A MODEL-DRIVEN ENGINEERING APPROACH TO REQUIREMENTS ENGINEERING
How These Disciplines May Benefit Each Other

Begoña Moros
Dpto. de Informática y Sistemas, Facultad de Informática, Universidad de Murcia (UMU), 30100 Murcia, Spain

Cristina Vicente-Chicote
División de Sistemas e Ingeniería Electrónica, Universidad Politécnica de Cartagena (UPCT), 30202 Cartagena, Spain

Ambrosio Toval
Dpto. de Informática y Sistemas, Facultad de Informática, Universidad de Murcia (UMU), 30100 Murcia, Spain

Keywords: Model-Driven Engineering (MDE), Requirements Engineering (RE), Requirements Meta-Model (REMM), Requirements Traceability

Abstract: The integration of Model Driven Engineering (MDE) principles into Requirements Engineering (RE) could be beneficial to both MDE approaches and RE. On the one hand, the definition of a requirements metamodel would allow requirements engineers to integrate all RE concepts in the same model and to know which elements are part of the RE process and how they are related. Besides, this requirement metamodel could be used as a common conceptual model for requirements management tools supporting the RE process. On the other hand, this requirements metamodel could be related to other metamodels describing analysis and design artefacts. This would align requirements to models and, as a consequence, requirements could be more easily integrated into the current MDE approach. To achieve this, the traditional RE process, focused on a document-based requirements specification, should be changed into a requirements modelling process. Thus, in this paper we propose a requirements modelling language (metamodel) aimed at easing the integration of requirements into a MDE approach. This metamodel, called REMM, is the basis of a requirements graphical modelling tool also implemented as part of this work. This tool allows requirements engineers to depict all the elements involved in the RE process and to trace relationships between them.

1 INTRODUCTION

It is well-known the importance of Requirements Engineering (RE) in the software development process. The most quoted study, the CHAOS Report conducted by the Standish Group, revealed that incomplete and changing requirements and specifications are the leading cause of software failures. Recent studies (Sommerville and Ransom, 2005; Damian and Chisan, 2006) have demonstrated the benefits of effective requirements process to software development organizations.

The RE process, as any other software development activity, requires supporting tools. Currently, commercial tools do not support the whole RE process, but only requirements management activities. A Requirements Management Tool (RMT), such as Requisite, Caliber, or DOORS (INCOSE, 2006), that stores information in a multi-user database, provides a robust solution to the restrictions of a plain text requirements specification (Wiegers, 2003). However, the conceptual models supported by most RMTs are rather simple since they commonly include only a general 'Requirement' concept and a 'sub-requirement' and 'associated requirement' relationships (Schätz et al., 2005). Conversely, a tool supporting the whole RE process, should be able to manage a lot of elements (i.e. stakeholders, external objects, different types of functional and non-functional requirements, etc.), and relationships between them. It is exactly in this point where Model Driven Engineering (MDE) can offer a new perspective to RE.

According to the MDE philosophy (Bézivin, 2005) models represent a particular view of the system and they are described in terms of formal metamodels. Thus, in order to describe a requirements view of a system a requirements metamodel must be defined. The definition of such a metamodel would be useful not only for integrating all the RE concepts in the same repository, but also for providing requirements engineers with a structured requirements reference model. As stated in (Weber and Weisbrod, 2003), the different types of requirements can become confusing and therefore, there is an urgent need for a requirements metamodel which formally defines the concepts and the relationships involved in the RE process.

On the other hand, the Software Engineering community has been paying attention to models as a cornerstone 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 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 and Rochefort, 2003).

Now that the need of defining a requirements metamodel has been justified, a metamodel description language to describe it must be selected. Three reasons make of MOF (*Meta-Object Facility*) (OMG, 2004) the most suitable candidate nowadays: (1) it is the OMG standard meta-metamodel (metamodel description language), (2) it is widely used by the MDE community, and (3) it is possible to find a stable, free, and open-source distribution of a reduced set of MOF provided by the Eclipse platform. Actually, the Eclipse community is one of the most active ones in MDE, as reflected in the increseangly growing number of Eclipse projects appearing around this new software development trend.

The proposed Requirements Meta-Model (REMM) is presented in the Section 2, and the graphical modelling tool implemented to support it is introduced in section 3. Then Section 4 outlines some related works and, finally, Section 5 presents the conclusions and future research lines.

