模型驱动的软件测试研究*)

王林章　李宣东　郑国梁
(南京大学计算机科学与技术系　南京 210093)

摘　要　MDA(Model-Driven Architecture)旨在基于平台无关的模型(PIM)和平台相关的模型(PSM)之间的分离和相互转换来复用平台无关的设计,实现异构中间件平台之间的集成和互操作,从而把软件开发的重点提前到 PIM 的设计上,也使得在模型级解决测试生成问题成为研究热点。本文从过程、方法与工具三个方面研究了模型驱动的测试,首先提出与软件开发过程集成的模型驱动的软件测试过程;其次讨论了与过程对应的模型驱动的测试方法,基于 PIM 生成平台无关的测试(PIT),定义并实现了从 PIT 到平台相关的测试(PST)的映射算法,使得 PST 能够直接在相应平台上执行以发现软件实现是否与规约一致;最后,对上述过程中的方法提供工具支持。本文过程和方法在一个基于三层 Web 应用的在线银行系统上得到了实现。

关键词　MDA,MDT,PIM,PSM,PIT,PST

Research on Model-driven Software Testing

WANG Lin-Zhang　LI Xuan-Dong　ZHENG Guo-Liang
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　MDA (Model-Driven Architecture) is proposed to reuse the platform-independent design and to realize the integration and interoperation of different middleware by separating and inter-transforming of platform-independent model (PIM) and platform-specific model (PSM). Thus the design of PIM is focused on during the software development, and to solve the problem of test generation at modellevel become popular. This paper discusses the process, the method and the supporting tools of the model-driven software testing. Firstly, the model-driven software testing process is integrated with the model-driven software development. Secondly, abiding by the above process, platform-independent test (PIT) is generated from PIM, and is then transformed into platform-specific test (PST) by mapping to the specific platform. The PST is used to run the implementation in certain platform to find the inconsistency between the specification and the implementation. Lastly, an automatic tool is designed to support above process and method. The method is applied in the testing of an online bank system.

Keywords　MDA,MDT,PIM,PSM,PIT,PST

1　引言

在解决企业级应用过程中,中间件技术起了很大的作用,但由于多个中间件技术平台的兴起,带来了跨平台计算问题,为了实现不同中间件平台之间的集成和互操作,OMG(Object Management Group)提出了一个标准的模型驱动的体系结构 MDA。MDA 划分了两种模型,一个是与平台无关的模型(Platform Independent Model , PIM),一个是与平台有关的模型(Platform Specific Model , PSM),将系统的功能描述和系统在特定平台上的实现分离开来。PIM 是对系统的功能和结构的形式化描述,是抽象的、与实现细节无关的,PSM 是在特定的目标平台上的系统功能的描述,它在 PIM 中添加了与平台有关的信息,但是 PSM 并不等同于实现,它仍然是用模型语言来描述的[1,2]。一个 PIM 可以映射到多个 PSM,如图 1 所示。

UML 是面向对象系统分析、设计的标准的建模语言,自从成为建模语言事实上的标准后,就得到学术界的推崇和工业界的支持,使得 UML 广为使用,基于 UML 的方法和实用技术的研究成为热点,UML 对面向对象软件开发全生命周期的支持也使得软件开发人员优先用它来描述和构建复杂的软件系统,UML 提供扩充机制,使得其便于表示不同目的的模型[3],在 MDA 中,UML 及其扩展用于表示 PIM 和 PSM。本文使用普通的 UML 来描述 PIM。由于 J2EE 有着广泛的应用[4],本文选用它作为研究的目标平台,对于这个特定的平台,需要对 UML 进行扩充后描述 PSM,比如 OMG 已经定义的 UML Profile for EJB[5]。

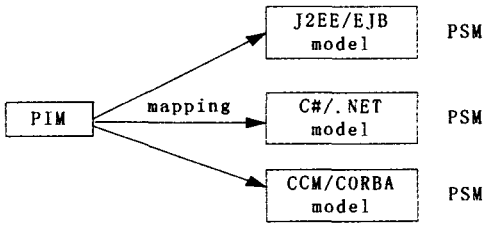

图 1　PIM 到 PSM 的映射

基于 UML 开发的软件系统,软件开发的分析、设计阶段工作成果多为各种模型图,系统的可用信息都在这些模型图中,给信息的提取带来了新的问题,从而也给测试带来新的课

*)基金项目:863 项目(2002AA116090);自然科学基金项目(60207036,60233020);973 项目(2002CB312001)。王林章　博士生,主要研究软件工程、软件测试、模型驱动的测试。李宣东　博士,教授,博士生导师,主要研究软件工程、形式化方法、模型检验。郑国梁　教授,博士生导师,主要研究软件工程、形式化方法。

