Dealing with Completeness in Requirements Engineering

Graciela D. S. Hadad
FITI, Universidad de Belgrano, Argentina & DIIT, Universidad Nacional de La Matanza, Argentina

Claudia S. Litvak
FITI, Universidad de Belgrano, Argentina & DIIT, Universidad Nacional de La Matanza, Argentina

Jorge H. Doorn
INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina & DIIT, Universidad Nacional de La Matanza, Argentina

Marcela Ridao
INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

INTRODUCTION

The Requirements Engineering (RE) goal is to systematize the process of requirements construction and management (Maculay, 1993; Reubenstein & Waters, 1991; Maté & Silva, 2005) along with the creation of a compromise among clients and users with developers, since both human groups must participate and collaborate together. To accomplish such tasks, the requirements engineers should understand and participate in the definition of the context of use for the software system to be developed. The requirements engineers usually ignore totally or partially both, the current and the foreseen future context of use. The latter must be conceived having as a resource the new tool: the system software itself. Frequently, nobody knows in detail such future context of use. To be able to participate in the definition of the future business process, the requirements engineers must understand the current business process in advance. Therefore, the Requirements Engineering process involves, as the first step, elicitation and modeling of the current business process and later, definition and modeling of the future business process. Both models have different purposes. The first one is used as a help for understanding the current business process and as a tool to validate with clients and users such understanding. The second one is used as a help for the planning of the future business process, to validate such plans with the clients and users, to specify the software requirements and to give an environment reference for software designers.

The Requirements Engineering process consists of three main activities: elicitation, modeling, and analysis of the application domain (Kotonya & Sommerville, 1998; Sommerville & Sawyer, 1997). Analysis includes the sub-activities of verification, validation and negotiation. The difficulties that requirements engineers must face to understand and elicit the clients and users' necessities are widely known. The more complex the application domain, the more difficult the definition or construction of software requirements becomes. Many times, requirements engineers must become themselves problem domain experts during the acquisition of knowledge about the application domain. Concurrently, Requirements Management deals with the changes in the existent requirements and the irruption of new ones (Kotonya & Sommerville, 1998; Sawyer & Kotonya, 2004).

RE provides methods, techniques and tools to help requirements engineers elicit and specify software requirements, ensuring their highest quality and completeness. However, the problem of completeness is a certain menace to requirements quality and it casts serious doubt on the whole Requirements Engineering process. Completeness is an unreachable goal and to estimate the degree of completeness obtained at a certain step in the software development process is also very difficult (Doorn & Ridao, 2008). The requirements engineer faces a Universe of Discourse (UofD) that

DOI: 10.4018/978-1-4666-5888-2.ch279

seems she / he will hardly ever fully know. This situation is not unique during the whole software development process; something similar happens while testing.

The use of statistical models, based on capture and recapture methods (Otis, Burnham, White, & Anderson, 1978) to predict the number of defects in a code artifact, was successfully introduced some time ago (Petersson, Thelin, Runeson, & Wohlin, 2003) and later, these models have been used to discover defects in requirements documents (Walia & Carver, 2008). In this article, the use of capture and recapture information is applied in the RE field in order to make an estimation of the number of non-discovered requirements after a requirements elicitation process.

The following section analyses the validation problem in RE. Then, a section describing the use of the Language Extended Lexicon (LEL) and Scenarios in RE is included. After that, the problem of estimating closed population is studied. Later, the use of capture and recapture in RE domain is introduced; finally, some future work and conclusions are presented.

BACKGROUND

No method in Software Engineering can ensure that enough information has been elicited and modeled to develop a software system that covers all the expectations and necessities of clients and users. Incompleteness negatively influences the quality of any produced artifact, such as a requirements model, a design model or a software component. The completeness problem in Software Engineering and Requirements Engineering is very similar to others in many areas of knowledge. Otis (Otis, Burnham, White, & Anderson, 1978) introduced a method to estimate the size of a closed population of wild animals based on the data gathered during repetitive capture of specimens. This method has been extended to the area of software inspections by several authors (Briand, El Emam, Freimut, & Laitenberger, 2000; Biffl, 2003; Thelin, 2004; Petersson, Thelin, Runeson, & Wohlin, 2003; Wohlin & Runeson, 1998) in order to estimate the number of undiscovered defects.

Requirements validation has become a complex task, mainly due to the kind of representation models used which require clients and users with special skills to understand them. As it is pointed out by several authors (Sommerville & Sawyer, 1997; Cysneiros & Yu, 2004), the requirements validation seldom discovers all defects, and the remaining defects reach later stages in the software development process. It has been proven that the use of natural language representation for requirements helps validation, especially when requirements are expressed using the client and user's vocabulary (Leite & Franco, 1990). To be able to provide such representation, the requirements engineer should acquire the clients and users' vocabulary. However, ambiguity is the main drawback of the natural language approach (Jackson, 1995; Sommerville & Sawyer, 1997; Berry & Kamsties, 2004). The construction of a glossary of clients and users' jargon helps reduce ambiguity and build the requirements specification in an ´understandable´ language, mainly for clients and users. Several experiences have shown that a glossary of clients and users' vocabulary is, in itself, a source of information to elicit valuable UofD information (Ben Achour, Rolland, Maiden, & Souveyet, 1999; Rolland & Ben Achour, 1998; Oberg, Probasco, & Ericsson, 1998; Regnell, 1999; Weidenhaupt, Pohl, Jarke, & Haumer, 1998).

In this entry, a Requirements Engineering process which begins with the construction of a Language Extended Lexicon (LEL) as its first activity (Leite, Doorn, Kaplan, Hadad, & Ridao, 2004) is addressed in order to analyze the impact of completeness. In this process, the LEL construction is followed by the building of Scenarios to understand and model the current UofD, and later, by the building of another set of Scenarios to figure out how the future UofD could be and to model it. Finally, this process ends extracting requirements from the latter set of Scenarios and producing a Software Requirements Specification.

THE REQUIREMENTS ENGINEERING PROCESS

The backbone of the Requirements Engineering process is to anchor every model in the UofD vocabulary. Knowledge acquired by means of observations, document reading, interviews and other elicitation techniques is first modeled using LEL and later using Scenarios (Leite, Hadad, Doorn, & Kaplan, 2000). LEL and Scenarios are verified for internal consistency and validated with the collaboration of clients and users. During Verification and Validation, completeness is a key issue since several guidelines for LEL and Sce-