Chapter 5.8
User Participation in the Quality Assurance of Requirements Specifications:
An Evaluation of Traditional Models and Animated Systems Engineering Techniques

Heinz D. Knoell
University of Lueneburg, Germany

ABSTRACT

Improper specification of systems requirements has thwarted many splendid efforts to deliver high-quality information systems. Scholars have linked this problem to, between others, poor communication among systems developers and users at this stage of systems development. Some believe that specifying requirements is the most important and the most difficult activity in systems development. However, limitations in human information processing capabilities and the inadequacy of the structures available for communicating specifications and obtaining feedback and validation help to exacerbate the difficulty. This chapter presents an overview of both longstanding and newer requirements specification models and evaluates their capability to advance user participation in this process and incorporate stated quality attributes. It also reports on preliminary evaluations of animated system engineering (ASE), the author's preferred (newer) technique, which indicate that it has the capability to improve the specification effectiveness.

INTRODUCTION

It is estimated that between 30% and 80% of software projects fail (Dorsey, 2003; Standish

Group, 1994), depending on whether the basis is budgets or number of projects. Many of these software projects fail because of their inability to adequately specify and eventually meet customer requirements (Zave & Jackson, 1997). The following quotation from The Standish Group (1994) provides an excellent summary of the situation and puts the problem in perspective:

In the United States, we spend more than $250 billion each year on IT application development of approximately 175,000 projects. The average cost of a development project for a large company is $2,322,000; for a medium company, it is $1,331,000; and for a small company, it is $434,000. A great many of these projects will fail. Software development projects are in chaos, and we can no longer imitate the three monkeys—hear no failures, see no failures, speak no failures.

The Standish Group research shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns are just the tip of the proverbial iceberg. The lost opportunity costs are not measurable, but could easily be in the trillions of dollars. One just has to look to the City of Denver to realize the extent of this problem. The failure to produce reliable software to handle baggage at the new Denver airport is costing the city $1.1 million per day.

Based on this research, The Standish Group estimates that in 1995 American companies and government agencies will spend $81 billion for canceled software projects. These same organizations will pay an additional $59 billion for software projects that will be completed, but will exceed their original time estimates. Risk is always a factor when pushing the technology envelope, but many of these projects were as mundane as a driver's license database, a new accounting package, or an order entry system.

The fact is that too many software projects fail, that these failures may be due to both technical and behavioral reasons. Obtaining accurate systems requirements (Zave & Jackson, 1997) and translating them into feasible specifications is a well-discussed problem; however, involving potential users in the development project is an important factor in this process. In Levina and Vaast's (2004) investigation of how innovations are brought into enterprises, they underscored the pivotal nature of user involvement in successful implementation.

The specification of the requirements of an information system occurs fairly early in the development lifecycle. To accomplish this task, users and developers collaborate to describe the processes and static structures that are involved in the application domain and define their relationships. Quite often both sides speak a different language, using terminology that may be unfamiliar to the other. Clients express themselves using (in the view of technocrats) informal business terminology; developers write system specification from a technical perspective. It is widely acknowledged that this miscommunication is the reason for the prevalence of poorly specified systems and the root cause of many of the failed information systems (Byrd et al., 1992; Raghaven et al., 1994).

To solve this problem, the information system community introduced several models to improve communication between developers and users, particularly the accuracy and understandability of the representation of the business process information that eventually will be used to create the design artifact. But even with these models, users often experience some difficulty in assimilating the essence of the specification details. Sometimes they do not participate in the process because of cognitive limitations and an inability to comprehend the models. However, user participation in this process is essential to the production of accurate specifications. This has intensified the need to provide representational schemes for communicating specifications that are both accurate and easy to understand.