Creating Software System Context Glossaries

Graciela D. S. Hadad
Universidad Nacional de La Matanza, Argentina & Universidad Nacional de La Plata, Argentina

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

Gladys N. Kaplan
Universidad Nacional de La Matanza, Argentina & Universidad Nacional de La Plata, Argentina

INTRODUCTION

Requirements engineering (RE) is the area of software engineering responsible for the elicitation and definition of the software system requirements. This task implies joining the knowledge of the services that a software system can and cannot provide with the knowledge of clients' and users' needs (Jackson, 1995; Katasonov & Sakkinen, 2005; Kotonya & Sommerville, 1998; Sommerville & Sawyer, 1997; Sutcliffe, Fickas, & Sohlberg, 2006; Uchitel, Chatley, Kramer, & Magee, 2006). Frequently, this activity is done by people with a software engineering bias. The underlying hypothesis of this choice is that users' needs are easier to understand than the software's possible behaviors. This is not always true; however, this is the metacontext in which most RE heuristics and methodologies have been developed. Understanding clients' and users' needs is far more complex than merely interviewing selected clients and user representatives, compiling all gathered information in one document. Defining how to put into service a complex software system within an organization requires envisioning how the business process of the organization will be in the future from both points of view: software organization and business organization. This is the key of the RE commitment: to imagine how the future business process will be. This RE commitment requires a good knowledge about how the business process actually is. Understanding the software system's preexistent context basically means understanding the clients' and users' culture. In other words, this part of the RE is a learning process.

BACKGROUND

The importance language has in any culture should be noticed. Language is an organized system of speech by which people communicate with each other with mutual comprehension. Also, it is very important to note that by the words it contains and the concepts it can formulate, language is said to determine the attitudes, understandings, and responses in any society. Language, therefore, may be both a cause and a symbol of cultural differentiation (Fishman, 1999; Hall, Hawkey, Kenny, & Storer, 1986). Language reflects environment and technology: Arabic has 80 words for camels, while Japanese has more than 20 words for rice and Inuit has more than 20 words for snow and ice (Nettle & Romaine, 2000). Clients and users have several special words that they use when discussing their activities. The requirements engineer must pick and understand as many of these words as possible as a first step in understanding clients' and users' culture.

Glossaries have been used in software engineering with different purposes, such as data dictionaries in early database books (Codd, 1982) to document entities, attributes, relations, types, and services of databases. Thus, it provides a common understanding of all the system names to the developer team and later to the maintenance team. However, data dictionaries are also an important component of structured analysis, data recording, data storage, and the details of processes (Gane & Sarson, 1982; Senn, 1989), though authors like Gane and Sarson suggest that the real name for them should be *project guide* instead of *data dictionary*. These data dictionaries are created during analysis and also used during system design. They satisfy five objectives: to manage details, communicate common meanings, document system characteristics, help the analysis of details and changes, and locate errors and omissions of the system.

In this article an RE process beginning with the construction of a language-extended lexicon (LEL) as its first activity (Leite, Hadad, Doorn, & Kaplan, 2000) is addressed, and the structure and creation of this LEL is described.

GLOSSARY CREATION

The word *dictionary* was coined by Henry Cockeran in 1623, but the first known dictionaries belong to the seventh century B.C., and they contain the most important data of

the Mesopotamian culture (MSN Microsoft Corporation, 2006). The first dictionaries were catalogues of unusual, difficult, or confusing words and phrases since the common vocabulary was considered to have no need of an explanation or a definition. The oldest glossary comes from the second century A.D. and contains technical Greek words used by Hypocrites. It was in later centuries that a catalogue of all the words of a language was built: for Arabic. So, the origin of glossaries and dictionaries was to give definitions of words and phrases of a particular domain; then, they were extended to an entire language in lexical dictionaries. Nowadays, there are different types of dictionaries covering different necessities, like dictionaries of synonyms and antonyms, dictionaries of idiomatic usage, etymology dictionaries, encyclopedic dictionaries, bilingual dictionaries, glossaries in textbooks, dictionaries of ideologies, slang dictionaries, and dictionaries of neologisms, among many others.

Most relevant or peculiar words or phrases (named LEL symbols) of the universe of discourse (UofD) are included in the LEL. Every symbol is identified by its name (including synonyms) and 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.

The LEL is created by filling the blanks in the LEL model (see Figure 1) using information obtained from the application domain. Intuition, supported with a good understanding of the LEL model, may be used to create the lexicon. Upon the experience and the skill of the authors, this may or may not lead to a well-conceived document. If so, apparently there is no need for heuristics. On the contrary, heuristics are needed, first to allow everyone to complete the process successfully and second to avoid weaknesses usually present in apparently good-quality LELs. Those weaknesses range from missing relevant symbols to the unnecessary insertion of some others, and the inclusion of excess of details in the symbol descriptions or the lack of them.

The lexicon creation process, depicted in Figure 2 using an SADT[1] model (Ross & Schoman, 1977), consists of five independent activities: (a) plan, (b) collect, (c) describe, (d) verify, and (e) validate.

As seen in Figure 2, the process shows a main stream composed of three tasks: plan, collect, and describe. There is a well-established feedback when the verification and validation activities take place. After verifying the LEL, the process returns to the *describe* activity, where corrections are made based on a DEO list.[2] After the *validate* activity, the process returns to the *collect* activity and/or the *describe* activity, depending on the validation DEO list, in order to make any necessary corrections. For easy reading, the SADT model does not show all the backtracking steps that may occur during the construction process. For instance, while describing a symbol, a wrongly assigned type may be discovered, thus a back step occurs in order to reclassify it (within the *collect* activity). Another example of going backward in the process could appear when a new term is identified while describing another. That is, the strategy is not at all a linear one. It is an iterative process where feedback is a constant mechanism. In addition to this continuous feedback, the main stream does not fully follow a cascade model since in practice its three main tasks may partially overlap. For instance, a symbol may

Figure 1. Language-extended lexicon model

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

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

Name: This 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