JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 9, Special Issue: TOOLS EUROPE 2007, October 2007

REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting[1]

Cristina Vicente-Chicote, Universidad Politécnica de Cartagena, Spain
Begoña Moros, Universidad de Murcia, Spain
Ambrosio Toval, Universidad de Murcia, Spain

Abstract

In order to integrate requirements into the current Model-Driven Engineering (MDE) approach, the traditional document-based requirements specification process should be changed into a requirements modelling process. To achieve this we propose a requirements metamodel called REMM (Requirements Engineering MetaModel) which includes the elements that should appear in a requirements model (requirements, stakeholders, test cases, etc.) together with the relationships that may appear between them. This metamodel is the basis of the REMM-Studio environment which enables: (1) to build graphical requirements models, (2) to validate them against the metamodel and against a set of additional OCL constraints, and (3) to automatically generate a navigable Software Requirements Specification (SRS) document as the main deliverable of the Requirements Engineering process. REMM-Studio is expected to ease the integration of requirements with other development models (e.g. component models) and to facilitate the validation of the SRS thanks to its navigability.

1 INTRODUCTION

The Software Engineering community has been paying attention to models as a corner stone of the software development process. Models are refined from one abstraction level to another by means of model transformation techniques with the aim of automating the development process as much as possible. In this context of Model-Driven Engineering (MDE) *"requirements must be modelled and we must have a continuity between requirements and final system implementation model. Thus the requirements traceability must be done of prime necessity at the model element level"* [Champeau'03]. In this vein,

[1] This work has been partially funded by the Spanish CICYT projects *MEDWSA* (TIN2006-15175-C05-02, Technical University of Cartagena) and *DEDALO* (TIN2006-15175-C05-03, University of Murcia), and by the Consejería de Educación y Ciencia de la Junta de Castilla-La Mancha project *DESERT* (PBC-05-012-3, University of Murcia).

Winkler advocates the integration of requirements into models [Winkler'06], and others, more specifically, the integration of textual requirements in a MDE approach [Conrad'04; Schätz'05]. The benefits of this integration have been proven in different application domains such as embedded systems [Conrad'04], web-enabled B2B applications [Marschall'03] and engineering systems [OMG'06].

As stated in [Weber'03], there is an urgent need for a requirements specification language (metamodel) which formally defines the concepts and relationships involved in the RE process. Such a metamodel could highly improve requirements reuse and could serve as a structured requirements reference model. The lack of such a reference model has been recognized by the INCOSE (INternational COuncil on Systems Engineering) that has started an initiative to collect information about good practice in RE. This initiative is being carried out by the INCOSE Requirements Working Group, and it is aimed at defining a Requirements "Book of Knowledge" [Dick'06].

For the RE process to be successful, it must be supported by a CASE or a CARE tool. Nowadays, commercial requirements specification and management tools (such as RequisitePro, CALIBER or DOORS) focus on textual requirements [INCOSE], while most of the tools supporting a MDE approach commonly leave textual requirements apart. The REMM-Studio integrated environment, developed as part of this work, gathers a set of model-based tools aimed at allowing requirements engineers to define a repository of reusable requirements catalogs. More precisely, REMM-Studio includes: (1) a graphical requirements modelling tool, (2) a model validation facility, and (3) an automatic model-to-text generator which produces a set of navigable documents from each requirements catalog.

The rest of this paper is organized as follows. First, the proposed *Requirements Engineering MetaModel* (REMM) is presented in section 2. Then, the REMM-Studio integrated environment and its tools are detailed in section 3. Section 4 summarizes some of the related approaches and, finally, section 5 presents the conclusions and future research lines.

2 REMM: THE PROPOSED REQUIREMENTS METAMODEL

Currently it is possible to find different requirements metamodels in the literature, each one including different concepts and relationships (see the *Related Work* section). This reveals the lack of consensus in the RE community and the absence of a requirements reference model.

The elements included in a requirements metamodel highly depend on the context it is used in [Dahlstedt'03]: release planning, requirements management, etc. Since our experience is in the context of requirements reuse [Toval'07], and more specifically in the reuse-based RE method called SIREN [Toval'02a], the concepts and relationships included in REMM, the Requirements Engineering MetaModel we introduce in this paper (see Figure 1), have been mostly taken from those described in SIREN, although we think they are applicable to a general RE approach.

The SIREN reuse-based RE method could be considered both a document-based and a repository-based approach since it revolves around a reusable repository of catalogs. Each of these catalogs contains a set of related requirements belonging to the same profile –e.g. security [Toval'02a], personal data protection [Toval'02b], etc.– or to the same domain –e.g. teleoperated systems [Nicolás'06]–. Requirements engineers may use this repository: 1) to improve the quality of the catalogs by adding new requirements or improving the existing ones, or 2) to reuse the existing requirements in their current projects.

