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

D6.4.3 Final Prototype For Service Composition and Adaptation Environment

Activity 2:	Core Research and Development	
Work Package 6:	Service Construction	
Due Date:		31/08/2010
Submission Date:		31/08/2010
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		ATOS
Revision:		1.0
Author(s):	Yosu Gorroñogoitia, Matteusz Radzimski (Atos), Freddy Lecue (UNIMAN), Matteo Villa, Giovanni di Matteo (TXT)	
Reviewers:	Sven Abels (TIE), Gianluca Ripa (CEFRIEL)	

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	04/06/2010	ToC	Yosu Gorroñogoitia (ATOS)
0.2	18/06/2010	First Contributions	Freddy Lecue (UNIMAN), Yosu Gorroñogoitia (ATOS), Matteo Villa (TXT)
0.3	03/08/2010	Update	Mateusz Radzimski (ATOS)
0.4	18/08/2010	Corrections after peer Review	Yosu Gorroñogoitia (ATOS), Matteo Villa (TXT)
1.0	30/08/2010	Final Editing	Yosu Gorroñogoitia (ATOS)

Table of Contents

List of Figures

Glossary of Acronyms

Acronym	Definition
DTCE	Design Time Composition Environment
EE	Execution Environment
PE	Process Editor
TG	Template Generator
LPML	Lightweight Process Modelling Language
DTC	Design Time Composer
SWS	Semantic Web Service
I/O	Input/Output
KPI	Key Performance Indicator
UI	User Interface
PA	Public Administration
ICT	Information and Communication Technology
FC	Functional Classification
NFP	Non Functional Property
SS	Semantic Space
D	Deliverable

Executive summary

SOA4All Service Composition and Adaptation Environment, also known hereafter as Design Time Composition Environment (DTCE) is part of SOA4All Service Construction Suite, consisting of the Studio Process Editor (PE) developed by T2.6 [7, 8, 9], DTCE itself developed by T6.4 [4, 5] and the Execution Environment (EE) developed by T6.5 [6]. DTCE provides full semi-assisted functional support, at design time, to model adaptive context-aware process models using the PE, according to the Lightweight Process Modelling methodology [1][2]. That is, DTCE offers services that complement the PE modelling functionality with advanced features that automate complex modelling tasks, such as process schema extraction, activity binding, data-flow generation, optimization, etc.

This document provides a general, integrated, coherent and holistic functional description of the final prototype of DTCE, close to the modeller perspective. This description is complemented by an enlightening modelling scenario taken from one of the use cases, which is used to illustrate the complete SOA4All Lightweight Process Modelling methodology at design time.

This document focuses on the functional description of the final DTCE, since we aim at remarking the benefits of the SOA4All Lightweight Process Modelling methodology as supported by DTCE tooling, rather than on the research and technical achievements, which were introduced in deep detail in [4] and [5]. Nonetheless, the main technical achievements between M24 and M30 are summaries in the Technical Annex.

1. Introduction

1.1 Purpose and Scope

This document describes the final prototype for the Service Composition and Adaptation Environment, also known hereafter as the Design Time Service Composition Environment (DTCE). In the previous deliverables [4], [5] and the accepted paper [13] we focused on the detailed theoretical background and research, and on the technical design of the tools that build the DTCE. Therefore, during the period between M12 and M24 we have been mainly developing the DTCE tools. In the current reported period (M24-M30) we have stabilized the DTCE tools, we have integrated new releases of dependent components provided by other technical WPs (WP1-WP5) and we have further developed some features introduced in the previous releases, as summarizes in the section 5.

However, this deliverable will focus on providing a comprehensive and coherent functional description of the DTCE as a whole, highlighting the main features offered by DTCE tools to support the Lightweight Process Modelling Methodology [1][2] at design time as it should be perceived from the process modellers' perspective, driven by a concrete example taken from the SOA4All use cases.

This deliverable aims at offering an appealing picture of the SOA4All tool support at design time for the modelling of optimized process models throughout meaningful and illustrating experiments performed in the context of the SOA4All use case scenarios.

1.2 Structure of the document

The rest of this document is structured as follows. In section 2.1, we introduce the complete design time life cycle for the modelling of optimized processes according to the Lightweight Process Modelling Methodology. Furthermore, this section elaborates possible usages of this life cycle at different stages of modelling projects, depending on their purpose. Following subsections further elaborate the concrete phases within the general life cycle, illustrated with a common process-modelling scenario. Section 2.2 explains the process-schema extraction phase, which populates the process template repository with historic knowledge, in order to be reused in future process-modelling projects. Section 2.3 explains the semi-assisted, light annotated driven, context-adapted process-modelling phase. Section 2.4 explains the optimization of processes based on KPI and semantic quality. Section 2.5 introduces a post-mortem analysis of the execution of the optimized process model, aiming at comparing the actual execution with the intended modelling. We highlight the main conclusions in section 3. Section 5 summarizes the main last technical achievements introduced in the DTCE tools during the period M24-M30 and points at the software repository and installation instructions.

2. Design Time Service Composition and Adaptation.

This section provides a detailed functional description (from the perspective of the modeller of the composition) of the SOA4All lightweight process modelling methodology and the DTCE tooling support, illustrated throughout a complete and practical modelling experiment on a concrete SOA4All use case scenario. Next section 2.1 introduces the overall modelling life cycle. Following sub-sections further develop for each concrete modelling phase.

DTCE tool consists of the following tools and services: Template Generator tool, Design Time Composer (DTC) service and Optimizer service. Studio Process Editor, included within the SOA4All studio and reported in [7], [8] and [9] completes the suite.

The SOA4All lightweight process modelling methodology is aimed by the same modelling principles that drove the specification of the Lightweight Process Modelling Language (LPML) [1][2][3]. That is:

- light semantics as the basis for the description of processes and their elements, including the data flow,
- coarse-grain description of the requirements for a process and their activities,
- extensive reuse of modelling blocks, such as process fragments and templates,
- contextual adaption,
- multiple activity bindings, etc.

Those modelling principles will address the semi-assisted modelling methodology illustrated in next sections.

2.1 Lightweight Design Time Service Composition and Adaptation life cycle.

The SOA4All Lightweight Process Modelling methodology is depicted in the next figure.

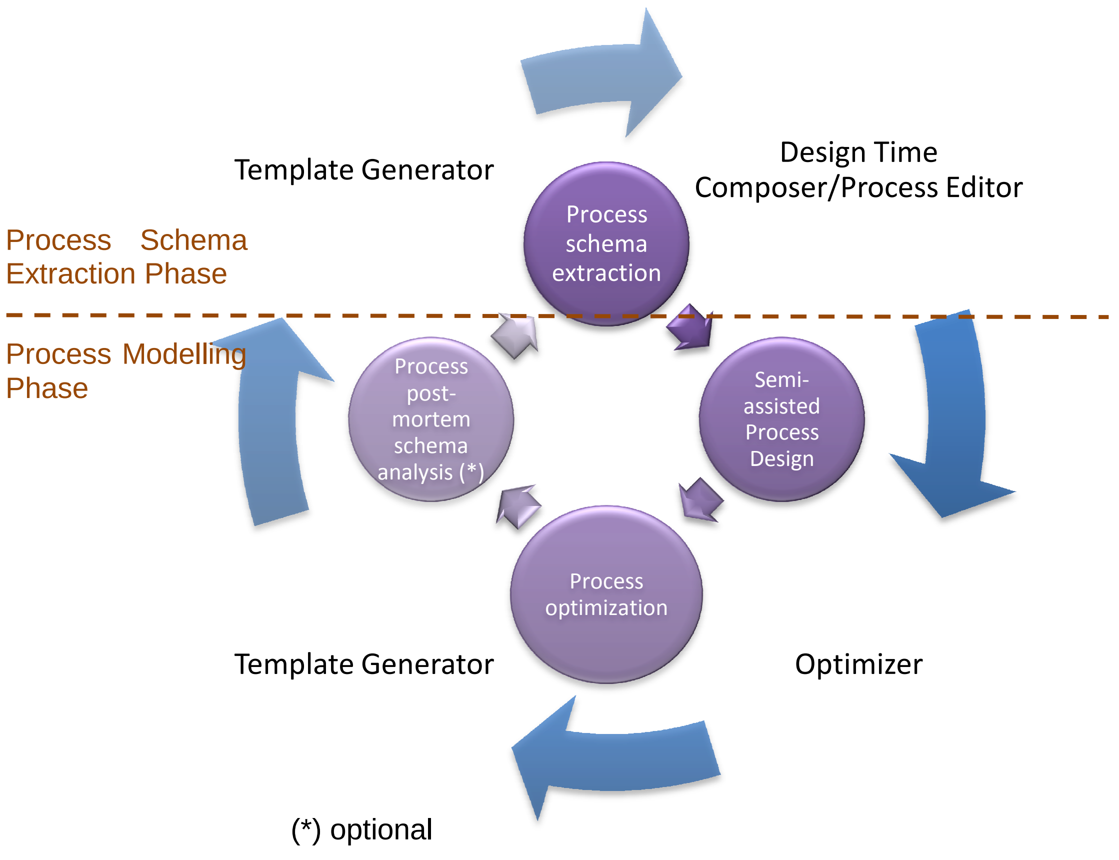

Template Generator
Design Time Composer/Process Editor
Process Schema Extraction Phase
Process schema extraction
Process Modelling Phase
Process post-mortem schema analysis (*)
Semi-assisted Process Design
Process optimization
Template Generator
Optimizer
(*) optional

Figure 1: SOA4All Design Time Process Modeling Life cycle

This iterative methodology distinguishes two main phases, which can be performed at different time, within different modelling projects and not necessarily by the same role:

- A phase for process schema extraction: its purpose is to populate a knowledge base repository of process schemas that model existing running domain processes and/or their fragments. Domain-specific process providers, who have enough domain expertise to analyse pre-executed processes through the post-mortem analysis of their execution logs, perform this task to extract the schemas. The Template Generator tool and the Process Editor tool support this phase.

- A phase for process modelling: in this phase, modellers create new optimal process models in the context of the same or other (ongoing or future) modelling projects, within the same or a different domain, but leveraging on the knowledge extracted in previous phase. The Process Editor tool, the Design Time Composer (DTC) service, the Optimizer service, and optionally the Template Generator tool support this phase.

This second main phase is properly the process-modelling phase itself. During this phase, a modeller produces an optimal executable process model ready for deployment. Optionally, executions corresponding to the deployed process can be analysed by post-mortem techniques in order to derive concrete execution schemas that are compared to the intended and modelled process, in order to understand how far the actual executed process differs from the modelled process.

The process-modelling phase consists of two main sub-phases (excluding the aforementioned optional post-mortem process analysis phase):

- An iterative semi-assisted process design phase, supported by the Process Editor tool and the DTC service. During this phase, the modeller produces a suboptimal executable process model, since she focus on the modelling aspects: control and data flow.
- A process-optimization phase, supported by the Process Editor and the Optimizer service. During this phase the modellers optimizes the process model created in the previous phase.

In both cases, the modeller uses the Process Editor tool to **manually**:

- Create the control flow of the process model.
- Describe the process model (globally) and its modelling elements (locally) through lightweight semantic annotations.
- Bind suitable SWS to activities.
- Create the data flow mapping, etc.

The modeller invokes, through the Process Editor, the DTCE services (DTC and Optimizer) in order to automate some modelling tasks, leveraging on the lightweight semantic annotations she included manually in the process model and on the available domain-specific knowledge bases (repositories) populated with information about SWS descriptions, process fragments and templates, ontology models, context models, etc.

During the process-modelling phase, the DTC service automates some cumbersome modelling tasks, such as binding SWS to activities, expanding unbound activities with matching sub-processes, creating the data flow connectors, checking the semantic I/O compatibility, adapting the process model to contextual information, etc.

