Collaborative Large-scale Integrating Project

Open Platform for EvolutioNary Certification Of Safety-critical Systems

Specification of the compliance-aware service infrastructure

D7.3

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP7: Transparent Certification and Compliance-aware Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination level:</td>
<td>PU</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>10 Dec. 2013</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Jérôme Lambourg (AdaCore)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:lambourg@adacore.com">lambourg@adacore.com</a></td>
</tr>
</tbody>
</table>

PROPRIETARY RIGHTS STATEMENT
This document contains information proprietary to the OPENCOSS Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the OPENCOSS consortium.
Contributors

<table>
<thead>
<tr>
<th>Names</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jerome Lambourg</td>
<td>AdaCore</td>
</tr>
<tr>
<td>Dariusz Oszczedlowski, Magdalena Gniewek</td>
<td>Parasoft</td>
</tr>
<tr>
<td>Maurice Assouline, Eric Andrianarison</td>
<td>Atego France</td>
</tr>
<tr>
<td>Huáscar Espinoza, Zurik Corera</td>
<td>Tecnalia</td>
</tr>
<tr>
<td>Thierry Ventadour, Rodolphe Arthaud</td>
<td>inspearit</td>
</tr>
</tbody>
</table>

Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>2013-05-22</td>
<td>Document creation and initial ToC</td>
</tr>
<tr>
<td>V0.2</td>
<td>2013-06-28</td>
<td>Initial version</td>
</tr>
<tr>
<td>V0.3</td>
<td>2013-07-01</td>
<td>Technologies for Webservices API</td>
</tr>
<tr>
<td>V0.4</td>
<td>2013-07-02</td>
<td>Short description of semantic technologies utilization</td>
</tr>
<tr>
<td>V0.5</td>
<td>2013-07-02</td>
<td>Review, comments &amp; updates (mainly section 1 and figure 2)</td>
</tr>
<tr>
<td>V0.6</td>
<td>2013-07-04</td>
<td>Candidate Process Management tools to interface with the OPENCOSS platform - Parasoft Concerto added</td>
</tr>
<tr>
<td>V0.7</td>
<td>2013-07-22</td>
<td>Chapter 4 updated after the TELCO (2013-07-16) Thierry Ventadour (inspearit)</td>
</tr>
<tr>
<td>V1.0</td>
<td>2013-10-10</td>
<td>Finalize the document: Executive summary, Conclusions, general homogeneity.</td>
</tr>
<tr>
<td>V1.1</td>
<td>2013-10-14</td>
<td>Final version after internal review.</td>
</tr>
<tr>
<td>V1.2</td>
<td>2013-12-10</td>
<td>Final version after review</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

## Abbreviations ................................................................. ................................................ 6

## Executive Summary .................................................................................... ............................................ 7

### 1 Introduction ......................................................................................... ...................................... 8
   1.1 Scope and Purpose ................................................................................................................... 8
   1.2 Relationship with other Deliverables ....................................................................................... 9
   1.3 Structure of the Document .................................................................................................... 10
   1.4 Lifecycle of this document ..................................................................................................... 10

### 2 Background ........................................................................................... ................................... 11
   2.1 Requirements for Process Management ................................................................................ 11
   2.2 Process Management Components ....................................................................................... 11

### 3 Data Model for Process Management ........................................................... 13

### 4 Component Specification .............................................................................. 16
   4.1 Process Specification and Standards Mapping Editor ............................................................ 16
      4.1.1 Use cases overall presentation ................................................................................. 16
      4.1.2 Editing processes ....................................................................................................... 17
      4.1.3 Mapping processes with standards or other processes ............................................ 21
   4.2 Process Execution and Compliance Monitoring Manager ..................................................... 27
      4.2.1 Use cases overall presentation ................................................................................. 28
      4.2.2 Initiate Project ........................................................................................................... 28
      4.2.3 Update process status ............................................................................................... 30
      4.2.4 Monitor process execution ....................................................................................... 32
      4.2.5 Specify formal process rules................................................................. 33

### 5 Candidate Technologies .............................................................................. 35
   5.1 Candidate Process Management tools to interface with the OPENCOS platform ............... 35
      5.1.1 Atego Process Director & Atego Process Consumer ................................................. 35
      5.1.2 Parasoft Concerto ...................................................................................................... 54
   5.2 Candidate technologies for the APIs ...................................................................................... 65
      5.2.1 Web services (in large: SOAP, REST, ...) ..................................................................... 65
      5.2.2 Open Services for Lifecycle Collaboration ..................................................................... 69
   5.3 Candidate technologies for the assessment .......................................................................... 71
      5.3.1 Process Mining .......................................................................................................... 71
      5.3.2 Semantic ontology ..................................................................................................... 74

### 6 Conclusion .................................................................................................... 77
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functional decomposition for the OPENCOSS tool platform</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Overview of Process Management, interacting with an external tool</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Components of the Process Assurance Management</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Reference Assurance Framework Metamodel</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>CCL Elements related to Process Management</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Use cases for the Process Editor</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>Example of a Standard and a Process to be mapped</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>Use cases for the monitoring of process execution</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>Process director libraries</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>Process contents and process steps authoring</td>
<td>36</td>
</tr>
<tr>
<td>11</td>
<td>Example of a process library (spiral) and BPMN view</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>Question and answer wizard for tailoring</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>Scheduling tasks and exporting Gantt</td>
<td>37</td>
</tr>
<tr>
<td>14</td>
<td>User logging page</td>
<td>38</td>
</tr>
<tr>
<td>15</td>
<td>Friendly access to process content</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>Capturing process progress</td>
<td>39</td>
</tr>
<tr>
<td>17</td>
<td>Visual process progress</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>Reports</td>
<td>40</td>
</tr>
<tr>
<td>19</td>
<td>Process and Project Libraries</td>
<td>40</td>
</tr>
<tr>
<td>21</td>
<td>OPENCOSS levels of modelling</td>
<td>41</td>
</tr>
<tr>
<td>23</td>
<td>Defined library types</td>
<td>42</td>
</tr>
<tr>
<td>24</td>
<td>Reference assurance project metamodel</td>
<td>42</td>
</tr>
<tr>
<td>25</td>
<td>Process metamodel</td>
<td>43</td>
</tr>
<tr>
<td>26</td>
<td>RefActivity &amp; Activity library type</td>
<td>44</td>
</tr>
<tr>
<td>27</td>
<td>Reference assurance project metamodel</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>Artefact metamodel</td>
<td>45</td>
</tr>
<tr>
<td>29</td>
<td>RefArtefact &amp; Artefact library type</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>RefTechnique &amp; Technique library type</td>
<td>47</td>
</tr>
<tr>
<td>31</td>
<td>Role library type</td>
<td>47</td>
</tr>
<tr>
<td>32</td>
<td>Organization library type</td>
<td>48</td>
</tr>
<tr>
<td>33</td>
<td>Master project example in process library</td>
<td>48</td>
</tr>
<tr>
<td>34</td>
<td>BPMN view of example project</td>
<td>49</td>
</tr>
<tr>
<td>35</td>
<td>View in project library</td>
<td>50</td>
</tr>
<tr>
<td>36</td>
<td>Exchanges with external process management tool</td>
<td>50</td>
</tr>
<tr>
<td>37</td>
<td>Overview of Concerto</td>
<td>55</td>
</tr>
<tr>
<td>38</td>
<td>Example of using Concerto task in OPENCOSS project</td>
<td>56</td>
</tr>
<tr>
<td>39</td>
<td>List of tasks in Concerto</td>
<td>57</td>
</tr>
<tr>
<td>40</td>
<td>Publish/Subscribe Messaging</td>
<td>58</td>
</tr>
<tr>
<td>41</td>
<td>An overview of multiple level conformance checking</td>
<td>72</td>
</tr>
<tr>
<td>42</td>
<td>Conformance checking in OPENCOSS</td>
<td>73</td>
</tr>
<tr>
<td>43</td>
<td>Protégé ontograf example</td>
<td>75</td>
</tr>
</tbody>
</table>
List of Tables

Table 1 Libraries and Atego Process Director concepts ................................................................. 40
Table 2 Atego Process Director concepts used for OPENCOSS implementation ........................... 41
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALM</td>
<td>Application Lifecycle Management</td>
</tr>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>CENELEC</td>
<td>Comité Européen de Normalisation Electrotechnique (European Committee for Electrotechnical Standardization)</td>
</tr>
<tr>
<td>DoW</td>
<td>OPENC OSS Description of Work</td>
</tr>
<tr>
<td>DX.Y</td>
<td>OPENC OSS deliverable X.Y</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>GSN</td>
<td>Goal Structuring Notation</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>PLM</td>
<td>Product Lifecycle Management</td>
</tr>
<tr>
<td>RTCA</td>
<td>Radio Technical Commission for Aeronautics</td>
</tr>
<tr>
<td>SACM</td>
<td>Structured Assurance Case Metamodel</td>
</tr>
<tr>
<td>TX.Y</td>
<td>OPENC OSS task X.Y</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>WP</td>
<td>OPENC OSS work package</td>
</tr>
</tbody>
</table>
Executive Summary

This document (D7.3) is the third deliverable of WP7. This WP aims at defining a safety certification management infrastructure to support the certification process. The overall goal of D7.3 is to specify the detailed architecture for the compliance-aware service.

Several inputs have been taken into consideration for specifying this compliance-aware service. In addition to the various high-level requirement documents, we also considered reuse aspects and adoption by the industry. In particular, the compliance-aware service is built around the notions of process compliance and of process execution compliance. As there already exist many tools and methods to define processes and track their execution, and considering that many companies already use such tools and that process tracking is not in itself one of the goals of OPENCOSS, we decided to concentrate on the certification aspects of process definition, on the ways to retrieve the relevant information from external tools, and on expressing this information using the Common Certification Language (CCL). Those considerations have led to new detailed requirements for system components.

Those considerations have led to the detailed requirements expressed in chapter 4.

As part of this design process, we also contributed to the definition of process aspects for the CCL, concentrating on certification-relevant aspects and integration with the other services provided by OPENCOSS (in particular evidence management, provided by the WP6). These process aspects have been roughly described in this document in the chapter 3. Please note, however, that the document D4.4 represents the official definition of the current version of the CCL.

Finally, in order to verify that our set of requirements is relevant, we identified many candidate process management tools, and evaluated the best way to make them interact with the OPENCOSS platform.

The results of this deliverable will serve as basis for T7.4, which will develop the design presented in this deliverable. It is also expected that this deliverable will evolve as OPENCOSS progresses and as new needs or design issues are discovered.
1 Introduction

1.1 Scope and Purpose

The goal of D7.3 is to specify the compliance-aware service infrastructure of OPENCOSS. This service is built around a Process Assurance Management module used to describe and assess a process in regard to safety and regulation constraints imposed by safety standards: this means that process aspects that are outside of this safety area are not necessarily considered.

OPENCOSS is not going to propose a full process management as would be expected from an ALM/PLM tool. Instead, it will support the use of an external tool for managing processes, importing from this tool all of the process aspects related to safety and certification. This external tool could either be a tool dedicated to process management, such as Atego Process Director, or an ALM/PLM tool. OPENCOSS will provide the means to feed the platform with process-related information from the external tool. This information must be sufficient to enable the assessment of conformance to safety standards in relation to processes, as shown in the overall OPENCOSS architecture below.

![Diagram of OPENCOSS tool platform](image)

**Figure 1 Functional decomposition for the OPENCOSS tool platform**

Deliverable D7.2 identified 3 main functional areas for the Process Assurance Management functionality, as expressed in Figure 2. Those 3 functional areas are:

- Process model translation and import;
- Project Process execution notification, and;
- Assessment of project process conformance to standards;

Those functional areas are detailed further in section 2.1: Requirements for Process Management.
As shown in the upper part of Figure 2, the external tool may allow description and export not only of the description of a project specific process (CMMI uses the term of defined process for this), but also (optionally) of the process-related aspects of a Reference Framework. This could be an alternative way, besides the “OPENCOSS standards editor”, of entering this part of a standard into the OPENCOSS platform. For instance, the tool Atego Process Director allows users to create “process libraries”, which are intended to store the description of “reference” processes or “best practice guidelines” processes, as well as “project libraries”, which can contain the processes of multiple projects, as tailored instances of processes based on the process library.

1.2 Relationship with other Deliverables

D7.3 is related to other OPENCOSS deliverables, which have served as input, with which consistency must be kept, or that will use its results. These deliverables, and the relationship of D7.3 with them, are as follows:

- D2.3 (OPENCOSS platform architecture) presents the overall OPENCOSS tool platform architecture, whose process management-related components are specified in detail in WP7.
- D2.4 (Detailed specification of usage scenarios) complements this deliverable by presenting details about how the OPENCOSS platform will be used and some initial user interface mock-ups.
• D3.2 (Integration requirements and test plan) will specify the test cases for validation of the components specified in this deliverable.
• D4.3 (Intermediate Common Certification Language: Conceptual Model) and D4.4 (Common Certification Language: Conceptual Model) present the CCL, which constrains how processes will be managed in WP7.
• D7.2 (Detailed requirements for evidence management of the OPENCOSS platform) includes the requirements that must be implemented in WP7 and thus satisfied by the compliance management components.
• D7.5 (Implementation of the process-specific service infrastructure) will develop the design presented in this deliverable.

1.3 Structure of the Document

The rest of the deliverable is structured as follows. Section 2 introduces the background on which the specification of the components for process management is based. Section 3 presents a data model for process assessment management. The components are specified in Section 4. Section 5 describes the main candidate technologies for development of the process management services. Section 6 presents our conclusions.

1.4 Lifecycle of this document

This document is meant to be a live document, and as such is supposed to evolve along with the actual implementation of the compliance-aware service, and so might be modified if design issues or missing requirements are found later on in the project.
2 Background

This section presents the background work on which the creation of this deliverable has been based. Such work corresponds to the results of other OPENCOSS deliverables and to activities that have been specifically performed for D7.3.

