REDUCING REQUIREMENTS TO EIS SPECIFICATIONS GAP USING RM-ODP ENTERPRISE VIEWPOINT

Christophe Addinquy and Bruno Traverson
Valtech 80 avenue Marceau F-75008 Paris France
EDF R&D 1 avenue du Général de Gaulle F-92140 Clamart France

Keywords: Requirements, Viewpoints, SysML, RM-ODP, Enterprise Information Systems.

Abstract: RM-ODP (Reference Model - Open Distributed Processing) standard prescribes architectural viewpoint specifications, but does not address traceability with requirement expression. In this article, we propose a three-layer approach to requirements modelling, from the system high level goals, to the detailed business rules and extra-functional requirements. Then, these descriptions are connected to key elements of the RM-ODP enterprise viewpoint. This approach is illustrated by a case study.

1 INTRODUCTION

The requirements management, in all its entirety is a wide area that may be classified, as shown in figure 1, in three main dimensions.

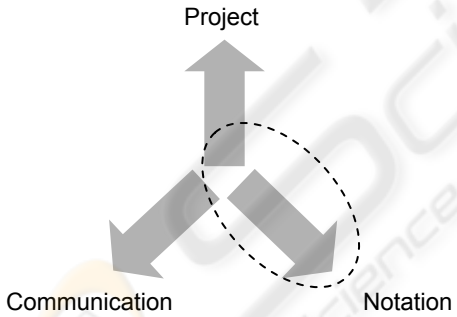

Figure 1: The three dimensions of the requirements management.

The Project axis covers the "who" and "when" about requirements, because everything doesn't occur at the same time. Early phases may be focused on the big picture and later phases on iterative refinement of detailed requirements.

The Communication axis dig the way the requirements are captured, written and reviewed between the analyst and the business specialist. Indeed, requirements are mainly about understanding the needs, and the way people interact and make the requirements happen is essential.

The Notation axis is all about the way the requirements are formalized, structured and followed. Managing requirements depend exclusively on it.

This article focuses on notation. We also call it "requirements modeling" here. For the seek of illustration, some requirements are expressed using SysML (Systems Modeling Language) (OMG, 2006).

We have defined a three-steps process to obtain modeling of what the system must be and what qualities it must have. The goal of this article is not to describe the iterative process of requirements specifications, but of describing the relationships between and inside these 3 levels and how they are related to an architectural description based on RM-ODP (Reference Model – Open Distributed Processing) viewpoints (ISO, 1995).

Hence, RM-ODP also covers the "why" about the system especially in the enterprise viepoint. Establishing links between these two levels of specifications enables us to feel the gap between requirements expressions and system specifications.

In section 2, we describe in more detail our modeling appoach for requirements using three levels. Section 3 comes back to RM-ODP viewpoints and, more specifically, to the global consistency issue and the enterprise viewpoint notation. Then, section 4 illustrates the connection of requirements modeling elements to RM-ODP enterprise viewpoint specification elements on a case study. Finally, section 5 recaps assets and limits of the experiment and draws some perspectives.

Addinquy C. and Traverson B. (2007).
REDUCING REQUIREMENTS TO EIS SPECIFICATIONS GAP USING RM-ODP ENTERPRISE VIEWPOINT.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 31-38
DOI: 10.5220/0002378000310038
Copyright © SciTePress

2 REQUIREMENTS

The three levels of requirements are summarized on figure 2.

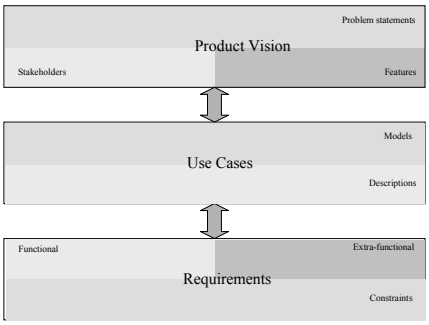

Figure 2: Three levels of requirements specifications.

