Collaborative Large-scale Integrating Project

OPENCOSS
Open Platform for EvolutioNary Certification Of Safety-critical Systems

Common Certification Language: Implementation
D4.5

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP4: Common Certification Language</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>28 June 2013</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Martijn Klabbers (TU/e)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:M.D.Klabbers@tue.nl">M.D.Klabbers@tue.nl</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>Martijn Klabbers, Luna Yaping Luo, Tom Verhoeff</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>Katrina Attwood</td>
<td>University of York</td>
</tr>
<tr>
<td>Alejandra Ruiz, Huáscar Espinoza, Angel Lopez, Idoya del Río, Ma. Carmen Palacios</td>
<td>TECNALIA R&amp;I</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>15-10-2013</td>
<td>Document creation and initial ToC</td>
</tr>
<tr>
<td>V0.2</td>
<td>30-10-2013</td>
<td>First concept draft</td>
</tr>
<tr>
<td>V0.7</td>
<td>12-11-2013</td>
<td>Ready for internal review</td>
</tr>
<tr>
<td>V0.8</td>
<td>13-11-2013</td>
<td>Ready for PB review</td>
</tr>
<tr>
<td>V1.0</td>
<td>15-11-2013</td>
<td>Release to the EC</td>
</tr>
<tr>
<td>V1.1</td>
<td>15-12-2013</td>
<td>Update after some iterations on prototyping</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Abbreviations .............................................................................................................................................................. 6  
Executive Summary ..................................................................................................................................................... 7  
1 Introduction ................................................................................................................................................... 8  
  1.1 Scope and Purpose ........................................................................................................................................... 8  
  1.2 Relationship with other Deliverables .......................................................................................................... 9  
  1.3 Structure of the Document ............................................................................................................................ 9  
2 Background .................................................................................................................................................. 10  
  2.1 Context: Usage scenarios and use cases ...................................................................................................... 10  
  2.1.1 Mappings: between concepts and between instances ........................................................................... 11  
  2.1.2 Example: Argumentation use case ......................................................................................................... 11  
  2.2 The multiple roles of the CCL .................................................................................................................... 12  
  2.2.1 CCL defines concepts required for assurance (GMM) ........................................................................ 14  
  2.2.2 CCL to trace concepts (for reuse) .......................................................................................................... 14  
  2.2.3 CCL as a communication device within the OPENCOSS platform .................................................... 15  
  2.3 Requirements of the CCL .............................................................................................................................. 15  
  2.4 Quality aspects of languages .... .................................................................................................................. 15  
  2.5 Configuration of the safety assurance project ............................................................................................ 17  
  2.5.1 Mock-up, Use case: Capture information from standards ................................................................ 18  
  2.5.2 Mock-up, Use case: Create a safety assurance project ........................................................................ 18  
  2.5.3 Mock-up, Use case: Define a safety assurance project baseline ............................................................ 20  
3 Common certification language (CCL) ........................................................................................................... 22  
  3.1 Introduction ......................................................................................................................................... 22  
  3.2 CCL concepts from D4.3/D4.4 ................................................................................................................. 23  
  3.2.1 The Generic Meta Model .................................................................................................................. 24  
  3.2.2 Class descriptions ............................................................................................................................... 25  
  3.3 Implementation alternative: extended meta-models for mappings between concepts .......................... 28  
  3.3.1 Alternative Mock-up, Use case: Capture information from standards ............................................... 29  
  3.3.2 Meta-model evolution to support reuse .......................................................................................... 29  
  3.4 Evaluation ............................................................................................................................................ 31  
4 Conclusions .................................................................................................................................................. 32  
5 References ................................................................................................................................................... 33  
6 Requirements and Quality Aspects on the CCL – a first evaluation ............................................................... 34  
7 XML representation of GMM’s reference assurance framework ..................................................................... 38
List of Figures

Figure 1: Functional decomposition for the OPENCOSS tool platform ................................................................. 9
Figure 2: Generic argument module pattern for WCET .......................................................................................... 12
Figure 3: Differences in assurance provided by WCET analysis report in IEC 61508 and DO-178B contexts (note that, 
in addition to the conceptual differences between SILs and DALs and the additional requirement for independent 
verification in DO-178B, there is a need to present the report as part of an overall Software Accomplishment 
Summary in DO-187B, rather than as a freestanding report) ........................................................................... 13
Figure 4: Capture Standard Data ........................................................................................................................ 18
Figure 5: Safety Assurance Project Wizard ......................................................................................................... 19
Figure 6: Consult or Modify Assurance Project Data ............................................................................................ 19
Figure 7: Create Project Baseline ......................................................................................................................... 20
Figure 8: Refine project baseline ........................................................................................................................ 21
Figure 9: OPENCOSS Model Framework ........................................................................................................... 23
Figure 10: Metamodel views ................................................................................................................................. 24
Figure 11: CCL Metamodel Relationships ............................................................................................................ 25
Figure 12: Reference Assurance Framework Metamodel (Part 1: Core Model Elements) ....................................... 26
Figure 13: Reference Assurance Framework Metamodel (Part 2: Inheritance Relationships) ............................... 27
Figure 14: Creating an (application domain) specific metamodel using model transformations ......................... 28
Figure 15: Modelling the ISO26262 Standard ...................................................................................................... 29
Figure 16: Meta Model Evolution ........................................................................................................................ 30
List of Tables

Table 1: IEC 61508 and DO-178B WCET guidelines......................................................................................................................... 11
Table 2: Requirements and Quality aspects on the CCL (from D4.2 [8])...................................................................................... 34
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>DoW</td>
<td>OPENCOSS Description of Work</td>
</tr>
<tr>
<td>D&lt;X,Y&gt;</td>
<td>OPENCOSS deliverable &lt;X,Y&gt;</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>T&lt;X,Y&gt;</td>
<td>OPENCOSS task &lt;X,Y&gt;</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>WP&lt;X&gt;</td>
<td>OPENCOSS work package &lt;X&gt;</td>
</tr>
</tbody>
</table>
Executive Summary

This document describes the common certification language, the framework for supporting safety assessments, including concrete syntax (notation) and quality aspects like the usability, interoperability, and understandability of the language.

This document is the first deliverable in task 4.3, Certification language implementation and tool support. This task will implement the conceptual domain model in the form of a domain-specific modelling framework and language. A DSML is any artificial language that can be used to express information or knowledge of systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure. It builds upon the deliverables of Task 4.2 Certification language conceptual model in the form of an extension to an existing modelling language. Based on the conceptual framework described in D4.4 (Common Certification Language: Conceptual Model), this deliverable describes the implementation of this conceptual model.

Here the project is exploring two alternative and complementary implementation strategies, which will be brought together to share the best of both worlds as the prototyping work evolves. From the evaluation of the approaches, The implementation of the CCL (Common Certification Language) metamodels in the GMM (Generic Metamodel) supports specific functions within the domain of safety demonstration. At this stage it provides a necessary step in achieving the OPENCOSS goals; the CCL defines the concepts for safety assurance, it can trace concepts for reuse, and offers a communication device within the OPENCOSS platform. Furthermore it is ready for expanding its automatic support in order to achieve cost reduction in recurring work, increase in safety, and form a sound basis for innovations in the safety assurance process; the goals of OPENCOSS.