The following subsections outline the requirements for process management, specify the process management components, and describe the functionalities of the OPENCOSS platform regarding process compliance verification and assessment, and process execution notification.

2.1 Requirements for Process Management

In D7.2, six main functional areas were defined for specifying component-level requirements related to process management by the OPENCOSS platform. Among them, three are related to the compliance-aware service:

- **Process description mapping**: the goal is to provide an API to allow the translation of parts of a process description – described using the native formalism of the external tool – into the OPENCOSS description language (the CCL: Common Certification Language).
- **Execution notification**: this part will also consist in providing an API allowing the ALM/PLM tool to feed the OPENCOSS platform with process execution information allowing assessment of the certification process.
- **Conformance to standards**: this part is fully internal to OPENCOSS, and will provide assessment features to the OPENCOSS end-user. This means conformance of the process description to the standard, with potential arguments attached to this conformance assessment.

The analysis of these functional areas resulted in the specification of a set of 24 component-level requirements.

2.2 Process Management Components

The overall OPENCOSS tool platform architecture (Figure 1 Functional decomposition for the OPENCOSS tool platform) contains a module for process assurance management. This includes both APIs to communicate with external ALM/PLM tools as well as internal functionalities to assess the process with regard to applicable standards or safety demonstration.

As shown in Figure 3, the module consists of three components:

- **Process Specification and Standards Mapping Editor**, which includes services to import process specifications and to map to what the standard recommends.
- **Process Static Analysis Manager**, which includes process evaluation (assessment of process completeness and adequacy) and process change impact analysis (identification and analysis of possible effects resulting from changes in processes).
- **Process Dynamic Execution and Compliance Monitoring Manager**, for linking with ALM or workflow execution tools in order to monitor compliance to standards.

This deliverable deals with the specification of those components.
Figure 3 Components of the Process Assurance Management
3 Data Model for Process Management

This section presents WP7-specific data needs by means of a data model (Figure 5 CCL Elements related to Process Management) in the form of an ecore diagram.

It must be noted that this section expresses concepts of the CCL that are defined in D4.4: Common Certification Language. This explanation is here only to ease the understanding of this document by highlighting process-management specific aspects of the CCL that are used in Sections 4 and 5.

Process management uses two parts of the CCL to perform its tasks.

The first CCL element used by the Process Management service is the Reference Assurance Framework metamodel, which captures the high-level concepts and relationships required to model company standards, safety standards, and best practices. This metamodel is used to express the Process definition in terms of activity constraints.
Figure 4 Reference Assurance Framework Metamodel
The second CCL element used by the Process Management service is the Process Management metamodel. It is used to capture the Process execution.

This figure is not complete, and is linked to meta-classes defined in the context of other OPENCOSS work-packages. Below are the explanations of process-specific meta-classes, and pointers to the documents explaining the ‘imported’ notions that we also need to manipulate in the context of WP7.

For a complete description of the two above metamodels and their relationship with the other metamodels and concepts of the CCL, please see D4.4.
4 Component Specification

This chapter specifies components involved in the modelling of standards and processes, on the one hand, and the execution and monitoring of the applied process, on the other hand. It explores further the use cases initially identified in D7.2 and, for each use case, identifies associated functionalities and more detailed requirements.

**Warning:** Clearly, the initial version of the component cannot comply with all of the requirements identified here. Some prioritisation must be done. But unless they are found irrelevant, the requirements not implemented in OPENCOSS should be kept as a contribution for future releases.

4.1 Process Specification and Standards Mapping Editor

This corresponds to two separate components described in D2.3:

- Standards and Intents Editor
- Standards and Intents Mapping Editor

The main reason for merging them here is that:

1) The objects to be mapped are the objects to be edited in the first place
2) Mapping “on-the-fly” is the most efficient approach when possible (see §4.1.3.1, Map processes with the relevant standard for the various scenarios)

Thus, it is likely that the functionalities supporting mapping will be tightly integrated in the editor. In the rest of the document, we refer to this component as Process Editor.

4.1.1 Use cases overall presentation

The following requirements from D7.2 are addressed:

WP7-01-01 – The OPENCOSS platform should provide a means to define a user’s processes manually in the platform in the CCL formalism.

WP7-01-02 – The OPENCOSS platform should provide a means to import a process definition from external process management tools.

WP7-01-03 – The OPENCOSS platform should provide a means to map “CCLized” user’s processes (e.g. those expressed in the CCL formalism, see WP-01-01) with the relevant standard(s) (also expressed in the CCL formalism).

WP7-01-04 – The OPENCOSS platform should provide a means to keep traceability between user’s processes (expressed in the user’s initial formalism or native language) and the relevant standard.

WP7-01-05 – The OPENCOSS platform should provide and maintain (e.g. standards evolutions and new versions) all standards – which are in the scope of the OPENCOSS Platform – expressed in the CCL formalism (“CCLized”).

These are expressed as requirements; we rephrase them slightly and refine them as use cases (Figure 6).
4.1.2 Editing processes

4.1.2.1 Provide and maintain standards in the CCL formalism

**D7.3UC001 StdProvide**

Satisfies WP7-01-04 ("Provide and maintain all standards of the OPENCOSS Platform expressed in the CCL formalism").

The Process Editor shall be used mainly to edit process descriptions and to establish and maintain the mapping to a standard. Such a standard will itself have to be modelled with CCL, and parsed by the Process Editor. Though the standard uses another subset of the CCL formalism than the processes, a standard defines a generic process to which actual processes must be mapped. While it uses specific objects such as ReferenceArtefact, ReferenceActivity and Role instead of Activity, Artefact and Participant, the concepts on both sides are similar, and the Process Editor can provide similar views, functionalities and an interface for the following:

- Edition of safety standards
- Edition of company processes
- Edition of project-specific processes, which may or may not be derived from company processes.

In other words, since the Process Editor must be able to parse and interpret standards to handle the mapping of processes, since the concepts of processes are analogous to those of standards they are mapped to, and since they should have similar representations, the Process Editor should also serve as a Standard Editor.
Means of compliance:

1. At least a significant part of one actual standard such as DO-178b entered in the Process Editor.
2. A dummy standard using ALL relevant concepts of the CCL model (i.e. applicable to the representation of standards) to ensure its complete coverage, should the actual standards use only a subset of it.

Since the objects defined are different, play a different role, and have separate lifecycles, they should be archived and versioned independently (e.g. stored in separate data storage). Besides, more than one process can, of course, comply with the same standard.

Means of compliance:

1. A set of processes using ALL relevant concepts of the object model (i.e. applicable to the representation of processes) to ensure its complete coverage.

This use case encompasses manual edition of processes and automatic import of existing processes created by means of other tools. Indeed, some requirements apply whatever means are used to create the process descriptions.

Means of compliance:

1. A set of processes using ALL relevant concepts of the object model (i.e. applicable to the representation of processes) to ensure its complete coverage.

Note that for some objects that have no attribute (apart from their graphic representation attributes such as their position in diagrams), it is acceptable for users to delete them and recreate them. For all other objects on which users have spent time adding information – attributes, relationships to other objects – it is mandatory to offer the possibility to paste the object elsewhere.

Note that in an editor for a formal language, this raises a number of questions not addressed here, such as:

- Propagation of “cut” and “delete” on objects embedded in the object suppressed
- Propagation of “cut” and “delete” on objects connected to suppressed objects (e.g. should connections be supressed too or should they be left with one end pending?)
- Interpretation of Paste on objects: should a Paste operation reconnect the objects after a Cut? And after a Copy?
**D7.3-005** EditCopyDiag
The Process Editor shall offer the basic features to copy all diagrams to be pasted in documents (e.g. a text processor).

This can be achieved either by the generic Copy command applied to a collection of objects, or by a dedicated “Copy Diagram” command that does not require explicit selection – which can be tedious on a large diagram.

**D7.3-006** EditSaveStd
The Process Editor shall offer the basic features Save, Save As for standards.

**D7.3-007** EditSaveProc
The Process Editor shall offer the basic features Save, Save As for processes.

This latter requirement can be used not only to save a new version of the same process without affecting the original one, but also in a different use case, when the user derives a project-specific process from the organisation process.

We do not know at this stage whether explicit traceability should be maintained between the derived project-level process and the original organisation-level process. Several options can be considered, such as:

1) When the original organisation process (or any process) is duplicated by means of Save As..., the mapping to some applicable standard(s) is carried to the new process. The new process is therefore mapped directly to the standard(s), but there is no explicit traceability to the organisation process (apart from the unchanged names, remaining similarities or dedicated conventions).

2) Project-level processes derived from an organisation-level process must be saved with a dedicated command or option, and traceability to the source objects from the organisation process is explicitly kept.

• Within this option, we have sub-options:
  a. Mapping to applicable standards is carried to the new process anyway (no need to use an indirection to check compliance with standards).
  b. Mapping to applicable standards is not carried to the new process, but inferred through the original organisation process (fragile and more complex, no added value foreseen, but mentioned because it was our initial idea).

More thinking is necessary to select the best option, but 1 seems good enough and simpler to implement.

**D7.3-008** EditPrint
The Process Editor shall offer the basic features for printing large diagrams.

Since process diagrams can be quite large, it may be useful to allow users to resize automatically the printed diagram to fit in a given number of pages (like Excel does), to provide visual indication of page breaks in the Editor, to provide print preview, etc.

**D7.3-009** EditExportDiag
The Process Editor shall offer the basic features to export all diagrams or selected diagrams in common formats (png and svg at least).

An easier way to include process diagrams in documents than multiple copy-paste operations is to export all diagrams at once, with a single operation. These can then be included in documents with a live links and updated automatically.

**Large and complex diagrams**
Process diagrams and standards can be quite large (large number of objects such as artefacts and activities in particular), and quite complex (complex relationships between objects, complex rules to comply with). Thus, it will be necessary to provide features that facilitate the edition of large processes as well as the navigation.
The Process Editor shall make it possible to split a process or standard into multiple diagrams, while maintaining automatically their consistency (particularly for objects visible in more than one diagram).

Multiple views will allow the splitting of large processes; each activity, artefact or role shall appear in as many diagrams as needed. The Editor shall ensure that all views are kept consistent (all occurrences of the same object, while they may display a different subset of the information on the object, are in line with the shared object definition (name, attributes, etc.).

It would be cumbersome for users to navigate through all diagrams to spot a given object, and very difficult to synthesise partial information from all views to get a complete picture of an object (e.g. all the objects it is related to). Thus, the Editor should provide a dedicated view or dedicated functions that make it easy:

- To find an object by one or more of its characteristics (e.g. looking for an Artefact containing “Test Plan” in its description)
- To visualize all the information about this object – name, attributes, relationships to other objects – without having to open a particular view (e.g. for an artefact, seeing all the activities that contribute to produce it or require it)
- To modify this information.

The Process Editor shall make it possible to find objects directly from their characteristics (name, type, ...) and to modify them.

4.1.2.2 Import a process definition from external tools

Importing a process definition requires that some translation mechanism is applied to a formalism $S$ to obtain an equivalent definition (if possible) in the target formalism, i.e. CCL. Currently, there is no widely spread standard for process definitions (BPMN may be a valid candidate, but it is seldom seen in actual organisations as of today), and no commonly adopted standard to exchange data between competing tools.

Thus, we do not propose here to focus on one particular format, but suggest that the Process Editor provides an API that can be used to create, retrieve and modify all objects of a CCL model, either in batch tools performing conversion to CCL, or in plug-ins to be called from an “import...” command of the Process Editor.

The Process Editor shall provide an API to create, retrieve and modify CCL objects and to save process descriptions in CCL formalism.

The Process Editor shall provide means to develop plug-ins available from the Process Editor.

Means of compliance:

1. A demonstrator of the Import feature for one particular format, developed using the API, and available as a plug-in accessible from the editor.

1 Among the tools commonly used to define processes, we have seen MEGA, ARIS, EPF (public domain) and Rational Method Composer (IBM), Casewise Modeler, each with their own format, some using a database and others using flat files with more or less proprietary formats... and many proprietary tools.
2. A set of processes to be imported in the selected format using ALL relevant concepts of the object model (i.e. applicable to the representation of processes) to ensure its complete coverage.

### 4.1.3 Mapping processes with standards or other processes

#### 4.1.3.1 Map processes with the relevant standard

**D7.3UC005** ProMap

Mapping can occur at various levels:

1. between a process (organisation- or process-level) and one or more standards, either while it is being developed, of afterwards, to analyse or demonstrate compliance,
2. between two organisation-level processes or two standards, to analyse equivalence or to derive a new one from an existing one,
3. between a project-level process and the organisation-level process it is derived from.

What is more important from a “use case perspective”, because it has more impact on the detailed mapping activities, is the way the mapping is done due to history and circumstances.

1. A **new process or standard** (organisation-level or project-level) is written from scratch and it is known from the beginning that it must comply with an existing standard: traceability can easily be established on-the-fly, each item of the new model immediately mapped to its counterparts — since the item is possibly created specially to cover a requirement, it is straightforward to do the mapping at creation time.

2. A variant: A **new process or standard** (organisation-level or project-level) is derived from an existing process or standard: not only can the mapping be done on the fly, but most of the original links can be created automatically, then updated (e.g. weakened to “partial” or “noMap”, or even deleted) during the derivation process.

3. Multiple **pre-existing reference frameworks** (several processes, standards, a mix of both) must be mapped afterwards.

All three scenarios are likely to happen in real life, and each requires specific features to ensure an efficient mapping.

#### 4.1.3.1.1 Map processes on-the-fly

**D7.3UC006** ProMapFly

This use case applies when:

- The new process (or standard) is created from scratch
- The reference applicable standard(s) are known upfront

**Browsing the target standard**

When both conditions are met, it is possible for the author to map each item of the new model to its counterpart(s) as it is created. This is the ideal situation, because the effort for mapping depends in a linear way on the number of items to be created. In other words, if the model to be created contain \( n \) items (artefacts, activities, etc.), then the effort required to map these to the standard will be \( n \times k \), where \( k \) is the effort required to map one item to the standard.