During the optimization phase, the Optimizer service replaces the current binding set for each activity within the process with a new one, which offers a better global cost function for the process, according to both the semantic quality between the I/O of correlated activities and some global KPI specified by the modeller.

The optimized process model is ready for deployment within the SOA4All Execution

Environment [6] using the Deployer Service through the Process Editor UI.

This completes the overall description of the Lightweight Process Modelling methodology at Design Time, as supported by the DTCE tools. Next sections will go deeper into the details of each phase describing the creation of a process model corresponding to one of the SOA4All use cases, using the DTCE tools.

2.2 Process schema extraction

We show in this section how the Template Generator is used in order to create new processes schemas or templates[1].

We will go through a concrete example: the scenario is derived from the SOA4All WP7 scenario "Registration of a Business" [10], but it is referred to the situation prior to the adoption and use of the SOA4All tools. In this scenario, we have the public administration of the "City of X" dealing with requests from citizens to open new activities / businesses.

In the initial situation, requests from constituents are arriving to the public administration (PA) offices in several ways, and not following a precise order:

- via written form
- via e-mail
- via fax

Sometimes requests also arrive by phone; although this is not considered as the proper way, the PA is still handling such requests as "exceptional cases" of urgency.

Depending on the way the requests arrive, further actions may be necessary:

- in case of requests via e-mail, it may happen that e-mails have a valid digital signature, but in most of cases this is not happening, so PA employees have to get in contact with the requesting citizen to get confirmation about his identity. In most of cases, email turn to be sent just for spam purposes, or the citizen identity cannot be verified.
- in case of written, fax or e-mail requests, the PA is performing a check on the requested location for the new business, based on the information provided on the request form
- in case of requests via phone, the PA is directly checking the location with the citizen over the phone

A further set of checks by the PA civil servant takes place, as described in [10] check general lawfulness, check identity of the constituent, check legal form of the new business,

[1] Schemas or templates are used in this document with the same meaning. While process schema is more appropriate in the Template Generator context, process template is more commonly used during the modelling phase.

and a final check on the operation allowance. If some of these checks is not passed the process is terminated with a failure.

If checks are passed, the PA gets informs the tax office and communicates the result to the citizen sending an invoice. Currently the PA is contacting the citizen in the same way he provided his requests, so either by written reply, by fax, by mail or by phone.

Anyway, in case of phone requests, the PA is calling back the citizen to perform the checks via phone with the citizen.

The scenario is represented in the following picture:

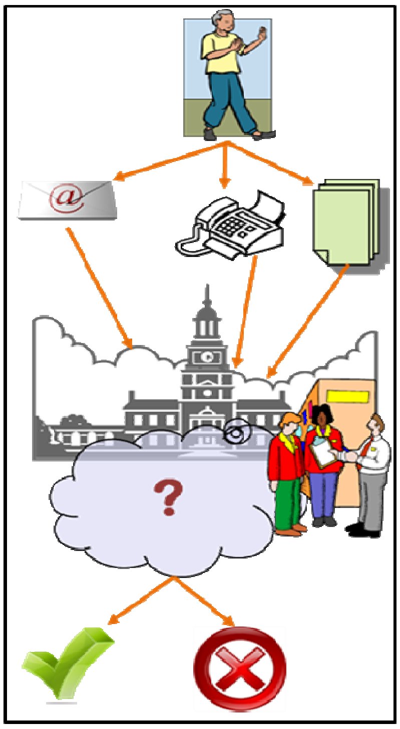


Figure 2: Registration of a new business without ICT support

As we can see, the whole process turns out to be very inefficient. The goal of the PA is **to re-engineer it**, supported by proper ICT tools. Barbara is the modeller who is supposed to perform this task

We will show now how the Template Generator can help Barbara in the initial phase of the re-design of the process, by generating a set of possible process schemas, which will support Barbara when working with the SOA4All Process Editor.

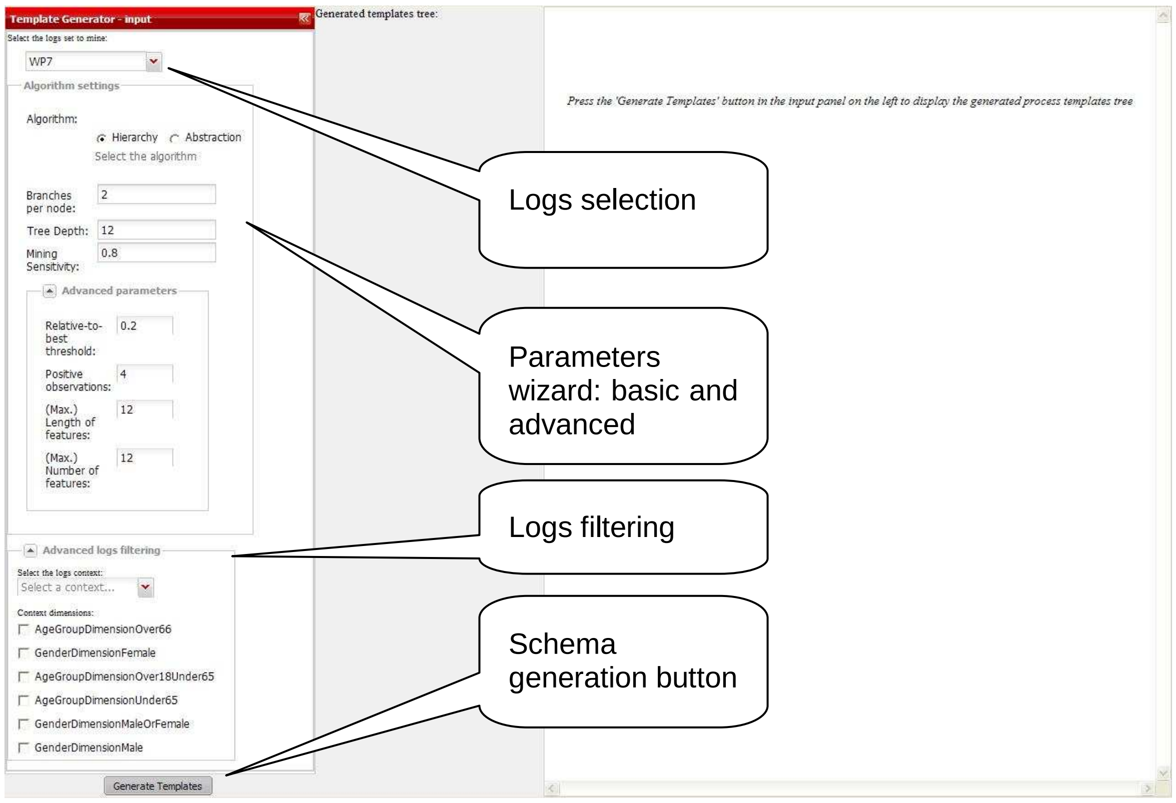
In fact, a complete manual process design can imply high effort, especially in situations like the one described in this example, when the process structure is not so clearly evident.

On the other hand, Process Mining techniques can help but are not sufficient, as they require a too specialist knowledge: Barbara needs support to identify simplification and rationalisation areas. The Template Generator allows to automatically build and identify process schemas at different complexity versus completeness leveles, and it allows to select and to create process templates, which can be then used in the SOA4All Process Editor

The benefits for Barbara are a: quicker and easier initial process design (less effort, better quality)

Step #1:

From the SOA4All Studio Process Editor, Barbara can launch the template editor, select the input logs she needs, and run the self-generation of schema.


Template Generator - input

Select the logs set to mine: WP7

Algorithm settings

Algorithm: Hierarchy [✓] Abstraction
Select the algorithm

Branches per node: 2
Tree Depth: 12
Mining Sensitivity: 0.8

Advanced parameters

Relative-to-best threshold: 0.2
Positive observations: 4
(Max.) Length of features: 12
(Max.) Number of features: 12

Advanced logs filtering

Select the logs context: Select a context...

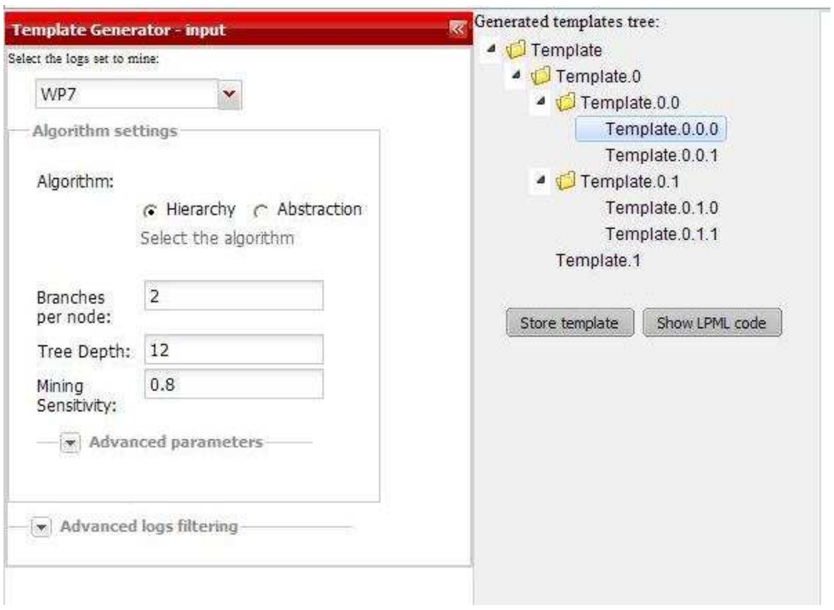
Context dimensions:
- [ ] AgeGroupDimensionOver66
- [ ] GenderDimensionFemale
- [ ] AgeGroupDimensionOver18Under65
- [ ] AgeGroupDimensionUnder65
- [ ] GenderDimensionMaleOrFemale
- [ ] GenderDimensionMale

Generated templates tree:

Press the 'Generate Templates' button in the input panel on the left to display the generated process templates tree

Figure 3: The Template Generator start-up screenshot

As a result, she obtains the following structure:


Template Generator - input

Select the logs set to mine: WP7

Algorithm settings

Algorithm: Hierarchy [✓] Abstraction
Select the algorithm

Branches per node: 2
Tree Depth: 12
Mining Sensitivity: 0.8

Advanced parameters

Advanced logs filtering

Generated templates tree:

