Collaborative Large-scale Integrating Project

Open Platform for EvolutioNary Certification Of Safety-critical Systems

Implementation of the CCL Editor
D4.6

Work Package:	WP4: Common Certification Language
Dissemination level:	Public
Status:	Final
Date:	16 December 2013
Responsible partner:	Martijn Klabbers (TU/e)
Contact information:	M.D.Klabbers@tue.nl

Contributors

Names	Organisation
Alejandra Ruiz, Idoya del Río, Ángel López, Huáscar Espinoza	TECNALIA
Martijn Klabbers, Yaping Luo(Luna)	Eindhoven University of Technology

Document History

Version	Date	Remarks
V0.1	2013-12-05	Table of contents, draft outline, first content
V0.2	2013-12-15	Reviewed first version
V1.0	2013-12-16	Ready for EC review

TABLE OF CONTENTS

List of Figures

List of Tables

No table of figures entries found.

Abbreviations

API	Application programming interface
CCL	Common Certification Language
DSL	Domain Specific Language
DX.Y	OPENCOSS Deliverable X.Y
EMF	Eclipse Modelling Framework
GUI	Graphical User Interface
GSN	Goal Structure Notation
SBVR	Semantic Business Vocabulary and Business Rules
TX.Y	OPENCOSS Task X.Y
WPX	OPENCOSS Work Package X

1 Executive Summary

This document is a summary umbrella document of implementation and investigation work done in T5.3 “Intermediate tool support for compositional certification”.

In particular the following core items constitute D4.6 achievements, referenced by this document:

- Installable OPENCOSS Platform tools 1st prototype
- User Manuals and installation Instruction
- Source code description

In parallel to the implementation work, further research and investigation on potential implementation techniques, frameworks and tooling strategies but also on theoretical groundwork have been performed by partners participating in T5.3 task. The results are briefly outlined in this document and they will be used in the 2nd prototype phase if applicable.

2 Implementation of OPENCOSS platform 1st prototype tools

The main goal of T4.3 at this stage is the provision of tools and OPENCOSS platform services to support prescriptive knowledge management. Following the general OPENCOSS strategy of an incremental approach for research and development, the first prototype phase concentrated on the general validation and implementation of reference frameworks, like standards and guidelines, and on the quick provision of end user tools to support other activities in the project as the analysis of the industrial case studies and the analysis of further tool requirements and usage scenarios. The integration into the OPENCOSS platform, the alignment with other parts of OPENCOSS and the stepwise extension towards further framework concepts will be subject of the prototype phases 2 and 3.

To achieve these goals, the Eclipse RCP platform – including supplementary Eclipse packages as EMF, GMF, Epsilon, Eugenia, and Xtext – was chosen as the technology foundation for the implementation work. It allows rapid prototyping but still supports extensibility in later phases due to high integration capabilities.

This deliverable is mainly concerned with the management of standards and guidelines. This part of the OPENCOSS platform manages the prescriptive rules and guidelines where safety critical systems need to adhere to, they form the basis for the assurance assessments. The functionality implemented in the first prototypes is described in the next sub-section.

As stated in the D4.5, the project is exploring two alternatives and complementary implementation strategies, which will be brought together to share the best of both worlds as the prototyping work evolves. The implementation approaches can be characterized as: (1) a straightforward approach and (2) an alternative approach which includes model-transformation techniques. The first one implements the conceptual model as it stands (See D4.4) and includes the modelling of the reference assurance framework. The alternative model-transformation-enriched approach uses the same framework, but adds more precision to the metamodels by introducing specific domain specific concepts that will be added to the GMM to form the Reference Assurance Framework Metamodel (RAFMM). In the next prototype the approaches will be merged to represent the best solutions, in terms of desired functionality as also in project feasibility.

2.1 Scope and Implemented Functionality

The scope for the first prototype is the provision of modelling tools for modular argumentation structures and assurance patterns as well as supplementary functions as preliminary pattern instantiation, context based user guidance, vocabulary support and contract definition. The major scope is highlighted with a red circle on the next figure showing the general functional overview of the OPENCOSS platform.