题。对测试而言，原先的基于规约或者是基于程序的方法都不能直接使用，基于模型的测试方法在这个领域有很大的优势。软件测试工作的核心主要是生成测试用例，在面向对象软件开发过程中系统的规约、设计、代码是生成测试的信息来源，是软件在其生命周期不同阶段的变体，当然规约、设计、代码在每一阶段的测试中都起相应的作用，特别地，系统规约是生成系统测试的测试用例的基础和系统测试的检验依据，软件代码是生成单元测试用例的基础和单元测试的检验依据，系统设计是生成集成测试用例和集成测试的检验依据[6]。

MDA 分离软件系统的逻辑与实现，分别用 PIM 和 PSM 来表示，并且通过 PIM 到 PSM 的转换来复用平台无关的设计，实现异构中间件平台之间的集成和互操作，并对转换工作提供工具支持，从而把软件开发的重点提前到 PIM 的设计上，减少了软件开发后期开发人员的参与。目前大多数研究都关注软件构造过程中模型驱动的软件开发，相应的测试研究较少。为了进一步研究 MDA 框架下与模型驱动的软件开发过程相适应的软件测试方法，本文基于模型的测试方法，研究模型驱动的软件测试方法。在软件开发早期即在业务逻辑建模时就开始软件测试工作，构造完 PIM 后，用基于模型的测试方法生成相应的测试，随着 PIM 转换到特定平台上的设计 PSM 时，完成测试的映射，使得在 PSM 到代码的转换工作结束后便可以开始测试执行工作，而且对 PIM 进行分析的同时也能发现其本身的缺陷，以便及时排除，以防缺陷随着软件开发过程的进展而被放大。能够在模型级解决问题以提高软件质量、缩短软件生存期时间，而目前大多数研究都关注软件构造过程中模型驱动的软件开发，相应的测试研究较少。

本文第 2 部分提出了一个与模型驱动的软件开发过程集成的模型驱动的软件测试过程，并与传统的模型驱动的软件测试过程作了比较；第 3 部分介绍了一个在线银行系统的模型驱动的开发；第 4 部分提出了模型驱动的软件测试方法，第 5 部分是相关工作的比较，最后是总结与将来的工作。

2 模型驱动的软件测试过程

随着软件开发和软件测试相关技术的研究和发展，一个明显的发展趋势是测试与软件开发过程集成，尽早开始测试活动，减少中间环节，直接复用软件系统开发过程中的分析、设计结果描述，用于获取不同目的的测试的测试用例生成所需的信息，如果可以将验证与确认也可以理解为广义的测试，那么建立支持基于 UML 软件开发全生命周期软件测试过程框架将成为可能。

面向对象的开发过程发展的趋势是迭代式增量开发过程[7]，而且用像 UML 可视化面向对象建模语言将分析、设计的建模过程的结果可视化，从而分析设计模型是软件系统在软件开发生命周期不同阶段的变体，其中包含最终系统的本质的信息。基于模型的软件测试过程是基于软件开发过程中分析、设计的模型，从不同开发阶段的分析、设计模型中提取信息用于生成不同测试阶段的测试用例就成为可能，如分析/规约模型用于生成不同抽象和粒度级别的系统测试用例，设计模型用于生成不同集成层次的集成测试用例。基于 UML 的测试方法的目标是希望在生成测试用例的过程中不增加用户除 UML 外其他知识，能够通过提高自动化程度尽量减少工作量，在分析、设计阶段就开始测试设计工作，通过对模型的可测试性和语法、语义的分析，不但能够发现模型本身存在的问题乃至分析、设计的缺陷，而且能够生成测试执行相应阶段所需的测试用例。目的是在软件开发早期进行软件测试的工作，以便尽早发现分析、设计、构造本身的缺陷，在软件测试执行前生成测试执行阶段需要的测试用例，以便管理人员尽早进行测试风险分析、测试工作量估计、合理安排测试资源。