The Product Vision collects the goals of the system to be specified. The approach chosen here is based on (Leffingwell, 1999). The system functional perimeter is described through Use Cases, a standard UML (Unified Modelling Language) approach (Rumbaugh, 2004) (OMG, 2005) dedicated to that purpose. These Use Cases are built upon the Product Vision previously settled down and must support a set of common rules described in the third level. These rules may be business rules (functional requirements), extra-functional requirements or design constraints. The structure of these low-level requirements is based on SysML, a UML Profile for system engineering.

2.1 Product Vision

The Product Vision captures both the description of the problems at hand and features expected to address these problems. The Product Vision is not expected to be complete and accurate, but it's expected to show the future system main drivers clear enough to get all project stakeholders understanding and support.

The problem statements permit to gain agreement on the perceived problem. The purpose of this step is to get a collective agreement about what's wrong with the way the business works today. Once perceived problems are identified, we have to find out the problems behind the problem, or what contributes to the perceived problem. These root problems are the one that should be addressed by the future system. Each root-cause problem should be expressed using the following statement (table 1).

Table 1: Problem statement format.

The problem of	*Root cause problem statement*
Affects	*People or processes affected*
The result of which	*Contribution of the root cause to the perceived problems*
A convenient solution should	*Proposed solution and few key benefits*

The features are a high-level expression of capabilities expected from the system. The features are the solution space expression of the needs expressed through the problem statement. A feature is expressed in one or two natural language sentence (often expressed by the user himself). No deep detail is required, but it must cover all the needs raised by the problems statements (Larman, 2001).

2.2 Use Cases Specification

The goal of the Use Case specification (figure 2) is to describe the specification of the whole system through usage scenario between the Actors (humans or systems interacting with the system under design) and the system considered as a black-box. The Use Cases are described at two levels: The Use Cases model and the Use Cases descriptions.

The Use Case model is a UML model which describes the whole set of Use Cases and Actors which covers the system functional perimeter. The Use Case model can be organized in packages (following functional areas, for instance) and are expressed with Use Case diagrams. The following figure (figure 3) shows a subset of the case study using such kind of diagram.

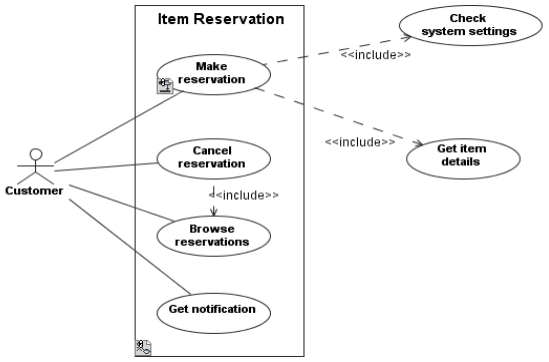

Figure 3: Example of a Use Case diagram.

The Use Case description expresses the intended behaviour. The UML is silent about what are the elements of a Use Case description, even if it allows the usage of state diagrams and activity diagrams within the Use Cases. The Use Case are better described through scenarios (often called "flow of events"), where each step describes an interaction

from the actor to the system or reverse. These scenarios may be expressed using text, with a short and concise statement for each step description (Cockburn, 2001). Three kinds of flow of events require consideration (Bittner, 2002): the main flow of events describes a successful scenario, alternative flows of events are less used variants which occur at some point in time during the main flow of events and errors flows of events describing steps leading to a Use Case failure.

The requirements described in the third level of description may be referenced inside the flow of events when they apply (Wiegers, 1999).

2.3 Requirements Modeling

The Requirements Specifications (figure 2) describe and structure three kinds of elements:

Functional requirements (Business rules): specific processing rules or behavior which must be enforced during design.

Extra-functional requirements: qualities the system must have, putting aside the functional considerations. The IEEE 830.1998 (IEEE, 1998) call them "system attributes" for the most part, as (Wiegers, 1999) and (Sommerville, 1997) do. SysML (OMG, 2006) also proposes as a non-normative extension, a set of 4 extra-functional requirements types. However, we found that the most useful categorization come from the Volere process (Robertson, 2006), which define 8 types of extra-functional requirements (look and feel, usability, performance, operational and environmental, maintainability, security, cultural, legal).

