Chapter 26

REVERSE ENGINEERING INTERACTION PLANS FOR LEGACY INTERFACE MIGRATION

Eleni Stroulia and Rohit V. Kapoor

Department of Computing Science, University of Alberta
259 Computing Science Center, Edmonton, Alberta, T6G 2E8 (Canada)
E-mail: {stroulia, kapoor} @cs.ualberta.ca – URL: http://web.cs.ualberta.ca/~stroulia/
Tel.: +1-780-492 3520 – Fax: +1-780-492-1071

Byte Ryte bv

Abstract  Legacy interface migration is becoming an increasingly important IT activity; many organizations are interested in cost effective and low risk processes for making their legacy systems accessible to new, web-based platforms. Most migration techniques proposed to date require a lot of human expertise. In this paper we discuss Mathaino, an intelligent, multi platform, semi-automated, and low risk solution for migrating legacy user interfaces to the web by wrapping them with web-accessible front-ends.

Keywords: Domain specific model-based approaches, Model-based and task-based approaches to UI design, Semi-automated generation of UI, UI modeling language, UI reverse engineering.

1. INTRODUCTION

Legacy systems are antiquated systems that are still in use in spite of their obsolete operation platforms and technology, because their substitution with new systems can be extremely costly and disruptive to the organization’s daily operation. As more and more organizations move to delivering their services on the Web, interface migration is becoming a major activity in the Information Technology field and an important research problem in software reengineering.

In general, software-engineering research on legacy application reengineering and migration has followed one of three approaches. First, several methodologies have been proposed to making the legacy mainte-
nance and evolution practices more principled, measurable and controllable [2,9]. A second line of research [10] has focused on the problem of code understanding for isolating components of the legacy system so that they can be reused in a new environment. Finally, a third line of investigation has focused on the problem of migrating text-based user interfaces to GUs, by inferring the widgets implied in the interaction provided by the original interface and translating them to corresponding widgets in the target environment [11]. Understanding the user interface from the legacy system's code is a complex and error-prone task. Furthermore it requires substantial infrastructure for each different programming language that needs to be examined. On the other hand, "translating widgets" from the legacy platform to corresponding widgets in the target platform suffers from two disadvantages. First, the process has to be repeated every time the legacy system needs to migrate to a new platform. Second this migration is likely to result in the interaction model on the target platform being similar to the legacy interaction model, which is undesirable since legacy interfaces are usually difficult to learn.

In our work with the Mathaino system and its precursor, URGent [8], we have adopted a novel approach to this problem. Our objective is to develop an intelligent system that can declaratively model the current user interface of a legacy system at the user-interaction level and then use this model to simultaneously generate functionally equivalent interfaces simultaneously on a set of other platforms. Mathaino learns a model of the system-user interaction for a particular task from several examples of the task performed by expert users of the legacy system. It uses a set of heuristics to generalize the observed user behavior and constructs a model of how users accomplish this task with the existing interface. This abstract model of the task-specific system-user interaction specifies the pieces of information exchanged between the user and the legacy interface, their inter-dependencies and the elementary interactions that enable this information exchange on the current interface. This abstract model can be executed by a set of platform-specific translators that generate, at run time, an appropriate interface using the widget toolkit available on their own platforms. In our research, we have adopted principles and results of research in model-based user-interface design [4,12,14,15] for selecting and appropriately organizing the graphical interface widgets for the new interface front-end. Furthermore, Mathaino adopts an XML-based syntax for specifying the interaction model of the legacy user interface, similar to other current research efforts [1,16].

In the rest of this paper, we present an overview of the Mathaino legacy-interface migration process (Section 2) and we illustrate it with an example (Section 3). We then discuss its run-time environment (Section 4) and we