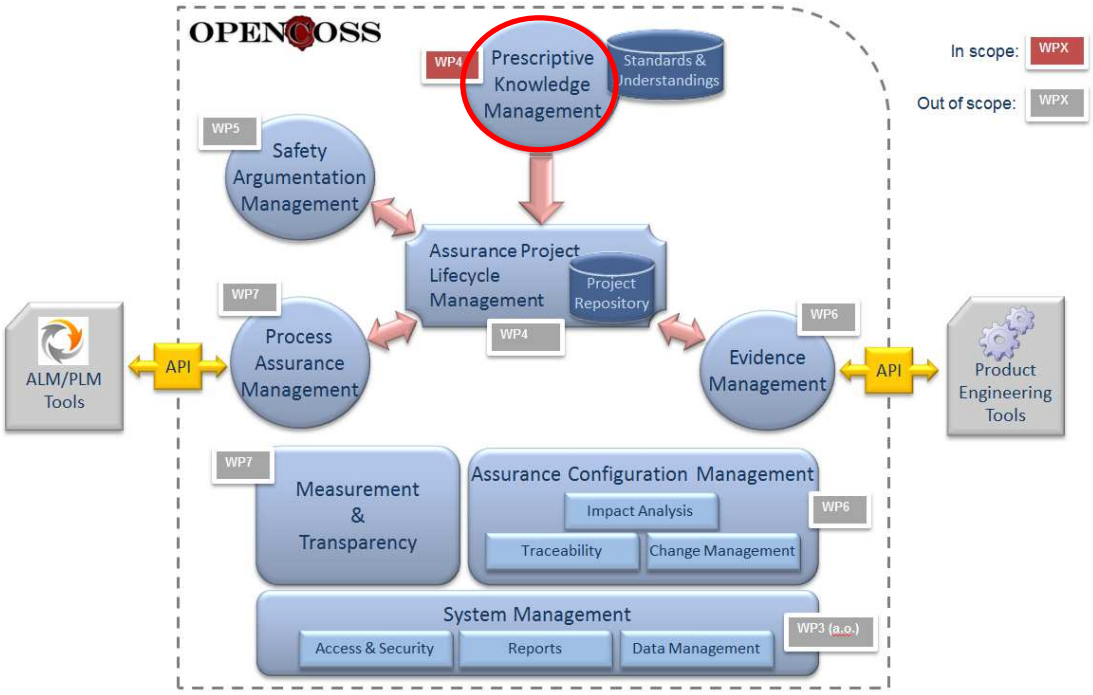

Figure 1. Functional decomposition for the OPENCOSS platform

From the requirements point of view this phase focuses on a set of high level requirements as defined in the prescriptive knowledge management part in D4.2. In Appendix Chapter 5 the original requirements overview is displayed. Each requirement together with the implementation done so far and realizing the requirement is shortly outlined in the following.

- Capture consistent domain-specific **definitions of core assurance concepts** relevant to safety assurance.
  Prescriptive knowledge management is developed using the reference framework model which has been develop within OPENCOSS. For the first prototype a graphical editor has been developed which implements the CCL reference framework mode.

- Capture the essential **relationships between assurance concepts**, in terms of required relationships and dependencies between them.
  The tool palette shown above indicates the different reference framework classes supported graphically in the first prototype and the relationships between them.

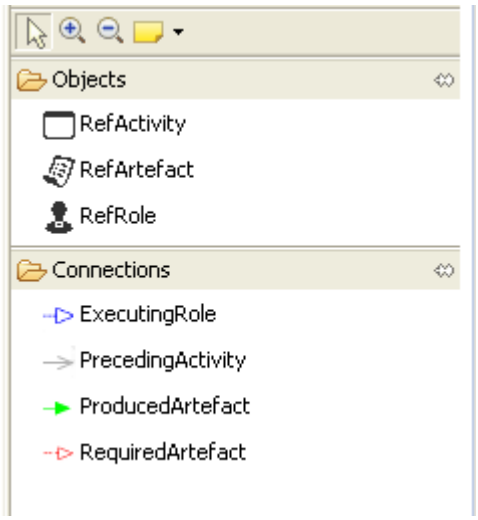

Figure 2. Palette with supported reference framework classes and relationships in the reference framework editor

- Capture consistent generic definitions of core assurance concepts relevant to **safety assurance from across the target domains**.
  To cover this requirement the mapping model has been created in other to store all equivalence mapping between two reference frameworks of different domains. This mapping is done using the reference framework editor.

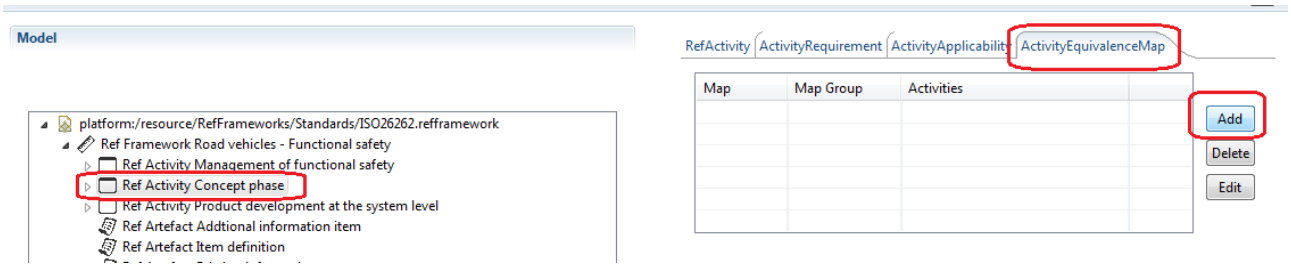

Figure 3. Activity Equivalence Map in the reference framework editor

2.2 Installation Guides & User Manuals

Here the installation guide and user manual of the straightforward implementation are discussed. Also the alternative implementation follows these guides. At this moment there is no installer or manual of the D4.6 alternative implementation, currently these are under construction. The difference between the implementation approaches is described in Appendix Chapter 6. This chapter focusses on the relevant differences in the user interface and the metamodel refine language.