Constraints: restrictions on the degree of freedom we have in providing a solution (Leffingwell, 1999). They falls into several categories: environment (limited memory or time), technological (standard platform defined), normative or economic.

Functional and extra-functional requirements support properties definitions. Again, standards such as (IEEE, 1998) or (OMG, 2006) proposes few ones but it must be adapted to fits organization or projects needs. We choose to use SysML, because this UML-based notation allows not only requirements definitions and properties associations, but also relationships either between requirements or between requirements and other modelling elements (such as test cases or classes).

The example bellow (figure 4, borrowed from the normative document), represent two main requirements decomposed in four child requirements, with two of them copied from a master (reusable) requirement.

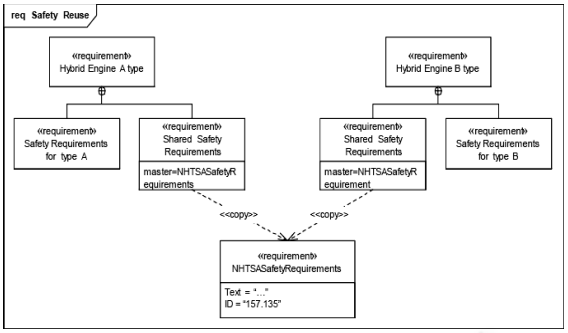

Figure 4: Example of a SysML requirements diagram.

3 VIEWPOINT SPECIFICATIONS

According to IEEE recommendation (IEEE, 2000), architectural description of software-intensive systems should relay on different viewpoints. This approach leads to multi-dimensional specifications where each dimension addresses a particular concern. Hence, complexity is handled by focusing on relevant aspects of each viewpoint. However, this strategy also leads to a global consistency issue. Each viewpoint specification must be locally consistent (i.e. enforces viewpoint constraints) but also globally consistent (i.e. does not contradict other viewpoint specifications).

To illustrate this viewpoint approach, we use in the Reference Model for Open Distributed Processing (RM-ODP) whose key concepts are recalled hereafter. Then, we refers to related works on the global consistency issue of this kind of approach. Finally, we focus on the Enterprise viewpoint concepts that are mainly concerned by the bridging with requirements.

3.1 RM-ODP Viewpoints

RM-ODP recognizes five viewpoints: Enterprise, Information, Computation, Engineering and Technology.

Identifying those viewpoints allows the system specification to express at the same time but distinctly the business the Information System supports (Enterprise Viewpoint), the way it is modeled in the computer system regarding information and functions (Information Viewpoint, Computational Viewpoint, Engineering Viewpoint) and the technical choices of the computer system mapping user requirements (Engineering Viewpoint, Technology Viewpoint).

The key points of RM-ODP are the completeness of its concepts and structuring rules and the relevance of its abstraction levels.

3.2 Global Consistency

Global consistency of such multi-dimensional systems may be ensured by correspondence rules between viewpoint specifications. These latter must be verified at the construction of the viewpoint specification and during its evolution.

DASIBAO and ODAC approaches illustrate global consistency management at the building time. The ODAC project (Open Distributed Applications Construction) (Gervais, 2003) carried out by the LIP6 laboratory and the DASIBAO project (Method based on ODP for the Architecture of Information Systems) (Picault, 2004) carried out by EDF R&D define, each one of both projects, an approach for building consistent ODP systems. The system is built in following steps and by applying transformation rules to the models. However, this consistency is lost if one of the models is modified. Also, they impose a "top-down" approach which is not adapted to evolutionary systems.

EVOS framework manages correspondence links and permits to use them during evolution. This framework (Yahiaoui, 2006) is based on a Link Meta-Model and is implemented as an Eclipse plug-in. The Link Meta-Model goes beyond simple traceability because it contains active rules that permit impact management.

