Chapter 2.4
User Interaction and Interface Design with UML

Jesus M. Almendros-Jimenez
Universidad de Almeria, Spain

Luis Iribarne
Universidad de Almeria, Spain

ABSTRACT

This chapter will show you how to use and specialise UML diagrams for describing the user interfaces of a software system. In order to accomplish the description of user interfaces, the proposed technique considers three specialised UML diagrams called user-interaction, user-interface, and GUI-class diagrams, which will be built following a model-driven development (MDD) perspective. These diagrams can be seen as the UML-based UI models of the system. In addition, this chapter is concerned with code-generation to implement the user interfaces of the system by using GUI-class diagrams and user-interaction diagrams. A case study of an Internet book shopping system is introduced in this chapter to proof and illustrate the proposed user interaction and interface design technique.

INTRODUCTION

The emergence of the *unified modelling language* (UML) (OMG, 2005) as an industry standard for modelling systems has encouraged the use of automated software tools that facilitate the development process from analysis through coding. The *user interface (UI)*, as a significant part of most applications, should also be modelled using UML. UML diagrams could be used to model user interfaces, and automatic CASE tools could help to generate code for user interfaces from UML designs. In general terms, *visual modelling* allows the developers to visualize source code in a graphical form: graphical abstractions, such as flow charts to depict algorithmic control flows and structure charts or simple block diagrams with boxes representing functions and subprograms, and so on. UML provides system

architects with a *visual language* for specifying, constructing, and documenting the artefacts of software systems. In particular, user interfaces should be visually modelled in order to describe the behaviour of the window system in response to *user interactions*.

This chapter is firstly devoted to show how to *use* and *specialise* UML diagrams in order to describe the user interface and user interactions of a software system, following a particular *model-driven development* (MDD) perspective. Model-driven development involves creating models through a methodological process that begins with requirements and looks into a high-level architectural design. Model-driven development facilitates and improves the software analysis and design and code generation facilities from models prevent the loss of substantial information during the transition of a model to its implementation.

In our MDD perspective, we consider the following steps for user interface design and modelling:

1. Firstly, we use a UML *use case diagram* for extracting the *main user interfaces*.
2. Secondly, we describe each use case by means of a special kind of *UML activity diagrams*, called *user-interaction diagrams*, whose states represent *data output actions* and transitions represent *data input events*. This perspective allows the designer to model the user interaction (i.e., input-output interaction) in each main user interface.
3. Thirdly, each input and output interaction of the *user-interaction diagrams* allows the designer to extract *GUI components* used in each user interface. Therefore, we can obtain a new and *specialized version of the use case diagram* representing the user interface design, and a class diagram for GUI components: *user-interface* and *GUI-class* diagrams, respectively.
4. The *user-interaction*, *user-interface,* and *GUI-class* diagrams can be seen as the *UML-based user interface models* of the system.

This chapter will also deal with *code generation* techniques. In our MDD perspective, the UML-based user interface models can be used for generating *executable code* with the following advantages:

1. **Rapid prototyping of the developed software:** Software modellers would find it useful to quickly generate user interfaces from high-level descriptions of the system.
2. **Model validation and refinement:** Prototyping can detect fails in design and refinement and validation of model by testing user interfaces and user requirements.
3. **Model-based code generation:** Generated code would fit with developed models.
4. **Starting point for implementers:** Prototypes can be refined until final implementation.

BACKGROUND

In the literature there are some works dealing with the problem of user interfaces in UML.

Use Cases and UI Design

Some of these works (CommentEdition, 2000; Constantine & Lockwood, 2001; Nunes & Falcao, 2001; Paterno, 2001) are focused on the utilisation of UML *use case diagrams* as a "starting point" of the user interface design, or even as a "high-level description" of the structure of the user interface. However, there are some considerations about the use case diagram style. Following the UML philosophy, a use case diagram could not be suitable for extracting the user interfaces. Use case diagrams may include some use cases referred to parts of the system not related to user interfaces such as classes, human tasks, components of other systems interacting with us, and so on. Or even