

基于模型转换的 Android 代码自动生成技术的实现

孙圣博¹
SUN Shengbo

摘要

探讨了基于模型转换的 Android 代码自动生成技术,重点研究了从平台独立模型(PIM)到平台特定模型(PSM)的转换方法。通过元模型的自动转换技术,预定义转换规则将 IoT_PIM 转换为 Android_PSM,并使用 Acceleo 模板生成 Java 代码。开发了一个自动生成 Android 应用框架的工具,并通过实验案例验证了其有效性。所提出的技术提高了开发效率,确保了代码的一致性和可维护性。

关键词

模型转换; Android 代码; 自动生成技术; 转换规则; 代码生成

doi: 10.3969/j.issn.1672-9528.2024.11.006

0 引言

随着移动互联网和物联网(IoT)的快速发展,Android 应用的开发需求不断增加。传统手工编码效率低下,容易出现错误。模型驱动架构(MDA)通过模型转换和代码自动生成,可以显著提高开发效率和代码质量。本文研究了如何将 IoT 平台独立模型(PIM)转换为 Android 平台特定模型(PSM),并进一步生成 Java 代码,实现自动化开发。

1 IoT_PIM 到 Android_PSM 的模型转换方法

在模型驱动架构(MDA)中,模型转化是实现平台独立模型(PIM)向平台特定模型(PSM)转变的核心步骤。为有效地将 IoT_PIM 转换为 Android_PSM,本文采用基于元模型的自动转换技术,通过预定义的转换规则对 PIM 模型应用一系列变换,从而生成 PSM 模型。具体而言,本文的方法中,参与转换的源模型为交互式前端建模语言(IFML)元模型,目标模型为 Android 平台的 PSM 元模型。该转换通过解析 PIM 模型中的各个元素,并依照转换规则映射到 PSM 模型中的相应元素来实现^[1]。

2 IoT_PIM 到 Android_PSM 的转换规则

为了实现 IoT_PIM 向 Android_PSM 的转换,本文依据 IoT_PIM 元模型与 Android_PSM 元模型之间的语义相似性,建立了相应的转换规则,并详述了从 IoT_PIM 到 Android_PSM 的转换。本文采用了一种形式化的模型转换规则表示方法,因此所建立的转换规则具有以下特点:将 IoT_PIM 的领域模型、界面模型和行为模型分别转换到 Android 平台上^[2]。

2.1 领域模型转换规则

领域模型转换主要涉及类、属性、操作和关联的转换。这里以 ClassTransformation 为例进行说明,其他规则不再赘述。ClassTransformation 规则用于将 IoT_PIM 中的类转换为 Android 平台的 Java 类。转换过程中需要保持源类和目标类名称的一致性,同时完成类属性和类关联的转换。具体规则如下:

```
Rule ClassMappingRule {  
    From: IoT_PIM:Class  
    To: Android_PSM:JavaClass  
    Map:  
        name: source.name  
        attributes: to JavaAttribute  
        operations: to JavaMethod  
        associations: to JavaField  
}
```

2.2 用户界面模型转换规则

用户界面模型转换涉及 ViewContainers、ViewComponents、Event 等元素的转换,这里以 ViewContainersTransformation 规则为例进行说明。ViewContainersTransformation 规则用于将 IoT_PIM 中的 ViewContainers 转换为 Android 平台的 Layout。转换过程中需要保持 ViewContainers 和 Layout 间的名称和范围一致性,并完成嵌入的 ViewComponent 和 Event 的转换^[3]。具体规则如下:

```
{  
    From: IoT_PIM:UIModel:ViewContainer  
    To: Android_PSM:Layout  
    Map:
```

1. 曲阜师范大学 山东日照 276827

```
name: source.name
components: to AndroidWidget
events: to AndroidListener
}
```

2.3 行为模型转换规则

行为模型转换规则主要涉及 Action 的转换。这里以 ActionTransformation 为例进行说明，其他规则不再阐述。ActionTransformation 规则用于将 IoT_PIM 中的 Action 转换为 Android 平台的 Activity。转换过程中需要对 Action 的 Operation 和 Definitions 建立与 Android 平台 PSM 的对应关系。具体规则如下：

```
Rule ActionMappingRule
From: IoT_PIM:BehaviorModel:Action
To: Android_PSM:Activity
Mapping:
  name: source.name -> target.name
  operations: foreach(op in source.operations)
    map op to ActivityMethod
  definitions: foreach(def in source.definitions)
    map def to ActivityField
```

