

Document history			
V	Date	Author	Description
<i>0.1</i>	<i>2011-01-10</i>	<i>NAO</i>	<i>ToC and first content</i>
<i>0.2</i>	<i>2011-01-10</i>	<i>HEO, ANM</i>	<i>Review and comments about the structure and content</i>
<i>0.3</i>	<i>2011-01-17</i>	<i>NAO</i>	<i>Content</i>
<i>0.4</i>	<i>2011-01-24</i>	<i>HEO</i>	<i>Introduction and conclusions</i>
<i>0.5</i>	<i>2011-01-25</i>	<i>ANM</i>	<i>Review</i>
<i>1.0</i>	<i>2011-01-31</i>	<i>NAO</i>	<i>Final version</i>

Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The document reflects only the author's views and the Community is not liable for any use that may be made of the information contained therein.

Summary

The D6.3-D Software Engineering Process for Certification Metamodel is a public document delivered in the context of WP6, Task 6.3: Specification of a Certification Metamodel for Energy Management Deployments with regard to how has been modelled the process for certification.

This document introduces the underlying architecture and use cases that describe the Software Engineering Process for Certification that has been implemented in the next deliverable D6.3-E Executable Process Framework for Certification.

Contents

SUMMARY	3
CONTENTS	4
ABBREVIATIONS	5
1. INTRODUCTION	6
2. CERTIFICATION METAMODEL (FINAL VERSION)	7
2.1 GENERAL APPROACH	7
2.2 METAMODEL OVERVIEW	10
2.3 QUALIFICATION REFERENCES METAMODEL	12
2.4 DEVELOPMENT PROJECT METAMODEL	15
2.5 ASSESSMENT PROJECT METAMODEL	17
3. CERTIFICATION USE CASES	19
4. CONCLUSION	30
ACKNOWLEDGEMENTS	32
REFERENCES	32
ANNEX A. CERTIFICATION PROTOTYPE PLUG-INS	33
A.1. ES.ESI.GEMDE.CERTIFICATION QREFERENCE PLUG-INS	33
A.2. ES.ESI.GEMDE.CERTIFICATION.DPROJECT PLUG-INS	34
A.3. ES.ESI.GEMDE.CERTIFICATION.ASSESSMENT PLUG-INS	34
A.4. ES.ESI.GEMDE.CERTIFICATION.FRAMEWORK PLUG-IN	35
A.5. ES.ESI.GEMDE.UTILS PLUG-IN	36

Abbreviations

eDIANA Embedded Systems for Energy Efficient Buildings

1. Introduction

This document is the fourth deliverable of Task 6.3, "Software Engineering Process for Certification Metamodel" (D6.3-D), in WP6, "Compositional Verification, Validation and Certification". While tasks 6.1 and 6.2 focus on Verification and Validation issues, Task 6.3 deals with Certification aspects.

The term "certification" identifies a process whose goal is to provide evidence of compliance with an industrial standard (ISO 61508, EPBD, etc.). While the term certification is often understood as a "legal" certification against a certification body, we always use the term "certification" to indicate the process to which a embedded system (such as those that are a part of the eDIANA platform) is subject to in order to assess its compliance to specific good practices or standards. This includes the compliance of the process by which this embedded system has been built.

Due to this diversity of standards, the prototype targeted in eDIANA (T6.3) focuses on creating a general certification "language" by means of a structured semi-formal metamodel, which will act like a template for certification requirements specification. This metamodel will be used to build domain-specific libraries of certification models, which will act as a knowledge database, providing information about eDIANA-related standards (IEC 61508, EPBD, etc.) and other eDIANA-specific implementation requirements (some kind of eDIANA-compliant label). Such "knowledge database" is used to build a set of guidelines akin to "spell-checking", in which a number of compliance checks are performed to assess the degree of compliance of embedded system products against eDIANA-related standards.

Deliverable D6.3-D describes the final version of the certification metamodel and provides full details about the metamodel use cases. The deliverable D6.3-B has been the basis to create this certification metamodel. This metamodel allows for specifying the certification items/requirements with respect to a common notation. Using a common notation for different certification standards enables for certification items management in a common format, certification evidence management, certification assessment, and re-certification between different standards. Deliverable D6.3-C has surveyed and assesses process languages (SPEM, BPMN, OPEN, etc.) to explore the possibilities to extend the Certification metamodel and its prototype (D6.3-E).

This deliverable is the technical support of the certification tool prototype implementation. The user manual will be delivered as a companion document to D6.3-E. Also, the use of this prototype with eDIANA components will be assessed in WP7 (pre-industrial demonstration).

2. Certification Metamodel (Final Version)

This section describes the Certification Metamodel, which is the conceptual and data information basis of the Certification Framework. As previously mentioned, D6.3-B has presented a first version of this metamodel. However, as the Framework prototype has evolved and validated during the project; some modifications has been introduced. In addition, deliverable D6.3-C has evaluated a number of process specification languages (some of them are standards). The goal was to study possible future interactions of the Certification prototype with engineering process modelling. Although these interactions will not be supported by the eDIANA certification prototype, some considerations have been taken into account in the certification metamodel for this possible connection. This final version of the certification metamodel includes all these improvements.

2.1 General Approach

The proposed approach for certification in eDIANA aims at *defining and prototyping methods and tools for incremental certification using a unified representation of certification assets*.

The principles behind this approach are summarized as follows:

- Management of:
 - (a) clear definition of certification requirements,
 - (b) evidences
 - (c) claims and arguments for assessment.
- Certification assessments vary from project to project, but we should be able to reuse certification assets from previous projects.

