第37卷 第1期
Vol.37 No.1

计 算 机 工 程
Computer Engineering

2011年1月
January 2011

·软件技术与数据库·

文章编号：1000—3428(2011)01—0039—02 文献标识码：A 中图分类号：N945

模型驱动的测试用例自动生成框架

刘 扬，李亚芬，王 普

(北京工业大学电子信息与控制工程学院，北京 100124)

摘 要：提出一个基于模型驱动架构(MDA)的测试用例生成框架，其中，平台无关的系统模型通过水平转换成平台无关的测试模型，平台无关的测试模型通过竖直转换生成相应的测试用例。利用MDA转换工具ATL和MOFScript制定相应的转换规则作用于元模型，使测试者只须提供源模型和测试数据即可生成相应的测试用例。

关键词：模型驱动架构；平台无关模型；平台无关测试模型；测试用例

Automatic Generation Framework of Model-driven Test Cases

LIU Yang, LI Ya-fen, WANG Pu

(College of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China)

【Abstract】 This paper proposes a test cases generation framework based on Model-Driven Architecture(MDA), in which Platform-Independent Model(PIM) is converted into a Platform-Independent Test(PIT) model through level conversion, and platform-independent test model is converted into the corresponding test cases through vertical conversion. MDA conversion tools including ATL and MOFScript are used to develop the corresponding transformation rules acting on the meta-model, so that testers only need provide source model and test data to generate the corresponding test cases.

【Key words】 Model-Driven Architecture(MDA); Platform-Independent Model(PIM); Platform-Independent Test(PIT) model; test case

DOI: 10.3969/j.issn.1000-3428.2011.01.014

1 概述

模型驱动测试是一种基于模型的测试，它利用了模型转换技术，包括模型、元模型和转换规则，其中，转换规则定义了元模型各元素之间的映射[1]。基于模型驱动架构(Model-Driven Architecture, MDA)的测试是在软件开发早期，即在业务逻辑建模时开始软件测试工作。在构造完平台无关模型(Platform-Independent Model, PIM)后，用基于模型的转换方法生成相应的平台无关测试(Platform-Independent Test PIT)模型；随PIM转换到特定平台上的平台相关模型(PSM)由测试模型生成相应的测试用例。对PIM进行分析的同时也能发现其本身的缺陷，及时排除，防止缺陷随着软件开发过程的进行而被放大。

当软件的实现技术变化或平台迁移时，一般通过定义PIM到新的PSM的映射实现软件开发，同时定义并实现从测试模型到新平台的测试用例的转换，这样通过复用测试模型节省了测试成本。当软件需求发生变化时，用PIM来捕获变化的需求只需从新的PIM重新生成新的测试模型，其余的处理与原来一样，这样能够灵活处理不断变化的需求。

测试用例是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以测试某个程序路径或核实是否满足某个特定需求。本文基于MDA技术提出了模型驱动测试的基本框架，由该框架确定生成测试用例的具体方法。

2 基于MDA的测试用例生成框架

2.1 模型驱动测试的基本框架

MDA体系可以应用在系统模型和测试模型上，模型驱动测试的基本框架如图1所示[2]。在竖直转换中，左半部分的PIM可以转换成PSM，PSM再转换到系统代码。与系统模型的转换类似，右半部分的PIT转换成PST模型，PIT和PST均可直接转换生成测试用例。在水平转换中，系统模型的PIM可以转换为测试模型PIT，再由PIT转换到PST，或者由PIT直接生成测试用例。与之类似，PSM可以转换到PST，再由PST转换到测试用例。对比这2条水平转换的路线，一个是由PIM转换到PIT，一个是由PSM转换到PST，它们在本质上是一样的，都是由系统模型转换到测试模型。不同的是，PIM、PIT是比PSM、PST更为抽象的模型，并且均是平台无关的模型，因此，PIM转换到PIT要更具有代表性，应用范围更加广泛，而且易于实现。

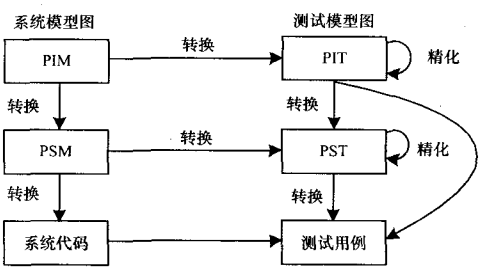

图1 模型驱动测试的基本框架

此外，因为转换生成的PIT、PST一般不是很完整的，所以在每个转换步骤之后，通过相关的模型工具可以对测试模型进行精化。精化之后的PIT、PST可以用于其下一阶段的模型转换。

