FP7 Project 2007- Grant agreement n°: 218575

Project Acronym: **INESS**

Project Title: **INtegrated European Signalling System**

Instrument: Large-scale integrating project
Thematic Priority: Transport

Document Title: Requirements specification Description

Due date of deliverable	2009-10-31
Actual submission date	2010-03-08

Deliverable ID:	D.G.3.2
Deliverable Title:	Requirements specification Description
WP related:	Methods and Tools for Safety Case
Responsible partner:	TUBS
Task/Deliverable leader Name:	Jörg R. Müller
Contributors:	TUBS, BBR, Funkwerk

Start date of the project: 01-10-2008 Duration: 36 Months

Project coordinator: Paolo De Cicco
Project coordinator organisation: UIC

Revision: 1 Dissemination Level[1]: CO

[1] PU: Public, PP: Restricted to other programme participants (including the Commission Services), RE: Restricted to a group specified by the consortium (including the Commission Services), CO: Confidential, only for members of the consortium (including the Commission Services).

Document Information

Document type: Report
Document Name: INESS_WS G_Deliverable 3.2_WS_Finalized_Report_Ver2010-03-08
Revision: 9
Revision Date: 2010-03-08
Author: Jürgen Schröder, Jörg R. Müller (TUBS), G. v. Buxhoeveden
Dissemination level: **CO**

Approvals

	Name	Company	Date	Visa
WP leader	Jörg R. Müller	TUBS		
WS Leader	Jörg R. Müller	TUBS		
Project Manager				
Steering Board				

Document history

Revision	Date	Modification	Author
1	2009-07-02	Creation of document	J. Schröder
2	2009-07-27	Merge with BBR	G. Buxhoeveden
3	2009-07-31	List of function / Collections of Requirements deleted	J. R. Müller
4	2009-09-25	minor modifications	Jörg R. Müller
5	2010-01-06	Method/Tool section expanded	G.Buxhoeveden
6	2010-01-26	Tool section extended	G.Buxhoeveden
7	2010-01-27	Table 1 added, use of natural language	G.Buxhoeveden
8	2010-03-03	Added Tool Enterprise Architect Section	G.Buxhoeveden
9	2010-03-08	Rewritten, Figures 1-4 added, Results of Workshop taken into accout, etc	Jörg R. Müller

TABLE OF CONTENTS

Glossary

The following abbreviations are applied in this document

DMS	Document Management System
EA	Enterprise Architect
MDA	Model Driven Architecture
MS	Microsoft
SRS	System Requirements Specification
SW	Software
TA	Trend Analyst
UML	Unified Modeling Language

Section 1 – Executive Summary

The context of workstream G

The aim of workstream G is to reduce time and money for the Safety Case in industry, i.e. operators as well as suppliers, by avoiding unnecessary or redundant procedures. To achieve this aim one can identify four phases in workstream G (see figure 1).