Figure 1 describes the certification tool prototype. This figure shows the modules or components of the certification framework, as well as their interactions and relationships. The certification framework has three main modules (which match with the management aspects a), b) and c) abovementioned in this section):

- A. ***Certification Reference Manager***. It enables certification requirements management.
- B. ***Certification Project***. It supports certification evidence management.
- C. ***Certification Assessment***. It allows users to perform certification assessments.

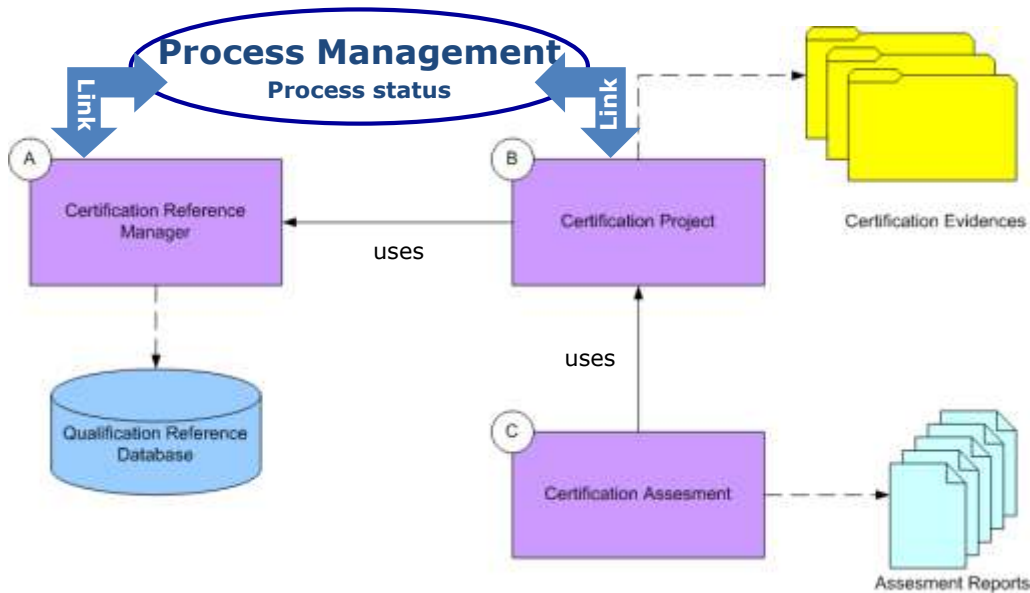


Figure 1 Certification framework prototype (modules A, B and C)

In addition, it must be noted that two links exist with the process management tools. As described in D6.3-C, these links will allow future extensions to:

- Simplify certification assessment by tracking evidences from early phases. The link of Module A with a process management tool allows for identifying process activities and tasks with certification requirements.
- Automatically update the status of project process regarding certification requirements. The link of Module B with a process management tool allows for dynamically update process status with certification compliance actions.

In order to understand the relationship between the modules in the certification framework and the certification metamodel itself, we describe the functionality and the main stakeholders of each module:

A. **Certification Reference Manager.** This module has the following basic functionality:

- Create, maintain and refine certification requirements
- Provide a common structure and notation for certification requirements
- Map certification requirements from different standards or internal practices

The main stakeholders or roles of module A are:

- Quality assurance manager (for qualification of eDIANA components or platform against internal rules defined in the eDIANA project)
- Certification library providers (certification against standards, e.g. IEC 61508)

B. **Certification Project.** This module has the following basic functionality:

- Selection of an applicable subset of certification requirements in the context of development projects
- Project-specific interpretation of certification requirements
- Compliance checks against evidence

The main stakeholders or roles of module B are:

- Quality assurance manager (internal qualification)
- Architects, analysts and developers

C. **Certification Assessment.** This module has the following basic functionality:

- Assessment support for inspection entities
- Access to Evidence and additional justifications
- Reporting

The main stakeholders or roles of module C are:

- Inspection (Certification) Entities
- Quality Managers (internal qualification)

To support these functionalities, the certification metamodel has been also broken down into three sub-metamodels, each allocated to the modules A, B and C respectively:

- **Qualification References metamodel** (it supports module A)
- **Development Project metamodel** (it supports module B)
- **Assessment Project metamodel** (it supports module C).

In the next section, an overview of the whole metamodel of the framework will be presented, and the remaining sections will describe in detail the different metamodels of each module.

2.2 Metamodel Overview

Due to the complexity of the complete scenario that applies to the Certification Framework, the Certification Metamodel has been implemented as composed of three associated metamodels.

The Qualification References metamodel, Figure 2, provides the mechanism to describe the standards, guides or best practices that will be used to certify the development of the product. This metamodel should cover the first step of the Software Engineering Process, the definition of the standard that the project/product/component.

The next Development Project metamodel is associated to the Qualification References metamodel through the classes:

- QualificationReference $\leftarrow \rightarrow$ DevelopmentProject
- QualificationRequirement $\leftarrow \rightarrow$ ProjectRequirement

The Development Project metamodel the mechanism to build the evidences to all the requirements the standard presents.

When a new Development Project is generated, it must have been associated to a Qualification Reference already defined in the Framework. Thus, the Project Requirements can be generated from the requirements of the associated Qualification Reference, and in the same way, and associated to those requirements. These associations will allow a quick access to the information of the standard when the evidences of each Project Requirement are determined.