For the large part the GMM is implemented and applied in the first prototypes. The implementation is based on the Eclipse EMF and provides a consistent basis for the two main implementation goals of the CCL: (1) The CCL metamodel implementation should serve as an intermediary language between tools and as a domain specific language for users of the OPENCOSS platform; therefore it should provide a machine-processable data structure to interface tools to the OPENCOSS platform and (2) it should provide a usability basis for the core and back-end of the many editors that the OPENCOSS platform should provide. The CCL is also shows face validity on the mock-ups that have been created. Furthermore, in the near future the implementation will probably meet 37 of the 41 requirements, and 18 of 20 quality aspects. At this moment the CCL is also validated by its role in the usage scenario’s and it is showing face validity in the mock-ups created for some of the use cases.

This deliverable is also the preparation of the reference (assurance) framework editor. In reflection of the ongoing prototyping work, this deliverable will be updated iteratively.
1 Introduction

1.1 Scope and Purpose

The OPENCOSS project aims to devise an OPENCOSS tool platform; it is a common certification framework that spans different vertical markets for railway, avionics and automotive industries and to establish an open-source safety certification infrastructure. 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.

Work Package 4 is working towards the development of a harmonised terminological and conceptual understanding of safety assurance across and within the OPENCOSS target domains. The core of the OPENCOSS tool platform is the Common Certification Language (CCL) will serve as an enabling technology for the approaches to compositional argumentation, evidence management and process compliance;, the main technical challenges to be developed in WPs 5-7. The CCL as an output of WP4 contains two contributions. Firstly, we will produce standardised models addressing key concerns of the project’s which models can be characterised for each standard or domain. Secondly, we will produce a means by which the terminology used for core assurance concepts can be harmonised and compared, which is called the vocabulary.

The goals of WP4 are derived from the OPENCOSS project goals as described in [1]. We feel that all goals are relevant to the CLL implementation which is there to improve the current situation in safety-critical system development. In abbreviated form the goals are:

• Demonstrate reduction of recurring costs for safety assessment by 30% or higher (G1)
• Demonstrate a potential reduction of product safety risks by 20% (G2)
• Demonstrate a potential gain for product innovation and upgrading by 20% (G3)

The goals are translated in project’s objectives, for which the relevant objectives for the CCL are listed here:

• Define a language (CCL) to improve mutual recognition agreement of safety assessments from different industrial markets. (O1)
• Define a compositional and evolutionary approach that reuses successful safety arguments (O2)
• Develop an open-source tool infrastructure that will allow developers and other safety assurance stakeholders to:
  o Reduce uncertainty and (re)certification costs by following a measurable and auditable process (O3.a)
  o Assess their compliance with safety standards and practices (O3.b)

These goals and objectives are ambitious and call for an extensive search for the best implementation. To this end, we explored two implementation approaches, each concretized in a number of the prototype editors for comparison purposes: reference framework editor, assurance project editor, argumentation editor, evidence editor, assurance process editor, mappings editor, and the future vocabulary editor. In both implementation approaches the conceptual model from D4.4 (Common Certification Language: Conceptual Model) [3], is implemented in the form of a domain-specific modelling framework and language. A domain-specific modelling language (DSML) [14] is any artificial language that can be used to express information or knowledge of systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure.

A first round of prototype implementation is there to validate the CCL conceptual work and implementation. Next to the implementation setup of the CCL, this deliverable also shows the first results of the validation as well. In the next prototyping round the approaches will be merged together to combine the best of both worlds. Decisions regarding the implementation are in the form of cons and pros in terms of understandability, extendability, for the relevant usage scenarios and use cases, illustrated in mock-ups. Figure 1 shows how these functional areas, based on the main usage scenarios, fit into the global OPENCOSS architecture.
1.2 Relationship with other Deliverables

Work Package 4 has relationships with:

- WP2 for High Level Requirements definition and Architecture
- WP5 for the compositional argumentation and the representation in the Common Certification Language (CCL) of the concepts necessary to this end,
- WP6 to manage evidence based on the concepts in the CCL.
- WP7 to manage the assessment process, metrics, and create a safety aware development process.

Within Work Package 4, the D4.5 is based on the conceptual CCL of D4.4 [3] and implements the requirements of D4.2 [8]. It is also the basis for all other technical work packages that use the CCL implementation as an input for the editors. The D4.6 will, for example, describe the implementation of the reference assurance framework editor (for modelling assurance references, such as safety standards and guidelines).

1.3 Structure of the Document

The remainder of the D4.5 is structured as follows: Chapter 2 explains the background of the CCL implementation, stating the goals of the implementation in the context of the usage scenarios and use cases, and an illustration of the implementation of the use cases in a mock-up. Chapter 0 continues to describe the actual implementation of the Common Certification Language (CCL), and a small evaluation, Chapter 3.4 presents the conclusions of this deliverable. Chapter 5 contains references to related literature, Appendix Chapter 6 shows an overview of the CCL requirements and the deployment in the first prototype, and Appendix Chapter 7 contains a textual (XML) representation of an important part of the Generic Meta Model (GMM): the Reference Assurance Framework Metamodel (RAFM).
2 Background

This chapter shows the context where the Common Certification Language (CCL) operates in, the multiple roles in the OPENCOSS platform, and the requirements and quality aspects it needs to meet. Since the CCL is a language that operates in the core of the OPENCOSS platform, sketches of the involved editors, which directly work with the CCL, are included in this chapter as well.

2.1 Context: Usage scenarios and use cases

This section presents the different usage scenarios identified for the use of the CCL, as described in D2.4 [7]. An overview of relevant use cases is given in this section as well as a description of the main contribution of the CCL to these use cases: defining the concepts and the relationships between them using mappings. The role of the CCL is demonstrated in an example argumentation use case.

The general processes of the OPENCOSS platform, where the CCL is involved in, are depicted in Figure 1. From top to bottom, we can identify the processes that are related to:

1. Prescriptive Knowledge Management, where the information of standards is captured in the standard model and can be consulted,
2. Assurance Project (lifecycle) Management, where the safety assurance project is created as well as an initial project model and the project is further defined using the standard model into a refined project model with baseline.
3. Safety Argumentation Management, where the safety claims are linked to the evidence using explicit argumentation,
4. Evidence Management, where the evidence characterisation is defined and the target evidence is constructed and evaluated, the evidence chain is traced.
5. Process (Assurance) Management, where the assurance process is checked for compliance with the reference framework and safety assurance performance are measured.
6. Interfacing with other tools, like application lifecycle management/product lifecycle management (ALM/PLM) tools, and product engineering tools.

Each usage scenario has been decomposed in specific use cases, where the most relevant for the CCL are as follows:

1. Prescriptive Knowledge Management:
   (a) Capture information from standards,
   (b) Navigate Knowledge About Standards,
2. Assurance Project Lifecycle Management:
   (a) Create a safety assurance project,
   (b) Define a safety assurance project baseline,
   (c) Navigate Safety Assurance Repository,
3. Safety Argumentation Management:
   (a) Assessment Argumentation Management,
   (b) Define Concepts Required for Assurance,
   (c) Manage Modular argumentation,
4. Evidence management:
   (a) Determine the Evidence to Provide,
   (b) Collect Evidence Item Information,
   (c) Specify Traceability Evidence Items,
5. Process Assurance Management:
   (a) Check process compliance against assurance reference framework
   (b) Measure and estimate Safety Assurance Metrics
6. Interfacing:
   (a) Communication between Tools as the OPENCOSS platform and ALM/PLM and product engineering tools.

