowledge of the context ontology.

Experiments showed that DTC resolved unbound activities either a) by replacing them with matching process templates that are sensitive to the same contextual dimensions contained within the modeller contextual information (or other contextual information attached to the process model), or b) by binding them to similar sensitive services.

For instance, given two process models annotated with different contextual information annotations (for different eGovernment modelling departments with different preferable payment method: credit card payment, bank payment) the process models resolved by DTC are completely different. One model checks credit card information during the reception of the input form and invokes a credit card payment service, while another process looks for customer bank details and relies on a bank payment check human activity. This variability on the resulted process model should be hidden to the user since the Process Editor would add the contextual annotations behind the scenes.

The number of experiments done by the Service Construction work package was somehow limited, due to the relative high number of domain specific (use cases) artefacts that are required, whereby the experiments were restricted to those available from the eGovernment case study. Moreover, the lack of support for contextual information in other SOA4All components (Provisioning, Discovery, etc) makes difficult a proper evaluation of the context-aware process modelling by end-user modellers. Nonetheless, the obtained results on context-aware process modelling at design time were satisfactory enough to deserve additional testing and further improvements.

At **runtime context awareness** support provided by the Execution Environment consists of the capability of dynamically bind services to an activity with respect to the current context of the user. Some experiment has been conducted in order to test and evaluate the context awareness of the EE. In the Annex B, the notification process, depicted in Figure 17 and Figure 18, was used in order to demonstrate that the LPML model, supported by the features provided by the EE at deploy time and runtime, can be considered a process model that supports context awareness. Indeed, **Adaptability and Context-awareness** are supported at runtime by the EE exploiting two attributes, introduced in [32], to be added the LPML process description, in particular:

- replacementCondition: represents an element in a taxonomy that defines the set of pre-defined replacement conditions. The replacement conditions are the situations in which the EE will try to substitute a service with another one. If not specified the default replacementCondition is “fault”, other selection criteria are faultAfterRetry and noResponse. In the notification process example, **Error! Reference source not found.** the replacementCondition is “always”. This means that the EE will select a concrete service to be invoked only at runtime.
- selectionCriteria: represents an element in a taxonomy that defines the set of pre-defined selection criteria. The selection criteria are applied by the EE for selecting a substitute service from a list of alternatives. The default criterion is “rating”, other selection criteria are best price or best response time. In **Error! Reference source not found.** the notification process example the selectionCriteria used is “UserPreferences”. It means that the service is selected at runtime in relation to the preferences of the end-user interacting with the current process instance (that in this situation represents the user context).

The contextual information is obtained by EE as inputs of the process or from the outputs of some service invoked during the current process instance execution.

Dynamicity and adaptability of the EE allow for a process design activity faster and simpler. Indeed, selection criteria and replacement condition attributes allow the modeller to:

- postpone some decision about the services to be bounded at runtime;
- reduce the complexity of the initial process graph;
- keep it simple also when some change in the process model has to be introduced.

Replacement conditions and selection criteria adds “dynamicity and adaptability” at the process, context-awareness and flexibility in terms of effort needed for modifying and adapting it to new situations at runtime. Flexibility and adaptation at runtime is dependent from the quality of the contextual model connected to the process, more concepts are identified in the model, more flexible is the process execution, and more adaptation cases are supported at runtime.

3.5 Process optimization

Evaluation of the optimizer has been driven along three main directions: i) **scalability**, ii) optimization **performance** and iii) **quality of optimization**. The latter criteria are the most relevant to evaluate the optimizer component and to run comparisons with existing approaches. Since optimization problems are NP-Hard, it is crucial to evaluate the behaviour of our component in the context of SOA4All i.e., thousands of services. In addition, this level of evaluation has been motivated by the analysis of state-of-the-art approaches which all consider performance analysis with a large amount of services. Finally, this sections aims also at showing the benefits of combining non-functional and functional properties to optimize compositions.

We analyse the performances of our approach by

- discussing, in Section 3.5.2, the benefits of combining QoS and functional criteria;
- comparing, in Section 3.5.3, the evolution of the composition quality's factors over the GA (Genetic Algorithms) generations by considering both static and dynamic constraint penalties;
- observing, in Section 3.5.4, the evolution of the composition quality over the GA generations by varying the number of tasks and candidate services;
- studying, in Section 3.5.5, the behaviour of our approach regarding large scale compositions;
- evaluating performance, in Section 3.5.6, after decoupling the GA and the (on-line) DL reasoning processes which are both required in our approach;
- comparing, in Section 3.5.7, GA with IP (Integer Programming)-based approaches; and
- focusing, in Section 3.5.8, on the performance of the GA process by comparing the convergence of our approach with the [21].

3.5.1 Context of Experimentation

3.5.1.1 Services, Semantic Links and their Qualities

The services are defined by their semantic descriptions using an EL (where subsumption, satisfiability are tractable [19].) ontology (formally defined by 1100 concepts and 390 properties) in the Telecommunication domain, provided by a commercial partner. The use of proprietary services has been motivated by the poor quality of existing benchmark services in terms of number of services or expressivity (limited functional specification, no binding, restricted RDF-based description).

On the one hand, we have incorporated estimated values for QoS parameters (i.e., price and response time). The QoS values of services were varying according to some Gaussian distribution [20], [21], [27] function, and better duration time offers corresponded to higher prices. On the other hand, the functional quality of semantic links (i.e., common description rate and matching quality) has been automatically computed, given semantic descriptions of services and a DL (Description Logic) reasoning process. These descriptions have been randomly selected from the ontology but satisfying the semantic descriptions of tasks they are supposed to be assigned.

Compositions with up to 30 tasks and 35 candidate services per task (i.e., a potential number of 35^2 candidate semantic links between each pair of tasks) have been considered in Sections 3.5.2, 3.5.3, 3.5.4, 3.5.6 and 3.5.8, especially for obtaining convincing results towards their applicability in real (industrial) scenarios.

The quality of the composition is evaluated by means of the percentage gap (described as

Max. Fitness (%) in Sections 3.5.4, 3.5.5, 3.5.8, and % of Global optimum in Section 3.5.7) of the GA (Genetic Algorithm) solution with respect to the global optimum. The latter is obtained by running the IP (Integer Programming) approach with no time limit.

3.5.1.2 Genetic Algorithm Process

Our GA is extending the GPL library JGAP (<http://jgap.sourceforge.net/>). The optimal compositions are computed using an elitist GA where the best 2 compositions are kept alive across generations. A crossover probability of 0.7, a mutation probability of 0.1 and a population of 200 compositions are used. The roulette wheel selection has been adopted as selection mechanism. Since the GA performance varies with the value assigned to global constraints we use a simple stopping criterion of 400 generations. Indeed, under severe global constraints, our GA may not find a feasible solution. Beyond these values for parameters there is a little deterioration in terms of performance.

For each experiment, GA is executed 50 times and average values are reported. Standard deviation has been observed as always below the 5% of the mean values. Experiments were also replicated on compositions of different sizes and complexity, confirming the results reported below. The experiments have been conducted on Intel(R) Core(TM)2 CPU, 2.4GHz and 2GB RAM.

3.5.2 Benefits of Combining Quality Criteria

In this first experiment (Figure 3), we focus on the benefits of combining QoS and functional criteria. To this end, we study the impact of higher functional quality on the costs of data integration enabling the composition of services. The data integration process aligns the data flow of a composition by manipulating and transforming the semantic descriptions of contents of outgoing message and incoming messages of annotated web services.

Our approach and the method of [21] are compared on ten compositions $c_{i, 1 \leq i \leq 10}$ with $Q_{cd}(c_i) = 0.1 \times i$ (Common Description quality) and $Q_m(c_i) = 10^{i-10}$ (Matching quality) as functional quality to reflect gradually better quality.

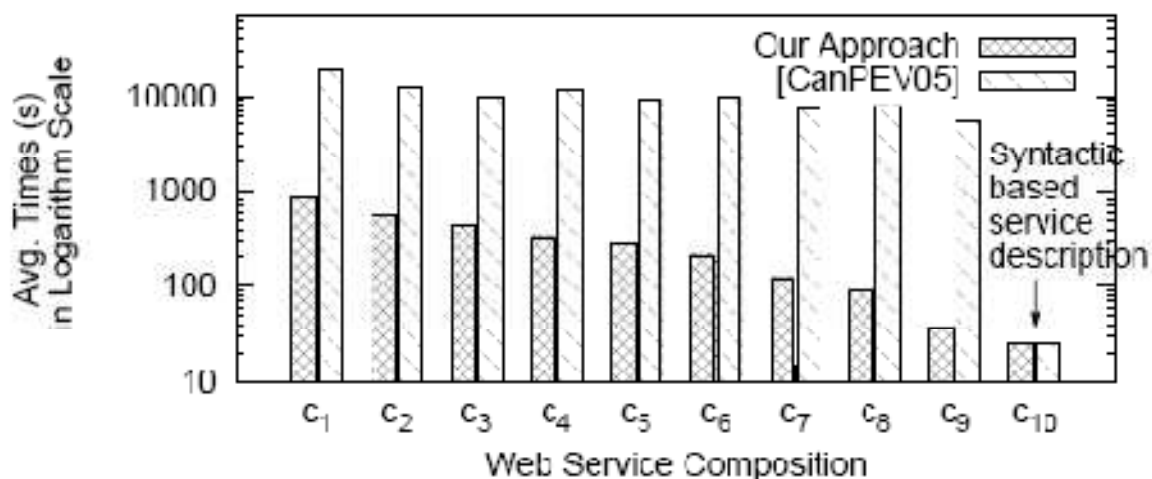


Figure 3 Costs of Data Integration.

