UML与软件体系结构描述语言之间的转换机制研究*)

张广泉[1] 戎 玫[2] 陈琳琳[1]

(苏州大学计算机科学与技术学院 苏州 215006)[1] (暨南大学深圳旅游学院 深圳 518053)[2]

摘 要 形式化描述和可视化描述是目前主要的两类软件体系结构描述方法,形式化描述以体系结构描述语言ADL为代表,可视化描述以统一建模语言UML为代表。目前软件体系结构描述领域的热点之一是研究这二者之间的结合,转换是其中一种重要的方式。基于此,本文对UML和基于时序逻辑的体系结构描述语言XYZ/ADL之间的转换问题进行了研究,定义了二者之间的转换规则。

关键词 软件体系结构描述,XYZ/ADL,UML,转换

Research on Conversion between UML and Software Architecture Description Language XYZ/ADL

ZHANG Guang-Quan[1] RONG Mei[2] CHEN Lin-Lin[1]

(School of Computer Science and Technology, Soochow University, Suzhou 215006)[1]

(Tourism College, Jinan University, Shenzhen 518053)[2]

Abstract The main methods of software architecture description are formal description and visual description, in which architecture description language ADL is typical of the former and UML is typical of the latter. Now the combination ofthem is a hotspot in, and conversion between the two is a most important method. So, the paper researches the conversion between UML and XYZ/ADL, an ADL based on temporal logic, defines the conversion.

Keywords Software architecture description, XYZ/ADL, UML, Conversion

1 引言

软件体系结构描述是软件体系结构重要的研究内容,也是研究和应用软件体系结构的基础。目前两类主要的软件体系结构描述方法分别是以体系结构描述语言ADL为代表的形式化描述方法,和以统一建模语言UML为代表的可视化描述方法[1]。ADL通常以某种形式化理论(如CSP、时序逻辑等)为基础,具有严格的语法和语义,能有效支持所描述系统的分析、求精和验证,但因其不够直观,所以目前在工业界还没有得到广泛应用。UML是一种语义丰富、通用、可视化的建模语言和事实上的工业标准,易于理解和交流。它提供丰富的视图从多个视角来描述系统的不同侧面,可有效运用于软件系统的建模、分析与设计。但是UML对软件体系结构如连接件等元素建模能力较弱,对体系结构的描述只能到达非形式化或半形式化的层次。

可以看出ADL形式化语义的精确性可以弥补UML语义精确性的不足,UML直观易懂的特点也正好可以弥补ADL这方面的不足,二者在描述体系结构方面具有很强的互补性,因此,UML和ADL结合问题的研究成为目前软件体系结构描述领域的热点。转换是研究二者结合的重要手段,基于此,本文对UML和基于时序逻辑的体系结构描述语言XYZ/ADL之间的转换问题进行了研究。

2 体系结构描述语言XYZ/ADL到UML的转换

体系结构描述语言XYZ/ADL[2]是对时序逻辑语言XYZ/E[3]的扩充,XYZ/E虽然可以描述软件体系结构[4],但它缺乏体系结构的相关概念。XYZ/ADL在此基础上扩充,不仅可以在统一的时序逻辑框架下描述从系统静态语义到实现之间不同抽象层次的规范,便于体系结构的逐步求精描述以及相关性质分析,还有一套XYZ/E的CASE工具集作为底层支撑,便于对软件体系结构进行描述、分析、求精、验证,直至最终系统的编译实现。

体系结构描述语言XYZ/ADL的主要设计元素有组件、连接件、端口、角色、性质、配置、体系结构、风格,以及粘接和绑定操作,它们之间的关系如图1所示。

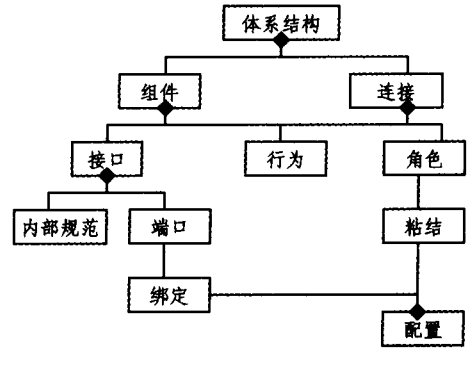


图1 XYZ/ADL设计元素间的关系

从XYZ/ADL到UML的转换可分为两个部分:

1)XYZ/E到UML的转换;