What makes the task relatively simple is that, in this ideal scenario, some objects of the new model are created precisely to address a requirement of the applicable standard. It is then very easy for the author to document the mapping at creation time.

**D7.3-014** MapStdBrowser

The Process Editor shall provide means to browse the target standard (the model to which the model currently edited is to be mapped), and to find target items by name, with wildcards, and/or type (artefact, activity, etc.)
Smart search

However, \( k \) is itself a function of the target standard: it takes more time to establish mapping to a large or complex standard than to a simple one, unless some assistance can be provided to users to browse the target standard efficiently. In other words, \( k = K \times m \), where \( K \) is a constant time and \( m \) designates the size of the standard to be covered. Since the model to be developed is itself impacted by the size of applicable standards, mapping typically evolves as the square of the size of the model.

To facilitate the mapping activity, we suggest a “smart search” feature in the target standard, based on links already established.

Let \( S \) be the source model being edited (e.g. a process), and \( T \) the target model to which it must be mapped. \( i_s \) and \( i_T \) are respectively items of \( S \) and \( T \). A mapping relationship between these two items is noted as \( i_T \in \mathcal{M}(i_s) \).

We note \( \mathcal{L}(i_s, r) \) (respectively \( \mathcal{L}(i_T, r) \)) the set of elements that have a relationship \( r \) to \( i_s \) within \( S \) (respectively to \( i_T \) within \( T \)). For instance, if \( i_s \) is an activity, and \( r \) is the relationship \textit{produces}, then \( \mathcal{L}(i_s, \text{produces}) \) is the set of artefacts \textit{produced} by activity \( i_s \).

The set of elements of \( T \) proposed by the smart mapping is defined as follows:

For a given element \( i_s \) of \( S \),

- for all \( i_s \) such that \( i_s \in \mathcal{L}(i_s, r) \),
- for all elements \( i_T \) of \( \mathcal{M}(i_s) \),

the set elements proposed for mapping is the union of all \( \mathcal{L}(i_T, r) \).

For example, if a process activity is mapped to one or more activities in a standard, for an artefact of the process activity, the \textit{proposed} elements will include all artefacts produced by mapped activities of the target standard. It is important here to understand that \textit{no automatic mapping is done}, and that

- the \textit{proposed} elements for mapping are still to be mapped explicitly by the author; the point is only that they are much easier to find in a much smaller collection of elements;
- it is still possible to map elements that are not proposed by the smart mapping, which must remain an optional feature (statistically, many can probably be found with the smart mapping, but not all).

Consider the following example:

![Diagram of Standard and Process](image-url)

**Figure 7. Example of a Standard and a Process to be mapped**
Once we have established, for example, a partial mapping between “Plan tests” and “Establish Verification Strategy”, then:

- Following the “executed by” link on both sides, “Project Leader” and “Test Manager” are candidates for being mapped to “verification Manager”
- Following the “produces” link on both sides, “Test Strategy” and “Test Plan” are candidates for being mapped to “verification Plan”.
- Once it is confirmed by the author that there is some mapping (e.g. partial) between “Test Strategy” and “Verification Plan”, following backwards the “produces” link from “verification Plan”, the editor can identify that “Plan Tests” can also be proposed for a mapping to “Define Verification Criteria”.

**MapSmartBrowser**
The Process Editor shall be able to make relevant proposals for mapping a process to a standard based on links already established.

This will dramatically reduce the value of $k$, which will become close to a constant value, not depending on the size of the target model. Thus the complexity of mapping will depend mainly on the size of the model to be developed, and less on the standard to be covered. It is closer to a linear function of $n$, instead of a parabolic one evolving as $n^2$.

In a Graphic User Interface, the smart mapping feature could be activated or deactivated quickly with a checkbox. When activated, the list on elements to be mapped would be reduced to distant elements of the “same” type in the target model and with an indirect relationship.

$$\text{selectedElement} \rightarrow \text{relatedElement} \rightarrow \text{mappedDistantElement} \leftarrow \text{relatedDistantElement}$$

**Provide visibility on (non-)coverage**

It is essential for the author of the new model to identify easily which parts of the reference standard or process are not covered.

Thus, the Process Editor should facilitate the identification of elements not covered yet in the target model, or partially covered (map with value “partial” or “nomap”, for which covered can be addressed by other elements of the model being edited). This can be done, for example, by a filtered list of target elements, with an attribute showing whether there are no mapping indications at all, or whether there is insufficient mapping links (partial or no map only), and access to those elements.

**MapNotCovered**
The Process Editor shall allow easy identification of elements of the target model not covered (yet) by the model being edited.

It should also be possible for the author to indicate that a given item will not be covered (or not better than it is currently), e.g. because it is out of scope or not applicable in a particular context (e.g. specific software-related requirements not applicable to hardware). This way, these items will no longer be displayed in the list of items “to be covered”.

**MapNotCoveredChk**
The Process Editor shall allow the author to mark elements of the target model not to be covered by the model being edited.

4.1.3.1.2 Derive a process from another one

**ProMapDer**
This scenario will occur when:
• A variant of an existing standard is created (a new version, an extension, a specialization for a specific domain, or a variant for a different domain)
• A process is created from scratch specifically to address a given standard; it is possible to create the process as a copy, then to adapt the copy
• A project-specific process is derived from a company standard.

In this scenario, the Process Editor can help the user with a few dedicated features:
• A “Copy and map” feature, similar to a “Save As…”, but with each duplicate object initially mapped to its source
• A new status for mapping links, e.g. “ToReview”, indicating that these links were created automatically and need to be explicitly qualified (Full, Partial, NoMap).
• Specific edition features to transfer mapping links from one object to another, to merge or split existing objects while preserving the mapping links.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7.3-018 MapCopyAndMap</td>
<td>The Process Editor shall propose a “Copy and Map” feature, to create a process or standard by copying an existing process or standard, in which each duplicate object is linked back to the duplicated object.</td>
<td></td>
</tr>
<tr>
<td>D7.3-019 MapMarkAutoLinks</td>
<td>Mapping links created automatically during a “Copy and map” operation will optionally have the default status “ToReview”.</td>
<td></td>
</tr>
</tbody>
</table>

4.1.3.1.3 Map existing processes and standards

**D7.3UC008 ProMapExist**

This scenario may be the most likely to happen. A company wants to assess that processes already in place comply with an applicable standard or with customer applicable process requirements.

The effort required to do such a mapping typically depends on the size of both models. Thus, the Process Editor should provide appropriate means to facilitate the mapping activity.

A dedicated view, showing both models to be mapped, will make it easier to establish the mapping:
• By allowing to do the mapping from both sides (i.e. either by selecting a source object to be mapped, or a target object to be covered)
• By optionally filtering “relevant” candidate objects for mapping to a given source or target object
• By providing visual feedback on the coverage status of objects on both sides

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7.3-020 MapDedicatedView</td>
<td>A dedicated view, showing both models to be mapped, will provide features to establish and consult the mapping in either direction.</td>
<td></td>
</tr>
</tbody>
</table>

All the other features already exposed previously will be essential to facilitate the mapping activity in this scenario, particularly the following (reminder):

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7.3-014 MapStdBrowser</td>
<td>The Process Editor shall provide means to browse the target standard (the model to which the model currently edited is to be mapped), and to find target items by name, with wildcards, and/or type (artefact, activity, etc.)</td>
<td></td>
</tr>
<tr>
<td>D7.3-015 MapSmartBrowser</td>
<td>The Process Editor shall be able to make relevant proposals for mapping a process to a standard based on links already established.</td>
<td></td>
</tr>
<tr>
<td>D7.3-016 MapNotCovered</td>
<td>The Process Editor shall allow easy identification of elements of the target model not covered (yet) by the model being edited.</td>
<td></td>
</tr>
<tr>
<td>D7.3-017 MapNotCoveredChk</td>
<td>The Process Editor shall allow the author to mark elements of the target model not to be covered by the model being edited.</td>
<td></td>
</tr>
</tbody>
</table>
4.1.3.2 Keep traceability between processes and the relevant standard

This use case has several triggers:

- A new version of a reference (target) process or standard is issued, and it is necessary to reassess existing processes mapped to the previous issue against the new one.
- A new version of a process (or standard) is issued, and it is necessary to check that the new issue still complies with applicable standards.

The two situations are quite different:

- In the latter case, the author can check, continuously, while he or she is modifying the process, what is the impact on the mapping (if an object is deleted or modified, what are the reference objects it is mapped to, and what needs to be done to maintain coverage of these objects).
- In the first case, the standard evolves, a new version is issued, possibly with a list of changes, but the model for this new issue is not available. As it is eventually produced, the links to processes will have to be migrated.

4.1.3.2.1 New issue of a reference model: impact analysis and transfer of mapping

Ideally, the Process Editor would allow the user to open and visualise simultaneously both versions of the target standard and the model to be checked or updated — already mapped to previous version.

The activity mainly consists in:

- Spotting differences between the two versions of the target standard (either using some “compare” feature of the editor, or using a documented comparison)
- For each difference (object added, deleted or modified) checking whether this impacts the mapping and how
- Transferring the mapping when it is still relevant by redirecting (or duplicating) validated mapping links to the new version (possibly with a different status)
- Updating the source model to address new coverage needs caused by the modifications of the target standard (new or modified requirements, etc.), identified in the previous steps

We assume that the new issue of the standard will generally not be available directly as a CCL model, and that a list of changes helps users of the standard propagating changes where applicable.

We also assume that while some items may be identified in a unique and constant way (e.g. requirements in a standard sometimes have a unique ID that is constant over successive versions and never reassigned), many items are not identified in a constant way; for example, paragraph numbers are used to identify topics, and these may change from one version to another, so the same number may designate a different topic, or the same topic change number. For example, between CMMI 1.1, 1.2 and 1.3, the same practice may change number and the same number can designate a different practice.

Existence of references (unique IDs) kept constant across successive versions

When some constant ID does exist, the Process Editor should allow the capture of this ID as an attribute of each item, to facilitate the analysis of differences and the transfer of mapping links between versions. For example, new objects in the standard (to be covered by a new version of the process), or objects deleted from the standard (possibly replaced by some other objects to which previous links should be transferred after analysis).
Absence of constant references across successive versions

If such constant references are not available, or not for all objects, the user shall use (or produce) a detailed list of changes in the standard. Using “Copy and Map” (see D7.3-018 and D7.3-018), the user can obtain a new CLL description, to be updated, but already linked to the process and with all links keeping their original value (not marked as “ToReview”).

The user shall then update the CLL description to reflect the changes to the target standard. When the user does so, all mapping links connected to modified objects will be marked as “ToReview”, so that any previous mapping information associated to these objects is considered suspicious.

D7.3-023  MaintSuspicious  A “tracking mode” of the Process Editor will set the status of mapping links to “ToReview” when either of the connected objects is modified.

See also D7.3-018 and D7.3-023.

4.1.3.2.2 New issue of a process mapped to a stable reference

D7.3UC011  ProTracNewP

This situation will arise more frequently, since company processes may constantly evolve. While the author makes changes to the process, he should simultaneously make changes to its CLL model. This should preferably be done on a copy, e.g. using “Copy and Map”. The user shall then update the CLL model to reflect the changes to the target standard. When the user does so, all mapping links connected to modified objects will be marked as “ToReview”, so that any previous mapping information associated to these objects is considered suspicious. See D7.3-018 and D7.3-023.

4.1.3.3 Check coverage of relevant standard

D7.3UC012  ProChkCov

Once a process is created or modified, Quality Assurance will ensure that it actually covers applicable standards (or higher-level processes). To help establish a diagnostic, the tool should:

- Indicate whether some parts of the standard are not covered, intentionally (e.g. because they are not applicable) or not.
- Indicate which parts were intentionally not covered (e.g. not applicable in the context), so that QA can review them and confirm they are not applicable.
- Indicate which other parts were not covered, so that QA can estimate what corrective actions are required.

The author creating or updating the process will need similar information to estimate the remaining work and to spot the areas of the standard still to be covered.

This is covered by the following requirements, already identified in previous use cases:

D7.3-016  MapNotCovered  The Process Editor shall allow easy identification of elements of the target model not covered (yet) by the model being edited.

D7.3-017  MapNotCoveredChk  The Process Editor shall allow the author to mark elements of the target model not to be covered by the model being edited.

4.1.3.4 Check relevance of the mapping

D7.3UC013  ProChkRel
A Safety Assessor can be involved in the review of the mapping, not just to check the coverage of the applicable standard — which can be done without too deep a knowledge of these standards. The Safety assessor should more specifically:

- Check whether the mapping is relevant (objects of the process are mapped to appropriate targets in the applicable standard)
- Check that the values (Full, Partial, NoMap) are correct
- Check justification for mapped elements
- Check justification for elements declared out of scope.

This will require the same feature as the previous use case but additionally, it is essential a view of the Process Editor that supports the review process:

- By allowing systematic browsing of elements not covered
- By allowing systematic browsing of all mapping links and justification
- By presenting for each mapping link all necessary information about mapped elements, side by side.

| D7.3-024 MaintMapView | A view shall facilitate the review of the mapping by displaying simultaneously all relevant information about each mapping link and linked objects, and by facilitating the systematic visit of all mapping links, as well as all objects not mapped. |

### 4.2 Process Execution and Compliance Monitoring Manager

This section addresses the following requirements from D7.2:

- **WP7-02-01** *Provide a process-viewer* — OPENCOSS platform should provide a process-viewer which visualizes the status of each process item
- **WP7-02-02** *Provide API to automatically update the status of process execution* — OPENCOSS platform should provide API to automatically update the status of execution of the process actions
- **WP7-02-03** *Provide means to manually update the status of process execution* — OPENCOSS platform should provide means to manually update the status of execution of the process actions
- **WP7-02-04** *Deduce basic process information from work products* — OPENCOSS platform should be able to deduce basic process information from work products

We assume in the rest of this section that a Process Viewer is available. The Process Viewer is possibly available as an independent tool, or just as a view of the Process Editor — that has no strong impact on the requirements.
4.2.1 Use cases overall presentation

![Diagram of use cases for monitoring process execution]

**Figure 8. Use cases for the monitoring of process execution**

4.2.2 Initiate Project

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVinit</td>
<td>The Process Viewer shall allow the user to create a project execution.</td>
</tr>
<tr>
<td>PVinitStd</td>
<td>The Process Viewer shall allow the user to attach a project execution to its applicable processes or standards.</td>
</tr>
</tbody>
</table>

**D7.3UC021 MonInitPrj**
Before monitoring process execution, the user must initiate the project.

- Create a project and provide basic information (name, ID, ...)
- Identify what is the reference process or standard (a standard may be directly applicable, or indirectly via the company reference process or project specific process). Note that several standards can be applicable in some cases (e.g. customer process and domain standard).
- Instantiate project data — this part is more complex and will be detailed hereafter.