In the same way, the Assessment Project Metamodel is associated to the Development Project Metamodel through next associations between classes:

- DevelopmentProject $\leftarrow \rightarrow$ AssessmentProject
- ProjectRequirement $\leftarrow \rightarrow$ ReqEvaluation

These associations will provide a quick connection to the requirements of the project and their evidences, allowing the user to state if the evidence is enough to declare the requirement fulfilled or not.

Next sections offer a detailed description of each submodel explained in this section.

2.3 Qualification References metamodel

This section explains in detail the classes of the Qualification References metamodel, as well as the relations between them.

The main class is **QualificationReference**. This class provides support to define some attributes of the standard like identifier, description, version, organization, and date.

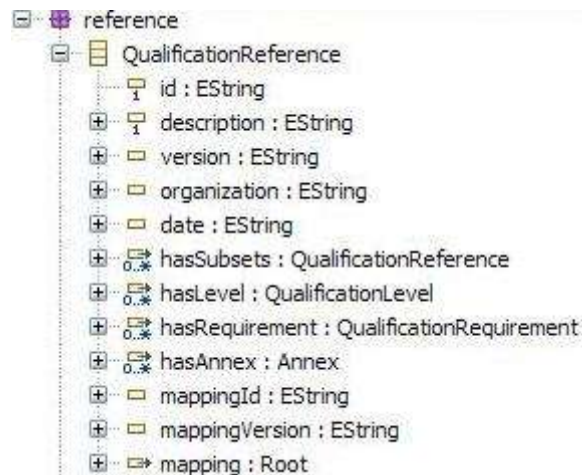


Figure 3 Qualification Reference

A class of QualificationReference can contain classes of the same type (hasSubsets reference), this characteristic allows the definition of standards that are composed of several parts, and these parts are entities that can not be defined as requirements or a set of requirements. To illustrate this characteristic, the standard IEC 61508 analyzed in D6.3-A has 7 parts that, being a set of requirements, has to be defined as a QualificationReference or a subset of a QualificationReference.

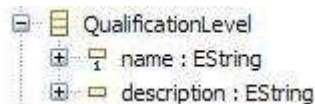


Figure 4 Qualification Level

An instance of QualificationReference can have several **QualificationLevel** (hasLevel reference). Those levels allow the user to define several levels of accomplishment of the standard

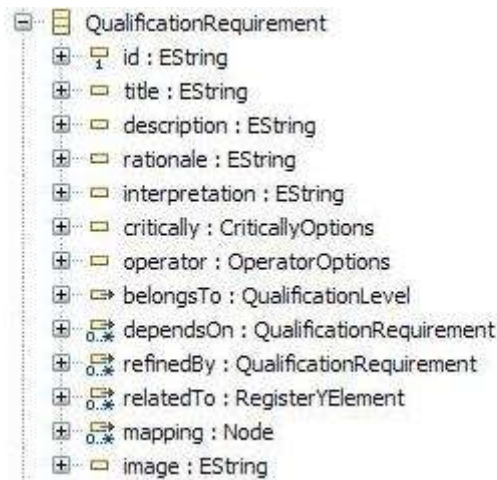


Figure 5 Qualification Requirement

An instance of **QualificationReference** is composed of several **QualificationRequirement** (hasRequirement reference). This class provides the structure to define the requirements of the standard, to accomplish this task, some fields are defined: identifier, title, description, rationale of the requirement, interpretation, critically, image. The definition of the requirements is made over a tree structure that groups requirement (refinedBy reference). The attribute operator enables the qualification of these grouped requirements as a conjunction or a exclusive disjunction.

A requirement of the standard can be associated to a defined level of accomplishment of the standard (belongsTo reference).

Sometimes, it is necessary to associate a requirement to a table that provides more information of the requirement; the attribute relatedTo provides a mechanism to relate the attribute to a table defined inside a annex.



Figure 6 Annex

The class **Annex** has been defined to group the necessary tables of the standard. An annex is composed by tables and entries of the tables. The definition of the entries at this level allows their reuse in several tables without compromising the size of the framework.

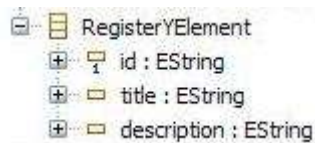


Figure 7 RegisterYElement

The class **RegisterYElement** is an abstract class that enables that a table can be composed of registers or other tables.



Figure 8 Table, Entry, Register

The framework supports two dimensional tables since some standards provide information in tables depending on two parameters. A **Table** class is composed by **Registers**. This register provides information about the recommendation the standard provides, a description and a comment. The register is associated to a level of accomplishment of the standard and to an element of RegisterYElement that can be a table or an entry. This structure enables the building of tables like this:

	QualificationLevel 1	Qualification Level 2	...	Qualification Level n
Entry 1	Recommendation	Recommendation	...	Recommendation
Entry 2	Recommendation	Recommendation	...	Recommendation
Table 1	Recommendation	Recommendation	...	Recommendation
....
Entry n	Recommendation	Recommendation	Recommendation	Recommendation
Table n	Recommendation	Recommendation	Recommendation	Recommendation

Table 1 Table of annexes

2.4 Development project metamodel

This section explains in detail the classes of the Development Project metamodel, as well as the relations between them.

The main class is **DevelopmentProject**. This class provides support to define some attributes that will characterize the project that will be developed, taking into account the associated standard. Some attributes are: name, organization, openingDate, responsible, description, status and comment.

