Chapter III

Data Quality-Based Requirements Elicitation for Decision Support Systems

Alejandro Vaisman
Universidad de Buenos Aires, Argentina

Abstract

Today, information and timely decisions are crucial for an organization’s success. A decision support system (DSS) is a software tool that provides information allowing its users to make decisions timely and cost effectively. This is highly conditioned by the quality of the data involved, usually stored in a data warehouse, and by a sound and complete requirements analysis. In this chapter we show that conventional techniques for requirements elicitation cannot be used in DSS, and present a methodology denoted DSS-METRIQ, aimed at providing a single data quality-based procedure for complete and consistent elicitation of functional (queries) and nonfunctional (data quality) requirements. The outcomes of the process are a set
of requirement documents and a specification of the operational data sources that can satisfy such requirements. We review the state-of-the-art in the field, and show that in spite of the tools and methodologies already proposed for the modeling and design of decision support systems, DSS-METRIQ is the first one that supports the whole process by means of an integral technique.

Introduction

It is a well-known fact that, among the phases of the software development process, analysis and specification of functional and nonfunctional requirements is a crucial one. The lack of good requirements specification is a major cause of failure in software development (Thayer, 2002). The software engineering community has developed many useful tools for requirements analysis in transactional systems. These kinds of systems deal with the day-to-day operation of an organization. Decision support systems (DSS) are of a completely different kind: they are focused on integrating data and models in order to improve the decision-making process. The data that feed a DSS generally reside in a data warehouse. The software development cycle of DSS has particularities that require applying methodologies different than the ones used for operational systems. The reason for this is twofold: on the one hand, traditional methodologies have been thought and designed with transactional systems in mind; on the other hand, specific methodologies applicable to DSS arose as ad-hoc answers to practical needs, and most of them are just mere enumerations of activities that must take place in order to implement the system, focusing on populating the data repository while ignoring important issues like the impact of changes in the operational data sources, or worse, if these data sources satisfy the users’ information requirements. New sources of failure are present in DSS: correctness and trustworthiness of the information are the basis of the decision-making process. We do not only need to understand the user’s information needs, but also account for keeping the data repository up-to-date according to user specifications. Also, update processes and their frequency must be considered, as well as the analysis of the quality and completeness of the data sources.

It follows that there is a need for techniques that, besides accounting for the software process cycle and functional requirements, also consider the quality of the information the system will deliver. There are several reasons for this. For instance, most of the time, people developing information systems do not consider the impact of low quality data (Kimball, Reeves, Ross, & Thornthwaite, 1998). Low data quality is more a rule than an exception. Just to give an example, it has been detected in the U.S., that approximately 50 to 80% of the computerized criminal records are inaccurate, incomplete, or ambiguous (Strong, Yang, & Wang, 1997). So far, the contribution of software engineering for addressing the problems stated has been