Advanced and Delayed Information in Requirements Engineering

Gladys N. Kaplan
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

INTRODUCTION

Requirements Engineering is the area of the Software Engineering responsible the development of the best set of requirements for a software system that will help in the solution of a problem or to improve some activity in the business process. It is important to stress the word *best* to clearly indicate that there are not a unique set of requirements for a given application context. In other words, the requirements engineer not only built a requirements set for a given context but frequently must choose among more than one variant. Requirements Engineering researchers are continuously proposing heuristics, guidelines, models and processes which tend to its completeness, quality, correctness and consistency. Many proposals have been put forward by many researchers (Jacobson, Christerson, Jonsson & Overgaard, 1992; Reubenstein & Watts, 1991; Bubenko & Wrangler, 1993; Macaulay, 1993; Leite & Oliveira, 1995).

Errors in requirements increment software development and maintenance cost notoriously (Katasonov & Sakkinen, 2006). The later, that a requirements error is detected, the higher the correction cost turns out to be. Incidence of error correction cost has been widely studied by many researchers (Davis, 1993; Bell & Thayer, 1976). Errors in requirements may be due to several reasons such as: poor communication among requirements engineers and clients and users, poor or lack of requirements validation, sternness of models being used, especially to model relevant information captured from the Universe of Discourse (UofD). A correct set of requirements for a software project becomes an important part of its success (Rumbaugh, 1994; Sawyer, 2005). Thus, software requirements should be correct, unambiguous, consistent and as complete as possible (IEEE, 1998).

BACKGROUND

The Requirements Engineering process has several activities which involve elicitation of knowledge and creation of one or more models to record it. Eliciting and modeling software requirements or related information are two highly related activities (Zowghi & Coulin, 2005; Hull, Jackson & Dick, 2005). They may be coupled in several ways, being canonicals: "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 to which it belongs. 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 what he or she is watching, reading or listening to. Then, he or she will discard any information which 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 with 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"

DOI: 10.4018/978-1-4666-5888-2.ch688

at the same time, and also, none is finished during an important period. 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 reduced.

Most authors, explicitly or implicitly prefer "model driven elicitation" over "elicitation driven modeling" (Potts,Takahashi & Antón, 1994; Loucopoulos & Karakostas, 1995) Leite, Hadad, Doorn & Kaplan, 2005]. 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 the elicitation technique, document reading, interviews, observation any or other (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 to deal with Extemporaneous Information (EI) (Kaplan, Doorn & Hadad, 2008).

The research in which this article is founded, was collected using a given process (Leite, Doorn, Kaplan, Hadad & Ridao, 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 entry, the existence of EI is studied and a proposal for its appropriated handling is given.

EXTEMPORANEOUS 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 with 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. Then in an interview becomes more visible the combination "Advanced Information - risk to lose it."

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 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 (Figure 1).

It should be also taken into account that AI may increase accordingly with the amount of changes planned for the business process after software installation.

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 itself 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 thinker 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 and 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 be also very alike. The requirements engineers