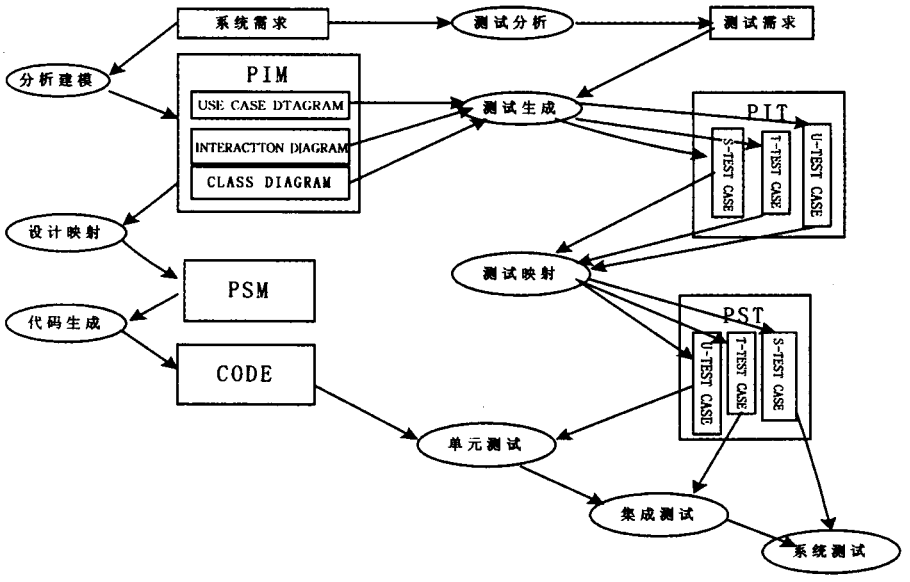

图 2 基于模型驱动的软件开发与测试过程

这种基于模型的测试过程在软件测试的研究与发展中起到了很大的作用，也被逐步应用到工业界。MDA 的提出对软件开发过程带来了实质性的变革，图 2 的左边部分描述了一个基于 MDA 的软件开发过程，开始于 PIM 的构造，PIM 仅表示业务功能和行为，是逻辑应用的表示，一般由业务专家和建模专家完成，模型驱动的软件开发使得软件构造的焦点集中于 PIM 的构造，从 PIM 到 PSM 的映射、从 PSM 到代码的转换都可以由工具支持实现自动化，与此相适应，图 2 右边部分所示模型驱动的软件测试过程，仅使用开发阶段的 PIM，结合由系统需求得到的软件测试需求使用基于模型的测试方法可以分别生成单元、集成、系统测试各阶段的测试用例，包含测试执行的条件和预期的结果。类似 PIM 到代码的

转换，分别定义一组转换规则实现 PIT 中三类测试用例到 PST 中三类测试用例的转换，使得 PST 可以直接在相应的平台上直接执行相应的单元测试、集成测试和系统测试。这种模型驱动的软件测试过程继承了基于模型的软件测试和模型驱动的软件开发的优点，同时也把测试工作的中心提前到 PIT 的构造上，后续的映射工作实现工具支持，减少了测试人员的参与，也避免了新的错误的引入。

3 模型驱动的软件开发

本文的工作基于本项目组另一部分工作[8]，即 MDA 框架下 PIM 到 PSM 的映射，目标是描述一个完整系统的 PIM 模型集分别映射到具体的实现平台上得到相应的 PSM 模型集，目前已经实现了 PIM 静态模型到 PSM 静态模型的转换，也就是类图和配置图的转换，所以本文只介绍 PIM 类图的设计结果，及其到 PSM 类图的映射结果。

本文研究了一个网上银行系统[26]的模型驱动的开发和测试。网上银行（onlineBank）系统主要提供三种服务：余额查询、交易列表和转账服务，而且使用系统提供的服务前必须登录到系统。本系统假设：(1)如果想成为该系统的客户，必须至少有一个账户；(2)一个客户可以有多个账户，也可以多个客户共同拥有一个账户。根据应用需求，我们列出四个用例：登录、查询账户余额、列出交易内容、转账。图 3 是 onlineBank 系统的用例图。

3.1 构造系统的 PIM

在分析阶段为每个用例设计出类图，最后将各个用例的类图合并，去除冗余类，得到 onlineBank 系统的类图，如图 4 所示，这里简单起见，各个类的属性和操作没有显示。在 UML 描述的类图中，存在三种类型的类：

1. 边界类（Boundary Class）：边界类描述了系统内部的行为和外界环境之间的所有交互，包括与用户通过图形界面产生的交互、与其它角色（例如代表其他系统的角色）的交互以及与硬件设备的通信等。边界对象是边界类的实例，它隔离了系统内部的动作与外界环境。每个角色/用例对对应一个边界对象；在图上用构造型《Boundary》表示。

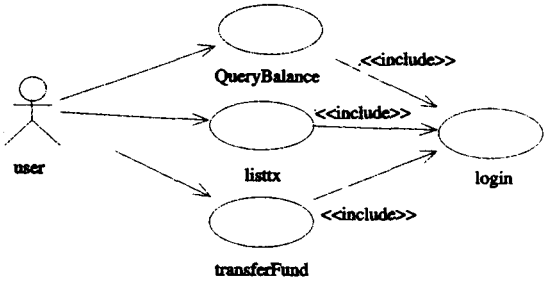

图 3 网上银行系统用例图

2. 实体类（Entity Class）：实体类描述了系统的主要信息，实体对象通常是持久的。实体类的主要作用是描述和管理系统内部的信息，系统中主要的概念在模型中通过实体类来体现；在图上用构造型《Entity》表示。