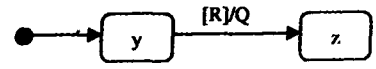
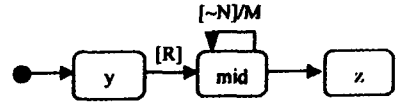
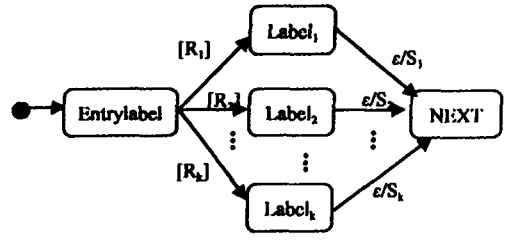
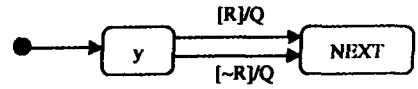
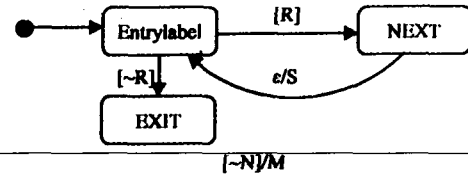
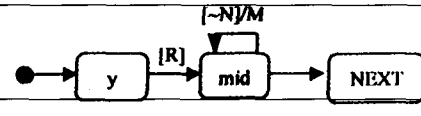
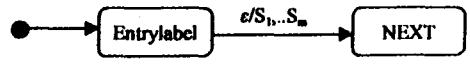
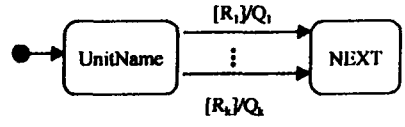
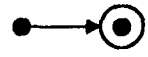
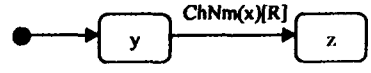
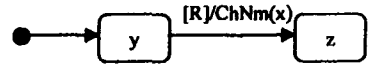
2)XYZ/ADL体系结构设计元素到UML的转换。

XYZ/ADL的语义基础是XYZ/E,XYZ/E是一种以线性时序逻辑为基础,基于状态转换的形式化语言。根据其特点,

*)基金项目:江苏省高校自然科学研究项目(05KJB520119);中国科学院计算机科学国家重点实验室开放课题(SYSKF0303)。张广泉 教授,博士,CCF高级会员,主要研究方向:软件工程、形式化方法。

将 XYZ/E 的各类基本结构及命令格式转换为 UML 状态机， 如表 1 所示。

表 1 XYZ/E 基本语句结构的映射

XYZ/E 基本结构		XYZ/E 语句描述	UML State Machine
BE		LB=y∧R⇒$O(Q∧LB=z)	
		LB=y∧R⇒M$U(N∧$OLB=z)	
SE	分情形	?[LB=EntryLabel∧R1⇒LB=Label1 |R2⇒LB=Label2 … |Rk⇒LB=Labelk; LB=Label1{|S1|}$OLB=EXIT; LB=Label2{\|S2\|}$OLB=EXIT; … LB=Labelk{|Sk|}$OLB=EXIT;	
	选择	LB=y∧R⇒$O(Q1∧LB=NEXT|Q2∧LB=NEXT)	
	循环	*[LB=Entrylabel∧R⇒ ($OLB=NEXT\|$O=EXIT); LB=Label{|S|}$OLB=Entrylabel]	
	等待	LB=y∧R⇒M$U(N∧$OLB=NEXT)	
	继续	LB=Entrylabel{|>>[S1; … Sm]|}$OLB=NEXT	
PE	选择元	UnitName;!![R1|>Q1,…Rk|>Qk]	
结束		LB=STOP	
通道输入		LB=y∧R⇒ChNm? x∧$OLB=z	
通道输出		LB=y∧R⇒ChNm! x∧$OLB=z	

XYZ/ADL 体系结构设计元素到 UML 的转换又分为两个步骤：

1)分析体系结构设计元素和 UML 建模元素的语义，找出语义相似的对应元素；

2)用 UML 扩展机制和对象约束语言 OCL 对 UML 中的相应元素进行约束。

表 2 列出了 XYZ/ADL 的体系结构设计元素和 UML 建模元素的转换关系。

用 UML 扩展机制和对象约束语言对 UML 建模元素具体的约束描述，可详见文[5]。

表 2 XYZ/ADL 设计元素与 UML 建模元素的转换关系

XYZ/ADL	UML 建模元素	扩展和约束后的元素
XYZ/E 基本结构	状态机	≪XStateMachine≫
体系结构	模型 Model	≪XArchitecture≫
组件	类 Class	≪Xcomponent≫
连接件	类 Class	≪Xconnector≫
端口	接口 Interface	≪Xport≫
角色	接口 Interface	≪XRole≫