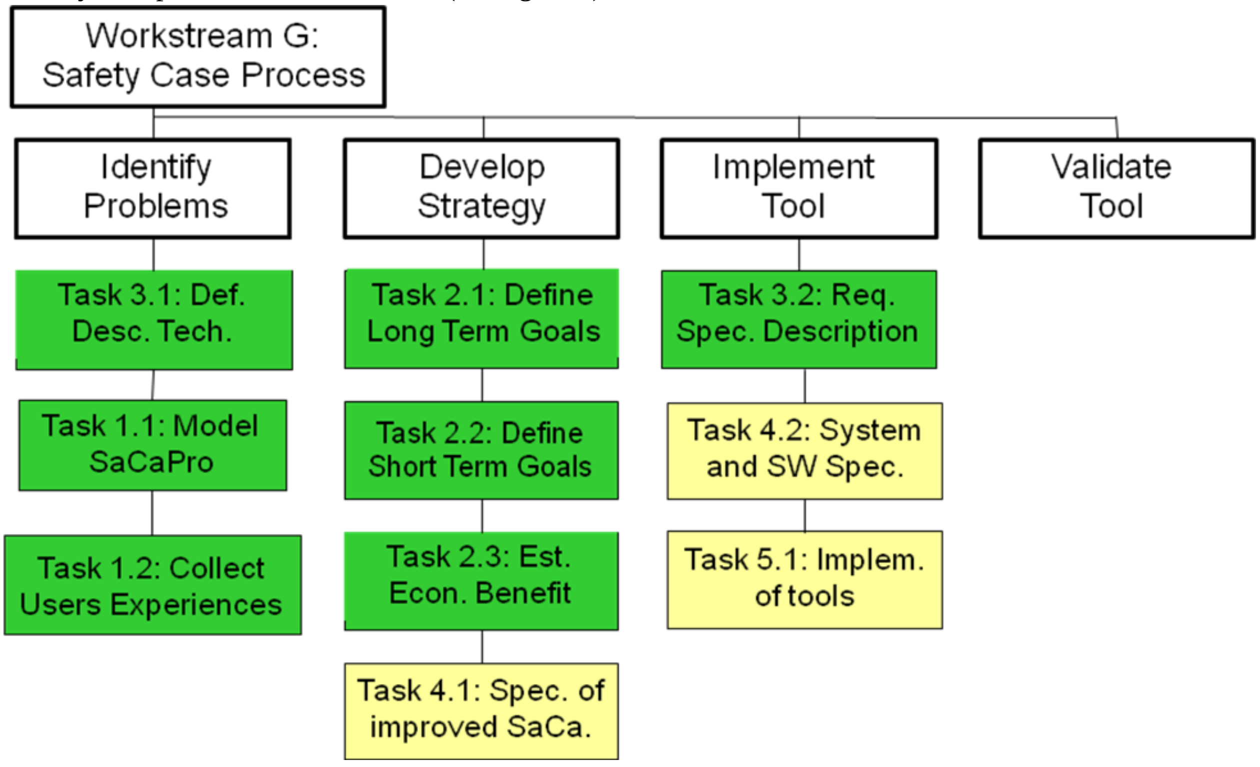

Workstream G: Safety Case Process
Identify Problems
Develop Strategy
Implement Tool
Validate Tool
Task 3.1: Def. Desc. Tech.
Task 1.1: Model SaCaPro
Task 1.2: Collect Users Experiences
Task 2.1: Define Long Term Goals
Task 2.2: Define Short Term Goals
Task 2.3: Est. Econ. Benefit
Task 4.1: Spec. of improved SaCa.
Task 3.2: Req. Spec. Description
Task 4.2: System and SW Spec.
Task 5.1: Implem. of tools

Figure 1: One can specify four phases to achieve the aim of Workstream G

The tasks in green have already been finalized. The yellow ones are currently under development..

The aim of task G.3.2

In this task description means and tools to describe the requirements for the supportive tool to be developed in WS G have to be chosen. Within the context of this tasks the term “description means” encompasses “description language” and “description method”.

The following requirements could be identified in various discussions.

- The language(s) to be chosen shall facilitate the generation of unambiguous but readable requirements.
- The method(s) to be chosen shall support the modelling of the identified functions in an appropriate manner.
- The tool(s) to be chosen shall support the language and method and shall be industriy proven.

Please note: It is not the aim of G.3.2 to specify the software requirements. This is to be done in task G.4.2 “System and Software Specification”. Here, in G.3.2, it is the aim to specify the languages, methods and tools to describe the software requirements.

Section 2 – Requirements specification Description

2.1 Introduction

The specification of software (SW) can roughly be diveded into the specification of the architecture of the SW and its inteded behaviour. These two aspects can be specified non-formally with natural language as well as on the basis of (semi-) formal specification languages. Based on this, appropriate methods and tools have to be chosen to support the specification of the corresponding aspect of the SW. These relations are depicted in the two-dimensional Figure 2.