3. 控制类（Control Class）：控制类用来对系统的行为建模。控制对象不必实现系统行为，而是通过与其他对象合作实现用例。这种将行为与模型附带的潜在信息分离的做法使得系统发生改变时可以独立处理变化的部分。在图上用构造型《Control》表示。

本文为了说明模型驱动的软件测试，选取其一个典型功能片断的类图进行处理，图 5 为客户注册到帐户这一功能片断的类图。

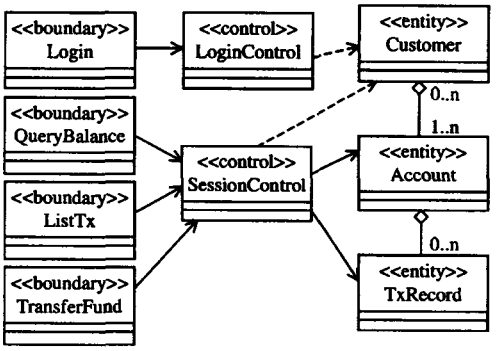

图 4 网上银行系统的类图（PIM）

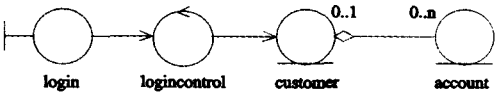

图 5 客户注册的类图

3.2 由 PIM 映射到 PSM

本文中用 UML4EJB 描述 PSM，根据文[5]中的定义的 UML 元模型到 UML profile for EJB 元模型的映射，定义 PIM 中的模型与 PSM 模型的映射。对于 PIM 类图到 PSM 类图的映射，分几个步骤处理：(1)PIM 类图中的每个类应该映射成 PSM 类图中的相应的类或 EJB 子系统；(2)PIM 类图中类与类之间的关系映射到 PSM 类图中相应类与 EJB 子系统之间的关系，转换过程中 PSM 新增加的类之间存在制定的关系；(3)PIM 类图中每个类所包括的属性和操作以及类本身的一些性质如可见性、修饰符等映射到 PSM 类图中相应部分，同时在 PSM 中添加 EJB 特有的属性和方法。图 6 是由网上银行系统的 PIM 类图映射得到的 J2EE 平台相关模型（PSM）中的类图。第 2 部分介绍了模型驱动的软件开发是模型驱动的软件测试的基础，从这个层面上看，完成了 PIM 到 PSM 的映射说明了模型驱动的软件测试具备了条件，文[8]定义的 PIM 与 PSM 之间的映射规则也是测试用例映射的基础。

4 模型驱动的软件测试方法

由于目前只完成 PIM 静态模型到 PSM 静态模型的映射，所以本文的模型驱动的测试方法研究目前也是基于静态模型，特别是类图。当系统开发由分析过渡到设计时，需要加入特定平台实现相关的信息，MDA 定义了 PIM 到 PSM 的映射规则，从元层上实现源模型（PIM）到目标模型（PSM）转换。传统的基于模型的测试方法不考虑平台相关的信息与平台无关的信息的分离，或者为特定平台定制，或者作为一般的方法只考虑平台无关的模型信息。为了在生成和执行测试上充分利用 PIMS 和 PSMS 的分离，本文在传统基于模型的测试的基础上提出一个模型驱动的测试策略，并将传统的测试任务精化成两步：首先根据给定的测试覆盖准则，用已有的基于 UML 模型的测试用例生成方法从 PIM 生成平台无关的测试用例 PIT（包含预期输出）；然后在 PIMS 映射到 PSMS 的同时，使用相应的转换规则将 PIT 映射成相应平台下可直接执行的 PST。对于测试的执行仍然与原来基于模型的测试方

法一样，使用工具实现自动测试，或者特别的测试必须手工进行，记录测试结果，并对测试有效性作评估分析。本文主要研究模型驱动的测试方法，先基于 PIM 类图生成测试用例，然后定义并实现 PIT 到 PST 的映射。