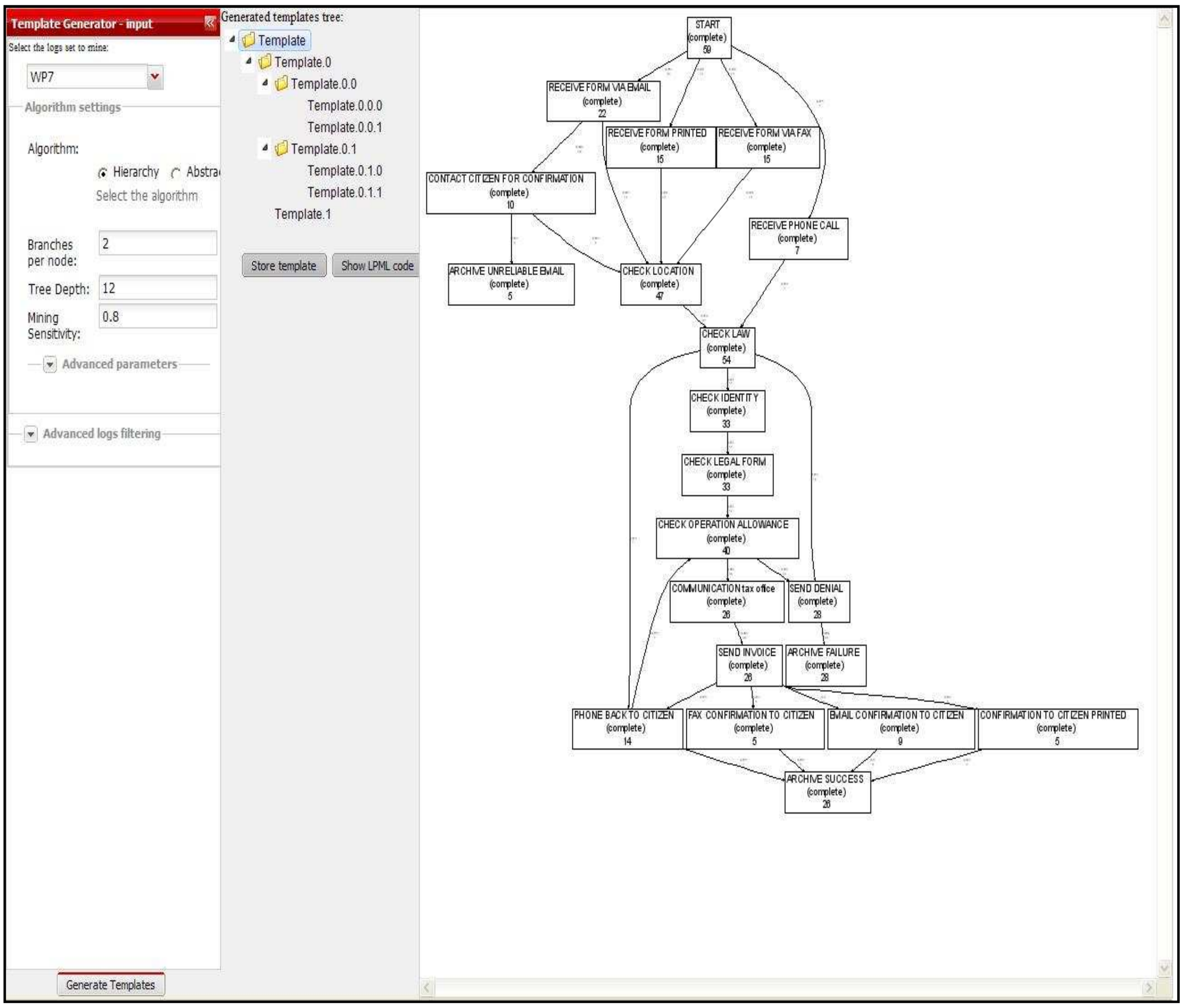
```
Template
  Template.0
    Template.0.0
      Template.0.0.0
      Template.0.0.1
    Template.0.1
      Template.0.1.0
      Template.0.1.1
  Template.1
```

The root of the tree represents the most complete schema, while leaves are specific sub-cases.

The specific structure obtained can be changed by changing some tool parameters: the initial parameters configuration is optimised to get the best compromised between simplicity and completeness of the tree structure.

Figure 4: The templates hierarchy obtained

Step #2: By clicking on ”Template” Barbara can visualise the process schema associated to the root of the whole hierarchy, which represents the most complete but also the most complex schema, including all possible branches and activities. Such schema is represented in the following picture:


Template Generator - input

Select the logs set to mine: WP7

Algorithm settings

Algorithm: [✓ Hierarchy] Abstra

Select the algorithm

Branches per node: 2

Tree Depth: 12

Mining Sensitivity: 0.8

Generated templates tree:

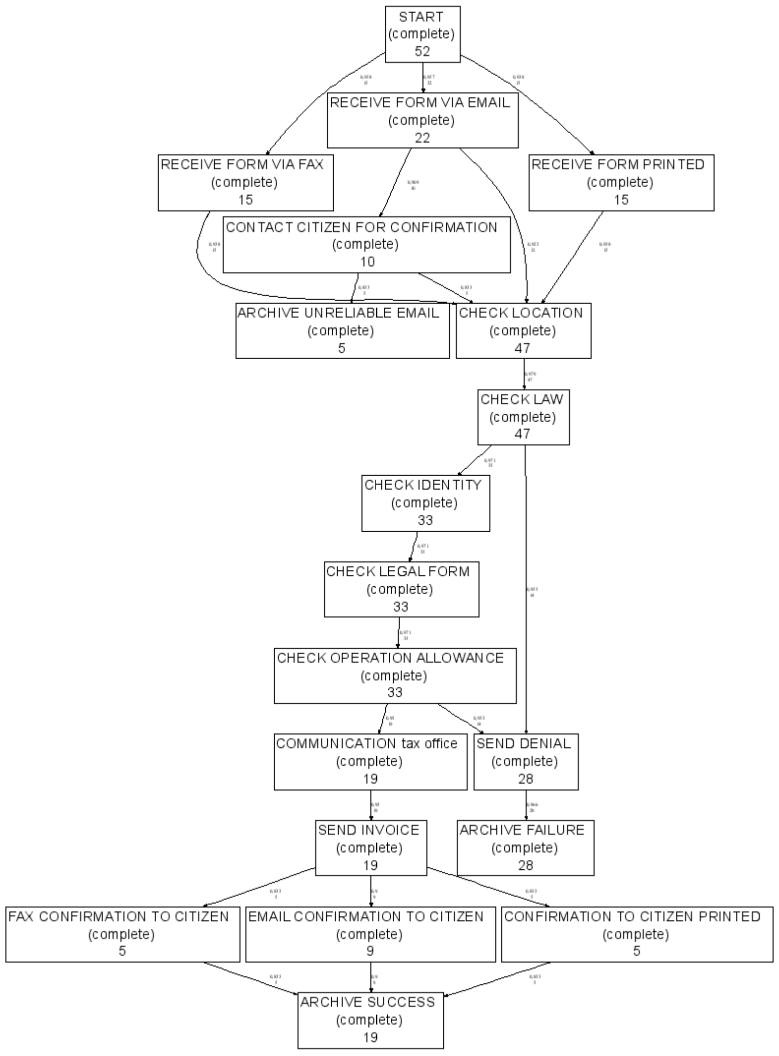
- Template
  - Template.0
    - Template.0.0
      - Template.0.0.0
      - Template.0.0.1
    - Template.0.1
      - Template.0.1.0
      - Template.0.1.1
  - Template.1

Advanced parameters
Advanced logs filtering
Generate Templates
Store template
Show LPML code
START (complete) 59
RECEIVE FORM VIA EMAIL (complete) 22
RECEIVE FORM PRINTED (complete) 15
RECEIVE FORM VIA FAX (complete) 15
CONTACT CITIZEN FOR CONFIRMATION (complete) 10
RECEIVE PHONE CALL (complete) 7
ARCHIVE UNRELIABLE EMAIL (complete) 5
CHECK LOCATION (complete) 47
CHECK LAW (complete) 54
CHECK IDENTITY (complete) 33
CHECK LEGAL FORM (complete) 33
CHECK OPERATION ALLOWANCE (complete) 40
COMMUNICATION tax office (complete) 26
SEND DENIAL (complete) 28
SEND INVOICE (complete) 26
ARCHIVE FAILURE (complete) 28
PHONE BACK TO CITIZEN (complete) 14
FAX CONFIRMATION TO CITIZEN (complete) 5
EMAIL CONFIRMATION TO CITIZEN (complete) 9
CONFIRMATION TO CITIZEN PRINTED (complete) 5
ARCHIVE SUCCESS (complete) 26

Figure 5: Root schema

We can observe how this schema includes all the possible cases and sub-cases described in the scenario (requests from phone, email, etc...). Barbara is not satisfied with this schema, as it looks un-necessarily complex – in fact, some of the situations here represented should not be considered after the re-engineering phase, like phone handling of requests.

Step #3: Barbara decides to look at the first level of the tree. Schema “Template.0” (left picture) shows a schema where the phone call cases are not taken into account – in fact such cases are represented into “Template.1” (right picture):


START (complete) 52
RECEIVE FORM VIA EMAIL (complete) 22
RECEIVE FORM VIA FAX (complete) 15
RECEIVE FORM PRINTED (complete) 15
CONTACT CITIZEN FOR CONFIRMATION (complete) 10
ARCHIVE UNRELIABLE EMAIL (complete) 5
CHECK LOCATION (complete) 47
CHECK LAW (complete) 47
CHECK IDENTITY (complete) 33
CHECK LEGAL FORM (complete) 33
CHECK OPERATION ALLOWANCE (complete) 33
COMMUNICATION tax office (complete) 19
SEND DENIAL (complete) 28
SEND INVOICE (complete) 19
ARCHIVE FAILURE (complete) 28
FAX CONFIRMATION TO CITIZEN (complete) 5
EMAIL CONFIRMATION TO CITIZEN (complete) 9
CONFIRMATION TO CITIZEN PRINTED (complete) 5
ARCHIVE SUCCESS (complete) 19

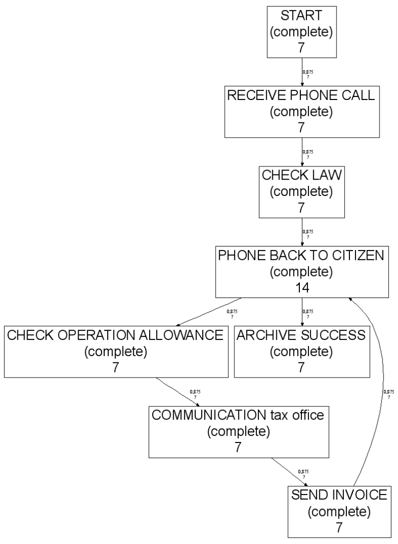

START (complete) 7
RECEIVE PHONE CALL (complete) 7
CHECK LAW (complete) 7
PHONE BACK TO CITIZEN (complete) 14
CHECK OPERATION ALLOWANCE (complete) 7
ARCHIVE SUCCESS (complete) 7
COMMUNICATION tax office (complete) 7
SEND INVOICE (complete) 7

Figure 6: Schemas “Template.0” (left) and “Template.1” (right)

Step #4: Phone calls are considered as exceptional cases, so they should not be taken into account in the future re-engineered process, and so she decides to drop such cases from the schemas. Anyway, schema “Template.0” still appears too complex, as it includes also the cases of email request, including the steps required to verify emails sender. She decides to explore “Template.0.0” (left) and “Template.0.1” (right).

