Specification of the evidence management service infrastructure

D6.3

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP6: Evolutionary Evidential Chain</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>9 December 2013</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Jose Luis de la Vara (Simula Research Laboratory)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:jdelavara@simula.no">jdelavara@simula.no</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>Fabien Belmonte, Laurent Pitot-de-la-Beaujardiere</td>
<td>ALSTOM Transport</td>
</tr>
<tr>
<td>Giorgio Tagliaferri, Vincenzo Manni</td>
<td>RINA Services SpA</td>
</tr>
<tr>
<td>Martijn Klabbers, Luc Engelen</td>
<td>Eindhoven Univ. of Technology</td>
</tr>
<tr>
<td>Thierry Ventadour</td>
<td>Inspearit (before DNV ITGS)</td>
</tr>
<tr>
<td>Janusz Studziszba, Michał Siemaszko, Magdalena Gniewek, Dariusz Oszczędłowski</td>
<td>Parasoft SA</td>
</tr>
<tr>
<td>Jose Luis de la Vara, Mehrdad Sabetzadeh, Sunil Nair, Rajwinder Kaur Panesar-Walawege</td>
<td>Simula Research Laboratory</td>
</tr>
<tr>
<td>Huáscar Espinoza, Ángel López, Idoya del Río, María del Carmen Palacios, Alejandra Ruiz</td>
<td>TECNALIA Research &amp; Innovation</td>
</tr>
<tr>
<td>Alberto Melzi</td>
<td>Centro Ricerche Fiat S.C.p.A.</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>2013-02-01</td>
<td>Document creation and initial ToC</td>
</tr>
<tr>
<td>V0.2</td>
<td>2013-05-06</td>
<td>Structure update</td>
</tr>
<tr>
<td>V0.3</td>
<td>2013-05-10</td>
<td>Update by Parasoft (Tool analyses)</td>
</tr>
<tr>
<td>V0.4</td>
<td>2013-05-20</td>
<td>Update by Simula (Background)</td>
</tr>
<tr>
<td>V0.5</td>
<td>2013-05-22</td>
<td>Update by AdaCore (Qualifying Machine)</td>
</tr>
<tr>
<td>V0.6</td>
<td>2013-10-08</td>
<td>Update by TECNALIA (Tool Components)</td>
</tr>
<tr>
<td>V0.7</td>
<td>2013-10-14</td>
<td>Deliverable update and review</td>
</tr>
<tr>
<td>V1.0</td>
<td>2013-10-15</td>
<td>Deliverable finalization</td>
</tr>
<tr>
<td>V1.1</td>
<td>2013-11-18</td>
<td>Deliverable update after review</td>
</tr>
<tr>
<td>V1.2</td>
<td>2013-12-09</td>
<td>Deliverable update after review</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

**Abbreviations** ................................................................................................................................. 7

**Executive Summary** .......................................................................................................................... 8

1 **Introduction** ..................................................................................................................................... 9
    1.1 Scope and Purpose ......................................................................................................................... 9
    1.2 Relationship with other Deliverables .......................................................................................... 10
    1.3 Structure of the Document ........................................................................................................... 11

2 **Background** ..................................................................................................................................... 12
    2.1 Requirements for Evidence Management ................................................................................... 12
    2.2 Evidence Management Components .......................................................................................... 12
    2.3 The Notion of Safety Evidence .................................................................................................... 13
    2.4 Safety Evidence Traceability ........................................................................................................ 14
    2.5 Commonalities in Safety Evidence Management among Domains .............................................. 18
    2.6 Existing Tools and Technologies ................................................................................................. 18
    2.7 Conceptual Needs for Evidence Management .............................................................................. 21

3 **Data Model for Evidence Management** ......................................................................................... 23

4 **Component Specification** ............................................................................................................... 26
    4.1 Data Manager Server .................................................................................................................... 26
        4.1.1 Data Storage .......................................................................................................................... 27
        4.1.2 Data Change Management .................................................................................................... 27
    4.2 Evidence Manager Server ............................................................................................................. 28
        4.2.1 Evidence Characteristics Manager ....................................................................................... 28
        4.2.2 Evidence Reporter .................................................................................................................. 29
        4.2.3 Evidence Event Manager ....................................................................................................... 29
        4.2.4 Evidence Analyser .................................................................................................................. 29
        4.2.5 Evidence Compliance Manager ............................................................................................. 30
        4.2.6 Evidence Exporter .................................................................................................................. 30
        4.2.7 Evidence Importer .................................................................................................................. 30
    4.3 Evidence Manager Client ............................................................................................................... 30
        4.3.1 GUI Evidence Editor ............................................................................................................... 30
        4.3.2 GUI Evidence Analyser ......................................................................................................... 30
    4.4 Traceability Manager Server .......................................................................................................... 30
        4.4.1 Trace Link Engine .................................................................................................................. 31
        4.4.2 Trace Rule Engine .................................................................................................................. 31
        4.4.3 Impact Analyser ..................................................................................................................... 31
    4.5 Traceability Manager Client .......................................................................................................... 31
        4.5.1 GUI Trace Editor .................................................................................................................... 31

5 **Candidate Technologies** .................................................................................................................. 32
    5.1 The Qualifying Machine ................................................................................................................. 32
    5.2 EMF ................................................................................................................................................ 33
    5.3 Semantic Technologies ................................................................................................................... 34
        5.3.1 Advantages of Using Semantic Web in the OPENCOSS Platform ........................................... 34
        5.3.2 Draft of Implementation .......................................................................................................... 35
        5.3.3 Disadvantages and Risks of Using Semantic Web in OPENCOSS .......................................... 35

6 **Conclusion** ...................................................................................................................................... 36
References .......................................................................................................................................................... 37

Appendix A. Tools that Could Be Reused for Evidence Management .............................................................. 38

Appendix B. Detailed Analysis of EMF for Evidence Management ................................................................ 53
   B.1 EMF Technology Core ............................................................................................................................... 53
   B.2 Evidence Metamodel in Eclipse ............................................................................................................. 53
   B.3 EMF Client Platform ............................................................................................................................... 54
   B.4 EMF Store ................................................................................................................................................ 55
   B.5 Using EOperation for validation in EMF ............................................................................................... 56
   B.6 EMF Validate feature .............................................................................................................................. 56
   B.7 EMF Validation Framework .................................................................................................................... 57
   B.8 UNICASE ................................................................................................................................................ 58
   B.9 EMFTrace ................................................................................................................................................. 59

Appendix C. Analysis of Semantic Technologies for Evidence Management ................................................... 61
   C.1 Introduction ............................................................................................................................................. 61
   C.2 Implementing OPCOSS with Semantic Technologies .......................................................................... 61
   C.3 Technical Background .......................................................................................................................... 64
      C3.1 Metadata and Why Tags/Keywords are not Sufficient .................................................................... 64
      C3.2 Types of Repositories ...................................................................................................................... 66
      C3.3 Why A Semantic Repository .......................................................................................................... 67
      C3.4 Benefits of Semantic Technologies ............................................................................................... 69
   C.4 Glossary ................................................................................................................................................... 70
   C.5 References .............................................................................................................................................. 74

Appendix D. Questionnaire for Survey on Impact Analysis .................................................................................. 76
List of Figures

Figure 1. Functional decomposition for the OPENCOSS tool platform .......................................................... 10
Figure 2. Components for evidence management .......................................................................................... 13
Figure 3. Basic model of safety evidence ..................................................................................................... 14
Figure 4. SafeTIM ...................................................................................................................................... 17
Figure 5. Evidence taxonomy .................................................................................................................... 20
Figure 6. Artefact metamodel ..................................................................................................................... 22
Figure 7. Evidence metamodel .................................................................................................................. 22
Figure 8. Evidence management-specific data model .................................................................................. 23
Figure 9. OPENCOSS Tool Components for Evidence and Traceability Management .......................... 27
Figure 10. Illustration of the DataManagerServer operation principles .................................................... 28
Figure 11. Illustration of the EvidenceManagerServer operation principles ............................................. 29
Figure 12. Qualifying Machine input metamodel ......................................................................................... 32
Figure 13. Qualifying Machine web-based client ......................................................................................... 33
Figure 14. Evidence metamodel editor ....................................................................................................... 54
Figure 15. Evidence model ......................................................................................................................... 55
Figure 16. Artefact property editor ............................................................................................................... 55
Figure 17. EMF Validation Framework ........................................................................................................ 58
Figure 18. UNICASE .................................................................................................................................. 59
List of Tables

Table 1. Overall motivation for safety evidence traces ................................................................. 16
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>CASE</td>
<td>Computer-Aided Software Engineering</td>
</tr>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>CDO</td>
<td>Connected Data Objects</td>
</tr>
<tr>
<td>CENELEC</td>
<td>Comité Européen de Normalisation Electrotechnique (European Committee for Electrotechnical Standardization)</td>
</tr>
<tr>
<td>DoW</td>
<td>OPENCOSS Description of Work</td>
</tr>
<tr>
<td>DX.Y</td>
<td>OPENCOSS 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>GUI</td>
<td>Graphical User Interface</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>MDE</td>
<td>Model-Driven Engineering</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OCL</td>
<td>Object Constraint Language</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</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>SKOS</td>
<td>SKOS Simple Knowledge Organization System</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TX.Y</td>
<td>OPENCOSS task X.Y</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>WP</td>
<td>OPENCOSS work package</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XSL</td>
<td>eXtensible Stylesheet Language</td>
</tr>
</tbody>
</table>
Executive Summary

This document (D6.3) is the third deliverable of WP6. This WP aims to define a safety certification management infrastructure for an evolutionary evidential chain. The overall goal of D6.3 is to specify the detailed architecture for the safety certification management infrastructure, in particular for its evidence management services.

Several inputs and aspects have been taken into account for specifying the evidence management infrastructure. Apart from the requirements for evidence management and the building blocks of the overall OPENCOSS tool platform, the notion of safety evidence, safety evidence traceability needs, and existing, related tools have been analysed in depth. In addition, we have participated in the development of the evidence-related parts of the CCL, thus in the analysis of evidence-related conceptual needs, and compared the commonalities in evidence management among different domains.

In addition to a data model for evidence management, D6.3 presents the detailed specification of two main components of the OPENCOSS tool platform architecture: evidence characterization and traceability editor, and evidence analysis manager. For each component, a static view is specified, as well as the interaction between their building blocks. An analysis of candidate technologies for the implementation of the OPENCOSS evidence management infrastructure is also presented.

The results of this deliverable will serve as basis for T6.4 (Implementation of the evidence management service infrastructure), which will develop the design presented in this deliverable. D6.3 will also used as input for future versions of D3.2 (Integration requirements and test plan) in order to specify integration and unit test cases. It is also expected that the deliverable will evolve as OPENCOSS progresses, and new needs regarding evidence management are discovered, and some decisions are made. For example, safety evidence change impact analysis has to be further studied in the near future. Therefore, new, updated versions of the deliverable will be released.
1 Introduction

1.1 Scope and Purpose

Safety assurance and certification are amongst the most expensive and time-consuming tasks in the development of safety-critical embedded systems. European innovation and productivity in this market is curtailed by the lack of affordable (re)certification approaches. Major problems arise when evolutions to a system entail reconstruction of the entire body of certification arguments and evidence. Further, market trends strongly suggest that many future embedded systems will be comprised of heterogeneous, dynamic coalitions of systems of systems. As such, they will have to be built and assessed according to numerous standards and regulations. Current certification practices will be prohibitively costly to apply to this kind of embedded systems.

The OPENCOSS project aims to devise a common certification framework that spans different vertical markets for railway, avionics and automotive industries and to establish an open-source safety certification infrastructure (hereafter referred to as OPENCOSS tool platform). The infrastructure is being realised as a tightly integrated solution, supporting interoperability with existing development and assurance tools. The ultimate goal of the project is to bring about substantial reductions in recurring safety certification costs, and at the same time increase product safety through the introduction of more systematic certification practices. Both will boost innovation and system upgrades considerably.

WP6 is concerned with defining the part of the OPENCOSS platform that will support an evolutionary chain of certification evidence. A chain of certification evidence is a set of pieces of evidence that are related (e.g., the agent that has created a requirements specification, the test derived from the requirements, the agent that executed the tests, the report where the tests results are documented, etc.). By evolutionary, we mean that a chain of evidence can suffer changes (e.g., a requirement is changed), thus evolve. As a result, the chain of evidence might not be adequate anymore for safety certification (e.g., the related test cases might have to be updated).

Therefore, WP6 needs to provide the necessary methods and supporting tools for the management of the evidence used in the safety certification of critical systems and also to pay particular attention to situations in which the evidence changes or evolves. When evidence changes, it must be possible to determine whether the set of evidence for a system is still adequate or if new evidence and thus re-execution of certification-related activities is necessary. This can also apply to reuse of evidence between standards and domains. That is, how adequate a chain of evidence compliant with a given standard is to comply with another standard (in the same or in a different domain).

This document (D6.3) is the third deliverable of WP6. Its overall goal is to specify the detailed architecture for the safety certification management infrastructure, in particular for the evidence management services. The infrastructure is expected to include services for certificate (i.e., evidence) lifecycle management, traceability management, and management of the compositional rules defined in WP5.

D6.3 corresponds to a refinement of the overall architecture for the OPENCOSS tool platform (Figure 1). More specifically, and as explained in Section 2, components targeted at evidence management are further specified in this deliverable. Such components are responsible for the implementation of the requirements for evidence management in the OPENCOSS tool platform.

This deliverable is strongly based on the results provided in previous deliverables, which presented, for instance, requirements for evidence management and an overview of the overall OPENCOSS tool platform architecture. Nonetheless, several new activities have been necessary. The main activities have been (1) to
further investigate safety evidence traceability needs, (2) to study in depth tools that could be adapted or reused in the OPENCOSS tool platform, and (3) to collaborate in the development of the evidence-related CCL parts.

Figure 1. Functional decomposition for the OPENCOSS tool platform

1.2 Relationship with other Deliverables

D6.3 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 D6.3 with them, are the following:

- D2.3 (OPENCOSS platform architecture) presents the overall OPENCOSS tool platform architecture, whose evidence management-related components are specified in detail in WP6.
- 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 evidence will be managed in WP6.
- D6.2 (Detailed requirements for evidence management of the OPENCOSS platform) includes the requirements that must be implemented in WP6 and thus satisfied by the evidence management components.
- D6.4 (Specification of adapters to development and safety assurance tools) is responsible of the specification of the evidence management services targeted at the integration of the OPENCOSS tool platform with external tools, such as those used in the development of safety-critical systems.
• D6.5 (Intermediate implementation of the evidence management service infrastructure) and D6.6 (Implementation of the evidence management 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 evidence management is based. Section 3 presents a data model for evidence management. The components are specified in Section 4. Section 5 describes the main candidate technologies for development of the evidence management services. Section 6 presents our conclusions. Appendices A to C provide details about several tooling aspects mentioned but not described in the rest of sections and for which no references (e.g., in the form of technical reports) are available. Finally, Appendix D shows the current status of a questionnaire under development for a survey on impact analysis.
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 D6.3

The following subsections outline the requirements for evidence management and the evidence management components, discuss the notion of safety evidence, analyse safety evidence traceability, introduce the commonalities in safety evidence management among the domains at which OPENCOSS is initially targeted, explain the analysis of existing technologies and tools that has been performed, and present the evidence management-specific conceptual needs of OPENCOSS.

2.1 Requirements for Evidence Management

In D6.2, five main functional areas were defined for specifying component level requirements related to evidence management of the OPENCOSS tool platform:

- **Evidence storage**, concerned with the determination, specification, and structuring of the evidence items of an assurance project.
- **Evidence traceability**, concerned with the specification and adequate maintenance of traceability between evidence items of an assurance project.
- **Evidence evaluation**, concerned with the assessment of the completeness and adequacy of the body of evidence of an assurance project, and of specific criteria defined for evaluation of individual evidence items.
- **Evidence change impact analysis**, concerned with the identification and analysis of possible effects resulting from changes in the body of evidence of an assurance project.
- **Integration with external tools**, concerned with the possibility of importing and exporting information from and to external tools, and information synchronization with them.

The analysis of these functional areas resulted in the specification of a set of 111 component level requirements.

2.2 Evidence Management Components

The overall OPENCOSS tool platform architecture (Figure 1) contains a module for management of the full lifecycle of evidence items and evidence chains. This includes evidence traceability management and impact analysis. In addition, the module communicates with external engineering tools that manage evidence items (requirements management, implementation, V&V, etc.).

As shown in Figure 2, the module consists of three components:

- **Evidence Characterisation and Traceability Editor**, which deals with evidence storage and evidence traceability;
- **Evidence Analysis Manager**, which deals with evidence evaluation, evidence change impact analysis, and;
- **External Tool Integration Manager**, for integration with external tools.

This deliverable deals with the specification of the two first components. Specification of the third one is in the scope of D6.4.
2.3 The Notion of Safety Evidence

Although much research has been conducted on safety evidence [17], past works have not made an explicit and clear distinction between an artefact and its use as evidence. Therefore, the notion of safety evidence remains difficult to understand and safety evidence management can be hindered.

In general, evidence is the available body of facts or information indicating whether a belief or proposition is true or valid [16]. In relation to safety, we define safety evidence (also referred to as safety-related evidence) as the artefacts that contribute to gain confidence in the safe operation of a system. Safety evidence also aims to show fulfilment of the requirements of a safety standard in the context of safety compliance and certification. A broader analysis of definitions of evidence and of safety evidence can be found in [22].

Demonstration of safety compliance is usually costly and time-consuming, and can be very challenging [17][18]. Firstly, system suppliers have to collect artefacts such as hazard specifications, test results, and activity records. This can be hindered because of difficulties in understanding a safety standard or in determining and gaining confidence in safety evidence. Secondly, practitioners usually have to manage large quantities of evidence and structure it adequately to show compliance. Otherwise, its sheer volume and complexity can jeopardize safety certification.

These problems can be exacerbated if a distinction between artefacts and their use as evidence is not made. Despite the fact that artefacts and safety evidence are not the same, we are not aware of any model that has explicitly and clearly differentiated them. For example, using an artefact as evidence for several safety claims implies the existence of emerging properties and relationships, different for each claim, such as the confidence in the evidence for supporting a claim. Therefore, and based on conceptual modelling principles (e.g., [20]), artefacts and pieces of evidence are different concepts.
Furthermore, these issues can lead to certification risks [1]. In other words, a system supplier might not be able to develop a safe system, show that a system can be deemed safe, or make a third party gain confidence in system safety. An unclear distinction between an artefact and its use as evidence can also hinder safety evidence management. For example, it can be difficult to determine what evidence information can be reused between projects, or the effects of safety evidence change.

As a solution to problems outlined above, we propose a (basic) model for safety evidence (Figure 3): an artefact is used as a piece of (safety) evidence for some claim, which aims to make someone gain confidence in system safety. Both an artefact and a claim are necessary for having evidence (i.e., for having evidence of the validity of a claim). In essence, safety evidence can be seen as a role of an artefact.