The CCL mainly consists of concepts involved in the use cases above and the relationships between those concepts. One of the important aspects in these use cases is the relationship between concepts that the user, from safety engineer to safety assessor, needs to create and check. These user-defined or created relationships are materialized in
mappings. In the following subsections we explain more about the concept of mappings and demonstrate the use of the CCL in an argumentation example. For more information on the usage scenarios and the use cases, we refer to the D2.4, OPENCOSS Usage Scenarios.

2.1.1 Mappings: between concepts and between instances.

The user defined relationships between concepts in the various models that the CCL will support are called mappings or “partial mappings”. Partial mappings address the complexities of the relationships required in the OPENCOSS Framework and Vocabulary. In D4.4 [3], three types of mapping in OPENCOSS are described: full map, partial map, and no map. The objective of the partial mapping approach is to enable and enforce the explicit record of information concerning the degree of “map” that can be achieved between elements, for example resulting from a gap analysis between the original and reuse contexts.

From a usability perspective, those mappings can be divided into two levels: mappings between concepts and mappings between instances of concepts (on a concrete level). Mappings between concepts provide the information whether concepts in their definition or nature are the same, mappings between instances also include the instantiation or content of the concepts. For example, hazard concepts from Mil Std 822 and the ISO 26262 standard can be mapped from a conceptual point of view, however, if the underlying severity categories are different, the mapping should indicate only a partial map.

In terms of the Model Driven Architecture, these mappings can be described as follows:

- **Mapping between concepts**: The models in this mapping are all layer 2 MOF models, called M2-models. These would be, for example, the Generic Metamodel (GMM) or the Reference Assurance Framework Metamodel (RAFMM). In a reuse scenario, this implies that it only focuses on the mapping between domain concepts. Terminology will be used to support this kind of mapping. After the Vocabulary Meta Model (VMM) is fixed, most of this mapping could be done automatically. Moreover, it could support protecting users against making errors. (ISO 25010: 2011)

- **Mapping between instances**: The models in the mapping are all layer 1 MOF models, called M1-models. These would be, for example, standard models, company models or project models.

The implications on the implementation of these kinds of mappings will be apparent in Section 3.3 that describes an alternative implementation of the CCL metamodels and a refinement of the vocabulary metamodel in particular. The refinement involves that the vocabulary’s mapping between concepts is included in the specific metamodel of the reference (assurance) framework metamodel (RAFMM). The description of difference between CCL metamodels (the Generic Metamodel – GMM) concepts or classes and the RAFMM concepts or classes is indicated and documented in the model transformations from GMM to RAFMM.

2.1.2 Example: Argumentation use case

Determining worst-case execution time (WCET) is a recommended activity for safety critical systems developed according to the IEC 61508 and DO-178B guidelines. In Table 1, we could see that in these two standards, objectives and terminology for WCET are quite different. This would bring some questions like: “How can we reuse a WCET report generated in a 61508 context in a DO-178B context? In which way GMM could contribute on this?” Some answers could be found in the WP5 deliverables, like in Section 3.3 Contracts for capturing dependencies in D5.3. The argumentation meta-model in GMM is used for safety argument reuse, we introduced generic argument module pattern for WCET shown in Figure 2. In different contexts, this pattern could be specialized into different safety cases. Figure 3 shows the WCET safety cases for (a) IEC 61508 and (b) DO 178B respectively. In this case, the terminology used in both contexts can be labelled as a link to meta-model evolution chain. For example, if we want to map from 61508 SIL 4 to DO-178B level A and from evidence report to software accomplishment summary. Firstly, we could look into the conceptual mapping, using meta-models in meta-model evolution chain to have a basic idea of definition, original concepts in RAFM, mapping possibility etc. It could also detect incorrect concepts used in safety cases. For example, if the top level context is DO-178B, in the argumentation, it shouldn’t have some concepts like ASIL or SIL. If it happens, the editor should at least give an alert to users. After that, according to the ComplianceMap MM, we could finish this mapping manually.

<table>
<thead>
<tr>
<th>Table 1: IEC 61508 and DO-178B WCET guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 61508</td>
</tr>
</tbody>
</table>

FP7 project # 289011
The CCL has a number of different functions in the overall OPENCOSS solution. In the following subsections, we provide a brief discussion of the various roles of the CCL. In this section, we contextualise this discussion with a brief summary of the current status of the CCL approach.

The OPENCOSS DoW [1] envisages the common certification language as a means to secure mutual agreement on core certification concepts between the domains, and to discuss these concepts at an abstract level, ironing out the differences between the domains. In this idealistic view, the CCL can be used to build domain-specific models and allow the management and sharing of claims, evidence and arguments in a common format. Following discussions within the project, the approach presented in [2] and [3] has slightly modified this ambition for the CCL, in ways which are at once more pragmatic and more challenging.

Although there is a degree of commonality between the core concepts used to document the safety of systems across the target domains, the techniques used to assess safety and the strategies used to assert it, it is important to note that there are subtle differences between them. Many of the similarities are at a very abstract level, while at the lower levels, there are large conceptual differences between the domains. For example, although standards commonly have a concept of the criticality level of a function or component in terms of the risk errors in its operation or configuration may pose to overall system safety and the degree of mitigation required for such errors, the nature of this criticality level – and the way in which it influences the techniques used to provide assurance of system safety – differs substantially between the domains (for example, SILs and DALs are not interchangeable, but encapsulate very
different attitudes to risk). Even at a very abstract level, we may find major differences in the concepts deployed and the attitudes they encode. For example, at a very high level, any system will need to demonstrate an absence of threats to safety. But there are differences in the concepts of threats to safety which need to be accounted for at system level, and these differences result in subtle differences between the techniques and documentation practices used to demonstrate safety and the rigour and nature of the evidence that is required to demonstrate their absence. We should also note that safety engineering practice in particular domains has evolved to match the conceptual approaches in the domains. This is not simply a question of vocabulary: the concepts drive the techniques, at the most basic level. A very simple example of this is seen in the different contributions which can be made to an overall system safety argument by a report on worst-case execution time (WCET) analysis in IEC 61508 and DO-178B, which result in implications for additional information (as well as for changes in the way the evidence is presented) to be provided in a demonstration of assurance in a DO-178B context when the WCET report produced for IEC61508 is reused. This is illustrated in Figure 3.

Ostensibly, it is in everyone’s interests to develop more generic approaches to safety, so that there is closer commonality between the concepts used and communicated and the techniques and measures deployed across the domains. This way, manufacturers will find it easier to deploy components in multiple domains since the features and qualities they need to demonstrate for a particular subsystem will not vary, it will be easier for ISAs and subcontractors to work across a variety of domains, and, perhaps above all, it will be easier to communicate issues surrounding safety in an informed and balanced way to people who are not specialists in the domain – the travelling public, lawyers, journalists etc. Pragmatically, however, we note that such a universal, agreed approach to safety is not currently viable. It will be very risky, commercially and politically, for companies, regulators and the like to agree changes to their existing practice, without a clear view of their benefits – there is a degree of confidence in current domain-specific techniques which it will be difficult to recapture – or their impact on safety. Again, this is not simply a matter of vocabulary change, but requires a change in conceptualisation and the development of new, robust techniques to ensure parity. Essentially, the safety-critical industries are conservative in their attitudes and practices for good reason¹. It should also be noted that there are serious legal drawbacks to any attempt to “impose” a

¹ Note that this conservatism is not the same thing as complacently, and does not, of course, mean that there is no attempt at innovation within the industries; it is far from that. It is simply true to say that innovation in terms of improvement is perceived to be easier when the requirements for safety are clearly known and stated in terms that the domain understands. In other words, the yardstick against which the innovations will be measured are clear.
common conceptualization of safety – there may be complex issues of liability and indemnity to be considered, for example.