Instantiation of project data comprises for example:

- Assigning roles to actual people
- Indicating the actual artefacts to be produced
- Instantiating activities (a unique activity of a process can have multiples instances in a given project, acting on different artefacts, or because of iterations).

This list is far from complete, and we may need to revisit this heavily in a future version of the document. These three points, however, will serve as an example of what and how can or should be done at project level.

**4.2.2.1 Assigning roles to actual people**

It should be possible to assign roles to actual people.

- It will sometimes be necessary to demonstrate that responsibilities are actually assigned as part of the certification process
- Roles can also be used for certain certification-related features that could be supported by the process viewer in future versions (authorizations to perform certain operations, access rights to data, electronic signature)
In the simplest case, a role is assigned to one person only. In other cases, a role designates a set of people (e.g. the Change Control Board, Steering Committee). Last, a single role may have multiple instances. For example, “Verifier” may involve multiple persons, each individually responsible for verifying a different artefact.³

| D7.3-027 | PVinitPeople | The Process Viewer shall allow the registration of people involved in safety-related activities of the project. |
| D7.3-028 | PVinitGroups | The Process Viewer shall allow the registration of groups of people - one person can belong to one or more groups, or no group at all. |
| D7.3-029 | PVinitRoleAssign | The Process Viewer shall allow the assignement of roles to groups (acting collectively as one person) and to individual people. |

### 4.2.2.2 Indicating the actual artefacts to be produced

To initiate the project, the user will have to indicate the actual artefacts of the project (existing or to be produced). Indeed, these artefacts are the key evidence — with proper argumentation — that the requirements of the applicable standards are met.

In some cases, indicating artefacts can be easy: when the artefact consists in a single document, the user just needs to indicate the name and/or location of the file.

In some other cases, the designation can be more difficult, because the artefact is a physical object, or a record in a database, for example. Identification means should be generic enough to handle any situation (e.g. accept free text or custom attributes with arbitrary name and type), while offering basic functions for most common situations (e.g. a file browser to locate files, ability to open an internet navigator to follow an URL).

For example, it the process requires that a Safety Plan is issued, the user could designate ProjName12345_SP.docx as its actual implementation for the project ProjName.

| D7.3-030 | PVInitArtSingle | The Process Viewer shall allow the user to indicate the location of actual artefacts implementing the artefacts required by the standard or process under execution. This location can be provided in multiple ways (file location, URL, record ID, serial number…). |

For some other work products, it may be more tedious, for example because of the number of items. Consider individual test cases. Each may be seen as an individual artefact with a state of its own (reviewed or not, executed or not, passed or failed…), and requirements applicable to each item.

There should be means at configuration time to populate the list of artefacts automatically. Here are some examples:

- All files whose name match a regular expression in a given directory
- All records in a database returned by a certain query
- All physical items (parts) designated by nodes or leaves of a decomposition tree in a PLM tool

³ Also, a role may be assigned to several persons in the case it designates in a global manner several “sub-roles” which cooperate to execute a global activity, which has not been split up into several sub-activities. E.g. the activity “Build a house” could have a “[1..*]worker” executing role, which could be assigned to several persons. Actually multiple different roles are involved, but they are merged into a single “worker” role with multiplicity [1..*], because the activity is complex, non-atomic. The sub-roles could be distinguished if the activity were split into several sub-activities.
The simplest way to accommodate all situations is to provide a generic API to allow the automatic creation of objects in the Process Viewer — either at initialisation time, or on the fly, while the process is being executed. This API could be used directly from other tools, or via a scripting language, such as Python, TCL, Groovy, or Ruby.

<table>
<thead>
<tr>
<th>D7.3-031</th>
<th>PVAPI</th>
<th>The Process Viewer shall provide an API to allow the creation or modification of project objects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7.3-032</td>
<td>PVscript</td>
<td>The Process Viewer API shall also be accessible via a scripting language to automate certain tasks such as object creation, object updates, deduction of properties, extraction of data...</td>
</tr>
</tbody>
</table>

### 4.2.2.3 Instantiating activities

The effectiveness of activities can be seen and demonstrated mainly (if not only) through their outputs, i.e. the artefacts they produced, and argumentations (themselves the output of other safety-related activities). Tracking the execution of activities is usually done by dedicated tools such as project management tools, sometimes by process-supporting tools. Thus, it is not essential to replicate process monitoring in the Process Viewer, and it would be very cumbersome for users to track activities twice — once in the project management tool and once in the Process Viewer. However, it may be useful to incorporate some monitoring-related features:

- Coupling the Process Viewer to a Project Management tool can allow automatic update of safety-related activities status.
- Marking an activity as “complete” may trigger the collection of artefacts or the verification of specific requirements.
- Inconsistencies between the status of an activity declared as “complete” and the availability of expected deliverables or their status can be highlighted on the fly.

These points will be elaborated further in the §4.2.3, Update process status.

As far as initiating the project is concerned, we suppose that the requirement D7.3-031-PVAPI already identified is sufficient to address the coupling of the Process Viewer with external tool involved in Project Monitoring or execution of activities.

### 4.2.3 Update process status

<table>
<thead>
<tr>
<th>D7.3UC014</th>
<th>MonPrStat</th>
</tr>
</thead>
</table>

Updating process status can be done in two ways: manually or automatically.

- Manually: various actors of the process update data in the Process Viewer, or a safety-dedicated user updates data in the Process Viewer on their behalf.
- Automatically: various tools involved in the execution of the process can use the Process Viewer API to update data in it — possibly partially; alternatively, some scripts are run from the Process Viewer and access external data or trigger the API of external tools.

#### 4.2.3.1 Update process status manually

<table>
<thead>
<tr>
<th>D7.3UC015</th>
<th>MonPrStatMan</th>
</tr>
</thead>
</table>

Various actors can be involved in safety-related activities and, as such, could have access to the Process Viewer to update or read safety-related data.

In some cases, such update operations should be done only by selected people with appropriate level of authorisation. At this stage of the document, we do not specify explicit requirements related to authorisations, access control lists for data, identification, or electronic signature.

| D7.3-034 | PVmanUpdate | The Process Viewer shall allow authorised users to update the status and other properties of activities and artefacts. |
Having done so, the user may have introduced inconsistencies. For example, an activity could be declared as complete, while so of the output artefacts are missing, or while some artefacts are not in the expected state (e.g. “verified”), or have not demonstrated compliance with certain requirements.

The Process Viewer should highlight such inconsistencies.

An alternative would be that inconsistencies are simply not allowed — e.g. you cannot indicate that an activity is complete until all of its deliverables are available — but this is much more constraining for users, as well as for external tools updating a part of the information via the API.

| D7.3-035  | PVhighlightIncons | The Process Viewer shall highlight inconsistencies resulting from updates, particularly between the status of activities and resulting artefacts. |

After automatic updates (see below), inconsistencies may also result from the fact that external tools will provide only a part of the information. For example, some tools could automatically compute metrics on software, but will not be aware of the decision to accept values out of boundaries or the rationale for doing so.

### 4.2.3.2 Update process status automatically

| D7.3UC016 | MonPrStatAut |

Various kinds of tools could automatically help updating the data in the Process Viewer. Some examples follow.

**Test tools** — Test management tools (e.g. Quality Center) and/or automatic test execution tools could automatically update the status of some artefacts in the Process Viewer – has it been fully tested? Did some critical tests fail? Etc.

**Process Execution tools and Project Management tools** — Such tools could update the status of activities and, possibly, of some resulting artefacts.

**Requirements management tool** — These tools are often used to monitor a process, e.g. to check the maturity of requirements, but also their approval by relevant stakeholders, or their verification status.

**Configuration Management, ALM or PLM tools** — These tools are often used to record the status of artefacts, control the completeness of a baseline and eventually the overall status of a baseline (availability and status of all components, status of the integration, etc.)

Our intent here is not to list all the possible tools that could be interfaced, but rather to show that almost all kinds of tools involved in the execution of one safety-related activity are worth considering for interfacing. The major criterion is the amount of data to transfer (volume and density of information, frequency).

Note that information can be incomplete, as stated above, and may have to be checked or even completed by hand afterwards.

| D7.3-037  | PVautoUpdateMk | It shall be possible to indicate that data of specific objects in the Process Viewer is (potentially) incomplete and/or to be reviewed - in particular after automatic update or import from external tools. |

### 4.2.3.2.1 Obtain data from external tools

| D7.3UC017 | MonAPI |

Data from external tools can be made available to the Process Viewer in several ways.

- Using the Process Viewer API, external tools can directly push data to the tool: create new objects, update existing objects.
• By publishing data in a format that can be read directly by the Process Viewer, or by means of a
dedicated tool API — which requires that the Process Viewer itself can be scripted, or provides an
API accessible to a scripting language (e.g. TCL, Python or similar).

| D7.3-038 | PVautoUpdateAPI | An API shall allow external tools to create objects or to update
existing objects of the Process Viewer. |
| D7.3-039 | PVautoUpdateScript | The API of the Process Viewer shall be available through a
scripting language (such as TCL, Python or similar). |

4.2.3.2.2 Compute updated status

Once data has been updated in the process Viewer (whether it was done manually, automatically or in
some hybrid way), it may be necessary to compute and update related data, possibly to fetch additional
data from other sources, and eventually to spot and indicate potential inconsistencies.
All this can be done by the requirements already identified (API, scripting language, and possibility to mark
data that is inconsistent, incomplete or to be reviewed).

4.2.4 Monitor process execution

D7.3UC019 MonPrView

Monitoring process execution implies comparing achievements of the project against initial targets:
deliverables, on time, meeting requirements.
We assume that project management as such is done with dedicated tools. The intent of the Process
Viewer is only to provide visibility on safety-related activities. However, safety assurance is in itself a part of
the project, with its deliverables, activities and roles: as such, it can be handled by “ordinary” project
management tools.
Thus, the question is: are there any specific needs, related to safety assurance activities, which are not
already covered by “classic” project management tools? Of course, the Process Viewer may very well offer
these functions as well, but there is no urgency to duplicate such functions and no need to specify them
here.
We therefore exclude from our scope classic monitoring of activities (have they started or are they
completed on time? What is their current progress?), or the simple monitoring of artefacts (are they
available at the expected date and with expected status).

4.2.4.1 Monitoring the safety status of the product and components

We propose that the Process Viewer gives a global view of the complete PBS (Product Breakdown
Structure), with a visual indication of the status of each node with respect to safety assurance. For
example, a colour code could indicate whether expected evidence attached to each node is available or
not. By selecting a node, the user would have more detailed information on what is still missing; for
example: what evidence is still missing? Which requirements are not met or demonstrated yet?

| D7.3-040 | PVPBS | The Process Viewer shall give an understanding of the “safety status” of the
final product via a Product Breakdown Structure view, indicating the status of
each node with respect to safety and providing details on demand (e.g. missing
evidence attached, requirements not demonstrated...). |

4.2.4.2 Monitoring safety requirements and pieces of evidence

The applicable standards enumerate requirements regarding the product and its components, the process
develop to develop and to build them, artefacts used by or resulting from this process, and evidence regarding
compliance with these requirements.
The process (organisation-wide process or project-specific process) complies with the standard, and
guarantees that if the defined process is actually implemented and applied, then the project will also
comply with these requirements.
Several cases can be considered:

- Compliance with a requirement is checked outside of the tool (e.g., by inspection, review, or using other tools) and only the compliance status is recorded by hand in the Process Viewer.
- Compliance with a requirement is automatically updated by the Process Viewer, using mechanisms seen in §4.2.3.2, Update process status automatically: scripts are run to automatically check compliance with the requirement, given the availability of evidence and status of artefacts.

<table>
<thead>
<tr>
<th>Specification of the compliance-aware service infrastructure D7.3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The Process Viewer shall allow recording of the status of all requirements extracted from applicable standards, either by hand, or via the API.</td>
<td></td>
</tr>
</tbody>
</table>

To facilitate monitoring, the process Viewer shall provide a synthetic view of requirements not met yet and pieces of evidence still missing, and allow to investigate the details. For example:

- A view will show progress, e.g., the number of requirements met (globally and by criticality, with focus on critical requirements), and pieces of evidence gathered or missing.
- The user, by designating a category (e.g., critical requirements not met yet) will have access to the detailed list of items and, for each, be given the related information extracted from the standard (related requirements, artefacts, activities, etc.)

<table>
<thead>
<tr>
<th>Specification of the compliance-aware service infrastructure D7.3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The Process Viewer shall provide a synthetic representation of the progress of safety-related activities, e.g., evidence gathered vs. evidence required, safety requirements status, etc.</td>
<td></td>
</tr>
<tr>
<td>The Process Viewer shall give access to the detailed list of items meeting certain criteria from the synthetic view (e.g., evidence missing, critical requirements not met…)</td>
<td></td>
</tr>
<tr>
<td>The Process Viewer, when an item is selected in the progress view, shall provide related guidance.</td>
<td></td>
</tr>
</tbody>
</table>

4.2.5 Specify formal process rules

MonRules

Requirements are part of the applicable standard and define what exactly must be verified to ensure compliance with the standard.

