Automated Generation of Platform-Variant Applications from Platform-Independent Models via Templates

Nuno Amálio, Christian Glodt, Frederico Pinto, Pierre Kelsen

University of Luxembourg
Introduction

* Manual code development is expensive
* Platform-diversity: need to support various platforms
* Not easy to focus on design within code-based development
Talented people may be inspired to create novel applications, but only few have the time, energy and technical skill to dig into the intricacies of low-level programming...
Model-Driven Development

* Raise level of abstraction
* from code to models
* Sw development focused on design
* Reduce reliance on code experts
Model-Centric approach

* All platform-code generated from functional models (PIMs)
* Models are total: structure + behaviour
* Use of abstraction enables coverage of various platforms
* Early experimentation from models
Our Approach

- Product families described as PIMs
- Platform-specific artifacts generated from templates
- A catalogue of templates per platform
Our Languages

* To describe PIMS:
  * visual language EP+OCL

* To describe Templates: FTL

* Both languages have formal semantics
EP

* EP structures models around classes
* Classes comprise events and properties
  * Details of events described in OCL
* EP models divided into domains
  * Domains represent different subject matters
* Domains are linked using bridges
FTL (I)

* Expresses templates of any target language
* Generation is based on substitutions
* We used a java implementation of FTL theory (integrated in Democles)
module JavaClassCat
Class ==
    public class 《CName》 {
        [private 《propName》 : 《propTy》 ; ]
        [ClassMethod]
    }
public class Book {
    private title : String;
    private isbn : String;
}
Our Approach

EP Modelling and platform-generation framework

EP Modelling
- EP Model of App X
- EP Model Mapping for platform A
- EP Model Mapping for platform B

FTL Platform Mappings
- FTL Catalogue for platform A
- FTL Catalogue for platform B

OCL Translators
- OCL Translator for language A
- OCL Translator for language B

Legend
- instantiation
- definition
- component

Platform A
Code for App X

Platform B
Code for App X
Running Example

Simple Library Browser

Library Browser

Toy tool for model-driven code generation from EP models via FTL templates
FTL Template Catalogues

// Catalogue of FTL templates
// for the Android platform
module AndroidCat

import CoreJava
import AndroidGUI
import AndroidPersistence
import JavaStringUtils
import AndroidSupport
Demonstration...
Evaluation (1)

* From same EP PIM: generated code for iPhone and Android
* Platform application described in terms of abstract concepts
* Could be developed by non-platform experts

<table>
<thead>
<tr>
<th></th>
<th>iPhone</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTL Catalogue</td>
<td>1745</td>
<td>1661</td>
</tr>
<tr>
<td>Generated Code</td>
<td>7119</td>
<td>5702</td>
</tr>
</tbody>
</table>
Conclusions (I)

* EP Model is platform-independent
* Platform-specificity and variability hidden in templates catalogue
* Not everything describable in OCL
* Mapping EP Domain define generic events
Conclusions (II)

- FTL templates provide separate medium over which experts can express their knowledge.
- Use of EP models benefits usability. Templates not so easy to develop.
- Platform catalogues are reusable assets.
- Living repository of knowledge.
Conclusions (III)

- Use of EP enables formal model analysis
- Event propagation and animation
- Experimentation of design decisions using generated code
Future Work

* Apply approach to larger case study
* Verification of EP models