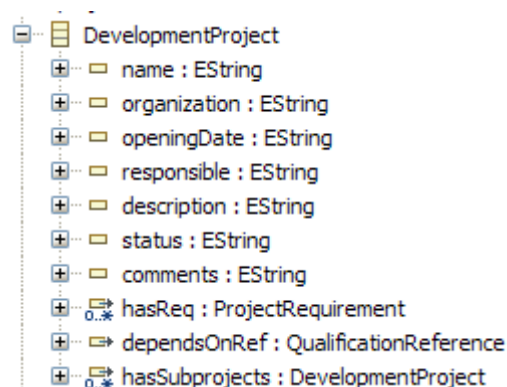


Figure 9 Development Project

This class corresponds to the core of the project/product/component that will be developed according to the selected standard. The association to the standard must be made when it is created (dependsOnRef reference), in order to generate all the necessary requirements (those requirements are associated through the hasReq reference). As well as the Qualification Reference can contain entities of the same class (subprojects), the same facility has been added to Development Project, in order to allow the user maintain the same structure.

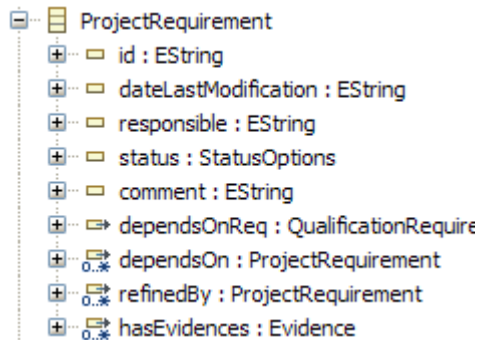


Figure 10 Project Requirement

The class **ProjectRequirement** provides the infrastructure that connects to the requirements of the standard (dependsOnReq reference). Some attributes characterize this requirement: identification (similar to the standard attribute), date of the last modification, responsible of implementing the requirement, status of the accomplishment and comments. An entity of this class can contain groups of requirements the same way that the standard requirements (refinedBy reference). Finally, this class can contain several evidences (hasEvidences reference).

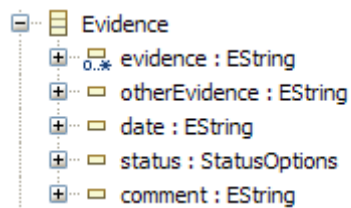


Figure 11 Evidence

The class Evidence gives the user the support to define the evidences that show how the standard requirement is accomplished. The evidences must be in a digital format to be included, if this is not possible, there is an attribute that offers the possibility of add text that will show how the requirement is fulfilled.

A requirement can have several versions of the evidences, they will be identified by the date and the status. There is the possibility of adding comments by the responsible of adding the evidence.

2.5 Assessment project metamodel

This section explains in detail the classes of the Assessment Project metamodel, as well as the relations between them. The Assessment Project is the third metamodel of the framework and it allows the assessor to do the assessment of the requirements and the associated evidences.

The main class is **AssessmentProject**. This class provides support to define some attributes that will help to define if the associated requirement has been fulfilled properly. The attributes are: name, organisation, assessor who performs the assessment of the project, date of the starting of the project assessment, date of the end of the project assessment, the assessment result and comments about the project assessment.

As well as the Development Project can contain entities of the same class (subprojects), the same facility has been added to Assessment Project (hasProject reference), in order to allow the user maintain the same structure.

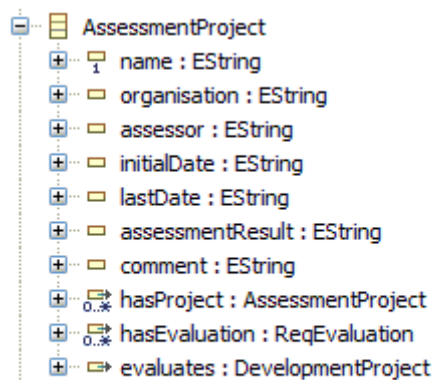


Figure 12 Assessment Project

The assessment project is associated to a DevelopmentProject through the reference evaluates, and it is composed of requirements evaluations (hasEvaluation reference).

The **ReqEvaluation** makes possible the definition of an assessment to each requirement.

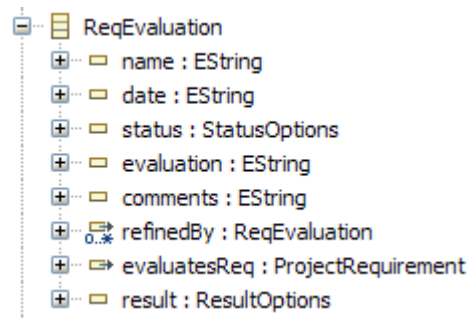


Figure 13 Req. Evaluation

The entities of this class can contain entities of the same class (refinedBy reference) and must be associated to entities of ProjectRequirement (evaluatesReq reference). Some attributes contribute to the definition of the requirement assessment: name of the assessment, date when the assessment has been made, status of the assessment, evaluation, comments and result.

3. Certification Use Cases

This section will describe the use cases that apply to the Certification Framework.

In next Figure 2 , the use cases describe the Software Engineering Process of developing a project/product/component attending to the certification. Those use cases marked in blue are the ones that the Certification Framework covers.