2 REMM: THE PROPOSED REQUIREMENTS METAMODEL

The elements selected to be included in a requirements metamodel, greatly depend on the development context (Dahlstedt and Persson, 2003). Since our experience is in the field of requirements reuse (Toval et al., 2007) and, more specifically, in the reuse-based RE method called SIREN (Toval et al., 2002), the concepts and relationships included in the proposed requirements metamodel have been mostly taken from those used 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 repository-based approach, since it revolves around a reusable repository of catalogues. A catalogue contains a set of related requirements belonging to the same profile (Toval et al., 2002) —e.g. security or personal data protection— or domain —e.g. the tele-operated system domain (Nicolás et al., 2006). Requirements engineers may use the repository: 1) to reuse existing requirements in their current projects, or 2) to improve the quality of the catalogue by adding new requirements or improving the existing ones.

The proposed Requirements Meta-Model, called REMM, is shown in Figure 1. The concepts included in REMM are explained in section 2.1, while the semantics of the relationships defined between them is detailed in section 2.2.

2.1 Concepts Included in REMM

In REMM all requirements are stored in `Catalogs` (in order to promote reuse) characterized by a `name` and a `type`. Two different types of catalogs (`CatalogType`) can be defined in REMM, according to the meaning proposed in SIREN: `DOMAIN` and `PROFILE`. A catalog contains different types of Requirements, sharing the same set of attributes: a unique identifier (ID), a textual description (`description`), `cost`, `priority` (taking values `HIGH`, `MEDIUM` or `LOW`) and `type.` Currently, only two types of requirements are considered: `FUNCTIONAL` and `NON-FUNCTIONAL`, although we are working in a broader classification according to the quality standard ISO/IEC 9126.

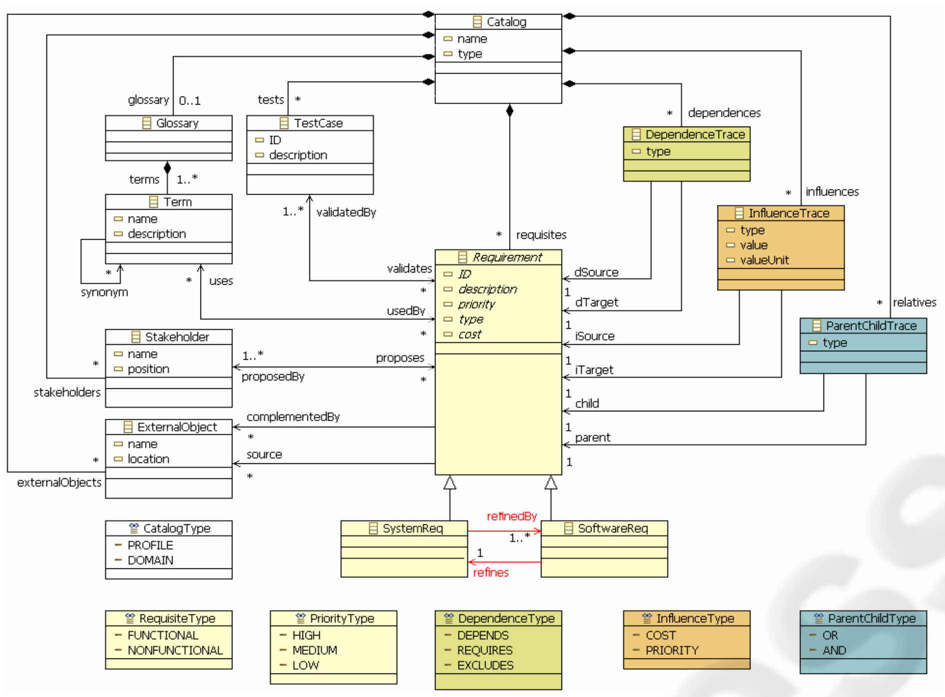


Figure 1: The proposed requirements metamodel (REMM).

Requirements are classified into system requirements (`SystemReq`) —representing a need of the system— and software requirements (`SoftwareReq`) —representing how a system requirement is going to be carried out.