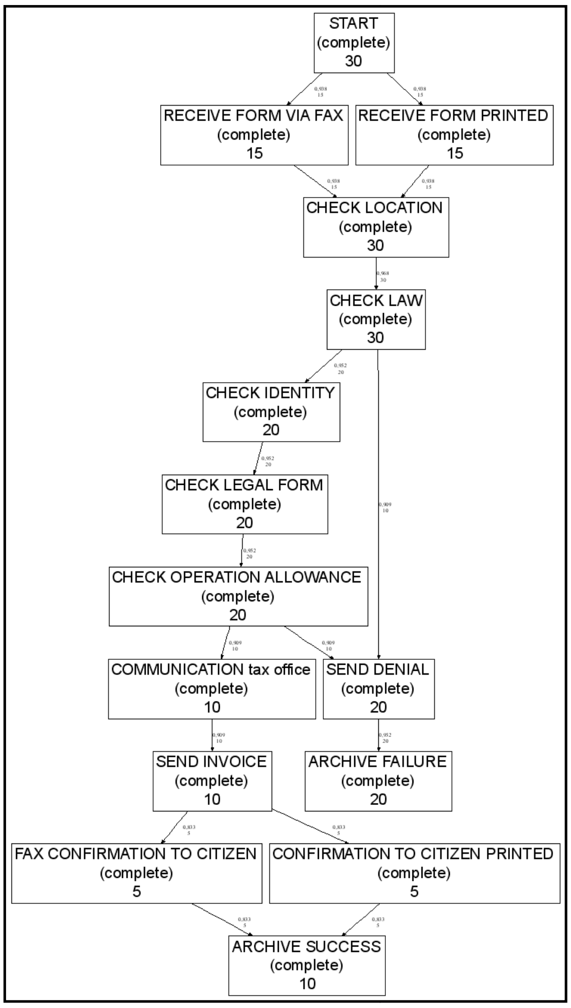
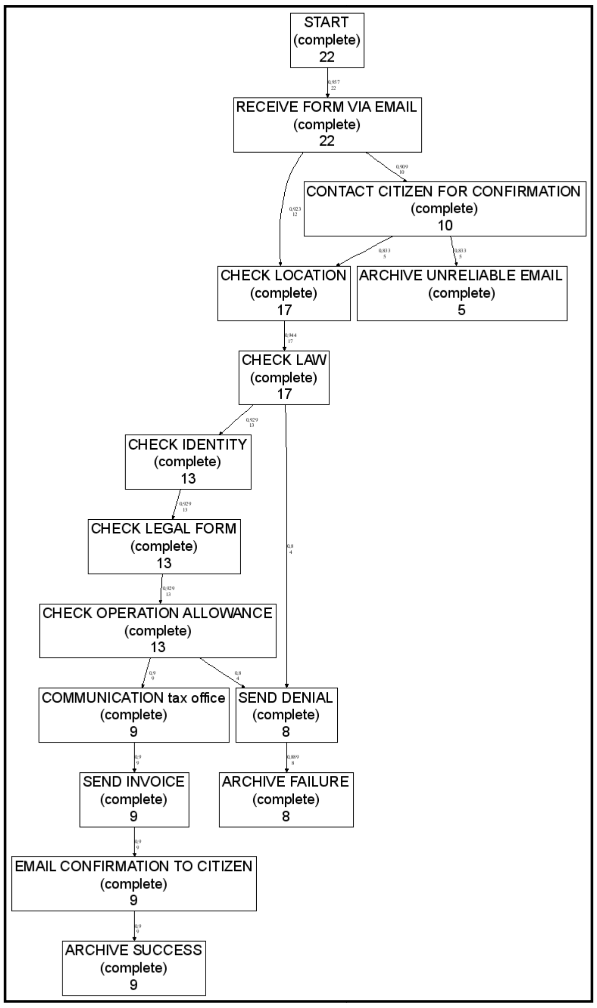
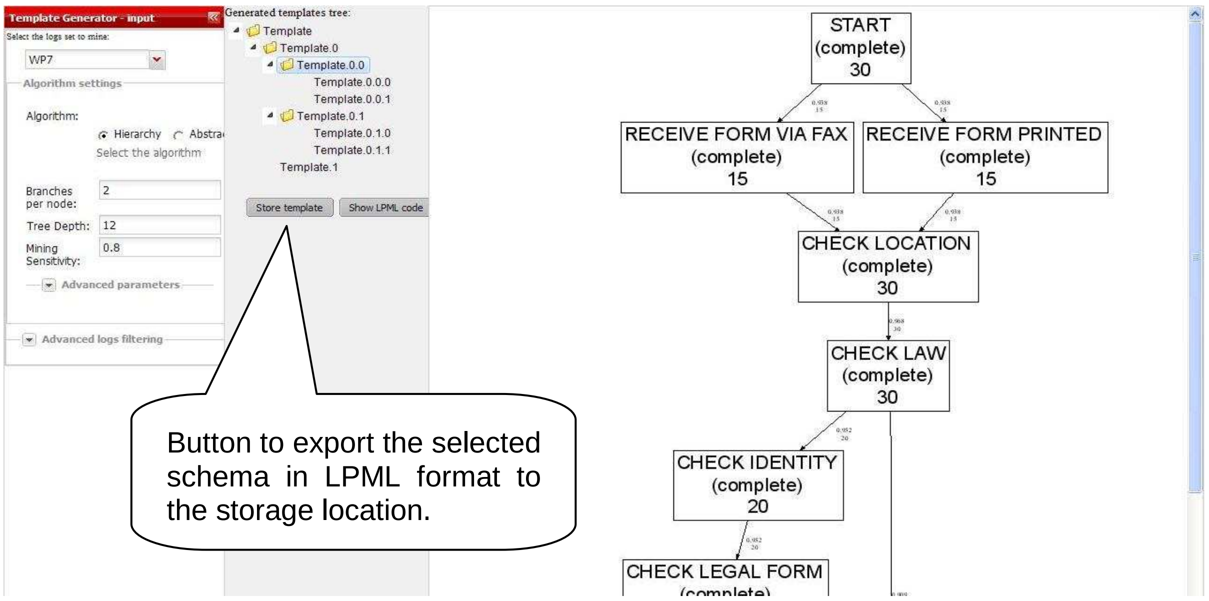

START (complete) 30
RECEIVE FORM VIA FAX (complete) 15
RECEIVE FORM PRINTED (complete) 15
CHECK LOCATION (complete) 30
CHECK LAW (complete) 30
CHECK IDENTITY (complete) 20
CHECK LEGAL FORM (complete) 20
CHECK OPERATION ALLOWANCE (complete) 20
COMMUNICATION tax office (complete) 10
SEND DENIAL (complete) 20
SEND INVOICE (complete) 10
ARCHIVE FAILURE (complete) 20
FAX CONFIRMATION TO CITIZEN (complete) 5
CONFIRMATION TO CITIZEN PRINTED (complete) 5
ARCHIVE SUCCESS (complete) 10

START (complete) 22
RECEIVE FORM VIA EMAIL (complete) 22
CONTACT CITIZEN FOR CONFIRMATION (complete) 10
CHECK LOCATION (complete) 17
ARCHIVE UNRELIABLE EMAIL (complete) 5
CHECK LAW (complete) 17
CHECK IDENTITY (complete) 13
CHECK LEGAL FORM (complete) 13
CHECK OPERATION ALLOWANCE (complete) 13
COMMUNICATION tax office (complete) 9
SEND DENIAL (complete) 8
SEND INVOICE (complete) 9
ARCHIVE FAILURE (complete) 8
EMAIL CONFIRMATION TO CITIZEN (complete) 9
ARCHIVE SUCCESS (complete) 9

Figure 7: Schemas "Template.0.0" (left) and "Template.0.1" (right)

Step #5: Email-based requests are now separated from the other procedures (fax and printed forms). She realizes that schema "Template.0.0" is a good candidate, as it does not include exceptional behaviors that will not be replicated in the future process. She continues digging into "Template.0.0" and she gets to the leaves of the hierarchy tree: schemas "Template.0.0.0" (left) and "Template.0.0.1" (right) represent too specific cases so **she finally selects "Template.0.0"**.


Template Generator - input

Select the logs set to mine: WP7

Algorithm settings

Algorithm: [✓ Hierarchy] Abstra

Select the algorithm

Branches per node: 2

Tree Depth: 12

Mining Sensitivity: 0.8

Generated templates tree:

- Template
  - Template.0
    - Template.0.0
      - Template.0.0.0
      - Template.0.0.1
    - Template.0.1
      - Template.0.1.0
      - Template.0.1.1
  - Template.1

Advanced parameters
Advanced logs filtering
Store template
Show LPML code
Button to export the selected schema in LPML format to the storage location.
START (complete) 30
RECEIVE FORM VIA FAX (complete) 15
RECEIVE FORM PRINTED (complete) 15
CHECK LOCATION (complete) 30
CHECK LAW (complete) 30
CHECK IDENTITY (complete) 20
CHECK LEGAL FORM (complete)

Figure 8: export of the selected “Template.0.0” to the storage location in LPML format

Step #6: Barbara can now save this schema to the Process Templates repository based on the 2.4 storage system (see Figure 8), which is also used by the process editor allowing a seamless integration. As such, the schema can be further re-called into the Process Editor to define it at a more specific **level, as explained in next sections.**

2.3 Process Modelling

Previous section has explained the techniques SOA4All offers to populate the process template storage with process fragments that are extracted by process mining techniques from past process execution logs. This population process can be performed within any time frame by any organization department in order to increase its business process knowledge base.

The same or a different department further reuses this knowledge base in ongoing or future modelling projects. That is, they intensively reuse the knowledge base of process fragments and templates in new modelling projects within the same or similar application domain.

In this and next section, we describe the features offered by the SOA4All tooling to semi-automate the context-aware adaptive modelling of new business process, at design time, by reusing existing domain knowledge, such as the aforementioned process template storage, but also other domain specific knowledge: domain models (ontologies), contextual information, service repositories, etc. The integrated service construction suite (i.e. Process Editor, Design Time Composer, Optimizer) supports this semi-automated modelling phase. Process Editor [8] is used to collect, from the modellers (such as Barbara), the process model information that DTC and Optimizer need to automate some modelling tasks.

This section describes the automated features provided by final prototype of DTC while next section describes the final prototype of Optimizer.

DTC provides context-aware adaptive automation of common modelling tasks, using knowledge intensive techniques as describe in [4] and [5]:

- It binds activities to SWSs whose semantic descriptions match the activity annotations.
- It expands activities with process fragments whose semantic descriptions match the activity annotations.
- It creates the data-flow connectors that wire the data input of some activities with the data output of predecessor activities.
- It checks the semantic compatibility of the I/O of the process activities and their bound SWSs according to the data flow, filtering incompatible bindings.
- It manages contextual information that adapts the changes introduced by DTC in the process model.

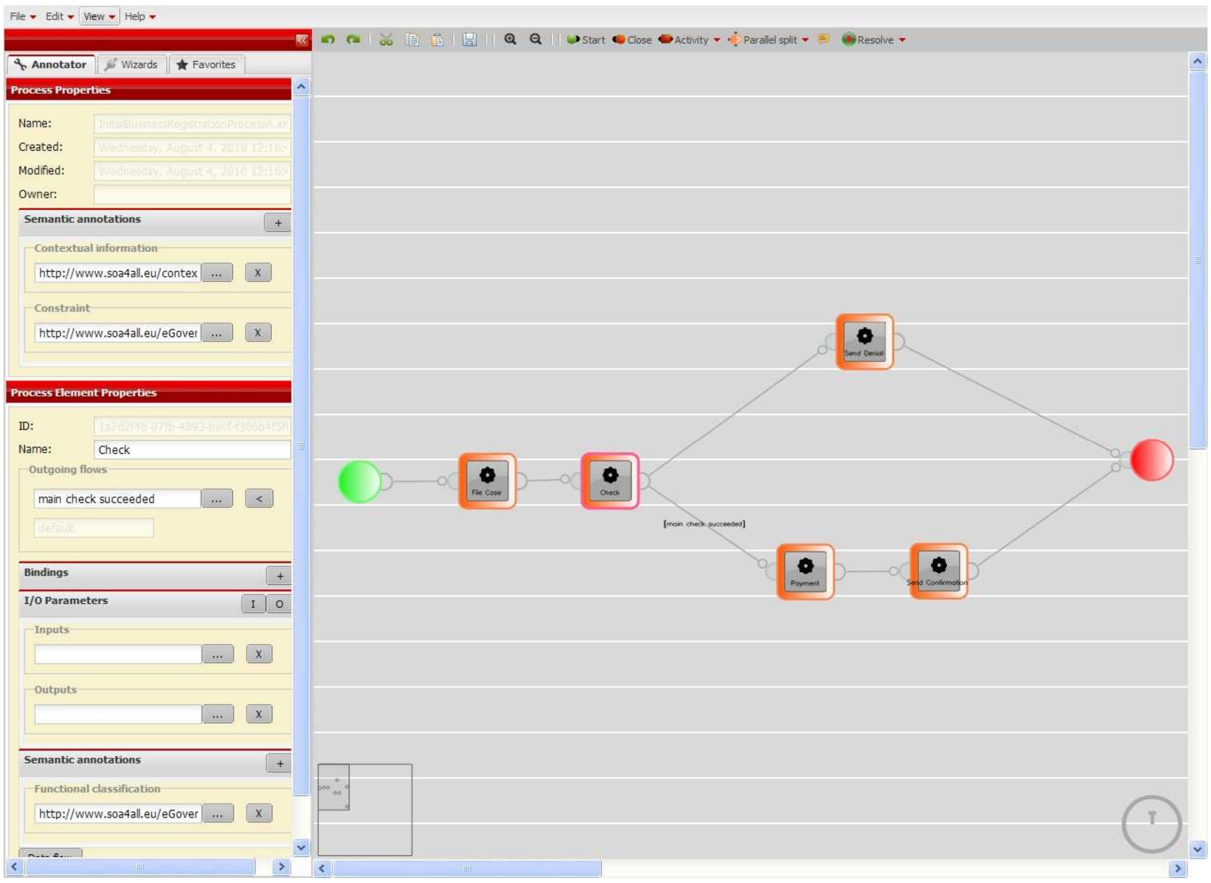
DTC exploits the semantic annotations introduced by the modeller in the process model. Annotations may take the following forms:

- Local annotations, which describe concrete modelling elements such as activities, gateways and flows: FC, I/O, NFP (requirements, preferences).
- Global annotations, which describe the whole process model: contextual information requirements and preferences.