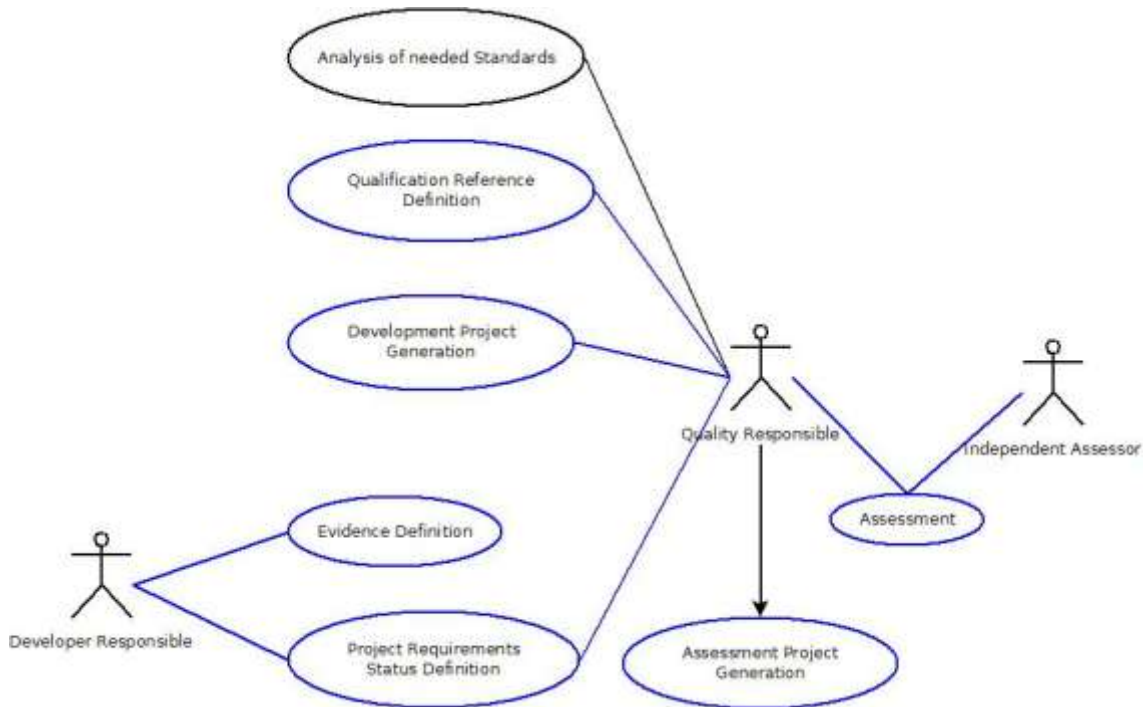


Figure 14 Use Cases

The use cases related to the Certification are detailed next:

Use Case ID:	1		
Use Case Name:	Qualification Reference Definition		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	Quality Responsible. He is the responsible of determining the quality levels of the project/product/component to develop, as well as the standards, part of standards, best practices that must be fulfilled.
Description:	This use case covers the creation or introduction of the standards and their requirements into the Certification Framework.
Trigger:	Selection of the standards to fulfil in the project.
Preconditions:	<ol style="list-style-type: none"> 1. Analysis of the standards. 2. Knowledge of the selected standards. 3. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.qreference plug-ins.
Postconditions:	<ol style="list-style-type: none"> 1. The framework must have the requirements of the standards, practices well described and related between them.
Normal Flow:	<ol style="list-style-type: none"> 1. Decide the Qualification reference start point. 2. Complete the fields of the Qualification reference. 3. Check if subsets of the reference are needed. 4. Group requirements. 5. Define requirements. 6. Complete requirements. 7. Establish relationships between requirements. 8. Check if there must be annexes and tables. 9. Introduce information about annexes and tables. 10. Establish relationships between requirements and tables.
Alternative Flows:	<ol style="list-style-type: none"> 3. If there are subsets, repeat step 1 and 2 for the subsets. 8. If there are not annexes or tables, finish.
Exceptions:	If the Qualification Reference is provided, this use case is not necessary.
Includes:	n/a
Priority:	High.
Frequency of Use:	Every time a new standard or practice will be used to certify a product/project/component.
Business Rules:	n/a

Special Requirements:	n/a
Assumptions:	The actor has knowledge about the project/product/component that will be certified, as well as the standard that has to be apply.
Notes and Issues:	n/a

Table 2 Qualification Reference Definition Use Case

Use Case ID:	2		
Use Case Name:	Development Project Generation		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	Quality Responsible. He is the responsible of determining the quality levels of the project/product/component to develop, as well as the standards, part of standards, best practices that must be fulfilled.
Description:	This use case covers the creation of the requirements that the project/product/component must fulfil, selecting them from the associated standard.
Trigger:	Start of the development project.
Preconditions:	<ol style="list-style-type: none"> 1. The required standards have to be introduced into the Framework as Qualification References. 2. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.dproject plug-ins. 3. The directory where the standards are located must be in preferences.
Postconditions:	<ol style="list-style-type: none"> 1. The development project must be created inside the Framework. The necessary requirements of the standard that has to be fulfilled has to be implemented as project

	requirements.
Normal Flow:	1. Generate a new project. 2. Select the desired standard. 3. Check the requirements that must be fulfilled. 4. Generate the project
Alternative Flows:	n/a
Exceptions:	n/a
Includes:	n/a
Priority:	High
Frequency of Use:	Every time a new project/product/component has to be certified
Business Rules:	n/a
Special Requirements:	n/a
Assumptions:	The actor has knowledge about the project/product/component that will be certified, as well as the standard that has to be apply
Notes and Issues:	n/a