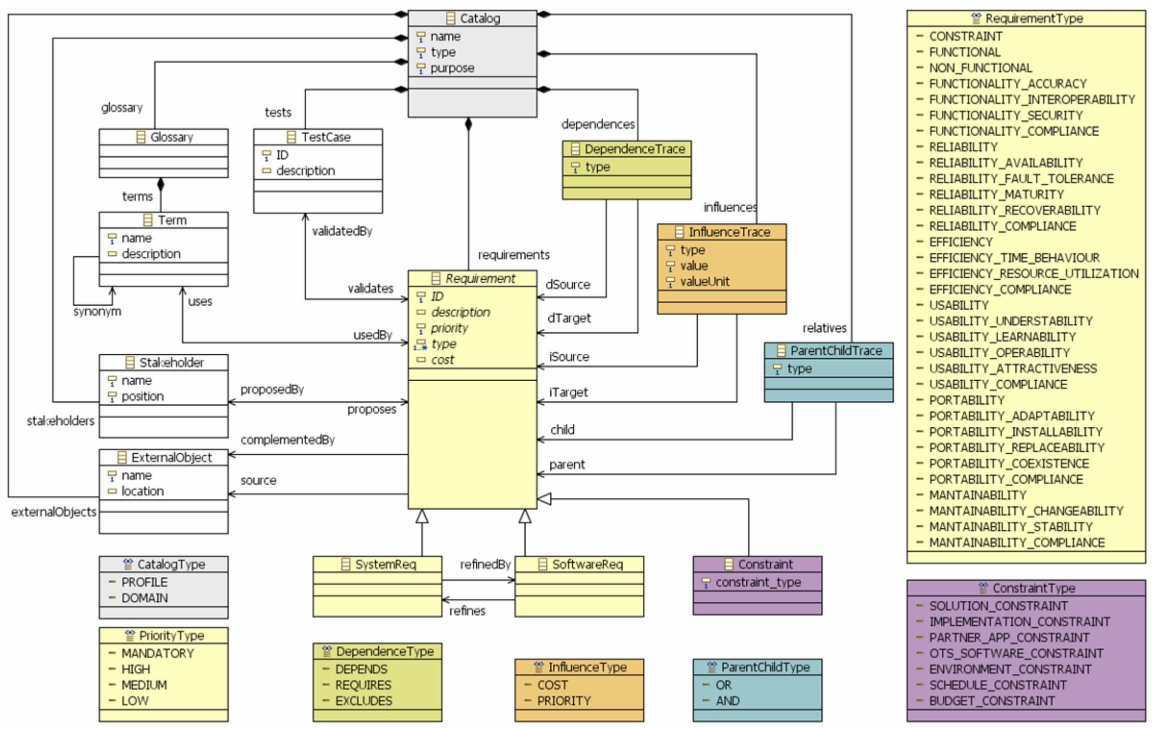


Figure 1: The proposed Requirements Engineering MetaModel (REMM)

The concepts and the semantic of the relationships included in REMM are briefly explained in the following subsections.

Concepts included in REMM

In REMM, all requirements specifications are stored in a `Catalog`, which is aimed at promoting requirements reuse. A catalog is characterized by a `name`, a `purpose` and a `type`. Two different types of catalogs can be defined in REMM according to the meaning proposed in SIREN [Toval'02a]: `DOMAIN` and `PROFILE`. A `Catalog` may contain three different kinds of requirements: (1) system requirements (`SystemReq`), that represent a need of the system, (2) software requirements (`SoftwareReq`), that represent how a system requirement is going to be carried out, and (3) constraints (`Constraint`), representing restrictions on the degrees of freedom we have in providing a solution.

All requirements are characterized by a unique identifier (`ID`), a textual description (`description`), a `type` (taking one or more of the values defined in the `RequirementsType` enumerated set), a `cost`, and a `priority` (taking values `MANDATORY`, `HIGH`, `MEDIUM` or `LOW`). Constraints have a fixed priority (`MANDATORY`) and type (`CONSTRAINT`) and they add a `constraintType` attribute which must take one of the values defined in the `ConstraintType` enumerated set. The values included in the `ConstraintType` set have been extracted from the VOLERE requirements specification template [Robertson'06], while those included in the `RequirementsType` set have been defined using the ISO/IEC 9126 [ISO/IEC'91] quality standard.

Normally, requirements are divided into functional ones and non-funcional ones. However, requirements engineers may find it useful to specify different kinds of requirements according to those features that commonly determine the final software system quality. The ISO/IEC 9126 standard defines six software quality categories (*Functionality*, *Reliability*, *Usability*, *Efficiency*, *Maintainability*, *Portability*), each one divided into several subcategories (see Figure 2). We have used this classification to define the `RequirementsType` set included in REMM.

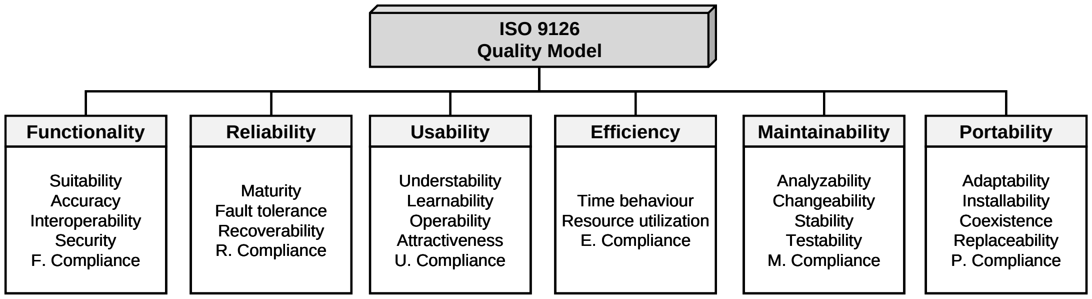


Figure 2: Categories and subcategories of the ISO 9126 quality model

It is worth noting that some of the ISO/IEC 9126 subcategories have not been included in REMM (e.g. *Suitability*, *Analyzability*, *Stability* and *Testability*) since we consider they are not requirement types (strictly skeaping), but the general principles that should guide the requirements specification and development processes. For instance, a requirement which specifies when the system must be *available* (e.g. "the system must be *available* at least from 8 AM to 8 PM") is quite common, but it makes no sense specifying when the system must be *suitable* or *stable* (all systems must be designed to always exhibit these properties). Note also that, for the sake of flexibility, the requirements `type` attribute has been defined as a list (attribute with a `1..n` multiplicity). This mechanism allows designers to select a priorized list of categories for each requirement, since sometimes it is not easy to classify them under a unique group.

Each requirement is proposed by a `Stakeholder` and thus it is important to record some information about them (`name`, `position`), just in case requirements engineers need to contact them for further information about their requirements. Some requirements might have been extracted from a law, a standard, or an organization policy. To represent these external information sources we have included the `ExternalObject` concept in the metamodel. Additional information used to complete or to explain a requirement description is also considered an external object (e.g. textual, graphical or multimedia files). External objects are described by their `name` and `location` (how they can be accessed).