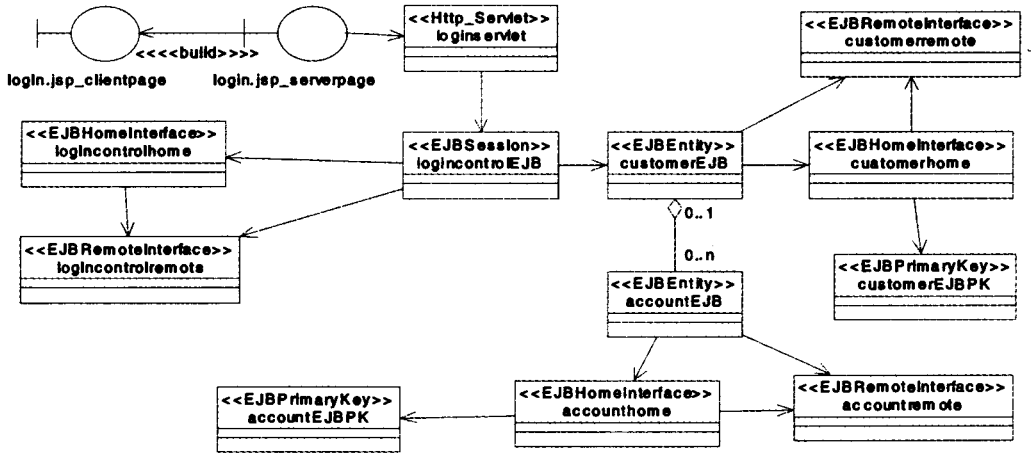

图 6　网上银行系统在 J2EE 平台上的类图

4.1　测试用例

构造测试用例是软件测试过程中的第一个关键步骤，测试用例识别测试中要实现的条件，是验证软件实现被需求所成功接受必须的。在系统分析阶段描述系统时，UML 用例图、交互图、类图及状态图是一个系统原型模型的最小提交集，同时，这里的模型都是用普通 UML 描述的，显然属于 PIM。我们前一阶段的工作是研究从 UML 模型生成软件测试各个阶段的测试用例的方法，图 2 所示的测试过程中也可以看到，可以从用例图生成用于系统测试阶段的测试用例，交互图可以用于生成集成测试的测试用例，而类图以及相应的状态图可以用于生成相应类的单元测试的测试用例。

测试用例一般包含输入和预期输出，但会因测试类型的不同和生成测试用例的信息来源的不同而不同。本文基于 UML 类图生成测试用例，类图描述了系统的结构关系，测试的目的也是测试类之间关系是否正确，这样的测试用例属于结构测试的测试用例，也成为白盒测试用例，为了便于对测试用例的映射处理实现工具支持，本文给出测试用例的形式化定义：

定义 1(测试用例)　测试用例 *TC* 可以表示为一个五元组：$TC = \langle ID, precondition, inputseq, expect\ result, outcome \rangle$。其中：

- *ID*，是测试用例的唯一标识号；
- *Precondition*，执行该测试用例的前提条件，可以为空，也可以为必须先执行的测试用例的 *ID*；
- $Inputseq = \{(object, attribute, value) \mid (object, method, ->) \mid (eid, event)\}$，输入参数序列，包括变量名及其所属对象和输入值；可以是具体变量的输入值，可以是对象方法调用的偏序序列，也可以是系统级的事件流序列；
- *Expect result*，测试输入引起的可观测的结果的描述，可以是一组具体变量的输出值，或者是系统级的输出事件；
- $outcome = (p \mid f)$，该测试执行后的判定结果，p 表示测试通过，f 表述测试不能通过，也即引发错误。

4.2　从 PIM 生成 PIT

图 5 的网上银行系统客户注册功能片断的 PIM 类图描述了系统中的几个类，类之间的联系，包括联系的类型和重数，当 PIM 类图向 PSM 类图转换时，这样的关系应该被保持并实现，类间联系可能的故障模型包括不正确的重数、更新/删除异常、遗漏链和错误链几种情况，测试就是要发现类联系的实现中的上述错误，由于在 MDA 框架下，PIM 可以映射为 PSM，PSM 可以直接生成代码，所以要验证作为实现模型的最终代码是否正确实现了需求，只需要从 PIM 类图中生成测试用例，用于测试逻辑关系的实现中有无错误即可，本文使用文[6]中方法基于图 5 的网上银行系统的 PIM 类图中为测试重数关系是否正确生成的一个测试用例为：

{tc1,{},{(account,multiplity,0),
customer,multiplity,1)},{},{p)}

因为 PIM 中没有平台实现的任何信息，是软件的核心逻辑的描述，所以基于 PIM 生成的测试用例也是平台无关的，称之为 PIT。这样的测试用例只能是待测试系统逻辑的一般测试，在最终的软件实现平台上还不能直接执行，所以在 PIM 向 PSM、PSM 向代码转换的同时，还需要对这些 PIT 进行相应的映射处理，使得得到的平台相关的测试 PST 能够直接在相应平台上执行。