On one hand, as expected, the costs of data integration (through data flow) is trivial for both

approaches when compositions with the best quality are considered, e.g., c_{10} and the composition is without mismatch in data flow. On the other hand, these costs decrease with the increase of functional quality of compositions in our approach, whereas they are steady but very high for compositions computed by [21]. This is due to i) the lack of specification of functional quality (hence a hard task to build semantic data flow from scratch), and ii) the manual approach used to link data in the composition. Therefore, appropriate qualities of semantic links are very useful for discovering data flow in composition, then limiting their costs of data integration.

3.5.3 Evolution of Satisfaction Constraints

To compare the different evolution of QoS and non-functional constraints during optimisation, we present results obtained optimising a composition containing 35 candidate services for each of 30 distinct tasks.

As shown in Figure 4, the optimisation problem is constrained on common description rate, matching quality, execution price and response time. The evolution shows how the GA is able to find a solution that meets the constraint and, at the same time, optimises the different parameters of the composition quality function (i.e., maximizing the common description and matching quality while minimizing execution price and response time). For our optimisation problem, the dynamic fitness does not outperform the static fitness. Even different calibrations of the fitness weights did not help. Different tests with significance level of 5% showed that differences were not significant, except for the common description rate where the dynamic fitness increased faster, although at the end of the evolution the result achieved was not significantly different.

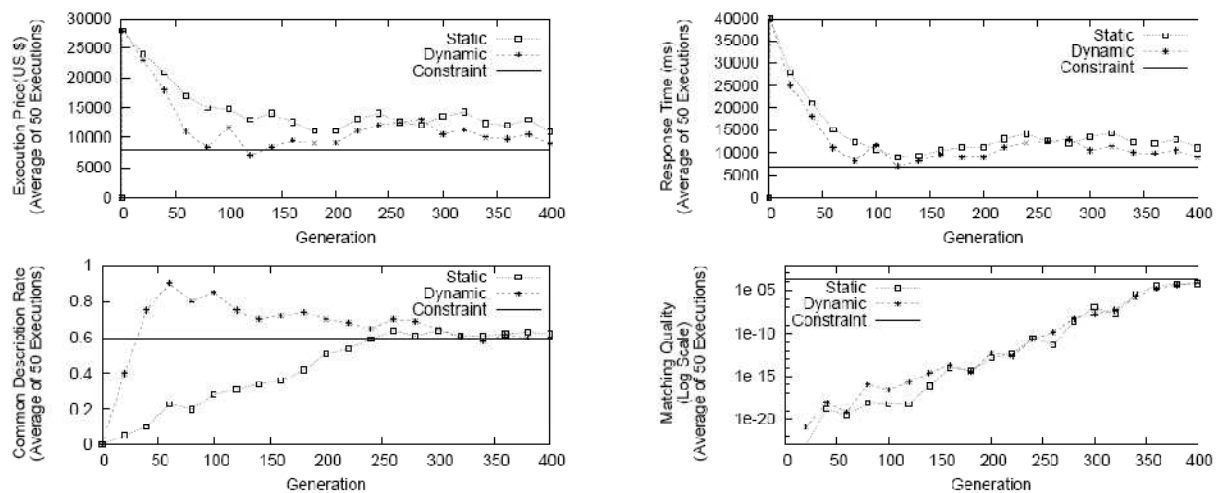


Figure 4 Evolution of Satisfaction Constraints - Static versus Dynamic Fitness.

3.5.4 Evolution of the Composition Quality

Figure 5 reports the evolution of the composition quality over a number of GA generations, for different number of tasks, with 35 candidate services per task. In such a configuration, the potential number of semantic links between two tasks T_i and T_j is $|s_i| \times |s_j|$ with $|s_i|$ and $|s_j|$ be respectively the number of potential services for T_i and T_j . This figure

illustrates different levels of convergence to an optimised composition that meets the constraints.

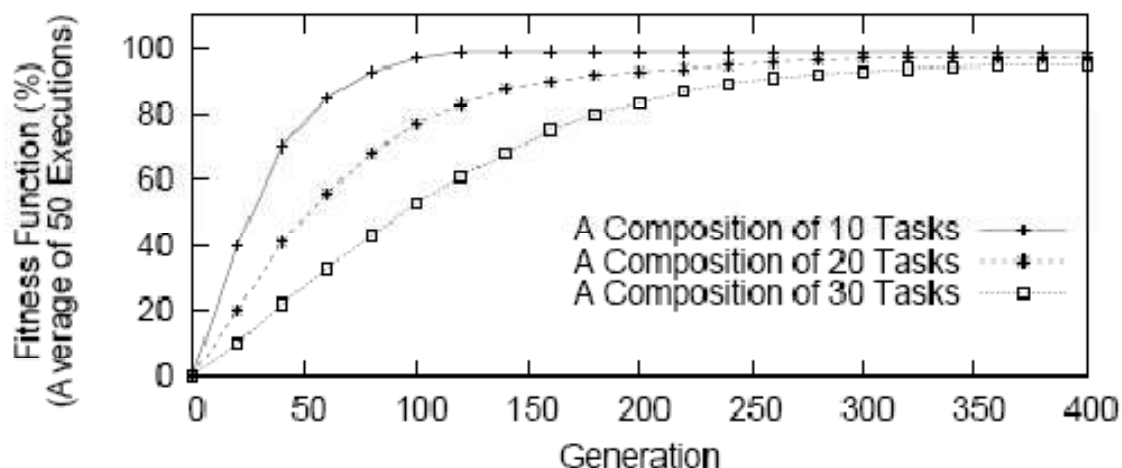


Figure 5 Evolution of the Composition Quality.

Table 3 presents the computation costs, the number of generations required to obtain the maximal fitness value:

Tasks Num.	Max. Fitness (%)	Generation Num.	Time (ms)
10	99	120	1012
20	97	280	1650
30	95	360	3142

Table 3 Overview of Computation Costs.

The convergence results of Table 3 can be improved by further investigating population diversity, evolution and selection policies [25] or enhancement of initial population policy.

3.5.5 Towards Large Scale based Compositions

We study our approach with a large number of tasks (up to 500 tasks) and candidate services (500). We focus on its scalability and the impact of the number of generations as well as the population size on the GA success.

As illustrated in Table 4, increasing both the number of generations and the population size does actually result in better fitness values for problems with a larger number of tasks and candidate services.

Regarding the optimisation of a composition of 500 tasks with 500 candidate services, a number of generations of 400 and a population size of 200 do result in a low fitness value of 24% of the maximum, whereas considering a number of generations of 3000 and a population size of 1000 achieve 95% of the maximum.

Tasks Num.	Max. Fitness (%)	Generation Num./ Population Size	Time (ms)
100	85	400/200	4212
	96	700/400	9182
300	47	400/200	5520
	95	1500/500	19120
500	24	400/200	7023
	95	3000/1000	51540

Table 4 Large Scale Compositions.

Note that better fitness values can be reached by further increasing the sizes of generations and populations. However doubling these sizes only improves the fitness value by 2%. This shows that each optimisation problem converges to a limit. In our GAs based approach, this limit is often identified as a local optimum (approximately 95% of the global optimum in our experiments).

3.5.6 Decoupling GA Process and DL Reasoning

Contrary to QoS given, in general, by providers, the quality of semantic links is estimated according to DL reasoning through Subsumption, Difference and Least Common Subsumer LCS [23]. Since most of the computational costs of our approach are mainly dependent on both reasoning mechanisms and the GA based optimisation process (i.e., the five steps process in [7]), we decouple them and report the breakdown of the computation costs presented in Table 3 Overview of Computation Costs. in Figure 6. For each individual (or composition) of each generation we evaluate the quality of their semantic links. The results of all computations are stored in order to avoid redundant computation of quality of similar links.