For these reasons, OPENCOSS has concluded that it is infeasible – at least within the timescales of the current project – to attempt produce agreed definitions of core safety concepts, and an agreed “safety language” to which domain-specific and company-specific usage can be mapped. This is discussed in detail in Section 3 and 4 of [3]. The CCL is not used to provide a “conceptual layer” of core terms and concepts to which domain-specific terms and concepts are mapped, with reuse informed by a ‘translation’ exercise by which the nature of the mapping between them is indicated by their relative closeness to the core concept (or a higher-level concept, where necessary). Instead, pairwise mappings between domain-specific terms are explicitly captured, in terms of the similarities and differences between them. This point-to-point mapping of terms and concepts provides a degree of flexibility and addresses the complexity of the domain, though there are obvious challenges for automation.

The CCL as envisaged in [2] and [3] is a conflation of a model-based and a vocabulary-based approach. Models of standards requirements and project-level assurance assets will be produced, to ensure that project assets conform to the requirements of the relevant standards. The general concepts of safety assurance as reflected in the standards are captured in a conceptual metamodel to which the standards model and project-level asset models conform. These generic concepts will also provide the basic structure for a thesaurus-style vocabulary, in which the language used to describe and characterise assurance assets in the standards are arranged. A series of flexible mappings are defined between the terminology and concepts, in such a way as to indicate the constraints on reuse of assets across or within domains. The vocabulary is also used to characterise assets developed within OPENCOSS – for example, to populate the standard-specific argument structures to be developed in WP5.

The generic metamodel (GMM) for the CCL is presented in [3]. Models of the safety standards and the project-level assets are to be developed in the next phase of work, as is the CCL vocabulary.

2.2.1 CCL defines concepts required for assurance (GMM)

As outlined above, the CCL will record and define concepts required for assurance of safety-critical systems. These concepts are derived from the safety standards relevant to the OPENCOSS target domains. Definitions will be provided in the CCL vocabulary, and will reflect the usage of the standards and the partner companies (where the latter is distinct from that of the standards).

At present, only the conceptual metamodel for the CCL has been developed. This metamodel is presented in [1]. The metamodel defines very high-level concepts to capture “the sort of things required for assurance”, for example “reference assurance framework” (a safety standard, recommended practice of company standards, to which a project has to comply), “requirement”, “activity”. At this very high level of abstraction, the metamodel provides a layer at which the complex and subtle differences between the conceptualisation of safety within the target domains are resolved. It is meaningful, for example, to talk of “integrity level” in the abstract, without reference to the distinctions between SILs and DALs and all that follows from those distinctions in terms of processes and assurance objectives. The metamodel therefore provides a structure within which the detail of the standards can be defined and compared, and the very high-level commonalities can be used as a basis for comparison between the detailed concepts and for guidance on the boundaries within which reuse of assets is possible, or tailoring is required. This use of “superconcepts” to assist in reuse is discussed in detail in Section 3 of [2].

2.2.2 CCL to trace concepts (for reuse)

At the more detailed levels of standards models, project models and vocabulary, the CCL – structured according to the core metamodel described above – fleshes out the generic concepts with concrete descriptions and definitions of standard- and project-specific concerns and artefacts relevant to safety assurance. For example, the generic (CCL metamodel) concept of a “Reference Asset” is concretised first as a “IEC 61508 WCET report” of type Reference Asset, which defines the characteristics of a WCET report as defined in the requirements of IEC 61508. It may then be further concretised by an actual physical artefact: the WCET report which has been developed in accordance with these requirements. Concrete forms in the vocabulary – which derive from and reflect the usage of the standards and projects – are used to populate the generic assets developed by OPENCOSS.
The CCL therefore provides traceability between concepts at different levels of abstraction, and between concepts and their concrete forms. This traceability is used in two ways. Firstly, the traceability between the standards-level concepts and their concretisation in the project assets is used to demonstrate the compliance of project assets to the standard. In some cases, the project asset will be a straightforward instantiation of the standard-concept: i.e. the actual WCET report will meet all of the required characteristics of a 61508 WCET report as defined in the standard model. Elsewhere, the flexible mapping mechanism afforded by the CCL (see [3]) will allow for a clear record of the ways in which an asset does not quite comply with the standard. This is important, for example, in situations where a project is being developed according to more than one standard. Secondly, the traceability between concepts and their concrete forms is used to inform the guidance which the OPENCOSS Platform gives as to the implications – for compliance and assurance arguments – of reuse of a project-level assurance asset developed to one standard within another project which is being developed according to a different standard. Here, the standard-level models will provide clearly defined attributes of assets, which attributes can be used to provide the basis for the justification of the mappings. The quality of the mapping available provides the basis of detailed advice provided to the user.

### 2.2.3 CCL as a communication device within the OPENCOSS platform

The CCL vocabulary and models provide a transparent, flexible interface which can be used to provide for the exchange of data between tools in the OPENCOSS Platform. This subject is discussed at some length in [4].

### 2.3 Requirements of the CCL

D4.2 [8] describes the detailed requirements for the CCL. These requirements are represented in Table 2 in Appendix chapter 6. In the “Deployed” column the table shows for each requirement whether the current CCL implementation fulfils this requirement, and the column “Alternative” provides more details about an alternative solution as described in Section 3.3. In some cases the solution direction is indicated. This is discussed in more detail in Section 3.4.

### 2.4 Quality aspects of languages

The CCL can be regarded as a Domain Specific Modelling Language (DSML, see [1]). The CCL targets the safety and more general the assurance assessments’ domain specific language with a number of subdomains, mainly formed by the application domains and the jargon companies have created within those application domains. The CCL defines the concepts in the domain of safety critical systems and unifies them into one meta-model.

From the conceptual framework that the CCL offers, a generic metamodel (GMM) is defined. The GMM aims to provide a machine-processable framework defining the relevant concepts in assessment artefacts that should be understandable for the users in the safety assessment domain (see also requirement ). The GMM also aims to create an interchangeable language for the tools related to safety assessments and compliance-aware development processes.

When looking at the implementation of the conceptual domain knowledge, we have to take a number of non-functional requirements in mind. These are typical requirements coming from the safety domain, from the circumstances where the OPENCOSS platform will operate in, and the tools that need to be connected using the GMM.

2 Conceivably, these attributes, and the degree of exactitude of the match between them, could be used to provide a quantitative measure of the “similarity” between assets. This is by no means a straightforward task, however, since the relative importance of a particular attribute of the asset (i.e. a particular requirement of the standard) in the assurance/compliance requirements of the source context (for which the asset is developed) and the reuse context (into which it is reused) needs to be ascertained and weighted in the quantification. It may not be sufficient, for example, to say that “4 out of 5 attributes match”, if the 5th attribute is highly significant to the degree of assurance required. Work on this aspect of the mapping is currently being undertaken in T4.2.
For the tools it means that the framework should provide compatibility, especially interoperability, modifiability, in order to support an evolutionary approach. Preferably, the framework should be generated automatically, at least partly. For the editors it has the noble target to provide a usable environment with the focus on user error protection, which results in safer products and at the same time a maintainable one with the emphasis on reusability. In D4.4 it is assumed that the GMM focusses on assessment artefacts and excludes other concepts. The vocabulary covers the links between the GMM and the domain-specific concepts (originating from the application domain, company, project) The user is encouraged to use GMM concepts and discouraged to use other concepts.

