A broad vision of ProjectIT-Requirements, a new approach for Requirements Engineering

Carlos Videira [1], Alberto Rodrigues da Silva[2]

[1] INESC-ID, Universidade Autónoma de Lisboa,
Rua de Santa Marta, nº 56, 1169-023 Lisboa, Portugal
cvideira@acm.org

[2] INESC-ID, Instituto Superior Técnico,
Rua Alves Redol, nº 9 –1000-029 Lisboa, Portugal
alberto.silva@acm.org

Abstract

The development of information systems is a complex task, involving technical, human and organizational issues. The results of this activity are not always successful, and are apparent in the lack of software quality and in costs and schedules overruns. The first initiatives developed to solve these problems were concerned with the implementation issues. It is now widely accepted that the real problem comes from the early activities and information gathered in the beginning of the projects, the most important being those connected with system requirements. Although the efforts made and the initiatives proposed, there is not a widely accepted practice or standard in this area, comparable to what we have achieved, for example, in the modeling activities. This paper presents a brief description of the state of the art in the area of requirements specification, explains the motivation to develop a new initiative, which we have called "ProjectIT-Requirements" and describes the goals we want to achieve with the project and the global context in which it integrates.

Keywords: Software Engineering; Requirements Engineering; Requirements Specification.

1 INTRODUCTION

The software development process is normally defined as a structured sequence of activities, executed in a systematic and uniform way, performed by roles with well defined responsibilities, which process a set of inputs to produce some outputs [Silva 2001]. It also includes the definition of techniques, notations, tools, standards and guidelines that should be used.

When we started to develop information systems, we were highly constrained by the limited technology capacities. The applications built were not a result of a formal development process, but rather of the so-called "build and fix" approach. The users were rarely involved in the process: after all, if they were not capable of interacting with computers at that time, why should they express what they want? So the requirements were poorly identified. Nevertheless, the number and complexity of information systems was growing, which only justified the importance of developing new approaches.

Since then, the approaches followed to develop information systems have introduced a level of formalism in order to solve the quality, budget and schedule problems. The expression "Software Engineering" was used in 1967 during an OTAN conference [Pressman 2000] to show the importance of applying more systematic and rigorous approaches to software development, which had proven effective when applied in other engineering disciplines.

Nevertheless, the problems keep happening and The Chaos Report, a study periodically published by the Standish Group [http://www.standishgroup.com], has consistently shown these results during the last years, as Figure 1 shows.

In *Actas da 5ª Conferência da Associação Portuguesa de Sistemas de Informação.*

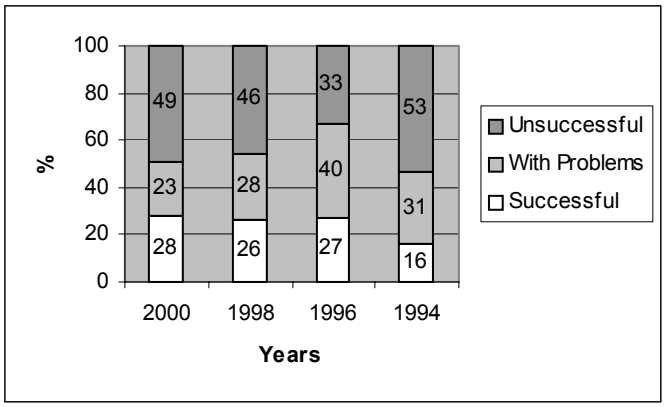

Figure 1 – Level of success of IS/IT projects, according with the Standish Group

This paper presents an initiative in the area of requirements engineering, which proposes a new pragmatic approach to successfully achieve some of the goals of this discipline. Section 2 describes the requirements concept and presents an overview of the requirements engineering activities. Section 3 describes some requirements engineering practices, which somehow influence our work. In Section 4, we describe the ProjectIT initiative, whose main goal is to contribute with new ideas to improve the software development process. Section 5 is dedicated to ProjectIT-Requirements, the component that addresses the issues in the area of requirements engineering; we describe its goals and basic principles, its architecture and functionalities.