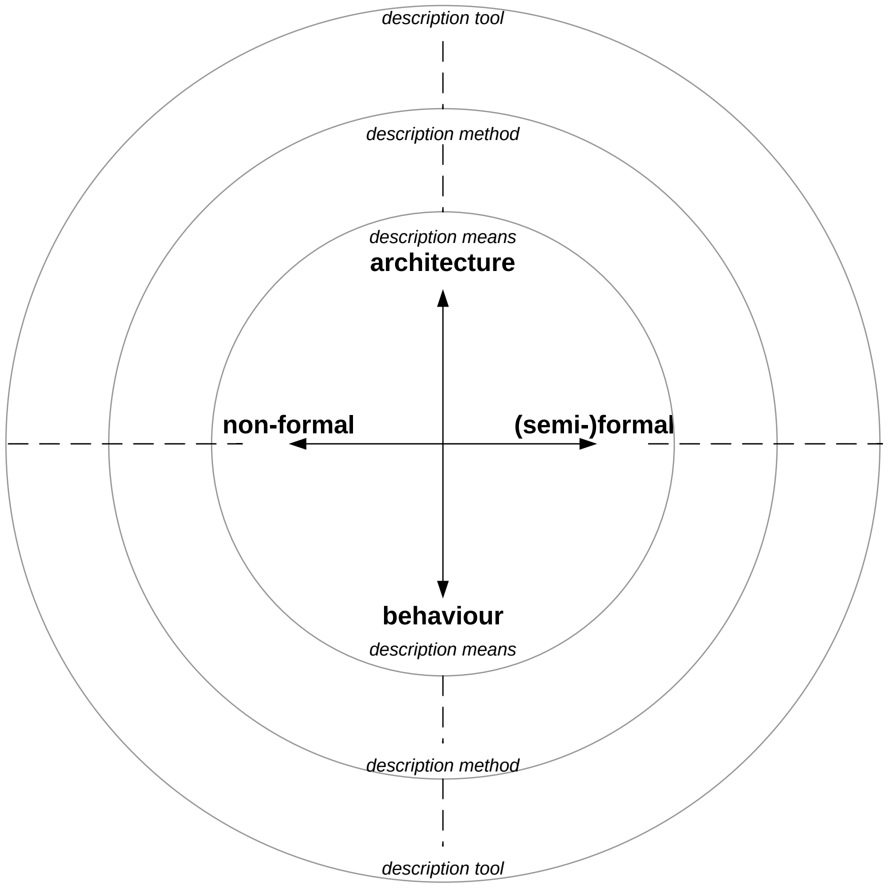


Figure 2: Relations between architecture, behaviour, formalization and language, method and tool

There exist dozens of non-, semi- and formal languages and methods to specify the different aspects of the requirements of a software tool. Figure 3 gives an overview of the chosen languages, methods and tools for the various specification tasks. One can see, that the following has been chosen:

Pure language: Natural language

Pure method: Structuring of natural language

As mixture of language and method:

- Use cases
- Class diagrams

Supporting tools: MS Word, MS Visio, MS Excel and Enterprice Architect.

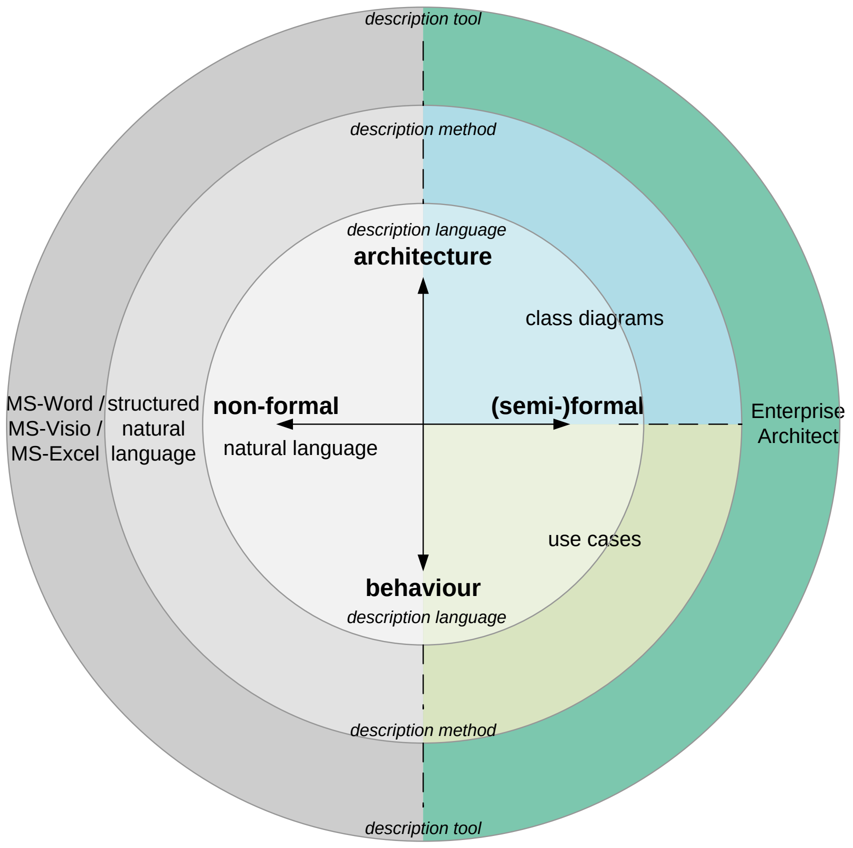


Figure 3: Different modeling languages, methods and tools for different tasks

Please note: The identified description means and tools have been identified as beeing most probable the best to be chosen. It may turn out during the specification or implementation tasks, that further approaches or tools may be usefull. Agaist this background, it is intended to use the identified means, but changes may be appropriate during the projects lifetime.

The following sections of this chapter describe the relation of this task to task G.3.1 (section 2.2), examine and justify the chosen description languages and methods (section 2.3) and the chosen tools to support the specification (section 2.4).

2.2 The relation of deliverable G.3.2 to deliverable G.3.1

In deliverable G.3.1 “Definition of process description technology” a set of modelling languages and methods has already been examined. This had to be done against the following background:

- In G.3.1, it was the aim to almost exclusively model processes (i.e. the processes in the norms EN 5012x)
- An elaborated description of these processes in natural language already existed.
- The processes were, at least to a remarkable exted, already known by practitioners.

That means, the description of the processes had to be formalized in order to have an exact and unambiguous basis for the discussions with the partners.

In task G.3.2 the background is quite different and can be described as follows:

- Mainly functions, not processes, are to be specified.
- There are no descriptions of these funtions – they have to be specified through discussions.
- The specified functions have to be implemented.

