一种运用模式将CIM转换到PIM的方法*)

曹晓夏　缪淮扣　孙军梅

(上海大学计算机工程与科学学院　上海 200072)

摘　要　模型转换在MDA软件开发方法中扮演着非常重要的角色,尤其是从CIM到PIM的转换。本文给出了一种从CIM转换到PIM的方法。在CIM中,我们通过特征模型来组织需求,同时用软件体系结构来组织PIM中的各个要素。这个转换中的核心内容是模式的应用。在CIM的需求模型中,本文将特征分层,从而将需求分为不同的层次。同时模式也被分为不同的层级,其中包括体系结构模式和设计模式。针对不同层级的特征模型,应用不同层级的模式进行变换,从而得到分层的体系结构。当需求发生变化时,首先确定这种特征的变化是在哪个层级上的,然后在不同的体系结构层级上变换相应的功能,从而实现PIM的相应变化。本文最后以自行开发的Object-Z的支持工具为例来说明所给出的方法。

关键词　MDA,CIM,PIM,特征模型,模式,责任

An Approach to Transforming from CIM to PIM Using Pattern

CAO Xiao-Xia　MIAO Huai-Kou　SUN Jun-Mei

(School of Computer Engineering and Science Shanghai University, Shanghai 200072)

Abstract　Model Transformation plays an important role in MDA, especially in the transformation from CIM to PIM. This paper presents an approach to transforming CIM to PIM in a feature-oriented view. We use the feature model to construct user's requirements in CIM, and use software architecture to organize elements in PIM level. The application of pattern is the core in this transformation. We partition the features in different layer. So the requirement is partitioned into different layer too. At same time, the pattern is partitioned into different layer such as software architecture pattern and design pattern. We apply these patterns in different layered feature model, then produce a layered architecture. Once the requirement changes, the function will change in the corresponding architecture. Finally, we use Object-Z support tool we developed as an example to demonstrate our approach.

Keywords　MDA, CIM, PIM, Feature mode, Pattern, Responsibility

1　引言

模型驱动构架(MDA)是OMG定义的一个软件开发框架。在MDA中,模型在软件开发过程中扮演了非常重要的角色,软件开发过程是由对软件系统的建模行为驱动的[16,17]。模型是以精确定义的语言对系统(或系统的一部分)做出的描述。在MDA中,包含4种模型:CIM(Computation Independent Model,计算无关的模型)、PIM(Platform Independent Model,平台独立的模型)、PSM(Platform Specific Model,平台相关的模型)、code(代码,这里把代码看作是特殊的模型)。而MDA开发软件的过程就是从CIM到code依次转换的过程。当前的研究集中在PIM到PSM的转换方面,而较少人关注到CIM到PIM的转换。很多的研究者认为,基于已有的技术,PIM到PSM的转换可以实现自动化,而从CIM到PIM的转换则较为困难。CIM对需求的模型化常常缺少一个好的结构,CIM到PIM的转换很大程度上取决于设计者的经验和创造力,因而PIM的质量就很难把握[1]。而我们认为,从CIM到PIM的转换是软件开发中极其重要的部分,是从"做什么"的需求描述到"怎么做"的设计描述的转换过程,也是软件精化的核心问题。我们的工作主要集中在怎么样进行从CIM到PIM的转换。文[1,10]在这个方面作了有益的探索。文[1]中,通过将特征分解为责任(responsibility),将责任组合称为组件,责任间的关系构成为组件间的关系,从而形成了系统的体系结构,完成了PIM的设计。

