D5.3.1 Process Optimization (Prototype I)

Authors: TUDA (lead)

Delivery Date:	Due Date:	Dissemination Level:
2013-03-15	2013-02-28	PP

This deliverable provides a description of the first prototype implementation of task T5.3 – Process Optimization. As stated in the Description of Work (DOW), this deliverable is a prototype (software). As such, this document is reduced in length and its only purpose is to briefly describe the prototype functionality as well as to give installation instructions and usage clarifications to the user. This document will be shipped together with the software itself.

SEVENTH FRAMEWORK PROGRAMME

Document History	
Draft Version	V0.1, ASC, July 27th, 2012 V0.2, TUDA, Feb 12th, 2013 V0.3, TUDA, Feb 15th, 2013 V0.4, TUDA, Feb 18th, 2013 V0.41, UVA, Feb 22th, 2013 V0.5, TUDA, Feb 26th, 2013 V0.6, TUDA, Mar 01st, 2013 V0.7, TUDA, Mar 11th, 2013 V0.8, TUDA, Mar 12th, 2013 V1.0, TUDA, Mar 15th, 2013
Contributions	TUDA - Dieter Schuller
Internal Review 1	Ahm Shamsuzzoha, UVA
Internal Review 2	Tiago Gomes, AZEV
Final Version	March 15th, 2013

Table of Contents

List of Figures

Executive Summary

ADVENTURE envisages Virtual Factories for which the optimal selection of partners providing appropriate manufacturing services should not constitute a complex, manual task requiring enormous human efforts for being accomplished. Thus, in order to automatically achieve optimized manufacturing processes, an optimization component is developed in the course of the ADVENTURE project. This component offers the functionality to propose and provide optimized assignments of offered manufacturing services to the activities defined in Smart Processes – based on non-functional service attributes.

Prototype D5.3.1 represents the main outcome of Task T5.3 (Process Optimization) for the first 18 months of the project. It delivers the first functionalities of the Optimization component that will allow ADVENTURE to provide optimized Smart Processes based on non-functional service attributes.

As stated in the Description of Work (DOW), this deliverable is of a prototype (software) nature. Hence, the purpose of this document is to briefly describe and visualise the prototype's functionality, as well as to give installation instructions and usage guidance.

This first deliverable depicts the initial functionality implemented within the two main software components of the Process Optimization: Process Analysis and Process Optimization (see also D3.3 Technical Specification). While the process analysis part takes care of developing and implementing the optimization problem, the actual process optimization part solves this problem by applying standard solution techniques. These solution techniques will be provided by a standard "solver", which is basically exchangeable – as long as the used solver provides the capability to solve linear optimization problems (see Section 2 for further details).

1 Introduction

ADVENTURE – ADaptive Virtual ENterprise manufacTURing Environment – is a project funded in the Seventh Framework Programme by the European Commission. ADVENTURE creates a framework that enhances the collaboration between suppliers, manufacturers and customers for industrial products and services.

1.1 ADVENTURE Project Aims

The framework proposed by ADVENTURE provides mechanisms and tools that facilitate the creation and operation of manufacturing processes in a modular way. ADVENTURE combines the power of individual factories to achieve complex manufacturing processes. It provides tools for partner-finding, process creation, process optimization, information exchange as well as real-time monitoring combined with the tracking of goods and linking them to Cloud services.

There have already been several research projects that address the combination of different independent manufacturers to so-called virtual factories. Most of these research projects focus primarily on the business-side in general and on aspects like partner-finding and factory-building processes in special. However no proven tools or technologies exist in the market that provide the creation of virtual factories applying end-to-end integrated Information and Communication Technology (ICT). ADVENTURE is aiming to provide such tools and processes that will help to facilitate information exchange between factories and move beyond the boundaries of the individual enterprises involved. The collaborative manufacturing process will be optimised by enabling the integration of factory selection, forecasting, monitoring, and collaboration during runtime.

ADVENTURE builds on concepts and methods of Service-oriented Computing and benefits from the advancements in this field. The monitoring and governance of the collaborative processes will be supported by technologies from the Internet of Things such as wireless sensors. Existing tools and services that can be integrated will be considered during the development of the platform for ADVENTURE.

The increased degree of flexibility provided through ADVENTURE will benefit SMEs especially as it helps them to react quickly to changes and to participate in larger, cross-organizational manufacturing processes. Furthermore, ADVENTURE will help manufacturers in assessing the environmental friendliness of actual manufacturing processes and resulting products and services. Other objectives of ADVENTURE include research in areas such as service-based manufacturing processes, adaptive process management, process compliance, and end-to-end-integration of ICT solutions.

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to accompany the prototype implementation of task T5.3 Process Optimization. As such, its main purpose is to briefly clarify the scope of the prototype and to present the download and installation instructions of the software. As the main focus of the task is the development of the software itself, this

accompanying document targets more on providing a short summary of the main functionalities and serving as user guide for the current status of the development.

1.3 Document Status

The first prototypes of ADVENTURE software components have been developed in parallel to technical specification. This was planned in order to use these first prototypes to verify and to validate the requirements and the technical choices made, rather than to provide a complete functionality solution. Hence, the first prototype provides a limited set of basic features (even mock-ups), but no end-to-end complete functionality. The interactions between the different components are also not implemented yet, because as already noted the main objective of this prototype is to learn and adjust the interaction and calling points between the components, as well as provide valuable inputs to the technical specification about the state of the art, technical alternatives, test of tools, etc. This first prototype will evolve over the time, taking into consideration aspects like common graphical design for ADVENTURE user interfaces, embedding the tools into the Dashboard framework, interactions between the implemented software components, etc.

1.4 Target Audience

This document is listed in the DOW as PP, which means 'Restricted to other programme participants (including the Commission Services)' primarily since the audience of the document is largely internal and it is not a final version. It presents the first prototype of the software components of Smart Process Designer that is mainly targeted to the development team and to the user partners, rather than to external audience. The next version of this deliverable planned for M24 of the project with the complete functionality implemented will be made publicly available for external dissemination.

1.5 Abbreviations and General Terms

A definition of general, common terms and roles related to the realization of ADVENTURE as well as a list of abbreviations is available in the supplementary document "Supplement: Abbreviations and General Terms" which is provided in addition to this deliverable.
Further information can be found at: http://www.fp7-adventure.eu

1.6 Document Structure

This deliverable is broken down into the following sections:

- **Section 1:** Provides an overview of the entire document and the related pilot implementation, describing the objectives, constraints and status.