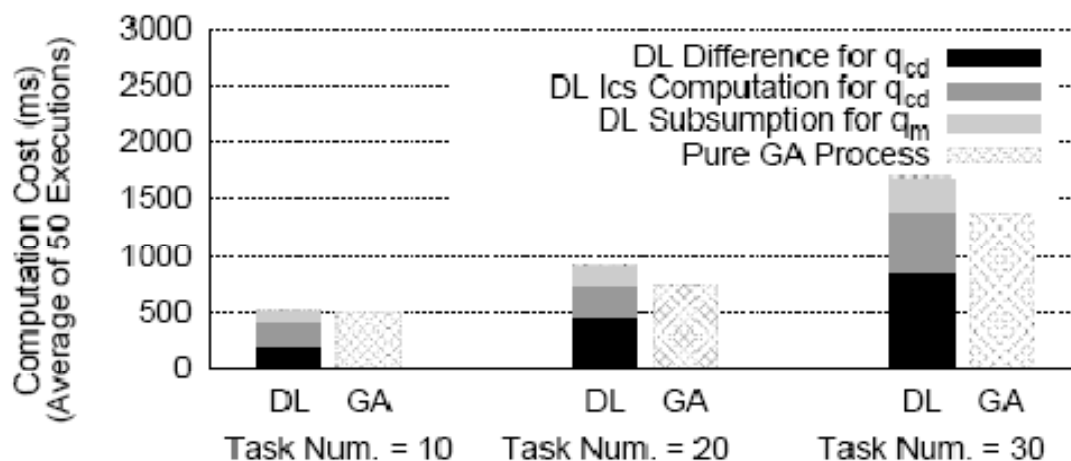


Figure 6 DL and GA Processes in our Approach.

DL reasoning is the most time consuming process in optimisation where the number of tasks and candidate services is greater than 10 and 35. This is caused by the complexity of common description quality computation through combination of DL Difference, LCS and Subsumption even if the subsumption problem is polynomial in EL DL.

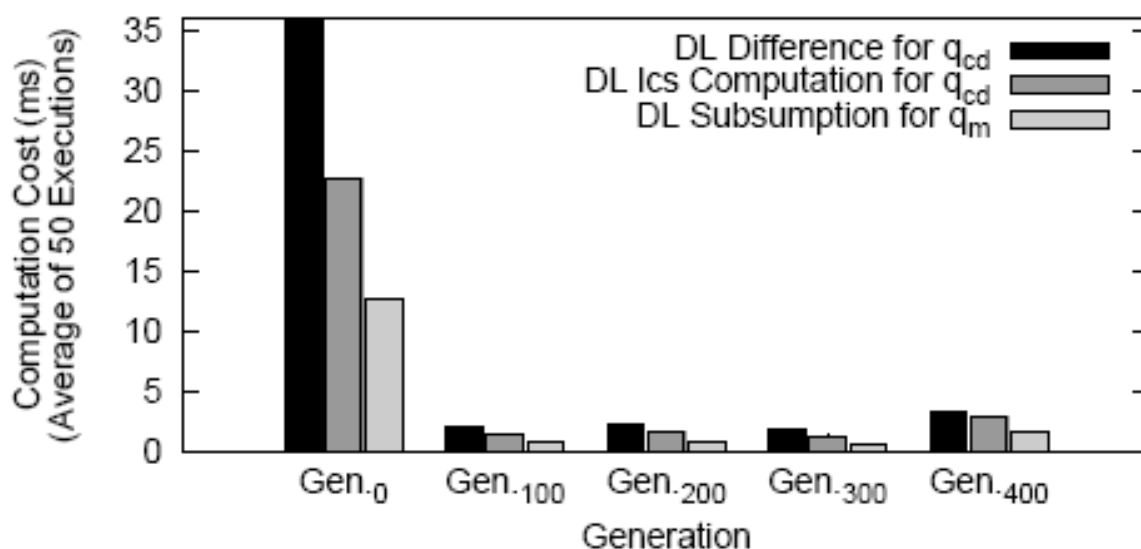


Figure 7 DL Computation Costs along GA Generations.

Figure 7 provides a more detailed view of DL computation costs along the GA generations for a composition of 30 tasks with 35 potential services each (42 semantic links are present). As expected, the DL computation costs are the most important in the initialisation step (see Gen.0) since a quality estimation of 42 semantic links are required to evaluate the quality of each of the 200 compositions. These costs are estimated to 4.2% of the overall DL computation costs along the 400 generations whereas these costs are estimated to only 0.24% for each next generation (mainly because of the low probability of mutation i.e., 0.1). In case of a higher probability of mutation, the proportion of latter costs increases for each generation. The more generations, the less the impact of the DL computation costs during the GA initialisation step. However, in case of reduced problems (e.g., in terms of tasks, services and number of generations), with a low probability of mutation, it should be better to pre-compute some semantic qualities, improving the performance of the whole optimisation process.

3.5.7 Comparing IP and GA-Based Approaches

As mentioned before, the IP-based approach is one of the most adopted method to solve optimisation in web service composition for both semantic and nonfunctional optimisation. An analytic description of this approach is out of the scope of this paper, for details see [27], [22], [24], or [18].

However, here, we compare our approach with the IP-based approach of [27] and [24] by respectively extending their quality criteria to i) semantic links and ii) QoS. To this end we focus on the computation (or convergence) time of both approaches to optimise a revisited fitness function of with close solutions ($< 1\%$ for each quality criteria) meeting the same constraints. The quality criteria of common description and price are unchanged, whereas the matching and response time qualities are linearized (e.g., by taking the logarithm for the aggregation function of matching quality) in order to satisfy the linearity constraint attached to the IP approach. The IP-based optimisation problem is solved by running CPLEX, a state of the art IP solver based on the branch and cut technique [24] (LINDO API version 5.0, Lindo Systems Inc. <http://www.lindo.com/>).

Here we studied a composition of 500 tasks wherein the number of candidate services varies from 1 to 400.

IP and GAs based approaches are compared along two set of experiments. Firstly, we focus on computation costs to obtain a sub-optimal solution that reaches 95% of the global optimum. The results reported in Figure 8 support the adoption of GAs for local optimum search with a large number of candidate services per task (> 340). Such results are, in parts, explained by the exponential number of IP variables required to represent the search tree as the set of potential compositions. On the contrary, the size of the GA problem is bound by the population and its genotype, which is bound by the number of tasks in the composition.

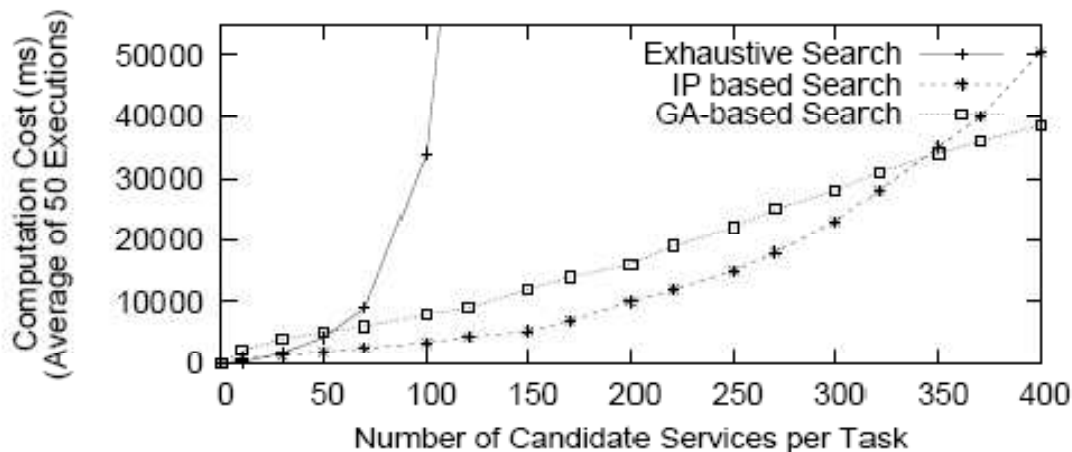


Figure 8 Costs for Reaching 95% of the Global Optimum.

The second set of experiments focuses on computation costs to obtain the global optimum rather than local optimum. The results shown in Figure 9 support the adoption of IP-based composition optimisation, especially in case of a global optimum search. Indeed, CPLEX (also used by [ArdP07]) is very efficient in finding a feasible solution which is very close to the global optimum. On the contrary, reaching the global optimum with GAs is more problematic.

