Requirements Patterns for COTS Systems

Oscar Mendez-Bonilla
Dept. LSI. UPC
Barcelona, Spain
omendez@lsi.upc.edu

Xavier Franch
Dept. LSI. UPC
Barcelona, Spain
franch@lsi.upc.edu

Carme Quer
Dept. LSI. UPC
Barcelona, Spain
cquer@lsi.upc.edu

Abstract

The reuse of knowledge obtained during the elicitation of requirements for different COTS projects is a subject that still needs more research to be done. In this work we propose the use of Patterns of Requirements for the first steps of the COTS selection processes and the software life cycle.

1. Introduction

The idea of giving help to the requirements elicitor in its work is not new. Among the existent approaches, we can point out the Volere shell [1] that provides the elicitor with common sections that usually appear in a requirements document, serving the elicitor to do not forget any of these sections. However, we think more help could be given to unify, standardize and create useful requirements specifications. In this paper we present our proposal of having requirement patterns that could be applied to different projects.

The existence of requirements patterns would help the elicitors to do not start from scratch and to reduce the time necessary for the requirements elicitation and the creation of effective requirement specifications, which hold the entire software cycle. The use of these patterns in particular projects would consist in the identification of the patterns and its instantiation to the concrete project; generating the requirements specifications and the bases for the call for tenders documents.

2. A catalogue of requirements patterns

The requirements in a software project may be classified [2] as: functional, non-functional, and non-technical requirements. After working in different projects corresponding to different domains (e.g. mail server systems, e-learning software, web content management systems, and others), we observed that most non-functional and non-technical requirements can be reused with small variations independently of the system's domain for which the elicitation is done.

This does not occur in case of functional requirements, that are quite similar when projects address the same domain, and as more distance exists among domains, less percent of requirements are similar in projects.

For this reason, we propose (figure 1) the existence of a general catalogue of non-functional and non-technical requirements patterns that will be common and useful for any project, and other specific patterns catalogues that would be suitable in projects for specific domains.

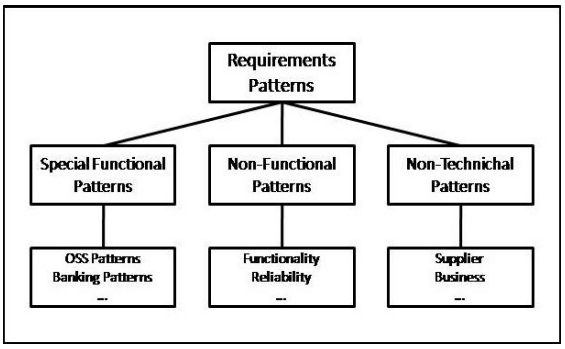

Figure 1. Requirements Patterns catalogs

As in any kind of pattern, our idea is the construction of requirement patterns from the experience in many projects, from the observation of similarities in requirements already used. When exist similar requirements that state the same goal, and this is identified in different software or COTS selection projects, we propose to observe and study if there exists patterns that generalize them. Our intention is to obtain a catalogue of patterns from requirements documents of finished projects; we plan to use this first version catalogue, generating the original requirements documents in an inverse process, and then test the

0-7695-3091-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCBSS.2008.34

catalogue usefulness with starting COTS selection projects, and with other kind of software development and life cycle.

We do not think on closed catalogues, but in evolving catalogues that will grow from the experience of past and new projects. In the patterns life cycle, new ones will arrive from new necessities and desires of the stakeholders, and this ones will be evaluated in every use of the catalog.

3. The requirement patterns

3.1. The structure of a pattern

The structure of a pattern will be the following:

- The pattern goal, which will drive the elicitor in the selection of the patterns to be applied in the project. The goal will also be used in a goal-based dependency network, which will help the elicitor in the selection and validation of new requirements, and with the generation of new ideas about the relationships between patterns.
- A pattern description, a short text that introduce the pattern, in the sense of an abstract.
- A fixed part that will be necessary to be included in any application of the pattern. A sentence that express the pattern itself, the general purpose of the pattern and his identification.
- Several extensions of the general part, which is the technical body of the pattern. These extensions are optional, since will not be necessary in any case where the general part is applied. They could also be included in the catalogues as different patterns. However we think it is better to have them together since all of them contribute to express a same goal for the COTS component that will be selected, or the necessity or requirement to be fulfilled.

Both, the fixed part and the extensions are composed of text that express the whole pattern, plus a set of parameters that will take specific values during the pattern application in a project.

In order to state clearly the semantics of the parameters, a metric will be defined for each parameter. We care that each metric accomplish the requisites to exist and be considered metric, and also we express the metric name, type and correctness conditions.