本文给出了一种将CIM转换到PIM的方法。对CIM中关于需求的描述,我们采用面向特征的视角来描述领域模型。所谓特征模型是包含一系列的特征以及特征之间关系的模型。特征驱动的开发是围绕一套核心的"最佳实践"来建立的。被选择的实践都不是新的,但它们的特定组合却是新的,每个实践都补充并且加强了其他的实践[14]。特征作为具有客户价值的功能用于驱动和跟踪开发过程[14]。特征模型从系统功能的角度描述了系统需要"做什么"。另一方面,PIM模型实质上是一种较高层次的设计模型,体系结构是PIM的核心内容,PIM通过体系结构将各种系统元素组织在一起以满足需求。关于体系结构,我们关注它两个方面的特性:其一是各个组件的划分,其二是组件之间的交互。由于特征是可调度的系统功能[15],而组件是由特征的组合——特征包来组成,这样组件与特征之间存在1对n的对应关系。而组件之间的交互则成为特征包之间的交互。PIM以体系结构为中心组织得出系统的设计模型,表达出较高层次的"怎么做"。

在从"做什么"到"怎么做"的转换过程中,本文引入了体系结构模式。模式作为一个问题族的一般解决方案,有助于提供一个被证明良好的一般计划[5]。模式通过描述核心组件、组件的职责和相互关系以及它们结合的方式来解决问题。每个模式处理一个软件系统的设计或实现中的一种特殊的重复出现的问题[5]。模式可以用来构建具有特定属性的软件体

*)本文受国家自然科学基金项目(批准号601730301)和国家973项目(编号:2002CB312001)资助。曹晓夏　博士生,主要研究方向为软件构架技术、形式化方法;缪淮扣　教授,博士生导师,主要研究方向为软件工程、形式化方法。孙军梅　博士生,主要研究方向为软件体系结构、形式化方法。

系结构[13]。我们通过对系统的特征模型进行操作，对应相应层次的模式，给出相应的体系结构。

2 MDA 中的模型转换

MDA 是一个贯穿整个企业级系统开发生命周期的方法，整合了软件、硬件、人员以及业务实践等元素，提供了一个系统的框架，利用工程上的方法来理解、设计、操作、进化企业级系统的各个方面[8]。在 MDA 中，软件开发过程是由对软件系统的建模行为驱动的[3]。MDA 开发生命周期和传统的生命周期的主要不同在于开发阶段创建的工件性质不同，它包括 CIM、PIM 以及 PSM。MDA 的开发过程就是从 CIM 到 PIM，再到 PSM，最终到 code 的依次转换过程，如图 1 所示。从 PIM 到 PSM 的转换，在业界已经有了较为具体和可行的方法，在文[6,7] 中有详细的定义与叙述。文[12]在这方面做了卓有成效的工作。而本文所关注的是如何进行从 CIM 到 PIM 的转换。

CIM 在有些时候被称作是业务模型或领域模型，它是从计算无关的视角来观察系统的。之所以说 CIM 是计算无关的，是因为它关注的是系统环境以及系统的需求，而忽略系统的组织结构和数据处理细节。CIM 在领域专家的与系统设计人员的沟通方面扮演了重要的角色[7]。

PIM 是从平台无关的视角来观察系统的。它描述了系统的结构与过程，而不涉及到具体的平台。PIM 不考虑操作系统、编程语言、硬件以及网络[6]。当然这种平台无关是相对的。一般来说，平台是指一个抽象概念上的平台。