3.3 Enterprise Viewpoint

First of all, a system is described from an Enterprise viewpoint by its main characteristics, as shown in figure 5 and, in a more detailed manner, in figure 6

NB: All the figures of this section are borrowed from the normative document (ISO, 2006).

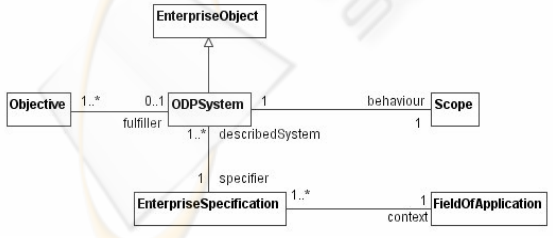

Figure 5: Main enterprise system concepts.

An enterprise specification describes an *ODP system* (a kind of *enterprise object*) and relevant aspects of its environment. The *ODP System* has a *scope***,** which defines the *behaviour* that the system is expected to exhibit. An enterprise specification has a *field of application* which describes its usability properties.

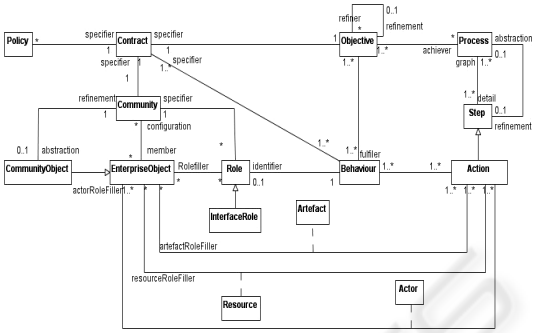

Figure 6: Community and behaviour concepts.

A *community* is a configuration of *enterprise objects*, formed to meet a single *objective*, which is expressed in a *contract* that specifies the required *behaviour* of the *community*. The configuration of a *community* is expressed in the way *enterprise objects* interact in fulfilling *roles* intended to meet the *objective* of the *community* concerned. A *behaviour* is expressed as a collection of *actions* (things that happen), with constraints on when they occur.

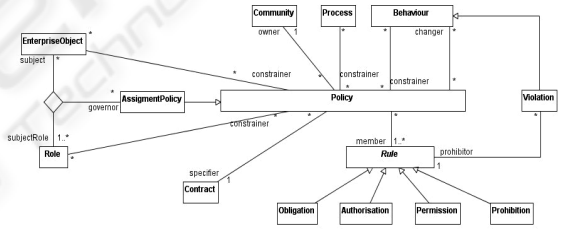

Figure 7: Policy concept.

A *policy* (figure 7) is a set of rules related to a particular purpose. It identifies the specification of *behaviour*, or constraints on *behaviour*, that can be changed during the lifetime of the ODP system, or that can be changed to tailor a single specification to apply to a range of different ODP systems.

The specification of a *policy* includes:
– the name of the *policy*;
– the rules, expressed as *obligations*, *permissions*, *prohibitions* and *authorizations*;
– the elements of the enterprise specification affected by the *policy*;
– *behaviour* for changing the *policy*.

4 LINKING REQUIREMENTS TO ENTERPRISE VIEWPOINT

The following case study is a partial description of an supervisor system used in a scientific environment (Salome, 2007). In this context, supervision basically permits to model a study by a graph of computing tasks (handled by physician or mathematical problem solvers). We've limited the study scope but haven't oversimplified the requirements to keep them realistic. We also give extracts of the enterprise specification in order to exhibit some correspondences between requirements and enterprise specification elements.

4.1 Requirements Specification

The supervisor system will provide a common supervising infrastructure for the existing projects. This mutualization should lead to developments and maintenance costs reductions.

One example of problems statements is given in the following table (table 2).

Table 2: Batch execution problem statement.

The problem of	The batch execution of the computation graph
Affects	The graph user.
The result of which	Is an unknown and unplanned execution which could take time.
A convenient solution should	Allows execution evaluation before and during runtime. The execution nodes distribution could also be rebalanced accordingly.

Main stakeholders are graph designer and graph user (table 3).

Table 3: Graph user stakeholder.