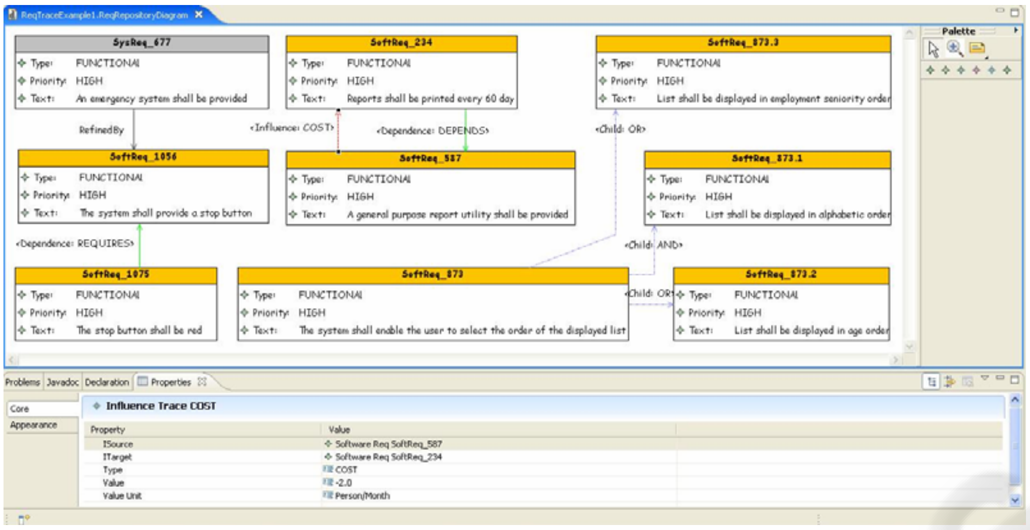
Each requirement is proposed by a `Stakeholder`, and thus it is important to record some information about them (`name`, `position`) just in case the requirements engineers need to contact them for further information about their proposed requirements. Requirements could come from a law, a standard, an organization policy or any other source. To represent these external information sources we have included the `ExternalObject` concept in the metamodel. Any file, multimedia resource, graphic, etc. used to complement or to explain the requirement description, are also considered external objects. Each external object is characterized by its `name` and `location` (the way it 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 glossary 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`), independent of the implementation, in parallel to requirements specification (Wiegers, 2003). This enables detecting ambiguous requirements when it is not possible to define a test case for them. The level of abstraction associated to a test-case depends on the requirement type. Usually, a system requirement is linked to one or more acceptance tests, while a software requirement is linked to one or more conceptual test cases.

2.2 Traceability Relationships Included in REMM

The taxonomy of traceability modes defined in (Pinheiro, 2003), 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).



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 2: Inter-requirements relationships examples. The properties of the `Influences` relationship are shown in the tab below.

At the moment, the REMM metamodel proposed in Figure 1 covers: pre-RS traceability, inter- and extra-requirements traceability. We have focused on requirements traceability relationships needed to support a general RE process including the elicitation, negotiation, validation and documentation phases. We think this is the necessary first step towards the definition of forward traces. The traces included in REMM are explained in the following sections.

2.2.1 Inter-Requirements Traceability

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 Figure 2).

Given two requirements named R1 and R2, belonging to the same requirement subclass, the following dependences can be traced between them:

- R1 `REQUIRES` R2, means that R2 is needed to fulfil R1, i.e. R2 is a precondition for R1 (see Figure 2).
- R1 `EXCLUDES` R2, means that R1 and R2 are alternative and only one of them could be selected to appear in each requirements model.
- R1 `INFLUENCES` R2, means that the inclusion of R1 in the requirements specification causes a change in the cost or in the priority of R2. For instance, in Figure 2, the inclusion of SoftReq_587 implies a cost reduction of 2 (value) person-months (valueUnit) in SoftReq_234.
- R1 `DEPENDS` R2, means that there exists some kind of relationship between them that is neither requires, nor excludes nor influences. It is just the way to explicitly show that R1 is related to R2.

Given a requirement R1 and two requirements R1.1 and R1.2, all of them belonging to the same requirement subclass, the following parent-child traces can be created between them (see Figure 2):

- R1 `AND` R1.1 means that to fulfil R1, R1.1 has to be fulfilled too. The requirement R1.1 refines the specification of R1.
- R1 `OR` R1.2 means that to fulfil R1, R1.2 could be fulfilled but is not mandatory. R1.2 gives some alternative way (not exclusive) to fulfil R1.

2.2.2 Extra-Requirements Traceability

The relationships between requirements and other artefacts (extra-requirements traces) include:

- A requirement is `proposedBy` a stakeholder.
- A requirement is `complementedBy` the information included in an ExternalObject (file, graphic, multimedia resource, etc.).
- A requirement could come from an ExternalObject (law, standard, policies, etc.) that is its `source`.
- Both, system and software requirements, should be traced to one or more test cases (`validatedBy`) where it is explained how to check if the requirement is fulfilled or not.
- The terms defined in the glossary are `usedIn` some system or software requirements. These

relationships allow requirements engineers to check if all the requirements using the same term are doing it consistently.