- **Section 2:** Describes the scope of the pilot implementation, its purpose and the main relationships with other modules being implemented at ADVENTURE in the first year.
- **Section 3:** Introduces the information needed to deal with the pilot, in terms of technical and non-technical requirements, software to be installed, etc.
- **Section 4:** Describes the needed steps to install locally the pilot software, and how to build it from the source code.
- **Section 5:** Presents the different screens and actions implemented at the pilot itself, how to access it, and how to test the different implemented options.
- **Section 6:** Depicts the current pilot limitations, and the expected improvements.
- **Section 7:** Describes the conclusions of the implementation of the first prototype.

2 Scope and Relationship

A short reminder of the functionalities of the ADVENTURE Process Optimization component:

- Specification of objectives and restrictions on non-functional service properties for single activities of Smart Processes as well as for overall Smart Processes which are to be satisfied by the optimization
- Optimization of Smart Processes based on non-functional service properties
- Modelling of interdependencies among partner services which are to be considered for the optimization

In the ADVENTURE architecture as presented in Figure 1 below, the Process Optimization component is located in the Process layer, collaborating with the rest of the ADVENTURE modules (especially Process Designer, Process Execution, and Data Provisioning and Discovery component) through the Message Routing component in the Data Exchange layer.

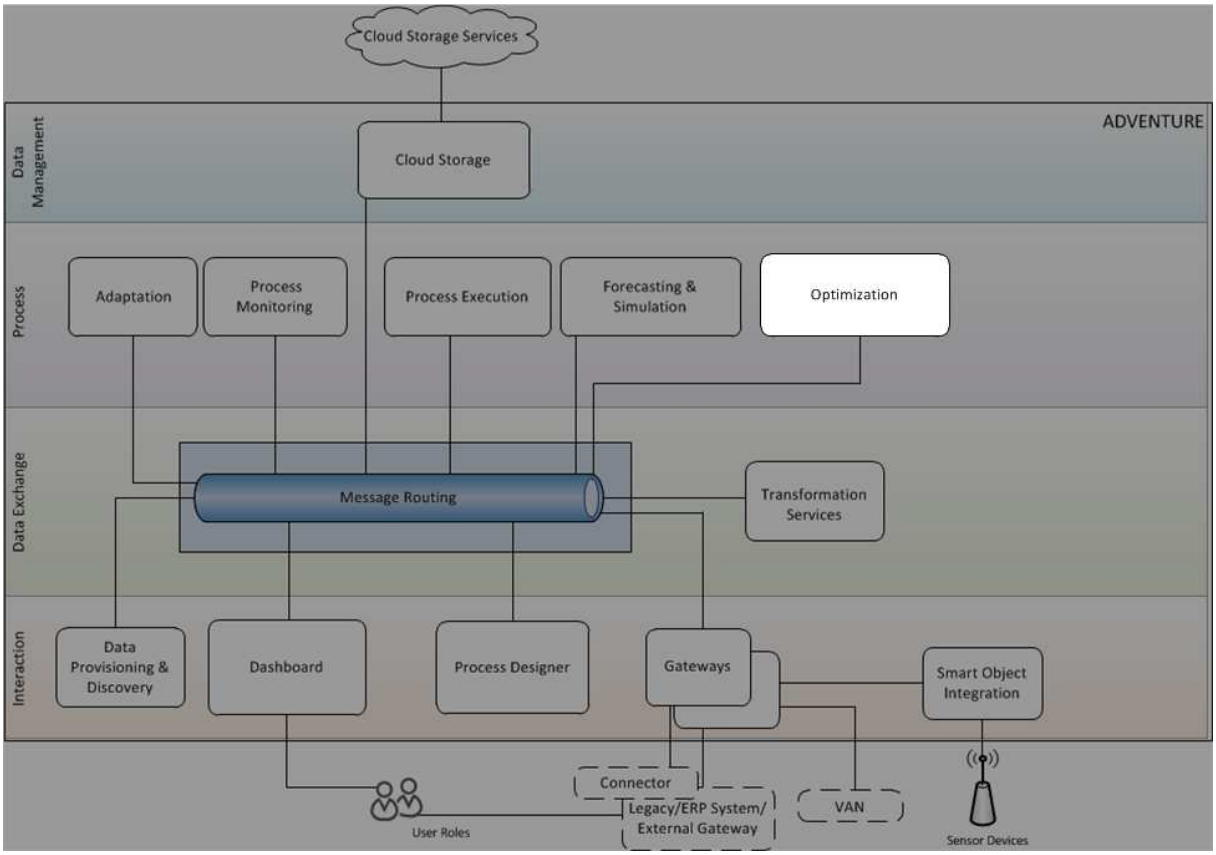

Figure 1: Overall architecture

As previously stated in former deliverables such as D3.2 Functional Specification or D3.3 Technical Specification, the Optimization component is capable of optimizing Smart Processes in the sense that it proposes an optimized assignment of manufacturing services to the activities of the Smart Processes. For this, a manufacturing process, which is referred to as “Smart Process”, has to be developed and modelled beforehand – using ADVENTURE Process Designer. The modelling of a Smart Process comprises specifying the process steps and activities, which have to be performed. Further, appropriate candidate services which come into question for executing the process steps and therewith performing the activities have to be provided. This means, if a manufacturing process has been modelled, i.e., if activities have been created along with candidate services which come into question for executing those activities and therewith accomplishing according tasks, the final selection of manufacturing services (among the provided, eligible ones) can be optimized with respect to non-functional aspects such as execution or delivery time, cost, CO2, etc. For this purpose, the Optimization component is able to provide optimized propositions indicating which services should be selected for executing which activities with respect to targets and restrictions. A target may thereby indicate the target “state” the proposed solution should approach and try to achieve. For instance, aiming at selecting those services that are cheapest may constitute a valid target. The according target state

which should be achieved is an assignment of services to the activities described in the Smart Process that causes lowest, aggregated cost (for the execution of the selected services). A constraint in this vein may be expressed by the requirement to not exceed a certain threshold for delivery time. This constraint not allows the Optimization component to select services such that the aggregated delivery of the selected services is greater than the specified threshold. For instance, it may be required to select those services which satisfy the requirement that the whole Smart Process is finally executed within 10 days at minimal cost. The “10 days” thereby represent the constraint, i.e., the condition, under which the process has to be executed, whereas the “minimal cost” represents the target or goal which thereby is aimed to achieve.