These annotations are analysed by DTC during composition phase in order to ensure that the overall process is consistent. If at some stage of composition local annotations are incompatible with global user requirements, such process model will be rejected.

In the rest of this section, we continue with the WP7 scenario we use as illustrative modelling example throughout this document.

DTC assists a modeller (such as Barbara or another) to create a process model for the WP7 business registration process [10]. A common process-modelling project does not start, if possible, from a blank process model, but tries to reuse some common domain existing knowledge. In this example, the modeller starts from one of the process fragments (templates) extracted by the TG and stored into the process template storage, as explained in the previous section. Initial draft process models can be located from the template storage using two methods. On the one hand, the modeller can browse the template repository by hand using the Process Editor and load candidate templates by name. On the other hand, the modeller can create a very simple process with only one activity (described by single annotations) that describes the desire draft model, and ask DTC to look for the best process fragment that suits the single activity. Regardless the approach, the modeller gets a process fragment that uses as the initial draft model. In our example, the modeller browses the storage and selects the process fragment for business registration process created by TG[2] as describe in previous section, as shown in next figure.



Process Properties

Name: InitialBusinessRegistrationProcessA.xr
Created: Wednesday, August 4, 2010 12:16:
Modified: Wednesday, August 4, 2010 12:16:
Owner:

Semantic annotations

Contextual information: http://www.soa4all.eu/contex

Constraint: http://www.soa4all.eu/eGover

Process Element Properties

ID: 1a7d2f46-97fb-4893-ba0f-f396b4f5f
Name: Check

Outgoing flows: main check succeeded

default

Bindings

I/O Parameters

Inputs:

Outputs:

Semantic annotations

Functional classification: http://www.soa4all.eu/eGover

Figure 9 Initial process fragment as obtained from Template Generator

In most of the cases, starting from a template model will require some manual amendments, since the process template is intended for a wide range of usages. Therefore, the modeller needs to adapt manually the process fragment to its actual usage, which may imply to adapt partially the work-flow and likely some domain specific local (activity) and global (process) annotations. Using the PE the modeller has modified the draft model: check activity is split into two ("Preliminary Check" and "Check"), annotations have been added, I/O parameter annotations modified.

When resolving an activity or the whole process, DTC considers not only the local activity annotations, but also the global annotations added by the modeller to the process. Global annotations can be requirements and preferences[3], as described in [4] and [5]. In our

[2] Note that the modeller can edit and modify process templates generated by the TG, before being stored within the template repository. In this case, the process template generated in the previous section and the one load here differ since the modeller simplified the former one in order to be more generic.

[3] In this document we refer to requirements and preferences in order to follow the common

example, the modeller have manually added some annotations: a requirement for using only cost-free SWSs, a preference on the notification channel and some contextual information[4] concerning the preferable payment method for the particular local government department that is modelling this process.

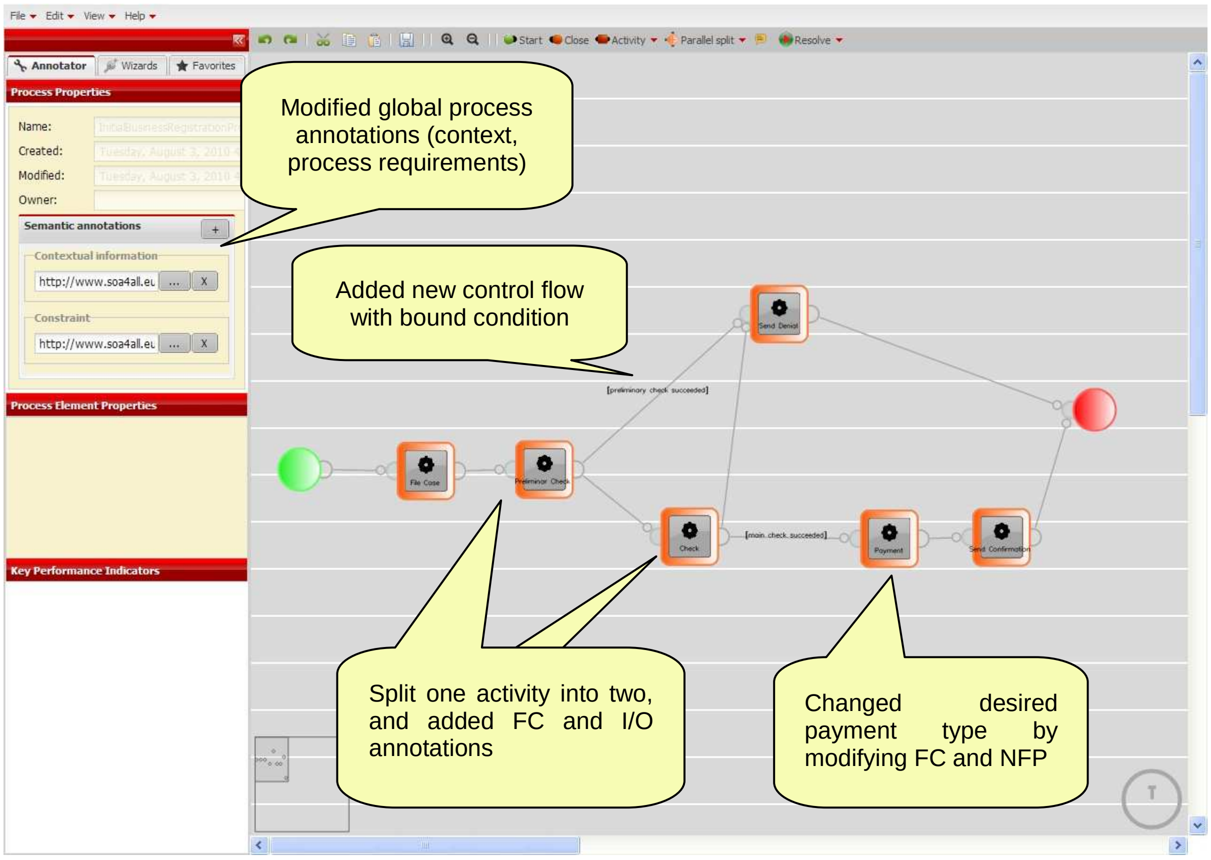

File Edit View Help
Start Close Activity Parallel split Resolve
Annotator Wizards Favorites
Process Properties
Name:
Created:
Modified:
Owner:
Semantic annotations
Contextual information
http://www.soa4all.eu
Constraint
http://www.soa4all.eu
Process Element Properties
Key Performance Indicators
Modified global process annotations (context, process requirements)
Added new control flow with bound condition
Split one activity into two, and added FC and I/O annotations
Changed desired payment type by modifying FC and NFP
File Case
Preliminary Check
Check
Send Denial
[preliminary check succeeded]
[main check succeeded]
Payment
Send Confirmation

Figure 10:Initial model after adjusted by the user (changes indicated in the yellow bubbles)

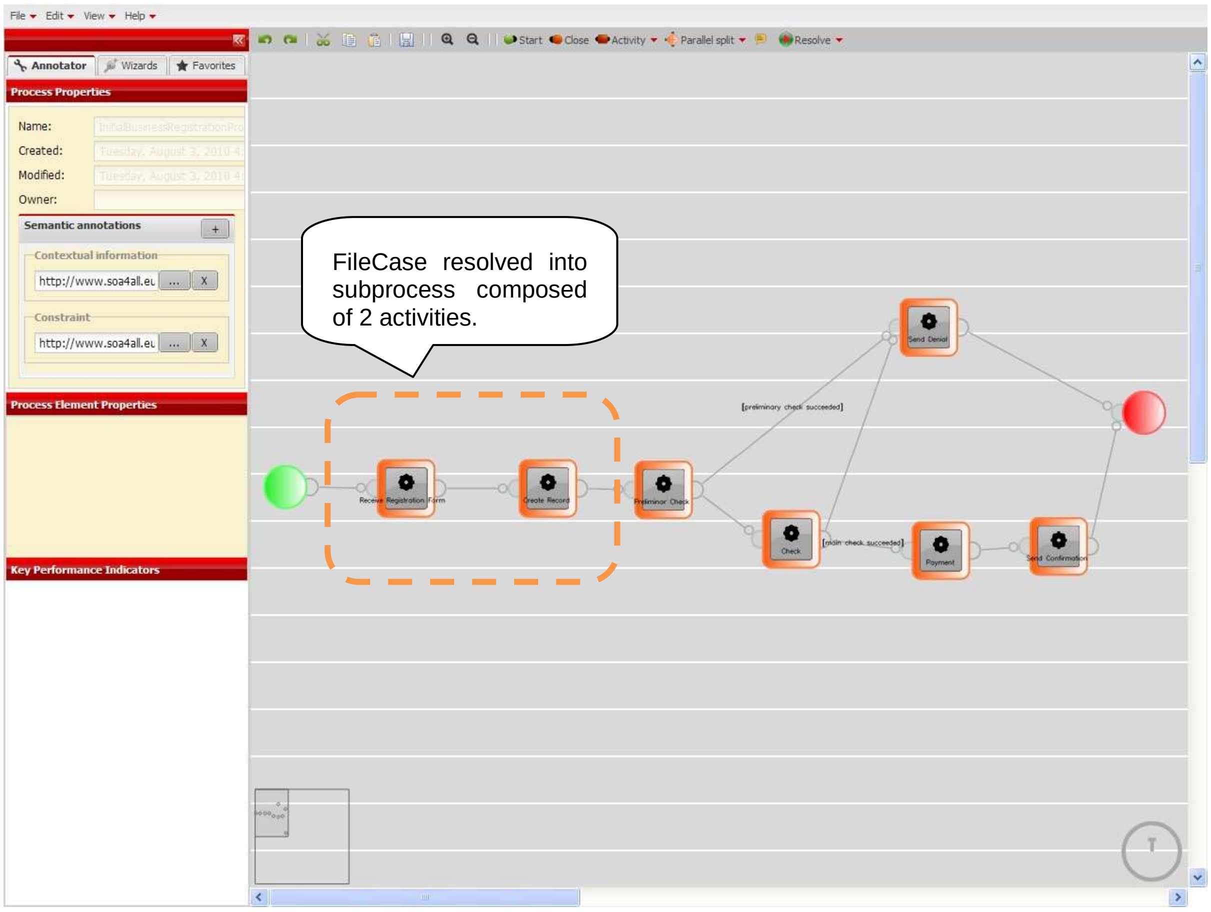
Once the initial draft model is ready, the modeller can start trying to bind some abstract activities (described by annotations, but already not bound to concrete services). She can do it manually (using PE features) or relying on the **DTC.bindActivity** method. In our example, the modeller selects the FileCase activity and invokes DTC.bindActivity menu in PE. However, DTC does not return a solution since it has not been possible to find any matching SWS based on the available knowledge. In other words, there were not any SWS whose

naming convention used in Service Discovery [12]. Formerly in [5], requirements and preferences were denoted as constrains and requirements, following the Parametric Design specification.

[4]Current PE prototype does not support the automatic insertion of contextual information obtained from contextual sources such as the user profile, whereby the modeler introduces contextual annotations by hand.