The steps necessary to install the first prototype are described in the document “OPENCOSS first developer guide” (Date 22/10/2013 - V0.5). This document is hosted, with the source code of the prototype, in the remote location https://svn.win.tue.nl/repos/opencoss-code/tags/prototype/0.5, under the doc branch.

This document is a developer guide of the first OPENCOSS tool prototype implementation. The developers can find the source code installing instructions, step by step, in order to set up their development environment and their workspaces to implement new functionalities to the OPENCOSS Prototype.
The user manual for Prescriptive Knowledge Management Module is detailed in the document “OPENCOSS first prototype user manual” (Date 23/10/2013 - V0.6; Section 4). This document is hosted, with the source

code of the first prototype, in the remote location https://svn.win.tue.nl/repos/opencoss-code/tags/prototype/0.5, under the doc branch.

In summary, the document above is a user manual of the first OPENCOSS tool prototype implementation. The users can find the installing instructions, the tool environment description, and the functionalities starting for the creation of Reference Frameworks (models representing Standards, Regulations, or Company-specific Processes), Assurance Projects and the associated Baseline (subset of Reference Framework to be applied in a specific assurance project), Evidence models (Artefacts), Process models (Activities), Compliance Maps (so far, compliance maps from Reference Artefacts to Artefacts), and Argumentation models.

2.3 Source Code

The source code of the first prototype following the straightforward approach can be found in the source code Subversion at https://svn.win.tue.nl/repos/opencoss-code. The prototype version considered in this document is 0.5 and can be found in a tagged baseline version under "tags/prototype/0.5". The trunk is reserved for future development.

For the alternative there is a different code base. It is located at:
https://svn.win.tue.nl/repos/opencoss/WP1\D1.4_in_progress\Automotive\Source code\phoenix.zip

After installing the first prototype and following the steps described in the document "OPENCOSS first developer guide" V0.5, all the source code can be found under the plugins branch.

Once all the plugins are installed, these are the necessary ones for Prescriptive Knowledge Management:

- org.opencoss.pkm.reffframework
  In this plugin, the reference framework metamodel is defined and stored, and the Java implementation classes for this model are generated.
- org.opencoss.pkm.reffframework.diagram
  This plugin is the diagram editor itself. It manages diagrams and includes a canvas to draw on, a palette with creation tools and default selecting and zooming capabilities, a property view and an outline view.
- org.opencoss.pkm.reffframework.edit
  The edit plugin includes adapters that provide a structured view and perform command-based edition of the model objects.
- org.opencoss.pkm.reffframework.editor
  This plugin provides the user interface to view instances of the model using several common viewers and to add, remove, cut copy and paste model objects, or to modify the objects in a standard property sheet.
- org.opencoss.pkm.reffframework.utils
  This is an additional plugin that permits the usage of images instead of shapes to draw Artefacts and Roles in the reference frameworks diagram.
- org.opencoss.infra.mappings
  In this plugin, the mapping metamodel is defined and stored, and the Java implementation classes for this model are generated.
- org.opencoss.infra.mappings.edit
  The edit plugin includes adapters that provide a structured view and perform command-based edition of the model objects.
- org.opencoss.infra.mappings.editor

This plugin provides the user interface to view instances of the model using several common viewers and to add, remove, cut, copy and paste model objects, or to modify the objects in a standard property sheet.

Figure 4 shows all the plugins described above.