Profile	Graph user
Key Responsibilities	Production of study results
Deliverables	Study results
Trends that make the job easier or more difficult	Visibility on the load balancing Visibility on the execution state of the computation graph
Problems that interfere with success	Initial context restoration after the execution
Definition of success for this user	Efficient execution of the computation graph

Some examples of features are the following.

Instantiation of a coupling graph on deployment architecture. A computation graph must be defined independently from deployment considerations. The execution configuration must be decided on a by execution basis.

Older supervisors backward compatibility. The target version of the supervisor system must be able to handle graph definitions of previous supervisors.

Graph validation prior to execution. It must be possible to check the graph validity and to get a raw estimation of the graph execution. The Graph user will then be able to adjust the configuration, based on these data.

The use case model is structured in three packages (figure 8).

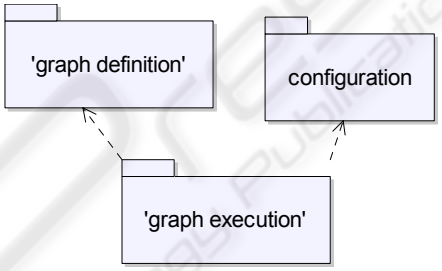

Figure 8: Structure of the Use Cases model.

Graph definition: gathers all use cases for creation, modification or import of coupling graphs.

Configuration: gathers the use cases on execution nodes configuration and applications deployment.

Graph execution: gathers the use cases on running, controlling and analyzing coupling graphs executions.

The "graph execution" package contains the Use Cases appearing on the following graph (figure 9). NB: Figures 9 and 10 have been intentionally left in French. We will come back to this point in the Conclusion section.

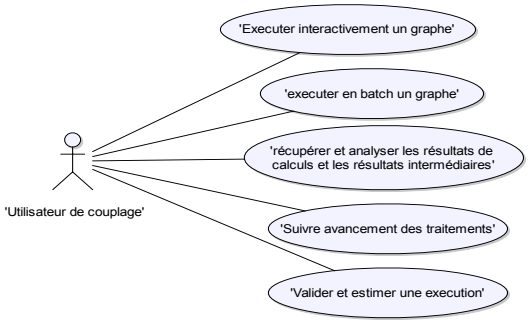

Figure 9: Use case diagram for the "graph execution" package.

Low-level requirements and constraints are described in SysML (figure 10).

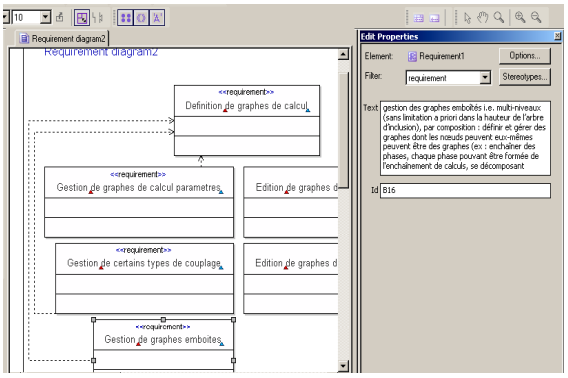

Figure 10: Example of requirement in SysML.

4.2 Enterprise Viewpoint Specification

This specification is expressed using the UML profile for ODP Enterprise specifications defined in (ISO, 2006). UML stereotypes are derived from the concepts described in section 3.3.

At the global level, the enterprise specification of the supervisor system gives the field of application and the communities of the supervision system (figure 11).

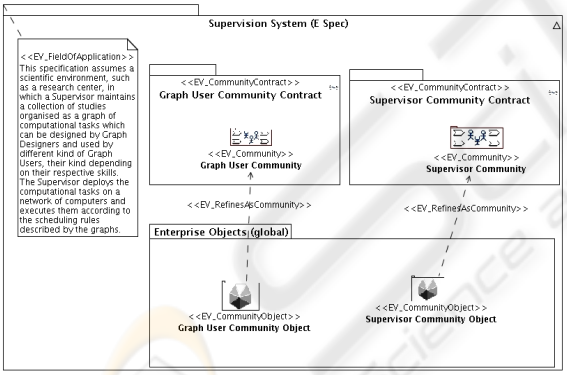

Figure 11: Supervision system description.

