Abstract Traditionally, the Software Engineering community has been interested in defining methods and processes to develop software by specifying its data and behavior disregarding user interaction. On the other hand, the Human-Computer Interaction community has defined techniques oriented to the modeling of the interaction between the user and the system, proposing a user-oriented software construction. This paper aspires to reconcile both visions by integrating them in a whole software production process. An approach based on conceptual-schema centric software development is presented, where conceptual primitives intended to specify static, dynamic and interaction aspects are properly provided. Furthermore, Model Transformation techniques are proposed to go from the problem space, represented by the Conceptual Schema, to the solution space, represented by the corresponding final software product. This proposal is underpinned by some current MDA-based technology, which makes user-oriented, model-based software generation a reality

Keywords: Conceptual modeling of user interface, Functional requirements, Model-based code generation, User Interaction and Model-driven approach

1. INTRODUCTION

Traditionally, modeling an Information System from a Software Engineering (SE) perspective basically consists in specifying its static (data-oriented) and dynamic (function-oriented) architecture. A lot of methods and techniques have been provided in the past to solve this specification problem,
including well-known data modeling techniques (e.g., the Entity-Relationship Model [5] and its extensions), and process modeling approaches (e.g., Structured Analysis, Data Flow Diagrams). Object-Oriented Modeling was seen in the nineties as the way to encapsulate statics (data) and dynamics (behavior) under the common notion of object, and new methods [3,24] and languages (e.g., UML [2]) have been proposed under this unified paradigm. The focus at the modeling step has always been put on those data and functional system aspects, while one very important issue was normally left at least for design time: the user interaction. Why interaction modeling is not considered at the same level than data and behavior modeling in the vast majority of SE-based software production methods? Isn’t interaction an essential part of the world description, as system data and functionality are?

It has been remarkable to realize that, even if the design and the implementation of User Interfaces (UIs) are recognized to be the most time-consuming step of any software production process, its modeling was rarely considered at the same level of data and function modeling when specifying a system. A whole community emerged to face that problem: the Human-Computer Interaction community (HCI).

To face and solve this dichotomy, one challenging goal in the context of both SE and HCI is to provide proper bridges between their best-known software production methods and techniques. Starting from the idea that SE is considered to be strong in specifying data and functional requirements, while HCI is centered on defining user interaction at the appropriate level of abstraction, a sound software production process must provide ways for specifying in a precise way data, functionality and interaction, all together. If any of those aspects is not properly faced, the whole software production process will fail, because the reality to be modeled is a mix of data, functionality and interaction. Consequently, software production methods that combine the most data-oriented and functional-oriented, conventional requirements specification, with the more interaction-oriented, UI modeling aspects are strongly required.

In this context, Model Transformation technologies (i.e., MDA approaches [13]) make possible to provide a global software process where all the relevant aspects of the analyzed problem (structure, behavior and user interaction) are specified from the beginning (Requirements Model). Those resulting models are first projected onto a Conceptual Schema and onto the final software product later. Based on the use of this Model Transformation approach, the intended contribution of this work is to provide the basis to build such a software production process, with two basic principles in mind:

1. Model Transformation is used as the basic software production paradigm, to automate the conversion of a source Requirements Model into its corresponding Conceptual Model and then converting this conceptual model