2 REQUIREMENTS AND REQUIREMENTS ENGINEERING

There are some variations about the definition of the requirements concept; words such as "needs", "features" or "functionalities" are frequently used as synonyms. The Oxford dictionary says that a "requirement is a need, a dependency for success". A definition proposed by Dorfmann and Thayer suggests that a "requirement is a software capability needed by the user to solve a problem he has to achieve a goal, or said another way, is a software capability that must be met or possessed by a system or component to satisfy a contract, standard, specification or other formalism" [Dorfmann 1990]. In an article published in 1993, Harwell states that a "requirement is something mandatory to be accomplished, transformed, produced, or provided" [Harwell 1993]. A classical definition from Kotonya says that a "requirement is a statement about a system service or constraint" [Kotonya 1998].

Being a critical part of the development process, we have conceived a set of activities to deal with requirements, called "requirements engineering". Nuseibeh provides an interesting overview of this discipline [Nuseibeh 2000]. Zave defines requirements engineering, when applied to software systems, as "the branch of software engineering concerned with the real world goals for, functions of, and constraints on software systems. Requirements engineering is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families" [Zave 1997].

The traditional requirements engineering approach involves a combination of several activities, mostly executed in the beginning of a project, as shown in Figure 2; a process of requirements change management should be applied to monitor and control requirements evolution and changes along the project. A complete description of each of these activities is out of the scope of this paper. In the next section we will present an overview of some existing methods that influence our work.

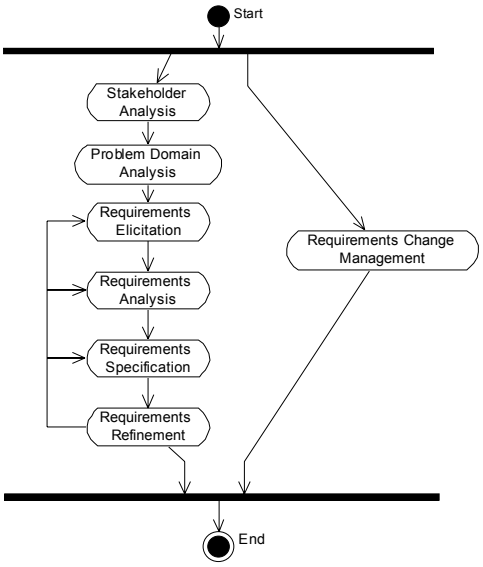

Figure 2 – Common activity diagram of the requirements engineering tasks

3 RELATED WORK IN THE AREA OF REQUIREMENTS SPECIFICATION

Existing requirements engineering methods can be classified through many perspectives. In this article, we will briefly review four areas we have elected to be of prime importance to our project: (1) requirements reuse, because currently few tools support it, and this is crucial for minimizing the manual work in requirements specification; (2) requirements specification initiatives based on UML, because we want to use UML for defining our model and for integrating with other project initiatives and tools; (3) formal methods, because formalization is important to improve the quality and precision of requirements specifications; and (4) agile approaches, because we want to keep things as simple as possible.

The existing requirements reuse initiatives sometimes follow different strategies. In [Villegas 2002] the authors propose a metamodel for reusing requirements, based upon the integration of information available in a number of different existing functional modeling techniques (like scenarios, use cases, activity diagrams, data flows, task documents and workflows). The information captured is subsequently provided for being reused later. Cybulski proposes a framework which integrates the needs of software reuse in the requirements engineering process, and describes a method (RARE) and a software tool (IDIOM) that support the definition of the domain model, later used in the requirements analysis, reuse and refinement process [Cybulski 2001].

UML has become a standard language for modeling software systems and many people have used it for requirements specification. However, some authors have argued that that UML has some deficiencies as a semiformal requirements specification language [Glinz 2000]. Other authors have proposed initiatives to formalize UML, by using formal specification techniques for the semantics of UML notations and diagrams. The Precise UML group developed a precise semantic model for UML class diagrams, to enable formal deductions [Evans 1998]. In [France 2000] the authors describe a requirements modeling technique, named UMLtranZ, based on UML class and use cases diagrams, expressed in a variant form of the Fusion Model [Coleman 1994]. The models can be transformed to formal Z specifications, which can be further analyzed to identify problems and inconsistencies in the requirements specified.