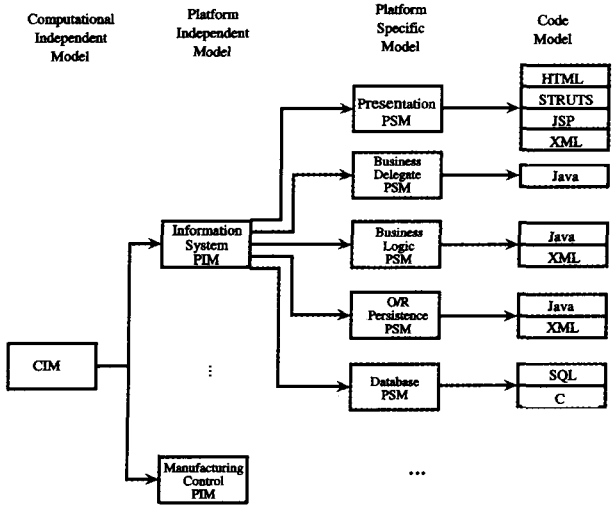

图 1 MDA 中各模型之间的转换

3 特征模型

3.1 特征模型的概念

特征是从用户的角度对系统感知。用特征对系统需求规格说明进行模块化组织是一种非常自然的手段[13]。这种面向特征需求规格说明的组织方式在 FODA(Feature-Oriented Driven Architecture)方法中，被引用到领域工程的研究和实践中。FODA 使用特征和特征之间的关系——特征模型——作为领域模型的重要组成部分。从需求规格说明的组织结构角度看来，特征提供了一种需求的分割组织方式，即以特征作为需求空间内的一阶实体，系统具有的特征及其相互关系构成了系统的需求空间；从需求的内涵来看，一个特征体现了系统具有的某种能力或特点，反映了需求获取的参与者对系统某种要求或理解；从需求的类型上看，一个特征可能是一种功能性的需求，或是对系统质量属性的要求，或是外部环境对系统的某种约束条件[10]。

3.2 特征模型的组织结构

在这里的特征模型中，我们采用层次的方式来组织特征，特征之间以整体-部分关系(whole-part association)联系在一起。在特征模型中，整体部分关系主要用来表示两种类型的语义：(1)整体特征对部分特征的控制和协调作用；整体特征对部分特征逻辑上的紧密结合性[10]。需求工程把软件需求分为 3 个不同的层次：业务需求、用户需求、功能需求[15]。文[10]中，将业务需求、用户需求、功能需求中所具有的特征分别称为服务(service)、用例(use case)、功能(function)。这样一个软件系统的需求规格说明表示为如图 2 所示的结构。每一个服务涉及到的用例组织成一个用例集合，在系统的边界上，用户通过用例与系统发生交互，从而完成特定的业务需求。

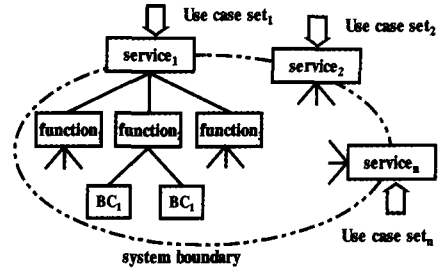

图 2 系统功能性特征的静态关系

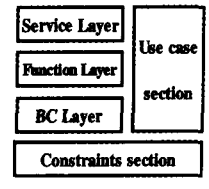

图 3 特征模型的具体形式

通过对系统需求的分析，我们得到特征模型的具体形式，如图 3 所示。该特征模型记录了系统具有的服务、功能、行为特点(behavior character)等特征以及特征之间的约束关系。用例部分的特征通过特征间的依赖关系与服务、功能或行为特征建立联系。

需要明确的是，术语“特征”是指那些可被调度的功能单元。我们使用用例去捕获和表示功能需求，但是用例主要不是从计划的角度得到的，因此不可能一直适合调度。同时还要引入另一个术语——特征来进一步组织需求模型，特征的引入意味着灵活性以及复用。

3.3 特征的精化

CIM 所描述的需求由几组不同的特征组成，而特征又可以精化为相对具体的一组特征。特征的精化是特征间的二元关系。不同抽象层次的特征组合构成系统的层级结构。而这种层级结构有效地描述了复杂系统的需求。文[1]中介绍了三种精化定义，精化可以被进一步分为三种具体的方法：分解(Decomposition)、特性(Characterization)、特化(Specialization)。三种方法的定义如表 1。