Examples of the kind of artefacts to which we refer are:
- Data items in DO-178C [21]
- Input and output documents in EN50128 [3]
- Information required and produced in IEC61508 [11]
- Work products in ISO26262 [14]

D4.4 presents more information in relation to the application of the model in the scope of the CCL.

In the scope OPENCOSS, WP6 is responsible of the management of the artefacts used as safety evidence in an assurance project. Among other purposes, such artefacts can be used for argumentation (e.g., in the scope of WP5) and for compliance assessment (e.g., in the scope of WP7).

![Figure 3. Basic model of safety evidence](image)

### 2.4 Safety Evidence Traceability

In software engineering, traceability is the degree to which a relationship can be established between two or more products (aka artefacts) of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another [12]. We define safety evidence traceability as the degree to which a relationship can be established to and from artefacts that are used as safety evidence.

Although traceability for safety-critical systems and more concretely safety evidence traceability have been addressed in past research, no work has provided yet a broad, complete picture of safety evidence traceability needs. Most of the research has only focused on the relationships between the artefacts used as evidence (e.g., [4]) and thus ignored their relationships with other safety assurance assets. The works that have explicitly or implicitly studied other aspects of safety evidence traceability (e.g., the relationship between evidence and argumentation [9]) have only dealt with a limited set of needs for safety evidence traceability. As a result, we decided to perform a detailed analysis of safety evidence traceability, its needs, challenges, and open issues. This analysis refines and extends the work performed in T6.1 (Baseline and requirements of the evidence chain infrastructure).
We have identified five main reasons for keeping safety evidence traceability:

- **Safety assurance**
  A fundamental need for any safety-critical system, regardless of having to comply with some specific safety standard, is to ensure that its hazards have been mitigated (or avoided). Otherwise, system safety might not be achieved. Maintaining traceability is essential to show that hazard mitigations have been developed, validated, and verified. For example, safety requirements can be specified, and they can be later verified by using some formal method or can be tested.

- **Compliance with safety standards**
  In many application domains, safety critical systems must comply with safety standards to allow their operation. Therefore, system suppliers have to show fulfilment of the requirements of the standards. Traceability can be a means for this activity. System suppliers might have to explicitly provide traceability specifications as a part of the information that constitutes evidence of compliance. Indeed, some standards mandate this information (e.g., DO-178C).

- **Change impact analysis**
  Changes in a safety-critical system and thus in its safety evidence are practically inevitable, both during its development and after its delivery. Practitioners must ensure that such changes will not have any undesired effect in system safety and in the body of safety evidence. Therefore, such changes have to be managed adequately. For example, it is necessary to assess how a change in a piece of evidence might affect others. Safety evidence traceability is necessary to perform such an impact analysis, in order to identify the potential consequences of a change or to estimate what needs to be modified to accomplish a change.

- **Evidence reuse**
  Reuse of a safety-critical component (or system) and thus of its evidence is important in industry, mainly in order to increase the return on investment in component development and to decrease system cost. However, it must be ensured that evidence reuse is adequate, or that a change in one use of a piece of evidence is propagated to other uses when considered necessary. Safety evidence traceability supports evidence reuse and the execution of the associated required activities.

- **Project management**
  Project management information such as that related to cost, effort, or degree of compliance is essential to make informed decisions during safety-critical system lifecycle. These decisions can be hard to make without adequate safety evidence traceability. For example, it allows the estimation of the cost of a possible change, and then practitioners could decide whether the change should be finally made or not.

As can be noticed, some of these motivations do not only relate to WP6, but also other WPs. For example, evidence reuse (and composition) is one of the areas under study in WP5.

The analysis of these needs resulted in the identification of nine main traces to maintain (Table 1). Such traces have been also included in SafeTIM, a traceability information model for safety evidence (Figure 4). The definitions of its entities and their relationships can be found in D4.4. In some cases, the entities and the relationships have been renamed or extended. In this sense, SafeTIM can be regarded as a fragment of the CCL focused on safety evidence traceability, but that has been refined to the specification of the whole CCL and further linked to the conceptual needs of other OPENCOSS areas (e.g., argumentation).

The principal challenges for safety evidence traceability correspond to:

- **Vast amount of artefacts to trace**
  An aspect that distinguishes system development in the safety-critical domain from other domains is the very high number of artefacts to create, maintain, and trace. Management of vast amount of data has always been a challenge for information systems, but it becomes even more demanding in the safety-critical domain due to regulatory compliance. In addition to the challenges inherent
to traceability, practitioners can have serious problems to ensure, for instance, consistency of safety evidence traces and thus of evidence. Guidance and tool support are necessary.

Table 1. Overall motivation for safety evidence traces

<table>
<thead>
<tr>
<th>Safety evidence trace</th>
<th>Motivations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between artefacts</td>
<td>Assu Comp CIA ERes PM</td>
</tr>
<tr>
<td>Between safety evidence and claims</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Between safety evidence and arguments</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Between artefacts and types of artefacts</td>
<td>X</td>
</tr>
<tr>
<td>Between pieces of evidence in relation to a same claim</td>
<td>X X</td>
</tr>
<tr>
<td>Between version of an artefact</td>
<td>X X</td>
</tr>
<tr>
<td>Between reuses of an artefact</td>
<td>X X</td>
</tr>
<tr>
<td>Between artefacts and activities</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Between artefacts/safety evidence and provenance</td>
<td>X X X</td>
</tr>
</tbody>
</table>

Assu: safety assurance; Comp: compliance with safety standards; CIA: change impact analysis; ERes: evidence reuse; PM: project management

- **Artefacts can be located in many different locations**
  Nowadays, it is usual that different parts of a safety-critical system are developed in different locations around the world. This also commonly means that the artefacts used as evidence are in locations different to where, for instance, a safety case is developed. This situation causes new problems in safety evidence traceability, such as the coordination of work among distributed development teams and difficulties to ensure that the result of this work is consistent and will not pose any certification risk.

- **Artefacts are created with and stored in different tools**
  System suppliers usually have a tool-chain for system development, consisting of tools targeted at, for instance, requirements management, system modelling, testing, and configuration management. Seamless integration of these tools for (automatic) safety evidence collection can be difficult or even impossible. Evidence combination can also be hindered because of the heterogeneity in the formats of the artefacts.

- **Confidence in the traces maintained**
  One of the main challenges that both system suppliers and certifiers face for safety certification is the need of having confidence in the traces maintained. Providing traces to and from safety evidence is far from enough, as the people involved in safety assurance and certification must aim to be sure that, for instance, the traces presented are consistent or that they really correspond to the system under assessment.
High effort and cost
Although better traceability practices can lead to a reduction of system development effort and costs, reality is that safety evidence traceability is still a very time-consuming activity that requires the involvement and thoughtful work of several people. As a result, practitioners can end up only dealing with a limited set of traces, usually those mandatory for compliance. However, this might pose certification risks later, or make system change management very expensive. Again, adequate guidance and tool support are very important to face this challenge.

Need for purpose, value-based traceability
In relation to the previous challenge, it is essential that the need for and purpose of safety evidence traceability is clear to those involved in the activity. Otherwise, traceability might not be managed as well as it should be. Practitioners must define and be aware of the value of tracing beyond the scope of a single project. For example, adequate safety evidence traceability can facilitate system reuse and change impact analysis in the future, and thus reduce costs. Although probably not exclusive to safety evidence traceability, we believe that these challenges are especially difficult to solve in the development and assurance processes of safety-critical systems because of their characteristics (e.g., rigorous safety assessments are a must).

Based on thorough analyses of the state of the art (e.g., [2][19]), we think that the most important open issues related to safety evidence traceability are:

To what extent is automated traceability applicable in safety-critical systems?
Techniques for automating the creation, recovery, and maintenance of traces (i.e., automatically executing these activities without human intervention) have been proposed during the last years, especially in relation to requirements management. Although many authors have argued their advantages, the evaluations performed have been based on small data sets and many times with artefacts that do not represent the practice. Therefore, the applicability of the techniques in industry is not clear. In addition, a major issue lies in the reliability of the generated traces and their completeness. Given the rigorous and strict criteria for assurance and certification safety-critical systems, industry might not accept the use of these techniques unless they are improved or their limitations are mitigated.

How can automated traceability improve industrial practice?
It must be investigated how automated traceability can really be applied in industry and thus how it can improve current practices. Fully automated traceability might not be a suitable solution, but tool-assisted traceability might be. For example, a tool could guide a practitioner when having to create and maintain traces, making recommendations or indicating possible inconsistencies. It might also be important to study the suitability and benefits of different visualization techniques.

What granularity should the traced artefacts have?
Finding the right level of granularity of the traces to create and maintain is essential for making traceability purposeful and useful. Highly fine-grained traces can be complex and unrealistic to capture and maintain, while a highly coarse-grained traceability can be of insufficient help (e.g., for change impact analysis purposes). Finding the appropriate level can even be a greater issue when there are large quantities of artefacts and they are poorly structured.

Are safety evidence traceability needs the same in different domains?
Given the current interest in the reuse of safety assurance assets such as safety evidence among different application domains, and that some differences have been found among domains, it is necessary to determine if safety evidence traceability needs are the same in different domains. It is also important to determine if safety evidence traceability practices in one domain are acceptable in another. Otherwise, efforts targeted at providing common solutions might not succeed.

What is the specific semantics of the traces in relation to change impact?
Without identifying the specific semantics of the traces (e.g., in relation to the effect of a change), performing a change impact analysis would result in an explosion of potential impacts, declaring that all the artefacts in a trace slice can be affected. Discovering and specifying the right trace semantics is necessary to increase the accuracy of impact analysis and facilitate it.

- **How can acceptance of new approaches in industry be facilitated?**
  With any new technology, acceptance in industry is a slow and demanding process. We believe that close industry-academia collaboration and evaluations in real settings are necessary to promote adoption in industry of new safety evidence traceability approaches.

### 2.5 Commonalities in Safety Evidence Management among Domains

We analysed the commonalities concerning safety evidence management among different application domains based on a systematic literature review conducted for D6.1 and a survey on the state of the practice conducted for D6.2. Details about the analysis can be found in [17][18].

The main conclusions for the automotive, avionics, and railway domains are as follows:

- Among all the types of safety evidence (Figure 5; taken from D6.1), the only type that has not been reported as used in the three domains is operator competence (automotive domain).
- Hyperlinks have not been reported as used for recording evidence traceability in the automotive domains. Matrices, models, metadata, and naming conventions are used in the three domains.
- Although the frequency of use varies, all the techniques for evidence structuring (structured text, textual templates, argumentation based graphical notations such as GSN, conceptual/information models, process models, and unstructured text) and for evidence assessment (expert judgment recording the rationale, expert judgment without recording the rationale, argumentation, quantitative assessment, qualitative assessment, and checklists) are used in the three domains.
- All the challenges identified in the literature (D6.1) seem to be important for practitioners in the three domains.

Given the commonalities among the three domains, and although the analysis has been performed with abstract information and not with safety-standard specific information, we believe that the provision of a common solution for safety evidence management for the three domains is feasible.

### 2.6 Existing Tools and Technologies

Another important activity that has been performed for D6.3 is to extend the tool analyses presented in D6.1 and D6.2 in order to determine:

- What existing tools can really be adapted or reused in the OPENCOSS tool platform
- The status of the tools (e.g., if they correspond to a finished project and no further development is expected)
- How easy the adaptation or reuse would be (e.g., if tool documentation is available)

The main types of tools that have analysed correspond to:

- Model-based tools (e.g., EMF)
- Traceability and impact analysis tools (e.g., EMFTrace)
- Storage tools (e.g., different kinds of databases)
- Resources for tool integration (e.g., semantic technologies)

Although many tools exist, the two overall conclusions from the analysis are as follows:
1. Many tools are not in a good state to be adapted or reused. They are no longer developed, do not have enough documentation, or are only research prototypes whose use in real projects can be hindered (e.g., because of scalability issues).
Figure 5. Evidence taxonomy
2. For some technologies that seem promising (e.g., semantic technologies), the OPENCOSS consortium lacks expertise and experience in their use. Therefore, some learning and training period will be necessary if these technologies are finally adopted.

Details about the analysis are provided in Appendices A, B, and C. A summary of what we consider at this moment as candidate technologies for implementation is presented in Section 5. Such technologies correspond to The Qualifying Machine, some EMF-based projects, and semantic technologies.

2.7 Conceptual Needs for Evidence Management

Last but not least, great effort has been spent for D6.3 in relation to the analysis and specification of conceptual needs for evidence management. Some of its results have been presented in the previous sections (e.g., the basic model of safety evidence and SafeTIM), and others are presented in D4.4. Cooperation, collaboration, and communication has been constant between T6.2 and T4.2 in order to:

- Develop the evidence-related CCL parts
- Align the content of D6.3 with the CCL
- Determine the WP6-specific conceptual needs

Such needs correspond to requirements that WP6 must meet but are out of the scope of the CCL. For example, WP6 needs to manage the changes made in the body of evidence and the actions necessary to handle the changes in a way that allows practitioners to work with a consistent and valid body of evidence. These specific needs are presented in more depth in the next section.

With regard to other steps towards determining the conceptual needs for safety evidence management:

- We have performed an in-depth analysis of existing compliance and evidence (meta)models (see, for instance, [7] for details).
- The main model has been the evidence metamodel proposed in SACM [15]. We have analysed how SACM concepts map into CCL needs, paying especial attention to those concepts related to artefacts and their use as evidence. The main conclusions from the analysis are:
  - Although promising and many of its concepts have been adopted and adapted for OPENCOSS needs, SACM still requires some improvements to facilitate its understanding and use.
  - A clear, explicit distinction between artefacts and their use as evidence is not made.
  - Some concepts need clarification. For example, the notion of evidence assertion can be ambiguous. Steps towards its clarification have already been made (e.g., [22]).
  - Some concepts seem to be redundant.

As further examples of the results of the work on the conceptual needs, Figure 6 and Figure 7 show (what we have called) an artefact metamodel and an evidence metamodel in the form of ecore diagrams [10], respectively. These metamodels have been refined and are described in detail in D4.4, and will be further developed in WP4. We also still have to decide in OPENCOSS what aspects of these metamodels will be implemented in each WP. For example, the evidence metamodel might finally be responsibility of WP5.
Figure 6. Artefact metamodel

Figure 7. Evidence metamodel
3 Data Model for Evidence Management

This section presents WP6-specific data needs by means of a data model (Figure 8) in the form of an.ecore diagram [10]. As explained above, this model corresponds to an adaptation and extension of the CCL for evolutionary evidence management purposes. That is, the concepts of the model are necessary for meeting the requirements specified in D6.2 that are out of the scope of the CCL.

![Figure 8. Evidence management-specific data model](image)

The classes shown in Figure 8 that are part of the CCL and thus are described in D4.4 are:
- Assurance Asset Event
- Participant

Other classes referred to in the description of the model and that are described in D4.4 are:
- Assurance Asset
- Describable Element
- Recordable Element

The rest of classes are described as follows.

**Traceable Element**
This class corresponds to an assurance asset that can be traced, changed, and/or affected by a change (e.g., an artefact).

*Attributes*
- **ID**: string
  The data that identifies a traceable element
- **Status**: Status Type
  The status of a traceable element

**Change**
This class corresponds to the information regarding the modification or revocation of a traceable element.

*Superclass*
- Recordable Element

*Attributes*
- **confirmed**: Boolean
  Information about whether a change has been confirmed (and thus executed) or not
• proposal time: data
  The moment at which a change is proposed
• execution time: data
  The moment at which a change is executed

Relationships
• in Traceable Element [1]
  The traceable element in which a change is made
• impact Traceable Element [0..*]
  The traceable elements affected by a change
• proposed by Participant [0..1]
  The participant that proposes a change
• confirmed by Participant [0..1]
  The participant that confirms a change
• managed in Action [0..*]
  The actions with which a change is managed
• triggers Assurance Asset Event [0..*]
  The assurance asset events that are triggered as a result of a change

Action
This class corresponds to the steps that have to be taken as a result of a change so that a body of evidence is valid.

Superclass
• Describable Element
• Recordable Element

Attributes
• creation time: data
  The moment at which an action is created
• finished: Boolean
  Information about whether an action has been completed or not
• finalisation time: date
  The moment at which an action is completed
• execution description: string
  A description of how an action has been performed

Relationships
• for Traceable Element [1]
  The traceable element at which an action is targeted
• created by Participant [0..1]
  The participant that creates an action
• assigned to Participant [0..1]
  The participant responsible for an action
• takes Action Suggestion [0..1]
  The action suggestion followed in an action

Action Suggestion
This class corresponds to the recommendations and thus possible ways in which an action can be performed.

Superclass
• Describable Element

Attributes
• execution description: string
A description of how an action suggestion should be executed

**Relationships**
- **for Change [0..*]**
The changes in which an action suggestion is followed

**Status Type**
This enumeration corresponds to the specification of the validity of a traceable element

**Literals**
- **To Validate**
The traceable element needs to be checked in order to determine if it is valid or not
- **Valid**
The traceable element is valid
- **Invalid**
The traceable element is not valid

The specification of data needs for evidence management will be extended (e.g., by specifying the lifecycles of the classes) once WP6 requirements are refined. This is expected to be done in the upcoming months, once a first iteration of prototypes implementation is completed, and thus some requirements are clarified, new requirements are discovered, and requirements are prioritized. Methods will also be specified for the classes.

Last but not least, and as mentioned in Section 6 and shown in Appendix D, a survey on safety evidence change impact analysis is currently being designed. Its results might lead to the refinement of the scope of the impact analysis aspects to address in WP6.
4 Component Specification

As described in Section 2.2, there are two components to be specified in the scope of this deliverable:

- Evidence Characterisation and Traceability Editor, and;
- Evidence Analysis Manager.

These components have been defined in D2.3 and represent a first approach at a high-level of the functional decomposition of the Evidence Management OPENCOSS module. After a more detailed design of this module, a different decomposition has been done, as explained and justified in this section.

The new decomposition is done at two levels:

(a) Layered Decomposition of Tool Services

As a technology-geared decision, we distinguish three tool component layers:

- **Data Management**, including data storage and change management services;
- **Core Components**, covering the main functionality of OPENCOSS tool components, and;
- **GUI Client** components, which intend to disaggregate services that can be at some point distributed in separate computing nodes (e.g., when using Web services to access the OPENCOSS platform).

(b) Functional Decomposition of Tool Services

This decomposition distinguishes two kinds of component groups:

- Components involved in the **management of artefacts** used as evidence, and;
- Components for the **traceability** management of any OPENCOSS information asset (e.g., argumentation, evidence, or process-related assets).

The main rationale of this decomposition is that the traceability functionality will be used not only for artefacts traceability but also for other OPENCOSS modules such as Assurance Process Management and Argumentation Management.

Figure 9 provides an overview of the tool components in the scope of this deliverable. Note that we also include components associated to D6.4. This aims to provide the general overview of the associated tool components.

The following subsections describe these tool components and their interaction.

