Adaptable Documentation and Traceability of Software Requirements

E. C. Genvigir[1 2] and N. L. VijayKumar[1]
[1]Laboratory for Computing and Applied Mathematics - LAC
Brazilian National Institute for Space Research - INPE
C. Postal 515 – 12245-970 – São José dos Campos - SP
BRAZIL
E-mail: elias, vijay@lac.inpe.br

[2] Federal Technological University of Paraná – UTFPR
Campus Cornélio Procópio
Av. Alberto Carazzai, 1640, CEP 86300-000 – Cornélio Procópio – PR
BRAZIL

Keywords: requirements, data analysis, metamodel, traceability.

This work presents a summary on the research conducted towards the development of a model to document and to visualize,dynamically, software requirements.

The research consisted of analyzing the necessary information focusing on documenting different types of requirements in an efficient way, and how quality factors may be part of those requirements. The research has also considered, the traceability (Aizenbud-Reshef et al, 2006) among the requirements and different types of artifacts developed during a software project.

The current solutions for the Requirement Engineering do not possess support for the inclusion or exclusion, of quality factors and elements to describe specific patterns of requirements as the use cases or forms of a certain project. The lack of that support makes the storage model and description of the static requirements difficult; in other words, those solutions hinder the definition of different types of requirements as well as the inclusion of attributes for the quality control of those requirements.

The existence of different types of requirements, as well as specific needs about quality are due to the inherent specifications to each project and the different characteristics of the processes used in the organizations during the development of those projects.

The study showed that the use of a flexible approach that allows the definition of new patterns of requirements with different characteristics, as well as the option to determine whether they should or not be demanded, constitutes a viable option for a solution to the problem. "Flexible", within the context of this research, points towards the capability of the model to understand to define, to register and to visualize dynamically new patterns defined according to the needs of the project, unlike a static model in that it possesses a pre-defined, nonmodified (ou melhor non-modifiable) storage mechanism.

As for the computational development of the model, it has been opted to use technologies that make use of hyperdocuments possessing the flexibility as their main feature, such as the definition of specific vocabularies in Extensible Markup Language - XML and the elements for dynamic visualization as Extensible Stylesheet Language for Transformation - XSLT.

For the support of the quality elements, some established standards were analyzed based on norms, such as IEEE-830 (IEEE,1998b) and IEEE-1233(IEEE,1998a). When considering IEEE-830, it was observed that the constant requirements in a specification of requirements of high qualitysoftware must be clear, complete, without ambiguity, implementable, consistent and testable. For those qualities to be achieved, the requirements should possess some of the following characteristics: required entered and exit information, restrictions, associated documents and others.

The analysis showed that not all the factors of quality of a requirement should necessarily be obligatory, because, depending on the characteristics of the project, the factor could or not be important. It was also observed that the creation of new norms or models associating new quality factors to the requirements could restrict a model in that

those factors could be defined in a static way. Finally, the first analysis also presented that the definition of a static model of data for the requirements created a large complexity in when dealing with a single XMLvocabulary.

The traceability was also treated in a flexible way, which contributed to the creation of the traces between the requirements and the different types of artifacts generated during the projectdevelopment. In this way, the application of the Software Process Engineering Metamodel - SPEM (OMG,2002), specifically the nucleus associated to the products of process work, allowed the creation of the necessary structure to register and later, to associate the several types of artifacts to the requirements using different types of traces.

After evaluating SPEM, a data model was elaborated to solve the problem of t registration of the artifacts.

When analyzing the artifact class diagram , it was observed that SPEM presented a solution for the three problems observed initially in the work: the quality factors do not need to be obligatory; enabling the inclusion of new norms or demands associated to the requirements; and the XML vocabulary used is simplified.

The adopted solution simplified the data model in more generalized classes in which an element of SPEM, such as WorkProduct, may be a class diagram, a use case, a code source or any type of work product that one may want to create or to use. That solution type allows flexibility as to which elements are to be used in a given project, which also facilitates the creation of a single XML vocabulary in which its items are the same, however, with distinct instances .

After the analysis, the model shown in Figure 1 has been created.

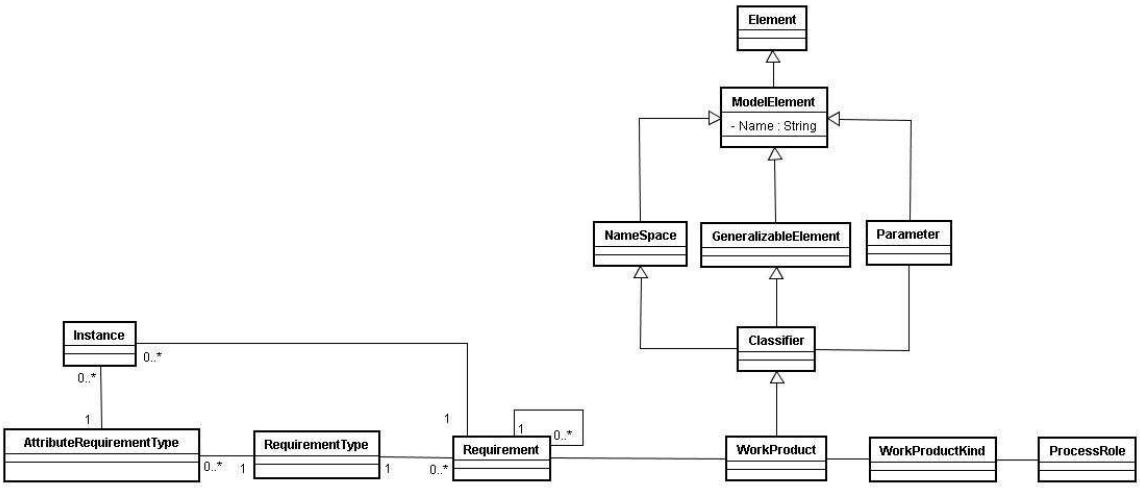

Figure 1. Class diagram.

In the obtained solution, Figure 1, the Requirements are defined by a RequirementType which, in turn, possesses AttributeRequirementType. The AttributeRequirementType possesses Instances that are the values instanced for a certain requirement. As the instances are the values of the attributes of a given requirement, the instances need to be associated to the requirement to which they belong. The existence of the association between the instance class and the requirement class is due to that fact.

It can also be observed, in Figure 1, that a Requirement may be associated to one or more WorkProduct (artifacts), whose association is the one that allows the traceability between requirements and artifacts.

The implementation of this model turned into a tool which allows registering different types of requirements, artifacts and traces as well as allowing their visualization in a dynamic way. However, other elements related to the requirements traceability process, as the automation of the process, the impact of changes and the evolution of the software, are focused for the continuity of this research. So, the presented solution is a preliminary result of a proposal that should understand the control on the registration, the documentation and the traceability of software requirements.

REFERENCES

Institute of Electrical and Electronics Engineers - IEEE Std - 1233 IEEE - *Guide for developing system requirements specifications*, New York, 1998a.

Institute of Electrical and Electronics Engineers - IEEE Std 830-1998 IEEE *Recommended practice for software requirements specifications*. New York, 1998b.

Object Management Group, *Software process engineering metamodel specification* (SPEM), Formal Submission, OMG document number formal/02-11-14, November 2002.

Aizenbud-Reshef, N. and Nolan, B.T. and Rubin, J. and Shaham-Gafni, J. (2006), *Model traceability*, IBM Systems Journal, 45 (3), 515-526.