- A tool to support the various phases from rough descriptions, to specification and finally to impelemtation is necessary.

This explains, why task G.3.1 and G.3.2 come to different (sets of) modelling languages, methods and tools as their resuls. The background and the aim determine the description means. And as the background as well as the aim is different, the results are different, too.

2.3 Description means: Languages and methods

To obtain a useful requirement specification, the specifications have on the one hand to be unambiguous, consistent and testable and on the other hand they have to be easily understandable and readable at least by the programmer of the software. The contradiction between readablity and unambiguousness of description languages is outlined in the following Figure 4. That means, although it is possible to write such requirements in natural language, it is often useful, if not necessary to use a structured or formal approach that leaves less room for ambiguouties and leads to well understandable specifications. On the other hand, these approaches often use a specific type of description means that has to be understood by the programmer. In addition, a requirement that has during discussions been identified as indispensable is the possibility to refine a specification in a step-by-step approach. Therefore, the “depth of the specification” will rise successively with the progress of the SW-project.

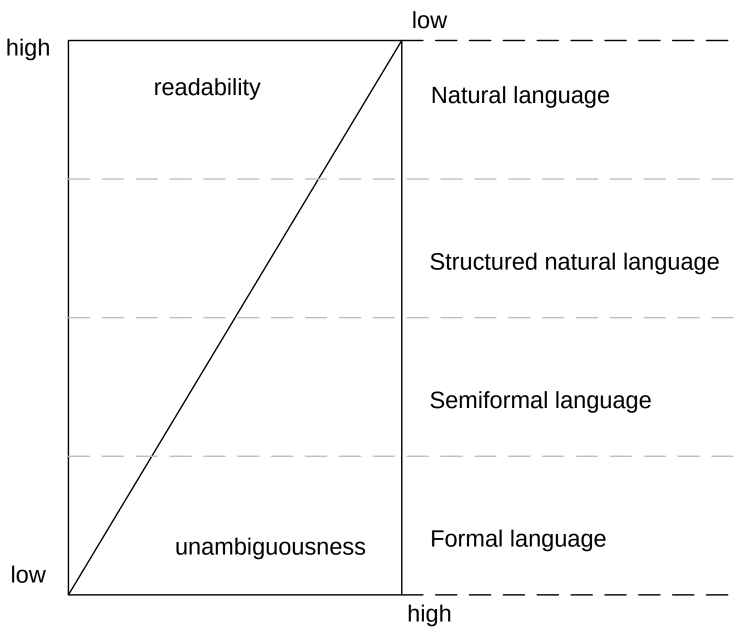


Figure 4: Description language – the relation between readablity and unambiguousness

Concerning the relation between description languages and description methods, one can state that it is often not possible to allocate a description mean to only one of these two sets. As an example from the reliablity field, one might think of the well known fault trees: On the one hand, the various gates and how to connect them, correspond to the language of fault trees. The way of creating a fault tree, e.g. starting from a “top event” and examining the conditions on a lower level that lead to this top level event, corresponds to the method of fault trees.

Therefore, desription means are often a mixture of description languages and description methods – see Figure 4.

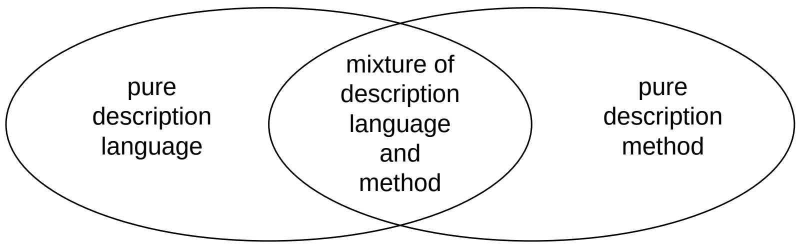

pure description language
mixture of description language and method
pure description method

Figure 4: Many description means are a mixture of language and method

The following subsections discuss the chosen description languages and methods and the reasons for our choice. Section 2.4 discusses the mentioned tools.

2.3.1 Non-formal description means: Natural Language and structured natural language

In this subsection, natural language as a "pure" language and the structuring of natural language as a method is being discussed:

2.3.1.1 Natural language as a specification language

It has been agreed, that the use of natural language is indispensable, especially but not only in early phases of the project to generate a common understanding of the project goals. Natural language can easily be understood by every stakeholder. Therefore, it follows that natural language is used to write the software requirements on a quite high level, where fundamental discussions are still possible and maybe even required. Further information concerning symbolism, normative basis etc. can be found in D.G.3.1, section 3.3.3.

A list of preliminary requirements specified in natural language derived from interviews and through discussion on the 2nd workshop can be found in D.G.4.2.

2.3.1.2 The structuring of natural language as a method to get more unambiguous specifications

To write requirements that are more unambiguous and still easy to understand, codes of best practice exist. These codes state that a restricted but unambiguous and clearly defined set of words and phrases have to be used, when specifying software in natural language. In Table 1 the recommended words and phrases are listed.