The quality attributes of software are described in the ISO 25010:2011 standard [9]. The attributes are divided in the (a) quality in use, related to all human-computer interaction in a particular context, and the (b) product quality model, relating to the static properties of software. The quality in use model is composed of five characteristics, some of which are further subdivided into subcharacteristics. The product quality model composed of eight characteristics, which are also further subdivided into subcharacteristics. Next to the static properties of software, this model also refers to the dynamic properties of the computer system.

The characteristics defined by both models are relevant to all software products and computer systems. The characteristics and subcharacteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness.

We mainly focus on the product quality model, and into usability specifically.

- **Usability**: as a characteristic composed of a set of subcharacteristics.
  - (Functional) Appropriateness: degree to which the functions facilitate the accomplishment of specified tasks and objectives (For example, a user is only presented with the necessary steps to complete a task, excluding any unnecessary or exceptional steps. Functional appropriateness corresponds to suitability for the task in ISO 9241-110.)
  - Recognisability: degree to which users can recognize whether a product or system is appropriate for their needs cf. functional appropriateness (Note that appropriateness recognisability will depend on the ability to recognize the appropriateness of the product or (2) system’s functions from initial impressions of the product or system and/or any associated documentation. The information provided by the product or system can include demonstrations, tutorials, documentation or, for a web site, the information on the home page.
  - Learnability: degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
  - Operability: degree to which a product or system has attributes that make it easy to operate and control. (Operability corresponds to controllability, (operator) error tolerance and conformity with user expectations as defined in ISO 9241-110.)
  - User error protection: degree to which a system protects users against making errors.
  - User interface aesthetics: degree to which a user interface enables pleasing and satisfying interaction for the user (This refers to properties of the product or system that increase the pleasure and satisfaction of the user, such as the use of colour and the nature of the graphical design
  - Accessibility: degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Since we are talking about a framework or language, a number of other tools heavily depend on the exact functioning of this framework. If we look at the influence on the quality in use for maintenance tasks, ISO 25010 also describes a number of other characteristics that influence the quality in use for maintenance tasks. These are:

- **Maintainability**: the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintenance is subdivided in a five of sub-characteristics
  - Modularity: degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
  - Reusability: degree to which an asset can be used in more than one system or in building other assets (Adapted from IEEE 1517-2004.)
  - Analysability: degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for
deficiencies or causes of failures, or to identify parts to be modified (Note: Implementation can include providing mechanisms for the product or system to analyse its own faults and provide reports prior to a failure or other event.)

- Modifiability: degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality (Several notes here: 1. Implementation includes coding, designing, documenting and verifying changes. 2. Modularity and analysability can influence modifiability. 3. Modifiability is a combination of changeability and stability.)

- Testability: degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met (Note: Adapted from ISO/IEC/IEEE 24765.)

- Portability: degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another (Note: 1. Adapted from ISO/IEC/IEEE 24765. 2. Portability can be interpreted as either an inherent capability of the product or system to facilitate porting activities, or the quality in use experienced for the goal of porting the product or system.)

- Adaptability: degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments (Note 1: Adaptability includes the scalability of internal capacity (e.g. screen fields, tables, transaction volumes, report formats, etc.). Note 2: Adaptations include those carried out by specialized support staff, and those carried out by business or operational staff, or end users. Note 3: If the system is to be adapted by the end user, adaptability corresponds to suitability for individualization as defined in ISO 9241-110.)

- Installability: degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment. (Note: If the product or system is to be installed by an end user, installability can affect the resulting functional appropriateness and operability.)

- Replaceability: degree to which a product can replace another specified software product for the same purpose in the same environment. (Note 1: Replaceability of a new version of a software product is important to the user when upgrading. Note 2: Replaceability can include attributes of both installability and adaptability. The concept has been introduced as a subcharacteristic of its own because of its importance. Note 3: Replaceability will reduce lock-in risk: so that other software products can be used in place of the present one, for example by the use of standardized file formats.

- Compatibility (from ISO 25010) consists of:
  - Co-existence: degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product,
  - Interoperability: degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Furthermore, ISO 25010 states that usability, in the perspective of a quality of use, also involves:

- Effectiveness: accuracy and completeness with which users achieve specified goals,
- Efficiency: resources expended in relation to the accuracy and completeness with which users achieve goals, and
- Satisfaction: degree to which user needs are satisfied when a product or system is used in a specified context of use.

The definitions here help to choose the implementation details for the OPENCOSS’ CCL and its generic meta-models (GMM).

### 2.5 Configuration of the safety assurance project

On OPENCOSS we have put into practice a prototyping strategy. In order to align with the industry needs we have identified some mock-ups that will correspond to the use cases described in this document.
2.5.1 Mock-up, Use case: Capture information from standards

Following the use case described in Section 2.1, 1(a), Figure 4 depicts a mock-up of the user interface for this use case. In this mock-up we see the concepts for the ISO 26262, part 3, defined in a reference framework. These concepts include amongst others: the Item Definition Report, HARA (Hazard and Risk Analysis), Safety Goals, etc. The process and prerequisites for the different activities are also defined, as well as the relationship with the concepts.

![Figure 4: Capture Standard Data](image)

2.5.2 Mock-up, Use case: Create a safety assurance project

In order to create a safety assurance project (See use case in Section 2.1, 2(a)) we propose the use of a wizard. This wizard will require the user to fill in some basic information essential for the project identification. Following you will see the mock-up of that wizard in Figure 5.
Create new safety assurance project

Project Name: EPARK_001
Location: C:\AssuranceProject\Epark

ID: EPARK_001
Description: Assurance project for the Epark parking system

Created By:
Responsible:
Current Version: 1.0

Figure 5: Safety Assurance Project Wizard

As a result of the wizard the user will be able to consult or modify the data on window as displayed in Figure 6:

Figure 6: Consult or Modify Assurance Project Data
2.5.3 Mock-up, Use case: Define a safety assurance project baseline

Following the use case in Section 2.1, 2(b), we are required to fill more technical data for a specific project. The data required should comply with a part or the whole reference model or standard captured on a previous phase. In order to get the data the first time, the user can follow a wizard that will support him/her on this activity. See Figure 7.

![Create Project Baseline](image)

**Figure 7: Create Project Baseline**
After the wizard is ended, the user is able to modify and update the information, as depicted in Figure 8.

**Figure 8: Refine project baseline**

...
3 Common certification language (CCL)

3.1 Introduction

The implementation of the CCL should meet the required functionalities and non-functionalities or quality attributes as defined in the usage scenarios, use cases, requirements, and, additionally, in the ideal properties of languages as describe in Chapter 2.

For the implementation we have chosen to use the Eclipse EMF solution. This environment ensures a consistent and interoperable environment. The EMF project is a modelling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. With EMF it is possible to make a domain model explicit which helps to provide clear visibility of the model. EMF also provides change notification functionality to the model in case of model changes. EMF will generate interfaces and factory to create your objects; therefore it helps you to keep your application clean from the individual implementation classes.