4.1 Data Manager Server

The **DataManagerServer** component is an infrastructure tool component used by every OPENCOSS module to manage data into a common, scalable, secure, and auditable way. One of the main rationales to disaggregate this tool component is because we might integrate some existing tools already providing this kind of functionality. One example is EMFStore, which is evaluated later in this document. The DataManagerServer must interact with two kinds of entities: data history and data clients. Figure 10 illustrates the general data management principle. Clients modify the data entities in parallel and synchronise with the server. The DataManagerServer keeps a history of the data entities, which can be queried to retrieve historical states and changes between different versions of the entities. At this stage, there are no prescriptions on the need of client-side data versions.

In addition to data storage, the expected functionality is data change management. The two components that support storage and change management are described next.
4.1.1 Data Storage

The DataStorage component shall manage a repository to store, distribute, and collaborate on CCL-based entities (i.e., data and/or models). The DataStorage component shall understand data/models and their structure and, therefore, shall support conflict detection and interactive merging on a data/model level as opposed to on a textual level. The DataStorage component shall support versioning to retrieve any state of models/data.

4.1.2 Data Change Management

The DataChangeManagement component shall manage fine-grained changes registered on the history of models/data (e.g., at the level of model element attributes). The differences shall be retrieved by Data Client Components (Figure 10) in order to compare different models/data states. The DataChangeManagement component shall support tagging to mark special versions and branching to allow concurrent development of models/data.
4.2 Evidence Manager Server

The EvidenceManagerServer is the main business logic component of the Evidence Management service infrastructure. It embeds the services for:

- Evidence characterization;
- Evidence evaluation;
- Evidence change impact analysis, and;
- Integration with external tools.

Figure 11 illustrates the link of the EvidenceManagerServer with other tool components and external artefact repositories. In order to perform Change Impact Analysis, this component gets information from the Traceability Manager Server. The EvidenceManagerServer also interacts with the DataStorageServer to get access to the OPENCOSS evidential information repository.

4.2.1 Evidence Characteristics Manager

The EvidenceCharacteristicsManager manages the specification of the necessary fine-grained models of evidence elements required for detailed safety assurance, compliance management and certification activities supported by the OPENCOSS platform. We call this modelling activity Evidence Characterization. The structure of such evidence characterization provides the basis for logical design of easily-constructed tools for storing, cross-referencing, evaluating and reporting the elements of evidence for safety assurance and certification in OPENCOSS.

Evidence Characteristics are of the different nature:

- General information about the evidential artefacts such as descriptions, versions, concrete files (or other references to their location in external tools), and their relationships to other artefacts or certification assets.
- Artefact properties such as confidence, accuracy, and completeness.
- Parts of an evidential artefact, such as sections of a document, views of a model or artefact items such as Hazards from a Hazard Log or Requirements from a Software Requirements Specification.

This component shall provide the following services:

- Reading and Editing of Evidence Characteristics from the Data Manager Server.
- Creation and Deletion of Evidence Element Items
- Filtering of Evidence Items by Evidence Characteristics criteria
- Creation and Usage of Libraries of Evidence Properties
- Propagation of Evidence Properties from parent evidence to its children or parts.
4.2.2 Evidence Reporter

The EvidenceReporter component provides the reporting functionalities, including:
- Creation of Customized Reports (e.g. evidence lists per project).
- Filtering of Items to be reported
- Evidence Analysis reports
- Evidence Traceability reports
- Evidence Events reports

4.2.3 Evidence Event Manager

The EvidenceEventManager component manages the lifecycle of evidential artefacts. The lifecycle includes any change on the status of an evidence item from creation to internal evaluation or its consolidation for assessment. The EvidenceEventManager component shall support the tagging by predefined types of states and the definition/association of activities/actions (linked to the Process Management tools in OPENCOSS) according to the status of the evidence. For instance, as a result of an evidence evaluation, it can be determined that additional testing must be performed on a given product. The new status must be associated to the triggering of the associated testing activity/action.

4.2.4 Evidence Analyser

The EvidenceAnalyser component provides an Evidence-specific functionality for Change Impact Analysis. This component uses the services provided by the ImpactAnalyser (and hence Traceability functionality) and EvidenceComplianceManager components.

The Change Impact Analysis verifies that a given artefact is consistent with its associated assets (claims, activities, artefacts, etc.). This analysis is achieved by storing the links and dependencies with those assets together with the evidence characteristics. If anything changes in the associated assets, then the artefact may potentially require some rework or to be re-generated. The impact analysis will mark such artefact and ask the user to perform actions on it: confirm that it is still correct despite the change, modify the artefact according to the change, or re-run the activity that created the artefact.
4.2.5 **Evidence Compliance Manager**

The EvidenceComplianceManager supports the measurement of the level of compliance of artefacts regarding standards, rules, regulations, or company-specific practices. While the Process Management tools in OPENCOSS will support the full functionalities for Compliance Management, this component in the Evidence Management infrastructure is related to the mapping of project-specific artefacts and the reference artefacts to comply with at the beginning of a given assurance project. The mapping can be specified by a simple link from a reference artefact to the concrete artefact model or by a more elaborated description on the level of accomplishment of the reference artefact (concept of Partial Map in CCL) or any additional information describing the level of compliance.

4.2.6 **Evidence Exporter**

The EvidenceExporter component supports the creation of data files to be read externally, concerned with evidence characteristics, traceability and analysis information managed by OPENCOSS. This shall support the most common standards formats such as XML, Excel, or database files as supported by the OPENCOSS storage server.

4.2.7 **Evidence Importer**

The EvidenceImporter component supports the import of data files, concerned with evidence characteristics, traceability and analysis as required by the OPENCOSS data structure. This shall support the most common standards formats such as XML, Excel, or database files as supported by the OPENCOSS storage server.

4.3 **Evidence Manager Client**

The EvidenceManagerClient is the component responsible for the presentation layer of the EvidenceManagerServer component. This component provides two groups of User Interfaces: Evidence Edition GUI and Evidence Analysis GUI. The following different kinds of user interfaces must be provided:

- **Navigation**: tree views, filtering, tables for artefact items and their parts.
- **Editor**: forms to edit artefact information.
- **Historical view**: possibility to see old versions of artefacts and their files.
- **Links to/from other OPENCOSS modules**: argumentation links to/from evidence, process links to/from evidence, and reference framework compliance links.

4.3.1 **GUI Evidence Editor**

The GUIEvidenceEditor component supports visualisation and editing of evidence-related information, including evidence characterisation, items filtering, evidence lifecycle and evidence properties.

4.3.2 **GUI Evidence Analyser**

The GUIEvidenceAnalyser component supports visualisation, configuration and execution of different analyses on evidence information, such as change impact analysis, compliance checks, and other consistency checks.

4.4 **Traceability Manager Server**

The TraceabilityManagerServer component provides support for evolution and continuous changes of assurance asset models (e.g., argumentations, artefacts, activities) at different levels of abstraction and by explicitly expressing dependencies between them. Traceability links are well suited to model such dependencies, since they enable analysis such as change impact analysis and other consistency checks.
This component provides the following services:

- Storage of dependency relations between elements of different assurance assets as traceability links.
- Discovery of relations between different assurance asserts.
- Support for the synchronization of assurance assets after changes of related assurance assets (e.g., claims to be changed after changes in artefacts).
- Access to traceability links for inspection, visualization and comprehension.
- Support for specification of traceability rules to identify traceability links.
- Evaluation of relations regarding completeness, consistency and certainty.
- Provision of change impact analysis functionality for any assurance asset.

### 4.4.1 Trace Link Engine

The TraceLinkEngine component is in charge of storing dependency information as traceability links to benefit from their rich semantics. This component shall be based on a Traceability metamodel, including the ability to model many-to-many links, design decisions, and confidence of links. This component shall include the following services:

- Transitive trace links (chain of links through assurance assets).
- Management of status of links: active, corrupted, inactive, etc.
- Multiple types (including their semantic and actions derived from the changes of the source and target of links) and hierarchies of links.

### 4.4.2 Trace Rule Engine

The TraceRuleEngine component allows users to create rules to manage traces. An example of a rule is: (a) to define the assurance assets involved in a traceability link, (b) to define the type of a traceability link (i.e., a reference artefact relationship; CCL concept), and (c) the actions/effects in response to changes in the link or the involved assurance assets. This component shall manage a repository of rules based on a metamodel for Trace Rules. Rules can be grouped in catalogues. A user must define the conditions under which a rule should perform a certain task and is then able to execute the rule.

### 4.4.3 Impact Analyser

The ImpactAnalyser works in close relationship with the Data Manager Server tool component that stores the artefacts of an assurance project. The management of the storage must be transparent for the end user, including to browse artefacts and to compare artefact versions. This helps to determine the exact changes that made an impact analysis flagging an artefact as ‘outdated’.

### 4.5 Traceability Manager Client

The TraceabilityManagerClient is the component responsible for the presentation layer of the TraceabilityManagerServer component. This component provides interfaces for trace link edition and trace rule edition and execution. Different kinds of user interfaces must be provided:

- **Navigation**: tree views, filtering, matrices for traceability links.
- **Editor**: forms to edit trace links and rules.
- **Historical view**: possibility to see old versions of traceability information.

#### 4.5.1 GUI Trace Editor

The GUITraceEditor component supports visualisation, rule configuration and rule execution of different types of traceability links.
5 Candidate Technologies

This section introduces the initially three main candidate technologies for implementation of the OPENCOSS evidence management infrastructure: the Qualifying Machine, EMF, and semantic technologies.

5.1 The Qualifying Machine

The Qualifying Machine is an open source artefact manager developed by AdaCore. The main concept behind it is to use a formal description (via a model) of all the artefacts of a project, and then reference them to perform various actions:

- Traceability management
- Change impact notifications
- Artefact navigation, visualisation, and edition
- Documentation management
- Meta-data extraction and storage
- Control of external tool execution

The Qualifying Machine is a good candidate as basic building block for the Assurance Configuration Management, part of WP6, whose purpose is to provide Impact Analysis, Traceability, and Change Management. It also matches the expected functionality that some components for evidence management (Section 4) must support.

The Qualifying Machine consists of two main components: a ‘server’ that references the actual artefacts from raw data using an input model (Figure 12), and a lightweight web-based client (Figure 13) that visualises the artefact database and the traceability links and provides some editing capabilities.

![Figure 12. Qualifying Machine input metamodel](image-url)
The server is easily extensible, as it embeds a python interpreter. The interpreter is used to:

- Programatically query the artefact and traceability link database
- Retrieve/set artefact meta-data
- Enable import of artefacts into the documentation
- Control external tools
- Parse data and extract artefacts from complex data structures

More technical details about the Qualifying Machine are available at http://qmachine.forge.open-do.org/

### 5.2 EMF

EMF is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model. In addition to the core EMF solution, there are several complementary projects that further enhance the functionality of EMF, such as EMF Store (for storage and collaboration of modeled data) and EMF Trace (for modeled data traceability). As a part of T6.2, the EMF technologies family has been investigated to determine how it could be used in OPENCOSS platform implementation.

The following EMF functionality has been identified as **useful** for OPENCOSS platform implementation:

- **EMF Store** can be used to store the OPENCOSS platform data model.
- **EMF library core** can be used to model OPENCOSS evidence metamodel. However, the OPENCOSS GUI, which we envision to be web-based, could not be leveraged from EMF GUI, which is Eclipse-based.
- **EMF Validation Framework** can be used to validate model instances in accordance to defined constraints (e.g., from a safety standard). **OCL** can be used in EMF for constraint specification and validation.
• **CDO** framework can be used to store model instances data. For the OPENCOSS tool platform it is important that CDO components can run in application servers or stand-alone and that they do not require Eclipse. Additionally it is possible to connect CDO to databases.

The following EMF functionality has been identified as **not useful** for the OPENCOSS tool platform implementation in the scope of WP6:

- **EMF editors** cannot be used, if we envision the OPENCOSS GUI using the web. Thus the web GUI would need to be specially developed, if we use EMF as a data model and storage for evidence.
- **UNICASE** project is not useful in the OPENCOSS tool platform, as it defines data models that do not fit to OPENCOSS evidence metamodel.
- **EMF Trace** adapters to CASE tools seem to provide little value for the OPENCOSS tool platform API implementation, as they are based on simple XSL transformations.
- **EMF Trace** traceability and impact analysis functionality cannot be used in the OPENCOSS tool platform as it is not documented and seems not to be supported.

Appendix E provides more information about the results from the investigation on EMF.

### 5.3 Semantic Technologies

Semantic technologies are a combination of software and semantic specifications that encode meanings separately from data and content files and separately from application code. This approach enables machines as well as people to understand, share and reason with data and specifications separately. With semantic technologies, adding, changing and implementing new relationships or interconnecting programs in a different way can be as simple as changing the external model that these programs share. New data can also be brought into the system and visualized or worked upon based on the existing schema.

Semantic technologies provide means to specify data description ontologies (RDF, OWL), frameworks and solutions which implement them, including data storage targeted for metamodel-like data (graph databases) and web-based tools.

Semantic technologies seem to be a good solution not only for implementation of WP6 Evidence Management functionality but also for other components of the OPENCOSS tool platform including WP4 safety standard metamodel definition and editor or WP7 safety standard compliance estimation.

Appendix F provides more information about Semantic Web technologies and a brief overview of the logic that led to the below conclusions.

#### 5.3.1 Advantages of Using Semantic Web in the OPENCOSS Platform

The following advantages of utilizing Semantic Web have been identified:

- Semantic Web seems to provide implementation solution for some of the most challenging components of the OPENCOSS tool platform. It could help to integrate all OPENCOSS platform pieces, and prevent the OPENCOSS team from falling into a trap of thinking in terms of "work packages scope only". One common technology would facilitate a holistic system thinking of all teams.

- Syncing of technologies/solutions across the platform is very important. As stated in D2.3, "The architecture design (D2.3) is therefore strongly connected to WP6 requirements and solution decisions."

- Semantic Web solutions seem to be mature. Technologies like RDF, OWL, SKOS are W3C standardized.
5.3.2 Draft of Implementation

This section presents a summary about which specific Semantic Web technology could be used to implement specific components of the OPENCOSS tool platform. Details about the suggested solution are shown in Appendix C.

The technology that could be used in WP6 and other WPs is:

- **WP4 (Safety Standard Metamodel)**
  - Definition: OWL, SKOS
  - Editor: Web Protégé, which provides a web interface
  - Storage: semantic repository
  - Automated Consistency Check: a reasoner such as Pellet

- **WP6**
  - Evidence Storage: graph database such as Neo4j and HyperGraphDB
  - Evidence Items Editor: Web Protégé
  - Impact Analysis and traceability: Pellet
  - Integration with external tools that handle unstructured data: Apache Tika" or "GATE".

- **WP7**
  - Safety standard compliance detection: Pellet or Hermit

5.3.3 Disadvantages and Risks of Using Semantic Web in OPENCOSS

The following disadvantages and risks of utilizing Semantic Web have been identified:

- This technology is not known well by any partner in the consortium. This means a learning curve and uncertain results.
- Most components of the OPENCOSS platform (not only evidence storage) need to use this technology, thus they need to be compatible.

Because of the above risks, it has been decided (as an initial work in implementation task) to develop a proof-of-concept using Semantic Web. It will be based on Protégé.
6 Conclusion

D6.3 has presented the current version of the design of the evidence management infrastructure of the OPENCOSS tool platform. Such infrastructure aims to support and enable the activities necessary for dealing with evolutionary evidence and with evolutionary chains of evidence.

Development of the deliverable has been based in previous work in OPENCOSS. More concretely, the design presented is strongly based on the requirements specified in D6.2 and in the overall OPENCOSS tool platform architecture presented in D2.3. In addition, more work has been necessary in relation to the notion of safety evidence, safety evidence traceability, commonalities in evidence management among different application domains, existing tools and technologies for evidence management, and conceptual needs for evidence management. Most of this work has been performed in close collaboration and coordination with other OPENCOSS tasks, especially with T4.2, which deals with CCL specification.

From this analysis and the input from OPENCOSS partners, we have specified a data model for evidence management. This model extends the CCL with WP6-specific needs for implementation of the CCL in the context of evolutionary evidence management. In addition, the evidence characterization and traceability editor component and the evidence analysis component of the OPENCOSS tool platform architecture have been refined specified in detail. The resulting design specification consists of three tool component layers (data management, core components, and GUI client) and two kinds of components groups (management of artefacts and traceability). The specification is generic, will be realised in T6.4, and also aims to allow other developers to implement their own OPENCOSS-based evidence management solution.

The initial set of candidate technologies for WP6 implementation has also been introduced. Such a set consists of the Qualifying Machine, EMF-based tools, and semantic technologies. These technologies, and probably others, will be further studied and applied in the scope of T6.4. Some of these technologies might be finally discarded.

In relation to the current and future work to perform in WP6 and other WPs, the aspects to which attention must be paid in the near future for refinement and extension of D6.3 are as follows:

- Refinement data needs for evidence management
- Link of D6.3 and D6.4, and specification of the interaction between the components specified in each deliverable
- Alignment of D6.3 and D4.4 once the CCL has been further validated
- The technologies with which the evidence management services will finally be implemented
- Safety evidence change impact analysis (continuing the work outlined in [6]; a survey is currently under designed, as outlined in Appendix D)
- Applicability of automated traceability technologies for safety evidence (as discussed in [19])
- How safety evidence is combined
- Safety evidence traceability visualization
- The possible equivalence of process-based and requirement-based evidence (in line with the initial work presented in [8])
- Evidence-based business process compliance (based on approaches such as [5])
- Evidence-based correspondence of requirements among different domains (e.g., with the proposal for analysis of regulatory requirements presented in [13])

Some of these aspects might finally be addressed in other technical WPs. WP3 will also have to specify test cases for validation of the design presented in this deliverable once the OPENCOSS tool platform is developed.
References


Appendix A. Tools that Could Be Reused for Evidence Management

This appendix presents the tools that have been analysed in the scope of D6.3 for determining if they could be reused or adapted for WP6 implementation purposes. The analysis complements and extends those presented in D6.1 and D6.2 by providing more details about the advantages and disadvantages of reusing the tools and by analysing more tools.