By “formal rule”, we mean something close to those requirements, but different in several ways:

- We insist on “formal” to express that these rules are unambiguous and, ideally, can be expressed in a formal language that allows automatic computation or execution of the rule.
- We choose to use “rule” rather than “requirement” because, while the rules are mainly intended to compute whether requirements are met or not, they could sometimes or even serve other purposes, such as:
  - check some properties related to a requirement without actually checking all aspects of the requirement,
  - automatically update items related to a modified item,
  - notify automatically some people involved in safety of some important events,
  - check consistency after some modifications,
  - preventing some actions that would result in the violation of a rule (e.g., marking a requirement as satisfied without providing related evidence)

Two requirements were already identified and could serve this purpose:

<table>
<thead>
<tr>
<th>Specification of the compliance-aware service infrastructure D7.3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>An API shall allow external tools to create objects or to update existing objects of the Process Viewer.</td>
<td></td>
</tr>
<tr>
<td>The API of the Process Viewer shall be available through a scripting language (such as TCL, Python or similar).</td>
<td></td>
</tr>
</tbody>
</table>

However, if such scripts must be triggered at specific times, like before or after a certain action, other mechanisms should be provided, such as hooks attached to operations (similar to those available in...
configuration management tools such as Subversion or Git, and the like), or callbacks attached to certain types of objects and triggered when they are modified.

| D7.3-045 | PVhooks | The Process Viewer API will provide means to trigger automatically scripts on certain events, either to trigger additional actions, or to make certain checks to allow the action or not. |
5 Candidate Technologies

5.1 Candidate Process Management tools to interface with the OPENCOSS platform

5.1.1 Atego Process Director & Atego Process Consumer

5.1.1.1 Major functionalities and benefits

Process management often relies on a document-based approach leading to large printed documents or complex intranet sites difficult to:

• Keep consistent
• Author
• Find relevant content in.

This often results in some decoupling between operations and defined processes, endangering project and quality managers control and monitoring.

**Atego Process director** solves these issues with:

• A thin client web application allowing users, authors and managers to easily access content
• A highly scalable, multi-user repository allowing to scale from SMEs to industrial groups
• A method-agnostic tool allowing versatile process:
  o Control
  o Repeatability
  o Estimation
  o Measurement
  o Improvement

Through the use of **Atego Process Perspective**

![Figure 9 Process director libraries](image-url)
5.1.1.1 Speed up authoring process

Out of the box, Atego Process Director provides best industry practices such as (not exhaustive):

- OOSEM
- Waterfall
- Spiral
- Staged
- To Budget
- DO178
- Etc.

You may also import existing company processes, or author new ones, mixing efficiently all these approaches.

To efficiently author new processes, in addition to a built-in process editor, you may use Word or any MetaWeblog API compliant tool (e.g. Microsoft Live Writer).

The graphical editor allows working easily on activities precedence and hierarchical decomposition from your web browser, should it be on a PC, tablet, or mobile phone.

5.1.1.2 Manage your process assets

During the authoring lifecycle, your processes are versioned and, when relevant, frozen for publication. Therefore authors may improve their processes while the previous baseline is currently used in operation. In addition, access rights are handled to provide relevant permissions to users logging through their web browser to the Atego Process Director server.
5.1.1.3 Speed process definition for a specific project

Defining the process of a specific project is a simple tailoring of the company process (master process) where you can:

- Define how iterations are required for a given process
- Activate or not some optional processes
- Add project specific processes if needed and not capitalized at company level

The tailoring is done through a question-and-answer wizard, that provides actual agility to project managers.

![Figure 12 Question and answer wizard for tailoring](image)

5.1.1.4 Prepare execution of the project

Execution of the project-specific process is supported by scheduling and assigning tasks to team members (tasks are instantiations of master project activities). Team members are informed by mail regarding allocated tasks and related initial schedule.

This assignment and related schedule can be exported to a Microsoft project planning allowing project manager and teams to share not only a common vision of the processes but also a common view of process execution timeframe.

![Figure 13 Scheduling tasks and exporting Gantt](image)
5.1.1.1.5 Execute the project

Project team members have access to a thin web client which allows them to quickly know:

- What role(s) and related task they are assigned
- Which tasks can be started
- What deliverables are relevant for them
- Discussions they are involved in
- Notifications of project events

They are presented process content in a friendly way, and they may interact with the project manager and process authors to handle any discussion regarding activity content or activity relevance within the project.

Team members will be able to report their progress on a given task (start, completion, progress, success or failure status)
5.1.1.6 Monitor and control the project

The project manager has a quick view on project progress through color codes:

- Yellow: running
- Green: performed OK
- Red: failed

Metrics will be recorded allowing further process improvement and reports may be generated.
5.1.1.2 Atego Process Director concepts and mapping with OPENCOSS

From a conceptual point of view, Atego Process Director & Atego Process Consumer handle four different levels

- Definition of the concepts to be used for process modelling
- Definition of master processes that can be considered company assets
- Definition of a project-specific process
- Monitoring of project process execution

Atego Process Director uses 2 library kinds

- **Process libraries**: hosting the standards (DO178, OOSEM…) or company generic processes
- **Project libraries**: hosting derived project processes used for operation. Multiple projects may derive from the same process library.

<table>
<thead>
<tr>
<th>Level</th>
<th>Atego Process Director module</th>
<th>Atego Process Director concept</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of the concepts to be used for process modelling</td>
<td>Process library</td>
<td>Process library types</td>
<td>Atego Process Director is standard agnostic and relies on a versatile internal model</td>
</tr>
<tr>
<td>Definition of master processes that can be considered as company assets</td>
<td>Process library</td>
<td>Master project = Process library definitions</td>
<td>-</td>
</tr>
<tr>
<td>Definition of a project-specific process</td>
<td>Project library</td>
<td>Project tasks are instantiated from master project activities</td>
<td>Project-specific tasks can be added if needed</td>
</tr>
<tr>
<td>Monitoring of project process execution</td>
<td>Project library</td>
<td>Task progress and metadata</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1 Libraries and Atego Process Director concepts

Taking into account OPENCOSS levels of modelling presented below, implementation of OPENCOSS concepts with Atego Process Director is addressed as follows:
### Figure 20 OPENCOSS levels of modelling

<table>
<thead>
<tr>
<th>Level</th>
<th>OPENCOSS</th>
<th>Atego Process Director</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of the concepts to be used for process modelling</td>
<td>CCL Reference assurance project metamodel at level 0 for level 1/1b</td>
<td>Process library types merging both metamodels</td>
<td>Atego Process Director allows a direct instantiation of process into task requiring alignment of Level 1/1b/2 (e.g. between CCL <code>RefActivity</code> and CCL <code>Activity</code>)</td>
</tr>
<tr>
<td></td>
<td>CCL Process metamodel at level 0 for level 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of master processes that can be considered as company assets</td>
<td>Standards at Level 1</td>
<td>Process library per standard</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>company processes at Level 1b</td>
<td>Process library for company</td>
<td>-</td>
</tr>
<tr>
<td>Definition of a project specific process</td>
<td>Project model at level 2</td>
<td>Project within a project library derived from above</td>
<td>Direct mapping from above level possible + additional tasks if required</td>
</tr>
<tr>
<td>Monitoring of project process execution</td>
<td>Project execution</td>
<td>Progress and metadata per Project task</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Atego Process Director concepts used for OPENCOSS implementation

### 5.1.1.3 CCL implementation within Atego Process Director

CCL metaclasses are implemented using *Process library types*. The following chapters describe CCL metaclasses implementation.
5.1.1.3.1 RefActivity & Activity

This Atego Process Director library type implements the activity concept of CCL, which appears in various levels:

- **ReferenceActivity** in the Reference Assurance Framework Metamodel to model Level 1 and Level 1b (standards and company processes)

- **Activity** in the Process metamodel to model Level 2 (process execution)
To allow Atego Process Director to ensure an efficient direct mapping between Level1b and Level1, the Atego Process Director library type gathers all the required attributes and relations required by both metamodels. In such a case Atego Process Director tasks used at Level 2 are the direct instantiation of Atego Process Director activities.

If a direct mapping is not sufficient, Atego Process Director allows creating Atego Process Director tasks at Level 2 to monitor process execution but this requires additional effort which is less efficient.

**Caution:** This Atego Process Director library type is defined in Atego Process Director library settings as ensuring the Atego Process Director process role

5.1.1.3.1.1 **Relationships**

A section *Used Techniques* (out) [0..N] implements the utilizes relationship (*ReferenceTechnique* and *Technique* as a target)

A section *Required Artefacts* (in) [0..N] implements the requires relationship (*ReferenceArtefact* and *Artefact* as a target)

A section *Produced Artefacts* (out) [0..N] implements the produces relationship (*ReferenceArtefact* and *Artefact* as a target)

A section *Executed By Role* (out) [0..N] implements the executed by relationship (*Role, Participant* (abstract), *person* (concrete), *tool* (concrete), *organization* (concrete) as targets)

*ActivityRelationShip* of type *Precedence* will be captured using Atego Process Director steps

*ActivityRelationShip* of type *Decomposition* will be captured using Atego Process Director steps

5.1.1.3.1.1.1 **Attributes for execution**

Atego Process Director metadata *Start time* implements *startTime* (process execution need at level 2)

Atego Process Director metadata *End time* implements *endTime* (process execution need at level 2)

Atego Process Director metadata *Status* implements *status* (process execution need at level 2)
5.1.1.3.2 ReferenceArtefact&Artefact

This Atego Process Director library type implements the artefact concept of CCL which appears in various levels:

- **ReferenceArtefact** in the Reference Assurance Framework Metamodel to model Level 1 and Level 1b (standards and company processes)

- **Artefact** in the Artefact metamodel
5.1.1.3.2.1 Relationships

A section **Required Artefacts** is defined in the Atego Process Director type for activity

A section **Produced Artefacts** is defined in the Atego Process Director type for activity

A section **Results From Technique** implements **resultsFrom** relation (**ReferenceTechnique** as target)

A section **Changes require modification of other artefacts** implements relation **ReferenceArtefactRelationship**

A section **Changes require revocation of other artefacts** implements relation **ReferenceArtefactRelationship**

**Caution** **ArtefactRelationship** to be discussed before implementation.

5.1.1.3.2.2 Attributes

No attribute is explicitly defined whereas **ReferenceArtefactAttributes** and **ReferenceArtefactAttributesValue** are in the metamodel
5.1.1.3.3 RefTechnique & Technique

This Atego Process Director library type implements the class which corresponds to specific ways to create a reference artefact.

5.1.1.3.3.1 Attributes

- aim: string implemented as an Atego Process Director section Aim

The purpose of a reference technique.

5.1.1.3.3.2 Relationships

- assigned to Criticality Applicability [0..*] (not implemented yet)

The criticality applicability to which a reference technique is assigned.
- aimed at Requirement [0..*] (not implemented yet)

The requirements at which the use of a reference technique is targeted.
5.1.1.3.4 Role

This Atego Process Director library type implements the CCL Role class and is declared in Atego Process Director library settings as ensuring Atego Process Director role in this library.

5.1.1.3.5 Tool

This Atego Process Director library type implements the class which corresponds to the software tools used in an assurance project.

Superclass
- Participant (abstract, not implemented)

Attributes
- version: string (implemented by section Version)
The version in use of a tool.

5.1.1.3.6 Organization

This Atego Process Director library type implements the class which corresponds to groups of people (companies, societies, associations, etc.) that are involved in an assurance project.

Superclass
- Participant (abstract class not implemented in Atego Process Director)
- 

Attributes
- address: string implemented with the section Address
The place where an organization is located.

Relationships
• part of Organization [0..*] implemented in the section Part of organization
The organization to which an organization belongs

![Organization(CCL) template](image)

**Figure 30** Organization library type

### 5.1.1.4 Example of project model using CCL

As a feasibility example a simple company project (level 1b) is used that includes:
- Parallel activities 2/2b
- Iterative activity 4
- Optional activity 5
- Performed 2 times activity 3
- Activity 6 with alternatives

![Process diagram](image)

**Figure 31** Master project example in process library
The equivalent BPMN view is the following

![BPMN view of example project](image)

Figure 32 BPMN view of example project

A project instantiated and tailored from the above company process is seen by team members as below (please note that color code is used for progress):
5.1.1.5  Atego Process Director exported files

Atego Process Director features different ways to export models information for automatic import into OPENCOSS platform:

- Export from the **Process library**, may be used for level 1 and 1b models (standards and company models)
- Export from the project in the **Project library**, may be used for level 2 (project models and execution data)
- SQL server database export
- Web service API
- Office exports (Excel, Word, PDF)
- HTML export

The two first ones are foreseen for OPENCOSS coupling and are described further below (if needed direct access to the API will be possible).

### 5.1.1.5.1 Exports from the Process Library

From the process library the export produces an xml file with the following structure:

```
<library>
  <name>CCL</name>
  <domain>processdirector.atego.com</domain>
  <types>
    <processes/>
  </types>
</library>
```

The subsection **types** contains the **Library types** which are implementing the CCL metamodel classes.
The RefActivity(top) is the head of the process tree.
In the links section one finds the instances of the relations authorized, e.g. the person who is leading the activity (which is of type Role).
In the steps sections are recorded the ActivityRelationship of decomposition type with the precedence implemented using position attribute.

An example of role is provided below with the project manager which is typed by the Role type.
Caution: Please note that the process section records all process concepts whatever their library type is activity, role, technique .... The typeid references the actual type of the concept.

