Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 279 (3) (2011) 3–25

www.elsevier.com/locate/entcs

Platform-Variant Applications from Platform-Independent Models via Templates

Nuno Amálio[1] Christian Glodt[2] Frederico Pinto[3]
Pierre Kelsen[4]

University of Luxembourg
6, r. Coudenhove-Kalergi, L-1359 Luxembourg

Abstract

By raising the level of abstraction from code to models, model-driven development (MDD) emphasises design rather than implementation and platform-specificity. This paper presents an experiment with a MDD approach, which takes platform-independent models and generates code for various platforms from them. The platform code is generated from templates. Our approach is based on EP, a formal executable modelling language, supplemented with OCL, and FTL, a formal language of templates. The paper's experiment generates code for the mobile platforms Android and iPhone from the same abstract functional model of a case study. The experiment shows the feasibility of MDD to tackle present day problems, highlighting many benefits of the MDD approach and opportunities for improvement.

Keywords: Software Product families, model-driven development, executable models, templates.

1 Introduction

A goal of model driven development (MDD) [21] is to enable software engineers to focus on design. This is achieved through the use of models expressing design concepts that abstract away from implementation and platform-specific details. Despite the increase in level of abstraction of programming languages and platforms in the past two decades, the diversity and complexity of current platform technologies makes manual development of code an arduous and expensive effort [21]. Modern platforms require considerable in-depth technical knowledge that is difficult to grasp by non-expert developers, a prominent example being mobile devices [15,14,1]. This cuts off an important source of creativity: talented people may be inspired to create

[1] Email: nuno.amalio@uni.lu
[2] Email: christian.glodt@uni.lu
[3] Email: fgasparp@gmail.com
[4] Email: pierre.kelsen@uni.lu

doi:10.1016/j.entcs.2011.11.035

novel applications, but only few have the time, energy and technical skill to dig into the intricacies of low-level platform programming. The problems of platform complexity and diversity suggest a move to a higher level of abstraction. However, novel features of modern devices require implementations to properly evaluate design decisions. Interaction [15,14,1], performance, and power consumption [22], common in mobile computing, are difficult to analyse from abstract models; they require experimentation with implementations.

In MDD, these issues can be tackled through a model-centric approach: all lower level code is generated from functional models of the system, also called *platform-independent models* (PIMs) in the *model-driven architecture* (MDA) [18], which is possible provided models fully describe the system's structure and behaviour. This approach tackles platform complexity and diversity, and enables the early construction of implementations from design models. Due to their level of abstraction, models can be articulated to describe families of related products, abstracting away from many intricacies of execution platforms. From such models, it is possible to build reusable transformations that enable the derivation of platform-variant products. Finally, from models and transformations to code, it is possible to obtain prototypes for experimentation of design decisions.

This paper presents an experiment with our MDD approach based on executable modelling and templates. Our approach enables the generation of code for various platforms from the same functional model. It is as follows:

- Applications are described using PIMs, describing structure and behaviour. PIMs are expressed in terms of abstract design primitives, yet concrete enough to enable generation of multi-platform code from them. PIMs' level of abstraction mitigates the need for platform expertise.
- Platform-specific artifacts are generated by instantiating templates of the platform's catalogue. The choice for the alternative execution platforms is a variation point in a product family, where variants are obtained automatically through code-generation by instantiating templates. Catalogues of templates constitute a repository of knowledge that is maintained by platform experts.

The approach presented here is based on formal languages: (a) models are expressed using the executable modelling language EP [16,17] supplemented with OCL [23], (b) catalogues of templates are expressed using the Formal Template Language (FTL) [4,2]. This approach gives generation of platform artifacts a first-class status: generative reusable assets are described in FTL. The experiment presented here evaluates this approach using a present day problem: building mobile-applications that have the same functionality, but need to run on different execution platforms. This is illustrated with Google's Android and Apple's iPhone mobile platforms.

2 Background

We give some background on both EP, the language used to express abstract models, and FTL, the language used to express templates.