### Alfresco

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="http://www.alfresco.com/">http://www.alfresco.com/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>Alfresco is an open source enterprise content platform that you can use in the cloud or behind your firewall. It helps to store and share the documents.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
<td>Last release: 11.12.2012</td>
</tr>
<tr>
<td></td>
<td>Last update: 10.04.2013</td>
</tr>
<tr>
<td></td>
<td>Ohloh summary: <a href="http://www.ohloh.net/p/alfresco">http://www.ohloh.net/p/alfresco</a></td>
</tr>
</tbody>
</table>

### Alignment API and Alignment Server

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="http://alignapi.gforge.inria.fr/">http://alignapi.gforge.inria.fr/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>The Alignment API is an API and implementation for expressing and sharing ontology alignments. The Alignment API itself is a Java description of tools for accessing a common format. It defines five main interfaces (OntologyNetworks, Alignment, Cell, Relation and Evaluator) and proposes the following services:</td>
</tr>
<tr>
<td></td>
<td>• Storing, finding, and sharing alignments</td>
</tr>
<tr>
<td></td>
<td>• Piping alignment algorithms (improving an existing alignment)</td>
</tr>
<tr>
<td></td>
<td>• Manipulating (thresholding and hardening)</td>
</tr>
<tr>
<td></td>
<td>• Generating processing output (transformations, axioms, rules)</td>
</tr>
<tr>
<td></td>
<td>• Generating tests and comparing alignments (diff, precision, recall).</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
<td>Last release: 25.03.2013</td>
</tr>
<tr>
<td></td>
<td>Last update: 1.04.2013</td>
</tr>
</tbody>
</table>

### Apache Jena

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="http://jena.apache.org/">http://jena.apache.org/</a> <a href="https://github.com/apache/jena">https://github.com/apache/jena</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>Java framework for building Semantic Web applications. Jena provides a collection of tools and Java libraries to help you to develop semantic web and linked-data apps, tools and servers. The Jena Framework includes:</td>
</tr>
<tr>
<td></td>
<td>• An API for reading, processing and writing RDF data in XML, N-triples and Turtle formats</td>
</tr>
<tr>
<td></td>
<td>• An ontology API for handling OWL and RDFs ontologies</td>
</tr>
<tr>
<td></td>
<td>• A rule-based inference engine for reasoning with RDF and OWL data sources</td>
</tr>
<tr>
<td></td>
<td>• Stores to allow large numbers of RDF triples to be efficiently stored on disk</td>
</tr>
</tbody>
</table>
• A query engine compliant with the latest SPARQL specification
• Servers to allow RDF data to be published to other applications using a variety of protocols, including SPARQL

**Additional Information**
Last release: 24.02.2013
Last update: 17.04.2013
Ohloh summary: http://www.ohloh.net/p/jena

### Apache Lucene

**Main Information Sources**
http://lucene.apache.org
http://en.wikipedia.org/wiki/Lucene

**Overall Description**
• Free/open source information retrieval software library
• Text indexer and search library, used in industry, working well, good documentation.
• Extractors for various formats (doc, pdf)
• We can use it when we need to search/analyse evidences text (in various formats)

### Apache OpenNLP

**Main Information Sources**
http://opennlp.apache.org/

**Overall Description**
The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron-based machine learning.

**Additional Information**
Last release: 15.04.2013
Last update: 15.04.2013
Ohloh summary: http://www.ohloh.net/p/apache-opennlp

### Apache Stanbol

**Main Information Sources**
http://stanbol.apache.org/
https://github.com/apache/stanbol

**Overall Description**
Apache Stanbol provides a set of reusable components for semantic content management. Its intended use is to extend traditional content management systems with semantic services. Other feasible use cases include: direct usage from web applications (e.g. for tag extraction/suggestion; or text completion in search fields), 'smart' content workflows or email routing based on extracted entities, topics, etc.

Apache Stanbol's main features are:
• Content Enhancement - Services that add semantic information to “non-semantic” pieces of content.
• Reasoning - Services that are able to retrieve additional semantic information about the content based on the semantic information retrieved via content enhancement.
• Knowledge Models - Services that are used to define and manipulate the data models (e.g. ontologies) that are used to store the semantic information.
- Persistence - Services that store (or cache) semantic information, i.e. enhanced content, entities, facts, and make it searchable.

### Additional Information
Last release: 16.01.2013  
Last update: 18.04.2013  
Ohloh summary: http://www.ohloh.net/p/stanbol  

### Apache Stanbol Enhancer

**Main Information Sources**
http://stanbol.apache.org/docs/trunk/components/enhancer/

**Overall Description**

Apache Stanbol Enhancer provides both a RESTful and a Java API that allows a caller to extract features from passed content. For enhancing content you simply post content to the Stanbol Enhancer. The Apache Stanbol Enhancer supports multiple enhancement chains. This feature allows you to configure multiple processing chains for parsed content within the same Apache Stanbol instance.

Some of the enhancement engines are:

1. **Preprocessing**
   - content type detection
   - text extraction from various document formats
   - extraction of metadata from document formats

2. **Natural Language Processing (NLP)**
   - Language Detection for textual content
   - Sentence Detection
   - Tokenization detection
   - Part of Speech (POS) Tagging
   - Chunk/Phrase detection
   - Named Entity Recognition (NER) (i.e. detection of occurrences of persons, places, organizations as well as custom entity types)
   - Morphological Analysis
   - General NLP processing

3. **Linking / Suggestions**
   - Links suggestion to several Linked Data Sources
   - Detection of occurrences of untyped entities as concepts, using local taxonomies as linking target
   - Sentiment Analyses (word/chunk level sentiment classifications)
   - Disambiguation of entities based on contextual information

4. **Postprocessing / Other**
   - Conversion of NLP processing results to RDF
   - Transformation of enhancements according to a target ontology
   - Retrieval of additional content for presenting the enhancement results

### Additional Information
Last release: 27.01.2013  
Last update: 17.04.2013  
Ohloh summary: http://www.ohloh.net/p/stanbol  

### Apache Tika - a content analysis toolkit

**Main Information Sources**
http://tika.apache.org/

**Overall Description**
The Apache Tika™ toolkit detects and extracts metadata and structured text content from various documents using existing parser libraries.

### Additional Information
- **Last release:** 22.01.2013
- **Last update:** 11.04.2013
- **Ohloh summary:** [http://www.ohloh.net/p/tika](http://www.ohloh.net/p/tika)

## AutoMeta - A semantic annotation tool

**Main Information Sources**
- [https://code.google.com/p/autometa/](https://code.google.com/p/autometa/)

**Overall Description**
AutoMeta (automatic meta data annotation) is an environment for semi-automatic (or automatic) annotation and meta-annotation of documents for publishing on the Web using RDFa, a W3C recommended annotation language. It also includes a RDFa extraction tool to provide the user with a view of the annotated triples. It is available in both CLI (Command Line Interface) and GUI (Graphical User Interface) interfaces.

**Additional Information**
- **Last release:** 23.01.2013
- **Last update:** 28.01.2013

## Callimachus

**Main Information Sources**
- [http://callimachusproject.org/](http://callimachusproject.org/)

**Overall Description**
Callimachus is a framework for data-driven applications based on Linked Data. Callimachus allows Web developers to quickly and easily create Web applications based on Linked Data. The user interface is consistent - it is RESTful from the ground up. Developers need only a Web browser to create a data-driven application. They can create Callimachus applications locally and host it on the Web.

**Additional Information**
- **Last release:** 5.03.2013
- **Last update:** 17.04.2013
- **Ohloh summary:** [http://www.ohloh.net/p/Callimachus](http://www.ohloh.net/p/Callimachus)

## COEUS - Semantic Web Application Framework

**Main Information Sources**
- [https://github.com/pdrlps/COEUS](https://github.com/pdrlps/COEUS)

**Overall Description**
COEUS is a streamlined back-end framework for rapid semantic web application development. Among some of its features, it allows to:
- Create custom warehouses, collecting distributed and heterogeneous data through integrating CSV, SQL, XML or SPARQL resources with advanced Extract-Transform-Load warehousing features;
- Use Semantic Web & LinkedData technologies in all application layers by enabling reasoning and inference over connected knowledge;
- Access data through LinkedData interfaces and enable a custom SPARQL endpoint;
- Connect multiple nodes together and with any other software, e.g. to create your own knowledge network using SPARQL Federation, enabling data-sharing amongst a scalable number of peers;
- Reduce development time by getting new applications up and running much faster, using the latest rapid application development strategies.
## COMA Community Edition

**Main Information Sources**
- [http://dbs.uni-leipzig.de/Research/coma.html](http://dbs.uni-leipzig.de/Research/coma.html)
- [http://sourceforge.net/projects/coma-ce/](http://sourceforge.net/projects/coma-ce/)
- [http://dbs.uni-leipzig.de/de/research/projects/schema_and_ontology_matching/coma_3_0/coma_3_0_community_edition](http://dbs.uni-leipzig.de/de/research/projects/schema_and_ontology_matching/coma_3_0/coma_3_0_community_edition)

**Overall Description**
COMA CE is the community edition of the well-established COMA project developed at the University of Leipzig. It comprises the parsers, matcher library, matching framework, and a sample GUI for tests and evaluations. It excels due to numerous matching strategies, which can be combined to large matching workflows, and which enable reliable match results between different kind of schemas. Schema and ontology matching aim at identifying semantic correspondences between metadata structures or models such as database schemas, XML message formats, and ontologies. Solving such match problems are of key importance to service interoperability and data integration in numerous application domains. The goal is to keep manual effort low.

**Additional Information**
- Last release: 24.01.2013
- Last update: 24.01.2013

## CRESCO

**Main Information Sources**

**Overall Description**
Construction of Evidence Repositories for Managing Standards Compliance is a tool capable to take UML class diagram of standards compliance evidences as an input and generate a database schema and WEB UI interface for managing evidences.

## Cytoscape

**Main Information Sources**
- [http://www.cytoscape.org/](http://www.cytoscape.org/)
- [https://github.com/cytoscape](https://github.com/cytoscape)

**Overall Description**
Cytoscape is an Open Source Platform for visualizing complex networks and integrating these with any type of attribute data. A lot of plugins are available for various kinds of problem domains, including bioinformatics, social network analysis, and semantic web.

**Additional Information**
- Last release: 14.02.2013
- Last update: 18.04.2013
- Ohloh summary: [http://www.ohloh.net/p/cytoscape](http://www.ohloh.net/p/cytoscape)

## EGit (and Git)

**Main Information Sources**
- [http://marketplace.eclipse.org/content/egit-git-team-provider#.URo_-6XZW-8](http://marketplace.eclipse.org/content/egit-git-team-provider#.URo_-6XZW-8)
- [http://git-scm.com](http://git-scm.com)

**Overall Description**
Eclipse Team provider for the Git version control system. Git is a distributed SCM, which means every developer has a full copy of all history of every revision of the code, making queries against the history very fast and versatile. EGit is just a client for Git SCM implemented as Eclipse plugin.

<table>
<thead>
<tr>
<th>EMFStore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong>&lt;br&gt;<a href="http://eclipse.org/emfstore/">http://eclipse.org/emfstore/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong>&lt;br&gt;Model repository (server) for EMF featuring collaborative editing and versioning of models. A server for storing and versioning EMF models. It supports a workflow similar to SVN code repository and covers visual conflicts resolution of concurrent model changes. This opens a whole bunch of other tools that should be taken into account, related to EMF, such as EMF.Edit for editing the model (see <a href="http://www.eclipse.org/modeling/emf/?project=emf#emf">http://www.eclipse.org/modeling/emf/?project=emf#emf</a>). The store can be used to manage evidences and artefacts links. Versioning would allow historical traceability with minimal development effort.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMFTrace</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong>&lt;br&gt;<a href="http://sourceforge.net/p/emftrace/wiki/Home/">http://sourceforge.net/p/emftrace/wiki/Home/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong>&lt;br&gt;Extension to the EMFStore repository by dependencies and traceability links between related models of different modelling languages, with Eclipse Public License. It Depends on EMFStore (see it first), so it is reasonable to evaluate it only if EMFStore is decided to use. EMFTrace extends the underlying repository by integrating several import interfaces for different CASE tools and automated traceability detection techniques.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong>&lt;br&gt;<a href="https://code.google.com/p/elk-reasoner/">https://code.google.com/p/elk-reasoner/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong>&lt;br&gt;ELK is a reasoner for OWL 2 ontologies that currently supports a part of the OWL EL ontology language. The goal of ELK is to provide a very fast reasoning engine for OWL EL. Currently, the supported OWL features and reasoning tasks are still limited (but already sufficient for important ontologies such as SNOMED CT). The aim of the project is to complete the implementation for all OWL EL features and relevant reasoning functions.</td>
</tr>
<tr>
<td><strong>Additional Information</strong>&lt;br&gt;Last release: 17.01.2013&lt;br&gt;Last update: 12.04.2013&lt;br&gt;Ohloh summary: <a href="http://www.ohloh.net/p/elk">http://www.ohloh.net/p/elk</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GATE - General Architecture for Text Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong>&lt;br&gt;<a href="http://gate.ac.uk/">http://gate.ac.uk/</a>&lt;br&gt;<a href="http://sourceforge.net/projects/gate/">http://sourceforge.net/projects/gate/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong>&lt;br&gt;GATE is a Java suite of tools originally developed at the University of Sheffield at beginning in 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for all sorts of natural language processing tasks, including information extraction in many languages. GATE includes an information extraction system called ANNIE (A Nearly-New Information Extraction System), which is a set of modules comprising a tokenizer, a gazetteer, a sentence splitter, a part of speech tagger, a named entities transducer and a co-reference tagger. ANNIE can be used as-is to provide basic information extraction functionality, or provide a starting point for more specific tasks. Languages currently handled in</td>
</tr>
</tbody>
</table>
GATE include English, Spanish, Chinese, Arabic, Bulgarian, French, German, Hindi, Italian, Cebuano, Romanian, and Russian. Plugins are included for machine learning with Weka, RASP, MAXENT, SVM Light, as well as a fast LibSVM integration and an in-house perceptron implementation, for managing ontologies like WordNet, for querying search engines like Google or Yahoo, for part of speech tagging with Brill or TreeTagger, and many more. GATE accepts input in various formats, such as TXT, HTML, XML, Doc, PDF documents, and Java Serial, PostgreSQL, Lucene, Oracle Databases with help of RDBMS storage over JDBC.

**Additional Information**
Last release: 30.11.2012
Last update: 18.04.2013
Ohloh summary: http://www.ohloh.net/p/gate

<table>
<thead>
<tr>
<th>Hermit OWL Reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://www.hermit-reasoner.com/">http://www.hermit-reasoner.com/</a></td>
</tr>
<tr>
<td><a href="https://code.google.com/p/hermit-reasoner/">https://code.google.com/p/hermit-reasoner/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>HermiT is reasoner for ontologies written using the Web Ontology Language (OWL). Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify subsumption relationships between classes, and much more. HermiT is the first publicly-available OWL reasoner based on a novel “hypertableau” calculus which provides much more efficient reasoning than any previously-known algorithm. Ontologies that previously required minutes or hours to classify can often by classified in seconds by HermiT, and HermiT is the first reasoner able to classify a number of ontologies which had previously proven too complex for any available system to handle. HermiT uses direct semantics and passes all OWL 2 conformance tests for direct semantics reasoners.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
</tr>
<tr>
<td>Last release: 25.03.2013</td>
</tr>
<tr>
<td>Last update: 25.03.2013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HyperGraphDB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://www.hypergraphdb.org/">http://www.hypergraphdb.org/</a></td>
</tr>
<tr>
<td><a href="https://code.google.com/p/hypergraphdb/">https://code.google.com/p/hypergraphdb/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>HyperGraphDB is a general purpose, open-source data storage mechanism based on a powerful knowledge management formalism known as directed hypergraphs. While a persistent memory model designed mostly for knowledge management, AI and semantic web projects, it can also be used as an embedded object-oriented database for Java projects of all sizes, a graph database, or a (non-SQL) relational database. HyperGraphDB's implementation is based on BerkeleyDB database. The graph model of HyperGraphDB is known as direct hypergraphs. In mathematics, a hypergraph allows its edge pointing to more than two nodes. HyperGraphDB extends this further by allowing edges to point to other edges, so HyperGraphDB offers more generality than other graph databases.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
</tr>
<tr>
<td>Last release: 4.11.2012</td>
</tr>
<tr>
<td>Last update: 15.04.2013</td>
</tr>
<tr>
<td>Ohloh summary: <a href="http://www.ohloh.net/p/hypergraphdb">http://www.ohloh.net/p/hypergraphdb</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JFact</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://jfact.sourceforge.net/">http://jfact.sourceforge.net/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>JFact is a Java port of the FaCT++ OWL DL reasoner (written in C++). Its design matches FaCT++ design very</td>
</tr>
</tbody>
</table>
closely, except for the low level details where Java offers different ways of doing things. As FaCT++ does, it supports OWL DL and (partially) OWL 2 DL.

### komma - Knowledge Modeling and Management Architecture

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="https://code.google.com/a/eclipselabs.org/p/komma/">https://code.google.com/a/eclipselabs.org/p/komma/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>KOMMA combines the Sesame RDF framework with concepts of EMF to create a flexible framework for the management and editing of OWL ontologies on top of the Eclipse Platform. KOMMA's main features include:</td>
</tr>
<tr>
<td>• Object-Triple Mapping (via IEntityManager - comparable to JPA's EntityManager)</td>
<td></td>
</tr>
<tr>
<td>• ModelSets and Models for management of ontologies (KOMMA's alternative to EMF's ResourceSets and Resources)</td>
<td></td>
</tr>
<tr>
<td>• Editing framework (port of EMF.Edit)</td>
<td></td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
<td>Last update: 23.11.2012</td>
</tr>
</tbody>
</table>

### MapPSO

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="http://sourceforge.net/apps/mediawiki/mappso/">http://sourceforge.net/apps/mediawiki/mappso/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>MapPSO is a tool for Ontology Alignment, which uses Discrete Particle Swarm Optimisation. A particle swarm is used to search for the optimal alignment. The algorithm is massively parallel and adapts naturally on parallel architectures. Its main features are:</td>
</tr>
<tr>
<td>• Discrete particle swarm optimisation as a massively parallel method to find an optimal alignment;</td>
<td></td>
</tr>
<tr>
<td>• Anytime behaviour possible;</td>
<td></td>
</tr>
<tr>
<td>• General framework for easy adaptation of base matchers to adapt to a particular alignment problem;</td>
<td></td>
</tr>
</tbody>
</table>
| **Additional Information**       | Last release: 19.03.2012  
Last update: 26.03.2013  
Ohloh summary: http://www.ohloh.net/p/mappso |

### MoDisco

<table>
<thead>
<tr>
<th>Main Information Sources</th>
<th><a href="http://www.eclipse.org/MoDisco/">http://www.eclipse.org/MoDisco/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Description</strong></td>
<td>MoDisco provides an extensible framework to develop model-driven tools to support use-cases of existing software modernization. It is an extensible and customizable Model Driven Reverse Engineering framework. The framework aims at discovering a model of a “legacy system”. The input (legacy system) to be modelled can be source code, raw data, available documentation, etc. Such a model provides a uniform representation of the system, which conforms to a given metamodel. The actual building of the model is realized thanks to a software component called “discoverer” - it has to be developed for each legacy system, some are already implemented (e.g., for Java model). Considering as inputs all kinds of possible legacy artefact (e.g., source code, databases, configuration files, documentation, etc.), MoDisco aims at providing the required capabilities for creating models and allowing there handling, analysis and</td>
</tr>
</tbody>
</table>
computation. As outputs, the framework targets the production of different types of artefact depending on the selected reverse engineering objective(s) (e.g., source code, data, metrics, visualizations, documentation, etc).

