C

Completeness Concerns in Requirements Engineering

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

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

INTRODUCTION

The difficulties that software developers must face to understand and elicit clients' and users' necessities are widely known. The more complex the context of the problem, the more difficult the elicitation of software requirements becomes. Many times, requirements engineers must become themselves problem-domain experts during the acquisition of knowledge about the context of the application. The requirements engineering (RE) objective is to systematize the process of requirements definition (Maculay, 1993; Maté & Silva, 2005; Reubenstein & Waters, 1991) along with creating a compromise among clients and users with developers since they must both participate and collaborate together. The requirements engineering process consists of three main activities: elicitation, modeling, and analysis of the application domain (Kotonya & Sommerville, 1998; Sommerville & Sawyer, 1997). Later, requirements management deals with the changes in the requirements and the irruption of new ones.

RE provides methods, techniques, and tools to help requirements engineers elicit and specify requirements, ensuring their highest quality and completeness. However, the problem of completeness is a certain menace to requirements quality and casts a serious doubt on the whole RE process. Completeness is an unreachable goal, and to estimate the degree of completeness obtained at a certain step in the project is even more difficult. The requirements engineer faces a universe of discourse (UofD) that he or she will hardly ever fully know. This situation is not unique during the whole software development process since something similar happens while testing.

The use of statistical models to predict the number of defects in a software artifact was successfully introduced some time ago (Petersson, Thelin, Runeson, & Wohlin, 2003). 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 undiscovered requirements after a requirements elicitation process.

The following section analyses the validation problem in RE. Then, a section describing the use of LEL (language-extended lexicon) and scenarios in requirements engineering is included. After that, the problem of estimating closed populations is studied. Later, the use of capture and recapture in the RE domain is introduced, and finally, some future work and conclusions are presented.

BACKGROUND

The completeness problem in software engineering and requirements engineering is very similar to others in many knowledge areas. Otis, Burnham, White, and 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 (Biffl, 2003; Briand, El Emam, Freimut, & Laitenberger, 2000; Thelin, 2004; Petersson, Thelin, Runeson & Wohlin, 2003; Wohlin & Runeson, 1998).

Requirements validation has become a complex task mainly due to the kind of representation models used, which requires clients and users with special skills to understand them. As it is pointed out by several authors (Cysneiros & Yu, 2003; Sommerville & Sawyer, 1997), the requirements validation seldom discovers all defects, which may 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 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 (Berry & Kamsties, 2004; Jackson, 1995; Sommerville & Sawyer, 1997). The construction of a glossary of clients' and users' jargon helps reduce ambiguity and build requirements specification in an understandable language. Several experiences have shown that a glossary of the clients' and users' vocabulary is, in itself, a source of information to elicit valuable UofD information (Ben Achour, Rolland, Maiden, & Souveyet, 1999; Oberg, Probasco, & Ericsson, 1998; Regnell, 1999; Rolland & Ben Achour, 1998; Weidenhaupt, Pohl, Jarke, & Haumer, 1998).

In this entry, an RE process that begins with the construction of an LEL as its first activity (Leite, Hadad, Doorn & Kaplan, 2000) is addressed in order to analyse the impact of completeness in it. In this process, the LEL construction is followed first 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 with the set of requirements of the software system to be developed.

THE PROCESS

The backbone of the process is to anchor every model in the UofD vocabulary. Knowledge acquired by means of observations, document reading, interviews, and other techniques is first modeled using LEL and later by means of scenarios (Leite et al., 2000). LEL and scenarios are verified for internal consistency and validated with the collaboration of clients and users. During verification and validation (V&V), completeness is a key issue since several steps during LEL and scenario inspections (Leite, Doorn, Hadad & Kaplan, 2005) and also some guidelines for validation are designed with completeness as their main target. However, this is far from being enough.

Language Extended Lexicon

Most relevant or peculiar words or phrases (named LEL symbols) of the UofD are included in the LEL. Every symbol is identified by its name (including synonyms) and by two descriptions: notion and behavioral response. The notion contains sentences defining the symbol and the behavioral response reflects how it influences the UofD. Figure 1 depicts the model used to represent LEL symbols.

Scenarios

Scenarios are used to understand the UofD first, and later to understand the problem and its functionality. Each scenario describes a specific situation of the UofD focusing on its behavior. The scenario model (see Figure 2) contains the following components: title, goal, context, resources, actors, episodes, and exceptions.

A scenario must satisfy a goal, which is reached by performing its episodes. Episodes represent the main course of action, but they may include variations or possible alternatives. Actors carry out episodes using resources. While performing episodes, an exception may arise, signaling an obstacle to goal achievement. The context is described detailing a geographical location, a temporal location, or preconditions. Context, resources, and episodes may have constraints, which are used to record restrictions of any kind. Constraints are used to characterize nonfunctional requirements.

Figure 1. Language-extended lexicon model

LEL: LEL is the representation of the symbols in the language of the application domain.
Syntax: $\{\text{Symbol}\}_1^N$

Symbol: It is an entry of the lexicon that has a special meaning in the application domain.
Syntax: $\{\text{Name}\}_1^N + \{\text{Notion}\}_1^N + \{\text{Behavioral Response}\}_1^N$

Name: It is the identification of the symbol. Having more than one name represents synonyms.
Syntax: Word | Phrase

Notion: It is the denotation of the symbol.
Syntax: Sentence

Behavioral Response: It is the connotation of the symbol.
Syntax: Sentence