5.1.1.5.2 Exports from the Project Library

From a project in the project library, the export is done through a Microsoft project export including actual project tasks with related schedule and progress. The file is structured as follows:
The first level of activities is generated as follows. **PercentComplete** tracks the actual progress.
5.1.1.6 Coupling to OPENCOSS platform

5.1.1.6.1 Coupling for level 1/1b models

The xml export will be used (its need is currently under discussion) to transfer data into a BPMN file. A BPMN subset shall be defined to at least cover concepts used for CCL implementation in Atego Process Director. This mechanism will be used to transfer to OPENCOSS:

- Standards models at level 1
- Company models at level 1b

5.1.1.6.2 Coupling for level 2 models

The Microsoft Project export will be used to transfer projects and execution data at level 2. As project data shall be extracted after Atego Process Director tailoring, the xml generated file from Process library is not relevant for that purpose. In addition, at this level all activities are mandatory (MS Project and Atego Process Director shared paradigm), this shall greatly simplify the required BPMN subset. The bridge will format data into the BPMN subset defined in previous section in order to feed OPENCOSS with the process structure.

Transfer of progress data shall be studied further.

5.1.1.7 Feasibility results and Open points

In general, Atego Process Director being a process focused tool and as it is using an agnostic metamodel (Atego Process Perspective), configuring Atego Process Director to support process part of CCL language is feasible. The two kinds of exports provided by Atego Process Director also allows coupling to OPENCOSS. More in details, first studies show the feasibility of:

- Customizing Atego Process Director to support process part of CCL concepts
- Capture of level 1/1b/2 level models in Atego Process Director
- Transfer of level 1/1b process data using the xml export
- Transfer of level 2 process structure using the MS project export

Some additional points require to be studied further:

- Transfer of process execution data through BPMN or through another technology as they are currently being defined in “Candidate technologies for the APIs” “Candidate technologies for the APIs”
- Some metamodel concepts which are not strictly related to process shall be discussed further to define the relevant Atego Process Director implementation (e.g. artefacts relationships)

5.1.1.8 Out of the box and OPENCOSS feasibility libraries

The following libraries have been set up on Atego servers for OPENCOSS partners. Please do request your Atego contact to provide you access to them:

- Eric.andrianarison@atego.com
- Maurice.assouline@atego.com
- Eric.boccanfuso@atego.com
- Sebastien.rocher@atego.com
5.1.1.8.1 Out of box

Out of the box tool may be used at the following address http://opencoss.processdirector.atego.com.

Going to the Library structure link on the right hand panel, allows to access common best practices implemented in Atego Process Director

5.1.1.8.2 Feasibility prototype

OPENCOSS configuration defined for feasibility can be accessed at the following address http://ccl.processdirector.atego.com.

Going to the Library types link on the right hand panel, allows to access to the implementation of major metamodel concepts.

Going to the Library structure link on the right hand panel, allows to access to the testing process defined

5.1.2 Parasoft Concerto

5.1.2.1 Introduction

Parasoft Concerto is Application Lifecycle Management (ALM) tool. It can be used to plan, manage and monitor any software project. It is targeted to all team members including developers, QA, managers, and analysts.

Concerto provides various piece of functionality:

- Requirement Management System (RMS) – requirements describing functionality to be implemented during the project, they can be defined, managed, reviewed.
- Task Management System (TMS) - task in Concerto represents a unit of work assigned to be completed by a project team member.
- Defect Management System (DMS) – management of bugs or problem reports found in the product.
- Iterations / Tasks / Planning – project can be split into milestones/iteration and each such phase can have work (tasks) planned and distributed between team members.
- Test definition and execution – test scenarios can be defined, managed, reviewed. Test runs can be scheduled, executed, verified.

Additionally Concerto supports traceability between artefacts, for example from requirements to tasks, source code, tests, and defects.
5.1.2.2 Task Management System in Concerto

Task Management System is a piece of functionality of Concerto which could be integrated with OPENCOSS WP7 Process Assurance Management.

Task in Concerto represents a unit of work assigned to be completed by a project team member. Task can contain the Sub-Tasks. Each sub task contains a “Parent Id” property which points to parent task.

A task can represent:
- work to implement a requirement
- work to fix a defect
- work to making a review
- work to learn some new technology
- any work which is done by any team member
A task has the following fields:

“Project” – project name to which this task is assigned

“Name” – name of task

“Description and Comments” - description what should be done in the task and what was done during the work

“Status” – represents the state of the work. Status values are: “open”, “in progress”, “completed”, “cancelled”

“Deviation” = (Planned Time) – (Actual Time + Remaining Time) – this field presents deviation of actual work from planned work.

“Planned Time”, “Actual Time”, “Remaining Time” – represent time measurement. We can plan the work and monitor actual progress. Each time is measured in minutes, hours, days, e.g. 1d 2h 31m

“Actual Time” – when a user starts work with defined task, he marks a task as “in progress”. Time for this task/work is being measured. When a user finishes the task, he marks a task as “completed”. Time is saved for this task.

Other useful fields: “Owner(s)”, “Iteration ID”, “Parent ID” ...

Custom field – a user can define any field which is useful in the project. Custom field could be: Text Input or Single Select List, for example new Text Input: label: “OPENCOSS_guidance”.

“Predecessor(s)” – task(s) specified in predecessor(s) field should be made before the task. For example: task2 “Verification review report of Hazard Analysis” has Predecessor(s): task1 “Hazard Analysis” defined, it means that “Hazard Analysis” should be done before “Verification review report of Hazard Analysis”.

A task has also modification history, sub-tasks hierarchy and traceability info, e.g. from requirement to tasks, assigned automated tests, test scenarios and test runs.

For example if we have two tasks:
1. task1: “Hazard Analysis and Risk Assessment”
2. task2: “Verification review report of Hazard Analysis and Risk Assessment”
Task1 can be defined as predecessor of task2: Task1 can have status: “in progress”, “open”, “completed”, “cancelled”. When the task status is “completed” we know that this task is finished and we can start further work: task2, which will have the status of work: “in progress”, “open”, “completed”, “cancelled”.

### 5.1.2.3 Concerto tasks vs. OPENCOSS process steps (Activity)

Concerto does not provide a functionality to define a process (e.g. in BPMN language), neither it provides any automated process execution. However it provides advanced Task Management System, which facilitates work planning in the project, distributing task among team members, monitoring, marking as done.

**We can treat a task as a step of process – equivalent to Activity (CCL entity) in OPENCOSS.**

In this chapter we describe how we can connect a task in Concerto to a process steps (Activity) in OPENCOSS.

### 5.1.2.4 Idea of integration of Concerto tasks and OPENCOSS process

#### 5.1.2.4.1 Publish/Subscribe JMS events

When any modification is done to a task, Concerto sends the JMS event with the information about task changes. OPENCOSS can be registered as a listener and JMS message can be used in OPENCOSS to get the information about task changes – work done in the project: some tasks might have been completed/finished, some might have been opened/ready for work. OPENCOSS can get the information which was created, changed or deleted in Concerto.

Concerto publishes messages to the Topic, from this Topic other clients can subscribe and consume the messages. In Pub/sub messaging each message can have multiple consumers and a client that subscribes to a topic can consume only messages published after it has created a subscription.
Concerto creates the message with the changed data and publishes/sends them under pub/sub domain. OPENC OSS can have the client which is registered as a listener to get the data from the topic.

The flow of communication between Concerto as a Process External Tool and OPENC OSS can go in the following way:

- “JMS – OPENC OSS Adapter” subscribes to topic of JMS Events sent by Concerto
- “JMS – OPENC OSS Adapter” subscription consumes JMS Events messages published by Concerto client
- “JMS – OPENC OSS Adapter” subscriber must continue to be active in order to consume messages
- the JMS Events are translated to OPENC OSS Process API and the API is called

“JMS – OPENC OSS Adapter” could consume messages asynchronously and could read coming mapMessage with the data from Concerto.

Example information from the JMS mapMessage sent by Concerto:

1. getEntityType: TASK – entity type, for OPENC OSS useful type is “TASK”
2. getActivityType: UPDATE/CREATE
3. getEntityId: 347 - task id
4. getActivityTime: Tue Jun 25 11:39:58 CEST 2013 – time when changes are made
5. getUser: User:<6, admin> - user who the changes made
6. getActivitiesArray - getting the changed data related to changed task

   a) during UPDATE task activities array could be:
   Remaining Time - added: 480, removed: 0, Status - added: 1, removed: 0

   b) during CREATE task activities array could be:
After reading the data from JMS mapMessage, the Adapter can call the Process API in OPENCOSS and for example create/update Activities using REST API described in 5.2.1.

To sum up, after each task creation or update in Concerto, a registered listener is notified and it triggers an adequate creation or update of Activity in OPENCOSS platform.

In the above scenario, after OPENCOSS platform is integrated with Concerto, each task change during project lifecycle is read from the JMS coming message. The scenario covers Level 2 Project scope integration. Concerto does not support certification, thus after Activity is created in OPENCOSS (based on Concerto task) a mapping between ReferenceActivity and Activity should be made manually in OPENCOSS platform.

Another scenario that one could imagine is the need to create Activities in OPENCOSS for already existing Concerto tasks (e.g., when OPENCOSS reference project is integrated with Concerto after the tasks are created in Concerto). Such integration could be achieved by the tasks export option from Concerto. Tasks from Concerto can be exported to .csv or MS project .mpp files, and then imported into OPENCOSS platform.

### 5.1.2.4.2 Mapping between: Tasks in Concerto and Activities in OPENCOSS:

The Tasks from Concerto can be mapped to Activities in OPENCOSS. To have this mapping it should be known which Task corresponds to which Activity in OPENCOSS. In order to define this mapping we propose two alternative solutions by adding a new attribute to Activity:

**Solution 1:**

**externalID** (externalID: EInt) - relate to taskID from the Task (Concerto)

In this solution a mapping is defined on OPENCOSS side (in Activity – externalID field)

**Solution 2:**

**GUID** (GUID: EInt) - unique Activity ID

In this solution a mapping is defined on external tool side – in Concerto task.

**Solution 1: Task ID in Concerto <-> externalID OPENCOSS:**

For example the Task: “Prepare HAZARD ANALYSIS AND RISK ASSESSMENT” has task ID: 344. This task ID will be sent in JMS message as task identifier where the changes are made.

When the status is changed from “open” to “in progress” the message will be send by Concerto:
ActiveMQMapMessage {commandId = 0, responseRequired = false, messageId = null, originalDestination = null, originalTransactionId = null, producerId = null, destination = null, transactionId = null, expiration = 0, timestamp = 0, arrival = 0, brokerInTime = 0, brokerOutTime = 0, correlationId = null, replyTo = null, persistent = false, type = null, userId = null, content = null, marshalledProperties = null, dataStructure = null, redeliveryCounter = 0, size = 0, properties = {Label0=Status, ActivityType=UPDATE, EntityType=TASK}, readOnlyProperties = false, readOnlyBody = false, droppable = false}

ActiveMQMapMessage{ theTable = {EntityField0=STATUS, NumericalValue0=true, RemovedValue0=0, ModificationDate=2013-06-25 10:26:31, UserName=admin, UserId=6, EntityId=344, AddedValue0=1, BTSUpdateEnabled=false, ActivityCount=1} }

In the above map message:

- EntityId=344 – Task ID – identification of entity changed
- EntityField0=STATUS – changed status field of entity
- Label0=Status – name of changed entity
- RemovedValue0=0 – removed value: “open”
- AddedValue0=1 – added value: “in progress”

Task ID (Concerto) should be mapped to externalID (OPENCOSS): externalID for Activity to know that for Task: ID = 344 – Activity: externalID = 344 some changes are made.

**Solution2: OPENCOSS_guid in Concerto <-> GUID in OPENCOSS**

In Concerto we can define a custom filed which would store the mapping to the GUID of Activity in OPENCOSS: OPENCOSS_guid with the value equals to GUID of Activity in OPENCOSS. When the JMS message is sent from Concerto with EntityId=xxx, JMS Adapter should ask Concerto (by calling Concerto API) about OPENCOSS_guid for EntityId=xxx and then for proper OPENCOSS_guid JMS OPENCOSS Adapter will know for which Activity in OPENCOSS the changes are made.

### 5.1.2.4.3 Mapping between Concerto Task status and OPENCOSS Activity attributes

This chapter presents a possible mapping between status property of Concerto task and OPENCOSS Activity class attributes.

1. Create task in Concerto

When a task is created in Concerto, a JMS message is sent with the following data: ActivityType=CREATE and EntityType=TASK. In OPENCOSS the new Activity can be created:

Activity with externalId attribute == Concerto taskId (mapping solution 1 described in the previous chapter) or Activity with GUID == OPENCOSS_guid (mapping solution 2 described in the previous chapter)

In Concerto a status of newly created task is set to “open”. In OPENCOSS the Activity will be created with empty “startTime” and “endTime” attributes.