表 1 精化方法

精化方法	定义
分解	将一个特征分解为它的各个组成特征
特性	识别一个特征的不同属性构成新的属性
特化	为一个通用的特征加入进一步的细节

在各种精化方法中，我们称较高抽象层次的特征为父特

征，精化的结果——较低抽象层次的特征称为子特征，它们之间的关系如表2。

表2 精化特征之间的关系

精化方法	父特征角色	子特征角色
分解	整体	部分
特性	实体	属性
特化	通用实体	特化实体

4 模式与体系结构

4.1 理解模式

模式是构造高质量软件体系结构的一个重要工具。模式很好地适应了软件体系结构现有的方法，它们建立在用来构造定义良好的软件系统的启用技术的基础上。

对模式的认识分为三种境界：

第一层境界：认为模式就是一种解决方案，这种境界认识到模式怎么去解决一个问题。

第二层境界：除了了解模式提供的解决方案以外，还能够认识到模式背后的指导原则。

第三层境界：认识到模式其实是一些关系，用来舒缓系统内部的冲突力。

模式提供的解决方案体现了这一点。因为，最终的解决方案肯定是一些模块通过彼此的交互建立起的一种关系来完成所需的功能。

模式并不是简单地给出了一些概念间的关系，它还具有更深刻的内涵：自身的意图、动机、适应性以及核心解决方案。一旦确定使用了一个模式，那么这个模式就为我们搭建了一个内容丰富的场景，这个场景可以非常有效地指导我们进一步的工作，启发我们发现新的对象。这样，基于模式构建的系统体系结构就变得抽象而不空泛，并且具有很好的延伸性。

4.2 模式与体系结构

软件体系结构是对子系统、软件系统组件以及它们之间相互关系的描述。子系统合组件一般定义在不同的视图内，以显示软件系统的相关功能属性和非功能属性。系统的软件体系结构是一件人工制品，这是软件设计活动的结果[13]。

判断模式取得成功的一个重要准则是它们在多大程度上达到了软件工程的目标。模式必须支持复杂的、大规模系统的开发、维护以及演化。模式针对软件体系结构的一个重要目标——用以定义属性进行特定的软件体系结构的构造[13]。

软件体系结构模式描述了软件系统基本的结构化组织方案，它们提供了一套预先定义好的子系统来制定它们的职责，包括用于组织它们之间的规则和指南[12]。体系结构模式代表了模式系统中的最高等级模式，它有助于明确一个应用的基本结构。之后的每个开发活动都遵循这种结构。文[13]中，给出了4类、9种体系结构模式以及5类、8种设计模式。我们在模型的转换中，分层使用不同类型的模式。

本质上，软件体系结构是对软件需求的一种抽象解决方案。在引入了体系结构的软件开发之后，软件体系结构架起了软件需求与软件设计之间的一座桥梁[15]。

5 从CIM到PIM的转换

本节给出如何将特征模型描述的需求模型CIM转换为以体系结构为中心的设计模型PIM。并且，由于软件开发是增量式迭代的开发过程，我们给出需求模型发生改变时相应的设计模型的变换方法。

5.1 转换概述

图4给出了从CIM转换到PIM的概要。在这个转换当中有三个核心概念：其一为需求层次的模型，在这个层次上，需求被组织成特征模型。其二为设计层模型，在这个层次上，各个基本元素被以体系结构为中心的框架组织在一起。在这两个层次的转换中，另一个引入的核心概念是模式。