Table 3 Development Project Generation Use Case

Use Case ID:	3		
Use Case Name:	Evidence definition		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	Development Responsible. He is the responsible of development of the project/product/component to develop. He must have knowledge about the project, its planning, milestones, technical issues... as well as the selected standards requirements that should be fulfilled.
Description:	This use case covers the definition of the evidences of the requirements that the project/product/component must fulfil.
Trigger:	A predefined activity has been completed
Preconditions:	<ol style="list-style-type: none"> 1. The development project must be created, with the necessary requirements. 2. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.dproject plug-ins.
Postconditions:	<ol style="list-style-type: none"> 1. The requirements of the development project must have at least an evidence that demonstrate that the requirement has been fulfilled. 2. By the end of the project, all the requirements must have associated an evidence at least.
Normal Flow:	<ol style="list-style-type: none"> 1. Open the project. 2. Find the requirements that correspond to the finalised activities. 3. Found. 4. For each requirement: <ol style="list-style-type: none"> a. Complete the data of the requirement: Last Modification Data, Responsible, Status, Comment. b. Create the associated evidence. c. Modify the date and the status, if necessary. d. Select the digital documents that demonstrate that the requirement has been fulfilled. e. Add textual evidences, if necessary. f. Add comments, if necessary. 5. Save and close the project.
Alternative Flows:	3. If there are no requirements to be satisfied, look through the plan with the quality responsible in order to avoid misunderstandings and mistakes.
Exceptions:	n/a
Includes:	n/a
Priority:	Medium/High
Frequency of	Every time a milestone of the project has been.

Use:	
Business Rules:	n/a
Special Requirements:	n/a
Assumptions:	The actor has knowledge about the project/product/component that will be certified, as well as the standard that has to be applied.
Notes and Issues:	n/a

Table 4 Evidence definition

Use Case ID:	4		
Use Case Name:	Project Requirements Status Definition		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	<p>Development Responsible. He is the responsible of development of the project/product/component to develop. He must have knowledge about the project, its planning, milestones, technical issues... as well as the selected standards requirements that should be fulfilled.</p> <p>Quality Responsible. He is the responsible of determining the quality levels of the project/product/component to develop, as well as the standards, part of standards, best practices that must be fulfilled.</p>
Description:	This use case covers the definition of the status of the evidences of the requirements that the project/product/component must fulfil.

Trigger:	When a milestone has been achieved.
Preconditions:	<ol style="list-style-type: none"> 1. The development project must be created, with the necessary requirements. 2. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.dproject plug-ins.
Postconditions:	<ol style="list-style-type: none"> 1. The requirements of the development project must have, at least, one evidence that demonstrate that the requirement has been fulfilled. 2. By the end of the project, all the requirements must have associated one evidence, at least. 3. By the end of the project, all the requirements and evidences must have the status: finished or other.
Normal Flow:	<ol style="list-style-type: none"> 1. Open the project. 2. Find the requirements that correspond to the activities made in the actual milestone. 3. Found. 4. For each requirement: <ol style="list-style-type: none"> a. Complete the data of the requirement: Last Modification Data, Responsible, Status, Comment. b. If not, create the associated evidence. c. Modify the date and the status, if necessary. d. Select the digital documents that demonstrate that the requirement has been fulfilled, if necessary. e. Add textual evidences, if necessary. f. Add comments, if necessary. g. Put requirement and evidence status as "finished" or "other". 5. Save and close the project
Alternative Flows:	<ol style="list-style-type: none"> 3. If there are no requirements to be satisfied, look through the plan with the quality responsible in order to avoid misunderstandings and mistakes.
Exceptions:	n/a
Includes:	n/a
Priority:	Medium/High
Frequency of Use:	Every time a milestone of the project has been completed.
Business Rules:	n/a

Special Requirements:	n/a
Assumptions:	The actor has knowledge about the project/product/component that will be certified, as well as the standard that has to be apply.
Notes and Issues:	n/a

Table 5 Project Requirements Status Definition

Use Case ID:	5		
Use Case Name:	Assessment Project Generation		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	Quality Responsible. He is the responsible of determining the quality levels of the project/product/component to develop, as well as the standards, part of standards, best practices that must be fulfilled.
Description:	This use case covers the creation of the assessment project with all the requirements evaluations.
Trigger:	When the development of the project has been finished.
Preconditions:	<ol style="list-style-type: none"> 1. The development project has been completed with the evidences. 2. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.assessment plug-ins.
Postconditions:	<ol style="list-style-type: none"> 1. The assessment project has been generated. 2. All the evaluations of the requirements have been created.
Normal Flow:	<ol style="list-style-type: none"> 1. Open eclipse and select a new assessment project. 2. Select the development project which it will be associated to. 3. Generate the assessment project.

	4. Save and close the project.
Alternative Flows:	
Exceptions:	n/a
Includes:	n/a
Priority:	Medium/High
Frequency of Use:	Once the project has been completed.
Business Rules:	n/a
Special Requirements:	n/a
Assumptions:	The actors has knowledge about the project/product/component that will be certified, as well as the standard that has to be apply and the necessary techniques and methodologies to do the assessment.
Notes and Issues:	n/a

Table 6 Assessment Project Generation

Use Case ID:	6		
Use Case Name:	Assessment		
Created By:	NAO	Last Updated By:	
Date Created:	2011-01-14	Date Last Updated:	

Actors:	Quality Responsible. He is the responsible of determining the quality levels of the project/product/component to develop, as well as the standards, part of standards, best practices that
---------	--