<table>
<thead>
<tr>
<th>MORe Reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://www.cs.ox.ac.uk/isg/tools/MORe/">http://www.cs.ox.ac.uk/isg/tools/MORe/</a></td>
</tr>
<tr>
<td><a href="https://code.google.com/p/more-reasoner/">https://code.google.com/p/more-reasoner/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>MORe (Modular Combination of OWL Reasoners for Ontology Classification) integrates HermiT (a fully-fledged OWL 2 reasoner) with ELK (a reasoner for the OWL 2 EL profile) in a modular way. In particular, MORe exploits module extraction techniques to identify a subset of the ontology that can be completely classified using ELK. MORe is designed in such a way that the fully-fledged (and slower) reasoner HermiT performs as few computations as possible, and the bulk of the computation is delegated to the more efficient, profile specific, ELK reasoner.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
</tr>
<tr>
<td>Last release: 28.02.2013</td>
</tr>
<tr>
<td>Last update: 28.02.2013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mulgara</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://www.mulgara.org/">http://www.mulgara.org/</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>Mulgara is a triplestore and fork of the original Kowari project. It is open source, scalable, and transaction-safe. Mulgara instances can be queried via the iTQL query language and the SPARQL query language. Mulgara is not based on a relational database due to the large numbers of table joins encountered by relational systems when dealing with metadata. Instead, Mulgara is a completely new database optimized for metadata management. Mulgara models hold metadata in the form of short subject-predicate-object statements, much like the W3C's Resource Description Framework (RDF) standard. Metadata may be imported into or exported from Mulgara in RDF or Notation 3 form.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
</tr>
<tr>
<td>Last release: 10.01.2012</td>
</tr>
<tr>
<td>Last update: 30.05.2012</td>
</tr>
<tr>
<td>Ohloh summary: <a href="http://www.ohloh.net/p/mulgara">http://www.ohloh.net/p/mulgara</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Neo4j</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Information Sources</strong></td>
</tr>
<tr>
<td><a href="http://neo4j.org/">http://neo4j.org/</a></td>
</tr>
<tr>
<td><a href="https://github.com/neo4j">https://github.com/neo4j</a></td>
</tr>
<tr>
<td><strong>Overall Description</strong></td>
</tr>
<tr>
<td>Neo4j is a graph database. It is an open source, embedded, disk-based, robust (fully ACID) transactional Java persistence engine that stores data structured in graphs rather than in tables. A graph (mathematical lingo for a network) is a flexible data structure that allows a more agile and rapid style of development. Neo4j can be turned into a SAIL compliant SPARQL endpoint (see <a href="http://components.neo4j.org.neo-rdf/">http://components.neo4j.org.neo-rdf/</a>), but translates SPARQL queries into actual Graph Traversals and stores the Triples as a connected graph.</td>
</tr>
<tr>
<td><strong>Additional Information</strong></td>
</tr>
<tr>
<td>Last release: 27.02.2013</td>
</tr>
<tr>
<td>Last update: 17.04.2013</td>
</tr>
<tr>
<td>Ohloh summary: <a href="http://www.ohloh.net/p/neo4j-graphdb">http://www.ohloh.net/p/neo4j-graphdb</a></td>
</tr>
</tbody>
</table>
### Nuxeo

**Main Information Sources**
- http://www.nuxeo.com/
- https://github.com/nuxeo/

**Overall Description**
Nuxeo provides an Open Source Content Management Platform enabling architects and developers to easily build, deploy, and run content-centric business applications. In the cloud or on premise, Nuxeo’s Enterprise Content Management (ECM) technology offers an integrated solution for Document Management, Case Management, Digital Asset Management and much more. Built on a modern, Java-based architecture, the Nuxeo Platform is architected for modularity and extensibility, unlike traditional ECM solutions.

**Additional Information**
- Last release: 01.2013
- Last update: 18.04.2013
- Ohloh summary: http://www.ohloh.net/p/nuxeo

### OpenKM

**Main Information Sources**

**Overall Description**
OpenKM is an open source Document Management System. It is a web-based document management application. It includes a content repository, Lucene indexing, jBPM workflow, interface built using Google Web Toolkit, support for 20+ languages, and a rich set of features.

**Additional Information**
- Last release: 4.04.2013
- Last update: 10.04.2013
- Ohloh summary: http://www.ohloh.net/p/9475

### OrientDB

**Main Information Sources**
- http://www.orientdb.org/
- https://github.com/nuvolabase/orientdb/

**Overall Description**
OrientDB has the flexibility of the Document databases and the power of the Graph databases to manage relationships. It can work in schema-less mode, schema-full, or a mix of both. It supports advanced features such as ACID Transactions, Fast Indexes, Native, and SQL queries. It imports and exports documents in JSON. OrientDB uses a new indexing algorithm called MVRB-Tree, derived from the Red-Black Tree and from the B+ Tree with benefits of both fast insertion and ultra fast lookup.

**Additional Information**
- Last update: 17.04.2013
- Ohloh summary: http://www.ohloh.net/p/orientdb

### OWL API

**Main Information Sources**
- http://owlapi.sourceforge.net/

**Overall Description**
The OWL API is an open source Java API and reference implementation for creating, manipulating and...
serialising OWL Ontologies. The latest version of the API is focused towards OWL 2. It includes the following components:

- An API for OWL 2 and an efficient in-memory reference implementation
- RDF/XML parser and writer
- OWL/XML parser and writer
- OWL Functional Syntax parser and writer
- Turtle parser and writer
- KRSS parser
- OBO Flat file format parser
- Interfaces for working with reasoners such as FaCT++, HermiT, Pellet, and Racer

### Payola

**Main Information Sources**
- http://payola.github.io/Payola/
- https://github.com/payola/Payola

**Overall Description**
Payola is an HTML5 web application that lets you work with graph data in a completely new way. You can visualize Linked Data using several preinstalled plugins as graphs, tables, etc. This also means, that you no longer need Pubby to browse through a Linked Data storage (via its SPARQL endpoint). Moreover, you can create an analysis and run it against a set of SPARQL endpoints without deep knowledge of SPARQL language itself. Analysis results are processed and visualized using the embedded visualization plugins. It is written in Scala language.

### Pellet

**Main Information Sources**
- http://clarkparsia.com/pellet/
- https://github.com/ansell/pellet

**Overall Description**
Pellet is an OWL 2 reasoner, a core component of ontology-based data management applications. It provides standard and cutting-edge reasoning services for OWL ontologies. For applications that need to represent and reason about information using OWL, Pellet is the leading choice for systems where sound-and-complete OWL DL reasoning is essential. Pellet includes support for OWL 2 profiles including OWL 2 EL. It incorporates optimizations for nominals, conjunctive query answering, and incremental reasoning.

### ProR (and RMF)

**Main Information Sources**
- http://www.eclipse.org/rmf/pror/

**Overall Description**
**Open-source tool for requirements management.** It includes simple requirements management features:

- Adding/editing requirements items
- Customizable fields in req items (any field label can be added)
- Linking between requirements
- Support for RIF format
- Eclipse plugin

It is still in incubation.

---

**Protégé**

<table>
<thead>
<tr>
<th>Main Information Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://protege.stanford.edu/">http://protege.stanford.edu/</a></td>
</tr>
</tbody>
</table>

**Overall Description**

Protégé is a free, open source ontology editor. The Protégé platform supports two main ways of modelling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Protégé is based on Java, is extensible (many plugins are available; see http://protegewiki.stanford.edu/wiki/Protege_Plugin_Library), and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. Protégé is supported by a strong community of developers and academic, government and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modelling. The web version, 'WebProtégé' is a lightweight, web-based ontology editor, also open source.

**Additional Information**

- Last release: 15.04.2013
- Last update: 12.04.2013

---

**RETO**

<table>
<thead>
<tr>
<th>Main Information Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://opensource.gsfc.nasa.gov/projects/RETRO/">http://opensource.gsfc.nasa.gov/projects/RETRO/</a></td>
</tr>
</tbody>
</table>

**Overall Description**

Tool for tracing textual software engineering artefacts to each other using standard information retrieval techniques. It allows you to find correlations among text documents. It uses smart heuristics, such as thesaurus and others, to find correlation even when text was written separately (not copied). Correlations (links) must be then reviewed by human to accept or reject correlations. This allows you to build a map of correlations among requirements specifications, problem reports or software users’ feedback. This software is released under the terms and conditions of the NASA Open Source Agreement (NOSA) Version 1.1 or later.

---

**Sesame**

<table>
<thead>
<tr>
<th>Main Information Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.openrdf.org/">http://www.openrdf.org/</a></td>
</tr>
<tr>
<td><a href="https://bitbucket.org/openrdf/sesame">https://bitbucket.org/openrdf/sesame</a></td>
</tr>
</tbody>
</table>

**Overall Description**

Sesame is an extensible open source Java framework for storing, querying and inferencing RDF data. It can be deployed as a web server or used as a Java library. Its features include the support for several query languages (SeRQL and SPARQL), inferencing support, and RAM or disk storage. Additionally, the central APIs of Sesame are storage-independent and are supported by many third-party RDF database vendors.

**Additional Information**

- Last release: 16.04.2013
### Semantic Assistants

**Main Information Sources**
- http://www.semanticsoftware.info/semantic-assistants
- http://sourceforge.net/projects/semantic-assist/

**Overall Description**
Semantic Assistants support users in content retrieval, analysis, and development, by offering context-sensitive NLP services directly integrated with common desktop applications (word processors, email clients, web browsers...), web information systems (wikis, portals), and mobile applications. They are implemented through an open service-oriented architecture, using Semantic Web ontologies and W3C Web Services.

Some of the features are:
- End-user friendly content analysis embedded in common desktop clients
- Standard W3C Web Services for Natural Language Processing (NLP)
- Semantic Assistants Server executes GATE NLP pipelines
- Abstraction Layer for easy integration of NLP into new clients
- Client plug-ins for OpenOffice, Eclipse, Firefox, Thunderbird, and more
- Integration with MediaWiki for embedding NLP in wiki systems
- Semantic (OWL-based) metadata for NLP service descriptions
- Comes with ready-to-deploy examples services for Information Extraction (ANNIE); others available for download

**Additional Information**
- Last release: 1.03.2012
- Last update: 10.04.2013
- Ohloh summary: http://www.ohloh.net/p/payola

### Subversive

**Main Information Sources**
- http://marketplace.eclipse.org/content/subversive-svn-team-provider#.URpAwKXZW-8

**Overall Description**
The Subversive project is aimed to integrate the Subversion (SVN) version control system with the Eclipse platform.

### TOPCASED-Requirements (and Polarsys)

**Main Information Sources**

**Overall Description**
This project aims at developing a way to ensure traceability between requirements and model elements and it provides tools to manage requirements. It aims at being generic for all modelling editors. Mature product for establishing traceability among requirements and elements of UML models and SysML models. It is based on Eclipse EMF. It includes the following WP-related functionality:
- Importing requirements from text documents (word) and spreadsheets (excel) and standard formats
- Building relation among elements of UML models and SysML models
- Automated exporting traceability matrix:
It could be used in OPENCOSS in relation to:

- Traceability matrix, which can be exported from TOPCASED, can be used by OPENCOSS to import relations.
- If OPENCOSS would be based on EMF, we can learn about EMF capabilities from this project.

**Additional Information**


---

### TraceClient

**Main Information Sources**

http://code.google.com/p/unicase/wiki/TraceClient

**Overall Description**

This plugin provides (semi-) automatic creation and usage of traceability links between 3 types of entities: requirements definitions, work items (tasks for users), and source code (committed to SVN) during development in Eclipse IDE. It uses Eclipse EMF. In a typical usage scenario, a user marks a work item on which he is currently working, and from then on when users reads any requirement definition, it becomes a candidate to be linked to the work item. Similarly when any code developed is committed to SVN, it gets candidate to be linked to the work item. When marking the work item as completed, user is asked to validate if the requirements/code list gathered automatically is ok, modify the list, and submit. This way we get requirements-work item- code links. Provides a graphical visualizer (using Eclipse ZEST plugin) and searchable table of the traceability.

---

### Traceclipse

**Main Information Sources**

http://www.cs.wm.edu/semeru/traceclipse/

**Overall Description**

Eclipse plugin that enables software developers to specify, view, and manipulate traceability links. It analyses Java code and use cases (txt) and establishes traceability links between them. It uses Apache Lucene. The plugin is currently unstable and works partially.

---

### twouse - Semantic Web + Model Driven Development

**Main Information Sources**

https://code.google.com/p/twouse/

**Overall Description**

With TwoUse Toolkit you bridge the gap between Semantic Web and Model Driven Engineering. It is an implementation of current OMG and W3C standards for software design, code generation and OWL ontology engineering.

**Additional Information**

Last release: 8.2010
Last update: 10.12.2010
Ohloh summary: http://www.ohloh.net/p/twouse

---

### Visual Understanding Environment (VUE)

**Main Information Sources**

http://vue.tufts.edu/
http://sourceforge.net/projects/tuftsvue/

**Overall Description**

VUE provides a flexible visual environment for structuring, presenting, and sharing digital information. It is an open source project based at Tufts University, focused on creating flexible tools for managing and integrating digital resources in support of teaching, learning and research. Among its many features, it
supports importing of ontologies defined in RDF-S or OWL formats, allowing for the creation of concept maps from pre-defined object and relationship types. A defined mapping vocabulary scaffolds map creation and supports computer-assisted map comparison and assessment. The visual characteristics of objects and relationships defined in an ontology may also be styled via a CSS file. VUE also provides support for in-depth analysis of maps with the ability to merge maps and export connectivity matrices to import in statistical packages.

### Additional Information
- Last release: 8.03.2012
- Last update: 16.03.2013
- Ohloh summary: http://www.ohloh.net/p/tuftsvue

### WebProtege

<table>
<thead>
<tr>
<th>Main Information Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://protegewiki.stanford.edu/wiki/WebProtege">http://protegewiki.stanford.edu/wiki/WebProtege</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overall Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebProtégé is an open source, lightweight, web-based ontology editor. WebProtégé provides a friendly and highly configurable user interface that can be adapted for the use of domain experts. It has support for form-based editing and full-fledged collaboration.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last release: 15.04.2013</td>
</tr>
<tr>
<td>Last update: 15.04.2013</td>
</tr>
</tbody>
</table>
Appendix B. Detailed Analysis of EMF for Evidence Management

This appendix extends the information presented in Section 5 and in Appendix A about EMF-based technologies with further details about their pros and cons as tools for safety evidence management. The appendix provides the detailed results of the investigation about which tools and solutions from the EMF family could be used for implementation of the OPENCOSS platform. EMF core and its sub-projects (EMFStore, Unicase, EMFTrace) have been investigated for reuse in WP6 implementation.

The general conclusions are presented in Section 5.2.

B.1 EMF Technology Core

EMF project is a Modeling Framework and code generation facility for building applications based on a structured data model.

Advantages:
- "Ecore - EMF model" - model of models from which any model can be defined. This can specify application data: object attributes, relationships between objects, operations available on each object, and simple constraints on objects and relationships. This is the Class Diagram, subset of UML.
- In EMF meta-models can be defined, for example the EvidenceMetaModel – see “Evidence meta-model in Eclipse”.
- Java code can be generated using a genmodel file to create, save, and load models (instances).
- EMF supports multiple formats for model and model instance input and output. In EMF we can read an XML file representing the instance of the model.

Disadvantages:
- EMF runtime environment is in Eclipse. In OPENCOSS it would be good to have web GUI so Eclipse runtime might be not useful.

Conclusions:
We could possibly try to use EMF core, but not EMF GUI, which is Eclipse based. The GUI needs to be developed in web.

B.2 Evidence Metamodel in Eclipse

A sample Evidence meta-model has been created in Eclipse EMF (based on D4.3; Figures 6 and 7).

Then Java code representing the meta-model has been generated. Java code for Model Editor (to be run in runtime eclipse environment) has been generated (Figure 14; see EMF Client Platform below).

Advantages:
- The generated code can be customized by adding: "@generated NOT" tags (so it will not be overridden in next generations).
- The data in the model can be modified in 2 ways:
  - EMF GUI editor - described below.
  - EMF Java commands: SetCommand, AddCommand, RemoveCommand, MoveCommand, ReplaceCommand, CopyCommand – using Java commands we can also run the undo/redo actions that are not available from editor.
Disadvantages:

- When there is a need to change a meta-model we need to regenerate the source code and restart EMF Client

![Figure 14. Evidence metamodel editor](image)

**B.3 EMF Client Platform**

EMF Client Platform is an editor for model instances. It provides a tree-based view for navigation of model instances, a detailed editor for one entity of the model, and additional customized views. We have created Evidence meta-model instances (Figures 15 and 16). For example, an Artefact named "Requirement_id1" (instanceOf the ArtefactType REQUIREMENT) with some properties (Artefact "Requirement_id1" have relationship with the ArtefactRelationship: sourceOf: REQ__SOURCE_CODE_100, REQ__SOURCE_CODE_101, REQ__SOURCE_CODE_102 and targetOf: REQ__DOC1, REQ__DOC2)

**EMF Type for ArtefactType functionality:**

In the ModelEvidence metamodel we have defined ArtefactTypeProperty(s) as the property(ies) of the ArtefactType. The ArtefactProperty could be instanceOf these ArtefactTypeProperty(s). We defined Project with a model based on the created meta-model: ArtefactType REQUIREMENT with properties: ArtefactTypeProperty: REQ_ID, REQ_IS_SOURCE_CODE_REQUIRED, REQ_START_DATE. Artefact Requirement_id1 which has instanceOf ArtefactType REQUIREMENT has properties which are instanceOf ArtefactTypeProperty: ArtefactProperty: Req_id_1 (value=1), Require_Source_Code (value=true).

ArtefactProperties are the Artefact properties, and describe artefact properties and the relationship with other Artefacts. For example when Artefact: Requirement_id1 has property: Require_Source_Code with (value="true") then this artefact should have relationship: sourceOf with ArtefactRelationship: "REQ__SOURCE_CODE_xxx". We can use the information Require_Source_Code (value="true") to check in a validation process if the artefact Requirement_id1 has Link sourceOf to artefactRelationship: "REQ__SOURCE_CODE_xxx".
Advantages:
- it is possible to define bi-directional relationship by adding new EReference with EOpposite Property.

B.4 EMF Store

EMFStore is a repository for storing and versioning EMF-based models. EMF Store has been installed and storing of instances of Evidence model has been tested.
Advantages:

- EMFStore can help the application to share instances of an EMF model among multiple clients: another client can check out the same project, modify an EObject, and commit the changes to the server. A project can be updated from the server and see the changes made by another client.
- EMFStore can keep track of the version history of the model.
- EMFStore supports:
  - Model Merging during the conflicts
  - Model Migration during migration from the changes made in Meta-model (Ecore)
  - EMFCompare to see differences
- EMFStore contains Tooling and Views for: Commit/Update/Merge, Repository-Browser, History-Browser.
- It can work in offline mode.