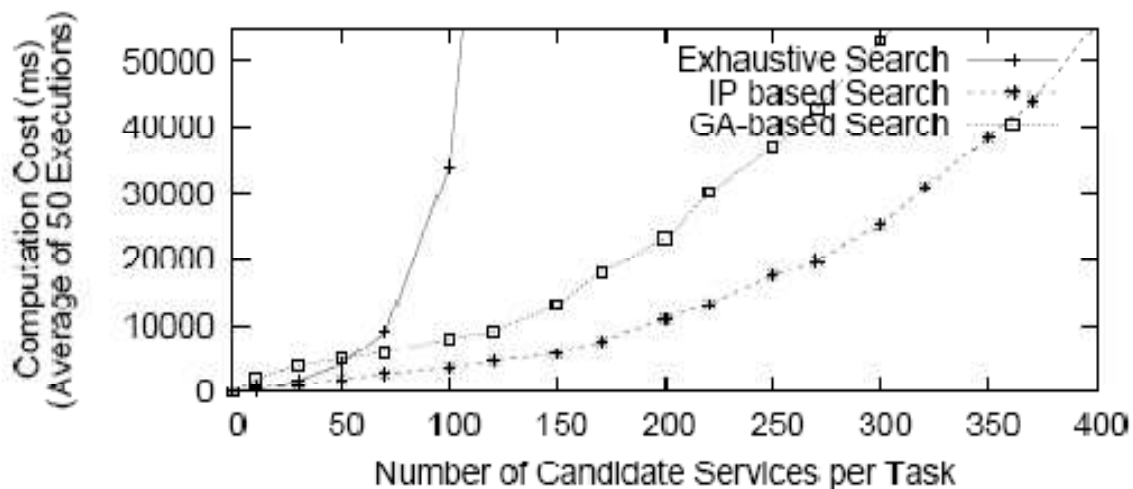


Figure 9 Costs for Reaching 99% of the Global Optimum.

Contrary to linear IP approaches, GAs do not impose linearity constraints on the quality aggregation rules (and thus on the objective function and constraints) and thus allow the handling of generic and customised quality criteria such as matching and response time quality criteria.

According to these results, a valuable direction of further work would be to define a mechanism for selecting the best approach to be adopted for a given composition task and domain, to ensure acceptable computation costs of optimal composition. The latter constitutes an important requirement for many scenarios, such as interactive (or soft-real-time) service-oriented systems.

3.5.8 Convergence of GA-Based Approaches

In this experiment, we compare the convergence of our approach in with the main alternative at present [21]. To this end the functional criteria of our approach are disregarded in order to focus only on the GA-driven aspects of the optimisation process.

Tasks Num.	Approach	Max. Fitness (%)	Generation Num.	Time (ms)
10	Our Model (18)	99	120	1012
	[CanPEV05]	97	156	1356
20	Our Model (18)	97	280	1650
	[CanPEV05]	94	425	2896
30	Our Model (18)	95	360	3142
	[CanPEV05]	85	596	6590

Table 5 GA-based Approaches (Population size of 200).

According to Table 5, the advantage of our approach is twofold. Firstly we obtain better fitness values for the optimal composition than the approach of [21] (actually an average of 97% of the maximum). Secondly, our approach converges faster than the approach of [21]. Our function also avoids getting trapped by local optimums by i) further penalizing compositions that disobey constraints and ii) suggesting a dynamic penalty, i.e., a penalty having a weight that increases with the number of generations.

These results support the adoption of our model when a large number of tasks and services are considered.

3.6 Process deployment adaptation

This section evaluates the SOA4All support to adapt LPML-based models to existing execution environments during the deployment time. Within SOA4All BPEL is adopted as executable language for processes. Since LPML-based models are lightweight models that use semantic annotations in order to simplify the modeller tasks and in order to raise the level of abstraction of the model itself, transforming the LPML model in an equivalent BPEL executable is not trivial. Furthermore, there are more than one possible BPEL executables that realize the same LPML model. In order to address this problem it was decided to adopt a set of conventions in the BPEL generation based on the principles of Template-based Composition and Abstraction.

Template-based Composition decomposes the overall functionality desired into sub-functionalities that are simpler to process, which in turn leads to a solution to the initially complex business problem. In the EE to simplify the translation from a business model of the

process to an executable model, for each process the deployment phase consist in the translation from the LPML process description to an executable process as described in [11] and LPM2BPEL generates some artefacts with a predefined model template.

Abstraction principle and independency of description with respect to implementation are the guidelines for the process deployment. An example is the invocation of SOAP and REST services (presented in [11]), each service invocation, described as an activity in the business process, is translate in an invocation to ServiceExecutor, which represents a single general interface for the invocation of the two types of services supported in the EE. The solutions agreed with the abstraction principle defined in [3]: details of SOAP and REST execution implementation are hidden to the service consumer.

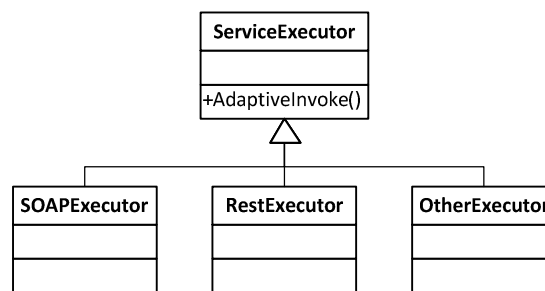


Figure 10 SOAP and REST Service support in the EE

The general interface depicted in [Figure 10] requires only the essential information (described in [11]) that the LPM Process description should provide to the EE in order to execute the service. The interface simplify the process deployment phase because enable automatism in the translation between the LPML and the executable BPEL. The solution is also extendable to other different service types, by adding the appropriate component able to invoke it.

The deployment process has been fully tested against two sample scenarios: i.e. Process 6 and process 7 described in Annex B.

As for the scenario described by Process 6 the deployment process is able to generate all the BPEL executable elements needed for executing the process, e.g.:

- BPEL variables for the inputs and outputs of each service to be invoked;
- BPEL utility variables for process internal computation;
- BPEL invoke activities statically bounded to the ServiceExecutor service and dynamically bounded to a REST service;
- BPEL assign elements that prepares the invoke activities;
- BPEL invoke activities statically bounded to a SPARQL engine service used for solving data flow between activities;
- BPEL assign elements that prepares the invocation of the SPARQL engine service;

-

As for the process 6 it is possible to automatically deploy and execute it.

As for the scenario described by Process 7 the deployment process is able to generate a BPEL model that still requires some manual fix in order to be actually executable.

The lessons learned from the two case studies implemented are:

- it is possible to obtain an executable process starting from the LPML representation and hiding a lot of technical aspects to the modeller;
- the SOA4All EE benefits from the exploitation of a standard BPEL engine because it is possible to move part of the process complexity in some service (i.e. the service executor) realizing a proper separation of concerns;
- the SOA4All EE benefits from the exploitation of a standard BPEL engine because it is possible to reuse lot of its functionalities and its scalability and performance optimization characteristics;
- at the moment the limitations in the automatic translation from the LPML model to the BPEL executable are mainly related to the structure of the process (e.g. cycles and parallel flow execution are not supported at the moment);
- the SOA4All EE benefits from the exploitation of a standard BPEL engine because even in complex cases (where it is not possible to automatically generate the complete executable BPEL) it is possible to manually fix the generated BPEL model exploiting a standard BPEL editor;
- in order to obtain an automatic adaptation process deployment for most of the possible situations it is necessary to test a great number of processes according to a spiral development lifecycle where, in every iteration, new cases are supported.

Template-based composition and abstraction solutions implemented in the EE increments the usability by non-modelling experts by raising the level of abstraction and solving many interoperability and configurability problems deriving from the management of different service types and heterogeneous data formats (as described later in this document).

3.7 Execution autonomous capabilities.

This section evaluates the SOA4All Execution Environment autonomous self* capabilities as described in [2] and [10].

Autonomic Computing principles and the idea of an EE able to **self-manage** itself, is very important for the EE, especially because it executes composition with services offered by third parties. EE do not have any control on the way the services are provided, Therefore, EE is equipped with **self-* capabilities** that allows it to react to the cases in which services provides incorrect results or are unavailable. These capabilities are in the Self-Adaptation Framework, the core part of the EE and described in [10], and implements self-adaptation

mechanisms using two main sub-components:

- The Binder: responsible for executing binding (and re-binding) actions at runtime based on the directions defined by rules. This component is able to execute various policies for selecting the candidate services. For instance, the services could be selected from a predefined list. The selection could be based both on functional and non-functional attributes expressed as Goals in the SOA4All terminology.
- The Adapter: responsible for executing adaptation actions at runtime to enable the Hybrid process support described below.

Self-protecting capabilities are supported by the EE using the binder and the adapter that proactiveness are able to detect fault or errors that can be occurs during the invocation of one of the service involved in the process and react configure themselves automatically.