MSM description fit into the activity annotations. In this case, the FileCase activity was too complex to be resolved by only one existing SWS operations.

An alternative is to ask DTC to resolve the activity by selecting the **DTC.resolveActivity** method in the PE menu. In this case, DTC tries not only to bind the activity to a matching service, but also to expand the activity with process fragments or templates matched against the activity annotations by inspecting their existing semantic descriptions stored in the Semantic Space (SS) [11]. The returned model, as shown in next figure, is more complete, since the matched template has replaced the FileCase activity. As commented before, any time a generic template is used as part of a process, the customization of the annotations of that fragment of the process is possible to adapt its generic usage to the particular one.



Name: InitialBusinessRegistrationPro
Created: Tuesday, August 3, 2010 4
Modified: Tuesday, August 3, 2010 4
Owner:
Contextual information: http://www.soa4all.eu
Constraint: http://www.soa4all.eu

Figure 11: Process model after “FileCase” activity resolved

To save time the modeller invokes the **DTC.resolveProcess** method in the PE menu. In this case, DTC applies any existing domain knowledge to resolve any missing information in the whole process model, including unbound activities, which are resolved by template expansion or SWS binding (depending of found matches). As a result, DTC either expands with matching templates or binds with SWS candidates most of the activities. Note that LPML allows multiple bindings. However, there are still few unbound activities (Receive registration form, Send Confirmation).

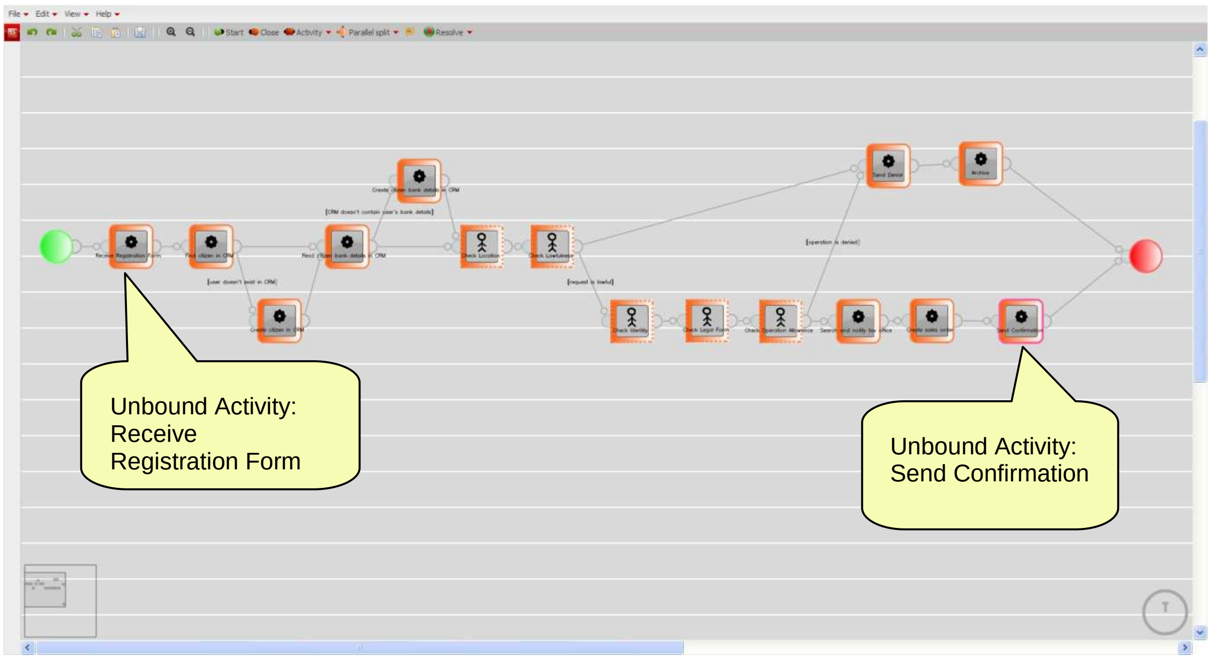

File Edit View Help
Start Close Activity Parallel split Resolve
Unbound Activity: Receive Registration Form
Unbound Activity: Send Confirmation

Figure 12: Process model after invoking "Resolve Process" option

Since some activities have not been bound, the modeller reconsiders their annotations and for each tries to bind them again invoking **DTC.bindActivity** menu. The final bound process model is shown below.

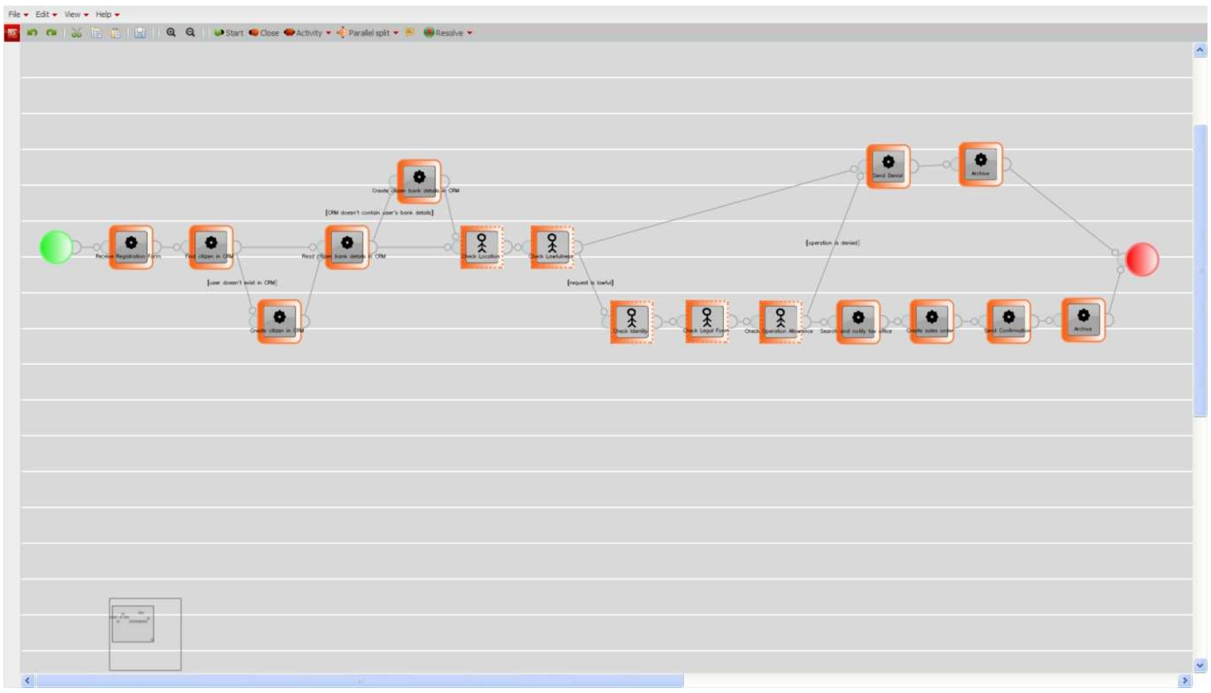


Figure 13: Completely resolved process model

Next steps concerns the data flow generation. Once the process model is complete (i.e. all the activities are bound), DTC is able to partially generate the data flow. The modeller invokes. **DTC.generateDataFlow** method in the PE menu and DTC returns a modified model with a tentative data flow generated. For each activity, the modeller should check and amend (if needed) the generated data flow using the PE data flow editor. Together with the dataflow generation, DTC check the I/O semantic compatibility of the whole set of binding (for all activities), filtering (removing) those bindings which are incompatible with any possible data flow. Note that DTC performs both tasks when the modeller invokes **DTC.resolveProcess**, as soon as DTC finds a complete process model and before returning the final process model.

Next pictures show the complete generated process model with a tentative data flow added. Note that, by the time of writing, Process Editor cannot render LPML models that include a data flow since that feature is under development. Therefore, we show all the LPML models that contains a data flow and are generated in this phase and in the next one (section 2.4) using an ad-hoc LPML Visualizer, developed since M18 [5] for tracking and debugging purposes. Once the Process Editor M30 version is release, LPML models containing a data flow will be supported and their data flow editable. .

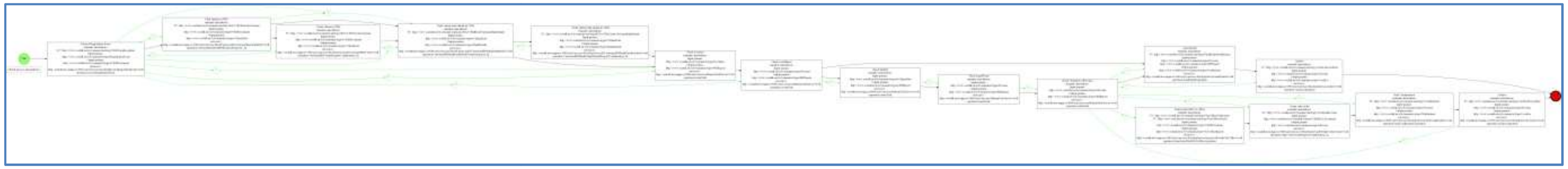
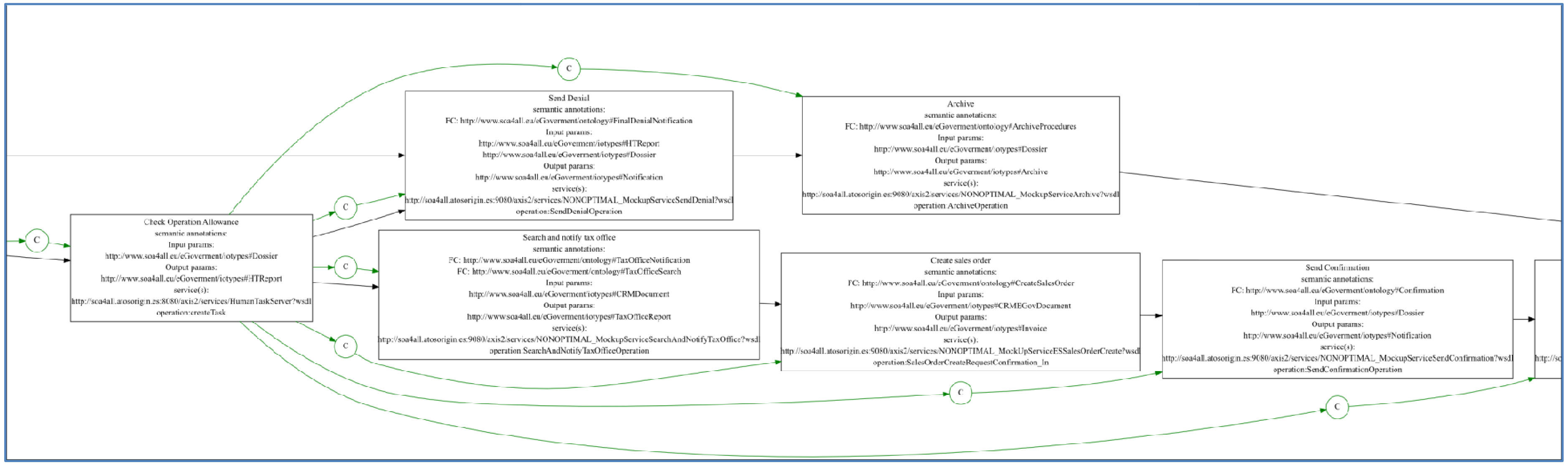


Figure 14: Process model with dataflow connectors in green (shown in LPML viewer)


C
Send Denial
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#FinalDenialNotification
Input params:
http://www.soa4all.eu/eGoverment/iotypes#HTReport
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Notification
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceSendDenial?wsdl
operation:SendDenialOperation
Archive
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#ArchiveProcedures
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Archive
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceArchive?wsdl
operation:ArchiveOperation
Check Operation Allowance
semantic annotations:
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#HTReport
service(s):
http://soa4all.atosorigin.es:8080/axis2/services/HumanTaskServer?wsdl
operation:createTask
Search and notify tax office
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeNotification
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeSearch
Input params:
http://www.soa4all.eu/eGoverment/iotypes#CRMDocument
Output params:
http://www.soa4all.eu/eGoverment/iotypes#TaxOfficeReport
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceSearchAndNotifyTaxOffice?wsdl
operation:SearchAndNotifyTaxOfficeOperation
Create sales order
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#CreateSalesOrder
Input params:
http://www.soa4all.eu/eGoverment/iotypes#CRMEGovDocument
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Invoice
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockUpServiceESSalesOrderCreate?wsdl
operation:SalesOrderCreateRequestConfirmation_In
Send Confirmation
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#Confirmation
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Notification
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceSendConfirmation?wsdl
operation:SendConfirmationOperation
http://so
Figure 15: Process model with dataflow connectors in green (detail)

Eventually, the modeller can invoke **DTC.checkIOSemanticCompatibility** in PE menu to check and filter incompatible bindings over complete models. This operation can be performed any time the modeller manually add a new binding to check whether it is or not compatible with the current binding set of affected activities.