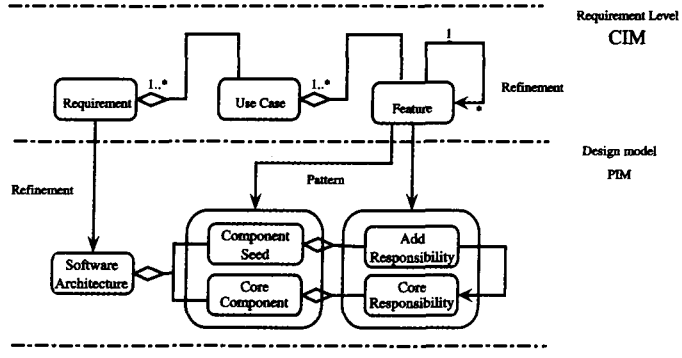

图4 CIM到PIM转换的概要图

在3.2节我们已经知道，需求的捕获首先是用用例完成的。在用例中提取特征，组织成为特征模型。在特征模型中我们将特征分层，并将特征划分为不同的特征包。抽象这些特征，寻找相对应的系统结构模式。应用这个模式得到相应的解决方案，这个解决方案给出了系统的体系结构，构成系统相对稳定的核心组件以及它们之间的关系。而相应的特征又构成体系结构中CRC类的职责(Responsibility)。当增加新的特征时，在相应的组件中增加相应的职责，或者构成新的组件，加入到大的体系结构中去，最终构成系统的PIM。这样就完成了从需求到设计的精化。

5.2 对特征的操作

对特征进行操作的目的是使特征的粒度能够适应相应体系结构模式的粒度要求，以便应用模式，得出相应体系结构——即功能组件以及组件间的接口。

特征的操作包括以下三种：分层、精化以及抽象。它们的定义如表3所示。

表3 特征操作的类型

特征操作	简要描述
分层	按照不同的抽象程度对特征进行分析
精化	特征的分解、特性与特化
抽象	提取特征的共同特性，提高特征的抽象层次，它是精化的逆过程

经过对特征的这些操作，我们得到了一个层次清晰的需求模型。

5.3 模式的层级应用

诺贝尔奖获得者赫伯特·A. 西蒙曾论述到："要构造一门关于复杂系统的比较正规的理论，有一条路就是求助于层级理论。我们可以期望，在一个从简单性进化到复杂性的实践中，复杂系统是层级结构的。"对于软件系统来说，层级原理是分析和构建它的基本原则。对需求模型中的特征分级，可以帮助我们运用不同层次的模式，建立不同层次的体系结构。

在转换的概述中，我们已经知道，一个CRC类中包含多个职责，而每个职责都对应一组相应的特征。这些职责又可以进行特征的操作(包括分层、抽象以及精化)，在这个层级上除去体系结构模式外，还可以应用设计模式，得到更进一步细

化的设计模型。

5.4 转换操作

针对前面得到的层级特征需求模型，我们从不同的层次上使用模式，得到相应的层级体系结构。从对模式使用的角度看，我们把转换的操作变换分为两种，其定义如表 4 所示。

表 4 转换的操作

转换操作	简要描述
映射	在由特征需求模型到设计模型的转换中，特征与相应的设计模块具有对应关系
转换	在由特征需求模型到设计模型的转换中，需求的结构与设计的结构不是对应关系

例如：在第 6 节给出的案例分析中，应用管道过滤器模式得到的编译器的体系结构，所做的转化操作为映射，而运用 MVC 得到编辑器模型的转换操作称之为转换。相应的特征包中的特征对应为 CRC 卡片中的对应职责(Responsibility)。在下一层次的精化中，将职责中对应的相应特征进行转换操作。

5.5 需求的变化

由于软件开发过程是一个迭代和增量式的开发过程，因而所得到的模型必须是可以扩展的。需求模型的变化是必然的，针对这种需求的变化，是否可以做出相应的变化，同时不改变系统的体系结构，是开发中的重要因素，这种要求保证了系统体系结构的稳定。当然，这种变化的必要条件是首先建立的是核心模块，确立核心模块以及它们之间的关系。

当需求发生变化，增加新的特征时，应该首先确定这种变化在特征模型的哪一个层次，并且对应于相应模式中的哪一个模块中，然后对这个模块进行修改或增加。

当已经精化的特征发生变化时，可以根据体系结构与特征之间的对应关系，跟踪这种变更对应的职责，继而修改相应的模块。

6 案例分析——Object-Z 支持环境

本节以我们自行开发的 Object-Z 支持工具为例来说明本文给出的方法。这里的 Object-Z 支持工具的功能包括 Object-Z 的输入、处理、输出。如表 5 所示。