This table is a modified cutaway. The origin was presented at the April 1998 Software Technology Conference presentation "Doing Requirements Right the First Time."

Table 1 Quality measures related to individual SRS statements

Imperatives: Words and phrases that command the presence of some feature, function, or deliverable. They are listed below in decreasing order of strength.	
Shall	Used to dictate the provision of a functional capability.
Should	Used to describe the intention to provide a functional capability. Therefore, should is much weaker than shall.
Is required to	Used as an imperative in SRS statements when written in passive voice.
Must or must not	Most often used to establish performance requirement or constraints.

Are applicable	Used to include, by reference, standards, or other documentation as an addition to the requirement being specified.
Responsible for	Used as an imperative in SRSs that are written for systems with pre-defined architectures.
Will	Used to cite things that the operational or development environment is to provide to the capability being specified. For example, The vehicle's exhaust system will power the ABC widget (ABC is not an abbreviation but only a placeholder).
Continuances: Phrases that follow an imperative and introduce the specification of requirements at a lower level. There is a correlation with the frequency of use of *continuances* and SRS organization and structure, up to a point. Excessive use of *continuances* often indicates a very complex, detailed SRS. Use *continuances* in your SRSs, but balance the frequency with the appropriate level of detail called for in the SRS. **below, as follows, following, listed, in particular, support.**	
Directives: Categories of words and phrases that indicate illustrative information within the SRS. A high ratio of total number of *directives* to total text line count appears to correlate with how precisely requirements are specified within the SRS. Incorporate the use of *directives* in your SRSs. **Figure, Table, For example, Note**	
Options: A category of words that provide latitude in satisfying the SRS statements that contain them. This category of words loosens the SRS, reduces the client's control over the final product, and allows for possible cost and schedule risks. You should avoid using them in your SRS. **Can**, **May**, **Optionally**	

2.3.2 Semi-formal description means: UML class-diagrams and use-cases

As soon as a common but only rough understanding has been achieved, the requirements are to be specified in more detail. Then, the need for exactness increases while the need for common understandability decreases. At this stage of the discussions between the client and the supplier the specification in natural-language is being refined in a step by step approach and finaly complemented by formal versions of the specifications. These specifications may even refer to the used SW-plattforms to implement the system.

It turned out in the discussions and during the workshops that UML, especially UML class diagramms and UML use-cases, are the most appropriate modelling means, when it comes to specify the requirements more formally. Especially the following reasons have been pointed out:

- UML is broadly and very well known in the SW-engineering community.
- There are plenty of tools available.
- Many partners already made very good experiences with UML and appropriate tools among the partners. So, one can say, UML is industry proven.
- UML is seen as a very good compromise between readabilty and unambigousness.

- The UML supports the designer as well as the developer of the SW-tool. I.e. UML is applicable through many phases of SW-engineering.

As has been mentioned in the introduction of Section 2.3 (see Figure 4), many description means can be seen as a mixture of language and method. This holds for UML use-cases, class-diagramms as well.

2.3.2.2 UML class-diagrams

A class in the context of SW engineering specifies a set of variables and methods (functions) that belong in some way to each other. In any other than a SW engineering context, classes specify types of general objects (e.g. the class “car” in contrast to a specific car).

Class diagrams rank among the most important UML diagrams. They are often used in order to get an overview of a system or to better depict the relations and dependencies with other diagram types, such as use-cases (see below). They show the static structure and relations of classes within a system. One gets a good overview of who (which class / classes) is related to whom (which class / classes) and what (the whole operations) they can do. The relations may be associations, aggregations or generalisations.

UML class-diagrams are used to describe the (static) structure of a system. The static structure of the existing tools used will not be described. Only modules or interfaces which glue the single components together might be worthwhile to be modeled as UML class-diagrams.

Further Information about UML class diagramms can be found in D.G.3.1.

2.3.2.1 UML use-cases

Use-cases are used in UML as aids to develop user-specific requirements, to represent user goals and wishes and to specify the required system behaviour [1], [2]. This way, the interaction of the system with its environment is specified. This type of representation allows a conscious decoupling of the design and behaviour of a system, and gives an overview concerning the way the actors initiate the use-case and who is involved in the use-case.

As large parts of the requirements will be met by the existing used Document Management System, a large number of corresponding use-cases is pre-defined by the existing system. For example, procedures for logging in/out, committing and extracting documents etc. will probably be used “as is”, as long as they serve the specified requirements.

Further Information about UML use-cases can be found in D.G.3.1.

2.4 Tools

In this section the supporting tools that will most probably be used for specifying the requirements to be met by the tool to support the safety case writer, are presented. According to Figure 3, these are MS-Word, MS-Visio and MS-Excel for the non-formal specification and Enterprise Architect for the semi-formal specification. In addition to this, the tool “Trend/Analyst” is presented, which may be used for requirements tracing.