As stated before, the implementation will cover two approaches: a straightforward approach and an alternative approach which includes model-transformation techniques. The first one implements the conceptual model as it stands (See D4.4) and includes the multi-layered approach as depicted in Figure 9. The model-transformation enriched approach uses the same framework, but adds more precision to the Level 0 by introducing specific metamodels transformed from the GMM. In the next prototype the approaches will be merged to represent the best solutions as indicated by the validation results from that time.
We refer to the specific editors in D4.6 (Reference Framework Editor), D5.4 (Intermediate implementation of tools for Argumentation/Compositional Certification), D6.5 (Intermediate implementation of the evidence management service infrastructure), the prototype user manual (OPENCOSS First Prototype User Manual), and the developer guide (OPENCOSS First Prototype Developers Guide) for the details on the implementation.

### 3.2 CCL concepts from D4.3/D4.4

The conceptual model of the CCL and its intermediate model are described in the deliverables D4.3 [2] and D4.4 [3]. These CCL conceptual models are used for the implementation of the CCL into the generic metamodel (GMM). First we will describe the overview of the GMM, and as an example, we will show here one of the main parts of the models that have been produced there.
3.2.1 The Generic Meta Model

The Generic Meta Model provides a large part of the CCL. For the implementation of the conceptual model into the generic meta-model (GMM), we first have researched on the prototype implementation of the main part of the GMM: the Reference Assurance Framework Meta-model (RAFM). In other work packages the other CCL metamodels have been implemented in an editor. We only implemented RAFM and ignore the fact that most of the users operate in a specific (application) domain and have a different interpretation of the concepts in the GMM. The more specific terms will be used in the Vocabulary Meta Model (VMM) which at this point is still in progress and does not provide enough detail for an implementation together with the RAFM.

The GMM is completely implemented in the Eclipse MOF environment. This ensures that the resulting models will be machine processible and human interpretable. Models can be translated to XML, for example, to ensure the possibility of machine processability. The Eclipse environment also ensures that the visual representation of the metamodels is understandable for humans. In this way the GMM conforms with the Readability and Usability Guidelines [12]. A critical note could be that the edges of the relations between classes in these metamodels (like in Figure 12 and Figure 13) should be round as is depicted in Figure 11.
3.2.2 Class descriptions

In Figure 11 the overview of all CCL metamodels into the GMM and their internal relationships are depicted. The Reframework is actually referring to the Reference Assurance Framework Metamodel. This model is detailed in Figure 12 and Figure 13. We refer to the D4.4 [3] for further details about these metamodels. The validation results for the implementation of the first prototypes have been added to this document and the metamodels in particular as well.

![CCL Metamodel Relationships](image)

Figure 11: CCL Metamodel Relationships

In Figure 11 the dashed arrow means: source Package X refers to target Package X.

For a brief overview of the changes in the first CCL, we refer to D4.4 [3], Appendix A.
Figure 12: Reference Assurance Framework Metamodel (Part 1: Core Model Elements)
Figure 13: Reference Assurance Framework Metamodel (Part 2: Inheritance Relationships)
3.3 Implementation alternative: extended meta-models for mappings between concepts

As indicated before, at this point the GMM has only a preliminary and abstract Vocabulary Meta Model (VMM). It is still under construction and does not provide the opportunity to support conceptual definitions, mappings between concepts that are specific to a certain metamodel. These specific application domain conceptual definitions, reflected in the mappings between concepts, typically are described in metamodels specific to a certain application domain.

As a partial interpretation of the Vocabulary Meta-Model (VMM) the conceptual mappings are implemented using extended model. Extending the CCL Metamodel into more domain specific metamodels provides the advantage to create a language that is closer to the world of experience of the safety engineer and will ease the effort to produce the reference assurance. Additionally, using the meta-models prevents users from making mistakes; meta-models should be defined once by the best expert(s) available. To achieve an implementation for the RAFM part as well as part of the VMM (conceptual mapping only) extended meta-models are applied; if new domain concepts are required for users’ purpose, VMM will be used. In other words, by using VMM, users can not only modify the existing RAFM concepts, but also introduce new domain concepts (in VMM) into RAFM.

The introduction of new domain concepts is typically done by (domain) expert users. They can add domain concepts expressed in a metamodel refine language (MMRL) to create the domain specific metamodel. The transformation preserves the link with the original GMM concepts and documents the change. The documentation helps to support reuse and provides explanatory help for the assessor to understand the relationships. Figure 14 explains this process in a graphical way. Also notice that the specific model editor needs to be recreated when new concepts are added. For the user it should be transparent that the actual editing environment changes, and also that the models created using a previous meta model, can be used without problems in the environment that includes the new concepts (forward compatibility). This notion is conceivable in the alternative setup.

Also note that at this point the GMM does not include specific safety concepts. This might be added to the GMM in a later stage. Whether or not these concepts will be there, the user does think in these concepts and from a usability perspective (connecting to the experience world of your user) it is better to include them either in the GMM or using meta model transformations in the specific metamodel.

![Diagram showing the process of creating an application domain-specific metamodel using model transformations.](image-url)

**Figure 14: Creating an (application domain) specific metamodel using model transformations**

The current VMM has been designed to enforce a separation between assurance artefacts and other safety domain concepts. This separation is purely conceptual and can also be made explicit in the user interface of the editor. This also makes it possible to use the more powerful meta-modelling techniques also for concepts in the vocabulary. The current meta model proposal cannot profit from the class specific attributes, and, therefore, could oversimplify the
modelling needs of safety engineers and assessors. This can lead to overgeneralisations, additional manual work, and less support for automatic consistency checks, which can all lead to user errors and a diminished product safety.

3.3.1 Alternative Mock-up, Use case: Capture information from standards

As a result of the alternative approach, the reference framework editor will look differently. Figure 15 depicts an example user screen of the editor. Notice that the user can work with the domain specific concepts. In this case it is the ISO 26262 that is being modelled, and instead of the general concept 'Artefact' the user can choose between WorkProduct or ExternalElement (Which in this case represents the External Data concept of the standard). The concepts have been added using the MMRL where WorkProduct is a relabeling of the Artefact, and ExternalElement, which is an extension of the concept Artefact. In the interface with other tools, as the back-end communication, the classes in this ISO26262 model are translated to the original GMM concepts to make sure that the data is communicated in a way that all other tools should understand them. Here, the links and documentation helps to decode the models to the commonly understood GMM concepts again.

![Alternative Mock-up, Use case: Capture information from standards](image1)

Figure 15: Modelling the ISO26262 Standard

3.3.2 Meta-model evolution to support reuse

As a prototype the CCL Metamodel provides a common approach to describe models in current certification process. Those models include standard models, company models and project models etc. For conceptual mapping, the meta-model evolution chain is proposed in Figure 16. It begins with the RAFM and VMM in the CCL Metamodel. For different domains, different terminology could be used. Because CCL Metamodel has the common concepts between those domains, it could contribute to the conceptual mapping cross-domain. By parity of reasoning, the mapping between standard conceptual meta-models, company meta-models and project meta-models can also be supported by this meta-model evolution chain. Besides, for concrete mapping, ComplianceMapMM will be used for mapping from one model to another model.
Figure 16: Meta Model Evolution
3.4 Evaluation

The CCL meets 7 of the 51 requirements directly as stated in D4.2 [8], the detailed requirements for the CCL. A large part, 37 of the 51 requirements have been realized in the D4.4 [3] or will be supported in the future. Generally the CCL uses concepts that are at least potentially cross domain, where possible, concepts have been generalized to more abstract terms in order to facilitate cross-domain reuse. An alternative way of the implementation is also evaluated and scored in the column ‘Alternative’. Here the different implementation techniques such as meta modelling and application-domain-specific concepts, model transformations or the semantic business vocabulary and business rules (SBVR) are mentioned, if they could provide a better solution.