Finally, to support the requirements validation phase, it is important to check that requirements are consistent and not ambiguous. A `Glossary` of terms has been included in REMM to allow requirements engineers to check if the concepts related to a certain requirement have consistent definitions. The catalog must include all the relevant concepts (`terms`) and their `synonyms`, if any. To check that requirements are not ambiguous it is well-known the convenience of defining conceptual test cases (`TestCase`) in parallel to requirements specification. This enables detecting ambiguous requirements when it is not possible to define a test case for them. The level of abstraction associated to each test case depends on the requirement type. Usually, a system requirement should be linked to one or more acceptance tests, while a software requirement should be linked to one or more conceptual test cases. The current version of REMM provides quite a limited support to test case definition. However, as explained in the conclusions, we are working on an improved version of the metamodel which will enable to trace requirements to analysis and design artefacts and to automatically generate some test cases to validate the models produced during a MDE product development process.

Traceability relationships included in REMM

According to one of the most widely accepted definitions, "*requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)*" [Gotel'94].

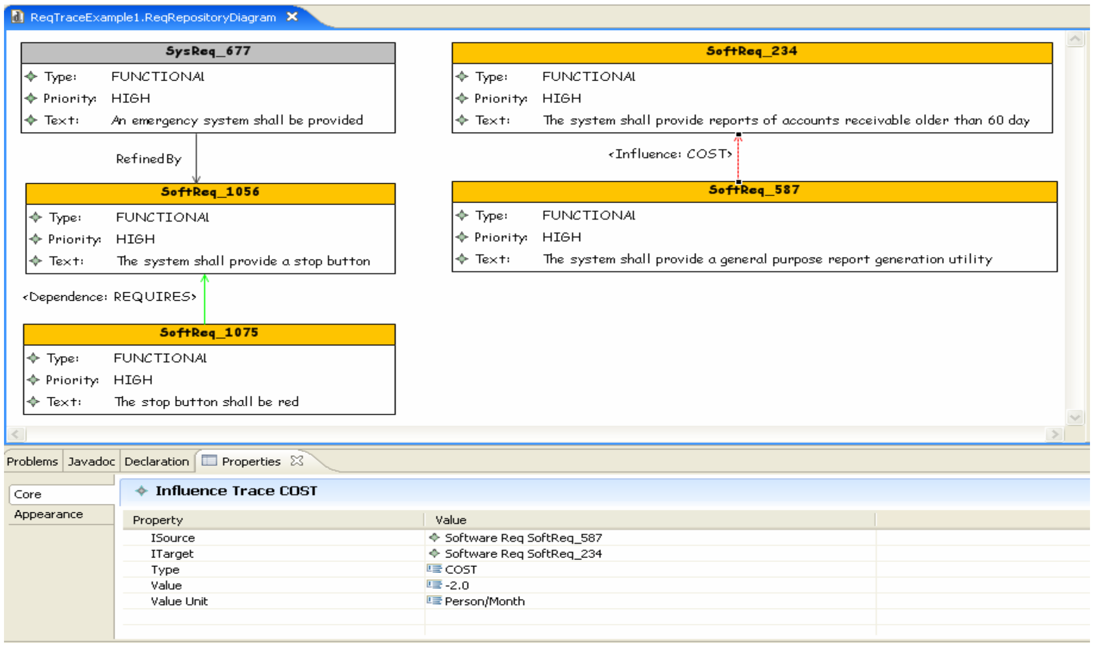
Pinheiro [Pinheiro'03] presents a taxonomy of traceability modes (note that this is different from a traceability reference model) that includes the following classification criteria:

- Requirements evolution: a requirement may be traced to aspects occurring before (Pre-RS traceability) or after (Post-RS traceability) its inclusion in the requirements specification.
- Types of the involved objects: a requirement may be traced to other requirements (inter-requirements traceability) or to other artefacts (extra-requirements traceability).
- The tracing direction: a requirement may be traced forward (to design or implementation components) or backward (to its source).

At the moment, the REMM metamodel covers: Pre-RS traceability, inter-requirements traceability and extra-requirements traceability to other IR artefacts (stakeholders, source, test cases, etc.). We have focused on the requirements traceability relationships needed to support a general RE process including: elicitation, negotiation, validation and documentation phases. We think this is the neccesary first step towards the definition of forward traces. Thus, the following traces have been included in REMM:

a) Inter-requirements traceability relationships

- One or more software requirements `refine` the information provided by a system requirement. This is the only relationship that enables the association between requirements defined at different abstraction levels (see examples in Figure 3).
- Given two requirements R1 and R2, belonging to the same requirement subclass (i.e. they are both system requirements, software requirements or constraints), the following dependences can be traced between them:
    - R1 `REQUIRES` R2. This relationship implies that R2 is needed to fulfil R1, i.e. R2 is a pre-condition for R1 (see example in Figure 3).
    - R1 `EXCLUDES` R2. This means that R1 and R2 are alternative and only one of them can be selected to appear in each requirements model.
    - R1 `INFLUENCES` R2. This relationship means that the inclusion of R1 in the requirements specification causes a change in the cost or in the priority of R2 (see example in Figure 3). For instance, the inclusion of R1 implies that the cost of R2 decreases in 2 (value) persons-months (valueUnit).
    - R1 `DEPENDS` R2. This means that there exists some kind of relationships between R1 and R2 that is neither requires, nor excludes, nor influences. It is just the way to explicitly show that R1 is related to R2.