As for the field of application, the specification of the supervision system assumes a scientific environment, such as a research center, in which a supervisor system maintains a collection of studies organized as graphs of computational tasks which may be described by graph designers and used by different kinds of graph users, their kind depending of their respective skills. The supervisor system deploys the computational tasks on a network of computers and executes them according to the scheduling rules described in the graphs.

Among the three main communities recognized at the top level of the description, the supervisor community is greater detailed in figure 12.

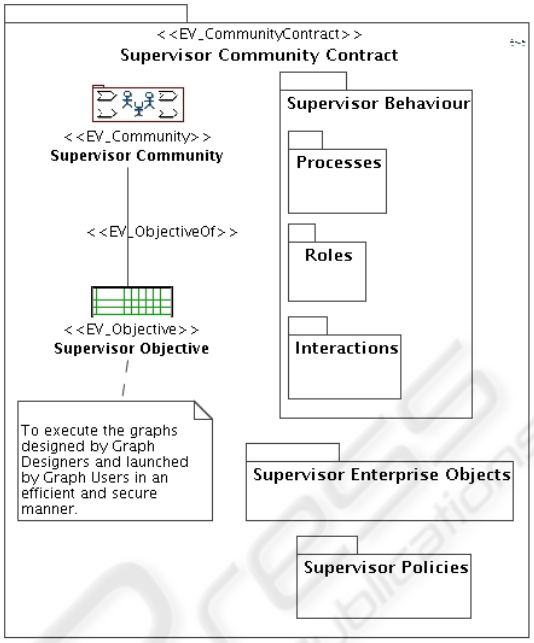

Figure 12: Supervisor community contract.

The objective of the supervisor system is defined as to execute computational graphs designed by graph designers and launched by graph users in an efficient and secure manner. Enterprise objects, behaviour and policies of the supervisor system are also described. A sample of supervisor policy is given in figure 13.

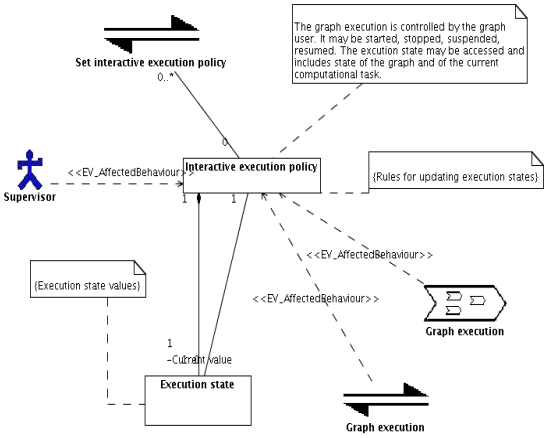

Figure 13: Interactive execution policy.

In the interactive execution mode, the graph execution is controlled by the graph user. It may be started, stopped, suspended or resumed. The execution state may be accessed and includes state

of the graph and of the current computational task.

4.3 Traceability between Requirements and Enterprise Viewpoint Specification

The case study described using the requirements formalism in section 4.1 and the enterprise language in section 4.2 has put into light tightly-related concepts in both specifications. These relationships may be generalized for traceability purpose. Some of those are summarized in the following.

First of all, the concept of "objective" in the ODP enterprise language corresponds to the "feature" concept in requirements notation. For instance, the objective expressed in the enterprise specification (supervisor objective of figure 12) is related to "instanciation d'un graphe sur une architecture de déploiement" and "paramétrage et vérification préalable à l'exécution" features of the requirements specification.

Then, "field of application" of the system may be expressed by extra-functional requirements of usability, maintainability, cultural or legal kind. Its "behaviour" may be illustrated by use cases in the requirements specification.

Lastly, extra-functional requirements of look and feel, performance, operational and environmental, or security kind may be handled by policies in the enterprise specification.