Self-configuration capabilities regard the reaction of the EE to fault or errors that can be the service substitution in case or alternative service list available or a predefined action depending from the error, to execute correctly the composition even in case of the unavailability of third part services. The behaviour of the EE requires more effort from the process designer at design time because it should configure the alternative service list, but simplify and reduce the tasks that otherwise should be provided manually by the SOA4All users of the process at runtime and raise the possibility that the process designed is automatically and successfully executed at runtime.

Autonomic capabilities have been tested since the first prototype of the EE as described in the Annex B. The WeatherForecast process example developed has shown that how to improve dynamic binding of services at run-time by exploiting semantic annotations of service descriptions, as described in [30]. In a second version of the WeatherForecast process example, described in [29], the need for the execution of self-adaptive processes was still valid. Furthermore, in the SOA4All industrial use case scenario proposed by the WP8, the self-adaptability of the EE was tested since the scenario provided a couple of services (a SOAP one and an REST one) for each activity in the process.

From these experiments we showed that the SOA4All Execution Environment improves usability (by “non-IT-expert” process modellers), reusability and flexibility as it exploits the semantically-enhanced service descriptions provided by the service providers. At the same time it has to be noted that most of the complexity of implementing self* capabilities of the EE has not disappeared but has moved from the process modelling activity to the service description formalization. This issue is addressed by SOA4All since it provides powerful languages and user friendly tools for this purpose.

A second lesson learned from developing these experiments is that, without considering SOA4All results, it was difficult to find and use freely available services to be used in a self-adaptive process. This was due to the low number of available web services that provide a meaningful description or at least a correct description. For this reason, in order to enlarge as much as possible the service alternatives, a lot of work in SOA4All was done for enabling the execution of process able to invoke different (in principle all) kinds of web services. This is evaluated in the following section.

As for scalability and performances evaluation has the following main points:

- EE uses BPEL for the process execution, BPEL is the de-facto standard for the composition and orchestration of Web services;
- EE uses a standard BPEL engine. Thus scale as standard commercial and open source BPEL engine scale;
- EE uses also some extensions included in a set of web services and the corresponding BPEL fragments added ad translation time (e.g. the data flow execution trough the SPARQL engine or the LILO through the grounding service). Thus the scalability/performance of the EE depends also on such tools.

3.8 Hybrid process support

This section evaluates the SOA4All Execution environment support for hybrid (REST and/or WSDL) process execution as described in [11].

The aspects that are being evaluated are:

- The generality of the solution;
- The extensibility of the solution;
- The percentage of cases that can be handled based on the services that are actually deployed on the internet and the relative weight of this percentage.

To evaluate the **generality of the solution** we compare the types of services supported in the EE in terms of “protocols and styles” and “data formats”.

By using the Lifting and Lowering approach described in [11] and by adding the semantic annotations to the service description for REST and SOAP service as described in [11]and [10] the EE support REST and SOAP services with data formats in RDF, XML and JSON.

As described in [28] finding, interpreting and invoking Web APIs requires extensive human involvement due to the lack of API machine-processable descriptions, for these reason it is difficult to say **how many services that are actually deployed on the internet are handled from SOA4All**. However, we consider the following results to be representative, since the source: ProgrammableWeb is currently the biggest directory of Web APIs.

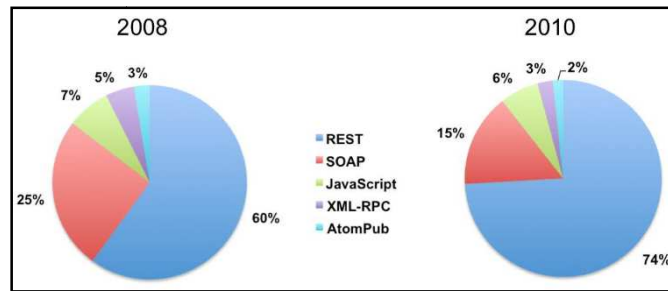


Figure 11 Distribution of API protocols and styles²⁶. Source: [31].

By comparing the distribution of protocols and styles used in the EE with the ProgrammableWeb information we can say that the large part of the service types present in the web can be supported.

Another aspect to **evaluate the percentage of supported services** and the **reusability** of the EE in other scenarios with other third part services is the distribution of the different data format. In Figure 12 there is the ProgrammableWeb classification.

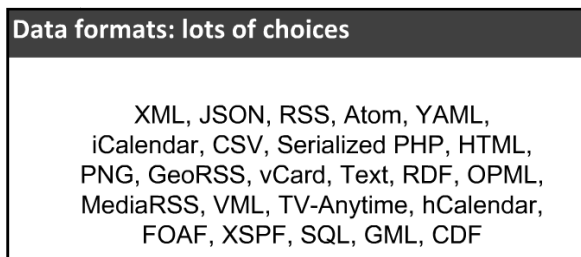


Figure 12 Possible data formats.Source:[31]

The JSON, XML and RDF are the data formats supported in the EE, and JSON and XML are the most popular for ProgrammableWeb and for the analysis of Web APIs on theWorldWideWebXMLconducted in [28].

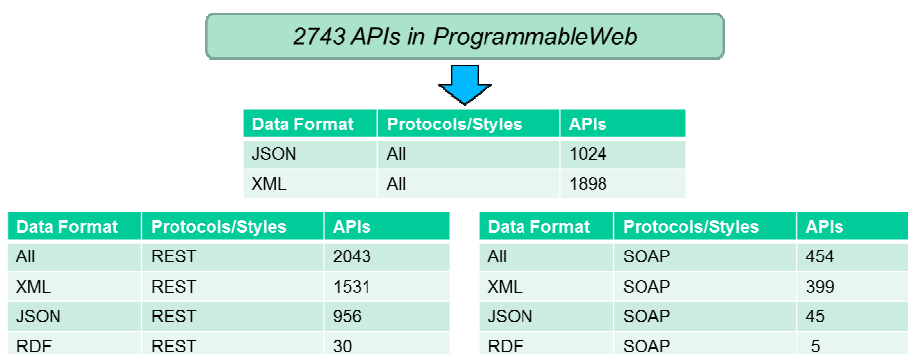


Figure 13 Most popular protocols, styles and dataformats²⁷

²⁶ Based on directory of 2300 web APIs listened at ProgrammableWeb, November 2010

²⁷Based web APIs listened at ProgrammableWeb, November 2010January 2011

Another relevant aspect is that the percentage of new APIs is using JSON.

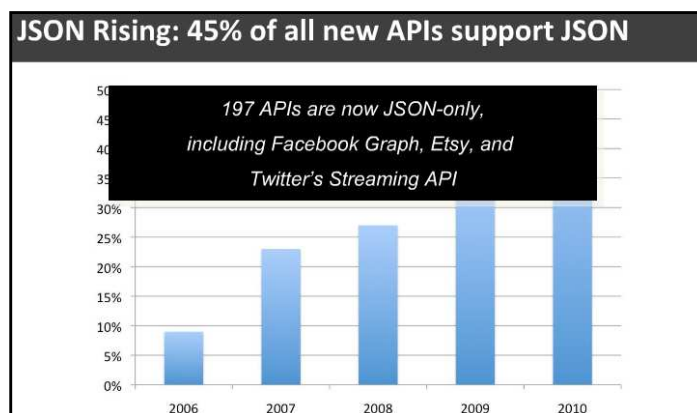


Figure 14 Percentage of new APIs with JSON support ²⁸[28]

Not all the service present on the web are supported from the EE without a bit of effort during the publishing inside the SOA4All environment, this due to the lack of standard present for the REST service description and web APIs in general. For example as for the URI Template specification used in the EE²⁹ was under review at the time of the implementation and REST service description are not machine readable.

Service Information	SOAP Service	REST Service
Service Description URL	SAWSDL; WSDL	MicroWSMO
Service Name	wSDL:service	wl:Service
Operation Name	wSDL:operation	GET; PUT, POST
URI Template	-	gregorio-uritemplate-03
Lowering Schema	XSLT	XSLT
Lifting Schema	XSLT	SPARQL CONSTRUCT query; XSLT

Table 6 What EE supports

EE supports all the typologies of services requires for the WP7, WP8 and WP9 use cases and other services with similar characteristics: same types of operations, lowering and lifting schema expressed in XSLT or SPARQL query, types of service description (see [Table 6] and for other details see [11] and [10]).

As depicted in [Table 7] each WP scenario cover a different aspect for the **hybrid process support** and EE.

²⁸ Based on directory of 2300 web APIs listened at ProgrammableWeb, November 2010

²⁹ For more information, see <http://tools.ietf.org/html/draft-gregorio-uritemplate-03>.