4.3　从 PIT 转换到 PST

要解决 PIT 到 PST 的映射问题，需要 PSM 转换到实现的有关知识，从平台无关模型中生成的 PIT 可以在一个一般的测试环境下执行的应用逻辑测试，这样的测试可以复用到软件的实际测试环境下平台相关的实现版本的测试。

本文研究的目标平台是 J2EE，由于 J2EE 平台上基于 Web 的应用软件运行特征与常规软件不同，系统级行为是客户通过呈现在用户面前的客户端的 JSP 之间的交互实现的，所以进行系统测试时测试人员应该输入信息激活相应功能后，系统执行该功能并作出系统级响应(输出)。用基于模型的测试方法从 PIM 生成的测试用例，是与系统的实现技术无关的，但是最终执行系统测试是在系统已经实现后才进行的，软件系统与具体平台上的实现技术相关，所以测试用例必须也进行映射才能直接执行。由于系统测试的输入、输出都是在客户端进行的，一般使用黑盒方法生成，与具体实现技术无关，所以系统测试的测试用例可以直接复用到 PST 中；对于用灰盒方法生成的集成测试用例和白盒方法生成的单元测试用例，由于其中包含了 PIM 中相关的属性、方法、消息等模型元素，而这些元素随着 PIM 到 PSM 的映射，在目标平台上有新的变体，所以要使得测试用例在目标平台上能够执行，必须将测试用例中这些模型相关的元素进行映射，而模型无关的

成分直接复用。

从前面 PIM 到 PSM 的映射转换可以知道在 PIM 中实现业务逻辑的类及类间的关系的信息，在 PSM 中得到了保持，但在 PSM 中由相应的 JSP、session Bean 和 entity bean 及其间相应的关系实现。所以 PST 不仅要保持 PIT 中的信息，还要将模型相关的成分映射成目标平台下 PSM 中相应成分，本文提出了一个将 PIT 映射为 PST 的算法，PIT 是基于普通 UML 的，而 PST 是针对 J2EE 平台的，该算法输入为 PIT 的全部信息，输出 PST 的全部信息。算法的具体描述如下：

算法 PIT2PSTmapping(PIT suite, PST suite)

```
输入  PIT suite
输出  PST suite
class PIM2PSMmaping{
    //PIM 到 PSM 的映射对照表
    PIMelements:UML;//PIM 中的元素
    PSMelements:UML4EJB;//PSM 中元素
}
class PIT suite{
    ID:string;
    Precondition:string;
    Inputseq1:{(object,attribute,value);
    Inputseq2:(object,method,->);
    Inputseq3:(eid,event)}};
    Expect result:string;
    Outcome:enume of (p ,f);
    Flag:Boolean;
}
class PST suite{
    ID:string;
    Precondition:string;
    Inputseq1:{(object,attribute,value);
    Inputseq2:(object,method,->);
    Inputseq3:(eid,event)}};
    Expect result:string;
    Outcome:enume of (p ,f);
}
PIT2PSTMapping(PIT suite, PST suite){
  Class TempPIT{ //局部变量
    ID:string;
    Precondition:string;
    Inputseq1:{(object,attribute,value);
    Inputseq2:(object,method,->);
    Inputseq3:(eid,event)}};
    Expect result:string;
    Outcome:enume of (p ,f);
}
begin
while PIT suite 中存在未处理的测试用例{
    取 PIT suite 中下一条未处理的测试用例 tempPIT,
    if tempPIT 是黑盒测试测试用例{
    将 tempPIT 复制到 PST suite 中;
    标记该测试用例已处理,指针下移,继续处理;
    }
    else{
    分析 tempPIT 中的输入和输出,提取出 PIM 中的元素;
    并在 PIM2PSM 对照表中找到对应的 PSM 中元素,修改 tempPIT 中相应属性;
    将 tempPIT 复制到 PST suite 中;
    标记该测试用例已处理,指针下移,继续处理;
    }
    }endwhile;
```

利用该算法为图的 PIT 实现映射后的 PST 为：

{tc1,{},{(accountEJB,multiplity,0),customerEJB,multiplity,1}},{},{p}}

这样的测试用例可以直接在由 PSM 生成的 J2EE 平台上的可执行代码上用于执行测试，可以用类似的方法将由 PIM 中其他模型生成的系统测试用例、集成测试用例、单元测试用例经过转换生成在最终平台实现上可以执行的测试用例。这样可以按照 2 中描述的在 MDA 框架下与模型驱动的软件开发过程集成的模型驱动的软件测试过程实现对一个系统的完全的测试。

4.4 支撑工具框架

为了对 2 节中过程和 4 节中的方法提供工具支持，本文提出了一个工具框架 MDTOOL，这是建立在文[15]中测试生成工具 UMLTGF 基础上的，由于我们目前只实现了 UML 到 J2EE 平台的映射，所以模型驱动的测试也只是实现了到 J2EE 平台的映射。我们使用普通的 UML 为 MDTOOL 建模，本文给出 MDTOOL 的类图，如图 7 所示。相应的功能描述如下：

