Chapter XVIII

Capture of Software Requirements and Rationale through Collaborative Software Development

Raymond McCall, University of Colorado, USA
Ivan Mistrik, Fraunhofer Institut für Integrierte Publikations - und Informationssysteme, Germany

Abstract

This chapter explains how natural language processing (NLP) and participatory design can aid in identifying system requirements. It argues that getting a complete list of requirements is often an iterative process in which some requirements are elicited only when users react to the system’s design. Costs of iterative requirements identification can be reduced by discovering new requirements during the design process, before implementation begins. This is facilitated when users participate in design, reacting to features as they are proposed. As users evaluate proposals, they often mention requirements not previously documented. Transcripts of participatory design sessions thus provide a rich source of new requirements for developers. The chapter explains how semantic grammars can be used to simplify the extraction of requirements from such
transcripts. The authors hope that an understanding of the value of participatory design and NLP will aid in the creation of better tools for support of software development.

Introduction

One of the main reasons that contemporary software projects are so difficult is that, by themselves, software developers do not have all the knowledge they need to create effective systems. The so-called “thin spread of application knowledge,” that is, developers’ poor knowledge of application domains, continues to plague the software industry more than 15 years after it was first described by Curtis, Iscoe, & Krassner (1988). Much of the knowledge that developers need is in the heads of the users of the proposed system (Rittel, 1972). Users have expert knowledge of use situations. Unfortunately their expertise is often in the form of tacit knowledge rather than explicit knowledge, and they only have access to much of it in the context of system use (Schon, 1983). As a consequence users typically cannot fully identify all their requirements before design of a system begins. Getting a full and accurate list of software requirements is thus not a simple, one-shot process. We argue that it is an iterative process in which some crucial requirements are not elicited until users can react to decisions about the system’s design.

Developers’ need to elicit requirements from users implies that successful projects depend on extensive collaboration with users. If, as we claim, some requirements can only be elicited when users react to design decisions, the question is then how to discover these requirements as soon as possible, so as to minimize the amount of work that needs to be redone. Above all, we want to identify new requirements before the system is in operation — when implementing new requirements is most expensive by far (Grady, 1999). Our conclusion is that the best approach is to get users’ reactions to the system as it is being designed. During this sort of collaboration – which is called “participatory design” (Schuler & Namioka, 1993) – users typically make comments that imply new requirements. We have devised tools that help developers identify and extract such requirements from user commentary.

Iteration in the Elicitation of Requirements

Iterative Software Development

In recent years there has been a substantial movement toward iterative approaches to software development in general and requirements identification in particular. The most full-blown set of approaches is known as “agile software development” (Highsmith, 2004; Martin 2002), though variations on this are known as “evolutionary” and “adap-
Model-Driven Development of Multi-Core Embedded Software
www.igi-global.com/chapter/model-driven-development-multi-core/51980?camid=4v1a