Model-Based Generation of Domain-Specific Programming Environments

Dominik Hurnaus, Christian Wirth[1,2]

Christian Doppler Laboratory for Automated Software Engineering
Johannes Kepler University
Linz, Austria

Abstract

Software development and adaptation in the domain of industrial machine automation often has to be done by engineers being experts in automation engineering but lacking the necessary expertise in software development. In order to support them, tool vendors have to provide sophisticated and elaborate programming environments which allow programming in high-level, domain-specific notations and in restricted, reliable settings. It has been shown that the implementation of such programming environments represents a tremendous effort. In this paper we present an approach for generating domain-specific interactive programming environments from models. Models express where users can adapt a control program, define the dependencies and constraints between various configuration settings and program variants, and specify how the programs are presented to the users. Our approach is based on the Eclipse Modeling Framework and the decision-oriented product line engineering tool DecisionKing. We demonstrate our approach using a case study from the automation engineering domain.

Keywords: Domain-specific languages, end-user programming, product-line engineering user interface generation, visual languages.

1 Introduction

Our industry partner Keba[2] produces a hardware and software platform for industrial automation engineering. Customers of our partner demand not only the automation software but also need an end-user programming environment tailored to their specific needs. Therefore, our industry partner develops end-user programming environments which have to be adapted to the specific needs of many different people competent in specific areas of automation engineering. The end-user programming system has to offer rich functionality in terms of programming, setting parameters and configuring the automation process. For example, while some engineers may be allowed to manipulate all lines of automation code, process experts interacting with the machine might only be allowed to select between predefined

[1] Email: hurnaus@ase.jku.at, wirth@ase.jku.at

[2] This work has been conducted in cooperation with Keba AG, Austria, and has been supported by the Christian Doppler Forschungsgesellschaft, Austria.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

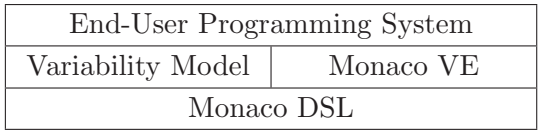

Fig. 1. The End-User Programming System based on the Monaco DSL

code fragments. Relying on traditional approaches, development of domain-specific programming environments that respect those aspects represents an immense effort.

2 Generating User Interfaces Based on Models

We are currently working on a model-based approach for the configuration and creation of end-user programming systems. The work relies on a layered approach as follows (Fig. 1): (1) The basis is formed by the domain-specific programming language Monaco, used to represent the hierarchical structure and the event-based behavior of automation systems. (2) Variability modeling techniques are used to define different variants of components, routines, and parameter settings. (3) The decision modeling approach and the DecisionKing [1] tool are employed for modeling dependencies and constraints between program variants as well as for representing higher-level configuration decisions together with their impacts. (4) Finally, a model specifies how higher-level decisions, program variants, and elements of control programs are be presented to different types of users in interactive interfaces. From those models highly customized end-user programming environments are generated.

A core element is the Eclipse Graphical Editor Framework (GEF) based Monaco Visual Editor (Monaco VE) [2], [3], an editor for the domain-specific language Monaco. The Monaco VE gives machine experts a representation of source code they can relate to: a flow-chart of the machines tasks - from high-level tasks down to fine-grained routines and error handling.

Using the product-line engineering tool DecisionKing the software developer can supply alternative variants of code fragments and a model to describe their relation. The EMF-model we suggest allows to give a detailed description of the structure of the user interface, how a user may manipulate the code for a machine, and the way it is presented. Some key elements in our model are the positioning of predefined views and editors in the Eclipse workbench, the declarative description of forms to display high-level decisions of the variant model, and navigators as a means to browse through the user interface.

In the presentation we show the prototype of the tool and a sample case study which is a reimplementation of an automation program of our industrial partner.

References

[1] Dhungana, D., Rabiser, R., Grünbacher, P., "Decision-Oriented Modeling of Product Line Architectures," Sixth Working IEEE/IFIP Conference on Software Architecture, WICSA 2007, Mumbai, India.

[2] Prähofer, H., Hurnaus, D., Mössenböck, H., "Building End-User Programming Systems Based on a Domain-Specific Language," 6th OOPSLA Workshop on DSM, Portland, OR, 2006.

[3] Prähofer, H., Hurnaus, D., Wirth, C., Mössenböck, H., "The Domain-Specific Language Monaco and its Visual Interactive Programming Environment," In Proc. of VLHCC 2007. IEEE Computer Society.