In the context of workstream G it has been decided to not use DOORS for requirments-tracing. This has been done against the following background: The requirements that are to be traced in WS-G only relate to tasks G.4.2 “System and Software Specification” and G.5.1 “Implementation of tools”. In this context, only a limited amount of requirements are to be traced and these are obsolete after the implementation of the supporting tool. This is different a different situation compared to e.g. WS D: There, functional requirements are to be managed and these are part of the majour goal of INESS. In addition, in WS D the number of requirements is significantly higher than that in WS G.

2.4.1 Tools to support the non-formal specification

Concerning the tool support for the non-formal specification many commonly used and industry proven tools are available. It has been decided on workshops and in various discussions that for this task Microsoft tools will suite very good:

- The specifications in the (structured) natural language can e.g. be accomplished very well by MS-Word.
- In addition the use of a commonly known graphic tool (e.g. MS-Visio) is intended to describe various relations and for clarification of any issue on a high level, i.e. in the first stages of SW engineering.
- MS-Excel may alos support the handling of information in "semi-structured" way.

Therefore, these tools seem to be appropriate to support the non-formal specification of the tool and it's behaviour.

2.4.2 The tool to define and trace requirements in a semi-formal way

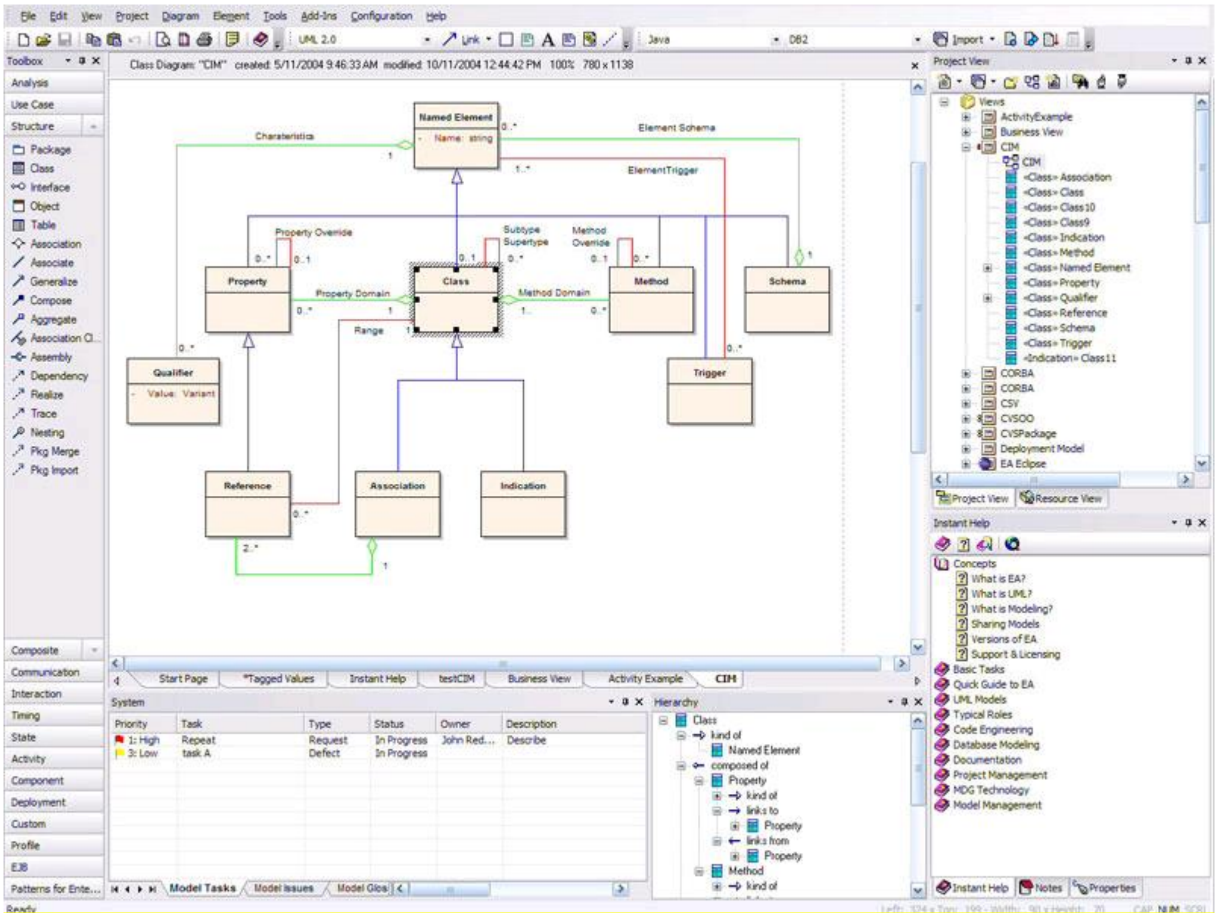
During discussions on the workshop, it has been decided to use the tool "Enterprise Architect" (see [3]) to support the semi-formal specification of the tool to be developed. The following reasons led to this decision:

Enterprise Architect (EA) is commonly known in the field of SW-engineering and is being widely used. Licenses have been sold to 60 countries up to now (March 2010). Furthermore, with a price of a few hundred Euros, the tool is very affordable. In addition, the tool is industry-proven: E.g. Funkwerk-IT has made very good experiences with this tool in the field of SW-engineering.