表 5 Object-Z 的第一层特征模型

特征	简要描述
输入	系统接受来自键盘的、鼠标的输入
处理	编辑输入的 Object-Z 规格说明，进而对其进行编译，这是系统的核心数据和功能
输出	给用户显示信息，获得来自模型的待展开数据

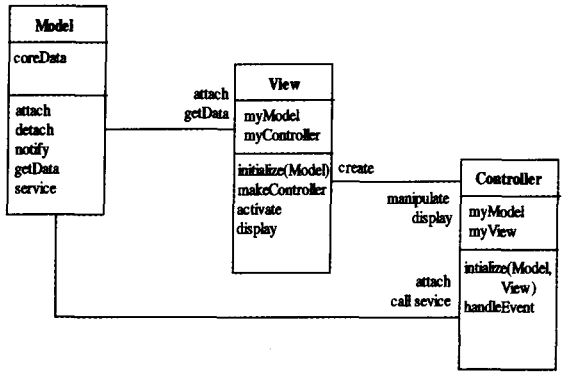

图 5 Object-Z 的基本类图

根据 Object-Z 的这些简要描述，将系统的简要需求对应相应的模型-视图-控制器模式，映射得到基本的系统结构设计模型，也就是第一个层次上的 PIM。限于篇幅，此处只给出相应的类图，如图 5 所示。

可以看到，体系结构模式给出了系统的框架，将系统的功能进行体系结构级别的功能划分，并且给出了各个功能模块间的接口。这是一个系统最根本的、也是最不易变更的部分。

对上述的特征“处理”做进一步的精化处理分为两个大类：其一为编辑功能，其二为编译功能。编辑功能又精化为三个功能，这三个功能又各自可以进一步精化。编译功能又分为四个步骤。表 6 给出了较为详细的特征模型。

表 6 Object-Z 的特征模型精化的处理部分

特征	简要描述
编辑	特征复制、剪切和粘贴的集合
复制	复制当前文件中所选定的内容到剪贴板中
剪切	剪切当前文件中所选定的内容到剪贴板中
粘贴	将剪贴板中内容粘贴到当前文件的选定位置
重复/撤销	特征重复以及撤销的集合
撤销	撤销未存储的最近的编辑操作
重复	重复最近的撤销并未存储的编辑操作
保存	将当前文件存储到磁盘中
解析	特征词法分析、语法分析、类型检查以及语义分析的集合
词法分析	遍历当前文件，查找识别单词、关键字等
语法分析	分析文件的语法是否正确
语义分析	分析当前文件的语义

在这一个层级的特征需求上，我们对对应模块-编译部分进一步精化，利用管道-过滤器模式来映射得出相应的设计模型，得到如图 6 所示的类图。

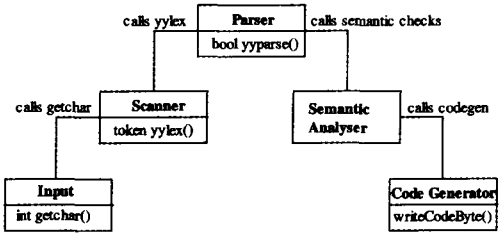

图 6 Object-Z 编译器部分类图

对于特征模型及其各层的精化模型，我们可以给出对应的设计模型，进而得到精化的 PIM 模型。

7 相关工作

文[9]给出了一种应用模式整合体系结构视图的方法，形成系统的框架。在这个方法中，系统 UML 视图是形成体系结构的来源，模式作为映射转换的规则来对系统进行整合。这种方法有利于验证和确认系统，去除设计中的不一致。但是这种设计模型的形成方法，是在同一个抽象层次上进行模型的精化，不能从需求转换为设计。

文[2]也给出了一种从系统目标得到软件体系结构的方法。该方法将系统中的代理(agent)看作软件体系结构中的组件，从系统规格说明的目标中识别出责任，然后分配给代理。组件间的交互则由数据间的依赖关系识别出来。但是这种方法在软件系统的代理较少的时候效果不太理想。