SysReq_677
Type: FUNCTIONAL
Priority: HIGH
Text: An emergency system shall be provided

SoftReq_1056
Type: FUNCTIONAL
Priority: HIGH
Text: The system shall provide a stop button

SoftReq_1075
Type: FUNCTIONAL
Priority: HIGH
Text: The stop button shall be red

SoftReq_234
Type: FUNCTIONAL
Priority: HIGH
Text: The system shall provide reports of accounts receivable older than 60 day

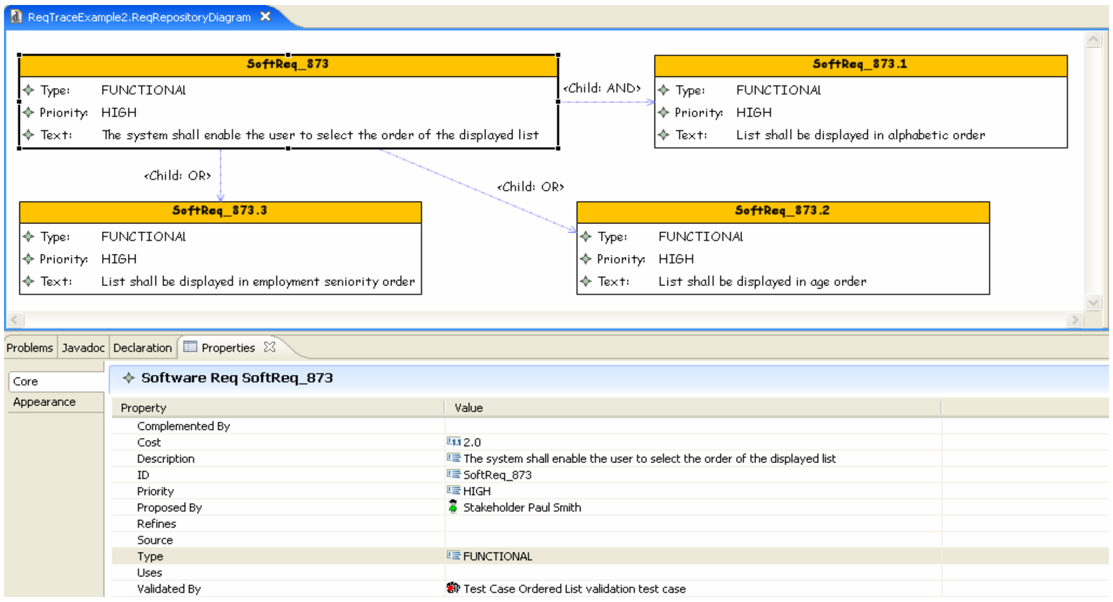
SoftReq_587
Type: FUNCTIONAL
Priority: HIGH
Text: The system shall provide a general purpose report generation utility

Influence Trace COST

Property	Value
ISource	Software Req SoftReq_587
ITarget	Software Req SoftReq_234
Type	COST
Value	-2.0
Value Unit	Person/Month

Figure 3: Inter-requirements relationships examples: *RefinedBy*, *Requires*, and *Influences*. The properties of the *Influences* relationship are shown in the tab below.

- Given a requirement R1 and two requirements R1.1 and R1.2, all of them belonging to the same Requirement subclass, where R1.1 and R1.2 are children of R1 (parent). This relationship means that R1.1 and R1.2 refine or extend the meaning of R1. Besides, the following parent-child traces can be created between them (see examples in Figure 4):
    - R1 `AND` R1.1, meaning that to fulfil R1, R1.1 has to be fulfilled too, i.e. the requirement R1.1 refines the specification of R1.
    - R1 `OR` R1.2, meaning that to fulfil R1, R1.2 could be fulfilled but it is not mandatory, i.e. R1.2 gives some alternative way (not exclusive) to fulfil R1 apart from R1.1.


ReqTraceExample2.ReqRepositoryDiagram
SoftReq_873
Type: FUNCTIONAL
Priority: HIGH
Text: The system shall enable the user to select the order of the displayed list
«Child: AND»
SoftReq_873.1
Type: FUNCTIONAL
Priority: HIGH
Text: List shall be displayed in alphabetic order
«Child: OR»
«Child: OR»
SoftReq_873.3
Type: FUNCTIONAL
Priority: HIGH
Text: List shall be displayed in employment seniority order
SoftReq_873.2
Type: FUNCTIONAL
Priority: HIGH
Text: List shall be displayed in age order
Problems
Javadoc
Declaration
Properties
Core
Appearance

Software Req SoftReq_873

Property	Value
Complemented By	
Cost	2.0
Description	The system shall enable the user to select the order of the displayed list
ID	SoftReq_873
Priority	HIGH
Proposed By	Stakeholder Paul Smith
Refines	
Source	
Type	FUNCTIONAL
Uses	
Validated By	Test Case Ordered List validation test case

Figure 4: *Parent-Child* relationships examples. A software requirement SoftReq_873 (parent) is selected and its properties are shown in the tab below.

b) Extra-requirements traceability relationships

- Each requirement is `proposedBy` a stakeholder.
- Each requirement may be `complementedBy` the information included in an ExternalObject (file, graphic, multimedia resource, etc.).
- Each requirement could have been extracted from an ExternalObject (law, standard, policies, etc.), that is its `source`.
- Both, system and software requirements, must be traced to one or more test cases (`validatedBy`), where it is explained how to check if they are fulfilled or not.

- The terms defined in the glossary might be `usedIn` some system or software requirements specification. These relationships allow requirements engineers to check if all the requirements using the same term are doing it consistently.

All the examples appearing in Figure 3 and Figure 4 have been extracted from [Davis'05] and they have been developed using the REMM-Studio graphical modelling tool that will be presented in the following section.

3 THE REMM-STUDIO INTEGRATED ENVIRONMENT

The Requirements Engineering MetaModel (REMM) discused in the previous section has been the basis on which REMM-Studio has been developed. The REMM-Studio integrated environment provides requirements engineers with three different tools: (1) a graphical requirements modelling tool, (2) a model validation tool, and (3) a tool for automatically generating navigable Software Requirements Specifications (SRS) from their models. The following sections briefly describe these three tools together with a brief discussion about why Eclipse has been the selected environment for implementing REMM-Studio.