As it occurs in requirements documents and specifications, some requirements depend on others, in purpose or in their definitions. There are relationships among requirements patterns. We will take one of the existent relation scales proposed in the literature in order to classify these dependencies.

We present an example of one non-functional requirement pattern in figure 2.

Pattern Goal: Reporting system failures.
Pattern description: This pattern expresses the necessity of the stakeholders to know when a system crash or failure happens, independently of the stakeholder role. Is required to take into account the new information system needs about reporting transactions and operations. The options of the pattern express different ways of alerting to responsible persons. Other pattern to take into account is: After crash report.
Fixed part: The solution shall give an alert in case of a system failure.
Extension part1: The alerts shall be of type **AL**.
Extension part2: The failures that shall produce the alert shall be of type **FL.**
Parameters:
AL: is a non-empty set of valid alerts
AL = SetOf (**Alert**)
Alert = Nominal [E-mail, SMS, Page, Fax, ...]
Correctness conditions: SizeOf(**AL**) > 0

FL: is a non-empty set of valid failures
FL = SetOf (**Failure**)
Failure = Nominal [Low space (hardware, software), Crash (server, network), ...]
Correctness conditions: SizeOf(**FL**) > 0

Classification:
Functionality –
Suitability –
Fault Suitability –
Crash Suitability
and
Reliability –
Maturity –
Robustness –
Operation robustness

Figure 2. Requirement pattern example

3.2. The classification of patterns

In order to classify the patterns in a catalogue, we propose to use the quality features in the Extended ISO/IEC 9126-1 quality model [3].

The *Extended ISO/IEC 9126-1 quality model* is an extension of the ISO/IEC 9126-1 quality model [4] that includes the decomposition of some subcharacteristics of the original ISO model (60 new sub characteristics and attributes), and includes a complete catalogue of non-technical features (134 new characteristics, sub characteristics and attributes). The quality features in the extension (characteristics, subcharacteristics and

attributes) are general to any business application software.

When the requirements patterns will be classified it, is possible that we will need to extend the departing Extended ISO/IEC quality model with new features. This is an aspect already considered during the extended quality model proposal [3]. This is what has occurred during the classification of the pattern in figure 2, when we wanted to classify them in the *suitability* quality feature; we needed to extend this subcharacteristic with two more levels of subcharacteristics, which are *fault suitability* and *crash suitability*.

On the other hand, it is important to point out, that one requirement pattern can be classified in more than one place in the extended quality model. This has also occurred with our pattern example that could also be classified below *Reliability – Maturity – Robustness – Operation robustness*.

4. Requirement pattern application

Our idea is that the requirements patterns could be used in software development processes, e.g. in an automatic conceptual model based software development, or in a traditional software life cycle. The idea of have structured, pre-builded necessities is to base the software design in this necessities customized to the stakeholders, that is, software customized to the measure of the stakeholders desires. The patterns can also be useful in commercial components selection processes, especially in the first steps of COTS selection, which means the definition of software characteristics, required for stakeholders, this definition could be linked to COTS software attribute taxonomies for an accelerated selection process. The patterns are also useful in the delicate step of call for tenders, helping to evaluate the different options from the tenders, against the requirements patterns selected by the stakeholders, the patterns could also help to generate quickly, the first documents with all the characteristics required in a call for tenders document, and then could also help to validate the offers.

5. Conclusions and future work

The existence of tool support would be important. We currently plan to define requirements patterns by means of DesCOTS-QM [5], a tool developed by our research group, which already provides a requirements patterns manager, and also allows its classification by means of an ISO/IEC 9126-1 based quality model.

We are developing a complete catalog of patterns, based in postmortem analysis of requirements books from real software projects, which will give us our initial catalogue of non-functional requirement patterns. After this, we will validate the patterns during the elicitation of requirements of upcoming projects.

6. References

[1] Robertson, S., and Robertson, J., *Mastering the Requirements Process*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1999.

[2] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.B., *Non-Functional Requirements in Software Engineering*, Springer-Verlag, New York, 1999.

[3] J.P. Carvallo, X. Franch, C. Quer, "Defining a Quality Model for Mail Servers", *2nd International Conference on COTS-Based Software Systems (ICCBSS), LNCS 2580*, Ottawa Canada, 2003.

[4] ISO/IEC 9126-1-2001 *Software Engineering Product Quality Part 1: Quality Model*, International Standards Organization, Geneva Switzerland, 2001.

[5] Juan Pablo Carvallo, Xavier Franch, Gemma Grau & Carme Quer. *"QM: A Tool for Building Software Quality Models". In* Proceedings of the 12th IEEE Requirements Engineering International Conference *RE 2004.Kyoto, Japan. IEEE Computer Society. 2004.*