Category: Software & Systems Design

Handling Extemporaneous Information in Requirements Engineering

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

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

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

INTRODUCTION

The key of the success or failure of a software project depends on solving the right problem (Rumbaugh, 1994; Sawyer, 2005). Thus, software requirements should be correct, unambiguous, consistent, and as complete as possible (Institute of Electrical and Electronics Engineers [IEEE], 1998). Errors in requirements raise software development and maintenance costs notoriously (Katasonov & Sakkinen, 2006). The later the requirement error is detected, the higher the correction cost turns out to be. Error correction costs have been widely studied by many researchers (Bell & Thayer, 1976; Davis, 1993). Errors in requirements may be due to several reasons such as poor communication among requirements engineers, clients, and users; poor or nonexistent requirements validation; and the level of sternness of the models being used, especially to model relevant information captured from the universe of discourse (UofD).

Requirements engineering is the area of software engineering responsible for proposing and developing solutions to elicit, model, and analyze requirements by means of heuristics, guidelines, models, and processes which tend to requirements' completeness, quality, correctness, and consistency. Many proposals have been put forward by many researchers (Bubenko & Wrangler, 1993; Jacobson, Christerson, Jonsson, & Overgaard, 1992; Leite & Oliveira, 1995; Macaulay, 1993; Reubenstein & Waters, 1991).

BACKGROUND

Eliciting and modeling software requirements or related information are two different but highly related activities (Hull, Jackson, & Dick, 2005; Zowghi & Coulin, 2005). They may be coupled in several ways, being canonicals, such as in model-driven elicitation and elicitation-driven modeling. In the former, the requirements engineer tries to capture only the information that he or she needs for the model under construction. In the latter, the requirements engineer creates all models at the same time recording every piece of information gathered in the model it belongs to. Each of these approaches has advantages and drawbacks.

If the information is elicited for a given model, the requirements engineer pays attention only to some part of the things he or she is seeing, reading, or hearing. Then, he or she will discard any information that is not focus oriented. When the requirements engineer starts the creation of another model, he or she will change the focus and perhaps will now pay attention to information previously disregarded, provided that he or she comes across the same information. Unfortunately, this does not always happen, especially when the source of information is people. In other words, model-driven elicitation tends to make completeness difficult.

If all information obtained is registered at the same time, every model of the process is opened at the same time, and also, none are finished during an important period. A lack of coherence among models, poor understanding of the information gathered, and misplaced information are the main drawbacks of this approach. However, the loss of information is minimized.

Most researchers explicitly or implicitly prefer model-driven elicitation over elicitation-driven modeling (Leite, Hadad, Doorn, & Kaplan, 2005; Loucopoulos & Karakostas, 1995; Potts, Takahashi, & Antón, 1994). This means that model-driven elicitation has to deal with the risk of information loss. Looking closely into such risk, it can be seen that regardless of the elicitation technique, whether involving document reading, interviews, observation, or any other method (Goguen & Linde, 1993), the requirements engineer does not get exactly whatever he or she is looking for at any time in the information gathering activity (Faulk, 1996). He or she will need some of the information captured in the later stages of the process; however, he or she also needs some of that information in advance. In other words,

some of the information gathered is out of time (earlier or delayed). Dealing with the risk of loss of information means dealing with extemporaneous information (EI).

The research on which this chapter is founded was collected using a given process (Doorn, Hadad, & Kaplan, 2002; Leite et al., 1997; Young, 2004). However, the conclusions obtained can be applied to any requirements engineering process provided that it builds more than one model, having some degree of precedence among them.

In this article, the existence of extemporaneous information is studied and a proposal for its appropriated handling is given.

ADVANCED INFORMATION

Every phase of the requirements engineering process has its own specific objective. Although the requirements engineer tries to adhere to this objective, usually he or she comes across pieces of unexpected information that will be later necessary in the process. This advanced information (AI) may come from any source of information. However, most likely, AI comes from people.

Two main factors influence the quality and the quantity of Advanced Information in interviews.

Expectations on the new software: When the interviewed person is waiting for the new software to fix some organizational problem, he or she will be biased to describe his or her expectations even when the requirements engineer is trying to elicit knowledge about current business practices or even trying to build a glossary of the macrosystem. On the other hand, when the interviewed person does not have any expectation on the software system, he or she will not be an important source of AI.

Relative position in the organization: When interviewing a person in a relatively high position in the organization (directors, managers), the requirements engineer should be prepared to deal with abundant AI usually expressed as objectives and goals with an important degree of abstraction. On the other hand, operative people in the organization tend to provide less AI.

It should also be taken into account that AI may increase accordingly with the amount of changes planned for the business process after software installation. To estimate such amount of changes, the following factors should be carefully watched, among others.

Internal factors:

- Quality improvement projects
- Outsourcing projects
- Opening of new business lines
- Production technology changes
- Production volume increases
- Products or services changes
- Global organization objective changes

External factors:

- Sociocultural changes affecting the organization
- Economic changes affecting business
- Customer preferences changes

When accessing written sources of information, the risk to lose AI reaches a peak when the document has a moderate amount of AI. This becomes obvious considering that every document with much AI will be tagged to be read later in the process.

The requirements engineer cannot be in any way considered as a passive actor in the process of knowledge elicitation. He or she may usually have useful ideas about the context in which the software system will be involved in the future. On the other hand, it is very well known that short memory might be unfair to creative people in all activities. Sometimes, after having a good idea, the creator recalls just that: He or she has had a good idea. This falls into the paradigm of AI, although it was not originated in clients or users.

It is hard to determine at that early stage whether the elicited information or the requirements engineer's ideas are valuable or not. Perhaps they are the stub of what will be later an important requirement or perhaps they will become unplayable in the future context of the software system.

While model-driven elicitation is applied, the requirements engineer is trying to focus on his or her main current objective, and AI is actually felt as a disturbance; the risk is either to fail to collect the desired information or to lose the main track. The problem, visualized in this way, is very similar to visiting every node in a graph with bifurcations. Thus, the solution should also be very alike. The requirements engineer must keep attached to the main activity as much as possible, but at the same time he or she should find a way to register the minimum information needed to be able to follow the secondary track later.

When registering the minimum information needed to follow the secondary track, an ad hoc notepad could be used or a more structured document might be also chosen.

EXTEMPORANEOUS INFORMATION

In order to choose between a preplanned and an ad hoc registering strategy, it is valuable to quote Faulk (1996) who stated that the use of regular and predictable structures with limited information eases the comprehension and visualization of large quantities of information. This article summarizes the advantages of the use of a minimum card that allows registering the basic information required to