Eclipse: the selected environment for implementing REMM-Studio

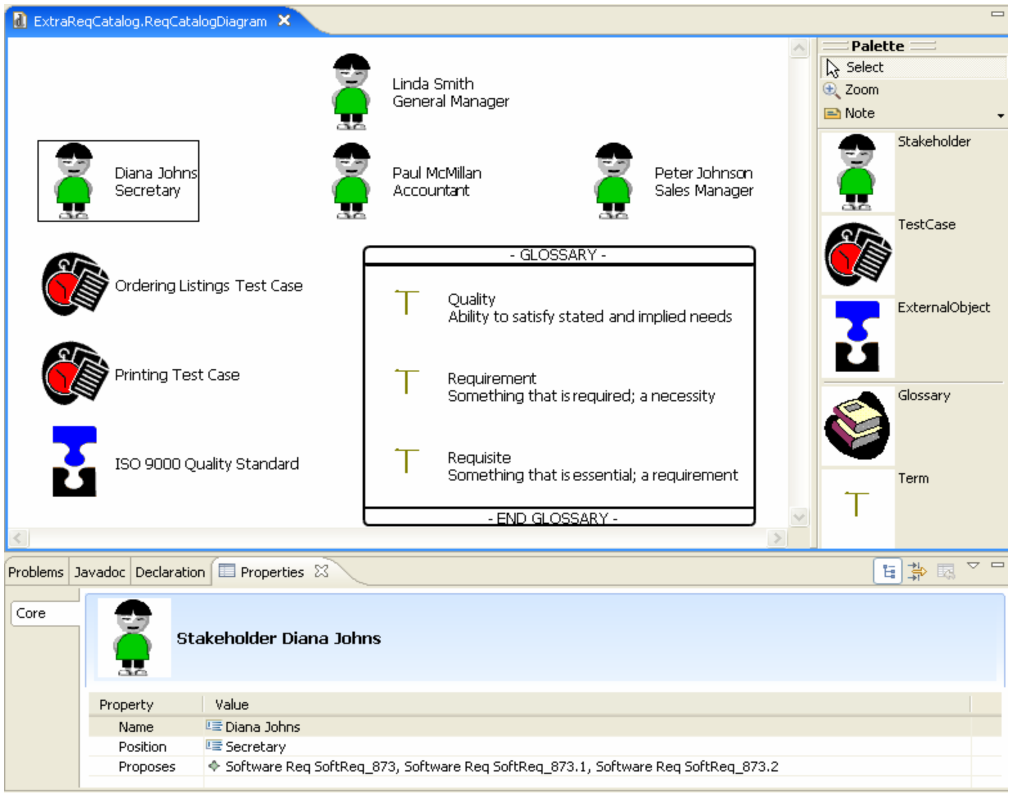
The Model-Driven Architecture initiative (MDA) is the OMG approach to MDE. Currently, the OMG offers a set of standards related to MDA which include MOF as the top level meta-meta-modelling language (language for describing metamodels, e.g. UML). Nowadays, the most widely used implementation of MOF is provided as an Eclipse plug-in called *Eclipse Modelling Framework* (EMF) [Budinski'03]. Although EMF currently supports only a subset of MOF, called EMOF (*Essential MOF*), it allows designers to create, manipulate and store both models and metamodels. This is the reason why many MDE-related initiatives are currently being developed around Eclipse and EMF, e.g. the *Graphical Modelling Framework* (GMF, http://www.eclipse.org/gmf/), or the *Generative Modelling Technologies* project (GMT, http://www.eclipse.org/gmt/).

Thus, the widespread use of Eclipse in the MDE community, together with the fact that it is a free, open-source, and platform-independent environment, led us to choose Eclipse as the development environment for implementing REMM-Studio.

The REMM-Studio graphical modelling tools

For any metamodel to be useful there must be a modelling tool that supports it. This tool must allow users to build textual or graphical models which conform with the metamodel. This section presents the two graphical modelling tools implemented as part of the REMM-Studio environment, both of them based on the REMM metamodel.

For the sake of simplicity, each tool has been designed to support only part of the metamodel, thus considerably simplifying the resulting diagrams. On the one hand, the *RequirementsTool* enables depicting requirements (system and software requirements and constraints) and the relationships existing between them, i.e: dependence, influence, and parent-child traces (see Figure 3 and Figure 4). On the other hand, the *CatalogTool* allows depicting the rest of the elements included in the metamodel, i.e: stakeholders, test cases, external objects, and the terms included in the glossary (see Figure 5).



Property	Value
Name	Diana Johns
Position	Secretary
Proposes	Software Req SoftReq_873, Software Req SoftReq_873.1, Software Req SoftReq_873.2

Figure 5: A graphical requirements model depicted using the *CatalogTool*.

All the models generated using any of these tools can be loaded and used from other models. For instance, if we want to specify the requirements proposed by the stakeholder Diana Johns (see Figure 5) we just need to load the model file (or files) where her requirements were defined. Then, all we need is to select the “proposes” attribute of this stakeholder and choose those requirements proposed by her from the list of all the loaded requirements. This information is automatically updated in the corresponding requirements models (i.e. the proposedBy field of the selected requirements will be set to the stakeholder Diana Johns), thus assuring inter-model consistency.

The two graphical modelling tools included in REMM-Studio have been developed using the Eclipse Graphical Modelling Framework (GMF), which is aimed at developing graphical model editors from any EMF metamodel. This plug-in allows designers:

1. To create a graphical representation for each domain concept appearing in the metamodel, e.g. a small person icon to represent stakeholders (see Figure 5).
2. To define a tool palette for creating and adding these graphical concepts to their models (see, for instance, the palette defined in the *CatalogTool* in Figure 5).

3. And finally, to define a mapping between all the previous artefacts, i.e. the metamodel elements, their graphical representations, and the corresponding creation tools.

The following section describes how users can validate the models created using these two graphical modelling tools within REMM-Studio.

The REMM-Studio model validation tool

EMF-based metamodels describe the syntax of a certain modelling language, i.e. its elements and the rules defining how they can be used together. However, some syntactically correct models may have no sense at all. For instance, according to REMM (see Figure 1), it is possible to create an `INFLUENCE` trace from a software requirement to a constraint and, obviously, this should not be allowed.

Currently, only the cardinality of EMF attributes and associations can be somehow restricted (setting the proper upper and lower bounds), but there is no mean to include any further restrictions or any semantic definition into a EMF metamodel. However, the Eclipse GMF plug-in, used to implement the graphical modelling tools described in the previous section, allows designers to define what should be considered a correct model and what should not, according not only to the metamodel, but also to some additional constraints defined using the OCL (*Object Constraint Language*) OMG standard. The model validation tool, included in REMM-Studio, has been implemented using this GMF facility. Table 1 presents some of the OCL constraints defined in the REMM-Studio validation tool.