All the examples shown in Figure 2 have been extracted from (Davis, 2005). These examples have been developed using one of the two graphical modelling tools develop on top of REMM. These tools are presented in the following section.

3 A GRAPHICAL MODELLING TOOL FOR REMM

Before we can build a graphical modelling tool for REMM, there are some previous questions which must be addressed. First of all, Section 3.1 analyses the two main trends in MDE: UML profiling and domain-specific language design. The selection of one meta-modelling technique or the other, will determine which kind of development environment must be used. The selected environment will be described in Section 3.2. Finally, in section 3.3 the two graphical modelling tools implemented on top of REMM are presented.

3.1 Meta-Modelling Techniques

As stated in (Abouzahra et al., 2005), nowadays there are two main trends in MDE. The first one promotes the use of standard modelling languages such us UML 2.0 (OMG, 2005) or SysML (OMG, 2006), while the second one advocates the benefits of Domain-Specific Languages (DSLs). UML 2.0 provides a rich set of modelling notations and offers some restriction and extension mechanisms (stereotypes, tagged definitions and constraints) which allow developers to adapt it to their particular domains. These customized versions of UML are commonly known as *profiles*.

On the other hand, DSLs commonly provide a reduced and well-focused set of concepts and tailored notations for describing specific domains. DSLs commonly offer the following advantages when compared to UML or UML profiles: (1) they are often much simpler and thus, easier to understand and use, (2) they better fit the designer modelling needs, and (3) currently, it is possible to find new tools that enable the definition, transformation and validation of models built not only using UML but also self-defined metamodels (e.g. MOF-based Eclipse metamodels).

All these reasons led us to the conclusion that, for the shake of simplicity and expressivity, it was worth to define a new metamodel which could include exactly the concepts we were interested in from the RE domain.

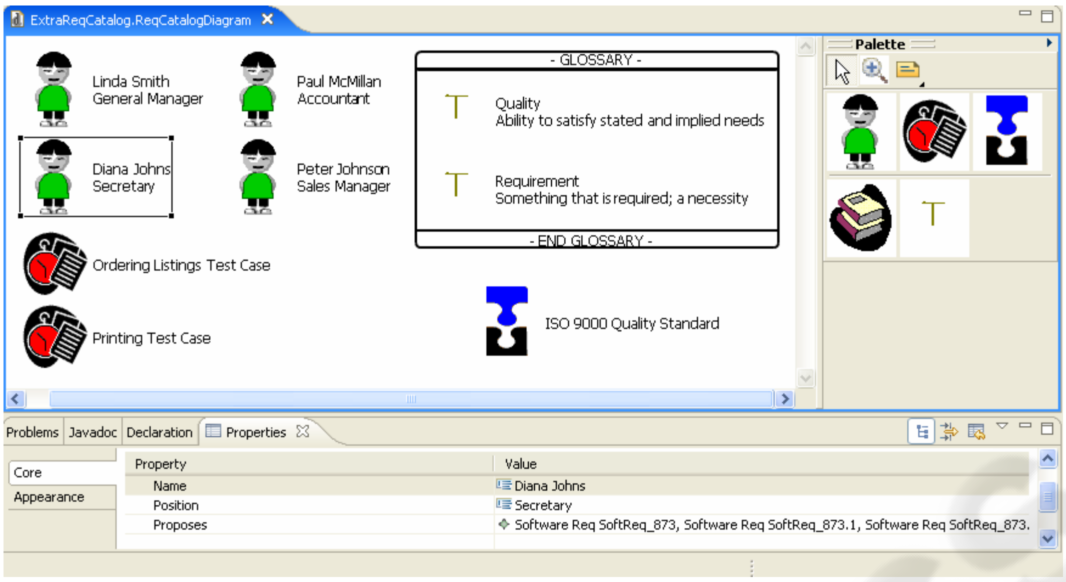
3.2 Selected Development Environment

We have selected to use the Eclipse platform to implement both the REMM metamodel and the supporting graphical modelling tool. To achieve this, the following Eclipse plug-ins have been used:

- The Eclipse Modelling Framework (EMF) (Budinski et al., 2003) which implements a reduced set of MOF, called Essential MOF (EMOF). Actually, REMM (see Figure 1) has been depicted using a graphical EMF modelling tool which produces an equivalent textual representation in EMOF (.ecore file).
- The Eclipse Graphical Modelling Framework (GMF), which allows designers: (1) to create a graphical representation for each domain concept included in the metamodel, (2) to define a tool palette for creating and adding these graphical concepts to their models (see Figure 3), and (3) to define a mapping between all the previous artefacts, i.e. metamodel elements, their graphical representations, and the corresponding creation tools.
- The OCL (*Object Constraint Language*) plug-in, developed in the context of the Model Development Tools (MDT) project, is an implementation of the OCL OMG standard (OMG, 2006) for EMF-based models. The definition of OCL constraints enables the validation of the models further than just checking the conformance with the metamodel (cardinality and static properties). An example OCL restriction, regarding the dependence trace is shown in Table 1).