In Appendix Chapter 6 the quality aspects are included as non-functional requirements to the list of requirements described in Section 2.3. We rated both the current application and the alternative as described in Section 3.3. Here we see that the CCL in general could meet most of the quality aspects relevant for a language, although 7 of the 20 non-functionals still needs to be validated.

A merge of the two solutions could clearly improve the system both in functional as non-functionals. A choice for one of the approaches should also include the feasibility of that solution. In the second prototype round more evaluations including feasibility will be held.
4 Conclusions

The implementation of the CCL (Common Certification Language) metamodels in the GMM (Generic Metamodel) supports specific functions within the domain of safety demonstration. At this stage it provides a necessary step in achieving the OPENCOSS goals; the CCL defines the concepts for safety assurance, it can trace concepts for reuse, and offers a communication device within the OPENCOSS platform. Furthermore it is ready for expanding its automatic support in order to achieve cost reduction in recurring work, increase in safety, and form a sound basis for innovations in the safety assurance process; the goals of OPENCOSS.

For the large part the GMM is implemented and applied in the first prototypes. The implementation is based on the Eclipse EMF and provides a consistent basis for the two main implementation goals of the CCL: (1) The CCL metamodel implementation should serve as an intermediary language between tools and as a domain specific language for users of the OPENCOSS platform; therefore it should provide a machine-processable data structure to interface tools to the OPENCOSS platform and (2) it should provide a usability basis for the core and back-end of the many editors that the OPENCOSS platform should provide. The CCL is also shows face validity on the mock-ups that have been created. Furthermore, in the near future the implementation will probably meet 37 of the 41 requirements, and 18 of 20 quality aspects. At this moment the CCL is also validated by its central role in the usage scenario’s

Since the OPENCOSS goals provide an ambitious challenge, two implementation approaches have been applied. The project has been exploring two alternative and complementary implementation strategies, which will be brought together to share the best of both worlds as the prototyping work evolves. This first implementation follows a mere implementation of the current stand of the conceptual work as provided in the D4.4. The second and alternative approach uses extensions on the metamodelling by model transformations, and also includes the use of the semantic business rules and business vocabulary (SBVR). For details we refer to the deliverables that describe the implementation of a number of editors: D4.6, D5.4, and D6.5.

Not all of the metamodels in the GMM have been developed in full. Since the implementation is not complete and not all intended functions have been demonstrated, therefore, more effort is needed to evaluate the implementation of the CCL metamodels. In the next iteration the CCL is further elaborated and validated in a second round of prototypes. In these prototypes the implementation approaches will be merged, based on the validation results and its feasibility to be implemented within the project time.

This deliverable is also the preparation of the reference (assurance) framework editor as described in D4.6. In reflection of the ongoing prototyping work, this deliverable will be updated iteratively.
5 References

[8] OPENCOSS Project, Detailed requirements for the common certification language (CCL), Deliverable 4.2.
6 Requirements and Quality Aspects on the CCL – a first evaluation

The requirements of the D4.2 [8] are described in Table 2. The table also includes the realisation of those requirements in the GMM in the first prototypes. This is indicated in the ‘Deployed’ column. The implementation for an alternative solution (as described in Section 3.3.) is indicated in the ‘Alternative’ column. If the implementation is already sufficiently described in the D4.4 [3], it is indicated likewise. The difference in deployment between the main implementation and the alternative, is briefly indicated by using the following key-words:

- Only supported by initial VMM: Application domain specific solutions are supported, but not aimed at certain models, which means that definitions can be applied to the wrong model (for example that definitions of the reference model can be applied wrongfully to the project model and that reuse of definitions is not supported)
- Cross domain: support for cross domain usage, but no support for application domain specific terminology, excluding the the preliminary vocabulary labelling and mapping which is established manually.
- Cross- and domain specific: support for cross domain usage and application domain specific concepts. Note that the relationship with the original GMM concepts is preserved and documented in the model transformation annotation.
- Manually: support is provided by the preliminary vocabulary labelling and mapping which is established manually.
- Machine supported: relationships with the original GMM concepts is preserved and documented in the model transformation annotation. This is only about conceptual vocabulary mappings.
- Future extension possible: in the current implementation the feature is not present, but the technology has a clear potential to meet the requirement in future.
- Future support by SBVR: in the current implementation the feature is not present, but by using the Semantic Business Rules and Vocabulary Rules it has a potential to meet the requirement in future.
- Needs refinement: the designated metamodel does not support the requirement sufficiently, yet, but there is a clear potential to meet the requirement in future.

Table 2: Requirements and Quality aspects on the CCL (from D4.2 [8])

<table>
<thead>
<tr>
<th>Feature</th>
<th>Req. ID</th>
<th>Requirement Name</th>
<th>Deployed</th>
<th>Alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface requirements</td>
<td>INT_01</td>
<td>The conceptual model of the CCL shall be readable by a human safety engineer with adequate training in the selected modelling approach and the rationale of the CCL.</td>
<td>Y (D4.4)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>INT_02</td>
<td>The conceptual model of the CCL shall be machine-readable</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>INT_03</td>
<td>The conceptual model of the CCL shall be presented using an implementation-independent representation technique</td>
<td>Y (D4.4)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>INT_04</td>
<td>The feedback to be provided by the CCL shall be presentable (either directly, or by machine intervention) in a manner which is readily intelligible to a human safety engineer with adequate training in the selected modelling approach and the rationale of the CCL.</td>
<td>Future extension possible</td>
<td>Future extension possible</td>
</tr>
<tr>
<td></td>
<td>INT_05</td>
<td>The feedback to be provided by the CCL should include a qualitative indication of the possibility of reuse of a given artefact in a given context.</td>
<td>Future extension possible</td>
<td>Future extension possible</td>
</tr>
<tr>
<td></td>
<td>INT_06</td>
<td>The feedback to be provided by the CCL should include quantitative feedback on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT_07</td>
<td>The feedback to be provided by the CCL should include guidance on the limitations on reuse of an artefact in a given context.</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>INT_08</td>
<td>The feedback to be provided by the CCL should include a clear statement of rationale for the indications of ‘reusability’ given.</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### Capture of consistent assurance concepts

| 01_01 | Capture of consistent generic assurance concepts | Y (D4.4) | Y |
| 01_02 | Capture of consistent domain-specific assurance concepts | Only supported by initial VMM | Cross- and domain spec. |
| 01_03 | Capture of relationships between assurance concepts | In progress | Cross- and domain spec. |

### Analysis of consistent assurance concepts

| 02_01 | Analysis of matches between assurance concepts from different domains | Manually, in progress | Machine supported |
| 02_02 | The CCL shall allow the establishment of relationships between standard/company/domain specific terms and the conceptual synthesis of terms provided by the CCL. | Manually, in progress | Machine supported |
| 02_03 | Analysis of the match between concepts from distinct domains or standards | Manually, in progress | Partly machine supported |
| 02_04 | Indication of the nature of the match between concepts from distinct domains or standards | Manually, in progress | Partly machine supported |
| 02_05 | Automated provision of guidance as to the aspects of two concepts from different domains or standards not matching | - | Future extension possible |