文[1]给出了一种从 CIM 到 PIM 的转换方法，该方法比

较灵活而完整。它利用责任(responsibility)来对特征进行精化,责任的组合构成了体系结构中的组件,完成特征所要求的功能,责任间的关系组合成为组件间的关系。但这种方法并未体现复用。

结论 本文给出了一种将 CIM 转换到 PIM 的方法。这里的 CIM 是由特征模型来描述的,而 PIM 则是由体系结构来描述的。本方法的核心内容是模式的应用,特别是体系结构模式的应用。这个转换过程在不同层次应用模式,从而自然完成从"做什么"到"怎么做"的变换。这种变换不仅仅是形式的变换,而且是本质的变换。而由于模式是有经验的专家经历过长时间的实际应用得出的,所以通过应用模式得出的体系结构是相对稳定和可靠的。

当然,这种转换不是没有缺点的。其一,由于模式有限,而实际的问题空间是变化无常的,所以不是每个问题都有恰当的模式与之对应。在今后的研究工作中,我们将致力于构架一个模式库,以及模式搜索的匹配算法。其二,这种转换并不是完全自动化的,我们需要进一步研究,将这种转换的基础做进一步的形式化,以提高转化的自动化程度。

参 考 文 献

1 Zhang Wei, Mei Hong, Zhao Haiyan, et al. Transformation form CIM to PIM: A Feature-oriented Component-based Approach. Lecture Notes in Computer Science, 2005, 3713: 248～263
2 Van Lamsweerde A. From System Goals to Software Architecture. In: Bernardo M, Inverardi P, eds. Formal A Methods for Software Architectures. LNCS 2804. Springer-Verlag, 2003. 25～43
3 Brown A W. Model driven architecture. Principles and practice Published online: 3 August 2004- Springer-Verlag, 2004
4 Leffingwell D, Widrig D. Managing Software Requirements——A Use Case Approach Second Edition, American: Addison Wesley Inc 2003
5 Gamma E, Helm R. Design Patterns ——Elements of Reusable Object-Oriented Software. American: Addison-Wesley Longman Inc, 1995
6 Pilone D, Pitman N. UML2. 0 in a Nutshell. American: O'Reilly, June 2005
7 Object Management Group. MDA GuideV1. 0. 1. http://www.omg.org/mda/. 12th June 2003
8 Souza D D. Kinetium Model-Driven Architecture Opportunities and Challenges Version 1. 1. ftp. omg. org/pub/docs/ab/01-03-02. pdf
9 Egyed A. Integrating Architectural Views in UML: [Technical Report]. USC/CSE-99-TR-514. University of Southern California Center for Software Engineering, 1999
10 张伟,梅宏. 一种面向特征的领域模型及其建模过程. 软件学报, 2003, 14(8): 1345～1356
11 孙昌爱,金茂忠,刘超. 软件体系结构研究综述. 软件学报, 2002, 13(7)
12 崔萌,袁海,史耀馨,等. 一种基于 MDA 的 UML 顺序图到状态图的转换方法. 南京大学学报(自然科学), 2004, 40(4)
13 Buschmann F, et al. 面向模式的软件体系结构 卷 1: 模式系统. 贲可荣,等译. 北京:机械工业出版社, 2003
14 Palmer S R, Felsing J M. 特征驱动开发方法原理与实践. 雄焕宇,王峰,彭设强,等译. 北京:机械工业出版社, 2003
15 Carmichael A, Haywood D. 快速开发最佳软件. 詹梅,杨卫东,等译. 北京:电子工业出版社, 2004
16 Frankel D S. 应用 MDA. 鲍志云 译. 北京:人民邮电出版社, 2003
17 Kleppe A. 解析 MDA. 鲍志云 译. 北京:人民邮电出版社, 2004