This deliverable (D5.3.1: Process Optimization (Prototype I)) depicts functionalities, relationships to other ADVENTURE components, and technical decisions. It describes the current state of the implementation. As the prototype is not yet interconnected with other ADVENTURE components, especially with the Process Designer, no “real” process models and descriptions which are expected to be provided by the Process Designer as described above are processed by the Optimization component. Instead, a custom, proprietary format is used for parsing and accounting for different process models. An example file indicating the format of the process models currently accepted by the Optimization component is provided in Table 1. The input file is assumed to be provided in the “.txt” format. It has to be noted that the numbers in brackets in the left column of Table 1 are only displayed for indicating line numbers. They actually do not constitute a part of the example file.

(1)	{ProcessModel}
(2)	workflow:[type=sequence,p1,AND,p8];
(3)	AND:[type=and,XOR1,SEQ3];
(4)	XOR1:[type =xor,SEQ1,SEQ2];
(5)	SEQ1:[type =sequence,p2,p3];
(6)	SEQ2:[type =sequence,p4,p5];
(7)	SEQ3:[type=sequence,p6,p7];
(8)	{QoS}
(9)	DeliveryTime,Cost,Availability,Capacity;
(10)	p1:(s11=[3,2,0.99,10.0],s12=[2,3,0.97,13.0]);
(11)	p2:(s21=[5,4,0.98,12.0],s22=[2,6,0.99,9.0]);
(12)	p3:(s31=[3,4,0.98,11.0],s32=[2,5,0.98,13.0]);
(13)	p4:(s41=[4,2,0.96,9.0],s42=[2,3,0.96,10.0]);
(14)	p5:(s51=[4,2,0.98,14.0],s52=[2,4,0.97,15.0]);
(15)	p6:(s61=[4,2,0.98,14.0],s62=[2,4,0.97,15.0]);
(16)	p7:(s71=[5,2,0.98,11.0],s72=[2,4,0.99,13.0]);
(17)	p8:(s81=[4,2,0.98,19.0],s82=[2,4,1,12.0]);
(18)	{Restrictions}
(19)	DeliveryTime=12;
(20)	Availability=0.85;
(21)	Capacity=10;

Table 1: Example file indicating a process model along with candidate services

As indicated in the example file in Table 1, which is expected as input for the Optimization component, the input is separated into different sections. The keyword for indicating a new section is thereby put in curly brackets. As can be seen, the different sections “{ProcessModel }”, “{QoS}”, and “{Restrictions }” have to be provided.

The first section – “{ProcessModel }” – specifies the process model, i.e., the structure along with activities. In line (2), the root element indicating the start of the workflow is specified as follows: “workflow:[type=sequence,p1,AND,p8];” This line indicates, that the root element is of the type “sequence”, and that it contains three sub elements, namely “p1”, “AND”, and “p8”. While “p1” and “p8” actually represent process steps (PS) and activities, respectively, of the Smart Process, the element “AND” has to be further specified. This is done in line (3). The element “AND” is defined to be of type “and”, i.e., it constitutes and AND-block which demands a parallel execution. The sub-elements of the AND-block are indicated as “XOR1” and “SEQ3”. While “XOR1” refers to an XOR-block (see line (4)), “SEQ3” specifies another sequence (see line (7)). The sub elements for the XOR block also constitute further sequences, namely “SEQ1” and “SEQ2”. Workflow specified in this example file is indicated in Figure 2.

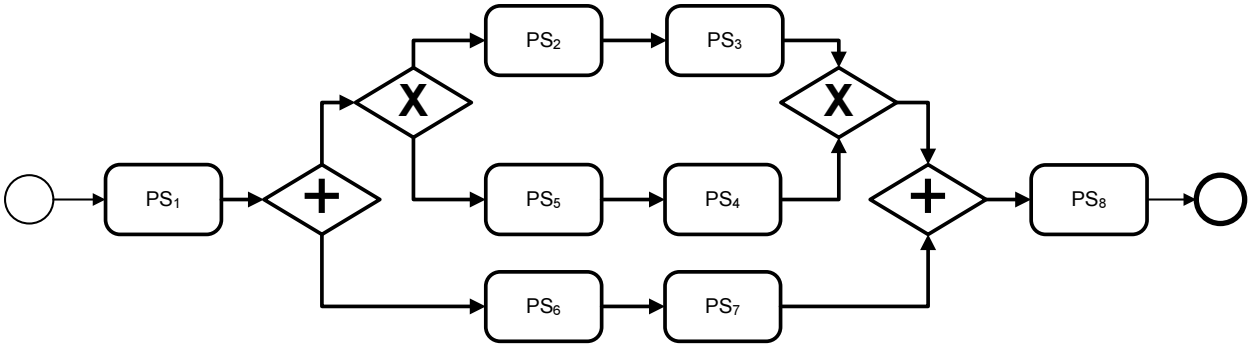

Figure 2: Example Smart Process

The section heading with “{QoS}” in line (8) indicates the section on the non-functional service attributes. In (9), it is specified that it is accounted for “Delivery Time, Cost, Availability, and Capacity”. In lines (10) - (17), the QoS of eligible candidate services appropriate to accomplish the different process steps “p1”-“p8” is indicated. The figures within the squared bracket thereby indicate the values for a candidate service of the mentioned non-functional service attribute. For instance, in line (10), two candidate services are specified – each within squared brackets – appropriate to accomplish activity “p1”. The first candidate service takes “3” days for delivery time, costs “2” TEUR, has an availability of “0.99”, i.e., 99%, and a capacity of “10”. For the second service, the according values are “2” days for delivery time, “3” TEUR as cost, availability of “0.97”, i.e., 97%, and a capacity of “13”.

In lines (18) - (21), the restrictions are finally depicted. For Delivery time, a total duration of not more than 12 days has to be guaranteed, an aggregated availability of 85% has to be achieved, and a capacity of 10 must not underrun. The property “cost” thereby acts as target. As implicitly assumed, it is aimed at minimizing total cost in the current version of this first prototype.