2. Update task status in Concerto
Concerto Tasks have the following statuses: “open”, “in progress”, “completed” or “canceled”. When a task is updated in Concerto, a JMS message is sent with the following data: **ActivityType=UPDATE, EntityType=TASK** and with additional data: EntityField0=STATUS, RemovedValue0=0 (removed value: “open”) and AddedValue0=1 (added value: “in progress”).

a) When the task status in Concerto is set to “in progress”, in OPENCOSS platform the following change should be made:
   - Set the “beginTime” attribute in Activity to the proper time stamp

b) When the task status is changed from “open” or “in progress” to “completed”, in OPENCOSS platform the following change should be made:
   - set “endTime” attribute in Activity to the proper time stamp

c) When the task status is changed from “open” or “in progress” to “cancelled”, in OPENCOSS the following changes should be made:
   - set “startTime” and “endTime” attribute in Activity to empty value/null

d) When the task status is changed from “completed” or “cancelled” to “in progress” the JMS message is sent and in OPENCOSS the following changes should be made:
   - set “endTime” attribute in Activity to empty value/null
   - create AssuranceAssetEvent for the Activity with the proper EventType (AssuranceAssetEvent class – Figure 8 from D4.4: Manageable Assurance Asset Metamodel)

5.1.2.4.3 Mapping between Concerto Task relations and OPENCOSS Activity relationship

In the OPENCOSS CCL conceptual model, Activity has relationship with ActivityRelationship class. ActivityRelationship corresponds to existence of a relationship between two activities. It has the “type”: ActivityRelationshipType (see below Metamodel) with the values:

- **Decomposition** – an activity is decomposed into several sub-activities
- **Precedence** – the execution of an activity precedes the execution of another activity

On the other hand, in Concerto there are the following relationships between the tasks:
sub-tasks which can be relevant to sub-activities in OPENCOSS “Predecessor(s)” field which could play the role of precedence relationship in OPENCOSS.

a) Sub-tasks in Concerto can be treated / mapped to Activity decomposition:

When a new sub-task is created, the jms message is sent with the information about creating new sub-task:

JMS message:

```
ActiveMQMapMessage {commandId = 0, responseRequired = false, messageId = null, …, properties = {
  Label15=Project, Label16=Parent ID, Label10=Relation, ActivityType=CREATE, Label7=Requirement ID, Label4=Status, Label14=Owner(s), Label13=Priority, Label12=Planned Time, Label8=Story Points, Label11=Name, Label9=Project, Label1=Owner(s), Label0=OPENCOSS_EventType, Label3=Type, Label2=Testing Required, EntityType=TA SK, …,
  AddedValue0=Created, …, AddedValue7=42, AddedValue6=344, AddedValue5=Implementing a Requirement, …, AddedValue9=40, RemovedValue9=null, ModificationDate=2013-06-25 11:39:58, RemovedValue2=null, RemovedValue1=, RemovedValue4=null, RemovedValue3=, RemovedValue6=, RemovedValue5=null, …,
  EntityId=347, …, AddedValue14=6, AddedValue15=OPENCOSS, AddedValue12=480, AddedValue13=3, AddedValue10=2, AddedValue11=Plan of HAZARD ANALYSIS AND RISK ASSESSMENT, …,
  AddedValue13=PRIORITY_ID, …, AddedValue12=TIME_ESTIMATE, NumericalValue9=true, …,
  EntityId10=PROJECT_NAME, EntityId14=OWNERS_ID, EntityId11=NAME, …,
  EntitiesField10=TASK_TYPE_ID, …
}
```

The JMS message is sent with: **ActivityType=CREATE** and **EntityType=TASK**, so in OPENCOSS the following objects could be created:

1) new ActivityRelationship object with ActivityRelationshipType = Decomposition

2) new Activity with:

   - externalId = 347 relate to new created Task: **EntityId=347**
   - guid = OPENCOSS_guid from newly created Task in Concerto
b) Task predecessor in Concerto can be treated / mapped to Precedence in OPENCOSS:

In Concerto when Predecessors(s) field is set to the Task ID: 344, it means that Task ID: 344 should be first done and then the Task id: 345 should be made. In OPENCOSS could be done first: “Prepare HAZARD ANALYSIS AND RISK ASSESSMENT” and then should be done: “Verification Review Report of the Hazard Analysis and Risk Assessment”.

After setting Predecessors(s), Concerto sends the JMS message with the info:

```
ActiveMQMapMessage {commandId = 0, responseRequired = false, messageId = null, …, properties = {Label0=Predecessor(s), ActivityType=UPDATE, EntityType=TASK}, readOnlyProperties = false, readOnlyBody = false, droppable = false} ActiveMQMapMessage{ theTable = {EntityField0=CUSTOM_FIELDS, NumericalValue0=false, RemovedValue0= RemovedValue0= RemovedValue0= RemovedValue0=, UserName=admin, UserId=6, EntityId=345 EntityId=345 EntityId=345 EntityId=345, AddedValue0=344 AddedValue0=344 AddedValue0=344 AddedValue0=344, BTSUpdateEnabled=false, ActivityCount=1} }
```

In map message:
- EntityId=345 – Task id – identification of entity changed
- EntityField0=CUSTOM_FIELDS – changed custom field of entity
- Label0=Predecessor(s) – name of custom field
- RemovedValue0= – removed value: “” - it was empty
- AddedValue0=344 – added value: “344” - Task id of the Task which will be Predecessor of the actual task

When JMS message is sent with: Label0=Predecessor(s), then in OPENCOSS the following data could be created/updated:

5.1.2.5 Conclusions
Concerto is Application Lifecycle Management System. It does not provide a functionality to define a process or execute it automatically, but it provides advanced Task Management System.

A Concerto task can be treated as a step of process which is equivalent to Activity in OPENCOSS (defined in CCL). During work in Concerto, when changes are made in a task, JMS events are sent. We can implement “JMS Adapter” which can:

- listen to Concerto’s JMS event with information about task changes
- call Process API in OPENCOSS. This API is described in 5.2.1. It can update OPENCOSS Activity, create ActivityRelationship and AssuranceAssetEvent objects.

Concerto tasks and OPENCOSS Activity classes have similar but not identical properties. The following mapping between these classes properties is proposed:

1. **Map between specific Concerto task and OPENCOSS Activity.**
   This can be achieved by 2 alternatives ways:
   
   **Solution1:** A new `externalID` attribute can be added to Activity class in OPENCOSS with the value which equals TaskID field from Concerto.
   
   **Solution2:** A new `GUID` attribute can be added to Activity class in OPENCOSS and new custom field `OPENCOSS_guid` can be added to Task in Concerto. Concerto `OPENCOSS_guid` custom field should be equal to OPENCOSS Activity GUID value.

2. **Map between Concerto Task status and OPENCOSS Activity attributes.**

   When new task in Concerto is created the JMS message is sent and new Activity can be created in OPENCOSS with proper externalID or GUID and empty “startTime” and “endTime” attributes.

   In Concerto task has status field with the following values: “open”, “in progress”, “cancelled”, “completed”. When the task status in Concerto is changed, the Activity in OPENCOSS should be updated in the following way:
   
   - setting Activity “startTime” when Task status is set to “in progress”
   - setting Activity “endTime” when Task status is set to “completed”
   - setting Activity “startTime” and “endTime” to empty value when Task status is set to “canceled”
   - setting Activity “endTime” to empty value when Task status is set to “in progress” from “completed” or “canceled” and creating new AssuranceAssetEvent for this Activity with the proper EventType

3. **Map between Concerto Task relations and OPENCOSS Activity relationship.**
3.1 Sub-tasks in Concerto can be relevant to ActivityRelationshipType: Decomposition in OPENCOSS. When a new sub-task is created in Concerto, a new Activity object and a new ActivityRelationship object (with the type: Decomposition) should be created in OPENCOSS.

3.2 Predecessor field in Concerto task can be mapped to ActivityRelationshipType: Precedence in OPENCOSS. When a predecessor field is set in Concerto task, a new ActivityRelationship object (with the type: Precedence) should be created in OPENCOSS.

5.2 Candidate technologies for the APIs

5.2.1 Web services (in large: SOAP, REST, ...)

There are several options to select from when creating a web-enabled API for OPENCOSS. We can select from two different approaches to exposing this API: either full-blown XML-based web services based on SOAP and WSDL, driven by e.g. JAX-WS-compatible stack, or a lightweight approach of JAX-RS to create simple, RESTful web services (this is approach adopted by IBM Rational Team Concert). We can also mix both these approaches.

To have all the benefits from using standardized solutions and be able to pick from several different implementations we should focus on either JAX-WS or JAX-RS – both these standards are widely used, well-established and have several alternative implementations. Here we gathered some facts about these standards.

5.2.1.1 JAX-RS

This is a short compilation of information on JAX-RS based on its Wikipedia page (http://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services):

JAX-RS: Java API for RESTful Web Services is a Java programming language API that provides support in creating web services according to the Representational State Transfer (REST) architectural pattern. JAX-RS uses annotations, introduced in Java SE 5, to simplify the development and deployment of web service clients and endpoints. From version 1.1 on, JAX-RS is an official part of Java EE 6. A notable feature of being an official part of Java EE is that no configuration is necessary to start using JAX-RS. For non-Java EE 6 environments a (small) entry in the web.xml deployment descriptor is required. In January 2011 the JCP formed the JSR 339 expert group to work on JAX-RS 2.0. The main targets are (among others) a common client API and support for Hypermedia following the HATEOAS-principle of REST. As of March 2013, it has reached proposed final draft stage. Public release (GA) will follow.

JAX-RS provides some annotations to aid in mapping a resource class (a POJO) as a web resource. The annotations include:

@Path specifies the relative path for a resource class or method.

@GET, @PUT, @POST, @DELETE and @HEAD specify the HTTP request type of a resource.

@Produces specifies the response Internet media types (used for content negotiation).

@Consumes specifies the accepted request Internet media types.
In addition, it provides further annotations to method parameters to pull information out of the request. All the @*Param annotations take a key of some form which is used to look up the value required.

@PathParam binds the method parameter to a path segment.

@QueryParam binds the method parameter to the value of an HTTP query parameter.

@MatrixParam binds the method parameter to the value of an HTTP matrix parameter.

@HeaderParam binds the method parameter to an HTTP header value.

@CookieParam binds the method parameter to a cookie value.

@DefaultValue specifies a default value for the above bindings when the key is not found.

@Context returns the entire context of the object. (for example @Context HttpServletRequest request)

5.2.1.2 JAX-WS

This is a short compilation of information on JAX-WS based on its Wikipedia page (http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services):

The Java API for XML Web Services (JAX-WS) is a Java programming language API for creating web services. JAX-WS is one of the Java XML programming APIs. It is part of the Java EE platform from Sun Microsystems. Like the other Java EE APIs, JAX-WS uses annotations, introduced in Java SE 5, to simplify the development and deployment of web service clients and endpoints. It is part of the Java Web Services Development Pack. JAX-WS can be used in Java SE starting with version 6.

The reference implementation of JAX-WS is developed as an open source project and is part of project GlassFish, an open source Java EE application server. It is called JAX-WS RI (RI for reference implementation) and is said to be a production quality implementation (contrary to the former reference implementation being a proof of concept). This reference implementation is now part of the GlassFish Metro distribution.

JAX-WS also is one of the foundations of WSIT (Web Services Interoperability Technology).

5.2.1.3 JAXB

One of the things which is common to both JAX-WS and JAX-RS is that most implementations have support for JAXB as a means to specify how to deal with the objects which travel through selected transport.

This is a short summary about JAXB based on its Wikipedia page (http://en.wikipedia.org/wiki/Java_Architecture_for_XML_Binding):

Java Architecture for XML Binding (JAXB) allows Java developers to map Java classes to XML representations. JAXB provides two main features: the ability to marshal Java objects into XML and the inverse, i.e. to unmarshal XML back into Java objects. In other words, JAXB allows storing and retrieving data in memory in any XML format, without the need to implement a specific set of XML loading and saving routines for the program's class structure. It is similar to xsd.exe and XmlSerializer in the .NET Framework.
JAXB is particularly useful when the specification is complex and changing. In such a case, regularly changing the XML Schema definitions to keep them synchronised with the Java definitions can be time consuming and error prone.

JAXB is a part of the Java SE platform and one of the APIs in the Java EE platform, and is part of the Java Web Services Development Pack (JWSDP). It is also one of the foundations for WSIT. JAXB is part of SE version 1.6.

5.2.1.4 Conclusion about presented technologies

Each of the presented technologies eases the burden of hand-managing the consistency of descriptors of exposed APIs. This is a most welcome thing when a need for agile development of a web-enabled API arises. Sticking to JAXB-driven annotations will help to expose both JAX-RE and JAX-WS, even simultaneously, with a reasonable amount of work.

It is worth noting that JAX-RE implementations can return results in many selected mime types. So you can get response in JSON, XML or even plain text.

Another thing is that both RESTEasy and Jersey have complete Java client APIs for remotely interacting with deployed JAX-RE APIs from another Java-based application.

5.2.1.5 Evaluation of selected JAX-RE implementations

There are several implementation of this standard, amongst them:

- Jersey (https://jersey.java.net)
- RESTEasy (http://www.jboss.org/resteasy)
- Restlet (http://restlet.org)
- Apache CXF (http://cxf.apache.org)
- Apache Wink (http://wink.apache.org)

We have some experience with most of these frameworks. Apache CXF is a mature framework but JAX-RE was not its first-class citizen from the beginning. We have no experience with Restlet – it has a reputation of a powerful yet a little bit complex framework. According to impression left by browsing several forums and Q&A pages, both Jersey and RESTEasy are considered the most friendly and very complete frameworks. This is also our own observation – we have strong experience with both. Which to choose depends on the complementary technologies used in the OPENCOSS core. For example, RESTEasy integrates very well with JBoss Seam. Currently we are strong users of Jersey and have no bad experiences with it. Jersey is the reference implementation of JAX-RE with good integrations capabilities, strong community and very good opinions on several forums and Q&A pages.

To illustrate condition of JAX-RE and some of mentioned frameworks, let’s take a look at some of the OHLOH stats. OHLOH (www.ohloh.net) is a site which provides comparison and metrics for open source projects. It allows to measure maturity the project, how huge its contributors base is and how active they are.

- Apache Wink
  - 137K lines of code
  - 3 current contributors
  - 6 months since last commit
  - 3 users on Ohloh
- Restlet
5.2.1.6 Evaluation of selected JAX-WS implementations

There are three popular implementations for JAX-WS

- Apache CXF (http://cxf.apache.org)
- Apache Axis 2.0 (http://axis.apache.org/axis2/java/core/)
- Reference Implementation, Metro stack (http://metro.java.net)

From these, CXF and Axis2 are the most recognizable ones and we have experience with both of these. Axis2 is a complete redesign of Axis – with architecture recreated from scratch. CXF is based on projects XFire and Celtix. CXF and XFire was integrated into numerous other projects like ServiceMix and Mule.

Axis2 was supposed to fix some issues of Axis1. Still, flexibility of integration of CXF is impressive and less cumbersome that in case of Axis2. Incorporating CXF into existing architecture is a blitz and serving services by just assigning servlet mapping allows for very flexible configurations. On the contrary Axis2 requires some rigid structure when deploying services and this determines the possible deployment model very seriously. Additionally, CXF has JAX-RS support – using dedicated solution, Jersey, will be a better choice, but this is just for reference.