The use of formal requirements specification languages, based upon mathematical notations, such as Z [Spivey 1989], VDM [Woodman 1993] or Larch [Guttag 1993], has also its supporters among the requirements engineering community. They use deductive and inductive reasoning techniques that can be automated and built into specification tools. The question mostly faced by these initiatives is the adoption of those languages by non-technical people. Apparently, some studies demonstrated the capability of the users to learn and apply such methods [Bowen 1994].

The use of controlled natural language for requirements specification, although not a formal approach, facilitates the process of writing, analyzing and verifying requirements [Johnson 1991]. Sometimes this specification can even be transformed into executable statements of software specification [Schwitter 1996], providing systems simulation capabilities. Others followed a different strategy, by providing tools and techniques that transform formal requirements specifications into natural language, so that the stakeholders can validate them [Feather 1989], [Johnson 1991].

Other initiatives worth of mentioning include (1) Kaos [Dardenne 1993], a formal language for reasoning and modeling the system's goals, and its use in conjunction with formal specification languages, such as Timed Automata, for specifying the system's requirements and its internal behavior [Dubois 1998]; (2) RML (Requirements Modeling Language), probably the first important attempt to use knowledge representation techniques in requirements engineering [Greenspan 1982], or (3) Telos, its successor [Mylopoulos 1990]; (4) RSML (Requirements State Machine Language), a formal system modeling language described in 1994 [Leveson 1994].

More recently, the agile software development approaches have also developed some specific techniques to be applied in the requirements activities. Ambler has discussed the best practices about agile requirements modeling, and how to integrate them with the rest of the agile development process [Ambler 2004]. Extreme Programming [Beck 1999] proposes the concept of user stories, a set of small sentences that express the user needs, in his own words. The idea is to simplify, whenever possible, the process of information gathering about the system functionalities, doing the effort needed to capture the current user needs and nothing else. Detailed requirements information represents frequently an unnecessary overhead at the beginning of the project, because requirements change.

4 THE PROJECTIT INITIATIVE

4.1 The project's context

In the context of the Information Systems Group of INESC-ID, in Lisbon, Portugal [http://gsi.inesc-id.pt/], we have been working for some time in these issues with the goal to contribute to the advance of the current state of the art. The group has been involved in the development of two investigation projects in the area of software engineering:

1. XIS, whose main contribution was the development of the XIS/UML profile intended to specify, simulate and develop information systems following a MDD approach [Silva 2003].

2. ProjectPro, a Web based collaborative system to support the definition of basic concepts (projects, releases, sub-systems, requirements) and their relationships [Moreira 2003].

Recently we decided to integrate both these projects in a single project, which we have called "ProjectIT". The idea is to provide a complete software development workbench, with support for requirements engineering, analysis and design activities, project management issues and code generation. Currently, the project's architecture is divided into a number of components, as shown in Figure 3.

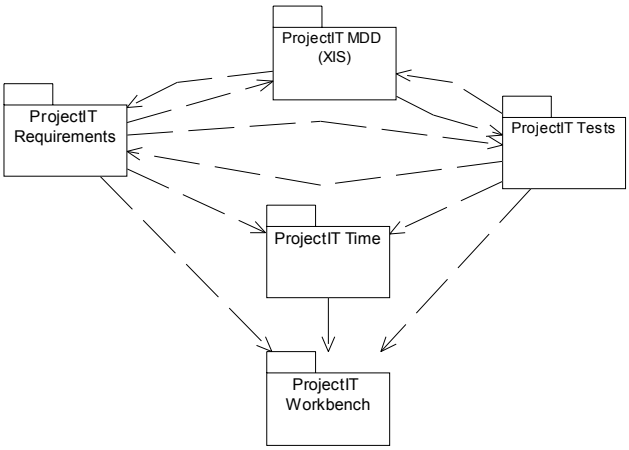

Figure 3 – ProjectIT architecture

ProjectIT-Workbench is the only mandatory sub-system, as it provides the basic and common infrastructure to all the others. ProjectIT-Time enables the definition and description of activities to be performed, and their workflow, in the context of a project.