Using these input parameters, the Optimization component parses the provided “process model” – in a tree based format. Using the provided restrictions, i.e., 12 days duration and 85% availability, constraints according to the process structure are developed demanding the total delivery time not to exceed 12 days, achieving a total availability of 85% for the selected services and having a minimum capacity of 10. Using the provided values on non-functional service attributes, the eligible candidate services are accounted for. Finally, according to the provided input parameters, an optimized solution is computed and handed back to the caller, which will be the Process Designer – in most of the cases. The result for the provided example is the following:

$P_1 \rightarrow S_{12}$	$P_2 \rightarrow S_{21}$	$P_3 \rightarrow S_{31}$	$P_4 \rightarrow S_{42}$	$P_5 \rightarrow S_{51}$	$P_6 \rightarrow S_{62}$	$P_7 \rightarrow S_{71}$	$P_8 \rightarrow S_{82}$

In order to achieve such optimized assignments of candidate services to activities of a Smart Process, the Optimization component is implemented basically as two logical units: the Process Analysis and the Process Optimization, as indicated in Figure 3 (see also D3.3 Technical Specification).

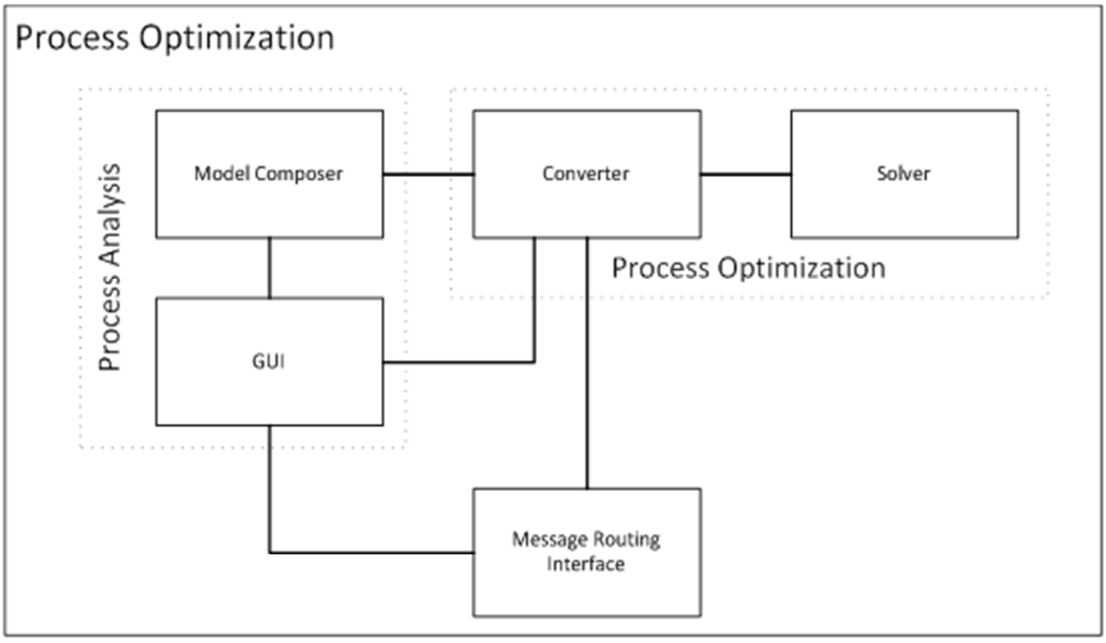

Figure 3: Optimization Component Structure

While the process analysis parts constitutes a 100% custom implementation, the process optimization part comprises a standard solver component and a (customized) interface to this solver. The two parts thereby form completely separable units. This way, the used and applied solver is easily exchangeable for a different solver, e.g., for reasons of extended performance or for licensing issues.

In its current version, the linear programming (LP) solver lpSolve[1], which comes under the LGPL license[2], is used as solver for the Optimization component. lpSolve is based on the revised simplex method[3] and applies the well Branch-and-bound[4] method for considering integers. Referring to the Technical Specification in D3.3, justifications for this decision are provided. In addition to the "standard" solver, also a "standard" interface has been implemented in addition to a custom interface. As standard interface to further (LP) solvers, Java ILP[5] has been used. In order to speed up the performance for computing solutions, the Optimization component is already set up to "collaborate"

[1] http://lpsolve.sourceforge.net/5.5/
[2] http://www.gnu.de/documents/lgpl-3.0.de.html
[3] see, e.g., http://www.simplexmethod.net/ for a short introduction
[4] see, e.g., http://www.seas.gwu.edu/~ayoussef/cs212/branchandbound.html for a short introduction
[5] http://javailp.sourceforge.net/

with the commercial solver CPLEX[6]. Thus, in case of proper licensing, a gain in performance in the magnitude of approx. 2000% can be achieved (see D3.3).

Accounting for "proper" process models provided by the ADVENTURE Process Designer will be realized in the next version of the prototype which is due in month 24.

A key goal for this prototype is to synchronize the implementation process for the Optimization component according to the implementation of other ADVENTURE components, including input and output formats. The process model is thereby expected to be provided by the Process Designer along with a list of eligible candidate services. The values on the considered non-functional service properties have on the other hand to be "actively" fetched by the Optimization component, using and querying the Data Provisioning and Discovery (DPD) component. This current prototype edition focuses on providing the basic functionality, i.e., proposing optimized process models with respect to non-functional service properties, along with a formulation and description of dependent components – both, internal, i.e., the Process Designer, as well as external, i.e., the LP solver, components. The real integration with the Process Designer and with the DPD will be realized in the final version of this prototype, i.e., a 'live' interaction is planned for later stages.

3 Requirements and Preparations

The Requirements and Preparations section describes the information needed to deal with the pilot, in terms of both technical and non-technical requirements, software to be installed, basic knowledge, etc.

3.1 For Users

The Optimization component is accessible as a Restful service at http://localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize. This service provided by the Optimization component actually expects additional inputs in order to achieve optimized process models, as previously stated in Section 2. For providing this input, a url pointing to an input file is expected. An example for such a url is provided in Section 4. Regarding the implementation status of the M18 Prototype I, the Optimization component is as yet not integrated with the other ADVENTURE components. Thus, the required input comprising the process model, the candidate services, the values of non-functional service attributes, as well as the constraints, have to be provided directly to the Optimization component. For this, while invoking the mentioned Restful service, a link to where to find the according input file (see Section 2, esp. Table 1) is necessary.