Table 1: One of the OCL constraints added to the REMM graphical modelling tool.

Description	Only one DepenceTrace is allowed from each requirement.
OCL constraint	`self.dSource.owner.dependences -> one ( d | ( d.dSource=self.dSource ) and ( d.dTarget=self.dTarget ) )`



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

Figure 3: A graphical requirements model depicted using the CatalogTool.

3.3 The REMM-Studio Graphical Modelling Tools

This section presents the two graphical modelling tools implemented as part of this work, both of them based on the REMM metamodel, and integrated in a common environment called REMM-Studio.

For the sake of simplicity, each tool has been designed to support part of the metamodel, considerably simplifying the resulting diagrams. On the one hand, the *RequirementsTool* enables the graphical description of system and software requirements, and the relationships existing between them, i.e.: dependence, influence, and parent-child traces (see Figure 2). On the other hand, the *CatalogTool* allows requirements engineers to depict the rest of the elements included in the metamodel, i.e.: stakeholders, test cases, external objects, and the glossary and its terms (see Figure 3). Both tools jointly used support the whole metamodel.

The models generated from each tool can be loaded and used from the other. For instance, in the catalog model shown in Figure 3, in order to specify the requirements proposed by the secretary, Diana Johns, the user can load the requirements model shown in Figure 2 and select some of them, i.e. 873, 873.1 and 873.2 software requirements. The information is automatically updated in the requirements model, that is, the proposedBy field of the three requirements is set to the stakeholder Diana Johns, thus assuring inter-model consistency.

4 RELATED WORK

There are a variety of approaches regarding RE and MDE integration. They differ not only in the focus but also in the selected meta-modelling technique and in the concepts included in different requirements metamodels.

Two main research directions can be found in the literature regarding combined MDE and RE approaches. On the one hand, some authors try to define a UML Profile for integrating requirements specifications into UML models (Letelier, 2002; Heaven and Finkelstein, 2004; Supakkul and Chung, 2005; Berenbach and Gall, 2006; Escalona and Koch, 2006; Vogel and Mantell, 2006). Most of them integrate non-functional requirements into UML uses cases (Supakkul and Chung, 2005; Berenbach and Gall, 2006), taking for granted that functional requirements are defined using uses cases. Sometimes, the defined UML profiles include concepts of new requirements models to be depicted using the UML notation. This is the case of WebRe (Escalona and Koch, 2006) and KAOS (Heaven and Finkelstein, 2004) profiles.

Regarding RE domain-specific metamodels, some proposals formalize the requirements specifications language (Beeck et al., 2002; Videira and Silva, 2005), while others try to model textual requirements (Marschall and Schoenmakers, 2003; Schätz et al., 2005).

It is worth emphasizing the OMG SysML (*Systems Modelling Language*) standard (OMG, 2006) [9]. This metamodel can be considered a domain-specific modelling language for systems engineering applications. Unlike other metamodels, SysML allows designers to model text-based requirements, which can be integrated with other development artefacts. Nevertheless, it is questioned if the set of inter-requirements relationships provided is enough (Herzog and Pandikow, 2005).

5 CONCLUSIONS

The work presented in paper has demonstrated how a MDE approach to RE can be advantageous to both disciplines. The REMM requirements metamodel and the graphical modelling tools implemented to support it, provide requirements engineers with a new way of describing most of the elements involved in a RE process.

In the future we plan to extend REM in order to include different types of non-functional requirements. We are also interested in the inclusion of some variability mechanism to enable the definition of parameterized requirements in order to promote their reuse.

Currently we are working in the automated generation of textual requirements specifications according to predefined templates (e.g. IEEE 830) or to user defined queries (e.g. retrieve all the requirements proposed by a certain stakeholder).

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish CICYT projects DEDALO (TIN2006-15175-C05-03, University of Murcia) and MEDWSA (TIN2006-15175-C05-02, Technical University of Cartagena).