	<p>must be fulfilled.</p> <p>Independent Assessor: He has deep knowledge of the selected standards, besides a high expertise in the standard relevant techniques and methodologies.</p>
Description:	This use case covers the definition of the status of the evidences of the requirements that the project/product/component must fulfil.
Trigger:	When a milestone has been achieved.
Preconditions:	<ol style="list-style-type: none"> 1. The assessment project must be created, with the necessary requirements evaluations. 2. Having installed Eclipse, (EEF included) and the es.esi.gemde.certification.assessment plug-ins.
Postconditions:	<ol style="list-style-type: none"> 1. The requirementsevaluations will have their fields "Evaluation" and "Result" with valid values. 2. The assessment project will have its field "Assessment Result" with a valid value.
Normal Flow:	<ol style="list-style-type: none"> 1. Open the assessment project. 2. For each requirement evaluation: <ol style="list-style-type: none"> h. Complete the data of the requirement evaluation: Name, Date, Status, Evaluation, Comments. 3. Complete the data of the assessment project: Name, Organisation, Assessor, Initial Date, Last Date, Assessment Result, and Comment. 4. Save and close the project.
Alternative Flows:	
Exceptions:	n/a
Includes:	n/a
Priority:	Medium/High
Frequency of Use:	When the project has been completed.
Business Rules:	n/a
Special Requirements:	n/a

Assumptions:	The actors has knowledge about the project/product/component that will be certified, as well as the standard that has to be apply and the necessary techniques and methodologies to do the assessment.
Notes and Issues:	n/a

Table 7 Assessment

4. Conclusion

The main goal of Task 6.3 is to define a generic framework to assess the compliance level of eDIANA components and deployments against standard and practices. We understand "compliance" as the extent to which eDIANA components and its development process have acted in accordance with requirements/recommendations set down in relevant standards and/or eDIANA-specific good practices. Figure 15 below describes the overall certification approach in eDIANA.

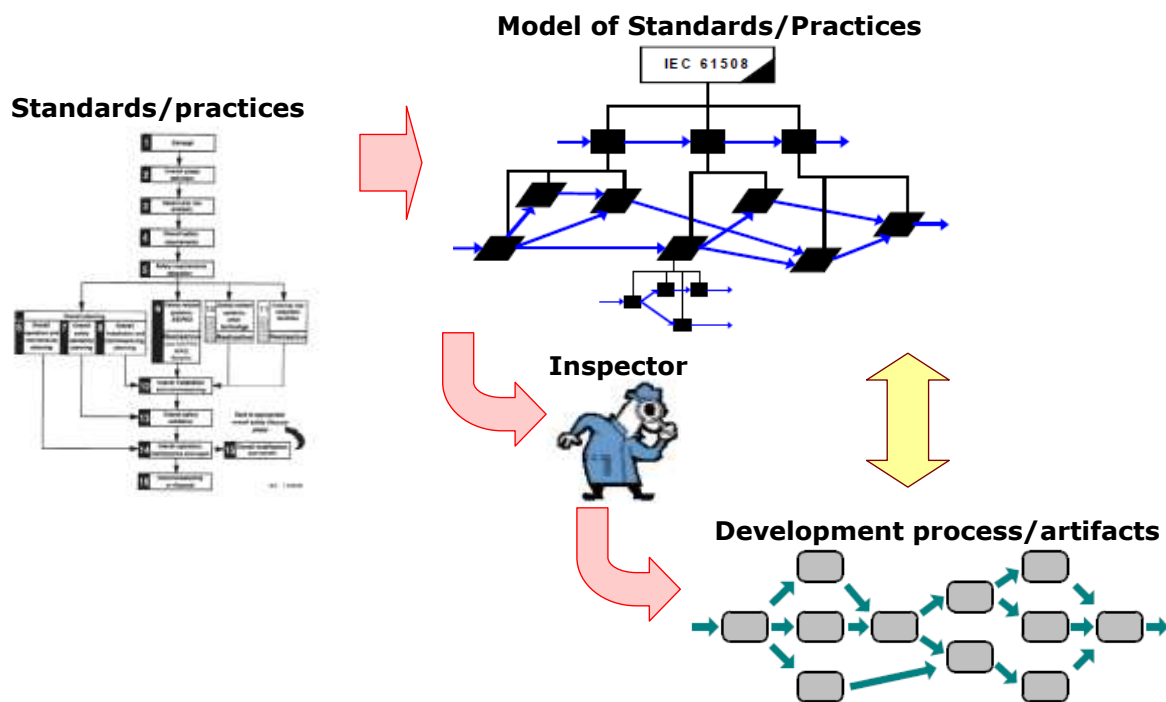


Figure 15 Certification Framework Metamodel

This deliverable (D6.3-D) describes the final version of the certification metamodel proposed for eDIANA, and provides full details about the metamodel use cases. The deliverable D6.3-B has been the basis to create this certification metamodel. This metamodel allows for specifying the certification items/requirements with respect to a common notation. This common notation for different certification standards (e.g., IEC 61508) enables for certification items management in a common format, certification evidence management, certification assessment, and re-certification between different standards. This final version of the certification metamodel and their use cases in a general engineering process is the result of prototype evaluations during the project.

Some improvements regarding the initial metamodel (D6.3-B) implemented in this final version are:

- Evidences management has been improved by supporting versioning control.
- The certification assessment concepts have been made explicit in a separate metamodel
- The metamodel of qualification reference supports now recommendation Tables (see Table 1).
- New attributes have been included in many meta-classes to support a richer information certification repository.