In order to run the Optimization component, the installation of an LP solver (such as lpSolve – see Section 2) is required in addition to the implementation of the Optimization component. In this respect, it is recommendable to have some knowledge about linear programming. A "rather" brief introduction is accessible at

[6] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

http://www.purplemath.com/modules/linprog.htm. Further, rather concise introduction can be found at http://www.math.ucla.edu/~tom/LP.pdf.

Further, the Optimization component provides a Graphical User Interface (GUI), which can be accessed using a standard internet browser. Similar to the accessing the mentioned Restful service directly, a link to the input file can be provided to the Optimization using the according GUI. In addition, the optimization problem indicated in the input file can be manually adapted and changed, using the GUI (see D3.2 Functional Specification as well as D3.3 Technical Specification). An according mock-up screen is available at

http://fp7-adventure.eu:8080/ADVENTURE-Optimization-Component/VAADIN.

3.2 For Developers

As previously stated, the Optimization component is accessible as RESTful service at http://localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize. In order to actually compute optimized process models, the input file (see Section 2, esp. Table 1), indicating the process model, the candidate services, the values on non-functional service attributes, as well as the constraints, have to be provided directly to the Optimization component. For this, a url pointing to the input file has to be appended to the link to the mentioned service. An example for such a url is provided in Section 4.

In order to deploy and run the D5.3.1 prototype, the following software and input, respectively, is needed:

- Java JSE6 or superior (http://www.java.com/en/download/index.jsp).
- Apache Tomcat (http://tomcat.apache.org/) or JBOSS 7 (http://www.jboss.org/jbossas/) application server.
- Eclipse, with subversion (e.g. http://www.eclipse.org/subversive/) plug-ins installed.
- Java ILP – this tool provides the mentioned standard interface to multiple LP solvers (http://javailp.sourceforge.net/)
- LP solver, such as lpSolve – free LP solver (http://sourceforge.net/projects/lpsolve/files/lpsolve/5.5.2.0/)
- Input file as .txt file – indicating the process model, the candidate services, the values on non-functional service attributes, and the constraints Process models

The current version of this prototype uses lpSolve as LP solver. Thus, the according solver has to be downloaded and installed. According installation instructions are provided in Section 4. Also other solvers such as CPLEX as previously stated may be used, which would have to be installed instead. In addition, as also JAVA ILP is used in order to provide a standard interface to other solvers. Thus, also Java ILP has to be downloaded.

4 Installation (Deployment)

The following steps are needed for a local deployment of the pilot software:

- Check the requirements for developers (Section 3.2), so that an appropriate JSE version and Application Server (AS) is installed on the computer. From now on it is assumed that TOMCAT AS is selected, but steps are similar in the case of selecting JBOSS AS.
- Setup for lpSolve (installation instructions are based on http://lpsolve.sourceforge.net/5.5/Java/README.html):
  - Download the following .zip file for the current version for lpSolve (v. 5.5.2.0) (for a 32 bit Windows System) from http://sourceforge.net/projects/lpsolve/files/lpsolve/5.5.2.0/
    - lp_solve_5.5.2.0_exe_win32.zip
    - lp_solve_5.5.2.0_dev_win32.zip
    - lp_solve_5.5.2.0_java.zip
  - Copy the lpSolve dynamic libraries from the archives `lp_solve_5.5_dev.(zip or tar.gz)` and `lp_solve_5.5_exe.(zip or tar.gz)` to a standard library directory for your target platform. On Windows, a typical place would be `\WINDOWS` or `\WINDOWS\SYSTEM32`.
  - Unzip the Java wrapper distribution file (lp_solve_5.5.2.0_java.zip) to new directory of your choice
  - On Windows, copy the wrapper stub library `lpsolve55j.dll` (can be found in lp_solve_5.5.2.0_java.zip\lp_solve_5.5_java\lib\win32) to the directory that already contains `lpsolve55.dll`
  - Copy the archive file `lpsolve55j.jar` from the Java wrapper distribution to a directory that is included in the CLASSPATH of your java program
  - If you installed the native stub library in a directory that is not included in the PATH variable, you have to define the Java system variable `java.library.path` which must point to the installation directory
- Process optimization setup:
  - Download the implementation of the Optimization component at (http://fp7-adventure.eu/components/ADVENTURE-Optimization-Component.war) into a local directory.
  - Download the sample files containing process models along with candidate services, values on non-functional service attributes, and constraints Process (http://fp7-adventure.eu/components/optimization-example-files.zip) into a local directory (e.g., C:/Process-Optimization/example-files/). One of the zipped files is named AND_XOR.txt. It constitutes the example file from Table 1.
  - Copy process-optimization.war to the application server deployment folder (<TOMCAT>/webapps).

- Access Process-optimization for instance using a web browser at localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize?processPath=*[url to process model]*
- For instance, the *[url to process model]* may be a local file such as "file:///C:/Process-Optimization/example-files/AND_XOR.txt", so that the whole access looks as follows: http://localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize?processPath=file:///C:/Process-Optimization/example-files/AND_XOR.txt

Additionally to binary (war) deployment, the Optimization component can be downloaded and built from its source code from the ADVENTURE Subversion (SVN) server. It is recommended to have a Subversion (e.g. subversive - http://www.eclipse.org/subversive/) plug-in installed on Eclipse, or use an external Subversion client application (like tortoise-svn at http://tortoisesvn.net/) to check-out the project code.

5 Execution and Usage

Once the prototype of the Optimization component is downloaded and appropriately installed (see Section 4), the Optimization component may be started and used.

In order to use the provided Restful service, start a TOMCAT server and open a web browser. Access the Optimization component at "localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize?processPath=*[url to process model]",* whereas *[url to process model]* indicates the url, i.e., the path the input file. For instance, the *[url to process model]* may be a local file such as "file:///C:/Process-Optimization/example-files/AND_XOR.txt", so that the whole access looks as follows: http://localhost:8080/ADVENTURE-Optimization-Component/rest/rest/optimize?processPath=file:///C:/Process-Optimization/example-files/AND_XOR.txt

The result of the optimization is shown as follows:

As indicated in Figure 4, the proposed, optimized assignment of candidate services to the activities of the Smart Process is provided.