REFERENCES

Abouzahra, A., J. Bézivin, et al., 2005. A Practical Approach to Bridging Domain Specific Languages with UML profiles. OOPSLA Workshop Best Practices for Model Driven Software Development San Diego, California, USA.

Beeck, M. v. d., P. Braun, et al., 2002. Model based Requirements Engineering for Embedded Software. In: *IEEE International Requirements Engineering Conference*, Essen, Germany.

Berenbach, B. and M. Gall, 2006. Toward a Unified Model for Requirements Engineering. In: *International Conference on Global Software Engineering (ICGSE 2006)*, Costão do Santinho, Florianópolis, Brazil, IEEE Computer Society.

Bézivin, J., 2005. On the Unification Power of Models Software and Systems Modeling (SoSym). 4(2): 171-188.

Budinski, F., D. Steinberg, et al., 2003. Eclipse Modeling Framework, Addison-Wesley Professional.

Champeau, J. and E. Rochefort, 2003. Model Engineering and Traceability. In: *UML 2003. SIVOES-MDA Workshop*, San Francisco. California.

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

Damian, D. and J. Chisan, 2006. An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Transaction on Software Engineering. 32(7).

Davis, A., 2005. JERM: Just Enough Requirements Management, Dorset House Publishing.

Escalona, M. J. and N. Koch, 2006. Metamodeling Requirements of Web Systems. In: *International Conference on Web Information System and Technologies (WEBIST 2006)*, Setúbal, Portugal.

Heaven, W. and A. Finkelstein, 2004. A UML Profile to support requirements engineering with KAOS. IEE Proceeding: Software. 151(1): 10-27.

Herzog, E. and A. Pandikow, 2005. SysML – an Assessment. In: *15th INCOSE International Symposium*.

INCOSE, 2006. Requirements Management Tools Survey, International Council on Systems Engineering, from http://www.paper-review.com/tools/rms/ read.php.

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

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

Muller, P. A., P. Studer, et al., 2005. Platform Independent Web Application Modeling and development with Netsilon. Software & System Modeling. 00: 1-19.

Nicolás, J., J. Lasheras, et al., 2006. A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance. In: *Learning Software Organizations + Requirements Engineering (LSO+RE 2006)*, Hannover. Alemania.

OMG, 2004. Meta-Object Facility (MOF) Specification v2.0, Object Management Group, from http://www.omg.org/docs/formal/02-04-03.pdf.

OMG, 2005. Unified Modeling Language: Superstructure v 2.0, The Object Management Group.

OMG, 2006. Object Constraint Language (OCL) Specification v2.0, Object Management Group.

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

Pinheiro, F. A. C., 2003. Requirements Traceability. Perspectives on Software Requirements (Kluwer Internation Series in Engineering and Computer Science, 753), Kluwer Academic Publishers.

Robertson, J. and S. Robertson, 2000. Requirements Management: a Cinderella Story. Requirements Engineering Journal. 5(2): 134-136.

Schätz, B., A. Fleischmann, et al., 2005. Model-Based Requirements Engineering with AutoRAID. In: *INFORMATIK 2005*, Rheinische Friedrich-Wilhelms-Universität Bonn.

Sommerville, I. and J. Ransom, 2005. An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement ACM Transactions on Software Engineering and Methodology, . 14(1).

Supakkul, S. and L. Chung, 2005. A UML Profile for Goal-Oriented and Use Case-Driven Representation of NFRs and FRs. In: *Third ACIS International Conference on Software Engineering Research, Management and Applications (SERA'05)* Central Michigan University, Mount Pleasant, Michigan, USA.

Toval, A., B. Moros, et al., 2007. Eight key issues for an effective reuse-based requirements process. International Journal of Computer Systems Science and Engineering. (accepted for publication).

Toval, A., J. Nicolás, et al., 2002. Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach. Requirements Engineering Journal. Springer. 6(4): 205-219.

Videira, C. and A. R. d. Silva, 2005. An overview of ProjectIT-RSL metamodel and prototype. In: *6ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI'2005)*, Portugal, Bragança.

Vogel, R. and K. Mantell, 2006. MDA adoption for a SME: evolution, not revolution - Phase II. In: *European Conference on Model Driven Architecture (ECMDA 2006)*, Bilbao, Spain.

Weber, M. and J. Weisbrod, 2003. Requirements Engineering in Automotive Development: Experiences and Challenges. IEEE Software. 20(1): 16-24.

Wiegers, K. E., 2003. Software Requirements, Second Edition, Microsoft Press.