The result of this design time semi-assisted modelling task is a complete, non-optimized process model for the business registration scenario. Next section describes how the modeller uses the Optimizer method integrated in the PE menu to optimize the model.

2.4 Process Optimization

In this section, we illustrate and describe the final prototype of Optimizer platform service component along the WP7 Business Registration Process scenario described in [10]. The theoretical grounding of the optimizer component has been described in [4] and [5].

The modelling process is a combination of manual modelling (using the Process Editor,) and assisted modelling using the Optimizer service. This test explains an illustrative jointly process modelling phase in which modeller (user) using the Process Editor, the result of the DTC service and the optimizer service are mutually collaborating.

The optimizer component provides non-functional- and semantic-based optimization common modelling tasks, using semantic reasoning as describe in [4] and [5]:

- It binds activities to the most appropriate SWSs whose semantic descriptions match the activity annotations, and non-functional parameters are optimal for activities.
- It optimizes the quality if data-flow connections that wire the data input of some activities with the data output of predecessor activities.

In the rest of this section, we continue with the WP7 scenario we use as illustrative modelling example throughout this document.

The modeller uses the Optimizer to optimize an existing process model for the WP7 business registration process [10]. In this example, the modeller starts from one of (pre-composed) process model inferred from successively the process fragments (templates) extracted by the TG and DTC, as explained in the previous sections. However, the optimizer can be applied on any process, without asking the intervention of DTC. To this end, pre-composed process models are located from the process storage using two methods.

On the one hand, the modeller can browse the process repository by hand using the Process Editor and load processes by name. On the other hand, the modeller can create a very simple process with different activities (described by single annotations) and that describes the desire composition model. Finally, the modeller can ask the optimizer to look for the best service binding regarding the activities descriptions involved in the process. Regardless the approach, the modeller gets a complete process that uses as the input of the optimizer. In our example, the modeller do not need to browse the storage and selects the process, but simply need to ask for optimization of the process designed by DTC. The optimization is then achieved through the Process Editor, which provides the possibility to optimize the previous service according to an “Optimize” button (right hand corner in Figure 12).

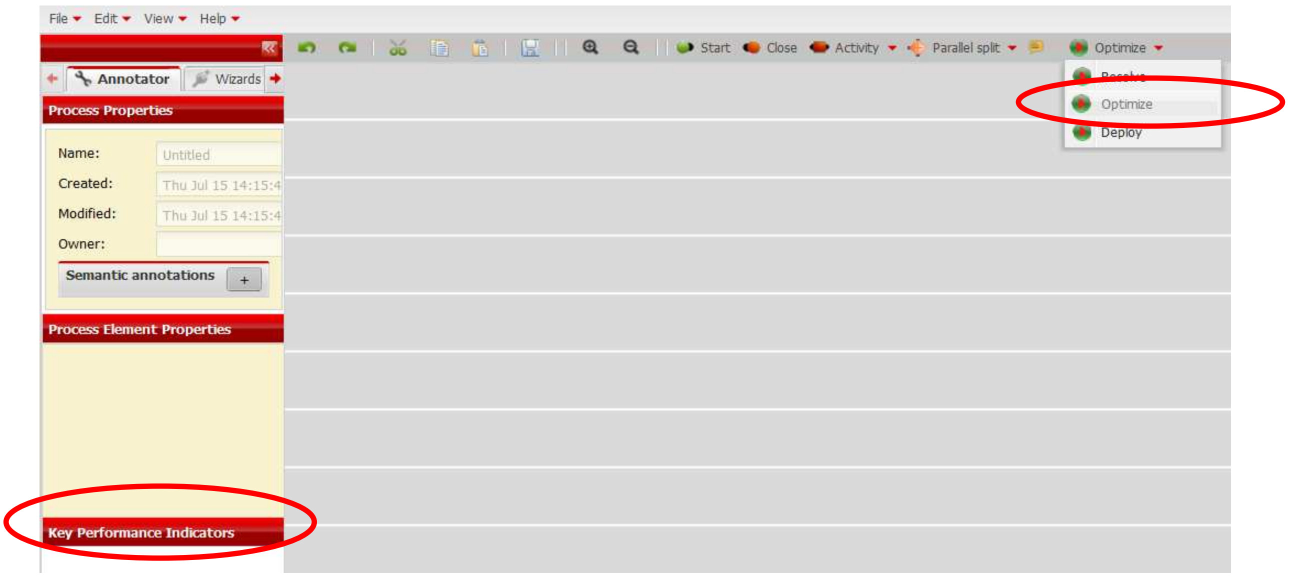


Figure 16: Optimizer Interaction with Process Editor

Before an Optimization invocation, the modeller is asked to provide the parameters she wants to optimize in the composition by means of “Optimize” button (left hand corner in Figure 6). For instance, the modeler could be interested in optimizing along (1) specific Key Performance Indicators KPIs: overall availability of the process (i.e., aggregation of availability of services), overall price, overall response time and (2) quality of semantic connections between services: overall matching quality, overall robustness.

During the optimization process, the new bound services are automatically discovered from the IServe repository (http://iserve.kmi.open.ac.uk/). Therefore, the modellers do not need to specify any service repository. However, in case the modeller want to attach a service not referred in the repository, she will need to first to describe the service in IServe and providing a valid endpoint to it.

Using the WP7 example, the process modeller asks for optimization of process model (generated by DTC) in Figure 8.

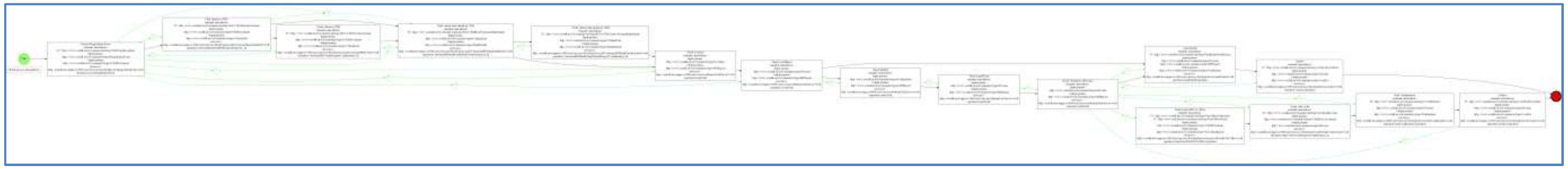


Figure 17:Non-Optimal Process

After the invocation of the optimizer, the optimal process is returned given the KPIs and semantic constraints provided by the modeller.

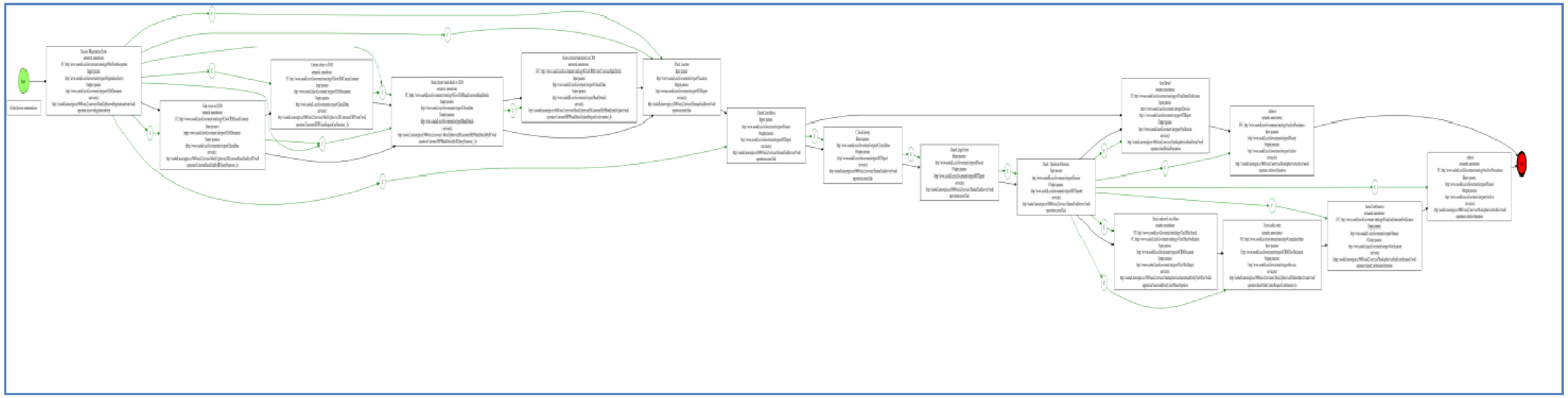


Figure 18: Optimal Process

While the input DTC is generally a rather goal-heavy process specification, the optimizer only accepts complete process models for which it seeks a better global cost function (in term of functional and non-functional qualities of services i.e., KPIs and quality of semantic connections between services in the composition). The optimizer transparently transforms compositions into their optimal versions by replacing service bindings and modifying the dataflow but without changing the workflow (i.e., its structure – there is no difference in control flow specification of non-optimal and optimal processes in respectively Figures 7 and 8). The only changes refer to services binding to activities as illustrated in Figures 9 and 10. This is justified by their impact in the overall quality of the process. Indeed the services bound to “Check Operation Allowance”, “Send Denial” and “Search and Notify Tax Office” activities in the non-optimal process ensure a quality of:

- KPI availability: 0.085
- KPI price: 5.23
- KPI response time: 12.5
- Semantic matching quality: 0.35
- Semantic robustness: 0.68

whereas the services bound to the previous activities in the optimal process ensure a quality of:

- KPI availability: 0.155
- KPI price: 5.03
- KPI response time: 4.5
- Semantic matching quality: 0.56
- Semantic robustness: 0.98

According to the latter figures, the optimal process has better quality than the non-optimal one. Obviously, relevant binding of services to activities is required to optimize processes.