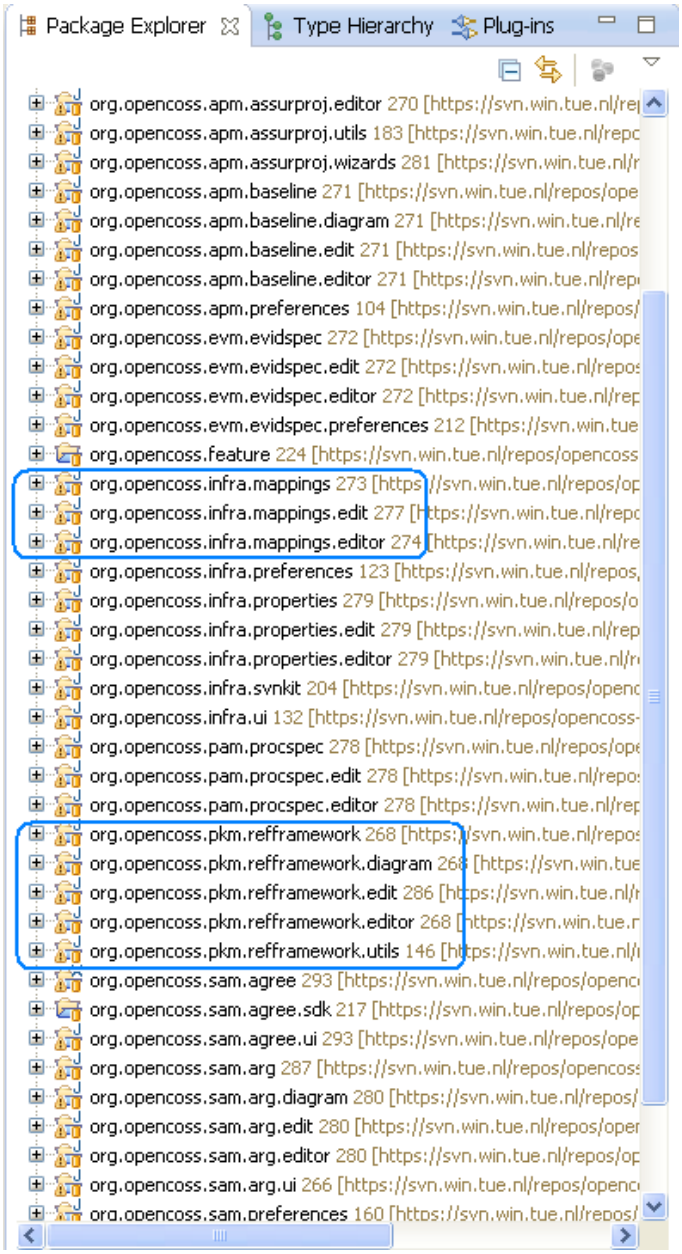

Figure 4. Prescriptive Knowledge Management management plugins

3 Research and Investigation

Additionally to the core implementation work described in Section 2, architecture and technology research that is related with implementation has been performed. The main results and conclusions are presented in this section.

3.1 Vocabulary Support

The support for vocabularies included only an abstract and concise meta model. During the implementation of the reference assurance framework, ideas have been created to support the vocabulary better. Mappings, ontologies, meta-models, and SBVR have been opted as solutions to support the correct labeling of concepts in the various application domains, preserving the link to the classes of the original GMM. For a small part and aspect of the vocabulary, the mapping between concepts, we have created a solution in the alternative implementation approach. (see 3.3)

3.2 Integration with Common Infrastructure

While the first prototype is based on the Eclipse RCP framework for rich clients – and mainly working on local files – the next iteration of the prototype phase envisages to also support a common central data infrastructure to share data with other tools and platform services. Main task of the development will be the investigation in the available persistence techniques and frameworks available for Eclipse and EMF based tools as well as the possibilities to still support both, a central database storage as well as a local file based storage in parallel to support the OPENCOSS community in choosing the appropriate technique when developing additional tools atop the OPENCOSS platform and frameworks. The investigations in the technology as well as the impact on the current prototype codebase and the required changes have been already started and will be continued in the next phase.

3.3 Model transformation for documentation and reuse

As a possible implementation to the vocabulary, and specifically the vocabulary part where the mappings between concepts is made, meta-models have been tried in the alternative implementation approach. The introduction of new application domain concepts is typically done by (domain) expert users. They can add domain concepts expressed in a metamodel refine language (MMRL) to create the application-domain-specific metamodel. The transformation is preserved as 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 1 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).

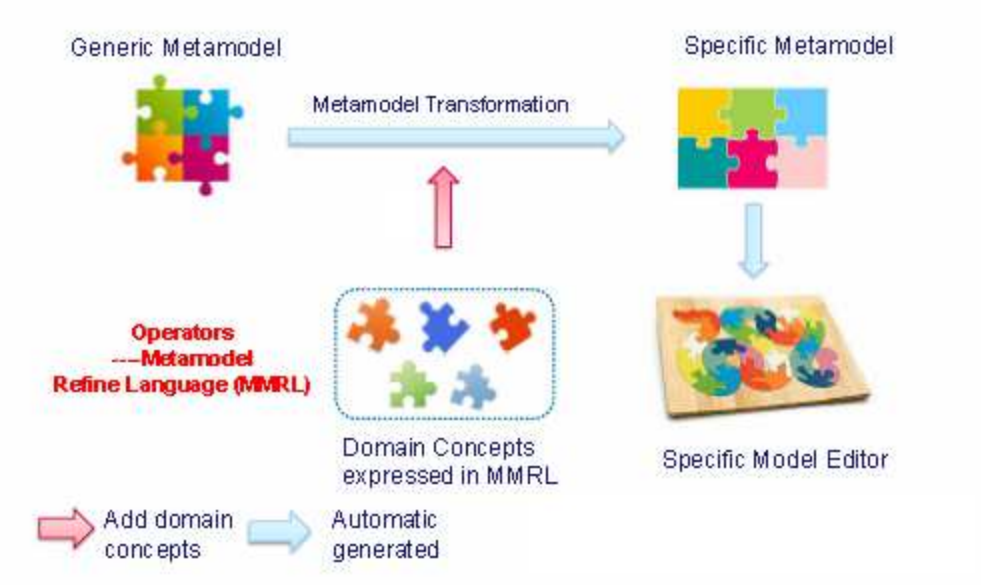

Figure 1: Creating an (application domain) specific metamodel using model transformations

4 Outlook

There are four main topics that will be tackled in the 2nd development phase.