ProjectIT-Tests is an initiative in the area of tests engineering, strongly related with requirements engineering. ProjectIT-MDD is an initiative in the area of information systems modeling and model driven development. For historical reasons, it can be also known by "XIS", which stands for "eXtensible Interactive Systems".

4.2 The opportunity

Our thoughts in the area of requirements engineering led us to conclude that, besides the business and technological context, there are important factors that justify new initiatives in the area of requirements engineering:

1. Requirements engineering is a crucial part of the software development process; to achieve success in this area, requirements must somehow be formalized and the confusions that exist must be avoided.

2. The investigation initiatives developed in the past, like those described in sections 2 and 3, have represented an important contribution, but for a number of different reasons they have not been widely adopted.

3. The available tools, such as CaliberRM [http://www.borland.com/caliber/], RequisitePro [http://www-306.ibm.com/software/awdtools/reqpro/], Doors [http://www.telelogic.com/products/doorsers/], RTM [http://www.chipware.com/], although representing an important step forward, are above all requirements management tools; they have many features to input requirements in the system, to classify them, to define relationships, and to analyze them, but most of them still lack the capabilities to automate important tasks, or to guarantee a consistent integration with the rest of the development process.

4. There is an evident gap between the academic and investigation initiatives developed and their application and implementation in tools adopted by the market.

5 PROJECTIT-REQUIREMENTS

ProjectIT-Requirements is the component of the ProjectIT architecture responsible for some of the issues related with requirements engineering. The main goal of the project is the development of a model for the definition and documentation of requirements, which, by raising the specification rigor, facilitates the reuse and integration with development environments

driven by models. We are currently interested in software requirements, one of the many types of requirements [Kotonya 1998], which are those that can more easily be "converted" in software design models by MDD approaches; we recognize that other types of requirements, in particular non-functional requirements (e.g., usability, performance, security, safety, maintainability), are more difficult to formalize and quantify, and require different approaches.

5.1 ProjectIT-Requirements Goals

The main idea behind the project is that we must combine the benefits of formalizing the requirements specification, with the need to use a format and notation that is understandable by every involved project stakeholder. So our hypothesis is that we should adopt a "controlled natural language", a subset of natural language with specific rules for requirements specification, with a limited vocabulary and a simplified grammar. The project ProjectIT-Requirements is a pragmatic requirements engineering initiative that aims to investigate and produce results in various issues, namely:

1. The definition and management of the project glossary, which is elaborated by identifying the business entities and their properties.

2. The definition and formalization of a requirements specification language, which can be represented textually and graphically, and provides the basic mechanisms to define rigorously the system requirements. In order to maximize the benefits of close integration with the other components of the ProjectIT architecture, benefiting from a single and global metamodel, we have decided to define a new language and not reusing an existing one.

3. The definition of mechanisms to provide the requirements management in the context of specific projects based in requirements architectures.

4. The capability of establishing relationships between requirements and other project-based concepts, for example, projects, people, modules, etc. It will be possible to relate requirements to software projects, to assign requirements to people, like the interested stakeholders or the programmer responsible for its implementation, and to define the workflow of a potential requirement until it gets officially approved.

5. The extension of the current XIS/UML profile in order to provide the seamless integration with modeling activities and the other ProjectIT sub-systems.

5.2 ProjectIT-Requirements principles

In order to achieve the results mentioned, the project follows some important principles, as Figure 4 shows.

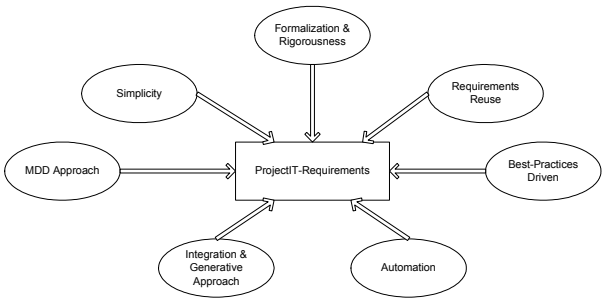

Figure 4 – Main principles that drive the ProjectIT-Requirements

We pretend to develop ProjectIT-Requirements so that every stakeholder in the software development process can use the tools. Raising the level of the rigor of the requirements specification, through formalization, enables to reduce the errors frequently found in the requirements engineering process [Kotonya 1998]. Doing it using a subset of natural language makes this formalization process understandable by everyone. Here we recognize a strong

influence of the Extreme Programming notion of a user-story; the difference is that we want to achieve the same result using a limited set of vocabulary terms.