		WP7 Scenario	WP8 Scenario	WP9 Scenario
Protocols and Styles	SOAP Service	✓	✓	✓
	REST Service		✓	
Data Formats	RDF Data Type		✓	✓
	XML	✓	✓	✓
	JSON		✓	
Schemas	Lowering		✓	✓
	Lifting		✓	✓
Process Elements	Data Flow	✓	✓	✓
	Condition	✓	✓	✓
	Human Tasks	✓		

Table 7 SOA4All scenarios from the EE point of view

In order to support invocation of services and to **extend the solution to other services** EE requires a minimal set of data described in [Table 6]. SOA4All provides tools for annotating web APIs to recognize that information. From results coming from discovery of services, process design and web service annotations tools depend the right invocation of services and the **extensibility** of the solution that is the possibility for users to extend and integrate new scenarios.

4. Conclusions

This document has detailed the results obtained from the technical evaluation conducted by WP6 to assess the Service Construction Suite. This evaluation was driven addressing some evaluation criteria that was identified from the main SOA4All principles and requirements, including principles from the SOA4All main pillars, technical requirements, use case requirements and additionally some comparison with regards the state of the art.

In this evaluation we have assessed the main Service Construction Suite features, in particular the Lightweight Process Modelling Language and tool features such as the assisted process modelling, knowledge intensive reusability, context-awareness process adaptation, process optimization, process deployment adaptation, execution autonomous capabilities and the hybrid process support.

We can conclude that SOA4All Service Construction Suite provides sound features addressing ICT-unskilled target users who are looking for a lightweight modelling and execution framework for business processes implemented as a composition of semantically annotated services. This feature is particularly well supported by the Lightweight Process Modelling Language and its assisting tools.

Moreover, the SOA4All Service Construction Suite provides solid and efficient implementation for the SOA4All requirements and addresses the SOA4All principles.

Nonetheless, we foreseen additional improvements, left for future work, that are described within the document as well.

5. Annex A

This table collects the most relevant evaluation criteria (scored with medium-high relevance in a 3-Likert scale) for Service Construction Suite:

Criterion	Sub-criteria	Defined in
Autonomy	Self-healing, Self-configuration	D1.1.1, D6.5.1
Flexibility	Machine and human based computation, Human task, different levels of complexity	D6.1.1, D6.3.1, D6.4.1
Dynamicity and adaptability	Context-awareness	D6.1.1, D6.3.1, D6.4.1, D6.5.1
Usability by non-experts	Template-based composition, ontology-based, centrality of mediation, problem solving	D1.1.1, D6.3.1, D6.5.1
Reusability		D1.1.1, D6.3.1
Statelessness		D.1.1.1
Discoverability		D1.1.1
Composability		D1.1.1
Openness		D1.1.1
Interoperability		D1.1.1
Scalability		D1.1.1, D6.4.1
Executability		D1.1.1
Efficiency		D.1.1.1
Optimization	Functional quality, Non Functional quality, Constrained based	D6.4.1, D6.4.2, D6.4.3

Table 8 Service Construction Evaluation Criteria

6. Annex B

This section describes all the processes generated during the project in order to test and evaluate the EE characteristics at different stages of the EE implementation. Table 9 lists, for each experiment described, the EE characteristics evaluated through the use cases scenarios realization.













	Context-awareness	Autonomous capabilities	Hybrid process support	Process deployment adaptation
P1 - WeatherForecast				
P2 - SOAP/RESTWeatherForecast				
P3 – LILO process				
P4 - Notification Process				
P5 – Business Registration (WP7)				
P6 – Play Media To Call (WP8)				
P7 -Get Product List (WP9)				

Table 9 Processes and evaluation criteria.

Process1: WeatherForecast

WeatherForecast process is the first process used to validate the capabilities provided by the first prototype of the EE (see [9] and [30]).

The scenario implemented by the Weather forecast process is to support car drivers in discovering the weather conditions at the destination place and at the estimated arrival time. In the scenario, users select a destination on their car navigation system, and weather forecast information are displayed together with the result of the trip planning.

The WeatherForecast process uses real services freely available on the Internet. In detail, the **process implemented involves two weather forecast SOAP services with similar capabilities but different interfaces**. The two candidate services considered in the case study do not have any internal or conversational state.

Key aspects within process design and execution:

- dynamic and adaptive reconfiguration in reaction to environmental changes;
- dynamic binding of services at run-time;
- automatic generation of adaptation script used to solve mismatches between services;
- SAWSDL description based a shared ontology for SOAP Service;
- System integrator selects the candidate services to be included in the composition.

Process2: SOAP_REST_WeatherForecast [D6.5.2] and [DegRZ10]

A second version of a WeatherForecast process was developed in order to introduce the need of using both SOAP and REST services inside a process (see [10] and [29]). The scenario implemented in this second version of the process lies in creating a weather forecast service that returns a weather forecast given as input the name of a city.

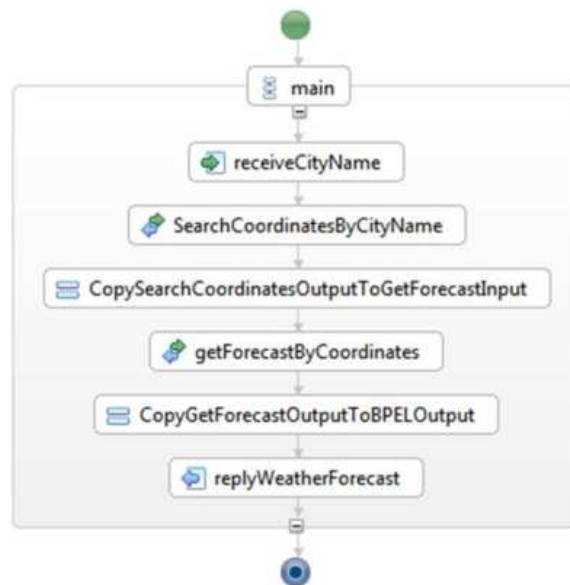


Figure 15 SOAP_REST_WeatherForecast Process

It executes in sequence the following steps:

- receives the city name string as input;
- invokes the search operation passing as input the name of the city, and obtaining as output the coordinates of the city, i.e., latitude and longitude;
- the output of the previous activity is passed to the following activity as input;

- invokes the ForecastService, service passing as input the coordinates of the city, i.e., latitude and longitude, and obtaining as result the weather forecast for the city;
- the output of the previous activity is passed to the following activity as input;
- returns the weather forecast.

The process is composed by two services:

- a REST Service that given a city name returns its geographical coordinates;
- a SOAP Service that requires as input the geographical coordinates of a location and returns the weather forecast.

The process was first implemented in the EE v2. The novelty introduced by the EE v2 was an early version of the support of both SOAP and REST Services within a process. With this process it was proved, even with some limitations on the characteristics of the services supported, the EE v2 was able to replace at runtime a SOAP services with a REST one, and vice versa. In order to solve mismatches between the two types of services EEv2 introduced the support to semantic description languages for SOAP and REST services: i.e. SAWSDL and MicroWSMO.

Key aspects within process design and execution:

- invocation of SOAP and REST services within a single process;
- run-time service replacement in case of error with services from an alternative service list;
- invocation of REST Service with MicroWSMO description;
- invocation of SOAP Service with SAWSDL description;
- invocation of REST Service only if they exchanges XML parameters.

Process3: LILO

LILO Process describes a simple process composed of only one activity. The process invokes getProductList operation of the ProductWebService. This service is the one developed in the WP9 in order to implement part of the Web Shop Catalogue scenario.



Figure 16 LILO Process

The LILO process was implemented/tested using the EE v3 (see [11]) that introduced the execution of Lifting and Lowering to/from RDF (besides the integration with other SOA4All

components like the Process Editor, the Analysis Platform and the SPARQL Engine). The LILO process supports lifting and lowering using lifting schemas and lowering schemas in XSL language.

Key aspects within process design and execution:

- Data types in RDF within the process;
- Invocation of SOAP and REST services exploiting lifting and lowering schemas.

Process4: Notification Process

The Notification Process is a sample process that aims at evaluating the context awareness capabilities of the EEv3: i.e. to dynamically bind services to an activity with respect to the current context of the user.

The process has five elements i.e. four invoke activities and a switch:

- the first activity invokes a service responsible of gathering the contact information of a user;
- the second operation is a switch;
 - if the contact information is an email the following activity will invoke a “SendEmail” service operation (second invoke activity of the process);
 - if the contact information is a postal mail address the following activity will invoke a “SendMail” service operation (third invoke activity of the process);
- the last invoke activity is a call back to the client (needed because the process is asynchronous).

If the modeller would like to add another notification method that consists on sending an SMS she has to add a new condition and a new activity. The resulting process model is depicted in Figure 17. This is needed because the process model used in this example is not context aware.