1. Meta Model extensions by using model transformations is still a topic that needs further research. The validation in the automotive domain has been fruitful, but too little to provide substantial evidence in the sense that it could offer an integrated solution for the vocabulary in the broader sense. Another topic strongly related to the meta-modelling solution is the possibility of reuse. This has not been tackled in the first prototype but the technique is very promising.

2. The support for vocabularies is planned to be extended in the next prototype phase and to go far beyond the definition and usage of simple terms as they are currently modeled in the CCL and used in the tools. Different techniques and standards such as SBVR and existing tools to support the definition as well as to support the usage of “electronic” versions of vocabularies in OPENCOSS tools will be further researched and analyzed. Based on rich vocabularies the semantic analysis and

validation of claims is supposed to be much more capable. The investigation in existing supporting tools and related projects has been already started in the current phase but no significant results in terms of direct vocabulary tooling that can be incorporated into the platform or that have been developed are available at this stage.

Basic vocabulary support on text properties of classes is ported to SBVR vocabularies. The current CCL meta-model for vocabulary is superseded by SBVR based vocabularies. Existing tools will be ported to SBVR if applicable. New functions atop the SBVR vocabulary will be designed and if feasible developed. Existing SBVR based functions as described above will be merged if applicable.

3. The integration with other parts of the OPENCOSS infrastructure (for example the common data infrastructure) is continued. That includes further research on best technologies of choice (e.g. TENEO based DB storage) as well as the adaptation of the current code base to any technology decision that is achieved across all work packages.

4. Integration with existing safety analysis or evidence tools that are on the market (as medini analyze for example) is envisaged to achieve a show case that is applicable in the real field. Demo use cases and scenarios will be developed that underpin the integration potentials and possibilities of the OPENCOSS platform towards tool vendors.

5 Appendix A

This appendix presents the detailed requirements specified in D4.2 that have been deployed in the first prototype of the Reference Framework Editor and Mappings Editor. These requirements are listed in Table 1. The column labelled as "Deployed" indicates the state of the deployment including a small comment, and the column labelled as “Alternative” is the state of deployment in the alternative solution and also a small explanation of the state of deployment. The requirements in bold and italic are the requirements directly related to the reference assurance framework editor. All quality aspects are relevant to this editor as well. Detailed explanation about this table can be found in the D4.5, Common Certification Language: Implementation.

Table 1: Requirements and quality aspects for the Reference Assurance Framework Editor

Feature	Req. ID	Requirement Name	Deployed	Alternative
Interface requirements	***INT_01***	***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.***	***Y (D4.4)***	***Y***
	INT_02	***The conceptual model of the CCL shall be machine-readable***	***Y***	***Y***
	INT_03	***The conceptual model of the CCL shall be presented using an implementation-independent representation technique***	***Y (D4.4)***	***Y***
	INT_04	***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.***	***Future extension possible***	***Future extension possible***
	INT_05	***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.***	***Future extension possible***	***Future extension possible***
	INT_06	***The feedback to be provided by the CCL should include quantitative feedback on the extent to which reuse of an artefact is possible in a given context.***	***-***	***-***
	INT_07	***The feedback to be provided by the CCL should include guidance on the limitations on reuse of an***	***-***	***-***

		artefact in a given context.		
	INT_08	***The feedback to be provided by the CCL should include a clear statement of rationale for the indications of 'reusability' given.***	***-***	***-***
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	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	Future support by	Future support by SBVR

		evidence	SBVR	
	03_08	Description of safety requirements	Cross domain in D4.4	Cross- and domain spec.
	03_09	Hazard expression	(not specifically in D4.4)	Cross- and domain spec.
	03_10	Determination of the Integrity Level in the different domains	Cross domain in D4.4	Cross- and domain spec.
	03_11	Description of the concepts from hazard-directed arguments	Needs refinement	Needs refinement
	03_12	Guarantee the completeness of the argumentation for certification	-	-
	03_13	Confidence in a safety argument	-	-
Management of safety-argument contracts	04_01	Contracts representation	Needs validation	-
	04_02	Characterization of safety argument modules	Y (D4.4)	Y (D4.4)
	04_03	Modular safety case concepts	Needs refinement	Supported by layers
	04_04	Characterisation of safety case module interfaces	Future support by SBVR	Future support by SBVR
	04_05	Characterisation of safety case assumptions	Future support by SBVR	Future support by SBVR
	04_06	Characterisation of safety case context	-	-
	04_07	Characterisation of other relevant aspects of safety case modules interfaces	-	-
	04_08	Means for the specification of consistency rules and warnings regarding the characterisation of safety case modules interfaces	Future support by SBVR	Future support by SBVR
	04_09	Provision of a library of rules and warnings regarding the characterisation of safety case modules interfaces	Future support by SBVR / Templates	Future support by SBVR / Templates
Evidence management	05_01	Evidence characterization	Y (in D4.4)	Y (in D4.4)
	05_02	Evidence reuse	-	-
	05_03	Support identifying evidence	-	-
	05_04	Informed reuse evidence	-	-
	05_05	Evidence assessment	-	-
Compliance	***06_01***	***Specification of***	***Y (D4.4)***	***Y, Partial***