UML 模型分析器。基于 OMG 提出 XMI 文档交换规范，能够直接读取 UML 表示的 PIM 模型的 XMI 文件（如 MDL 文件）[9]，进行解析，并获取相关的信息存入相应的中间表，以便测试用例生成器使用。

PIT 生成器。包括系统测试用例生成、集成测试用例生成、单元测试用例生成三个子模块，每个模块使用基于模型的测试方法实现，从 PIM 中的用例图可以生成系统测试用例，从 PIM 中的交互图和类图可以生成集成测试用例，从 PIM 中的类图和相应的状态图可以生成单元测试的测试用例，最后形成测试包（test suite）。

PIT2PSTMapper。对照 UML profile for EJB，将系统生成的 PIT 映射成 J2EE 平台上的 PST，并将系统测试用例、集成测试用例、单元测试用例分别组织。

测试用例管理。为使得生成的测试用例集可复用、可维护、无冗余，对每个测试用例设计了增加、修改、删除操作，用来在测试用例生成过程中管理这些测试用例集，使得测试用例能够被最大限度复用。

5 相关工作

基于模型的测试技术有三个关键技术：建模表示，测试生成算法，工具支持。用一种易理解、易表示的建模语言表示复杂的系统，使得能够被测试生成工具理解。工业界对 UML 的支持比较普遍，UML 提供了分别属于静态模型和动态模型的 9 种模型图，用于对面向对象系统进行建模，但由于不同的人在做系统分析和设计时建模的选择可能不一样，所以在工业界还没有一个直接使用软件构造过程的模型生成完整的测试的工具体系。在基于 UML 建模的面向对象软件测试研究领域，基于模型的测试方法主要是分别在软件开发的分析、设计、代码阶段基于各个阶段生成的分析模型、设计模型、实现模型生成相应的测试，用于测试最终的软件系统。而且目前主要集中在从 PIM 生成测试的研究上。文[9,10]介绍了从状态图生成测试用例的方法，文[11～13]描述了从顺序图生成测试用例，文[14,15]描述了从协作图生成测试用例的方法，类图、组件图和配置图用于配合动态模型生成测试的同时，用于静态测试所用的方法主要分形式化方法、半形式化方法和非形式化方法，大多数方法都实现了工具支持。

基于模型的测试实践为模型驱动的软件测试方法奠定了基础。模型驱动的测试方法研究是在 MDA 框架下的一个重要的课题，随着模型驱动的软件开发方法学逐步成熟并奠定了基础，就顺其自然地提出了模型驱动的软件测试，不过已发表的相关成果还很少，文[16]提出了一个模型驱动的软件测试方法，基于图转换的方法，实现一个 Web 系统的开发，在平台无关的可执行的图模型上生成测试用例，并把平台无关的模型作为用于生成预期结果的理想模型，用设计模式实现平台无关的测试到平台相关的测试的映射。本文的方法基于研究小组 MDA 框架下模型转换的研究和基于 UML 模型生成测试用例的研究工作，使用普通 UML 表示 PIM，用 UML4EJB 表示 PSM，容易使用，使得软件开发的重心可以前提到业务建模阶段，整个过程可以实现工具支持。但 UML 固有的缺陷，即语义描述不够精确，难以直接表示复杂的业务

逻辑，同样在这里阻碍该方法的通用化，另外本文只研究了基于 PIM 的类图的测试生成及映射工作，PIM 模型集中还有其他模型，特别是动态模型是生成动态测试的测试用例的基础，要使得从 PIM 中生成的所有测试用例都映射到 PSM 上，才能使系统的模型驱动的软件测试方法用于解决实际的测试问题。

结论和将来的工作 本文从过程、方法与工具三个方面研究了模型驱动的测试，首先提出一个基于 MDA 的、与软件开发过程集成的模型驱动的软件测试过程；其次讨论了与过程对应的模型驱动的测试方法，用基于模型的测试方法基于平台无关的业务模型生成一个一般的平台无关的测试(PIT)，在 PIM 到 PSM 的映射过程中，定义并实现从 PIT 到 PST 对应的映射，使得 PST 能够直接在相应平台上执行以测试对应层软件实现是否与规约一致；最后，对上述过程中的方法提供工具支持。这种模型驱动的软件测试在两个方面比传统软件测试有明显优势：一是当软件的实现技术变化或平台迁移时，通过定义 PIM 到新的 PSM 的映射实现软件，同时定义并实现 PIT 到新的 PST 的映射，这样通过复用 PIT 节省了测试的成本；二是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 PIT，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。模型驱动的测试用了系统的测试包生成方法，在软件生存期早期揭示软件的缺陷，提高了测试质量，支持复用从而减少了整体测试的时间，减低了复杂度，达到了更高的测试覆盖度。