Requirements activities are mostly manual, involving a lot of personal effort, so every contribution to automate them will be valuable. Integration with the other ProjectIT components will shorten the development cycle and enable the process to produce more consistent results. We also believe that simplicity of the concepts developed and of the model proposed is key to its success.

There are some initiatives in the software development process that have proposed relevant best practices, which we want to take into account and use appropriately. Besides the model driven ideas, we recognize a strong influence from the ideas of product line software development [Kuloor 2002] and agile development [Beck 1999]. Aspect-oriented requirements engineering [Rashid 2002] and viewpoints [Finkelstein 1994] are also initiatives we will monitor to understand how they can influence our work.

5.3 ProjectIT-Requirements in the context of ProjectIT

The main components of the ProjectIT-Requirements architecture in the context of ProjectIT are shown in Figure 5.

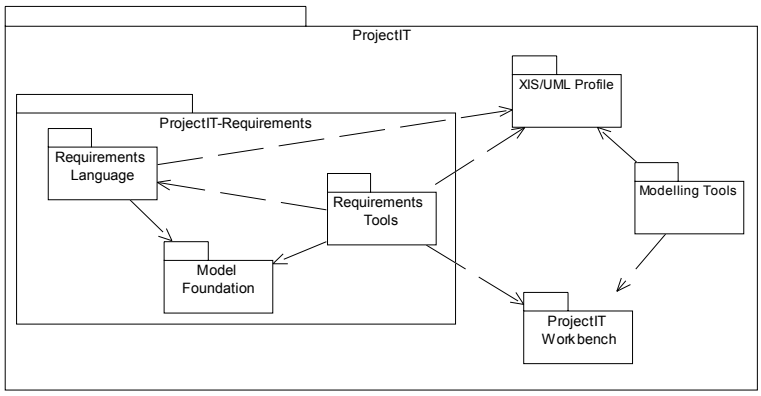

Figure 5 – Relationship between ProjectIT and ProjectIT-Requirements

The Model Foundation defines the project vision, its strategy, how it interacts with other ProjectIT components and the vocabulary used throughout the other parts of the project. The Requirements Language is defined by a model, which shows the concepts to be used for requirements description, and by a grammar for defining the rules to map these concepts into sentences. The first defines the language's syntax and the second its semantics. The Requirements Tools supports the definition of requirements and their exploration in real projects, in conformance with the language defined.

Some of the specific goals of the project will be achieved by establishing relationships with other components of the ProjectIT architecture. The relationship with the ProjectIT-Time sub-system supports the association of project activities to specific requirements, and enables the definition of people and time information in the context of requirements. With ProjectIT-Tests the relationship is the consequence of the need to associate requirements to the tests that validate and verify them, using whenever possible semi-automatic approaches.

The ProjectIT-Requirements architecture is based upon an integrated set of tools that together will enable the definition of requirements and their relationships and provide the support for reusing previously defined requirements and the integration with modeling tools, developed in the context of ProjectIT-MDD. This architecture is shown in Figure 6.