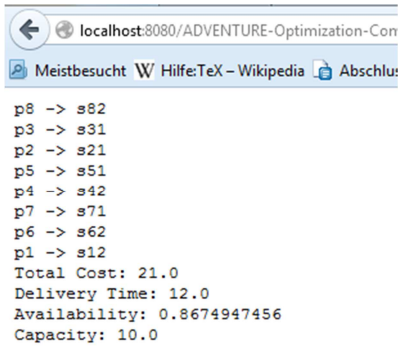

Figure 4: Result of Optimization Service

In order to run the GUI, run the file “Application.java” located at the package “gui” as “java Application”. Subsequently, the GUI, which is displayed in Figure 5, will be opened, will be opened.

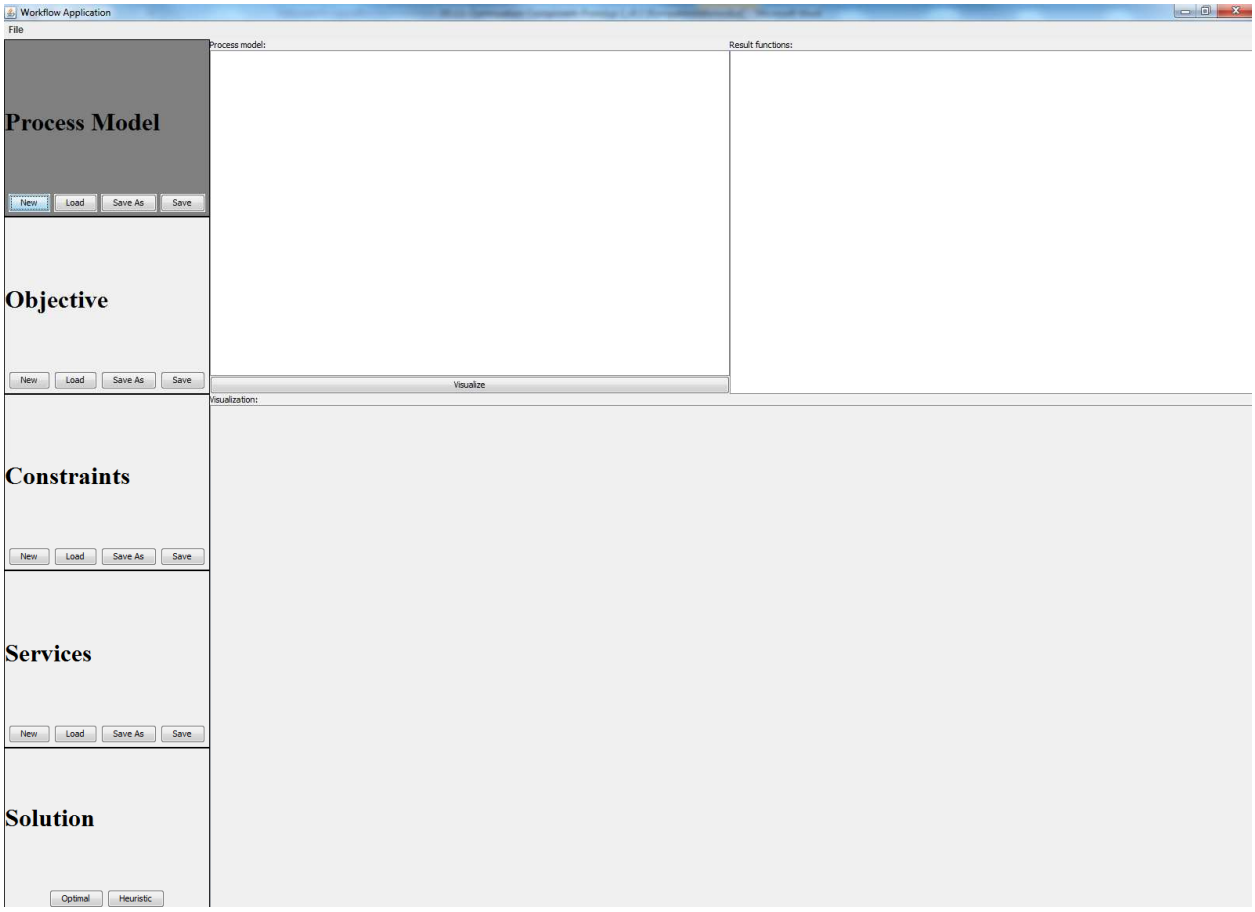

Figure 5: Welcome screen of the Optimization component

Clicking on “file→Load All” enables selecting the location, where the expected input file can be found. Having opened the file, the process model as well as the list of candidate services along with according values on non-functional service attributes and constraints on those attributes will be provided in the different tabs “Services” and “Constraints” as shown in Figure 6.