由于需求的变更由模型来捕获了，只要更新模型然后重新生成测试，并将变更影响到的测试加入到原有的测试中，很大程度上减低了测试维护的成本。使用工具又加强了组内的交流，模型、测试包的追踪为待测试系统和测试提供了一个无二义的唯一的视图。模型驱动的测试分离了测试逻辑和测试实现，测试人员集中于开发特定于应用的好的测试，而依赖于测试工具集的测试执行环境解决测试执行的问题。

MDA / UML 提供了一种支持面向对象的软件开发的全过程建模方法。我们所做的工作有利于缩小需求分析和系统设计阶段的差距，我们今后的工作一是解决 PIM 中动态模型到 PSM 的转换，以及相应的测试转换方法；二是实现一个系统到异构平台的迁移，复用 PIM 和 PIT，定义到新的平台的 PSM 和 PST 的映射算法，从而验证 MDA 的思想；三是设计一个 MDA 下系统的模型驱动的软件测试工具。

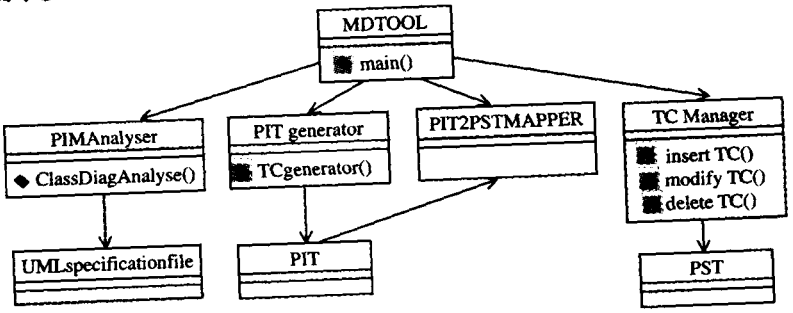

图 7 MDTOOLs 的类图

参 考 文 献

1 鲍志云译．解析 MDA．人民邮电出版社，2004．2

2 鲍志云译．应用 MDA．人民邮电出版社，2003．11

3 UML Specification 1．5[S]．available at http://www．omg．org/uml

4 Ahmed K Z，Umrysh C E．Developing Enterprise Java Applications with J2EE and UML

5 Sun Java Community Process JSR-26．UML/EJB Mapping Specification，Version 1．0．May 25，2001．Available from http://jcp．org/jsr/detail/26．jsp

6 Binder R V 著，华庆一，王斌君，陈莉 译．面向对象系统的测试[M]．人民邮电出版社，2001

7 Booch G，Rumbaugh J，Jacobson I 著，周伯生，冯学民，樊东平 译．统一的软件开发过程[M]．机械工业出版社，Addison-Wesley 2001

8 崔萌，等．基于 MDA 的 PIM 到 J2EE 平台 PSM 的转换方法．计算机应用与软件，2003(已录用)

9 Offutt A J，Abdurazik A，Generating Tests from UML specifications．[C]．In：Proc．of UML'99，Fort Collins，CO，Oct．1999．416～429

10 Hartmann J，Imoberdof C，Meisenger M．UML-Based Integration Testing．[C]in ISSTA 2000 conference proceeding，Portland，Oregon，Aug．2000．60～70

11 Choi B，Yoon H，Jeon J．A UML-based Test Model for Component Integration Test．[C] Workshop on Software Architecture and Component，Japan，1999，12：63～70

12 Basanieri F，Bertolino A．A Practical Approach to UML-based Derivation of Integration Tests．[C] Proc．of QWE2000，Bruxelles，November

13 Fraikin F，Leonhardt T．Sequence diagram based test automation．http://www．pi．informatik．tu-darmstadt．de/publikationen/technische%20Berichte/2002/pi2002-2．pdf

14 Offutt A J，Abdurazik A．Using UML Collaboration Diagrams for Static Checking and Test Generation．[C] Proc．3rd Intl．Conf．on the Unified Modeling Language (UML'00)，York，UK，Oct．2000．383～395

15 王林章，李宣东，郑国梁．一个基于 UML 协作图的集成测试用例生成方法．电子学报，2004，32(8)

16 Heckel R，Lohmann M．Towards Model-Driven Testing，TACoS - International Workshop on Test and Analysis of Component Based Systems，Warsaw，in conjunction with ETAPS 2003，2003