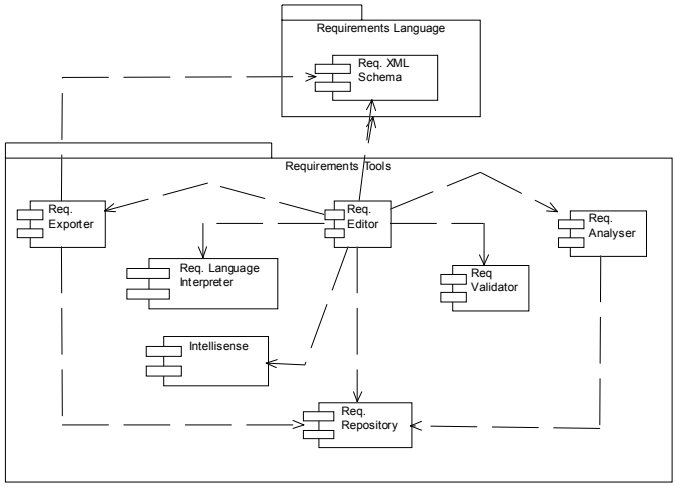

Figure 6 – ProjectIT-Requirements tools architecture

Requirements are managed by the **Requirements Editor**, which provides the central functionalities of the whole system. It acts like a traditional editor for introducing controlled requirements text, with additional options like the possibility of classifying requirements (for later reuse) and viewing traceability information. The **Requirements Interpreter** and **Intellisense** are both responsible for checking the requirements definitions, introduced in the editor, and compared against the rules defined by the **Requirements XML Schema**, which is generated from the model. We envision a tool that enables the writing of "requirements documents", like in a word processor, and as we write, it will warn us of errors violating the requirements language and grammar we have defined (very similar to what happens nowadays in the Word application, underlining in red some syntactical errors). A more detailed description of the current toolset functionality is presented in [Videira 2004].

6 CONCLUSIONS

The business and technological environments are becoming more complex and difficult to manage. Although the efforts made, we continue to achieve a significant lack of success in our software development projects. Recognizing that the most critical problems occur during the requirements engineering activities, we strongly believe that the integration with other project tasks, in a model driven development approach, reusing previously developed requirements, expressed in a more formal (yet universally understandable) way, and which are used by automated tasks, are points we consider to be of prime importance to guarantee the success of any initiative. This pretends to be the contribution of ProjectIT-Requirements.

We have already defined an initial version of the Requirements Specification Language (ProjectIT-RSL) and built a prototype. The process followed and the results achieved are described in [Videira 2004]. The future work includes improving and increasing the scope of the ProjectIT-RSL (for example, adding support for the specification of more complex and versatile operations and workflows) while, at the same time, we will research for more advanced features, such as requirements reuse based on requirements architectures. It is also our intention to use the developed tools in real projects, to test and proof the ideas we are proposing.

In this paper, we have described the motivation for the project and the project's principles and architecture. We pretend to make some innovative proposals, but we also intend to integrate the ideas of previously developed initiatives. We believe that the defined strategy can contribute to new ideas that can have a pragmatic impact in software development.

7 REFERENCES

Ambler, S, *Agile Requirements Modelling,* http://www.agilemodeling.com/essays/agileRequirements.htm, 2004

Beck, K., *Extreme Programming Explained: Embracing Change*, Addison Wesley, October 1999

Bowen, J., Hinchey, M., *Seven More Myths of Formal Methods*, PRG-TR-7-94,:Oxford University Computing Laboratory, 1994

Coleman, D., et al., *Object-Oriented Development: The Fusion Method*. Prentice Hall, Englewood Cliffs, NJ, Object-Oriented Series edition, 1994

Cybulski, J., *Application of Software Reuse Methods to Requirements Elicitation from Informal Requirements Texts*, PhD Thesis, La Trobe University, Australia, March 2001

Dardenne, A., A. van Lamsweerde, A., Fichas, S., *Goal directed requirements acquisition*, Science of Computer Programming, 20:3–50, 1993

Dorfmann, M., Thayer, R., *Standards, Guidelines, and Examples of System and Software Requirements Engineering*, Los Alamitos, California, IEEE Computer Society Press, 1990

Dubois, E., Yu, E., Petit, M., *From Early to Late Formal Requirements: a Process-Control Case Study*, Proc. 9th Int. Workshop on Software Specification and Design, Isobe, IEEE CS Press, 34-42, April 1998

Evans, A., France, R., Lano, K., Rumpe, B., *Developing the UML as a Formal Modelling Notation*, UML'98, Beyond the notation International Workshop, Mulhouse France, 1998

Feather, M., *Reuse in the context of a transformation-based methodology, in Software Reusability: Concepts and Models*, Biggerstaff, T.J. and Perlis, A.J. (Editors). New York, New York: ACM Addison Wesley Publishing Company. p. 337-359, 1989