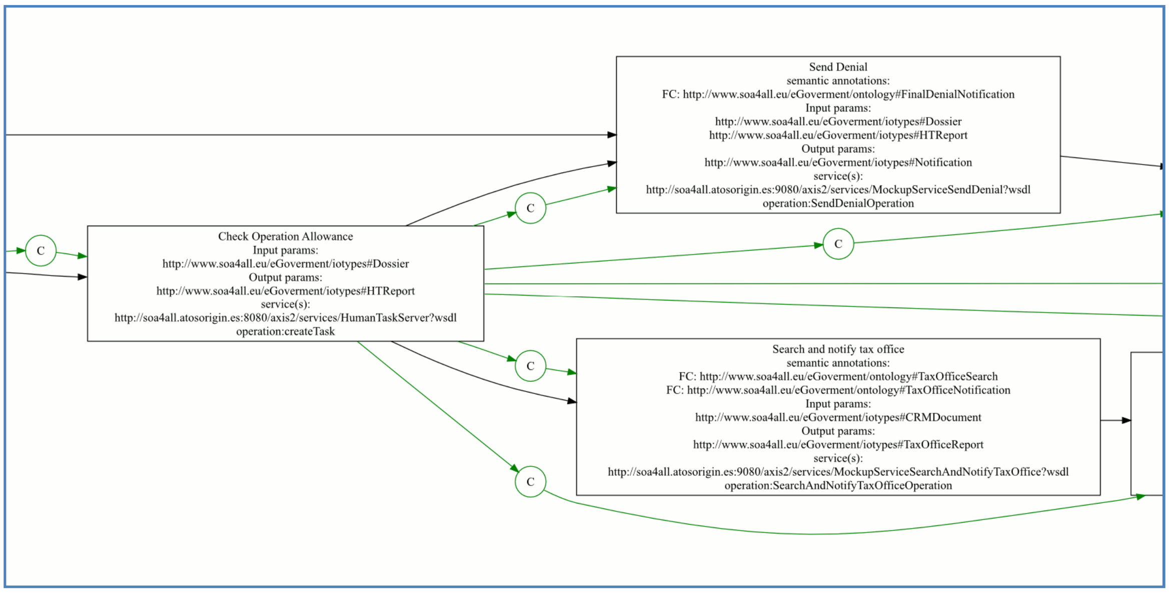

Send Denial
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#FinalDenialNotification
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
http://www.soa4all.eu/eGoverment/iotypes#HTReport
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Notification
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/MockupServiceSendDenial?wsdl
operation:SendDenialOperation
Check Operation Allowance
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#HTReport
service(s):
http://soa4all.atosorigin.es:8080/axis2/services/HumanTaskServer?wsdl
operation:createTask
Search and notify tax office
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeSearch
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeNotification
Input params:
http://www.soa4all.eu/eGoverment/iotypes#CRMDocument
Output params:
http://www.soa4all.eu/eGoverment/iotypes#TaxOfficeReport
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/MockupServiceSearchAndNotifyTaxOffice?wsdl
operation:SearchAndNotifyTaxOfficeOperation
C

Figure 19: Part of the Non-Optimal Composition

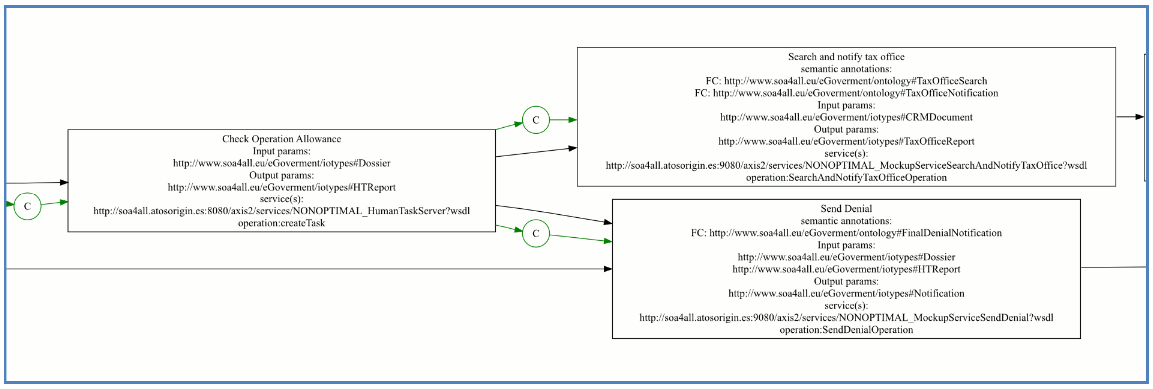

Search and notify tax office
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeSearch
FC: http://www.soa4all.eu/eGoverment/ontology#TaxOfficeNotification
Input params:
http://www.soa4all.eu/eGoverment/iotypes#CRMDocument
Output params:
http://www.soa4all.eu/eGoverment/iotypes#TaxOfficeReport
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceSearchAndNotifyTaxOffice?wsdl
operation:SearchAndNotifyTaxOfficeOperation
Check Operation Allowance
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
Output params:
http://www.soa4all.eu/eGoverment/iotypes#HTReport
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_HumanTaskServer?wsdl
operation:createTask
Send Denial
semantic annotations:
FC: http://www.soa4all.eu/eGoverment/ontology#FinalDenialNotification
Input params:
http://www.soa4all.eu/eGoverment/iotypes#Dossier
http://www.soa4all.eu/eGoverment/iotypes#HTReport
Output params:
http://www.soa4all.eu/eGoverment/iotypes#Notification
service(s):
http://soa4all.atosorigin.es:9080/axis2/services/NONOPTIMAL_MockupServiceSendDenial?wsdl
operation:SendDenialOperation
C

Figure 20: Part of the Optimal Composition

To conclude, the modeller creates a non-optimized model in the previous phase using the PE/DTC, then she expresses some NFP/KPI for optimization and invokes the Optimizer using the PE menu.

2.5 Process post-mortem schema analysis

A further goal that process modellers can achieve thanks to the Template Generator is to verify if the schema they modelled is valid and accurate, meaning if it is actually corresponding to the process executions: in fact, a process can be modelled including several branches which are then not used during run-time.

This goal can be quite easily achieved thanks to the Template Generator, as illustrated in the following picture:

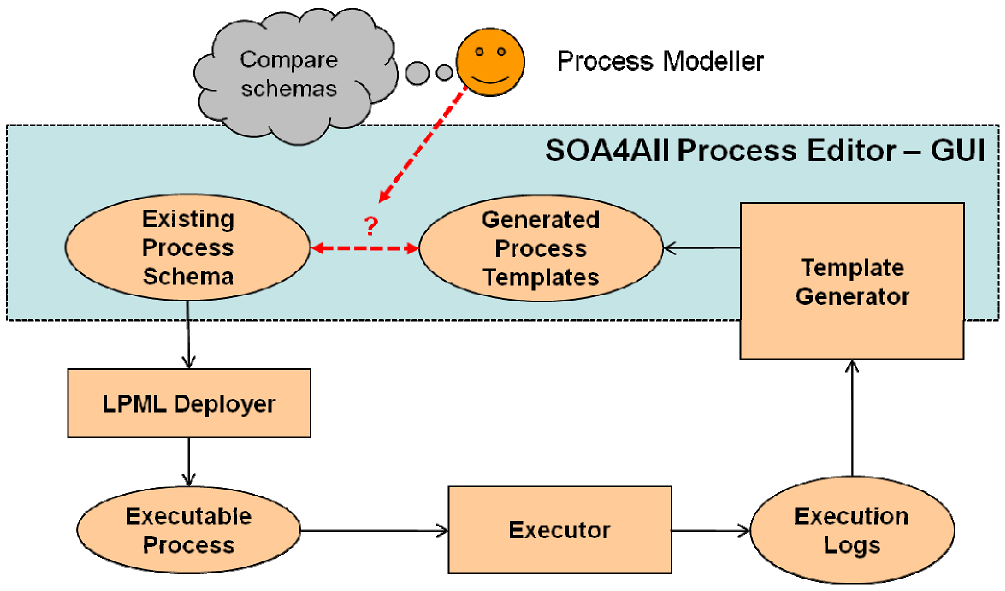


Figure 21: The Template Generator used to verify a process schema

An existing process is executed thanks to the SOA4All Executor.

The Template Generator captures execution logs and is able to generate an hierarchy of process schemas out of them. Such schemas will represent actual process executions.

Process modellers are able now to compare the schema that was deployed, with the schemas generated out of real executions, and check the consistency of such schemas. They can identify areas of improvement for their initial schema and change it accordingly in the Process Editor.

3. Conclusions

This document highlights the main functionality, as perceived by the process modeller, of the final prototype for Service Composition and Adaptation Environment, which provides semi-assisted support for some tedious and time-consuming modelling tasks at design time. The document introduces the tooling support at design time for the Lightweight Process Modelling Methodology and principles. Afterwards, it dig into the details of each design time phase and how it is supported by the different DTCE tools. We emphasize three main features: the usage of the Template Generator tool as a mean to populate a reusable repository of modelling templates obtained from the inspection of executed processes, the usage of the Design Time Composer tool to semi-assist some cumbersome process modelling tasks, and the usage of the Optimization tool to obtain full-optimized process models.

We do an illustrative and practical modelling exercise throughout the document as a conductor of the explanation of the main DTCE tooling features, in particular a model for the WP7 business registration process is completely created from scratch to its final optimized and ready for deployment version.

4. References

[1] SOA4All D6.3.1. Specification of Lightweight Context-aware Process Modelling Language, 2008.

[2] SOA4All D6.3.2. Advanced Specification Of Lightweight, Context-aware Process Modelling Language, 2009.

[3] SOA4All D6.3.3. Evaluation and Final Design Of Lightweight, Context-aware Process Modelling Language, 2010.

[4] SOA4All D6.4.1. Specification and First Prototype Of Service Composition and Adaptation Environment, 2009.

[5] SOA4All D6.4.2. Advance Prototype Of Service Composition and Adaptation Environment, 2010.

[6] SOA4All D6.5.2. Advanced Prototype For Adaptive Service Composition Execution, 2010.

[7] SOA4All D2.6.1 Specification of the SOA4All Process Editor, 2009

[8] SOA4All D2.6.2 SOA4All Process Editor First Prototype, 2009

[9] SOA4All D2.6.3 Advanced prototype of SOA4All Process Editor, 2009

[10] SOA4All D7.2 Scenario Definition, 2008

[11] SOA4All D1.3.3B Semantic Spaces: A Second Implementation, 2010

[12] SOA4All D5.3.2 Second Service Discovery Prototype

[13] Lécué, F. et Al. SOA4All: An Innovative Integrated Approach to Services Composition. Paper accepted at ICWS 2010

5. Technical Annex

This section summarizes the technical changes and improvements, mostly related to the integration with other platform services and API provided by other technical WPs. This section also refers the user to the source code and installation instructions of the DTCE tools:

Template Generator

Software: https://svn.sti2.at:/SOA4All/trunk/SOA4All-service-construction/SOA4All-serviceconstruction-module-templategenerator

Installation instructions:
The Template Generator code is part of the Process Editor, thus it is automatically installed with it. The TG has no particular configuration to be set.

Technical improvements:

- integration with logs repository
- algorithm parameters wizard
- full integration with latest Process Editor
- Improvements in GUI and better schemas navigation system
- Updated LPML export
- Storage of process schema to Templates Repository

Design Time Composer

Software: https://svn.sti2.at:/SOA4All/trunk/SOA4All-service-construction/SOA4All-serviceconstruction-dtcomposer

Installation instructions: Software: https://svn.sti2.at:/SOA4All/trunk/SOA4All-service-construction/SOA4All-serviceconstruction-dtcomposer/install.txt

Technical improvements:

- Full integration with Process Editor
- Full integration with M30 versions of Semantic Link Operator, Reasoner, Discovery, SemanticSpaces, WSL4J, iServe. Template Storage.

- Improved support for context-based modeling, global requirements and preferences in WSML-DMA
- Improved support for new data-flow mappings features provided by Semantic Link Operator in the SLO-DMA
- Rewriting SD-DMA based on integrated M30 SD prototype.
- Integrated new M30 LPML API.
- Improved performance: query and model caching, multi-threaded architecture

Optimizer

Software: https://svn.sti2.at:/SOA4All/trunk/SOA4All-service-construction/SOA4All-serviceconstruction-optimizer

Installation instructions: https://svn.sti2.at:/SOA4All/trunk/SOA4All-service-construction/SOA4All-serviceconstruction-optimizer/install.txt

Technical improvements:

- Full integration with Process Editor
- Full integration with M30 versions of Semantic Link Operator, Discovery, iServe
- Integrated new M30 LPML API
- Consideration of new data flow manipulation