Chapter XI
User Interface Generation from the Data Schema

Akhilesh Bajaj
University of Tulsa, USA

Jason Knight
University of Tulsa, USA

ABSTRACT

Traditionally, the data model and the process model have been considered separately when modeling an application for construction purposes. The system analysis and design area has largely ignored the issue of the relationship between the user interface (UI) and the underlying data schema, leaving UI creation within the purview of the human computer interaction (HCI) literature. Traditional HCI methods however, underutilize the information in the data schema when designing user screens. Much of the work on automatic user interface (UI) generation has met with limited success because of the added load on the human designer to use specialized scripts for UI specification. In this research in progress, the authors propose a methodology applicable to database driven systems that a) automatically infers a draft interface directly from an extended entity relationship (EER) model schema and b) lists the interactions that need to take place between the designer and the tool in order to generate the final user schema.

INTRODUCTION

The graphical user interface has become both ubiquitous and relatively uniform in providing access to applications for diverse users (Myers et al., 2000). From the early 1980-s, user interface (UI) management systems focused on providing human designers high-level specification languages such as state transition diagrams or event based representations to specify the interface in response to events (Jacob, 1986, Olsen, 1986). These representations have become progressively richer and model-based interface development tools today range from automatic interface generators to tools that offer advice based on task representations.

This research in progress is important because traditionally, the data model and the process model have been considered separately when modeling an application for construction purposes. The system analysis and design (SA&D) area has largely ignored the issue of the relationship between the user interface (UI) and the underlying data schema, leaving UI creation within the purview of the human computer interaction (HCI) literature. Traditional HCI methods underutilize the information in the data schema when designing user screens. However, business applications are usually database driven, and the UI for most business information systems represents processes that allow users to interact with the data. In this work, we take a first step in bridging this gap between the SA&D and HCI literatures, and propose a generalized methodology to generate a UI that uses the data schema as the foundation.

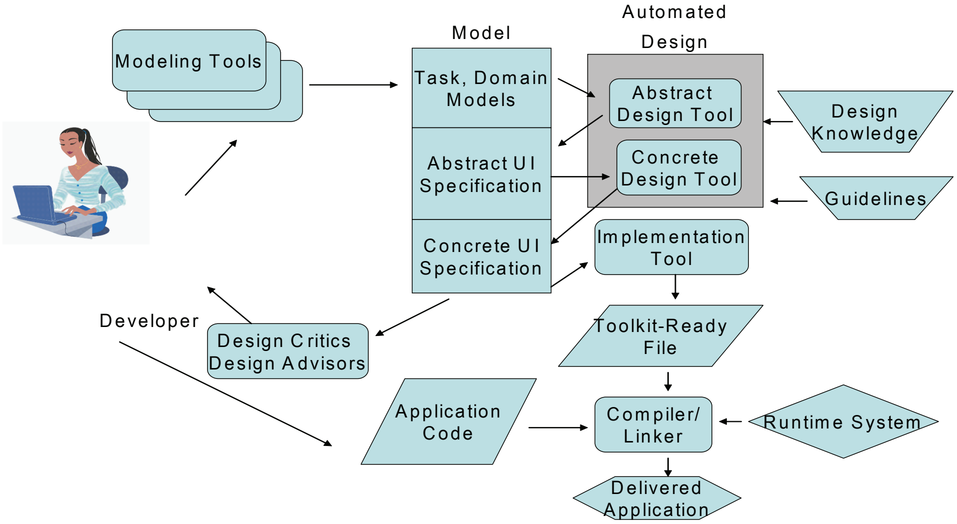
Figure 1 (Szekely, 1996) describes the model-based interface development process. The model component organizes the specification into three layers. Domain models correspond to the data schema. Examples of task models include data flow diagrams or other activity diagrams. An abstract UI specification provides a set of low level interface tasks such as selecting from a set of elements, information elements selected from the domain model, and how the two should be grouped. The concrete UI specification deals with the actual interface elements such as the windows, buttons, checkboxes and navigation buttons. Based on Figure 1, it is clear that the majority of model-based environments explicitly differentiate between task (process) models and data models.

The very great majority of business applications involve a database back-end with a front-end UI, and hence we utilize the extended entity relationship (EER) model to capture the data schema (Chen, 1976, Smith and Smith, 1977). Our methodology uses a set of rules to map EER objects automatically to provide a first cut user-interface, and then provides an opportunity for a structured dialog with the user to attempt to assuage some of the problems with the data-model-only approach.

A METHODOLOGY TO DERIVE A UI FROM AN EER SCHEMA

Before presenting the methodology, we list the concepts in the EER model that we will map. We

Figure 1. Model-based interface development process (Szekely, 1996)


Modeling Tools
Model
Automated Design
Task, Domain Models
Abstract Design Tool
Design Knowledge
Abstract UI Specification
Concrete Design Tool
Guidelines
Concrete UI Specification
Implementation Tool
Developer
Design Critics Design Advisors
Toolkit-Ready File
Application Code
Compiler/ Linker
Runtime System
Delivered Application