ABSTRACT

Nowadays there are sound Model-Driven Development (MDD) methods that deal with functional requirements, but in general, usability is not considered from the early stages of the development. Analysts that work with MDD implement usability features manually once the code has been generated. This manual implementation contradicts the MDD paradigm and it may involve much rework. This paper proposes a method to elicit usability requirements at early stages of the software development process such a way non-experts at usability can use it. The approach consists of organizing several interface design guidelines and usability guidelines in a tree structure. These guidelines are shown to the analyst through questions that she/he must ask to the end-user. Answers to these questions mark the path throughout the tree structure. At the end of the process, the paper gathers all the answers of the end-user to obtain the set of usability requirements. If it represents usability requirements according to the conceptual models that compose the framework of a MDD method, these requirements can be the input for next steps of the software development process. The approach is validated with a laboratory demonstration.

Keywords: Model-Driven Development (MDD), Requirements Elicitation Process, Tree Structure, Usability Requirements

DOI: 10.4018/ijismd.2014100101
1. INTRODUCTION

Model-Driven Development (MDD) paradigm (Embley, Liddle, & Pastor, 2011) states that the analysts’ entire effort should be focused on a conceptual model, and the system should be implemented by means of model to code transformations performed by a model compiler. A software production process is then seen as a set of conceptual models that are adequately transformed from requirements to code. A plethora of MDD methods and tools have been proposed, such as WebML (Ceri, Fraternali, & Bongio, 2000) or UWE (Koch, Knapp, Zhang, & Baumeister, 2008) among others.

There are two main dimensions to consider in MDD (Frankel, 2002): a “vertical” dimension and a “horizontal” dimension. In the vertical dimension there are at least three main layers that must be present in any MDD process:

1. A Requirements Modeling step, to produce a Requirements Model.
2. A Conceptual Model representation, where requirements are represented from the computer-oriented perspective.
3. The final Software Product (the Code).

The horizontal dimension focuses on the different expressiveness that must be present in the different conceptual perspectives of a MDD software process. Summarizing, these perspectives are:

- The data (static, system structure-oriented) perspective.
- The functional (dynamic, system behavior-oriented) perspective.
- The interaction (user interface-oriented) perspective.

While it can be argued that the two first perspectives (data and functionality) are largely explored by the different MDD approaches, it is surprising to realize that the interaction perspective is not at all so intensively explored. One could conclude that a Software Product is just the sum of a conceptual model where data and behavior are precisely specified, what is not exactly true, because the specification of the system interaction is an essential component of any software product. To confirm this situation, it is enough to consider the current modeling approaches that we find in practice. From the Data perspective, the question of what data models can be used to represent data has an immediate answer: ER and UML Class Diagrams are clearly among the most widely used and known. From the Functional perspective, since the appearance of the Data Flow Diagrams till the most modern UML diagrams designed to represent functionality, the offer is large. However, if the question is what models are specially used to represent System Interaction, the answer is not at all so immediate.

Extending a previous version presented at (Y. I. Ormeño, Panach, Condori-Fernandez, & Pastor, 2013), the goal of this paper is to explore the need of an interaction modeling, focusing on an essential software quality criteria that is mainly in the interaction scope: usability. Nowadays, in MDD, usability features are manually implemented once the code has been generated. According to Bass (Bass & John, 2003) and Folmer (Folmer & Bosch, 2004), these manual changes may involve changes in the system architecture, which can result in a lot of extra effort. Moreover, these manual implementations can produce a source code that contradicts the system’s characteristics expressed in the conceptual model.

In the previous work (Y. I. Ormeño et al., 2013) we defined how to elicit usability requirements according to existent usability guidelines. In this paper, we define how to include the usability requirements elicitation process in a MDD method. The main final goal of the paper is to define an approach to facilitate the usability requirements capture process for analysts who are not experts in usability engineering, and that want to include also the specification of usability requirements in a MDD-based approach.

The proposal to elicit usability requirements is based on the idea that first, an expert...
Teaching Software Architecture in Industrial and Academic Contexts: Similarities and Differences
www.igi-global.com/chapter/teaching-software-architecture-in-industrial-and-academic-contexts/188205?camid=4v1a

Testing of Future Internet Applications Running in the Cloud
www.igi-global.com/chapter/testing-future-internet-applications-running/72237?camid=4v1a