Specific characteristics are the enabling of:

- Modelling and managing of complex information: EA supports single persons, groups or huge organisations in modelling and managing of complex information-systems.
- Modelling, managing and tracing of requirements: EA supports the compilation ov requirements and their allocation to design requirements.
- UML-based designing and implementation of systems: Based on the standard UML 2.1, it allows to design and to document SW-systems.
- The visualisation and inspection of complex systems: E.g. it allows reverse engineering source code to understand and reveal the structure of the implementation. This especially is impartant, as we intend to use and adapt open source SW.
- The support of a system's life-cycle: From design-, implementation-, test- and maintenance-details, all information is tracable.
- The generation of plattform-independent models by model-driven-architecture: Model-driven-architecture (MDA) is a standard to translate plattform independent models with the help of a transformation into various plattform-dependent models. This again is very suitable for our tasks, as various partners use different plattforms for the company-specific tools.

Against this background, during the WS-G's workshop the participants agreed to use Enterprise Architect that has been proposed by Funkwerk IT; see Figure 6 for a screenshot of Enterprise Architect.



System

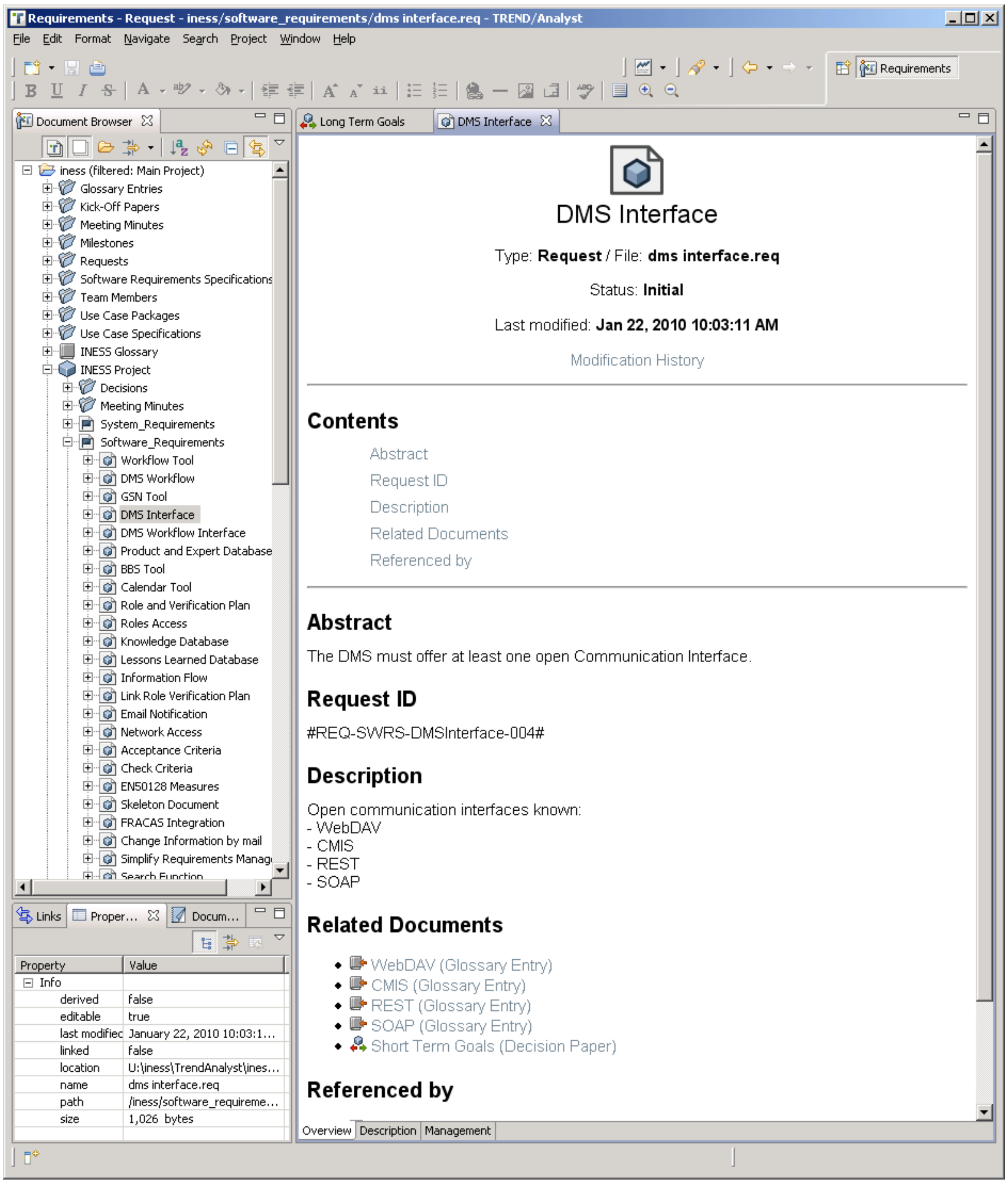
Priority	Task	Type	Status	Owner	Description
1: High	Repeat	Request	In Progress	John Red...	Describe
3: Low	task A	Defect	In Progress		