2.2 测试用例生成过程

基于上文的分析，在水平转换中选择PIM到PIT的转换作为研究对象。在竖直转换中，由于现阶段只对PIM测试，

作者简介：刘 扬(1983－)，女，硕士研究生，主研方向：模型驱动测试，Web应用技术；李亚芬，高级工程师；王 普，教授

收稿日期：2010-06-23 **E-mail**：liuyang-83127@163.com

因此选择由 PIT 直接生成测试用例作为研究对象，即本文生成测试用例的过程是先由 PIM 转换成 PIT，再由 PIT 转换为平台相关的测试用例，如图 2 所示。

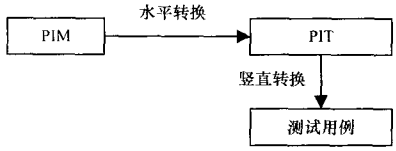

图 2 测试用例生成过程

3 测试用例生成方法

对于 PIM 系统模型，本文用目前最为流行的 UML 图表示；而对于 PIT 测试模型，用 2004 年 OMG 提出的 U2TP(UML 2.0 Testing Profile)来表示；对于具体的可执行测试用例，选用基于 Java 平台的 JUnit 单元测试用例。

3.1 生成测试用例的具体方法

由 PIM 系统模型生成 PIT 测试模型再生成测试用例的具体转换方法如图 3 所示：(1)利用 ATL 模型到模型的转换工具，通过相应的水平转换规则(模型到模型的转换规则)，作用于源元模型(UML 元模型)和目标元模型(U2TP 元模型)，把输入的源模型(UML 模型)转换成目标模型(U2TP 模型)。(2)利用 MOFScript 模型到代码的转换工具，通过相应的竖直转换规则(模型到代码的转换规则)，作用于源元模型(U2TP 元模型)和测试数据，把输入的源模型(U2TP 模型)转换成相应的测试用例[3]。其中，测试数据用于测试 UML 系统模型，它指定了参数值和预期返回值。通过改变测试数据文件中的内容，可以对同一个 UML 模型生成不同的测试用例。

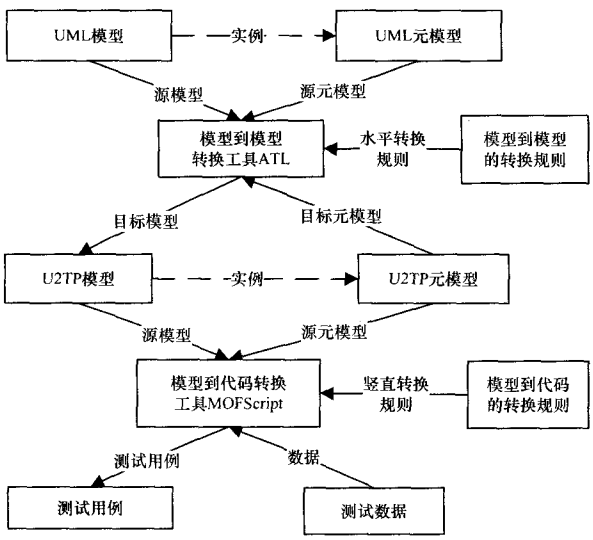

图 3 测生成试用例的具体方法

此外，图 3 中的 UML 元模型、U2TP 元模型和模型到模型的转换规则都是平台无关的，模型到代码的转换规则是平台相关的。当在相同的平台上测试不同的应用软件时，图中的各部分均不需要修改。测试者只需提供 UML 系统模型和测试数据，然后利用 ATL 和 MOFScript 工具生成 U2TP 测试模型(中间环节)和测试用例(最终结果)。

3.2 PIM 到 PIT 的转换

用于描述 PIM 的 UML 是一种通用的可视化建模语言。在 MDA 中，所有的模型均是以 UML 描述的。UML 模型相比其他描述性模型语言所表示的模型具有更加丰富的语义。OCL 是一种表达式语言，是 UML 的约束和查询语言。UML 和 OCL 相结合的方法扩展了 UML 的表达能力，使 UML 模型更精确更完备。