3 Android_PSM 到 Java 代码的转换

3.1 MDD 中 PSM 到代码的转换方法

MDD 是软件工程的一个分支，强调使用模型和模型转换作为软件开发的关键要素。开发人员使用通用（如 UML）或特定领域（如 IFML）建模语言来指定系统，并通过转换链生成最终产品。在遵循 MDD 方法时，有五种主要的代码生成方法：

PIM → NC（平台无关模型直接到原生代码）
PIM → PSM → NC（平台无关模型到平台特定模型再到原生代码）

PSM → NC（平台特定模型直接到原生代码）
PIM → CPC（平台无关模型到跨平台代码）
PIM → FSM → CPC（平台无关模型到特性特定模型再到跨平台代码）

本研究采用第二种方法来生成移动应用程序，即 PIM-PSM-Android 方法^[4]。

3.2 Accelele 模板及其转换

Accelele 是一种基于 Eclipse 的开源模型到文本转换工具，始于 2006 年，遵循 OMG 的 MOF2T 标准。可提供用户友好的插件，具有语法高亮和自动补全等功能，主要用于实现

代码生成。Accelele 通过编写模板定义如何从模型生成代码，使模型驱动开发更高效灵活。其使用步骤包括创建 Accelele 项目、编写模板、定义生成策略和执行转换等，为开发者提供了强大的代码生成能力^[5]。

模型加载：加载 EMF 兼容的 Ecore 模型，如 Android_PSM 模型。

模板应用：应用预定义的 Accelele 模板，这些模板定义了模型元素如何映射到代码结构。

代码生成：执行模板，遍历模型，生成相应的代码文件。

以下是一个简化的 Accelele 模板示例，展示了如何生成 Android Activity 类：

```
[template public generateActivityClass(a : Activity)]
[file (a.name + '.java', false, 'UTF-8')]package
[a.package/];import Android.app.Activity;import Android.
os.Bundle;

public class
[a.name/] extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.[a.layout/]);
    }[/file]
[/template]
```

通过使用 Accelele 模板，可以自动生成 Android Activity 类的基本框架，并根据需要扩展模板以生成更多的 Android 特定功能代码。

3.3 Android_PSM 到 Java 代码生成

Android_PSM 到 Java 代码的转换包括多个方面的映射和转换。表 1 总结了主要的转换规则。

表 1 转换规则

PSM 元素	Java 代码元素	转换规则
Activity	Java 类	生成继承自 Android.app.Activity 的类
Layout	XML 文件	在 res/layout/ 目录下生成对应的 XML 布局文件
UIComponent	成员变量	生成相应类型的成员变量和 findViewById 初始化
Method	Java 方法	生成对应的方法签名和基本实现
Intent	Intent 对象	生成 Intent 相关的代码，用于 Activity 间通信
Service	Java 类	生成继承自 Android.app.Service 的类

在代码生成过程中，Accelele 模板会遍历 Android_PSM

模型，根据上述转换规则生成相应的 Java 代码。例如，对于 Activity 的转换，会生成一个基本的 Activity 类结构：

```
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // 初始化 UI 组件并设置点击事件
        findViewById(R.id.button).setOnClickListener(v -> {
            // 处理点击事件
        });
        // 其他生命周期方法和自定义方法
    }
}
```

而 Acceleo 模板会根据 PSM 中定义的具体元素（如 UI 组件、方法等）填充这个框架，生成完整的 Java 代码。

为了进一步精确地描述 Android_PSM 到 Java 代码的转换过程，可以设 M 为应用描述模型， T 为 Freemarker 模板集合， C 为生成的应用代码框架^[6]。

定义转换函数 $F : (M, T) \rightarrow C$

转换过程可以表示为： $F(M, T) = G(X(M), T)$

其中， $X : M \rightarrow XML$ 是将模型转换为 XML 描述文件的函数。 $G : (XML, T) \rightarrow C$ 是代码生成器的函数，将 XML 描述文件和模板转换为最终代码。

对于模型 M 中的主要元素，可以定义以下映射规则：

```
G(X(e), T) = {
    f_activity(e, T_activity), if e ∈ Activity
    f_service(e, T_service), if e ∈ Service
    f_receiver(e, T_receiver), if e ∈ BroadcastReceiver
    f_layout(e, T_layout), if e ∈ Layout
    f_manifest(e, T_manifest), if e ∈ ApplicationConfig
}
```