Figure 6: Screenshot of Enterprise Architect (here: UML-class diagrams)

2.4.3 Requirements tracing with Trend/Analyst

Besides using Enterprise Architect, the open-source tool Trend/Analyst (TA) [4] may be used for the management of requirements.

Besides the common tracking and tracing functionalities TA provides a built-in glossary function, which further helps to clarify communication between system design and software implementation. A screenshot of the used open-source tool can be seen in figure 7.


Requirements - Request - iness/software_requirements/dms interface.req - TREND/Analyst

File Edit Format Navigate Search Project Window Help

Requirements

Document Browser

- iness (filtered: Main Project)
  - Glossary Entries
  - Kick-Off Papers
  - Meeting Minutes
  - Milestones
  - Requests
  - Software Requirements Specifications
  - Team Members
  - Use Case Packages
  - Use Case Specifications
  - INESS Glossary
  - INESS Project
    - Decisions
    - Meeting Minutes
    - System_Requirements
    - Software_Requirements
      - Workflow Tool
      - DMS Workflow
      - GSN Tool
      - DMS Interface
      - DMS Workflow Interface
      - Product and Expert Database
      - BBS Tool
      - Calendar Tool
      - Role and Verification Plan
      - Roles Access
      - Knowledge Database
      - Lessons Learned Database
      - Information Flow
      - Link Role Verification Plan
      - Email Notification
      - Network Access
      - Acceptance Criteria
      - Check Criteria
      - EN50128 Measures
      - Skeleton Document
      - FRACAS Integration
      - Change Information by mail
      - Simplify Requirements Manag
      - Search Function

Links | Proper... | Docum...

Property	Value
Info	
derived	false
editable	true
last modifiec	January 22, 2010 10:03:1...
linked	false
location	U:\iness\TrendAnalyst\ines...
name	dms interface.req
path	/iness/software_requireme...
size	1,026 bytes

Long Term Goals | DMS Interface

DMS Interface

Type: **Request** / File: **dms interface.req**

Status: **Initial**

Last modified: **Jan 22, 2010 10:03:11 AM**

Modification History

Contents

Abstract
Request ID
Description
Related Documents
Referenced by

Abstract

The DMS must offer at least one open Communication Interface.

Request ID

#REQ-SWRS-DMSInterface-004#

Description

Open communication interfaces known:
- WebDAV
- CMIS
- REST
- SOAP

Related Documents

- WebDAV (Glossary Entry)
- CMIS (Glossary Entry)
- REST (Glossary Entry)
- SOAP (Glossary Entry)
- Short Term Goals (Decision Paper)

Referenced by

Overview | Description | Management

Figure 6: TREND/Analyst – Requirements Modeling Tool

Section 4 – CONCLUSIONS

It has been decided to successively refine the specification: starting from descriptions in natural language and with non-formal figures, and ending in a semi-formal specification with UML. Against this background, the appropriate tools to support the specification tasks have been identified, i.e. Microsoft products for the non-formal ("high level") specifications and Enterprise Architekt to suport the formal specification up to the automated generation of the bodies of the functions and classes.

The identified languages and methods as well as the chosen tools support several of levels specification of the tools architecture as well as the specification of its behaviour. Moreover, all the description means and tools are commonly known and industry proven, which was an additional reason for chosing them.

Therefore, one can conclude that the identified description languages and methods and the chosen toolset enable WS-G to specify the system and its behaviour (in tasks G.4.2) in an appropriate manner.

Section 5 – BIBLIOGRAPHY

[1] Chonoles M.J., Schardt, J. A.; UML 2 für Dummies, Wiley-VCH, 2003

[2] URL: http://www.sigs-datacom.de/sd/publications/os/1998/02/OBJEKTspektrum_UM_kompakt.htm, Last call: 01/2009

[3] Sparx Systems, Enterprise Architect, http://sparxsystems.eu/

[4] TREND/Analyst, Gebit Software, http://www.gebit.de/loesungen/ta_download.php