Management		***Standards/Regulations***		***vocabulary implementation in meta models***
	06_02	***Comparison of Standards/Regulations***	***Manually***	***Partial Machine support***
	06_03	Level of compliance	Manually	Partial Machine support
	06_04	Compliance Arguments	Manually in D4.4	Manually in D4.4
Performance requirement	Req. ID	Requirement name	**Deployed**	**Alternative**
	01_02_01	Level of abstraction of domain-specific assurance concepts	Y (D4.4)	Domain specific solution using model transformations
Design constraint	Constraint ID	Constraint name	**Deployed**	**Alternative**
	DES_CONTR_01	Meta-modelling language to specify CCL	In progress	In progress
	DES_CONTR_02	Reuse of existing OMG language concepts	Manually documented	Documentation and update through model-transformations
	DES_CONTR_03	Planning for future re-use of CCL	Cross domain	Cross- and domain spec.
Non-functional	Characteristic	Subcharacteristic	**Deployed**	**Alternative**
ISO 25010	***Usability***	***(Functional) Appropriateness***	***Basic step***	***Support OPENCOSS goals***
		Recognisability	***Only supported by initial VMM***	***Concepts can be domain specific***
		Learnability	***Needs validation***	***Needs Validation***
		Operability	***Needs validation***	***Needs Validation***
		User error protection	***Manual work is error prone***	***Higher level of automatic support***
		User interface aesthetics	***Based on Eclipse***	***Based on Eclipse***
		Accessibility	***Needs validation***	***Needs validation***
	Maintainability	***Modularity***	***Modular***	***Modular***
		Reusability	***Due to manual mappings possibly***	***Reusability through model transformations possible***

			restricted	
		Analysability	Needs validation	Needs Validation
		Modifiability	Static GMM could be infeasible	Changes in GMM could be feasible
		Testability	-	-
	Portability	Adaptability	Needs validation	Needs validation
		Installability	Good	Poor
		Replaceability	-	-
	Compatibility	Co-existence	Needs validation	Needs validation
		Interoperability	Good	Good
	usability (quality of use)	Effectiveness	Needs validation	Needs validation
		Efficiency	Needs validation	Needs validation
		Satisfaction	Needs validation	Needs validation

6 Differences between straightforward and alternative implementation

In the first straightforward prototype, the manual describes the main page of the editor as depicted in Figure 2. An equivalent picture for the alternative implementation of the reference assurance framework is depicted in Figure 4. The same elements are depicted in a slightly different position.