其中， f_x 表示特定类型元素的转换函数， T_x 表示对应的 Freemarker 模板。

上述描述了如何将模型中的 Activity 元素通过 Freemarker 模板转换为 Java 类定义的基本结构。

4 工具实现与实验

4.1 Android 应用框架自动生成工具

Android 应用框架自动生成工具的主要任务是通过图形用户界面（GUI）和模型定义，来自动生成 Android 应用的代码框架。该工具架构分为 5 个层次，从底层到顶层分别是 Eclipse Platform、Platform Runtime、Tool Framework、工具层和应用层（如图 1 所示）^[7]。基础层是 Eclipse Platform，提

供相关的集成开发环境（IDE）支持。Platform Runtime 层包括基本的运行时组件，如 SWT、Jface 等，这些组件为上层封装提供基础支持。Tool Framework 层更为具体，包括编辑器（Editor）、视图（View）、向导（Wizard）等基础工具，这些工具广泛用于模型创建和编辑。在该层次中，还包括 GMF、Draw2d 等图形工具以及专门面向 Android 开发的 ADT 插件。

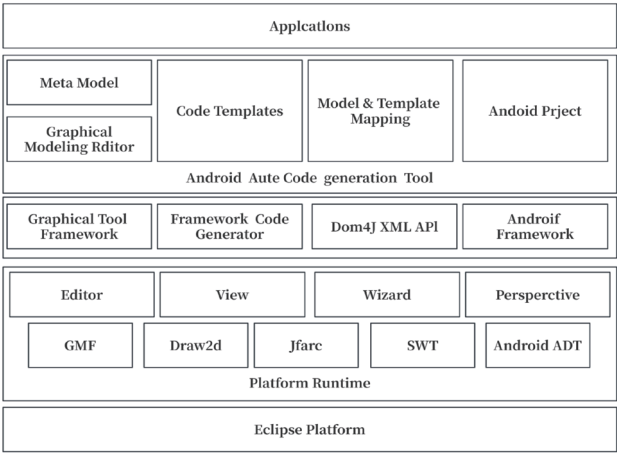


图 1 Android 应用框架自动生成工具架构图

其中，工具层包括图形工具框架、Freemarker 代码生成器、Dom4j XML API 和 Android Framework。这些组件用于定义元模型、编辑应用和生成目标代码^[8]。

应用层包含 Meta Model、图形化建模编辑器、代码模板和模板映射，共同构成自动代码生成工具，最终生成完整的 Android 项目。引入 Gradle 作为构建自动化工具，管理项目依赖关系，简化构建流程^[9]。开发者可在 Meta Model 中指定依赖和构建配置，工具会自动生成 Gradle 脚本。整体架构确保从模型定义到代码生成的高效性和一致性，提高了 Android 开发的自动化水平。开发者可通过工具界面微调 Gradle 配置，并使用 Gradle 命令或 IDE 集成功能构建、测试和部署应用。

4.2 实验案例

本文通过开发一个简易的电子书应用 eBook 来验证 Android 应用框架自动生成工具的有效性。eBook 应用包含多个功能模块，包括书籍列表显示、书籍下载、我的书籍、书籍内容查看等。应用架构由手机客户端和服务器端组成，用户可以在客户端进行书籍的下载与浏览，服务器端则提供书籍下载支持^[10]。本文通过自动生成工具生成应用基本框架，并在此基础上手动添加业务逻辑代码及服务器端代码。

具体而言，应用包含 3 个主要界面：BookListActivity（显示未下载书籍）、MyBookActivity（显示已下载书籍）和 BookContentActivity（查看书籍内容）。显示未下载的书

籍列表,用户可以通过点击列表中的书籍进行下载。进入 MyBookActivity 则可以查看已下载书籍列表,并通过点击书籍列表项进入 BookContentActivity 查看书籍内容。自动生成工具在通过创建新 Android 工程和应用模型文件后,拖拽工具栏元素定义应用模型节点和关联关系,例如组件之间的跳转和 Service 启动方式,并在属性视图中完成节点属性定义^[11]。最终,使用代码自动生成功能将模型转换为代码,从而得到应用的代码框架,如图 2 所示。