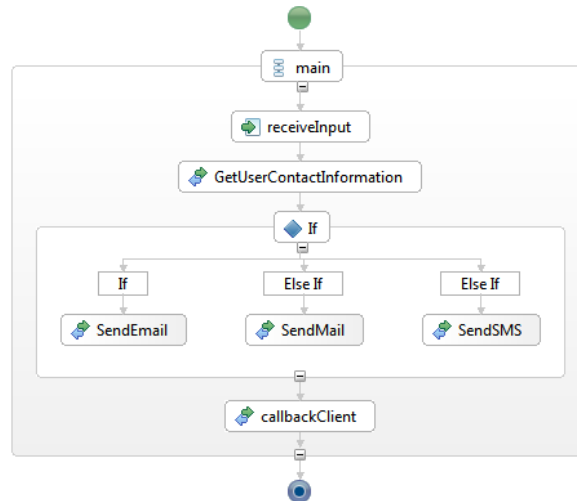


Figure 17 Notification process BPEL Fragment

On the contrary, the SOA4All Execution Engine underlying model is context aware. The simplified context aware executable process model supported by the SOA4All execution engine is depicted in Figure 18. In this new version of the process the SendNotification activity will bind the correct service to be invoked only at runtime, according to the user context derived (in this particular situation) from the result of the GetUserContactInformation activity.

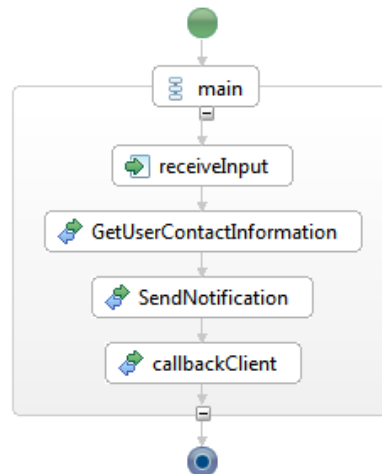


Figure 18 Notification process, BPEL Fragment simplified

Key aspects within process design and execution:

- Adaptability and Context-awareness;
- Services replacement conditions and services selection criteria.

Process5: WP7 Scenario [D6.5.3]

WP7 is one of the three SOA4All use cases and it has the public sector as its target domain. It is a lightweight business processes composed of public Web services (hosted by 3rd party service providers), and human activities (to be executed by end users).

WP7 use case includes one scenario that builds an open service platform for the public sector that can be used by the public servants of an administration to model and execute administrative procedures: the administrative procedure to register a new business at the responsible public administration of the City of X, Germany.

EEv3 Approach for WP7 Scenario:

The WP7 description focuses on the need for combining SOAP services and human tasks inside a lightweight process.

Registration business process is differentiated from other use case scenarios, not only for its complexity (it uses mostly all the modelling primitives available in LPML), but also because it uses human tasks. Human tasks are special process tasks that are performed by humans. Typically, a human task receives some input information (documents, forms, links, etc.) that are analyzed by an external reviewer who, upon task completion, returns a report (task acceptance, explanation, fulfilled form, etc).

When a process reaches a human task, it registers it within the Human Task Server (HTS) and blocks the execution until the task is evaluated and validated by an appropriate reviewer. Human tasks are registered for a particular role. Human task reviewers access the Human Task Server through the Human Task Client, which lists pending human task according to the role of the reviewer. The reviewer analyzes each task individually, considering its input information and producing an approval/disapproval report.

Key aspects introduced or extended within process design and execution:

- mostly all the modelling primitives available in LPML;
- human tasks.

Process6: WP8 Scenario [D6.5.3]

The scenario is based on a service called Offers4All which could be a service offering of a retail division of a telco such as BT Retail or a non-telco company. In the scenario, a composition is created allowing one user to contact nearby users.

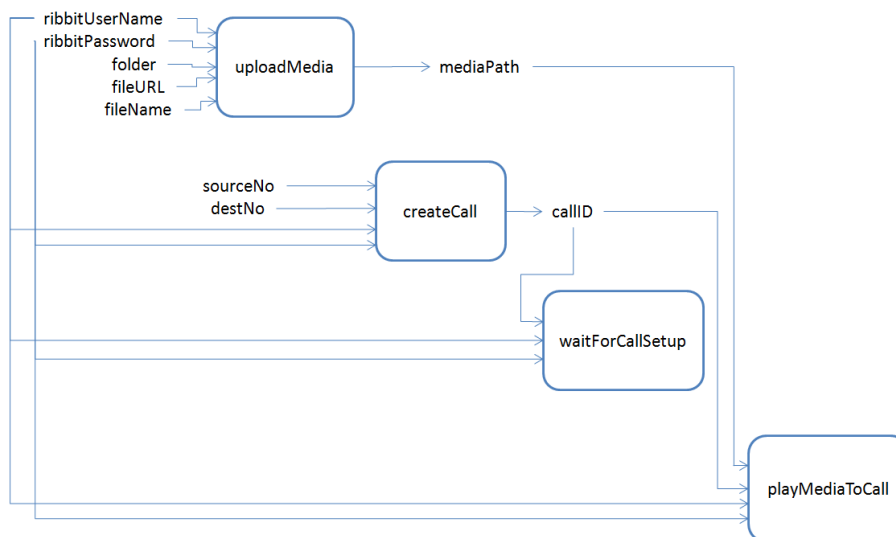


Figure 19 WP8 Scenario

The WP8 description focuses on REST service support through lifting and lowering schema provided by the service provider as part of the service description and on data flow described by the modeller as SPARQL queries

The REST services used in the WP8 scenario are provided with a semantically annotated service description that includes lifting and lowering schemas. Lifting and lowering schemas provided are scripts that can be executed in order to translate a service request or response in an RDF format according to a specific ontology.

Key aspects within process design and execution:

- REST service support;
- MicroWSMO Service Description;
- Dataflow;
- JSON and RDF data types support;
- Process able to invoke and substitute both REST and SOAP.

Process7: WP9 Scenario [D6.5.3]

The WP9 scenario shows how the SOA4All results may be applied in the eCommerce domain. A web shop catalogue is updated from three different sources that provides a service, with different interfaces but similar functionalities, for retrieving a list of products.

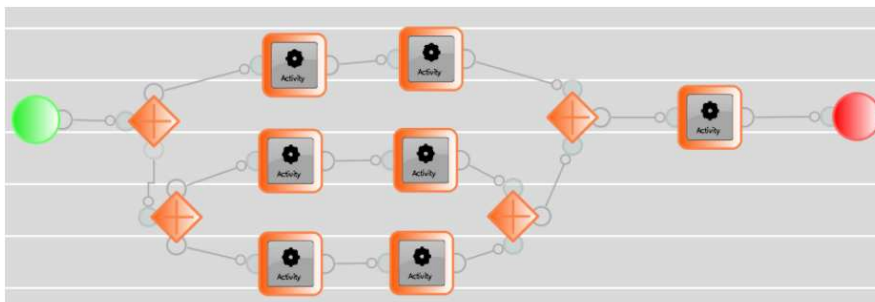


Figure 20 WP9 Scenario

The WP9 scenario needs parallel execution flows and synchronization via a gateway for invoking the three services and merge the results in a single list of products. Basically this means to model and execute a process similar to the one depicted in Figure 15.

The translation of the diagram should basically be to execute the 3*2 activities in parallel and afterwards to invoke the last activity. Activities in the WP9 scenario are realized using SOAP services.

Key aspects within process design and execution:

- parallel execution flows;
- synchronization via a gateway.