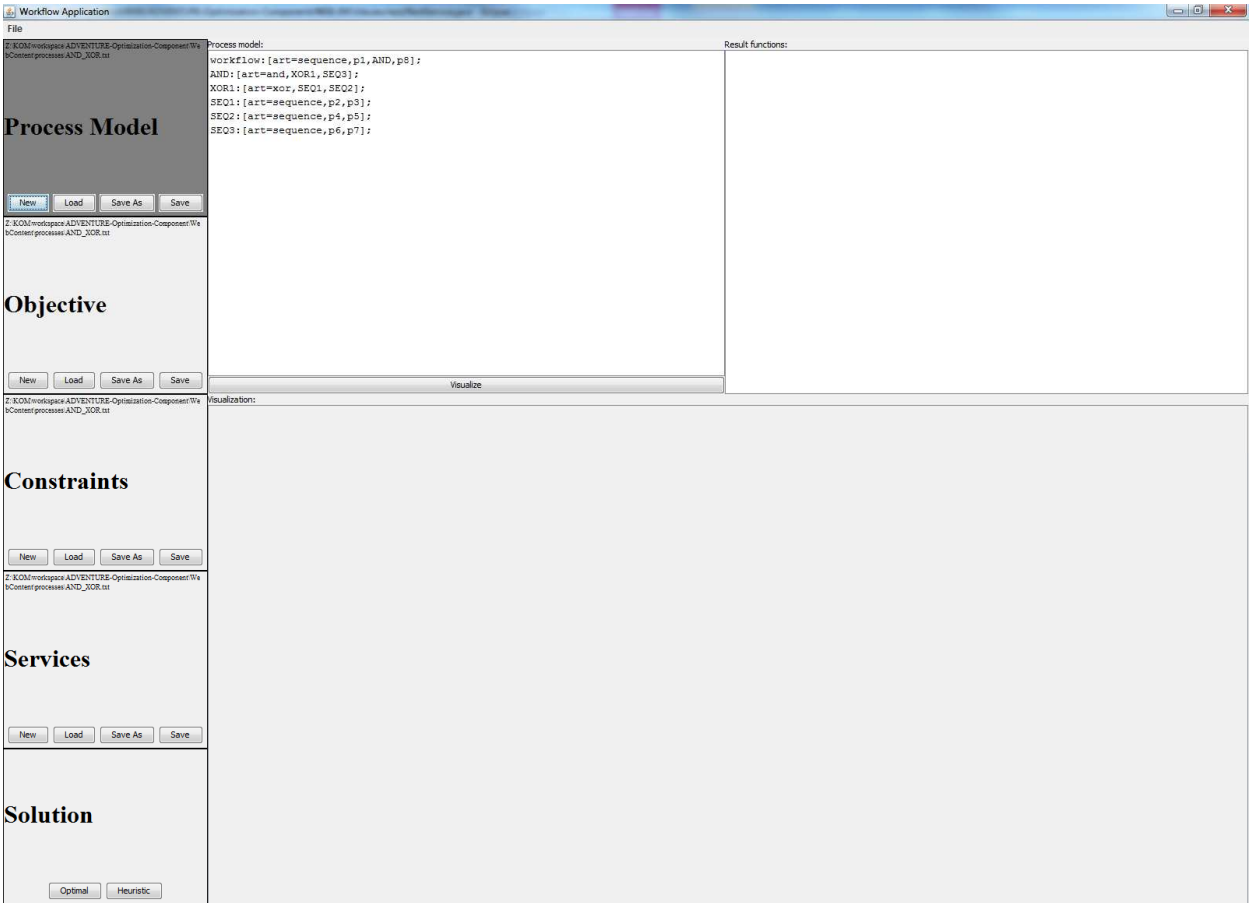

Figure 6: Opened Process Model

The loaded data may now be adapted, i.e., the process model may, for instance, be changed or according candidate services may be added. The structure of the Smart Process may be shown by clicking the button “Visualize”.

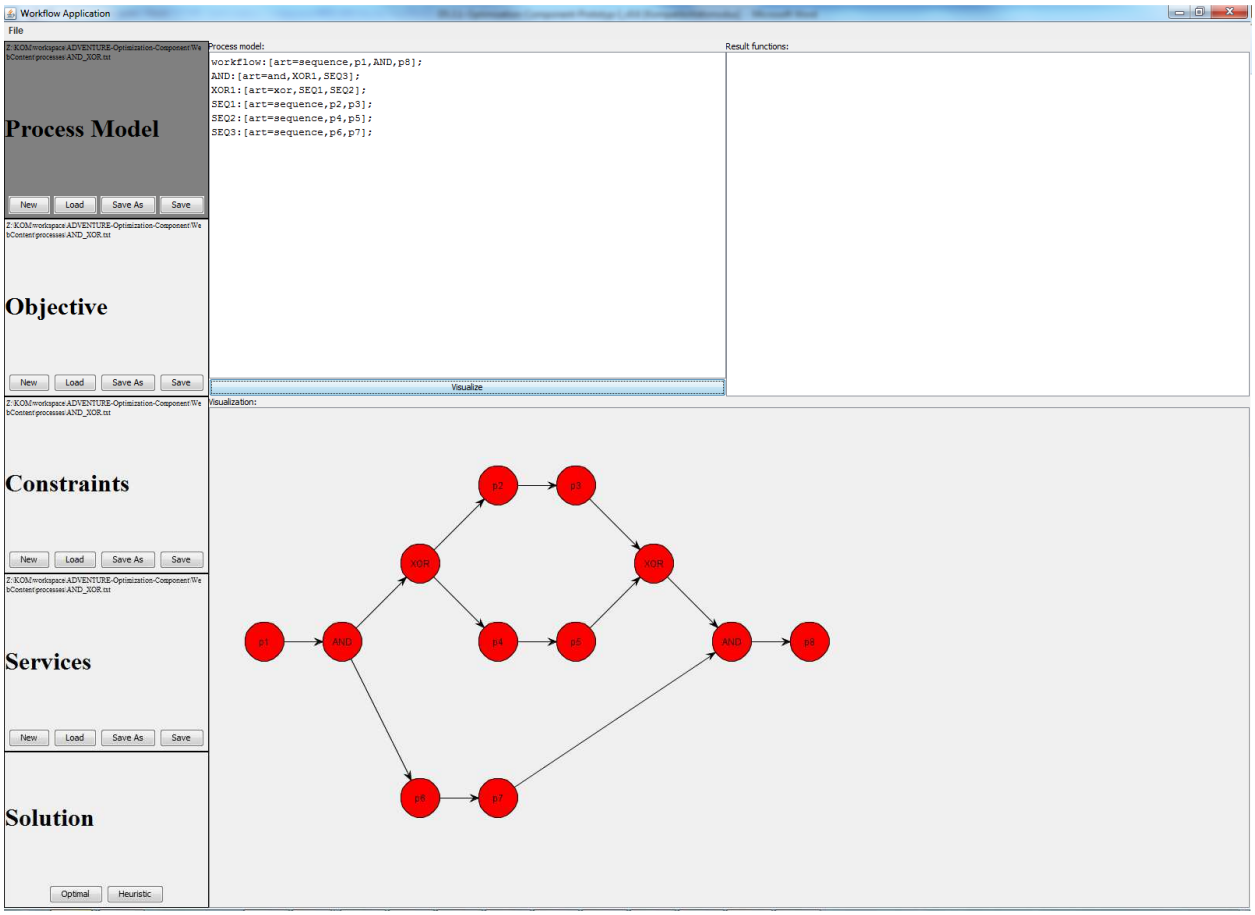

Figure 7: Visualization

Clicking the “Optimal” button in the “Solution” tab triggers the computation of an optimal solution, which is displayed in the result field of the GUI (see Figure 8). In addition, the constraints developed according to the provided restrictions are displayed in this field.