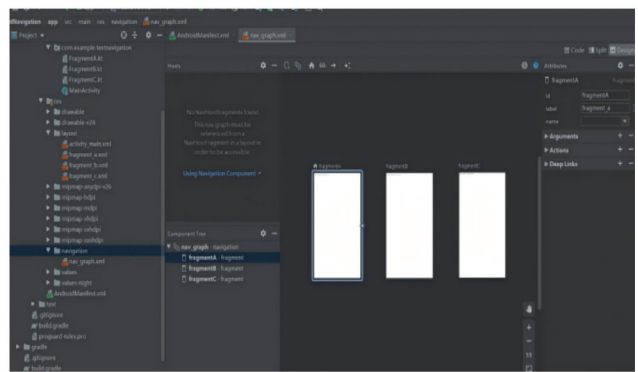


图 2 Android 应用框架自动生成工具模型编辑界面

在完成初始配置后,利用自动代码生成技术,获得了应用基础代码框架。在此基础上,进行了业务逻辑的实现和用户界面的优化^[12]。图 3 展示了 eBook 应用的运行界面,呈现了其核心功能模块,包括书目管理、个人书库、文本阅读器及版本信息等关键组件。其中,文本阅读器模块不仅支持单一电子书的内容呈现,还整合了多种旨在提升阅读体验的功能特性^[13]。

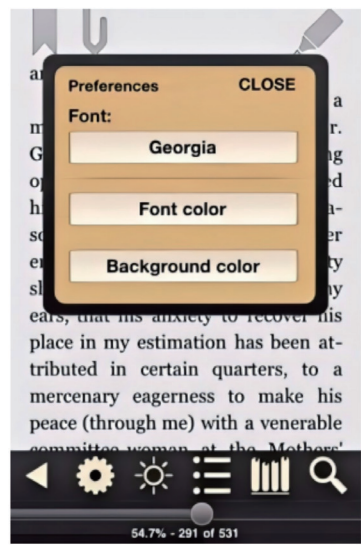


图 3 示例应用运行图

5 结语

本文通过研究模型转换技术,实现了从 IoT_PIM 到 Android_PSM 的自动转换,并通过 Acceleo 模板生成了 Java

代码。开发的自动生成工具有效验证了该方法的可行性,显著提高了 Android 应用开发的效率和代码一致性。未来的工作将进一步优化转换规则和模板,支持更多复杂的应用场景,实现更高效的自动化开发流程。

参考文献:

[1] 尹晶海,陈钰华.基于文本挖掘的 Android 相关人才招聘岗位数据分析[J].电脑知识与技术,2022,18(21):1-3.

[2] 闫怡梦,赵瑞莲,王微微,等.基于反射特征的 Android 测试用例自动生成[J].计算机系统应用,2022,31(12):350-358.

[3] 王旭阳,秦玉海,任思远.基于机器学习的 Android 混合应用代码注入攻击漏洞检测[J].信息安全研究,2023,9(10):940-946.

[4] 李默,芦天亮,谢子恒.基于代码图像合成的 Android 恶意软件家族分类方法[J].计算机应用,2022,42(5):1490-1499.

[5] 桑振,李坤明,庄海燕.基于可视化特征的 Android 恶意软件分类模型[J].长春师范大学学报,2022,41(4):56-61.

[6] 黄慧,李荆轩.车载 CAN 通信 Android 应用的设计模式研究与应用[J].汽车文摘,2024(1):1-7.

[7] 潘建文,张志华,林高毅,等.基于特征选择的恶意 Android 应用检测方法[J].计算机工程与应用,2023,59(21):287-295.

[8] 李琳.基于 Android 系统和增强现实技术在远程教育中的应用[J].数码设计(上),2022(17):99-101.

[9] 林立峰.基于 Android 系统和 Tensorflow Lite 的高速路面裂缝实时检测[J].福建交通科技,2022(12):130-134.

[10] 孙敏,成倩,丁希宁.基于 CBAM-CGRU-SVM 的 Android 恶意软件检测方法[J].计算机应用,2024,44(5):1539-1545.

[11] 邹振超,刘瑜.适用于 Android 平台的 SG-YOLO 人脸检测算法研究[J].计算机时代,2023(1):55-59.

[12] 张明嘉,黄丁樞,赵凯,等.基于 Android 的多模态情感识别 APP 的设计与开发[J].现代计算机,2023(5):99-103.

[13] 赵真,郭红梅,张莹,等.基于 Android 的建筑物信息采集系统设计与实现[J].地震研究,2023,46(3):453-460.

【作者简介】

孙圣博(1998—),男,山东枣庄人,硕士研究生,研究方向:软件工程。

(收稿日期:2024-07-25)