Description	All stakeholders must have different names.
Domain target element	*Stakeholder*
OCL constraint	`self.owner.stakeholders->one(s|s.name = self.name)`
Description	Only one DepenceTrace can link two requirements.
Domain target element	*DependenceTrace*
OCL constraint	`self.dSource.owner.dependences -> one (d	(d.dSource = self.dSource) and (d.dTarget = self.dTarget))`

Table 1. Some of the OCL constraints defined in the REMM-Studio validation tool.

Graphical models depicted and validated using REMM-Studio may serve as a nice complement to a requirements specification document. However, as stated in [Hoffmann'04], "*a RE management tool is not worth without powerful document generation capabilities*". This question is addressed in the following section where the REMM-Studio tool for automatically generating a navigable textual documentation from previously defined requirements models is presented.

The REMM-Studio document generator

As stated in the introduction, REMM-Studio is aimed at integrating RE into a MDE approach. However, it must not be forgotten that one of the main deliverables expected from a RE process, is a textual requirements specification document.

In order to add a document generation facility to the REMM-Studio environment, a new tool has been implemented which defines a Model-to-Text (M2T) transformation for automatically generating a set of html documents from previously depicted and validated requirements models. This tool has been implemented using the Eclipse MOFScript plug-in (http://www.eclipse.org/gmt/mofscript/), aimed at defining M2T transformations for MOF-based models. MOFScript includes, among others, the following facilities: model checking, parsing and querying, output file management, etc.

The document generation tool implemented as part of the REMM-Studio environment generates, among others, the following files: a SRS document based on the IEEE-830 [IEEE'99a] standard template (slightly modified as described later), a file containing all the requirements grouped attending to their type (constraints, system requirements and software requirements) or to the stakeholder who proposed them, a file containing the terms defined in the glossary, etc. Figure 6 illustrates the set of all the html files generated by the tool and the navigation relationships existing between them.

In order to define the M2T transformation from a set of REMM-based models to an IEEE-830 SRS html file, a mapping between the concepts included in REMM and the sections defined in the IEEE-830 standard template must be defined (see Table 2).

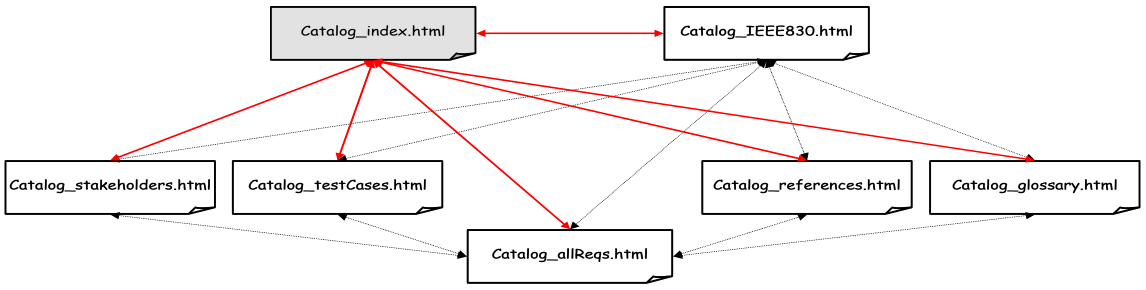


Figure 6. Set of HTML files automatically generated using the REMM-Studio document generator.

In order to define this mapping the following considerations had to be taken into account. Firstly, as stated in the IEEE-830 standard [IEEE'99a], the proposed SRS template is aimed at providing a "*specification for a particular software product, program, or set of programs that performs certain functions in a specific environment*". However, REMM-based models are aimed at defining reusable requirements catalogs, and not specific product requirements. Thus, some of the sections included in the IEEE-830 template (e.g. Section 2, regarding Product Overall Description) can not be filled in from the information stored in a REMM-based model (highlighted in red in Table 2).

IEEE-830 template	REMM elements
1. Introduction	
1.1 Purpose	Catalog purpose

1.2 Scope	***(empty, it does not apply)***
1.3 Definitions, acronyms, abbreviations	Glossary – Terms
1.4 References	External Objects
1.5 Overview	***(empty, it does not apply)***
2. Overall description	***(empty, it does not apply)***
3. Specific requirements	

3.1 External interfaces	ReqType = Functionality::Interoperability
	ReqType = Functionality::Accuracy

3.2 Functions	ReqType = FUNCTIONAL
3.3 Performance requirements	ReqType = Efficiency
3.4 Design constraints	Constraints
3.4.1 Standard compliance	ReqType = Functionality, … compliance
3.5 Software system attributes	
3.5.1 Reliability	ReqType = Reliability
3.5.2 Availability	ReqType = Reliability::Availability
3.5.3 Security	ReqType = Functionality::Security
3.5.4 Maintainability	ReqType = Maintainability
3.5.5 Portability	ReqType = Portability
3.5.6 Usability	**ReqType = Usability**
3.6 Other requirements	ReqType = NONFUNCTIONAL
3.7 Requirements interdependences	**Dependece, Influence, and ParentChild traces**

Table 2. Mapping between IEEE-830 and REMM.

Conversely, REMM-based models include some interesting information which does not fit in any of the sections defined in the IEEE-830 template (e.g. the different inter-requirements relationships defined in REMM). Thus, in order to include this information in the generated SRS document, we need to extend the template with some additional sections (highlighted in blue in Table 2).

Finally, it is worth noting that, although we have selected the IEEE-830 template for structuring the SRS document and html as the output file format, different M2T transformations can be similarly defined to obtain, for instance,

- A new SyRS (*System Requirements Specification*) document following the IEEE 1233 [IEEE'99b] standard recommendations.
- New SRS documents, structured according to different templates (e.g. VOLERE [Robertson'06]).
- An alternative print-friendly version of some files (e.g. pdf format).