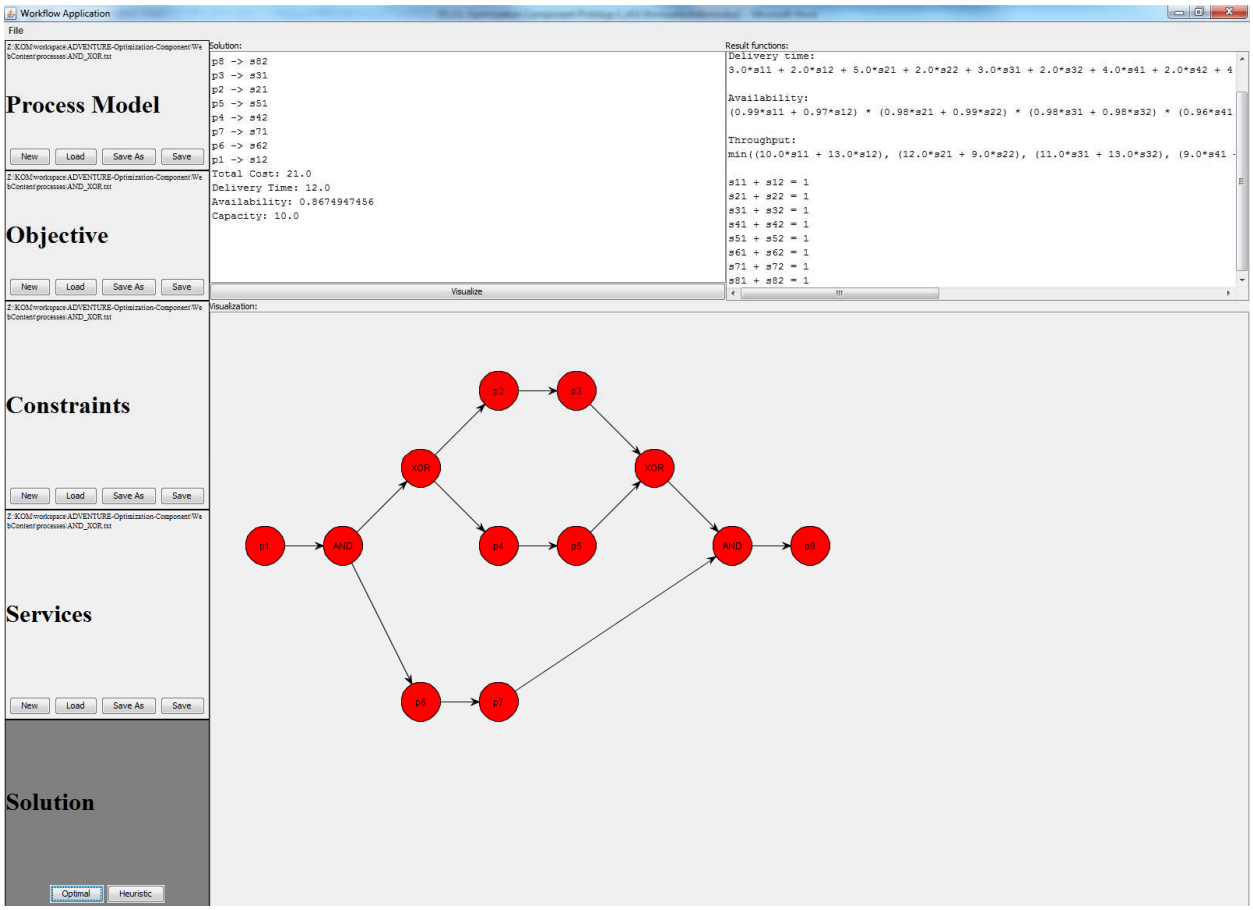

Figure 8: Optimized assignment of services to activities

6 Limitations and Further Developments

The current pilot of the Optimization component has as yet a limited functionality serving as a basis for the further developments. As a first prototype, it contains different known bugs and functionality flaws that will be solved during the development process.

To start with, the functionality is limited with respect to the workflow patterns which can be addressed and for which an optimal solution can be computed. In this version, it is only accounted for sequences, AND-blocks (i.e., AND-splits with according AND-joins) as well as for XOR-blocks (i.e., XOR-splits with according XOR-joins). Further patterns such as Repeat loops or OR-blocks (i.e., OR-splits with according OR-joins) will be considered in the final version.

In addition, the considered non-functional service attributes are considered to be static, which actually does not represent reality, where deviations, e.g., expected delivery times exist for which it actually should be accounted for stochastic non-functional service attributes. The final version will be able to also account for stochastic attributes.

The current prototype also is only able to accept “minimize total cost” as objective. This means that the Optimization component always will select those services which lead to lowest cost. Accounting for other objectives or combinations of other objectives will be possible in the final version of the prototype.

Further, only constraints restricting the non-functional service values on a global, i.e., process level are currently accounted for. For instance, constraints demanding the delivery time to be lower than 10 days represent restrictions on the whole workflow but not with respect to single services. For this, also taking restrictions on the service level will be realized in the final version.

Finally, the integration with the other ADVENTURE components, especially with ADVENTURE’s Process Designer, the DPD and the Process Execution will be achieved in the final version of this prototype.

The current document will serve as a base for delivering subsequent iterations of the Optimization component’s prototype. Thus, this document should be seen as a living document that evolves together with the prototype.

7 Conclusion

In order to automatically achieve optimized manufacturing processes in ADVENTURE, an Optimization component has been developed. This component computes and provides optimized assignments of manufacturing services offered by (potential) partner factories to the activities of Smart Processes which have been modelled using ADVENTURE’s Process Designer. These assignments are thereby based on non-functional service attributes. This document represents an accompanying manual, assisting an ordinary user as well as an expert in installing and using the according software tools.

For the first prototype of the Optimization component, initial functionalities have been implemented and an initial version of the intended GUI has been provided. The objective of this first prototype is enabling computing optimized Smart Processes based on non-functional service attributes. For this, the specification of targets and restrictions has been enabled and realized. This first prototype is capable of considering Smart Processes consisting of sequences, AND-blocks, and XOR-blocks. The support for further workflow patterns is envisaged for the final next version.

In contrast to the current version, the final prototype for the Optimization component will also consider restrictions on activity level and account for multiple, weighted targets in its next prototype version.