Non-Functional Requirements and UML Stereotypes

Guadalupe Salazar-Zárate
Technical University of Catalonia, Spain

Pere Botella
Technical University of Catalonia, Spain

INTRODUCTION

In Nuseibeh and Easterbrook (2000), an overview of the field of software and systems requirements engineering is presented. Therein is highlighted some key open-research issues for the future of the Requirements Engineering (RE). Some of the major challenges mentioned there, are related with the necessity of richer models for capturing and analyzing non-functional requirements. This paper draws some possible extensions of Unified Modeling Language (UML) (Booch, G., Jacobson, I. and Rumbaugh, J., 1998) in order to include non-functional requirements.

BACKGROUND

Within the Requirements Engineering processes (e.g., domain analysis, elicitation, modeling, validation, etc.), it is common to distinguish between functional and non-functional requirements. The relevance of functional requirements has been traditionally well-covered by the existing modeling techniques, where a lot of research has been done. However, non-functional requirements (NFRs for short), quality and constraint properties, are not usually covered by these modeling techniques. On the contrary of its functional counterpart and despite the critical role they have during system development, they have received little attention in the literature as mentioned by Chung, L., Nixon, B.A., Yu, E. and Mylopoulos, J. (2000). How can we model and reason about NFRs? How can these models be integrated with those modeling techniques? These are still some of the key challenges that need more research.

Functional and non-functional aspects regarding the external system behavior involve two different ways of evaluating and/or developing a given software system. On one hand, functional aspects are directly connected to *what the system does*, for example, the basic functions that a system (or a system component) must provide. On the other hand, non-functional aspects are related with how the system behaves with respect to some observable attributes such as performance, reliability, efficiency, reusability, portability, maintainability (i.e., some software qualities). To illustrate the wide variety of issues for software quality, in Chung et al. (2000, p.160) a list of non-functional requirements can be found.

There are two basic approaches to characterize non-functional requirements: *Product-oriented* and *Process-oriented* (Chung et al., 2000). The *product-oriented approach* basically focuses on the development of a formal framework so that a software product can be evaluated in relation to the highest degree it fulfills for non-functional requirements (constraints over non-functional properties). The *process-oriented approach* uses non-functional information to guide the development of software systems. Among the works dealing with this perspective of non-functionality, those of Chung et al. (2000) are without any doubt the most complete ones. Therein a NFR framework, to deal with diverse non-functional requirements to drive the design by justifying decisions during the software development process, is described. The framework also offers structured, graphical facilities for stating, managing, and inter-relating non-functional requirements while justifying decisions and determining their impact throughout the development process.

In Cysneiros and Leite (2001a, 2001b), and Cysneiros, L.M., Leite, J.C.S.P. and Neto, J.S.M. (2001) an approach that complements the work reported in Chung et al. (2000) is presented. Strategies can be found there that are concerned with the problem of how to identify and integrate non-functional requirements with functional requirements, in a process-oriented approach and using UML.

Another important language that focuses on non-functional requirements is the Goal-oriented Requirement Language (GRL) (ITU-T, URN Focus Group, 2002b), (http://www.cs.toronto.edu/km/GRL/). The GRL graphical language is used to support goal and agent-oriented modeling and reasoning of requirements. The GRL is built on the well-established NFR Framework (used for modeling NFRs) and the agent-oriented language i* (Yu, 1997) (used for the modeling, analysis, and reengineering of organizations and business processes).

Copyright © 2005, Idea Group Inc., distributing in print or electronic forms without written permission of IGI is prohibited.

OUR APPROACH TO NON-FUNCTIONAL REQUIREMENTS AND UML

UML has today become a standard modeling language for software systems development (Object Management Group, 2001). UML offers a graphical notation to create models. However, it is mainly focused on functional aspects of the software development.

In the following approach we outline some possible extensions of UML in order to include non-functional requirements. In the works by Botella, P., Burgués, X., Franch, X., Huerta, M. and Salazar, G. (2002), and Salazar-Zárate, G, Botella, P. and Dahanajake A. (2003), more detailed considerations about this topic can be found.