4 RELATED WORK

The work reported in this paper overlaps with three areas of previous work: requirements metamodels, requirements modelling tools, and M2T transformations aimed at generating textual requirements specifications from models. Related work in each area is described in turn and the differences are highlighted to make the contribution.

Firstly, regarding requirements metamodelling, it is possible to find many different proposals available in the literature. Some of them are focused in requirements interdependences [Carlshamre'01; Dahlstedt'03; Davis'05], while others pay more attention to requirements traceability relationships [Ramesh'01; Letelier'02]. Some proposals tackle the implications of integrating requirements into an overall MDE approach [Champeau'03; Marschall'03; Schätz'05; Berre'06; Vogel'06; Winkler'06]. Among the last ones, it is worth mentioning the OMG SysML (*Systems Modelling Language*) standard [OMG'06] which, focused on the systems engineering application domain, explicitly includes requirements in its metamodel. All these proposals have contributed to the definition of REMM, which not only includes the most relevant features of all of them, but also incorporates some additional concepts and traces.

Regarding the tools currently supporting the integration of requirements models in a MDE approach, most of the commercial requirements management tools [INCOSE] focus on textual requirements (e.g. Requisite, CALIBER, DOORS) and do not provide any graphical modelling facility. Besides, those of these tools which enable the definition of traceability relationships from requirements to other analysis or design models (e.g. Requisite and Rational) do not provide an integrated environment for defining the different types of models.

On the other hand, most commercial tools supporting a MDE software development approach, only allow users to specify requirements in one of the following ways:

- Textually, for instance using a spreadsheet editor. This is the case of Softeam Objecteering (www.objecteering.com).
- Building a SysML requirements model, where designers can specify textual requirements within their models. This is the case of Artisan Reqtify (http://www.artisansw.com/products/reqtify.aspx) and Telelogic Rhapsody (http://www.ilogix.com/sublevel.aspx?id=53).

REMM-Studio has been designed trying to overcome some of the limitations exhibited by these tools. Actually, REMM-Studio provides a graphical modelling tool, it is not limited to any specific domain (e.g. the system engineering application domain), and it provides a rich set of extra- and inter-requirements traceability relationships.

Regarding the generation of textual requirements documents, most requirements management tools provide this capability, offering different output formats and styles. Consequently, we have focused our study on those tools providing transformations from models to textual requirements. In this vein, the *Objectiver tool* (www.objectiver.com), proposed in [Lamsweerde'04], includes a document generator capable of producing requirements documents (in RTF or PDF format) organized in sections and subsections according to a goal refinement graph and to a template which reflects some company specific standards. A different approach is presented in [Maiden'05], which proposes the application of patterns to *i** models in order to derive requirements statement, structured following the VOLERE shell. Finally, [Vogel'06] uses MOFScript for generating Open Document Format files as part of the Business Requirements Document.

The main drawback of the Objectiver tool is that it is only suitable for goal oriented RE projects, while the main limitation of the Maiden proposal is that requirements statements are not explicitly included in the models but automatically inferred from them. The MOFScript M2T transformation proposed in [Vogel'06] is similar to the one presented in this paper, but the tool supporting this transformation is not part of an integrated environment, but part of a tool chain (different tools support each step of the RE process).

5 CONCLUSIONS AND FUTURE WORK

The main benefits of the MDE approach to RE presented in this paper, can be summarized as follows:

- We have presented REMM as a structured requirements reference model which includes both the concepts and relationships involved in the RE process.
- REMM provides a common framework for requirements specification, thus promoting requirements reuse.
- We have presented the REMM-Studio integrated environment which supports the integration of RE into a MDE approach. REMM-Studio is aimed at helping requirements engineers: (a) to build graphical requirements models, (b) to validate them against REMM and against a set of additional OCL constraints, and (c) to automatically generate navigable Software Requirements Specification (SRS) documents from their models using the IEEE-830 template.

Currently, we are working on an extension of REMM which includes the "CurrentProjectRequirements" concept. This extension will allow us to implement a new graphical modelling tool aimed at defining the set of requirements belonging to a specific software development project. Using this new tool, requirements engineers will be able to select some of the requirements already defined in any of the reusable catalogs, and also to add newly detected project specific requirements. Two new M2T transformations are also under development for automatically generating (1) the IEEE-830 Software Requirements Specification (SRS) document, and (2) the IEEE-1233 System Requirements Specification (SyRS) document, both of them obtained from the new "CurrentProjectRequirements" models.

In the future, we plan to extend REMM in two additional directions: (1) to incorporate parameterized requirements in order to further improve the level of reuse, and (2) to add new forward traces to link requirements models to other analysis and design models (e.g. component models). This second extension will enable tracing requirements during the whole software development process. Furthermore, we expect that the integration of formal requirements models into a MDE process will considerably improve the analysis and design phases enabling, among other things, the (semi-) automatic generation of test cases to early validate models using model simulators.

Finally, regarding the INCOSE initiative for defining a RE "Book of Knowledge", we modestly believe that REMM can contribute as a starting point of discussion for defining a RE reference model. Thus, we are planning to submit our proposal to the INCOSE Requirements Working Group for their consideration.

REFERENCES

[Berre'06] Arne Jørgen Berre, et al., "COMET (Component and Model based Development Methodology)", 2006, http://modelbased.net/comet/

[Budinski'03] Frank Budinski, et al., *Eclipse Modeling Framework*, Addison-Wesley Professional, 2003.

[Carlshamre'01] Pär Carlshamre, et al., "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning", *Fifth International Symposium on Requirements Engineering (RE'01),* Toronto, Canada, 2001.

[Conrad'04] Mirko Conrad, Ines Fey and Kerstin Buhr, "Integration of Requirements into Model-based Development", *International Workshop on Automotive Requirements Engineering (AuRE'04) in conjuntion with RE'04*, Nanzan University, Nogoya, Japan, 2004.

[Champeau'03] Joel Champeau and Emmanuel Rochefort, "Model Engineering and Traceability", *UML 2003. SIVOES-MDA Workshop*, San Francisco. California, 2003.