7. References

- [1] Quality Model for NEXOF-RA Pattern Designing. <http://www.nexof-ra.eu/?q=node/358>.
- [2] SOA4All D1.1.1 Design Principles for a Service Web. Di Nitto, E et al. <http://www.soa4all.eu/file-upload.html?func=startdown&id=25>
- [3] SOA4All D6.1.1 State of the Art Report and Requirements for Service Construction. Vogel, J et al. <http://www.soa4all.eu/file-upload.html?func=startdown&id=20>
- [4] SOA4All D6.3.1 First Specification of Lightweight Process Modelling Language. Schnabel, F et al. <http://www.soa4all.eu/file-upload.html?func=startdown&id=86>
- [5] SOA4All D6.3.2 Advanced Specification Of Lightweight, Context-aware Process Modelling Language. Schnabel, F., Xu, L., Gorroñoigoitia, Y., Radzimski, M., Lecue, F., Ripa, G. et al. <http://www.soa4all.eu/file-upload.html?func=startdown&id=127>
- [6] SOA4All D6.4.1 Specification and First Prototype of Service Composition and Adaptation. Gonzalez-C., R., Lecue, F., Villa, M. <http://www.soa4all.eu/file-upload.html?func=startdown&id=87>
- [7] SOA4All D6.4.2 D6.4.2 Advanced Prototype For Service Composition and Adaptation Environment. Gorroñoigoitia, Y., Radzimski, M., Lecue, F., Villa, M., Di Matteo, G. <http://www.soa4all.eu/file-upload.html?func=startdown&id=207>
- [8] SOA4All D6.4.3 Final Prototype For Service Composition and Adaptation Environment, Gorroñoigoitia, Y., Radzimski, M., Lecue, F., Villa, M., Di Matteo, G. 2010. <http://www.soa4all.eu/file-upload.html?func=startdown&id=239>
- [9] SOA4All D6.5.1 Specification and first prototype of the composition framework. Ripa, G., Zuccala, M., Mos, A. <http://www.soa4all.eu/file-upload.html?func=startdown&id=88>
- [10] SOA4All D6.5.2 Advanced Prototype For Adaptive Service Composition Execution. Ripa, G., De Giorgio, T., Gorroñoigoitia, Y., Mos, A., Ravoajanahary, F. <http://www.soa4all.eu/file-upload.html?func=startdown&id=208>
- [11] SOA4All D6.5.3 Final Prototype For Adaptive Service Composition Execution. Ripa, G., De Giorgio, T., Ravoajanahary, F. <http://www.soa4all.eu/file-upload.html?func=startdown&id=240>
- [12] SOA4All D2.5.1 Formative Evaluation and user-centred Design, Abdallah Namoune, Nikolay Mehandjiev, Freddy Lecue, Usman Wajid, Linda Macaulay. 2009
- [13] SOA4All D3.4.7 Defining extensions to WSMO for capturing contextual information. Grenon, P. <http://www.soa4all.eu/file-upload.html?func=startdown&id=138>
- [14] SOA4All D7.7 UseCase Architecture, Integration & Validation. J. Vogel et al. 2011.
- [15] SOA4All D3.2.7 Second Prototype for Description Logic Reasoner for WSML DL v2.0. D. Wrinkler and M. Pressning, 2010
- [16] SOA4All D1.5.3 Testbeds Validation, 2010. B Schreder et al. 2011.

-
- [17] PRO-M framework. <http://prom.win.tue.nl/tools/prom/>
- [18] [ArdP07] D.Ardagna and B.Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Software Eng.*, 33(6):369–384, 2007.
- [19] [BaaBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the envelope. In *IJCAI*, pages 364–369, 2005.
- [20] [CanDEV08] G.Canfora and M.DiPenta and R.Esposito and M.L.Villani. A framework for QoS-aware binding and rebinding of composite web services In *Journal of Systems and Software*, 81(10):1754-1769, 2008.
- [21] [CanPEV05] G.Canfora, M.DiPenta, R.Esposito, and M.L.Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceeding of Genetic and Evolutionary Computation Conference*, pages 1069–1075, 2005.
- [22] [CarSMAK04] J.Cardoso, A.P. Sheth, j.A. Miller, J.Arnold, and K.Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004.
- [23] [CohBH92] W.W. Cohen, A.Borgida, H.Hirsh Computing Least Common Subsumers in Description Logics In *Proceeding of Association for the Advancement of Artificial Intelligence*, pages 754–760, 1992.
- [24] [LecDL08] F.Lecue, A.Delteil, and A.L'éger. Optimizing causal link based web service composition. In *Proceeding of European Conference on Artificial Intelligence*, pages 45–49, 2008.
- [25] [MaZ08] Y.Ma and C.Zhang. Quick convergence of genetic algorithm for QoS-driven web service selection. In *Proceeding of Computer Networks*, pages 1093-1104, 2008.
- [26] L. Wolsey. *Integer Programming*. John Wiley and Sons, 1998.
- [27] L.Zeng, B.Benatallah, A.H. H. Ngu, M.Dumas, J.Kalagnanam, and H.Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327, 2004.
- [28] [MalPD10] M. Maleshkova, C. Pedrinaci, and J. Domingue, *Investigating Web APIs on the World Wide Web*, 2010 Eighth IEEE European Conference on Web Services, Dec. 2010, pp. 107-114.
- [29] [DegRZ10] T. De Giorgio, G. Ripa, and M. Zuccalà, *An Approach to Enable Replacement of SOAP Services and REST Services in Lightweight Processes*, ICWE Workshops, Springer, 2010.
- [30] [CavRZ09] Cavallaro, L., Ripa, G. and Zuccalà, M., *Adapting Service Requests to ActualService Interfaces through Semantic Annotations*, PESOS Workshop, Vancouver, IEEE Computer Society Press, 2009.
- [31] Open APIs: State of the Market, John Musser, December 2010<http://www.slideshare.net/jmusser/open-api-ecosystem-overview-december-2010>
- [32] [SchnabelEtAl10] F. Schnabel, Y. Gorrongoitia and M. Radzimski, F. Lecue and N.Mehandjiev, G. Ripa, S. Abels and S. Blood, A. Mos, M. Junghans and S.Agarwal, and J. Vogel - Empowering Business Users to Model andExecute Business Processes - The

Third Workshop on Business Process Management and Social Software, BPMS2010, 2010

- [33] F. Schnabel, Lightweight Process Modelling – A Methodology, Language, and Tools, PhD Thesis submitted to the University of St. Gallen, Switzerland, July 2010, to appear.
- [34] OMG, Business Process Modeling Notation Specification, OMG Final Adopted Specification dtc/06-02-01, 2006.
- [35] Becker, J., M. Rosemann, et al. Process Management, Springer-Verlag New York, Inc., 2003.
- [36] Recker, J. C., M. Indulska, et al., Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN, 2005
- [37] zur Muehlen, M. and J. C. Recker. How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation, Springer, 2008.
- [38] Recker, J. "BPMN Modeling – Who, Where, How and Why." BPTrends 5(3): 1-8, 2008
- [39] Silver, B. BPMN Method and Style, Cody-Cassidy Pre., 2009.
- [40] Wand, Y. and R. Weber. An ontological evaluation of systems analysis and design methods. Information System Concepts: An Indepth Analysis. E. D. Falkenberg and P. Lindgreen: 79-107, 1989.
- [41] van der Aalst, W. M. P., A. H. M. ter Hofstede, et al. "Workflow Patterns." Distributed and Parallel Databases 14(3): 5-51, 2003
- [42] Ruh, W. A., F. X. Maginnis, et al. . Enterprise Application Integration: A Wiley Tech Brief, John Wiley and Sons Inc., 2001.
- [43] SOA4All D2.6.1 Specification of the SOA4All Process Editor. Delchev, I., Vogel, J., Abels, S., Puram, S. <http://www.soa4all.eu/file-upload.html?func=startdown&id=79>
- [44] SOA4All D6.3.3 Evaluation and Final Design of the Lightweight Context-Aware Process Modelling Language. Un, P., Genevski, P., Gorrongoitia, Y., Radzimski, M., Ripa, G., Mos, A., Norton, B., Lecue, F. <http://www.soa4all.eu/file-upload.html?func=startdown&id=238>
- [45] SOA4All D3.4.6 MicroWSMO v2 - Defining the second version of MicroWSMO as a systematic approach for rich tagging. Fischer, F., Norton, B. <http://www.soa4all.eu/file-upload.html?func=startdown&id=206>
- [46] SOA4All D3.4.2 WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web. Kopecky, J., Vitvar, T., Fensel, D. <http://www.soa4all.eu/file-upload.html?func=startdown&id=84>
- [47] SOA4All D7.3 End User Service Design. Vogel, J., Schnabel, F., Stroh, F., Xu, L., Un, P., Mehandjiev, N. et al. <http://www.soa4all.eu/file-upload.html?func=startdown&id=155>
- [48] SOA4All D7.5 End User Service Implementation. Vogel, J., Neu, B., Pavlov G., Lecue F., Gorroñoitia Y, 2010

-
- [49] SOA4All D2.6.3 Advanced prototype of SOA4All Process Editor. Vogel J, et al. 2010
- [50] SOA4All D2.1.4 Service Provisioning Platform Second Prototype. Maleshkova, M., Alvaro Rey, G., Simov, A., Renie, B., Liu, D. <http://www.soa4all.eu/file-upload.html?func=startdown&id=229>
- [51] SOA4All D5.3.2 Second Service Discovery Prototype. Agarwal, S., Junghans, M., Norton, B. <http://www.soa4all.eu/file-upload.html?func=startdown&id=236>
- [52] SOA4All D2.5.2. Summative Evaluation Report, Abdallah Namoune, Usman Wajid UNIMAN, Nikolay Mehandjiev. 2010
- [53] SOA4All D2.6.2 SOA4All Process Editor First Prototype, Vogel J, et al. 2010
- [54] SOA4All D8.7 Evaluation of Prototypes. 2011 (To be released)
- [55] SOA4All D9.4.2 Evaluation of Prototypes. 2011 (To be released)
- [56] SOA4All D2.5.3 Final Summative Evaluation of SOA4All Studio 2011 (To be released)