(上接第 264 页)

```
% COMPONENT Class2==[
% PORT pname==datatype; (%IOP o: Class4. type)
□[portbehavior;]
% PORT port1fromClass1==datatype; □[portbehavior;]
...
% PORT portnfromClass1==datatype; □[portbehavior;]
]
```

定义 4 从类图 cd 到 XYZ/ADL 的转换定义为一个映射:

CtoX: cd -> XADL

其中 cds 是类图,XADL 是 XADLelelment 集。若 cd={a1,…,an},

则 CtoX (cd)=⋃AtoX (ai) i=1…n

4 相关工作介绍

目前国内外已有一些关于 UML 和 ADL 相结合描述软件体系结构的研究工作[6～8]。关于从 UML 到 ADL 的转换,文[6]研究了从 UML 状态图、活动图到 XYZ/E 的转换方法,文[7]中提出了一种从 UML 到结构化体系结构描述语言 SADL 的转换方法。关于从 ADL 到 UML 的转换,文[8]介绍了怎么样将 Wright 和 C2 的体系结构概念转换进 UML 的方法。

从 UML 到其他形式化描述方法的转换研究,国内外亦有不少工作[9～14]。它们分别对 UML 静态和动态建模机制中的部分视图进行形式化定义,其中文[9～11]是用基于 Z 的静态描述语言如 B、COOZ 等对 UML 类及类图所进行的形式化定义工作,文[12]则是用实时动作逻辑 RAL 对 UML 进行的形式化定义;文[13,14]是对 UML 动态视图进行的形式化描述。

结论 本文研究了 UML 和 XYZ/ADL 之间的双向转换问题。经过这样的双向转换后,在描述系统的软件体系结构时,就可以直接利用 UML 的工具同时获得直观和精确的描述。我们已采用这种结合后的新方法描述了一个具体的实例系统,限于篇幅,将另文给出。

参 考 文 献

1 于卫,蔡希尧. 软件体系结构的描述方法研究. 计算机研究与发展, 2000, 37(10): 1185～1191
2 唐稚松,等. 时序逻辑程序设计与软件工程(上、下册). 北京:科学出版社, 2002
3 朱雪阳,唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL. 软件学报, 2003, 14(4): 714～720
4 张广泉. 软件体系结构与 XYZ 系统:[中国科学院软件研究所博士后研究工作报告]. 2002
5 陈琳琳,戎玫,张广泉. 体系结构描述语言 XYZ/ADL 到 UML 的映射. 计算机应用, 2006, 26(2): 468～471
6 朱雪阳. 软件体系结构形式描述研究:[中国科学院软件研究所博士学位论文]. 2005
7 Kamdé M M, Crettz V, Strohmeier A, et al. Bridging The gap between IEEE1471. an Architecture description 、language and UML. Software and Systems Modeling. Springer-Verlag, 2002 (1): 113～129
8 Robbins J E, Medvidovic N, Redmiles D F, et al. Integrating Architecture Description Languages with a Standard Design Method. http://www.ics.uci.edu/～redmiles/publications/C026-RMR+98.pdf
9 韦银星,张申生. 曹健. UML 类图的形式化及分析. 计算机工程与应用, 2002, 38(10): 5～7
10 周欣,魏生民. 基于 B 语言的 UML 形式化方法. 计算机工程, 2004, 30(12): 62～64
11 庞军,王云峰,郑国梁. 基于 COOZ 对 UML 的类结构的形式化. 计算机工程与应用, 2000, 36(6): 86～89
12 Lano K, Bicarregui J. Formalising the UML in Structured Temporal Theories. http://www.dcs.kcl.ac.uk/staff/kcl/ ecoop98.ps
13 崔萌,李宣东,郑国梁. UML 实时活动图的形式化分析. 计算机学报, 2004, 27(3): 339～346
14 蒋慧,林东,谢希仁. UML 状态机的形式语义. 软件学报, 2002, 13(12): 2244～2250