[Dahlstedt'03] Asa G. Dahlstedt and Anne Persson, "Requirements Interdependencies- Moulding the State of Research into a Research Agenda", *Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '03) In connection with: CAiSE'03*, Klagenfurt/Velden, Austria, 2003.

[Davis'05] Alan Davis, *JERM: Just Enough Requirements Management*, Dorset House Publishing, 2005.

[Dick'06] Jeremy Dick, "Requirements Guide For All (REGAL): An INCOSE Initiative", *14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis/St. Paul, Minnesota, USA, 2006.

[Gotel'94] Orlena Gotel and Anthony Finkelstein, "An Analysis of the Requirements Traceability Problem", *1st International Conference on Requirements Engineering (ICRE'94)*, Colorado Springs, 1994.

[Hoffmann'04] Matthias Hoffmann, et al., "Requirements for Requirements Management Tools", *12th IEEE International Requirements Engineering Conference (RE'04)* Kyoto, Japan, 2004.

[IEEE'99a] IEEE, "IEEE Std 830, 1998 Edition, IEEE Recommended Practice for Software Requirements Specifications", *IEEE Standards Software Engineering. Volumen Four: Resource and Technique Standards*, The Institute of Electrical and Electronics Engineers, Inc., 1999a.

[IEEE'99b] IEEE, "IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications", *IEEE Standards Software Engineering. Volumen*

One: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc., 1999b.

[INCOSE] INCOSE, "Requirements Management Tools Survey", http://www.paper-review.com/tools/rms/read.php

[ISO/IEC'91] ISO/IEC, "ISO/IEC 9126: Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use - 1991", International Organization for Standardisation, 1991.

[Lamsweerde'04] Axel van Lamsweerde, "Goal-Oriented Requirements Enginering: A Roundtrip from Research to Practice", *12th IEEE International Requirements Engineering Conference (RE'04)* Kyoto, Japan, 2004.

[Letelier'02] Patricio Letelier, "A Framework for Requirements Traceability in UML-based projects", *1st International Workshop on Traceability in Emerging Forms of Software Engineering. (TEFSE'02)*, Edinburgh, U.K., 2002.

[Maiden'05] Neil A. M. Maiden, et al., "Generating requirements from systems models using patterns: a case study ", *Requirements Engineering Journal*, 10(4): 276-288 2005.

[Marschall'03] Frank Marschall and Maurice Schoenmakers, "Towards Model-Based Requirements Engineering for Web-Enabled B2B Applications ", *10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03)* Huntsville, AL, USA, 2003.

[Nicolás'06] Joaquín Nicolás, et al., "A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance", *Learning Software Organizations + Requirements Engineering (LSO+RE 2006)*, Hannover. Alemania, 2006.

[OMG'06] OMG, "OMG Systems Modeling Language (OMG SysML^{TM}) Specification", 2006, http://www.sysml.org/docs/specs/OMGSysML-FAS-06-05-04.pdf

[Pinheiro'03] Francisco A. C. Pinheiro, "Requirements Traceability", *Perspectives on Software Requirements (Kluwer Internation Series in Engineering and Computer Science, 753)*, Kluwer Academic Publishers, 2003.

[Ramesh'01] Balasubramaniam Ramesh and Matthias Jarke, "Towards Reference Models for Requirements Traceability", *IEEE Transaction on Software Engineering*, 27(1): 58-93, 2001.

[Robertson'06] James Robertson and Suzanne Robertson, "VOLERE Requirements Specification Template. Edition 11", 2006, http://systemsguild.com/GuildSite/Robs/Template.html

[Schätz'05] Bernhard Schätz, et al., "Model-Based Requirements Engineering with AutoRAID", *INFORMATIK 2005*, Rheinische Friedrich-Wilhelms-Universität Bonn, 2005.

[Toval'07] Ambrosio Toval, et al., "Eight key issues for an effective reuse-based requirements process", *International Journal of Computer Systems Science and Engineering*(accepted for publication), 2007.

[Toval'02a] Ambrosio Toval, et al., "Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach", *Requirements Engineering Journal. Springer*, 6(4): 205-219, 2002a.

[Toval'02b] Ambrosio Toval, Antonio Olmos and Mario Piattini, "Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection", *IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02)*, Essen, Alemania, 2002b.

[Vogel'06] Régis Vogel and Keith Mantell, "MDA adoption for a SME: evolution, not revolution - Phase II", *European Conference on Model Driven Architecture (ECMDA 2006)*, Bilbao, Spain, 2006.

[Weber'03] Matthias Weber and Joachim Weisbrod, "Requirements Engineering in Automotive Development: Experiences and Challenges", *IEEE Software*, 20(1): 16-24, 2003.

[Winkler'06] Stefan Winkler, "A modelling infraestructure for the integration of requirements artifacts", *RE 2006. Doctoral Symposium*, Minneapolis/St. Paul, Minnesota, USA, 2006.

About the authors

Cristina Vicente-Chicote is a Lecturer at the Technical University of Cartagena (Spain). She received a MSc in Computer Science from the University of Murcia (Spain) and a PhD in Telecomunications from the Technical University of Cartagena (Spain). Her current research interests include model-driven engineering, domain-specific language design, component-based software engineering, and software product lines. Contact her at Cristina.Vicente@upct.es.

Begoña Moros is a Lecturer at the Department of Computer Science, University of Murcia (Spain). She has a background in prototyping environment for UML, software development methods and requirements engineering. Her current research interest include model-driven engineering, requirements models and requirements traceability. She is developing her PhD in this field. Contact her at bmoros@um.es.

Ambrosio Toval is a full professor at the University of Murcia (Spain). He holds a BS in Mathematics from the Univ. Complutense of Madrid (Spain), and he received a PhD in Computer Science from the Technical Univ. of Valencia (Spain). His current research interesent include the design and implementation of conceptual UML model verification methods, requirements engineering processes, and computer-aided requirements engineering tools. Contact him at atoval@um.es.