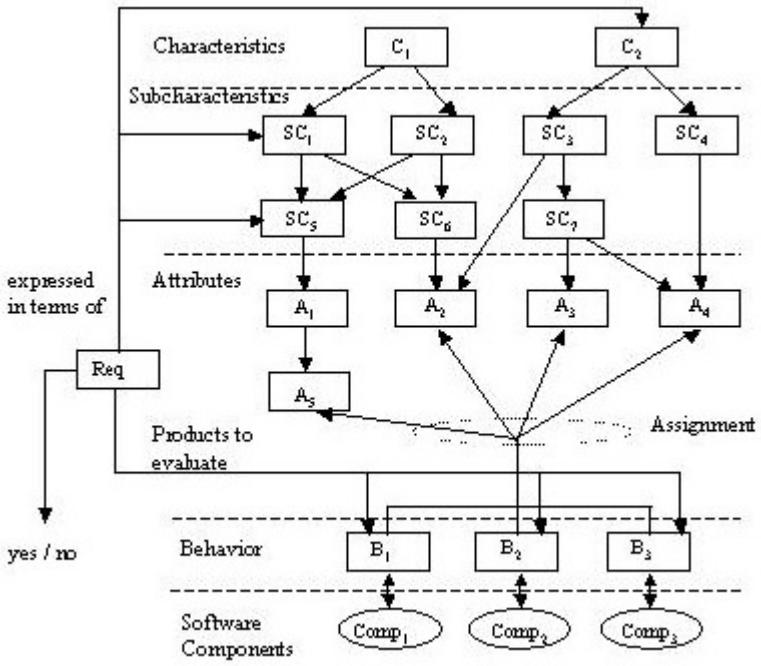
The International Standard ISO/EC 9126-1 (2001) can be used as a starting point to identify non-functional attributes of products that are potentially relevant to be modeled in a software development process. This standard provides a framework for software product quality, specifying quality characteristics to be used in the quality evaluation of software products. A *quality model* is defined by means of general *characteristics of software*, which are further refined into *subcharacteristics* in a multilevel hierarchy. In the standard, the characteristics of functionality, reliability, usability, efficiency, maintainability, and portability are being placed at the top of the hierarchy. Measurable *software attributes* appear at the bottom of the hierarchy. *Software quality metrics* allows developers to quantify up to what degree a software system meets non-functional requirements.

A language called *NoFun* (acronym for "NOn-FUNctional") is a notation that focuses on representing non-functional aspects of software systems, at the product level within the component-programming framework (Franch, 1998; Burgués & Franch, 2000; Botella et al., 2002). It is a formal language for description of software quality requirements using the ISO/IEC 9126 standard to summarize quality characteristics (see figure 1). Although the ISO/IEC 9126-1 (2001) replaces the previous version of 1991, the basic principles stay and we are still able to use the same layout therein described.

To achieve the goal of formalization within *NoFun*, three different kinds of capabilities are provided. First, there are modules for defining the different kind of concepts in the standard (characteristics, subcharacteristics and attributes). Second, values for these attributes may be given (in *behavior modules*) and bound to particular software components (i.e., the ones under evaluation). Third, additional constructions for representing quality requirements and assessment criteria are included.

In figure 1, an example of distribution of a quality model into modules is shown (extracted from Botella et al. (2002). Two characteristics defined in terms of four subcharacteristics appear in the upper part of the figure. Behavioral modules (denoted by the B_i), are abstractions of software components in the sense that they contain all the relevant information for quality evaluation. Quality requirements may be defined as restricting the values of

Figure 1. Layout of a quality model under NoFun language.



4 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/non-functional-requirements-uml-stereotypes/14573

Related Content

The Effect of Level of Negotiation Support Systems and Cultural Diversity on Coalition Formation: A Content Analysis
Xiaojia Guo, John Limand Fei Wang (2008). *Information Resources Management Journal (pp. 84-96).*
www.irma-international.org/article/effect-level-negotiation-support-systems/1352

Design of a Public Vehicle Tracking Service Using Long-Range (LoRa) and Intelligent Transportation System Architecture
Ricardo Salazar-Cabrera, Álvaro Pachón de la Cruzand Juan Manuel Madrid Molina (2021). *Journal of Information Technology Research (pp. 147-166).*
www.irma-international.org/article/design-of-a-public-vehicle-tracking-service-using-long-range-lora-and-intelligent-transportation-system-architecture/271412

Perspectives of Transnational Education
Iwona Miliszewska (2009). *Encyclopedia of Information Science and Technology, Second Edition (pp. 3072-3079).*
www.irma-international.org/chapter/perspectives-transnational-education/14028

An Intranet within a Knowledge Management Strategy
Udo Richard Averweg (2009). *Encyclopedia of Information Science and Technology, Second Edition (pp. 2221-2226).*
www.irma-international.org/chapter/intranet-within-knowledge-management-strategy/13889

Development of M-Government Projects in a Developing Country: The Case of Albania
Silvana Trimiand Kozeta Sevrani (2010). *International Journal of Information Technology Project Management (pp. 46-58).*
www.irma-international.org/article/development-government-projects-developing-country/46107