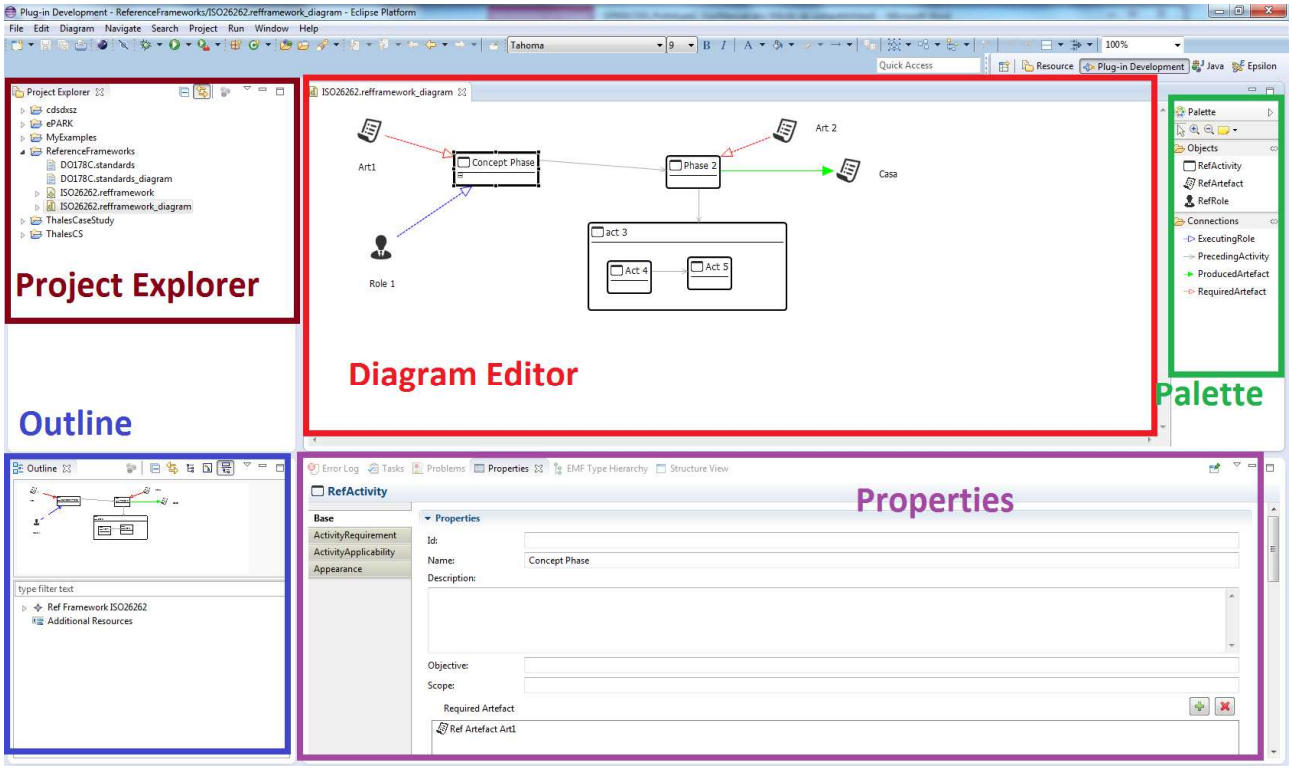

Figure 2: Reference Assurance Framework overview

The real difference is in the Palette. In Figure 3 the difference is indicated. Where the objects of the GMM are directly represented in the user interface of the reference assurance framework editor in the straightforward method, the other palette is showing specifically ISO 26262 concepts. Instead of displaying 'RefActivity' the alternative way indicates 'Activity'. 'RefArtefact' has a more complex relationship, since the ISO26262 has two distinct representatives: Work Products (internal project artefact) and External Data (external project artefact – here called ExternalElement). The relationship between these concepts is indicated in the metamodel refinement language (MMRL).

The specific example MMRL for this case is represented in Table 2. Here we see that the GMM concepts are translated into an ISO26262 tailored example: Reffframework becomes ISO26262Framework, etc. The MMRL has a number of commands, like 'rename'. This command does not more than providing a new label to an existing GMM concept. Other commands are: add, add notation, etc.

The MMRL allows for annotations. These annotations can be used to register the changes that take place. Annotations also code for lay-out instructions in the eventual editor; classes that have been added, instead of renamed, will be represented in a different color. The reason for this is to let the user focus on the concepts that are directly from the automotive.

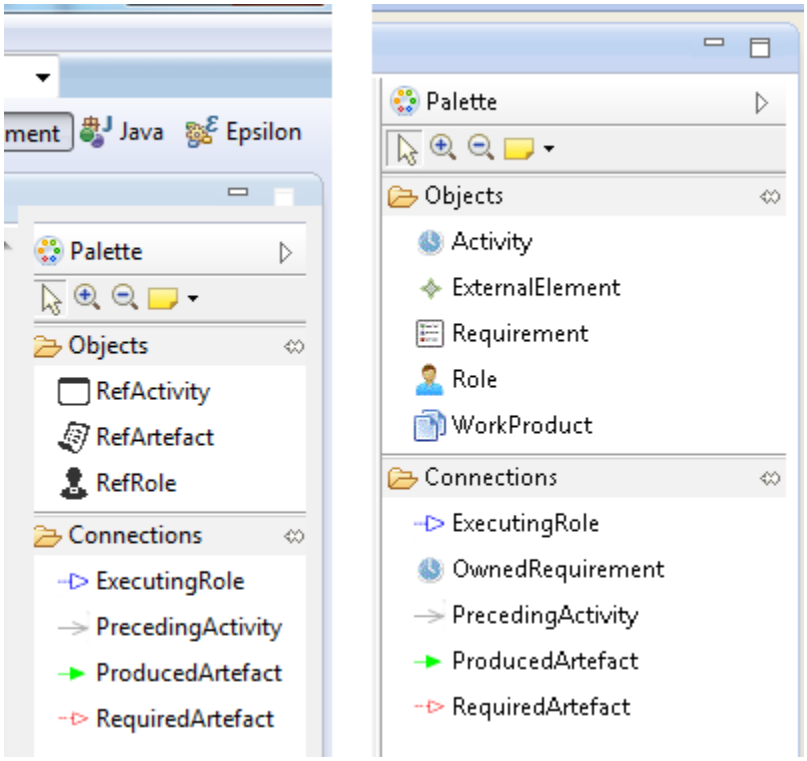

Figure 3: **The different palletes in the straightforward implementation (left) and the alternative one (right).**