Finkelstein, A., Nuseibeh, B., Kramer, J., *A framework for expressing the relationships between multiple views in requirements specification*. Transactions on Software Engineering. 20(10): p. 760-773, 1994

France, R., Grant, E., Bruel, J., *UMLtranZ: An UML-based rigorous requirements modelling technique*, Technical report, Colorado State University, Ft. Collins, Colorado, January 2000

Fuchs, N., Hofmann, H., Schwitter, R., *Specifying Logic Programs in Controlled Natural Language*, 94.17,: Department of Computer Science, University of Zurich, 1994

Glinz, M., *Problems and Deficiencies of UML as a Requirements Specification Language*, Proc. of the 10th IEEE Int. Workshop on Software Specification and Design, 2000

Greenspan, S., Mylopoulos, J. and Borgida, A., *Capturing More World Knowledge in the Requirements Specification*, Proceedings 6th International. Conference on SE, Tokyo, 1982

Guttag, J., Horning, J., *Larch: Languages and Tools for Formal Specifications*, New York. Springer-Verlag. 1993

Harwell, R. et al, *What Is A Requirement?*, in Proceedings of the Third International Symposium of the NCOSE, 1993

Johnson, W., Feather, M., *Using evolution transformation to construct specifications, in Automatic Software Design*, Lowry, M.R. and McCartney, R.D. (Editors). Menlo Park, California. The MIT Press. p. 65-91, 1991

Kotonya, G., Sommerville, I., *Requirements Engineering Processes and Techniques*, New York. Jonh Wiley & Sons, 1998

Kuloor, C., Eberlein, A., *Requirements Engineering for Software Product Lines*, Proc. of the 15th Int. Conference on Software & Systems Engineering and their Applications, Paris, 2002

Leveson, N., Heimdahl, M., Hildreth, H., Reese, J., *Requirements specification for process-control systems*. IEEE Transactions on Software Engineering, pp 684-706, September 1994

Moreira, V., *ProjectPRO, Plataforma de Gestão de Requisitos*, Relatório de Trabalho Final de Curso, Lisboa, July 2003

Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M., *Telos: Representing Knowledge about Information Systems*, ACM Transactions on Information Systems, October 1990

Nuseibeh, B., Easterbrook, S., *Requirements Engineering: a roadmap*. In Proc. of the Conference on The Future of Software Engineering, pages 35-46, Limerick, Ireland, 2000

Pressman, R., *Software Engineering: A Practitioner's Approach*: European Adaptation. 5th Edition, McGraw-Hill Education, Europe, 2000

Rashid, A., Sawyer, P., Moreira, A., Araújo, J., *Early Aspects: a Model for Aspect-Oriented Requirements Engineering*, Requirements Engineering 2002 (RE'02), Essen, Germany, 9-13 September 2002

Schwitter, R., Fuchs, N., *Attempto - from specifications in controlled natural language towards executable specifications*, in GI EMISA Workshop, Natürlichsprachlicher Entwurf von Informationssystemen. Tutzing, Germany, p. 163-177, 1996

Silva, A., Videira, C., *UML, Metodologias e Ferramentas CASE*, Centro Atlantico, Lisboa, 2001

Silva, A., Lemos, G., Matias, T., Costa, M., *The XIS Generative Programming Techniques*, Proc. of the 27th Annual Int. Computer Software & Application Conference, Dallas, 2003

Spivey, J., *An introduction to Z and formal specifications*, Software Engineering Journal. 4(1): p. 40-50, 1989

Videira, C., Carmo, J., Silva, A., The ProjectIT-RSL Language Overview, accepted for publication in UML 2004, Lisboa, October 2004

Villegas, O., Laguna, M. Garcia, F., *Reuse based Analysis and Clustering of Requirements Diagrams*, Proceedings of REFSQ'02, 8th Int. Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany, September 2002

Woodman, M. Heal, B., *Introduction to VDM*. McGraw-Hill, London, 1993

Zave, P., *Classification of Research Efforts in Requirements Engineering*. ACM Computing Surveys, 29(4): 315-321, 1997