本文用 U2TP 来描述 PIT 测试模型。U2TP 是基于 UML2.0 的测试建模语言，可用于从单元测试到系统集成测试各个级别的测试建模。它基于 MOF，可以独立于 UML 使用。4 个逻辑概念组织起来共同构成了 U2TP：测试架构(Test architecture)，测试行为(Test behavior)，测试数据(Test data)，测试时间(Test time)。Test architecture 指定了 U2TP 的结构；Test behavior 指定了 U2TP 的行为；Test data 代表贯穿 U2TP 测试行为始终需要的数据；Test time 是对 U2TP 测试行为的量化。

对于 PIM 到 PIT 的转换，本文选择了 ATL 的转换方法。ATL 是 ATLAS 转换语言的简称，是 ATLAS 研究组开发出来的一种符合 OMG 的一个 QVT 提案的模型转换语言，其基于 EMF(Eclipse Modeling Framework)。本质上，ATL 属于基于规则的模型转换语言，其中使用了 OCL 作为约束描述语言，使得转换过程更加精确。在研究 PIM 到 PIT 的转换过程中，把 PIM 作为源模型、PIT 作为目标模型，它们又分别有各自的元模型，如图 4 所示。利用 ATL 转换方法，制定出相应的转换规则，作用于各个元模型，实现模型间的转换。此外，所有的元模型都统一于元-元模型 MOF。

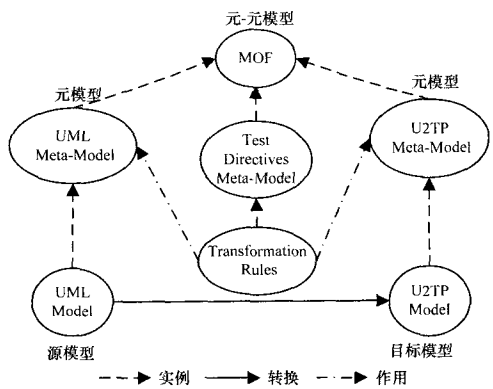

图 4 PIM 到 PIT 的转换

3.3 PIT 到测试用例的转换

对于生成的测试用例，本文选用基于 Java 的 JUnit 测试框架。JUnit 是一个开源的回归测试框架，它在极限编程领域很流行，在 Java 的单元测试中被开发人员广泛使用，在过去的几年中，它已成为单元测试事实上的标准。图 5 给出了 JUnit 的基本框架。

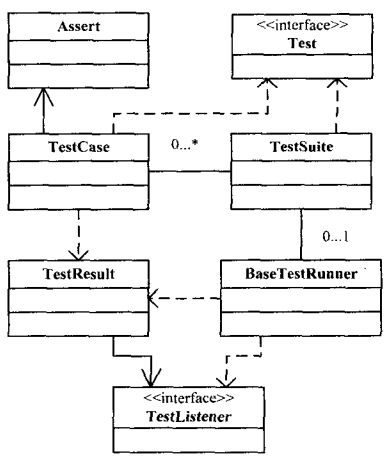

图 5 JUnit 框架

(下转第 44 页)

中为空，qFutureNode 中只有 A，然后进入 while 循环，开始执行过程。从 qFutureNode 中取 A 加入 qCurNode，再从 qCurNode 中取出 A。由于 A 没有直接前趋结点，因此 IsReadyRun 函数返回真值，进入 switch 的第一个分支。执行 A，并将 B 加入 qFutureNode，然后从 qCurNode 中删去 A。此时，qCurNode 为空，qFutureNode 中为 B。如此循环，直到完成整个调度图的执行。图 8 给出了主调度过程中 qCurNode 和 qFutureNode 的变化。

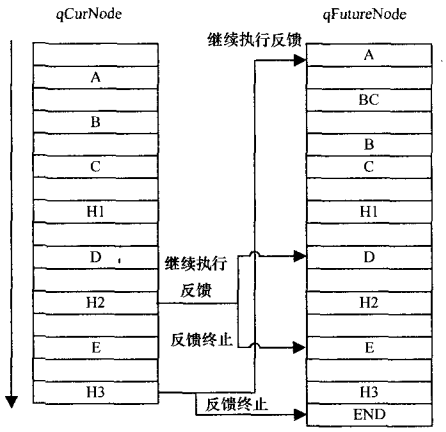

图 8 有向调度图的主调度过程

4 结束语

针对目前应用于复杂系统优化的多方法协作优化思想在实际应用中面临的困难，本文提出优化算法组件化实现的概念，并且结合科研项目实践总结出优化算法组件的接口描述规范，以满足多方法协作优化方案的调度需求，为多方法协作优化方案有向图调度模型建立奠定基础；其次，借鉴并行分布计算领域中的 DAG 任务调度思想，采用有向图描述多方法协作优化方案的调度，建立从多方法协作优化方案到有向图调度模型的转换，并结合一个例子演示调度计算的过程。支持多方法协作优化的算法组件化设计与调度方法在导弹总体设计优化领域得到较好的应用。

参考文献

[1] 罗文彩，罗世彬，陈小前，等. 导弹总体设计多方法协作优化[J]. 弹箭制导学报，2005，25(3)：16-19.