EMFStore by default uses ".emfstore/server" path in user home directory to store the data. EMFStore can run with XML-RPC, it can run via HTTPS and work in firewall environments. To configure EMFStore we should adapt the configuration file called es.properties in the following path: {user-folder}/.emfstore/server/profiles/default/conf/es.properties

B.5 Using EOperation for validation in EMF

We have added new validation rules to the Evidence meta-model (e.g., to ArtefactType EClass), by adding EOperation with two parameters: DiagnosticanChain and Context: EMap<?,?>

Advantages:

- Validation constraints can be modeled by EOperations.
- Validation code is generated automatically.
- We need to adapt our validation rules: determine condition when validation fails, update path of validation attribute, and update error messages.

Disadvantages:
Using EOperation in validation is not useful for OPENCOSS because we need to update (change) generated source code of validation methods.

B.6 EMF Validate feature

Advantages:

- We can define invariants - implemented as class method, defined on the model
- Named constraints - defined not on the model, implemented in external validator class as a method
- Schema based constraint - defined within schema, as with Named Constraint
- When using the Validate model item we can see the problems in Validation Problems Dialog and also in Eclipse's Problems View.

Disadvantages:

- All changes to validation rules need to be done in Java code (not in runtime)
- For invariant and for constraints we need to modify the generated code by replace 'if(false)' with our conditions.
B.7 EMF Validation Framework

Advantages:
- Support for constraint providers (used to provide constraints) for any EMF meta-model. We can define:
  - Static constraints – they are declared in the plugin.xml file
  - Dynamic constraints – they are defined in models or other resources (defined at runtime)
  - Batch constraints – they are evaluated by user demand from EMF editor
  - Live constraints – they are evaluated when the model is changed
- Model traversal algorithms - only for batch validation, traverse a subset of the model from the user selection
- Constraint parsing for language: Java and OCL (Figure 17)
- Configurable constraint bindings to application contexts; define the client context for the object to validate and constraints for enforcing on these objects
- Validation listeners for validation service

We can define EMF static constraints in plugin.xml file. For example:

```xml
<constraintProvider cache="true">
  <package namespaceUri="http://modelevidence/1.0"/>
  <constraints categories="modelEvidence.validation.ArtefactTypeConstraints">
    <constraint lang="Java"
      class="modelevidence.util.ArtefactTypeConstraint"
      severity="ERROR"
      mode="Batch"
      name="Valid Artefact Type Name"
      id="modelEvidence.validation.ArtefactTypeConstraints"
      statusCode="1">
      <description>
      Artefact Type name can not be empty
      </description>
      <message>
      Artefact Type name can not be empty
      </message>
      <target class="ArtefactType">
      </target>
    </constraint>
  </constraints>
</constraintProvider>
```

In ArtefactTypeConstraint class which extends from AbstractModelConstraint we need to override validate() method. For example:

```java
@Override
public IStatus validate(IValidationContext ctx) {
    ArtefactType artefactType = (ArtefactType)ctx.getTarget();
    String artefactTypeName = artefactType.getName();
    if ( artefactTypeName == null || artefactTypeName.length() == 0 ) {
        return ctx.createFailureStatus(
            new Object[] {"empty artefact type name"});
    }

    // More validation logic...
}
```
Dynamic Constraint Provider is used as an integrator to make constraints defined in different ways, to make them available to the EMF Validation Framework. Dynamic constraint provider extends the AbstractConstraintProvider class and extends setInitializationData() method to initialize its constraints. The constraints could be read/parse from the file during the runtime environment.

Advantages:
- Using dynamic constraint provider, we can read constraints in runtime environment
- Support for OCL, users can express constraints on model in runtime. Engine to validate the OCL rules.

We can query EMF models with OCL and we can use OCL in validation. We can use Interactive OCL console which helps to create OCL query on the models:

![Interactive OCL Console](image)

**Figure 17. EMF Validation Framework**

### B.8 UNICASE

Unicase is an application written in EMF which facilitates Software Development Lifecycle. It defines models extend for requirements, tasks, defects, and use cases. Unicase has been installed and analysed.

Disadvantages
- The meta-model is organized in the packages: tasks, organization, requirement, bug. OPENCOSS Evidence model has different meta-model than Unicase meta-model. We would like to be able to add for example the new custom artefacts.

Conclusions
We cannot reuse UNICASE in OPENCOSS.
B.9 EMFTrace

EMFTrace extends EMFStore. EMFTrace adds two features:

- Adapters to import the data from the external CASE tools. We can write new adapter and meta-model which will match the data from external tool to defined meta-model.
- Rules for traceability and impact analysis

EMF developers can prepare adapters that will import data from external CASE tools into EMF model. By default some adapters are provided: jUCMNav(URN files), pure::Variants(Feature Models), Visual Paradigm (UML), Eclipse UML2 Tools (UML), and Protege (OWL). We have tested one adapter: importing UML files into EMF model. It works fine.

Advantages

i) Developers can implement adapters (add new import and export features) for import/export new CASE Tools to EMFStore-Repository,
   It requires the following steps:
   - Define new meta-model (in EMF) for the data which we want to import
   - Define XSLT template that will transform imported CASE tool data to our EMF metamodel.
   - Import: in EMFTraceUI, org.emftrace.ui.modelimport package we need to define
     NewModelImportHandler class which extends EMFTraceImportHandler.
   - Export: in EMFTraceUI, org.emftrace.ui.modelimport package we need to define
     NewModelExportHandler class which extends EMFTraceExportHandler.

ii) EMFTraceCore contains components these components:
   - AccessLayer - access to the models stored in the repository
   - LinkManager - creates and maintains traceability links
   - ProjectCleaner - maintains projects
   - RuleEngine - executes rules and process their results
   We can add new EMFTrace component.
Disadvantages:

- Dependency/traceability detection and impact analysis are not well described in documentation on EMFTrace site – we cannot use them easily. We have searched on the Internet and we did not find more useful information than on the EMFTrace page.
- We have installed and run EMFTrace but we are not able to check traceability/impact-analysis functionality because docs are not up to date.
- EMFTrace has GUI options: “Search for Dependencies”, “Search for Transitive Dependencies”, “Validate Traceability-Links” and “Validate Traces”. But they are not documented anywhere.
Appendix C. Analysis of Semantic Technologies for Evidence Management

This appendix extends the information presented in Section 5 and in Appendix A about semantic technologies with further details about their pros and cons as tools for safety evidence management. The appendix presents an implementation proposal based on research into semantic (Web 3.0) technologies and state-of-the-art knowledge management and NLP tools.

C.1 Introduction

This description is the result of research for gathering information about existing open source knowledge management (KM), document management (DM), and content management (CMS) tools to reuse for the WP6 evidence repository. Due to the fact that none of the OPENCOSS functional blocks were implemented by the time in which the research was performed and D2.3 does not place any restrictions on how the functional blocks should be implemented, a 'bigger picture' approach was taken.

As stated in D2.3, "The architecture design is therefore strongly connected to WP6 requirements and solution decisions". We should remember that OPENCOSS is not the first project aimed at solving the problem of high cost of conducting safety assurance/certification projects for safety critical systems (see D6.1). However, it has defined a very distinctive feature which none of these earlier projects delivered, such as "Model Driven Engineering (MDE) of evolutionary evidential chain". The project also aims to provide "an intelligent, automated and highly customizable safety assurance and certification tool platform integrated into existing manufacturers' development and safety assurance processes and tooling".

Utilizing semantic technologies (Web 3.0) and state-of-the-art knowledge management and NLP tools and information should help in delivering these functionalities and might in fact require a rethinking of the high level design of the OPENCOSS platform architecture. However, what is most important is the potential savings of resources in not having to solve problems already solved by many before, not committing the same errors as history of knowledge management-related IT projects shows, and not having to implement solutions which already exist as enterprise-ready open source projects. Resources can be re-focused on integration/customization of selected solutions and implementing the business logic specific to the OPENCOSS platform.

C.2 Implementing OPENCOSS with Semantic Technologies

In order to understand how semantic technologies can be a good fit to implement the OPENCOSS platform, one must re-adjust ones' thinking to the Web 3.0 paradigm. If you are not familiar with these technologies or simply want a brief overview of the logic that led to these conclusions, please refer to the Technical background section below.

Consider the following:

- Think of process and standards for a given assurance project as an ontology expressed in OWL with rules and constraints expressed in SPIN or RIF
- Think of evidence as triples (RDF) of artefacts (actually, their metadata) linked to the corresponding ontology classes
- Evidence for a given assurance project is collected by adding relevant artefacts to repository and annotating (adding metadata), or letting the system automatically extract relevant data and annotate them (i.e., "ontology-driven metadata extraction")
The result is such that:

- Thanks to OWL (with rules and constraints expressed in SPIN or RIF), artefacts metadata expressed in RDF triples and a reasoner, a system can understand the relationships and act accordingly
- Differences between standards can be reconciled using ontology mapping/alignment
- Variations of a given standard (additional country-specific rules / ability to modify pre-defined standard/process, etc.) can be reconciled by inheriting and extending a given ontology
- Evidence chains can be visualized and managed thanks to connections defined via RDF bound to ontology

The above perspective was developed as part of research into solutions for WP6 artefact repository, but can be extended onto other facets of the OPENCOSS platform, where needed. In fact, according to the D2.3, the decision on how implement "Evolutionary Evidence Chain" (WP6) impacts the entire OPENCOSS platform.

The following description is a brief analysis of how semantic technologies can be utilized in the OPENCOSS platform, on a per WP basis.

**WP4: Common Certification Language (CCL)**

- **Safety standard metamodel definition**
  
  We could use semantic technologies for safety standard metamodel definition (textual model) as an ontology (expressed in OWL) w/rules & constraints (expressed in SPIN and/or RIF). Reuse between assurance projects becomes simpler: existing ontology can be extended or (for the purpose of compositional certification) split into smaller modular ontologies (one for each module, an OWL 2 ontology can import other ontologies) and referenced in the top-level ontology.

- **Safety standard metamodel editor**
  
  For MDE of the visual model, we could utilize one of the available ontology managers, such as WebProtege (web version) or Protege (desktop version).

- **Safety standard metamodel storage**
  
  For storage of the safety standard metamodel definition (textual model) for a given assurance project, we could use a semantic repository (e.g., Neo4j or HyperGraphDB)

- **Safety standard metamodel consistency check**
  
  For consistency check of the safety standard metamodel definition (textual model), we could use a reasoner (e.g. Pellet or Hermit), which provides automatic inferencing about facts, rules and constraints expressed in the ontology, and can help in verifying consistency.

**WP5: Compositional Certification**

This needs to be further investigated. Please see "Safety standard metamodel definition" above for possible solutions (reuse/modularization). Further analysis is needed.

**WP6: Evolutionary Evidence Chain**

- **Evidence storage**
  
  Having the safety standard metamodel (textual model) defined as an ontology, artefacts’ metadata can be easily bound via triples (RDF) to appropriate ontology classes. For storage of that metadata, we could use a semantic repository (e.g., Neo4j or HyperGraphDB).

- **Evidence items editor**
  
  For MDE, visualization and management of the evidential chain, we could utilize one of the available ontology managers such as WebProtege (web version) or Protege (desktop version). Since they are open source, extending them is possible. They also provide a plugin architecture, so it is possible that simply writing a plugin would be enough. In addition, a faceted browser or other graph visualization tools could be used in conjunction for better visualization of the evidential chain graph data (e.g., VUE or Payola). Also, for adding and editing evidence items, an annotator
and/or content analyser (utilizing ontology-driven metadata extraction to bind evidence metadata as triples to appropriate ontology classes) could be used (e.g., autometa or GATE).

- **Evidence impact analysis and traceability**
  For evidence impact analysis and traceability, we could use a reasoner (e.g., Pellet or Hermit), which provides automatic inference about facts, rules and constraints expressed in the safety standard ontology and evidence metadata bound to its classes.

- **External tools integration**
  The use of semantic technologies helps in resolving the often encountered limitations in knowledge management systems, in this case overcoming the limitation of integration of data from heterogeneous sources and formats. Integration with external tools thus becomes much easier. For example, for handling of unstructured data (i.e., when working with documents such as Word/PDF, etc.), we could use a content analyser such as Apache Tika or GATE.

**WP7: Transparent Certification- and Compliance-Aware Process**

- **Safety standard compliance detection**
  For safety standard compliance detection, we could use a reasoner (e.g., Pellet or Hermit), which provides automatic inference about facts, rules and constraints expressed in the safety standard ontology and evidence metadata bound to its classes.

- **External tools integration**
  For easier integration with development and safety assurance tools (e.g., for model to model translation), we could use one of the match managers available (e.g., MapPSO or COMA).

All the tools/solutions mentioned above are based on open, approved standards or specifications. Selected ones would need to be integrated/customized and the remainder of the functionality specific to the OPENCOSS platform (as outlined in requirements) implemented in business logic. All major product level requirements for the OPENCOSS platform would thus be delivered, including the application of MDE for evolutionary chains of evidence. Through the use of semantic web technologies (Web 3.0) and state-of-the-art knowledge management and NLP tools, the "evolutionary" and "intelligent" requirements of the OPENCOSS platform could be fulfilled and give plenty of room to expand its functionalities.

The next step should be creating a reference implementation based on one of the available frameworks or applications found. For example:

- Create an OWL ontology out of the "evidence taxonomy"
- Define constraints in SPIN or RIF
- Populate a chosen semantic repository with RDF triples
- Add a simple front-end for CRUD operations and faceted browsing and/or (graph) visualization of evidence chain

At least two enterprise-ready open source frameworks for building semantic web applications already exist:

- **Apache Jena**
  Code: [https://github.com/apache/jena](https://github.com/apache/jena)
  Ohloh summary: [http://www.ohloh.net/p/jena](http://www.ohloh.net/p/jena)

- **Sesame**
  Code: [https://bitbucket.org/openrdf/sesame](https://bitbucket.org/openrdf/sesame)
  Ohloh summary: [http://www.ohloh.net/p/sesame](http://www.ohloh.net/p/sesame)
These frameworks integrate the functionalities of repository, reasoned, and ontology manager, and provide many other functionalities. In addition, during the course of research, several applications (with source code) were found integrating such functionalities.

C.3 Technical Background

For a more detailed description of the reasoning that led to our proposal for the utilization of semantic (Web 3.0) technologies and state-of-the-art knowledge management and NLP tools, please see the following sections.

C3.1 Metadata and Why Tags/Keywords are not Sufficient

During research on ready solutions for storing artefacts used by the OPENCOSS platform, one of the major findings was that, in essence, it would be artefacts’ metadata that would be stored which the system would then use to manage, classify and make sense of a given artefact.

For the purpose of understanding information presented in this section, it is important to distinguish between syntactic and semantic metadata. The former ('syntactic') focuses on elements such as size of the document, location of a document or date of document creation. These do not provide a level of understanding about what the document says or implies. The latter ('semantic') describes contextually relevant or domain-specific information about artefact. For example, if the content is from the business domain, the relevant semantic metadata could be company name, ticker symbol, industry, sector, executives, etc., whereas if the content is from the intelligence domain, the relevant semantic metadata could be terrorist name, event, location, organization, etc. In short, metadata elements that offer greater depth and more insight "about the document" fall under the semantic metadata category.

Tags/keywords are such examples of semantic metadata. Adding tags/keywords to a document/artefact is a simple process for an ordinary user who doesn’t need to have systematic information classification background. Tagging became well known since the emergence of Web 2.0 several years ago. Web users can classify web items of their interest by using tags that reflect users' understanding to the items collected in each tag. However, free and relatively uncontrolled vocabulary has its drawback in terms of lack of standardization and semantic ambiguity. Three of these problems are polysemy, synonymy, and basic level variation.

Explained by example, synonymy simply refers to situations where tags/keywords in a tag/keyword collection exhibit many variations, i.e. multiple tags/keywords having the same or closely related meanings. In such situations items collected under synonymous tag are supposed to be collected under a consolidated tag rather than separate tags since they have similar meaning. Polysemy refers to situations where one tag has multiple meanings. Such condition causes bigger problems because although many items may be collected under a particular tag, these items are not related to each other because their meaning is different to users who annotate those items with this tag.

Another frequent problem when dealing with information classification and knowledge management is that one tag/keyword may have semantic relationship to other tags (e.g., "inn" is a kind of "hotel" which shows "more specific" and "more general" meaning). Such a condition may not be utilized to relate items collected under these two tags because they are simply treated as two different tags. The flattened non-hierarchical structure of tags/keywords leads to low search precision and poor resource navigation. These semantic relationships between tags/keywords have not been explored even though there exist rich relationships, which could provide valuable information.

Consolidating the multiple facets (i.e. different meanings) and the relationships of tags/keywords into a consolidated entity which will help better understand the tags/keywords used and documents/artefacts
annotated with them becomes desirable. Contextualizing artefacts added to an evidence repository requires a way of indicating how one artefact relates to other artefact(s) that is more powerful and more expressive than tags/keywords. We should re-imagine tags/keywords as rich connections that relate artefacts. Real relationships make explicit what tags/keywords implicitly suggest. Because relationships are so explicit, they are easy to grasp and easy to enter correctly and comprehensively. What is needed is a way of relating artefacts that is easy to use and comprehend even for people who are not trained librarians.

Several possible solutions exist, including using classification systems such as taxonomy or using conceptualization systems such as ontology. Taxonomies and ontologies could be viewed as a subset of NLP. NLP originally meant machines that would have full comprehension of human speech, i.e. that you could chat with your computer asking questions and getting intelligent answers. Although that level of computation is generally not available, much more modest subsets of language-based techniques exist. We consider features like automatic language detection, stemming, thesaurus support, entity extraction and phonetic matching all to be subsets of NLP that actually work. Automatically matching new artefacts to nodes in a taxonomy tree / ontology also counts as a form of contemporary NLP. As ontologies are far more powerful and can be used as the textual model for the CCL, that is what we focus on.

The use of an ontology provides the context for creating accurate semantic metadata. It is the key to provide actionable information and business insight. From data integration to application integration, the value of metadata has been long recognized. However, only with the progression from syntax and structure to semantics, an increase in the control and the creation of business insight from documents/artefacts will occur. It is through semantic metadata that both humans and software can associate meaning with documents/artefacts.

To extract optimal value from a document/artefact and to make it usable, obviously that document/artefact needs to have appropriate tags/keywords associated with it by analyzing and extracting relevant information of semantic interest. Some of the techniques available that can be used to achieve this, based on extracting syntactic and semantic metadata from documents/artefacts, include:

- Dictionary and thesauri - by matching words, phrases or parts of speech with a static or periodically maintained dictionary and thesaurus
- Document analysis - by looking for patterns and co-occurrences and applying predefined rules to find interesting patterns within and across documents/artefacts
- Ontologies - by capturing domain-specific knowledge including entities and relationships, both at a definitional level and capturing real-world facts or knowledge at an instance or assertional level
- Ontology-driven metadata extraction - assuming the ontology is kept up to date, this is the most flexible and comprehensive way, allowing to model fact-based domain-specific relationships between entities/artefacts

Utilizing the ontology-driven metadata extraction technique, the semantic metadata extracted from a document/artefact would include both direct relationships and indirect relationships. An important characteristic of semantic metadata is that it includes named relationships, while traditional statistical and concurrence analysis lead only to unnamed relationships. Such named relationships can tell us why entities/artefacts are related, which enables more automation and deeper insight. Through the use of semantic associations, entities not explicitly mentioned in the text can also be inferred or linked to a document/artefact. This type of linking is referred to as "indirect relationships". These relationships are application specific and completely customizable. Thanks to their inclusion, it is possible to traverse relationship chains to more than one level from within the document.