5 CONCLUSION

Requirement management is a key factor of success for computer-based systems. However, it is not always easy to link requirement expressions to their representations as specification elements or as software components.

We have described, in this paper, an experiment whose goal was to link requirements expressions including SysML requirements diagrams to high-level system specifications, here ODP enterprise specifications using UML4ODP enterprise profile.

The experiment has demonstrated the feasibility of the approach and has consolidated our vision of the complementary qualities of the various notations. ODP enterprise language is a bit too formalized for end-user requirement expressions but is the ideal candidate to bridge the gap between these letter and the more technical specifications of the system. Moreover, this coupling of specifications appears to be more than a simple duplication. It enables, as for a very simple example, traceability between terms expressed in French (easier access by French-only speaking end-user) and model elements named in English (easier externalisation end reuse).

With regards to specification languages, even though they are two UML dialects, SysML and UML4ODP enterprise languages may not be supported in the same UML tool. In this case, bridging the gap implies the additional technical challenge of making UML tools interchange data using the same XMI (XML mete-modeling Interchange) and UML versions.

We also seek to provide support for links established between requirements and enterprise specifications. As UML Trace may appear insufficient, alternative solutions may be built on top of already mentioned EVOS Link Meta-model (see section 3.2) or QVT (Query View Transformation) technologies.

REFERENCES

Bittner, K., Spence, I., 2002. *Use Case Modeling.* Addison Wesley.

Cockburn, A., 2001. *Writing Effective Use Cases.* Addison Wesley.

Dick, J., 2005. *Design Traceability.* In IEEE Software 2005, issue 6.

Egyed, A., Grünbacher, P., 2004. *Identifying Requirements Conflicts and Cooperation: How quality attributes and automated traceability can help.* In IEEE Software 2004, issue 6.

Gervais, M.P., 2003. *ODAC: An Agent-Oriented Methodology Based on ODP*. Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, Vol. 7, n¬°3, pp199-228, 2003.

IEEE, 2000. *Recommended practice for architectural description of software-intensive systems*. IEEE Std 1471–2000.

IEEE, 1998. *Recommended Practice for Software Requirements Specifications.* IEEE-830-98.

ISO, 1995. *Open Distributed Processing - Reference Model Part 1-4.* ISO/IEC 10746-1..4 :1995, ITU-T 901..4, 1995.

ISO, 2006. *Information Technology - Open Distributed Processing - Use of UML for ODP system specifications.* Final Committee Draft. ISO/IEC JTC1/SC7 WG19.

Larman, 2001. *Applying UML and Patterns: An introduction to object-oriented analysis and design and design and the Unified Process, second edition.* Prentice Hall.

Leffingwell, D., Widrig, D., 1999. *Managing Software Requirements, a unified approach.* Addison Wesley.

OMG, 2006. *OMG System Modeling Language (OMG SysML) Specification.* OMG 2006.

OMG, 2005. *Unified Modeling Language: Superstructure, version 2.0.* OMG 2005.

Picault, A., Bedu, P., Le Delliou, J., Perrin, J. , Traverson, B., 2004. *Specifying Information System Architectures with DASIBAO - A standard based method.* 6th International Conference on Enterprise Information Systems. Porto, Portugal, April 2004.

Robertson, S., Robertson, J., 2006. *Mastering the Requirements Process, 2nd edition.* Addison Wesley.

Rumbaugh, J., Jacobson, I., Booch, G., 2004. *The Unified Modeling Language Reference Manual, second edition.* Addison Wesley.

Salome, 2007. *http://www.salome-platform.org/*.

Sommerville, I., Sawyer, P., 1997. *Requirements Engineering, a good practical guide.* John Wiley & sons.

Wiegers, K., 1999. *Software Requirements.* Microsoft press 1999.

Yahiaoui, N., Traverson, B., Levy, N., 2006. *Evolution management framework for multi-dimensional Information Systems.* 8th International Conference on Enterprise Information Systems. Paphos, Cyprus, August 2006.

SciTePress Science and Technology Publications