The main issue tackled by Task 6.3 is the reduction of costs and time to regenerate the evidence-set when evolutions of the eDIANA components or the whole platform are performed. To demonstrate these saving, deliverable D6.3-E shows a first iteration of the eDIANA iEi compliance qualification. This example illustrates how different standards or best practices can be defined with the help of this prototype; and how it can be used to support the certification of projects that accomplish it. The full demonstrations will be assessed in WP7, T7.3 (As-is model delivery).

Acknowledgements

The eDIANA Consortium would like to acknowledge the financial support of the European Commission and National Public Authorities from Spain, Netherlands, Germany, Finland and Italy under the ARTEMIS Joint Technology Initiative.

References

- [1] EEF Project. <http://wiki.eclipse.org/EEF>.
- [2] Griffing, Catherine. Using EMF. International Business Machines Corp, 2002
- [3] Budinsky, Frank; Steinberg, David; Merks, Ed; Ellersick, Raymond; Grose, Timothy J.. Elclipse Modeling Framework. Addison-Wesley, 2004

Annex A. Certification Prototype Plug-ins

There are eleven plug-ins that form the complete Certification framework. The plug-ins *qreference*, *dproject* and *assessment* use the Extended Editing Framework component (EEF) to get advanced editing components for the properties of EMF elements and a default generation based on standard metamodels using these components.

A.1. es.esi.gemde.certification.qreference plug-ins

The Qualification Reference metamodel is the basis of the three plug-ins that form the implementation of the infrastructure that allows the creation and definition of the standards.

The plug-ins are:

- es.esi.gemde.certification.qreference: this plug-in contains the ecore model as well as the code that implements this model. This code consists of:
 - reference -- Interfaces and the Factory to create the Java classes
 - reference.impl -- Concrete implementation of the interfaces defined in model
 - reference.util -- The AdapterFactory
- es.esi.gemde.certification.qreference.edit: this plug-in contains adapter classes needed by the generated editor.
 - reference.provider: Content and label provider classes, property source support, and other convenience classes that allow EMF models to be displayed using property sheets.
 - EEF (src-gen directory): Provided widget has been customized for the framework
 - reference.components: Components for the control part
 - reference.parts: parts for views
 - reference.providers: Providers for the structure instantiation
- es.esi.gemde.certification.qreference.editor: this plug-in generates the editor for the reference model.

A.2. es.esi.gemde.certification.dproject plug-ins

The Development Project metamodel is the basis of the three plug-ins that form the implementation of the infrastructure that allows the creation and definition of the project that is associated to the standard and will be used to define the evidences that accomplished the requirements.

The plug-ins are:

- es.esi.gemde.certification.dproject: this plug-in contains the ecore model as well as the code that implements this model. This code consists of:
 - dproject -- Interfaces and the Factory to create the Java classes
 - dproject.impl -- Concrete implementation of the interfaces defined in model
 - dproject.util -- The AdapterFactory
- es.esi.gemde.certification.dproject.edit: this plug-in contains adapter classes needed by the generated editor.
 - dproject.provider: content and label provider classes, property source support, and other convenience classes that allow EMF models to be displayed using property sheets.
 - dproject.dialogs: the dialog that allows the selection of the evidences.
 - EEF (src-gen directory): Some widgets have been customized for the framework, more specifically the multivalued editor widget.
 - dproject.components: Components for the control part
 - dproject.parts: parts for views.
 - dproject.providers: Providers for the structure instantiation
- es.esi.gemde.certification.dproject.editor: this plug-in generates the editor for the reference model.

A.3. es.esi.gemde.certification.assessment plug-ins

The Assessment Project metamodel is the basis of the three plug-ins that form the implementation of the infrastructure that allows the creation and definition of the assessment of the development project that is associated.

The plug-ins are:

- `es.esi.gemde.certification.assessment`: this plug-in contains the ecore model as well as the code that implements this model. This code consists of:
 - `assessment` -- Interfaces and the Factory to create the Java classes
 - `assessment.impl` -- Concrete implementation of the interfaces defined in model
 - `assessment.util` -- The AdapterFactory
- `es.esi.gemde.certification.assessment.edit`: this plug-in contains adapter classes needed by the generated editor.
 - `assessment.provider`: content and label provider classes, property source support, and other convenience classes that allow EMF models to be displayed using property sheets.
 - `EEF (src-gen directory)`: Some widgets have been customized for the framework.
 - `assessment.components`: Components for the control part
 - `assessment.parts`: parts for views.
 - `assessment.providers`: Providers for the structure instantiation
- `es.esi.gemde.certification.assessment.editor`: this plug-in generates the editor for the reference model.

A.4. es.esi.gemde.certification.framework plug-in

This plug-in implements the infrastructure associated to the Eclipse environment. Specifically:

- `es.esi.gemde.certification.framework.ui.perspective`: it adds the perspective of the Certification Framework
- `es.esi.gemde.certification.framework.preferences`: it adds to the Eclipse preferences, the Certification Framework preferences
- `es.esi.gemde.certification.framework.ui.wizards`: it adds next wizards:

- Wizard that creates a new Qualification Reference
- Wizard that creates a new Development Project. This wizard allows the user associating an existing standard and generates the Project requirements, all of them already associated to the corresponding reference requirement. It also initializes some main fields.
- Wizard that creates a new Assessment Project. This wizard allows the user associating an existing development project and generates the requirements evaluations.

A.5. es.esi.gemde.utils plug-in

This plug-in provides facilities to

- launching applications associated to the evidences
- initializing and using images and icons inside the widgets

using CheckboxTree widget with navigation between children and parents.