The task of organizing the many bits of information for hundreds or thousands of documents/artefacts into rigorous data structures with the intent to have the system communicated and make sense of all these
facts, brings us to the alphabet soup of RDF, OWL, SPARQL, etc. (in general, semantic (Web 3.0) technologies), where the line between classic ontology and metadata blurs.

Semantic metadata in conjunction with semantic (Web 3.0) technologies (e.g., OWL and RDF) can fill a critical role in satisfying a number of requirements, providing solutions to:

- Extract, organize, and standardize/normalize information from many disparate and heterogeneous sources (including structured, semi-structured, and unstructured sources) and formats (database tables, Word documents, PDF documents, other internal documents and media, etc.), and static and dynamic sources that may be internal or external
- Identify interesting and relevant knowledge (entities and relationships between them) from heterogeneous sources and formats
- Analyze and correlate extracted information to discover previously unknown or non-obvious relationships between documents/artefacts and/or entities based on semantics that can help in making decisions
- Enable high levels of automation in the processes of extraction, normalization and maintenance of knowledge and content for improved efficiency
- Make efficient use of the extracted knowledge and content by providing tools that enable fast and high-quality querying, browsing and analysis of relevant and actionable information

As becomes evident, to realize the key distinctive features that the OPENCOSS platform wants to provide and by which it will distinguish itself from other similar projects that aimed at tackling the high cost of conducting safety assurance / certification projects for safety critical systems (i.e., providing "Model Driven Engineering (MDE) of evolutionary evidential chain" and being "an intelligent, automated and highly customizable safety assurance and certification tool platform integrated into existing manufacturers' development and safety assurance processes and tooling"), the proper extraction from documents/artefacts and utilization of semantic metadata can only be accomplished with the help of standardized semantic (Web 3.0) technologies.

Since we will be dealing mostly with metadata - i.e. metadata of evidence artefacts (documents' syntactic and semantic metadata expressed in RDF) and metadata of process/standard (CCL textual model expressed in e.g. OWL w/o SPIN rules), these technologies appear to be quite useful to implement OPENCOSS platform distinctive functionalities.

**C3.2 Types of Repositories**

Our proposed solution for the storage of metadata of evidence artefacts (RDF) and process/standard textual model (OWL), a semantic repository, is part of the NoSQL family. As will be explained in more detail in the next section, relational databases are simply not optimized for storage and handling of metadata.

Semantic repository/triple stores are the only standardized NoSQL solutions available at the moment. Built on a simple and uniform data model and providing a powerful, declarative query language, these systems offer data portability and toolchain interoperability. There are dozens of competing implementations available, meaning one does not have to commit to a specific product or vendor.

For those new to NoSQL storage systems, a brief overview is presented. As relational database systems (RDBMS) are well known and we concentrate on NoSQL solutions, they are omitted from the following comparison.

Present-day family of NoSQL databases/stores can be divided into seven categories, but for a narrow sense, NoSQL generally means only the following four groups:

- 'Key-value databases' - anyone who has worked with the likes of the Berkeley DB should know these systems; they are as simple as databases, being in essence variations on the theme of a
persistent hash table; examples include 'MemcacheDB', 'Tokyo Cabinet', 'Redis', 'SimpleDB', 'Amazon Dynamo', 'Voldemort', 'Dynomite', 'KAI', 'Azure Table Storage', 'MEMBASE', 'Riak', 'LevelDB', 'Chordless'.

- 'Document databases' - key-value stores that treat stored values as semi-structured data instead of as opaque blobs; examples include 'CouchDB', 'MongoDB', 'Terrastore', 'ThruDB', 'OrientDB'.

- 'Wide-column databases' - also known as 'Tabular databases', these draw inspiration from Google's 'BigTable' model; some of the examples include 'Cassandra', 'Hadoop/HBase', 'Hypertable', 'Cloudata', 'Cludera', 'SciDB', 'HPCC', 'Stratosphere'.

- 'Graph databases' - also known as 'semantic repository', 'metastore', 'triple store', and 'RDF database'; their similarity to relational databases is that they also allow for storage, querying, and management of structured data; however, the differences are quite substantial in that graph databases use ontologies as semantic schemata, which allows them to automatically reason about the data, and work with flexible and generic physical data models (e.g. graphs), which allows them to easily interpret and adopt "on the fly" new ontologies or metadata schemata; in simple terms, graph databases offer easier integration of diverse data and more analytical power; examples include 'Neo4j', 'InfoGrid', 'HyperGraphDB', 'AllegroGraph', '4store', 'Virtuoso', and many others.

The remaining three of the NoSQL family (i.e., 'object database', 'XML database' and 'multi-value database') can be omitted as they belong to the 'old world' and the key characteristics of NoSQL solutions are not very obviously reflected in them.

**C3.3 Why A Semantic Repository**

The short answer, as was hinted at in the previous section, is that relational databases are not optimized for storage and handling of metadata. As we will be dealing mostly with metadata - i.e., metadata of evidence artefacts (documents' syntactic and semantic metadata expressed in RDF) and metadata of process/standard (CCL textual model expressed in e.g. OWL w/ SPIN rules) - large numbers of table joins encountered by relational database systems when dealing with such data is the first and major argument against their use.

The three major differences between relational databases and semantic repositories are such that:

- There is no need to make schemas beforehand
- There is no need to link tables because one-to-many relationships can be done directly
- New data attributes (predicates) can be added on the fly and will be instantly available for querying because everything is automatically indexed

Semantic repositories are a completely new database optimized for metadata management. They embrace and build upon W3C's Linked Data technology stack and are the only standardized NoSQL solutions available at the moment. They use ontologies as semantic schemata, allowing them to automatically reason about the data, and work with flexible and generic physical data models (e.g. graphs, metadata in the form of short subject-predicate-object statements), which allows them to easily interpret and adopt "on the fly" new ontologies or metadata schemata. They also provide a powerful, declarative query language. As they are standardized and there are dozens of competing implementations available, if we decide on using such a type of database, we do not have to commit to a specific product or vendor.

Data stored in such repositories (RDF) can be thought of in terms of a decentralized directed labelled graph where arcs start with subject URIs, are labelled with predicate URIs, and end up pointing to object URIs or scalar values. Other valid ways to understand RDF data include the resource-centric approach (which maps well to object-oriented programming paradigms and to RESTful architectures) and the statement-centric view (the object-attribute-value or EAV model).

Semantic repositories have benefits such as:
• A simple and uniform standard data model - data models of RDF database systems that all share the same well-specified and W3C-standardized data model at their base

• A powerful standard query language - semantic repositories offer SPARQL, a standardized and interoperable query language that even non-programmers can make use of, and one which meets or exceeds SQL in its capabilities and power while retaining much of the familiar syntax

• Standardized data interchange formats - triple stores all have import/export capability based on well-defined, standardized, entirely implementation-agnostic serialization formats such as N-Triples and N-Quads

• Data portability - easy switching between competing database systems in-house or sharing data with external parties

• Toolchain interoperability - libraries and toolchains for RDF are typically only loosely coupled to any particular DBMS implementation, so if one learns how to use and program with Jena or Sesame for Java and Scala, RDFLib for Python, or RDF.rb for Ruby, it generally does not matter which particular RDF-based system one is accessing and RDF-based code does not need to change merely because one wants to do the equivalent of switching from MySQL to PostgreSQL

• No vendor or product lock-in - unlike switching between two non-RDF solutions, if the RDF database solution A was easy to get going with but eventually for some reason hits a brick wall, just switch to RDF database solution B or C or any other of the many available interoperable solutions

• Future proof - with RDF emerging as the definitive standard for publishing Linked Data on the web and being entirely built on top of indelibly-established lower-level standards like URIs, it's not an unreasonable bet that your RDF data will still be usable as-is by, say, 2038, something that cannot be asserted for any of the other NoSQL solutions out there at the moment, many which will inevitably prove to be rather short-lived in the big picture

• Support for globally-addressable row identifiers and property names

• Web-wide decentralized and dynamic schemas

• Data modelling standards and tooling for creating and publishing such schemas

• Metastandards for being able to declaratively specify that one piece of information entails another

• Inference engines that implement such data transformation rules

Quite simply, semantic repositories provide more flexibility. For example, adding new predicates (think columns in relational databases) is much easier than writing complicated ad hoc queries or performing inferencing and rule processing.

Also, as was already mentioned briefly, they are a standardized solution, which is very important since to make a leap forward in technology, standards are often a critical factor. There is a standard for the triple approach (RDF, RDFS, OWL, SPARQL, etc.). As per W3C website, in just a few years these have become the most important development next to HTML5. As with SQL databases, their big initial success was that there was a standard. This allowed companies to train new people very easily on that (back then) new technology and the standard made it easy to escape from one database company to the other. Competition between companies focused on features and performance but not on the basic access and storage technology.

Semantic repositories will soon be as robust, user-friendly and manageable as relational databases. RDBMSes may continue to perform a bit better on simple joins, but semantic repositories already produce better performance when it comes to complicated queries, rule handling and inferencing.

Given this robustness and usability, we decided to utilize semantic technologies in the ways mentioned in the section Implementing OPENCOSS utilizing semantic technologies. A semantic repository for the storage of metadata used by the OPENCOSS platform is a much more flexible and a future proof solution than a relational database.
C3.4 Benefits of Semantic Technologies

Put simply, semantic technologies are the next evolutionary step of the Internet, facilitating the emergence of what is already called Web 3.0. They can be applied in a variety of ways, for example:

- Data integration
- Improved search and exploration of data
- Decision assistants
- Business intelligence
- Knowledge management
- Content/document management
- Providing the framework for data to be utilized across application and enterprise boundaries
- Establishing a common format for combining information from various sources of data

The OPENCOSS platform can be classified as a type of knowledge management (KM) system and semantic (Web 3.0) technologies are nowadays driving a paradigm shift in KM, allowing to overcome limitations of traditional KM systems, (i.e., factors related to system quality and knowledge quality). As Michael K. Bergman, CEO and co-founder of 'Structured Dynamics', a provider of services and products for the semantic enterprise, mentions in his "Seven Pillars of the Open Semantic Enterprise" publication: "Thirty years of disappointing knowledge management projects and much wasted money and effort compel that better ways must be found".

Capabilities such as semantics and machine processability, enabled by semantic technologies, give the Semantic Web-driven KM system the potential for semantic integration and reducing information overload by:

- Overcoming the search limitation - semantic technologies enable us to overcome the barriers to knowledge retrieval, as all resources (i.e., internal or external documents/artefacts) are represented in RDF as metadata. This representation method makes it possible for users to query and get answers as if they are using database management systems. Inferencing/reasoning, context-aware capabilities, and the fact that even a specific part of a document may be represented as RDF, further enhance the ability to search for suitable knowledge.
- Overcoming the integration limitation - data integration obstacles such as when integrating unstructured data or knowledge from sources such as HTML, word processor files, and spreadsheet files, can also be overcome thanks to understanding of the meanings of the terms and automatically processing them by exploiting the RDF and ontology.
- Improvements of knowledge quality - through the use of technologies such as RDF and ontology, they provide ability for knowledge representation as well as reasoning for the KM system. As a result, users of the system can find more relevant and accurate knowledge and improve knowledge quality.

Semantic technologies can also help in avoiding some of the typical causes of knowledge-based IT failures, in essence rooted in a closed-world approach. As each assurance project can introduce variations to an already defined standard/process (additional country-specific rules, ability to modify pre-defined standards/processes, etc.), we are dealing with what is defined as an open-ended problem, and semantic technologies are particularly well suited for such open-ended problems where a complete data model is not known. This ability of a system built utilizing semantic technologies to incorporate new facts as needed, including new kinds of data not anticipated at the beginning, is in a sharp contrast to those for a relational database, where the first step is necessarily a prescribed definition of what will go inside and changing relatively basic elements in a relational database in order to incorporate new facts is often difficult and requires significant technical expertise. We will be also dealing with another open-ended problem, i.e., unstructured data (e.g., Word/PDF/Excel documents, when integrating with external tools, for which semantic technologies are also a perfect fit.
One of the requirements of the OPENCOSS platform is the ability to reuse standard/processes defined and artefacts collected as part of a previous assurance project. But reusing is only possible if the user knows what project and what part to look at and here also semantic technologies provide solution with linked data.

Semantic technologies enable data integration and interoperability. As was explained in more detail in the section "Metadata and why tags/keywords are not sufficient", semantic technologies are very useful for solving information classification problems and enable data integration and interoperability, by combining the expressivity of ontologies with support for automated reasoning (inference) and semantic metadata extraction. Matched with NLP techniques to integrate unstructured data, they allow to incorporate information locked up in documents and overcome the limitations of manually assigned tags/keywords. In addition, the RDF data model is capable of capturing any existing data relationship, and ontologies are capable of capturing any existing information schema. As a notable side effect, due to richness of available metadata, ontologies and automatic inferencing, many types of analyses (e.g., traceability) are possible with minimal effort.

Semantic technologies improve search and discovery. The graph structures of semantic schema means that any node can become an entry point to the knowledge space for discovery. The traversal of information relationships occurs from the selection of predicates or properties that create this graph structure in the first place.

C.4 Glossary

This section presents a glossary of semantic technologies (Web 3.0) terms.

**Closed World Assumption**
It is the presumption that what is not currently known to be true is false. "Closed World Assumption" (CWA) also has a logical formalization. It is the most common logic applied to relational database systems and is particularly useful for transaction-type systems. In knowledge management, the closed world assumption is used in at least two situations:

1. When the knowledge base is known to be complete (e.g., a corporate database containing records for every employee), and;
2. When the knowledge base is known to be incomplete but a "best" definite answer must be derived from incomplete information.

Compare to the "open world assumption" (defined below).

**Folksonomy**
A folksonomy is a user-generated set of open-ended labels called tags organized in some manner and used to categorize and retrieve Web content such as Web pages, photographs, and Web links.

**Inferencing**
Inferencing is the act or process of deriving logical conclusions from premises known or assumed to be true. The logic within and between statements in an ontology is the basis for inferring new conclusions from it, using software applications known as inference engines or reasoners.

**Knowledge base**
A knowledge base (abbreviated KB or kb) is a special kind of database for knowledge management. A knowledge base provides a means for information to be collected, organized, shared, searched, and utilized.
**Linked data**
Linked data is a set of best practices for publishing and deploying instance and class data using the RDF data model. It uses uniform resource identifiers (URLs) to name the data objects. The approach exposes the data for access via the HTTP protocol, while emphasizing data interconnections, interrelationships and context, useful to both humans and machine agents.

**Metadata**
Metadata (metacontent) is supplementary data that provides information about one or more aspects of the content at hand such as means of creation, purpose, when created or modified, author or provenance, where located, topic or subject matter, standards used, or other annotation characteristics. It is “data about data”, or the means by which data objects or aggregations can be described. Contrasted to an attribute, which is an individual characteristic intrinsic to a data object or instance, metadata is a description about that data, such as how or when created or by whom. In content management and information architecture, metadata generally means “information about objects”, that is, information about a document, an image, a reusable content module.

**Metamodelling**
Metamodelling is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modelling a predefined class of problems.

**Natural language processing**
NLP is the process of a computer extracting meaningful information from natural language input and/or producing natural language output. NLP is one method for assigning structured data characterizations to text content for use in semantic technologies. Hand assignment is another method. Some of the specific NLP techniques and applications relevant to semantic technologies include automatic summarization, coreference resolution, machine translation, named entity recognition (NER), question answering, relationship extraction, topic segmentation and recognition, word segmentation, and word sense disambiguation, among others.

**NoSQL**
“Not only SQL” (NoSQL) is a loosely-defined umbrella moniker for describing the new generation of non-relational database systems that have sprung up in the last several years. These systems tend to be inherently distributed, schema-less, and horizontally scalable.

**Notation 3**
Notation3, or N3 as it is more commonly known, is a shorthand non-XML serialization of Resource Description Framework models, designed with human-readability in mind. N3 is much more compact and readable than XML RDF notation. The format is being developed by Tim Berners-Lee and others from the Semantic Web community. N3 has several features that go beyond a serialization for RDF models, such as support for RDF-based rules.

**Ontology**
An ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. Loosely defined, ontologies on the Web can have a broad range of formalism, or expressiveness or reasoning power. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. Ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. From the explicitly declared facts, an inference mechanism allows to elicit additional implicit knowledge. Such a mechanism is implemented by means of ontology reasoners.
Open World Assumption

Open World Assumption (OWA) is a formal logic assumption that the truth-value of a statement is independent of whether or not it is known by any single observer or agent to be true. OWA is used in knowledge representation to codify the informal notion that in general no single agent or observer has complete knowledge, and therefore cannot make the closed world assumption. The OWA limits the kinds of inference and deductions an agent can make to those that follow from statements that are known to the agent to be true. OWA is useful when we represent knowledge within a system as we discover it, and where we cannot guarantee that we have discovered or will discover complete information. In the OWA, statements about knowledge that are not included in or inferred from the knowledge explicitly recorded in the system may be considered unknown, rather than wrong or false. Semantic Web languages such as OWL make the open world assumption.

Compare to the "closed world assumption" (defined above).

OWL

The Web Ontology Language (OWL) is part of a family of knowledge representation and vocabulary description languages, designed for defining and instantiating formal Web ontologies, based on RDF and standardized by the W3C. An OWL ontology may include descriptions of classes, along with their related properties and instances. Three different profiles are defined: OWL 2 EL, OWL 2 QL, and OWL 2 RL.

RDF

Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model but which has come to be used as a general method of modelling information, through a variety of syntax formats. The RDF metadata model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples in RDF terminology, thus expressing the relationships of resources. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. It was created mainly to support the machine-processability of relationship data.

Reasoner

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms.

Reasoning

Reasoning is one of many logical tests using inference rules as commonly specified by means of an ontology language, and often a description language. Many reasoners use first-order predicate logic to perform reasoning. Inference commonly proceeds by forward chaining or backward chaining.

RIF

The Rule Interchange Format (RIF) is a W3C Recommendation. RIF is part of the infrastructure for the semantic web, along with (principally) SPARQL, RDF and OWL. RIF includes three dialects, a Core dialect which is extended into a Basic Logic Dialect (BLD) and Production Rule Dialect (PRD).

Semantic enterprise

An organization that uses semantic technologies and the languages and standards of the semantic Web, including RDF, RDFS, OWL, SPARQL and others to integrate existing information assets, using the best practices of linked data and the open world assumption, and targeting knowledge management applications.

Semantic metadata
Metadata that describes contextually relevant or domain-specific information about content (in the right context) based on an industry-specific or enterprise-specific custom metadata model or ontology. For example, if the content is from the business domain, the relevant semantic metadata could be company name, ticker symbol, industry, sector, executives, etc., whereas if the content is from the intelligence domain, the relevant semantic metadata could be terrorist name, event, location, organization, etc. Metadata elements that offer greater depth and more insight "about the document" fall under the semantic metadata category.