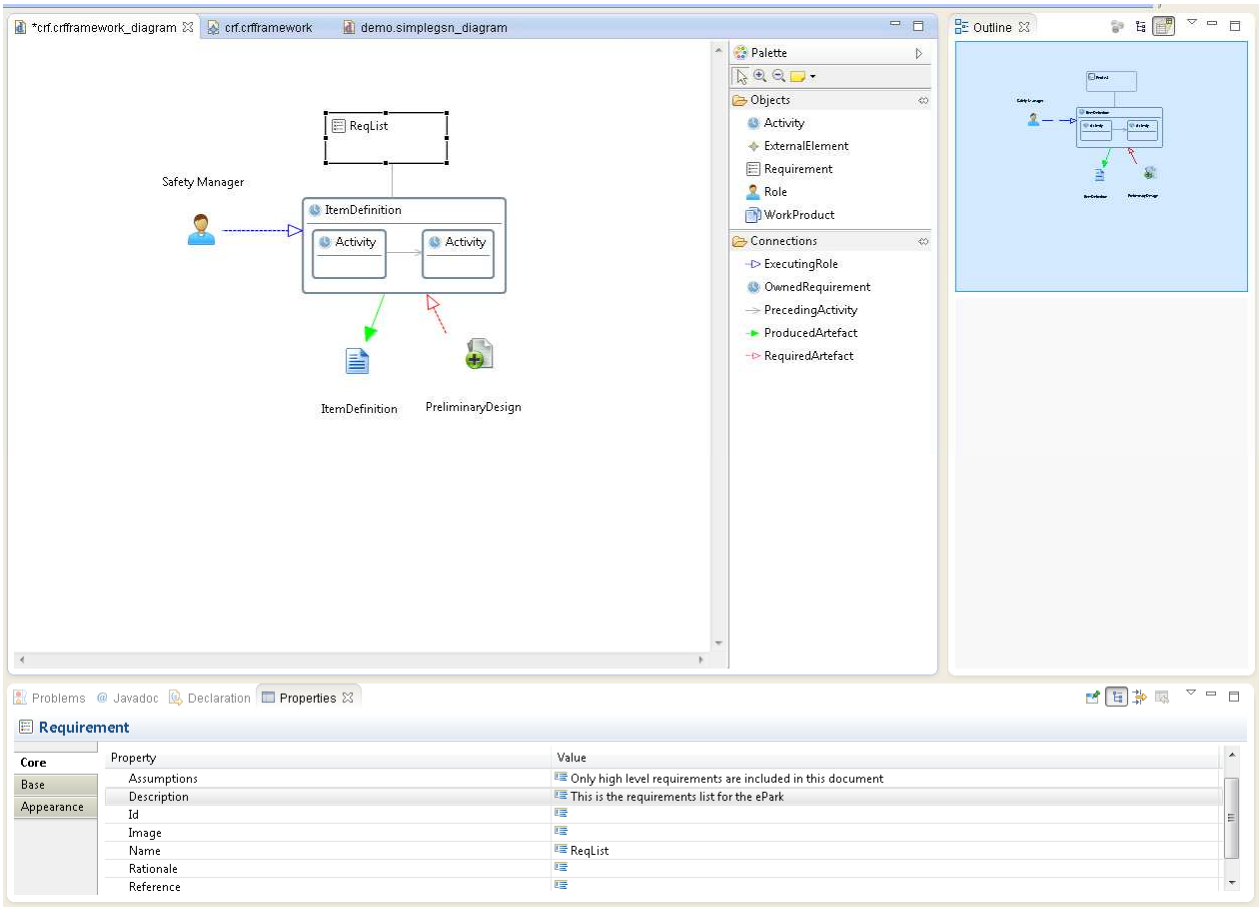

Figure 4: Reference Assurance Framework overview in the alternative implementation

Table 2: Example MetaModel Refine Language (MMRL)

```
operations
  rename refframework to ISO26262framework
  rename refframework.RefFramework to ISO26262Framework
  rename refframework.RefApplicabilityLevel to Importance
  rename refframework.RefCriticalityLevel to ASIL
  rename refframework.RefArtefact to WorkProduct
  rename refframework.RefActivity to Activity
  rename refframework.RefRequirement to Requirement
  rename refframework.RefRole to Role

   add class ExternalElement {
    superTypes {
       refframework.RefArtefact
    }
  } to refframework

   add annotation "gmf.node" {
    "label" = "name",
    "size" = "70,50",
    "label.placement" = "internal"
  } to refframework.RefRequirement

   add annotation "gmf.node" {
    "label" = "name",
    "size" = "70,50",
    "figure"="figures.ExternalElementFigure",
    "label.icon"="false",
    "label.placement" = "external"
  } to refframework.ExternalElement

    add annotation "gmf.link" {
  } to refframework.RefActivity.ownedRequirement
```