Chapter 4
The Human Role in Model Synthesis

Steven Gibson
California State University – Northridge, USA

ABSTRACT

This chapter highlights one concept representing the human role in requirements engineering and analysis for model synthesis. The production of design documentation to support model development requires elicitation of user requirements. The process of requirement elicitation plays a primary role in all Model-Driven Software Engineering (MDSE). Issues addressed include how requirements are gathered by the use of surveys, interviews, and questionnaires, and the importance of using validated constructs when gathering user information during requirement elicitation. Survey constructs, as used in requirements engineering, are analogues to the models in the final engineering product. A solution to improving the use of survey methods in the gathering of requirements is introduced. A small application is shown that suggests an example use of this proposed solution. This review of current practices explores areas where challenges are faced in the field with a concluding discussion that points to future trends in this research field.

INTRODUCTION

The use of Model-Driven Software Engineering (MDSE) methods is a valuable approach to address challenges requiring change from traditional software development regimes. One change from older methods to MDSE results in end-users sharing involvement in all aspects of the development process. For this reason and others, the software engineer is a facilitator between the end-user and the working product. Another change impacting software development methods is the likelihood that final software products often result from modifications or transformations of existing systems. Developers seldom have the option of designing a system from the ground up; instead, systems are often syntheses of existing products, uses, user practices, and data structures. The process of building an application can be seen as the transformation and composition of multiple domain models to synthesize a functional system. The process of building a software model delivers a system which is an abstract match with the problem

DOI: 10.4018/978-1-5225-3923-0.ch004

conception of the users. One challenge addressed here is how to standardize verification of the elicitation of the requirements.

Models encode human knowledge, experience, and expectations into graphical or structural forms. The primary concern of this chapter is exploring the methods which best represent the mental models of the users as part of a required system. This model representing the user requirements is described as a construct for this discussion. The resulting MDSE system presents a realization of the requested, required and projected functionality to address the domain challenge. The first focus, during requirements engineering, is on requirements elicitation, analysis, and interaction with the users. The next focus is on the principles of using communication for building understanding between users and engineers around the model systems. The model synthesis step directly follows from the successful requirements engineering process. Model synthesis can be seen as resulting from appropriate formalization of requirements (Desel, 2002).

This chapter focuses on the importance of users and domain specialists during the requirements stage of model development. The techniques discussed include writing and reading requirements, communicating with users about requirements, preparing documentation, synthesizing models, applying data gathering methods, identifying essential domain knowledge, and exploring requirement assumptions. Requirements analysis plays a role in the understanding of problems, communicating with users and driving the system development process. Requirements engineering serves a key role in model-driven software engineering. Boehm (1981) reported that approximately 60 percent of all errors in system development projects originate during the phase of requirements engineering.

Requirements engineering directs attention on aspects of analysis and design and does not address the full development life cycle of model-driven software engineering. These methods can cover the needs of a wide variety of applications in both large and small projects. Our focus here addresses requirements elicitation and requirements analysis. A definition for requirements has not been formally agreed to by software engineering researchers, although a definition described by Hull, Jackson and Dick (2011, p. 6) includes the ideas of statements which identify characteristics that are unambiguous, and are necessary for system acceptance. Requirements engineering involves developing documents to address the problems as described by users. The analyst and user establish a set of conventions to describe and discuss the problem domain and scope of the system. Analysis can be seen as the methods used to study the concepts, procedures and activities in a problem domain. Each problem domain in a certain modeling situation is dependent on the contexts and goals in the system environment. The first steps of analysis for a MDSE project is carried out in order to increase understanding of the problem domain.

When performing requirements engineering of planned MDSE systems, the requirements documents become the grounding for the entire system. It is important to continually update the set of descriptions which cover the system requirements and the models which generate implementation outcomes. The latest approaches in requirements documentation methods and communication tools, for the purpose of model synthesis, are discussed and explored in this chapter. The philosophy underlying requirements engineering for MDSE should be that gathering information for modeling is appropriate for all domain challenges, and will always deliver some value.

A subtopic of requirements engineering is the importance of analyzing legacy systems for the purposes of requirements documentation. Older legacy systems often need to be interwoven together with the overall system requirements. Model-driven software engineering creates functionality by the transformation of models based on deep knowledge of existing and available systems and tools. The models are derived from requirements of the problem space describing expected inputs and outputs of