**Semantic repository**

Semantic repository (also known as metastore, triple store, graph database, RDF database) is an engine similar to the database management system (DBMS). It supports storage, querying, and management of structured data. The major differences with the DBMS are such that:

- Semantic repositories use ontologies as semantic schemata; this allows them to automatically reason about the data
- They work with flexible and generic physical data models (e.g. graphs). This allows them to easily interpret and adopt "on the fly" new ontologies or metadata schemata.

As a result, semantic repositories offer easier integration of diverse data and more analytical power.

**Semantic technology**

Semantic technologies are a combination of software and semantic specifications that encodes meanings separately from data and content files and separately from application code. This approach enables machines as well as people to understand, share and reason with data and specifications separately. With semantic technologies, adding, changing and implementing new relationships or interconnecting programs in a different way can be as simple as changing the external model that these programs share. New data can also be brought into the system and visualized or worked upon based on the existing schema. Semantic technologies provide an abstraction layer above existing IT technologies that enables bridging and interconnection of data, content, and processes.

**Semantic Web**

The Semantic Web is a collaborative movement led by the World Wide Web Consortium (W3C) that promotes common formats for data on the World Wide Web. By encouraging the inclusion of semantic content in web pages, the Semantic Web aims at converting the current web of unstructured documents into a “web of data”. It builds on the W3C’s Resource Description Framework (RDF).

**SKOS**

SKOS or Simple Knowledge Organisation System is a family of formal languages designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured controlled vocabulary; it is built upon RDF and RDFS.

**SPARQL**

SPARQL (pronounced “sparkle”) is a query language for RDF data on the Semantic Web (analogous to SQL for relational databases). Its name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language.

**SPIN**

SPARQL Inferencing Notation (SPIN) is a way to represent a wide range of business rules. It is the de-facto industry standard to represent SPARQL rules and constraints on Semantic Web models. SPIN also provides meta-modelling capabilities that allow users to define their own SPARQL functions and query templates. Finally, SPIN includes a ready to use library of common functions.

SPIN can be used to:
• Calculate the value of a property based on other properties - for example, area of a geometric figure as a product of its height and width, age of a person as a difference between today's date and person's birthday, a display name as a concatenation of the first and last names
• Isolate a set of rules to be executed under certain conditions - for example, to support incremental reasoning, to initialize certain values when a resource is first created, or to drive interactive applications

**Syntactic metadata**
Syntactic metadata focuses on elements such as size of the document, location of a document or date of document creation that do not provide a level of understanding about what the document says or implies.

**Tag**
A tag is a keyword or term associated with or assigned to a piece of information (e.g., a picture, article, or video clip), thus describing the item and enabling keyword-based classification of information. Tags are usually chosen informally by either the creator or consumer of the item.

**Taxonomy**
In the context of knowledge systems, taxonomy is the hierarchical classification of entities of interest of an enterprise, organization or administration, used to classify documents, digital assets and other information. Taxonomies can cover virtually any type of physical or conceptual entities (products, processes, knowledge fields, human groups, etc.) at any level of granularity.

**Topic Map**
Topic maps are an ISO standard for the representation and interchange of knowledge. A topic map represents information using topics, associations (similar to a predicate relationship), and occurrences (which represent relationships between topics and information resources relevant to them), quite similar in concept to the RDF triple.

**URI**
Uniform Resource Indicator: A global identifier for the Web, standardized by joint action of the W3C and IETF. A URI may or may not be resolvable on the Web (see URL).

**URL**
Uniform Resource Locator (URL) is a global identifier for Web resources standardized by joint action of the W3C and IETF. A URL is resolvable on the Web and is commonly called a "Web address".

**Vocabulary**
A vocabulary in the sense of knowledge systems or ontologies are controlled vocabularies. They provide a way to organize knowledge for subsequent retrieval. They are used in subject indexing schemes, subject headings, thesauri, taxonomies and other form of knowledge organization systems.

**W3C**
World Wide Web Consortium (W3C) is an international community that develops standards for the World Wide Web, such as HTML, XML and RDF.

C.5 References

• Christopher Bunk. "Applying Semantic Technology to Knowledge Management" (http://km.nasa.gov/pdf/182796main_bunk_applying_semantic.pdf)
• Endang Djuana, Yue Xu, Yuefeng Li. "Constructing tag ontology from folksonomy based on WordNet" (http://www.academia.edu/2386623/Constructing_tag_ontology_from_folksonomy_based_on_WordNet)
• Michael K. Bergman. "Glossary of Semantic Technology Terms" (http://www.mkbergman.com/1017/glossary-of-semantic-technology-terms/)
• Arto Bendiken. "How RDF Databases Differ from Other NoSQL Solutions" (http://blog.datagraph.org/2010/04/rdf-nosql-diff)
• Amit Sheth. "Semantic Meta Data for Enterprise Information Integration" (http://www.information-management.com/issues/20030701/6962-1.html)
• Stijn Debrouwere. "Tags don't cut it" (http://stdout.be/2010/04/07/tags-dont-cut-it/)
• New Idea Engineering, Inc. "What’s the difference between Taxonomies and Ontologies?" (http://www.ideaeng.com/taxonomies-ontologies-0602)
Appendix D. Questionnaire for Survey on Impact Analysis

This appendix presents the questionnaire for a survey on the state of the practice concerning safety evidence change impact analysis. The survey is being designed in collaboration with Markus Borg (Lund University, Sweden), Krzysztof Wnuk (Lund University, Sweden), and Leon Moonen (Simula Research Laboratory, Norway). Data collection is currently being conducted.

In the questionnaire, a PR sign (PR) denotes that the position of a page has been randomized, an asterisk (*) denotes that the corresponding question requires an answer, and an OR sign (OR) denotes that the options of the question to answer about have been randomized.

SURVEY ON SAFETY EVIDENCE CHANGE IMPACT ANALYSIS FOR CRITICAL SYSTEMS

Introduction

Most critical computer-based and software-intensive systems in domains such as avionics, railway, and automotive are subject to some form of safety assessment by a third party (e.g., a certification authority) as a way to ensure that these systems do not pose undue risks to people, property, or the environment. The most common type of assessment is compliance with safety (or safety-related) standards, usually referred to as safety certification. Examples of safety standards include the general IEC61508 standard for electrical/electronic/programmable electronic devices in a wide range of industries, and more specific standards such as DO-178C for avionics, the CENELEC standards for railway, and ISO26262 for the automotive sector.

Demonstration of compliance with a specific standard involves gathering and providing convincing safety evidence. By safety evidence, we refer to the artefacts that contribute to developing confidence in the safe operation of a system and that are used to show the fulfilment of the criteria of a safety standard. Examples of artefact types that can be used as evidence include safety analysis results, testing results, reviews, and source code.

Such artefacts can evolve during the system lifecycle. The corresponding changes must be managed and change impact analysis might be necessary in order to guarantee that the changes do not jeopardise system safety or compliance with a standard. By safety evidence change impact analysis, we refer to the activity that attempts to identify, in the body of safety evidence, the potential consequences of a change. Possible consequences can be the need for adding, modifying, or revoking some artefact.

The purpose of this survey is to gain insights into how industry deals with safety evidence change impact analysis. The survey is part of the work in OPENCOSS (http://www.opencoss-project.eu/), a European research project that is developing an open-source infrastructure for safety assurance and certification of critical systems. Your answers will help us to develop solutions that fit the current practices and needs regarding safety evidence change.

The survey is targeted at practitioners that are or have been involved in safety evidence change impact analysis. This includes people who provide safety evidence (e.g., safety engineers or testers of a company that supplies components), people who check safety evidence (e.g., an independent safety assessor), and people who request safety evidence (e.g., a person that represents a certification authority).
Completing the survey is expected to take less than 20 minutes. Please answer the questions in the context of the projects targeted at developing a safety-critical system in which you have participated. All the responses will be held confidential and anonymous.

Finally, if you are interested in the results of the survey, please contact Jose Luis de la Vara (jdelavara@simula.no) or Markus Borg (markus.borg@cs.lth.se).

Thank you very much for your participation in the survey.

**Background information**

*How did you find this survey?*
- Post on LinkedIn
- Post on a mailing list
- Personal invitation
- Other - please specify:

*What is the main application domain in which you have worked on safety evidence change impact analysis?*
- Aerospace
- Automotive
- Avionics
- Defence
- Machinery
- Maritime
- Medical
- Mining
- Nuclear
- Off-highway equipment
- Oil and gas
- Process automation
- Railway
- Robotics
- Trucks
- Other - please specify:

*In relation to what safety standards have you been involved in safety evidence change impact analysis?*  

*In what country or countries have you principally worked upon safety evidence change impact analysis?*  

*What is the main role of the organization for which you have worked regarding the development of safety-critical systems?*
- Certification authority
- Component supplier
- Consultant
- Developer/manufacturer of final systems
- Development tool vendor
- Independent safety assessor
*What is your main role in the organization?*

*How long have you been involved in activities related to safety evidence change impact analysis?*
- Less than 1 year
- Between 1 and 2 years
- Between 2 and 5 years
- Between 5 and 10 years
- More than 10 years

*How many projects dealing with safety evidence change impact analysis have you participated in?*
- Less than 5 projects
- Between 5 and 10 projects
- More than 10 projects

Circumstances under which safety evidence change impact analysis is addressed

Safety evidence change impact analysis might be performed in different scenarios and for different artefact types used as safety evidence. You will be asked about these aspects in this section.

*How often have you been involved in safety evidence change impact analysis in these general situations?*

**Frequency:**
- Never
- Few projects (i.e., rarely)
- Some projects (i.e., sometimes)
- Most of the projects (i.e., very often)
- Every project (i.e., always)

**Situations:**
- Reuse of existing components in a new system
- Modification of a new system during its development
- Modification of a new system as a result of its verification and validation
- Modification of a system during its maintenance
- New safety-related request from an assessor or a certification authority
- Re-certification of an existing system after some modification
- Re-certification of an existing system for a different operational context
- Re-certification of an existing system for a different standard
- Re-certification of an existing system for a different application domain

If you would like to add any further general situations in which you have been involved in safety evidence change impact analysis, please do so in the box below, and also indicate their frequency (for example, Situation X: some projects; Situation Y: few projects, etc.)
*For the artefacts used as safety evidence, how often is safety evidence change impact analysis performed as a consequence of changes in the following artefact types? OR

Frequency:
- Never
- Few projects
- Some projects
- Most of the projects
- Every project
- I don’t know

Artefact types:
- System Lifecycle Plans (e.g., development plans, validation and verification plans, modification procedures, and operation procedures)
- Reused Components Information (e.g., historical service data and reliability specifications)
- Personnel Competence Specifications (e.g., personnel training and experience assessment)
- Safety Analysis Results (e.g., the results from Fault Tree Analysis and Failure Mode and Effects Analysis)
- Assumptions and Operation Conditions Specifications (e.g., the constraints on the working environment of a system)
- Requirements Specifications (e.g., safety requirements or performance requirements)
- Architecture Specifications (e.g., system components and AADL diagrams)
- Design Specifications (e.g., the internal characteristics of system components and SysML diagrams)
- Traceability Specifications (e.g., the relationships between requirements and test cases and between requirements and design)
- Test Case Specifications (e.g., the inputs, execution conditions, and predicted results using a system)
- Tool Supported Validation and Verification Results (e.g., testing results, simulation results, and formal verification results)
- Manual Validation and Verification Results (e.g., inspection results and review results)
- Source Code (e.g., Ada code or C code)
- Safety Cases (documented argument aimed at providing a compelling, comprehensive, and valid case that a system is safe for a given application in a given operating environment)

If you would like to add any further artefact types from which safety evidence change impact analysis is performed, please do so in the box below, and also indicate their frequency (for example, Artefact type X: some projects; Artefact type Y: few projects, etc.)

*For the artefacts used as safety evidence, how often are the following artefact types affected by changes to the body of safety evidence? OR

Frequency:
- Never
- Few projects
- Some projects
- Most of the projects
- Every project
- I don’t know

Artefact types:
- System Lifecycle Plans
- Reused Components Information
- Personnel Competence Specifications
- Safety Analysis Results
- Assumptions and Operation Conditions Specifications
- Requirements Specifications
If you would like to add any further artefact types affected by changes to the body of safety evidence, please do so in the box below, and also indicate their frequency (for example, Artefact type X: some projects; Artefact type Y: few projects, etc.)

<table>
<thead>
<tr>
<th>Artefact types:</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Lifecycle Plans (e.g., development plans, validation and verification plans, modification procedures, and operation procedures)</td>
</tr>
<tr>
<td>Reused Components Information (e.g., historical service data and reliability specifications)</td>
</tr>
<tr>
<td>Personnel Competence Specifications (e.g., personnel training and experience assessment)</td>
</tr>
<tr>
<td>Safety Analysis Results (e.g., the results from Fault Tree Analysis and Failure Mode and Effects Analysis)</td>
</tr>
<tr>
<td>Assumptions and Operation Conditions Specifications (e.g., the constraints on the working environment of a system)</td>
</tr>
<tr>
<td>Requirements Specifications (e.g., safety requirements or performance requirements)</td>
</tr>
<tr>
<td>Architecture Specifications (e.g., system components and AADL diagrams)</td>
</tr>
<tr>
<td>Design Specifications (e.g., the internal characteristics of system components and SysML diagrams)</td>
</tr>
<tr>
<td>Traceability Specifications (e.g., the relationships between requirements and test cases and between requirements and design)</td>
</tr>
<tr>
<td>Test Case Specifications (e.g., the inputs, execution conditions, and predicted results using a system)</td>
</tr>
</tbody>
</table>

**Tool support**

Tools can support and facilitate safety evidence change impact analysis. Such tools can vary depending on the artefact types from which the analysis originates. For example, an organization can use some change impact analysis tool for requirements or for source code. It is also usually necessary to show how the change, its consequences, and the actions to address the consequences have been managed. We refer to this information as evidence of safety evidence change management. Such information might be stored in some tool. You will be asked about these aspects in this section.

*For the artefacts used as safety evidence, please rank the level of automation offered by the tool support used for performing an impact analysis when the following artefact types change. OR*

**Levels of automation:**
- Fully manual (no automation in the process; e.g., impact determined by reading documentation and asking colleagues)
- Decision support available (limited support for narrowing down a selection of possible impact; e.g., search tool used to seek impact, repositories easy to browse thanks to information structure)
- Semi-automated recommendations (tools suggest artefacts that might be impacted but humans must confirm)
- Highly automated recommendations (tools report impact and humans have the authority to veto the suggestions)
- Automatic impact analysis (tools determine the impact without human involvement)
- I don’t know

**Artefact types:**
- System Lifecycle Plans (e.g., development plans, validation and verification plans, modification procedures, and operation procedures)
- Reused Components Information (e.g., historical service data and reliability specifications)
- Personnel Competence Specifications (e.g., personnel training and experience assessment)
- Safety Analysis Results (e.g., the results from Fault Tree Analysis and Failure Mode and Effects Analysis)
- Assumptions and Operation Conditions Specifications (e.g., the constraints on the working environment of a system)
- Requirements Specifications (e.g., safety requirements or performance requirements)
- Architecture Specifications (e.g., system components and AADL diagrams)
- Design Specifications (e.g., the internal characteristics of system components and SysML diagrams)
- Traceability Specifications (e.g., the relationships between requirements and test cases and between requirements and design)
- Test Case Specifications (e.g., the inputs, execution conditions, and predicted results using a system)
- Tool Supported Validation and Verification Results (e.g., testing results, simulation results, and formal verification results)
- Manual Validation and Verification Results (e.g., inspection results and review results)
- Source Code (e.g., Ada code or C code)
- Safety Cases (documented argument aimed at providing a compelling, comprehensive, and valid case that a system is safe for a given application in a given operating environment)

If you would like to add any further artefact types and the level of automation for performing an impact analysis when they change, please do so in the box below (for example, Artefact type X: fully manual; Artefact type Y: semi-automated recommendations, etc.)

<table>
<thead>
<tr>
<th>Artefact Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Lifecycle Plans</td>
<td></td>
</tr>
<tr>
<td>Reused Components Information</td>
<td></td>
</tr>
<tr>
<td>Personnel Competence Specifications</td>
<td></td>
</tr>
<tr>
<td>Safety Analysis Results</td>
<td></td>
</tr>
<tr>
<td>Assumptions and Operation Conditions Specifications</td>
<td></td>
</tr>
<tr>
<td>Requirements Specifications</td>
<td></td>
</tr>
<tr>
<td>Architecture Specifications</td>
<td></td>
</tr>
<tr>
<td>Design Specifications</td>
<td></td>
</tr>
<tr>
<td>Traceability Specifications</td>
<td></td>
</tr>
<tr>
<td>Test Case Specifications</td>
<td></td>
</tr>
<tr>
<td>Tool Supported Validation and Verification Results</td>
<td></td>
</tr>
<tr>
<td>Manual Validation and Verification Results</td>
<td></td>
</tr>
<tr>
<td>Source Code</td>
<td></td>
</tr>
<tr>
<td>Safety Cases</td>
<td></td>
</tr>
</tbody>
</table>

If you would like to add any further artefact types and the tools that are used for performing an impact analysis when they change, please do so in the box below (for example, Artefact type X: tool W; Artefact type Y: tool Z, etc.)

*What tools are used to store the evidence of safety evidence change management? If you do not know the tools, please indicate "I don't know".

Challenges

When dealing with safety evidence change impact analysis, different challenges can arise and thus hinder this activity. Implicitly, this means that some improvement opportunities exist. You will be asked about these aspects in this section.

*How often have you faced or observed the following challenges regarding safety evidence change impact analysis? OR

Frequency:
- Never
- Few projects (i.e., rarely)
- Some projects (i.e., sometimes)
- Most of the projects (i.e., very often)
- Every project (i.e., always)

Challenges:
- Difficulty in estimating the effort required to manage a change
- Too coarse granularity of the traceability between artefacts to accurately know the consequences of a change
- Excessive detail of the traceability between artefacts, making traceability management more complex than necessary for impact analysis purposes
- Unclear meaning of the traceability between artefacts in order to know how to manage a change
- Insufficient traceability between artefacts to accurately know the consequences of a change
- Long time for evaluating the consequences of a change
- Insufficient confidence by assessor or certifiers in having managed a change properly
- Vast number of artefacts to trace
- Insufficient tool support
- Lack of a systematic process for performing impact analysis
- Difficulty in determining the effect of a change on system safety
- Difficulty in deciding if a component can be reused
- Difficulty in assessing system-level impact of component reuse

If you would like to add any further challenges, please do so in the box below, and also indicate their frequency (for example Challenge X: every project, very important; Challenge Y: few projects, moderately important, etc.)

*How do you think that safety evidence change impact analysis could be improved?*

Follow-up studies

Please provide the following information if you are interested in participating in follow-up studies.

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Email</th>
</tr>
</thead>
</table>