OHLOH stats of presented frameworks:

- Apache CXF
  - 769K lines of code
  - 17 current contributors
  - 7 days since last commit
  - 98 users on Ohloh
- Axis2
  - 448K lines of code
  - 6 current contributors
  - 26 days since last commit
  - 133 users on Ohloh
- WSIT (Metro stack)
  - 928K lines of code
  - 13 current contributors
  - About 1 year since last commit
  - 16 users on Ohloh
5.2.1.7 Conclusion

Using one or both of the presented standards to expose some of the OPENCOSS functionality through a remote API should be an easy task due to high popularity and simplicity of these solutions. These standards are well established, built upon other widely adopted standards (JAXB etc.) and are easy to develop and maintain due to annotation-driven configuration capabilities. That brings the promise of many integration opportunities. For example, when implementing API for process assurance management this API can be exposed for interaction not only with common REST/JSON and WS/XML clients but for BPEL engines and other web-enabled, standards-based consumers as well.

5.2.2 Open Services for Lifecycle Collaboration

This section provides an overview of Open Services for Lifecycle Collaboration (OSLC) based on information from the OSLC website (http://open-services.net/). More detailed information about OSLC is provided on the OSLC website.

OSLC is an open community creating specifications for integrating tools. IBM is also very active in this community. These specifications allow lifecycle tools to integrate their data and workflows. The OSLC community is organized into workgroups. Each workgroup explores integration scenarios for a given lifecycle domain and specifies a common vocabulary for the lifecycle artefacts needed to support the scenarios.

To ensure coherence and integration across the domains, each OSLC domain specification builds on the concepts and rules defined in the OSLC Core and common specifications. The OSLC Core specifies the integration techniques for integrating tools. It consists of standard rules and patterns for using HTTP and RDF and a small number of resource types that are used in the OSLC domain specifications. The OSLC domain specifications define additional resource types, but do not add new protocols. The OSLC Core specification in conjunction with one or more of the domain specifications describe the OSLC protocols offered by a domain tool.

The goal of OSLC is to create specifications to support integration between tools. OSLC specifies a minimum amount of protocol and a small number of resource types to allow tools to work together. Even within resource types, only properties that are required for integration are defined. OSLC also tries to accommodate a wide variety of implementation technologies, and be equally relevant to both existing and new tools.

OSLC is based on the W3C Linked Data. In OSLC, each artefact is an HTTP resource that is manipulated using the standard methods of the HTTP specification (GET, PUT, POST, DELETE). Each resource has an RDF representation. OSLC mandates RDF/XML, but can have representations in other formats, like JSON or HTML.

5.2.2.1 OSLC Integration Techniques

OSLC offers two techniques for integrating tools – “Linking data via HTTP” and “Linking Data via HTML User Interface”. Both of these techniques build on the HTTP and RDF foundation of OSLC.

Linking data via HTTP

OSLC specifies a common tool protocol for creating, retrieving, updating and deleting (CRUD) lifecycle data based on HTTP and RDF using the Linked Data model. This protocol can be used by any tool or other programmatic client to talk to any other tool that implements the specifications. Linking is achieved by embedding the HTTP URL of one resource in the representation of another.
Linking Data via HTML User Interface
OSLC specifies a protocol that allows a tool or other client to cause a fragment of the web user interface of another tool to be displayed, allowing a human user to link to a new or existing resource in the other tool or see a preview of information about a resource in another tool. This enables a tool or other client to exploit existing user interface and business logic in other tools when integrating information and process steps.

5.2.2.2 OSLC ServiceProvider
OSLC defines the concept of a ServiceProvider to allow tools to expose artefacts to support integration. A ServiceProvider represents a group of artefacts that are provided by a tool not the tool itself. A single instance of a tool will typically hosts multiple ServiceProviders, for example one for each “project” or “product”.

ServiceProviders are the central organizing concept of OSLC, enabling tools to expose resources and allowing consumers to navigate to all of the resources, and create new ones. Characteristics of ServiceProviders are:

1. All OSLC resources live in some ServiceProvider.
2. Clients can retrieve the list of existing resources in a ServiceProvider.
3. New OSLC resources are created in a ServiceProvider (either directly through an HTTP POST, or via a dialog).
4. A ServiceProvider is itself an OSLC resource with an HTTP URL.

When adopting OSLC for a new or existing tool, it is necessary to define which concepts in the tool should be mapped to OSLC ServiceProviders.

5.2.2.3 OSLC Specifications
The following OSLC specifications exist. See [http://open-services.net/specifications/](http://open-services.net/specifications/) for more information.

Core and Common Specifications
- Core
- Configuration Management
- Reporting

Domain Specifications
- ALM-PLM Interoperability
- Architecture Management
- Asset Management
- Automation
- Change Management
- Estimation and Measurement
- Performance Monitoring
- Quality Management
- Reconciliation
- Requirements Management

5.2.2.4 OSLC Tools
The following OSLC tools are available. See [http://open-services.net/resources/](http://open-services.net/resources/) for more information.

Eclipse Lyo
The Eclipse Lyo project focuses on providing an SDK to help the Eclipse community to adopt OSLC specifications and build OSLC-compliant tools.

- OSLC4J: A Java toolkit to help build OSLC-ready Java applications.
- Test suites: Verify that OSLC implementations are compliant with the specifications.
- Reference Implementations: OSLC examples.

**OSLC4Net**

A toolkit for developing OSLC consumers and providers for .NET environments that contains the following tools:

- OSLC4NET SDK, which is very similar to the OSLC4J component of Eclipse Lyo: OSLC resources can be modelled as .NET objects, annotated with OSLC attributes, and REST services provided by ASP.NET MVC 4 used for HTTP requests and request handling.
- OSLC Change Management consumer implementation in the form of a test suite.

**5.2.2.5 Conclusion**

Using OSLC to provide data integration between external tools and the OPENCOSS platform would be possible. Two integration scenarios related to process management have been identified. These are:

- Import of externally defined process into OPENCOSS platform.
- Notification of external process execution to the OPENCOSS platform.

To support the import of an externally defined process, a model to model transformation will be perform to transform the external representation of the process model to CCL. The model to model transformation tool could use OSLC to read/create data both with the external process tool and the OPENCOSS platform. The model to model transformation could be executed using the OSLC Automation specification, although this is optional as the model to model transformation could be executed manually. This approach would require the external process tool to expose process models via OSLC ServiceProviders, the OPENCOSS platform to allow creation of process models via an OSLC ServiceProvider, and for the model to model transformation tool to be able to access/create process models using the OSLC ServiceProviders. A domain specification would need to be developed defining the resource types for both the external and CCL process models.

Notification of external process execution to the OPENCOSS platform could be implemented using OSLC. This approach would require the OPENCOSS platform to allow creation/update of process execution status via OSLC ServiceProviders. The external process tool could then create/update the CCL process execution status when necessary using the OSLC ServiceProvider. A domain specification would need developed defining the resource type CCL process execution status.

**5.3 Candidate technologies for the assessment**

**5.3.1 Process Mining**

In this section, we present a modeling approach based on process mining to support assessment.

As process models play a crucial role in large organizations, process mining is proposed to discover, monitor and improve real processes by extracting knowledge from event logs readily available in today's systems. There are mainly three types of process mining:

- Discovery: The discovery approach is to extract a process execution log from an event log without using any a-priori information.
• Conformance: The conformance approach is to check if the process execution log extracted from an event log conforms to a given process model. In OPENC OSS, we would only use this type of process mining.

• Enhancement: The enhancement approach is to extend or improve an existing process execution model using event logs.

Figure 39 An overview of multiple level conformance checking

In OPENC OSS, one of our goals is to check whether the process description conforms to safety standards. As shown in Figure 39, the equivalence mapping between level 1 and level 2 is our core objective. The reference framework models or baseline models in level 2 will be described using CCL metamodels (reference framework metamodel, baseline metamodel, etc.). The project process models will be defined in our process metamodel. When industrial partners use those project process models in practise, event logs will be generated. Then a process model will be updated and the execution date will be stored in activity class objects. Although current safety standards do not force industry to check the compliance at the event level (level 0), we could treat the compliance argument between level 1 and level 2 as a higher level of conformance checking. Then we could check if reality, as recorded in project process models, conforms to the baseline models built according to safety standards and vice versa. Therefore, conformance process mining could be used for our purpose.
In OPENCOSS, the input for conformance checking between level 0 and level 1 in figure 1 will be both an event log and a process model. Figure 2 shows how an event log can be replayed to detect and quantify the commonalities and discrepancies between the behaviour recorded in an event log and the behaviour of a process model. After that, behaviour that does not fit will be highlighted in the visualization of both event log and process model. Then stakeholders could decide whether these non-conformances should be tackled or not. Besides, the conformance checking itself can also be an evidence for process compliance argumentation.

From the implementation point of view, a generic open-source framework, called ProM\(^4\), could be applied for implementing process mining tools. We could use conformance checker plug-in of ProM for conformance checking in OPENCOSS. It provides three views for conformance checking results:

- **Fitness analysis;** it measures the fitness degree between the process specification and the execution instances.
- **Behavioral appropriateness;** it analyzes and detects the unfit or unmatched behavior between process models and event logs, which also provides a way of visualizing the activities relations in process models.
- **Structural appropriateness;** as there could be different ways to express the same behavior using the same modeling language, the structural appropriateness could evaluate if the process model is well formulated or conforms to the design guidelines.

Note that there are several limitations of using process mining techniques in OPENCOSS. The event logs and process models should be structured in some fixed format. If not, a number of manual works will be involved. For example, currently, the process modelling language supported by ProM, such as Petri Net and BPMN, is limited. As a part of CCL, we will design our own process metamodel to describe baseline models.

\(^4\) [http://www.promtools.org/prom6/](http://www.promtools.org/prom6/)
or project process models. Therefore, in the future, we may need to develop new plug-ins for ProM, which will bring a big challenge.

5.3.2 Semantic ontology

Here we shortly present another approach to performing assessment.

This approach is in general based on an idea to use OWL ontologies and automatic reasoning for assessment of the certification process. It would be up to an automatic reasoner to decide if all the predicates expressed in OWL are met and thus correctly classify project as sticking to the certification requirements.

The OWL (OWL 2 DL) is a decidable fragment of first order predicate logic, with some decidable extensions that go beyond first order logic.

Here are some basic characteristics of OWL-based ontologies:

- Consist of classes, individuals and object/data properties
- Open World Assumption (we cannot assume something does not exist until it is explicitly stated that it does not exist; because something has not been stated to be true, it cannot be assumed to be false)
- We cannot assume that individual is not a member of a particular class simply because it has not been asserted so
- Different ways of creating taxonomy – disjunctions, subclassing (subclassing can be multi-based)
- Classification can be based on e.g. property type, domains and ranges, cardinality
- Properties can be inverse, transitive, symmetric, asymmetric, reflexive, irreflexive

To illustrate how we can drive assessment process using OWL and automatic reasoning, let’s evaluate the following example based on a simplified abstraction of a part of the CCL model. It shows a simple scenario in which we have a claim that all unit tests for all the artefacts must have status set to “passed”.

This example is created using the Protégé editor. It is a popular ontology editor and knowledge-based framework, part of the Protégé platform which allows modeling ontologies using desktop or web clients. Protégé can work with OWL, RDF/RDFS and XML Schema. Its plugin-based architecture allows to provide additional functionality by third-party provider – e.g. reasoners, editor views etc. For details see Protégé home page: protege.standford.edu.
Figure 41 Protégé ontograf example

Figure 42 shows a Protégé’s ontograf view of our simplified domain with reasoning already applied. Here’s a selected subset of property assertions (only a subset of these is presented for simplicity):

- Individuals with property hasConfidenceImpactType equals to supportConfidenceImpactType are classified as SupportingPieceOfEvidence
- Individuals with property hasConfidenceImpactType equals to challengeConfidenceImpactType are classified as NotSupportingPieceOfEvidence
- If any Claim has any individual of NotSupportingPieceOfEvidence connected through the hasPieceOfEvidence property (inverse of forClaim), this Claim is classified as UnsupportedClaim

The diagram shows the situation in which after invoking the reasoner all PieceOfEvidence individuals are classified as SupportingPieceOfEvidence, resulting in passedAllUnitTestsClaim not being classified as UnsupportedClaim, thus in this simple scenario we can say that passedAllUnitTestsClaim meets the requirements.

Let’s see how things change when one of the PieceOfEvidence reports failed tests, thus claiming hasConfidenceImpactType equals to challengeConfidenceImpactType:
As you can see, in this situation the reasoner automatically classifies this PieceOfEvidence individual as a NotSupportingPieceOfEvidence, thus classifying passedAllUnitTestsClaim as UnsupportedClaim.

This example is a very simplified one, but it shows how different the approach to problem solving is. You carefully create the knowledge base, provide constraints and relation definitions and then it is up to reasoner to decide upon what observations about current state of the model are made.

Conclusion

This approach to drive assessment process is interesting and very powerful, but achievement of good results depends heavily on the quality of ontologies created. After further research, we are reluctant to suggest semantic technologies as a backing technology for OPENCOSS. It would have to be applied to all OPENCOSS building blocks and other project participants are already moving towards more traditional approaches. Limited knowledge of semantic technologies and no real experience with these are too risky with such a time constraints that OPENCOSS is facing.
6 Conclusion

D7.3 has presented the current version of the requirements and design for the compliance-aware service infrastructure, in particular how this assessment can be performed and how the service can interact with and benefit from external process management tools.

The deliverable has demonstrated the deep relationship between the Common Certification Language and the tooling aspects of OPENC OSS.

Given the iterative approach used by the OPENC OSS project, this document is meant to evolve as new needs or issues are discovered, or the CCL is refined.

Finally, several candidate technologies have been presented. Now we need to prototype the actual use of those technologies in the development phase.