[2] 罗文彩，罗世彬，陈小前，等. 多方法协作优化算法协作策略研究[J]. 系统工程与电子技术，2005，27(7)：1238-1242.

[3] 朱延广，梅 珊，赵 雯，等. 支持复杂产品总体优化设计的多算法协作优化框架研究[J]. 系统仿真学报，2007，19(11)：2417-2420.

[4] Robin L H. The Common Optimization Interface for Operations Research: Promoting Open-source Software in the Operations Research Community[J]. IBM Journal of Research and Development, 2003, 47(1): 57-66.

[5] 罗亚中，唐国金. 基于面向对象技术的优化算法类库分析与设计[J]. 航空计算技术，2003，33(1)：62-64.

[6] 陈 彬，刘宝宏，黄柯棣. 组件式仿真系统中基于 Web 的管理控制方法[J]. 计算机工程，2009，35(24)：250-252.

[7] Maheswaran M, Siegel H J. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems[C]//Proceedings of the 7th IEEE Heterogeneous Computing Workshop. [S. l.]: IEEE Press, 1998.

[8] 周 深，杨路明，段桂华. VLCC 中的 DAG 并行算法[J]. 计算机工程，2009，35(19)：151-153.

编辑 陈 文

(上接第 40 页)

图 5 中的 7 个类或接口是 JUnit 的核心，每个类或接口的职责分别为：(1)Assert 类，当条件成立时 assert 方法保持沉默，但若条件不成立就抛出异常；(2)TestResult 类，包含了测试中发生的所有错误或者失败；(3)Test 接口，可以运行 Test 并把结果传递给 TestResult；(4)TestListener 接口，如果测试中产生事件(开始、结束、错误、失败)，则通知 TestListener；(5)TestCase 类，定义了可以用于运行多项测试的环境；(6)TestSuite 类，运行一组 test case(它们可能包含其他 test suite)，它是 Test 的组合；(7)BaseTestRunner 类，test runner 是用来启动测试的用户界面，BaseTestRunner 是所有 test runner 的超类。

文献[4]提供了一个从 U2TP 测试模型到 JUnit 的映射。根据这个映射，可以制定相应的转换规则，根据该规则由 U2TP 测试模型生成 JUnit 测试用例。

对于从 PIT 测试模型到测试用例的转换，本文选用 MOFScript 转换方法。MOFScript 是模型到文本的转换语言，它把基于元模型的模型转换成输出文本。

4 结束语

本文提出了一个由 PIM 生成测试用例的方法。在 MDA 软件开发中，PIM 可以转换到各个特定平台上的 PSM(例如 Java 平台)；与之相对应，在测试用例生成过程中，选用 U2TP 作为 PIT 模型，由 U2TP 可以生成各个特定平台的测试用例(例如，Java 平台上的 JUnit 测试用例)。本文利用 ATL 模型转换工具，由平台无关模型生成测试模型；再利用 MOFScript 模型到代码的转换工具，由测试模型生成测试用例。

今后的工作是运用本文基于 MDA 的测试用例生成的方法，以 Web 应用系统为对象[5]，自动生成 JUnit 的测试用例并对该系统进行测试。

参考文献

[1] Heckel R, Lohmann M. Towards Model-Driven Testing[J]. Electronic Notes in Theoretical Computer Science, 2003, 82(6): 1-11.

[2] Zhen Rudai. Model-Driven Testing with UML 2.0[C]//Proc. of the 2nd European Workshop on Model Driven Architecture(MDA) with an Emphasis on Methodologies and Transformations. Toledo, Spain: [s. n.], 2004.

[3] Javed A Z, Strooper P A, Watson G N. Automated Generation of Test Cases Using Model-Driven Architecture[C]//Proc. of the 2nd International Workshop on Automation of Software Test. Minneapolis, USA: [s. n.], 2007.

[4] OMG. ptc/2004-04-02-2004 UML 2.0 Testing Profile[S]. 2004.

[5] 刘焕洲，缪淮扣. Web 应用程序建模和测试用例生成方法[J]. 计算机工程，2008，34(6)：60-62.

编辑 张 帆