### Representation of safety-related concepts

<p>| 03_01 | Description of requirement coverage | Future extension possible | Future extension possible |
| 03_02 | Capture of specific safety normative constraints/objectives | Manually | Possible Machine Support |
| 03_03 | Comparison of safety normative constraints/objectives | Manually | Possible Machine Support |
| 03_04 | Description of safety assurance processes | GSN meta-model needs refinement | GSN meta-model needs refinement |
| 03_05 | Description of safety analysis techniques | Manually | Manually |
| 03_06 | Support of the different types of safety arguments | Needs refinement | Needs refinement |
| 03_07 | Description of safety claims, assumptions, context and evidence | Future support by SBVR | Future support by SBVR |
| 03_08 | Description of safety requirements | Cross domain in D4.4 | Cross- and domain spec. |
| 03_09 | Hazard expression | (not specifically in) | Cross- and domain spec. |</p>
<table>
<thead>
<tr>
<th>Management of safety-argument contracts</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>03_10</td>
<td>Determination of the Integrity Level in the different domains</td>
<td>Cross domain in D4.4</td>
<td>Cross- and domain spec.</td>
</tr>
<tr>
<td>03_11</td>
<td>Description of the concepts from hazard-directed arguments</td>
<td>Needs refinement</td>
<td>Needs refinement</td>
</tr>
<tr>
<td>03_12</td>
<td>Guarantee the completeness of the argumentation for certification</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>03_13</td>
<td>Confidence in a safety argument</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Management of safety-argument contracts</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>04_01</td>
<td>Contracts representation</td>
<td>Needs validation</td>
<td>-</td>
</tr>
<tr>
<td>04_02</td>
<td>Characterization of safety argument modules</td>
<td>Y (D4.4)</td>
<td>Y (D4.4)</td>
</tr>
<tr>
<td>04_03</td>
<td>Modular safety case concepts</td>
<td>Needs refinement</td>
<td>Supported by layers</td>
</tr>
<tr>
<td>04_04</td>
<td>Characterisation of safety case module interfaces</td>
<td>Future support by SBVR</td>
<td>Future support by SBVR</td>
</tr>
<tr>
<td>04_05</td>
<td>Characterisation of safety case assumptions</td>
<td>Future support by SBVR</td>
<td>Future support by SBVR</td>
</tr>
<tr>
<td>04_06</td>
<td>Characterisation of safety case context</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>04_07</td>
<td>Characterisation of other relevant aspects of safety case modules interfaces</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>04_08</td>
<td>Means for the specification of consistency rules and warnings regarding the characterisation of safety case modules interfaces</td>
<td>Future support by SBVR</td>
<td>Future support by SBVR</td>
</tr>
<tr>
<td>04_09</td>
<td>Provision of a library of rules and warnings regarding the characterisation of safety case modules interfaces</td>
<td>Future support by SBVR / Templates</td>
<td>Future support by SBVR / Templates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evidence management</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>05_01</td>
<td>Evidence characterization</td>
<td>Y (in D4.4)</td>
<td>Y (in D4.4)</td>
</tr>
<tr>
<td>05_02</td>
<td>Evidence reuse</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>05_03</td>
<td>Support identifying evidence</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>05_04</td>
<td>Informed reuse evidence</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>05_05</td>
<td>Evidence assessment</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compliance Management</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>06_01</td>
<td>Specification of Standards/Regulations</td>
<td>Y (D4.4)</td>
<td>Y, Partial vocabulary implementation in meta models</td>
</tr>
<tr>
<td>06_02</td>
<td>Comparison of Standards/Regulations</td>
<td>Manually</td>
<td>Partial Machine support</td>
</tr>
<tr>
<td>06_03</td>
<td>Level of compliance</td>
<td>Manually</td>
<td>Partial Machine support</td>
</tr>
<tr>
<td>06_04</td>
<td>Compliance Arguments</td>
<td>Manually in D4.4</td>
<td>Manually in D4.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance requirement</th>
<th>Req. ID</th>
<th>Requirement name</th>
<th>Deployed</th>
<th>Alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>01_02_01</td>
<td>Level of abstraction of domain-specific assurance concepts</td>
<td>Y (D4.4)</td>
<td>Domain specific solution using model</td>
<td></td>
</tr>
<tr>
<td>Design constraint</td>
<td>Constraint ID</td>
<td>Constraint name</td>
<td>Deployed</td>
<td>Alternative</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>DES_CONTR_01</td>
<td>Meta-modelling language to specify CCL</td>
<td>In progress</td>
<td>In progress</td>
</tr>
<tr>
<td></td>
<td>DES_CONTR_02</td>
<td>Reuse of existing OMG language concepts</td>
<td>Manually documented</td>
<td>Documentation and update through model-transformations</td>
</tr>
<tr>
<td></td>
<td>DES_CONTR_03</td>
<td>Planning for future re-use of CCL</td>
<td>Cross domain</td>
<td>Cross- and domain spec.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-functional</th>
<th>Characteristic</th>
<th>Subcharacteristic</th>
<th>Deployed</th>
<th>Alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 25010</td>
<td>Usability</td>
<td>(Functional) Appropriateness</td>
<td>Basic step</td>
<td>Support OPENCOSS goals</td>
</tr>
<tr>
<td></td>
<td>Recognisability</td>
<td>Only supported by initial VMM</td>
<td>Concepts can be domain specific</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learnability</td>
<td>Needs validation</td>
<td>Needs Validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operability</td>
<td>Needs validation</td>
<td>Needs Validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User error protection</td>
<td>Manual work is error prone</td>
<td>Higher level of automatic support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User interface aesthetics</td>
<td>Based on Eclipse</td>
<td>Based on Eclipse</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accessibility</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Modularity</td>
<td>Modular</td>
<td>Modular</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reusability</td>
<td>Due to manual mappings possibly restricted</td>
<td>Reusability through model transformations possible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analysability</td>
<td>Needs validation</td>
<td>Needs Validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiability</td>
<td>Static GMM could be infeasible</td>
<td>Changes in GMM could be feasible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Testability</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>Adaptableability</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Installability</td>
<td>Good</td>
<td>Poor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Replaceability</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Compatibility</td>
<td>Co-existence</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interoperability</td>
<td>Good</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>usability (quality of use)</td>
<td>Effectiveness</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Efficiency</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Satisfaction</td>
<td>Needs validation</td>
<td>Needs validation</td>
<td></td>
</tr>
</tbody>
</table>
7 XML representation of GMM’s reference assurance framework

Below is an XML-representation of the Standard Meta Model of the Generic Meta Model in Eclipse’s ecore. This is an example of a textual representation of meta models.

```xml
<?xml version="1.0" encoding="UTF-8"?>
   xmlns:emf="http://www.eclipse.org/emf/2002/Ecore" name="gmm" nsURI="http://gmm/1.0"
   nsPrefix="gmm">
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Requirement" eSuperTypes="#//NameElement">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="assignedTo" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#/EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="objective" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#/EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="sourceOf" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#/EString"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="CriticalityApplicability">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="hasCriticality" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#/EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="hasApplicability" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#/EString"/>
</eClassifiers>
</ecore:EPackage>
```
<eStructuralFeatures xsi:type="ecore:EAttribute" name="removalEffect" eType="#/ChangeEffectType"/>
<eStructuralFeatures xsi:type="ecore:EReference" name="target" lowerBound="1" eType="#/ReferenceArtefact"/>
<eStructuralFeatures xsi:type="ecore:EReference" name="recordedIn" upperBound="-1" eType="#/ReferenceArtefact"/>
</eClassifiers>
</ecore:EPackage>