AUTOMATIC GENERATION OF GRAPHICAL USER INTERFACES FROM VDM++ SPECIFICATIONS

Carlos Alberto Loureiro Nunes

Dissertation supervised by Prof. Ana Paiva

Faculty of Engineering of the University of Porto

1. Motivation

One of the main concerns of software engineering is the development of reliable software despite its complexity. Formal methods, in the context of software engineering, address this concern. VDM is one of such formal methods, and VDM++ its specification language.

The development of VDM++ specifications normally involves the use of an interpreter to interact with the modelled (by the specification) system. Although available VDM++ tools allow that external programs use their interpreters, they only provide a connection method. A graphical user interface to interact with the model must be created from the ground up. This means defining and implementing the interface and developing the necessary “glue” between the interface and the VDM++ tool/interpreter. Automatically generating a graphical user interface would release the developer from using the interpreter to execute and test the VDM++ specification.

Additionally, the generated interface could be considered an iterative prototype.

2. Objectives

As previously described, the main goal of this work is the automatic generation of graphical user interfaces, but this goal can be divided into:

- Defining an approach to automatically define the elements and layout of a graphical user interface from a VDM++ specification.
- Defining an approach to automatically assign functionality to the generated user interface, so that the interface can interact with the underlying modelled system.

3. Approach Description

The following 5 subsections describe the main details and technologies of the generation approach developed from the research work. The sections follow a logical order.

3.1. Related Work

The research community has extensively studied the subject of graphical user interfaces. There are various tools and techniques for graphical user interface development. Each one focusing on a specific aspect or problem of graphical user interfaces (some already using some degree of automatic generation). But, for the specific subject of this dissertation, the contributions of previous work are limited. Most approaches or tools focus on assisting the developer in his interface development tasks, instead of removing the developer from the development process. Despite this it was still possible applying existing technologies (e.g.: user interface markup languages) to the automatic generation approach.

3.2. VDM++ Tools

The creation and development of VDM++ specifications is supported by a set of VDM tools. In order to focus on the main goals of this work, the generation approach, when possible and convenient takes advantage of these tools.

As a side note, the topic of this dissertation is similar to a future work proposal of one of these VDM tools.

3.3. The Approach

The approach can be divided into five components: interpreter; class reader; interface manager; window container; and internal representation.

The interpreter allows the execution of VDM++ expressions and the use of the executable constructions of a VDM++ specification. Without this component interaction with VDM model would not be possible. An existing interpreter was used.

The class reader is used to create and maintain a internal representation of the VDM++ specification. This includes information regarding operations, functions, constructors and other VDM++ elements.

The interface manager is used to create and manage the windows of the interface, and serves as an intermediary between the interface and the underlying system model. An existing user interface language, SwiXML, as well as its rendering engine assist the manager.

The window containers function as a backend to the windows. They provide a actions to the events of the user interface as well as serving as container for the individual interface window.

The internal representation is a internal depiction of VDM++ specification, from which the user interface is generated.

3.4. Generation Strategy

As previously stated, a VDM++ specification is the basis of the generation process. A VDM++

specification can describe, and is used to describe, almost any kind of system. And the VDM++ specification language does not possess any element specifically oriented for graphical user interface generation. Taking into account these characteristics, the generation strategy relies primarily on method signature analysis to create the interface. But is also capable of relying on annotations to the specification (defined in the context of this work) to extract further information to guide the generation process.

3.5. Dependencies

A graphical user interface at a given point may have disabled elements. For example, an operation might require information not yet available, therefore its graphical controls should be disabled. To address this problem, operation dependencies are monitored and their satisfaction verified.

4. Conclusions

The automatic generation approach is capable of creating a fully functional graphical interface from a VDM++ specification. The interface is also capable of enabling and disabling graphical elements, by checking dependencies extracted from the specification. This achieved while following the grammar of the VDM++ specification language, and without requiring of the user active participation in the interface generation process. Other conclusions, concerning, for example, research strategy can also be extracted from this research work.