3 UML 到体系结构描述语言 XYZ/ADL 的转换

UML 提供了丰富的视图和图表对系统进行建模。其动态建模元素中的顺序图、活动图到 XYZ/ADL 的转换，文[6]已经进行了研究，本文选取 UML 静态建模中的核心——类图进行研究。

对类图的转换也分为两个步骤：

1)对类图进行形式化定义；

2)对定义的类图元素进行相应的转换。

下面分别介绍这两个过程。

3.1 类图的形式化定义

定义 1 类图 cd={a| a ∈association}。association 是关系。

定义 2 关系 association 为七元组(ob1,ob2,m1,m2,r1,r2,asstype)，其中 ob1 和 ob2 表示关联的类名，m1 和 m2 表示 ob1 和 ob2 端的多重性，r1 和 r2 表示关联在 ob1 和 ob2 端的角色名，asstype 表示关联的类型。类的类型 type ∈{类，接口}，关联的类型 asstype ∈{一般，自身，限定，组成，聚合，实现，依赖，泛化}。

在有方向性的关联中，对 ob1 和 ob2 作如下规定：

- 如果是组成或聚合关联，则 ob1 表示整体元素，ob2 表示部分元素；
- 如果是实现关系，则 ob1 为类，ob2 为接口；
- 如果是依赖关系，则 ob1 表示需要依赖的元素，ob2 表示被依赖的元素；
- 如果是泛化关系，则 ob1 表示继承的元素，ob2 表示被继承的元素。

关联中大部分角色对于 XYZ/ADL 没有太大意义，所以在对应的关系中，r1 和 r2 用-代替。没有角色的关系中，r1 和 r2 亦用-代替。多重性是数或数的集合，本文以前者为例。

如图 2 所示的类图，可以定义为如下三个关系：

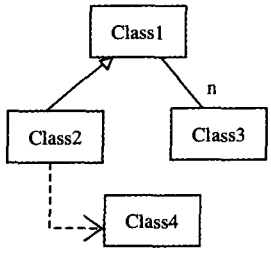


图 2 一个类图实例

1)(Class1,Class3,1,n,-,-,一般)

2)(Class2,Class1,1,1,-,-,泛化)

3)(Class2,Class4,1,1,-,-,依赖)

3.2 对类图元素 association 的转换

因为 UML 对体系结构的连接件建模能力较弱，所以一般 UML 类图中均没有对连接件显式建模，故类只能转换成 XYZ/ADL 中的组件 Component，相应的接口转换为端口 Port。

定义 3 从 association 到 XYZ/ADL 设计元素的转换定义为一个映射：

AtoX: association -> XADLelement

不同类型的关系，其映射关系如表 3 所示：

表 3 associaition 到 XYZ/ADL 的映射

asstype	对应的 XYZ/ADL
组成 聚合	% COMPONENT ob1==[% COMPOSITION==[o1,…,om2; ob2. type]]
实现	% COMPONENT ob1==[% PORT ob2==datatype; □[portbehavior;]]
依赖	% COMPONENT ob1==[% PORT pname==datatype; (% IOP o: ob2. type)□[portbehavior;]]
泛化	% COMPONENT ob1==[% PORT port1fromob2==datatype; □[portbehavior;] … % PORT portnfromob2==datatype; □[portbehavior;]] 其中，port1fromob2，…，portnfromob2 是 ob2 中的端口。
自身	% COMPONENT ob1==[…] where [∀o: Instance (ob1)・o ∈ o. r1 ∧ count (o. r2)>=0 ∧ count (o. r2)<=m2 ∨o ∈ o. r2 ∧ count (o. r1)>=0 ∧ count (o. r1)<=m1]
限定	% COMPONENT ob1==[…] where [(∀o1: Instance (ob1)・∃ set1={o| o ∈o1. r2 ∧ o. qualifier=o1. qualifier} ∧ count (set1)>=0 ∧ count (set1)<=m2) ∧ (∀o2: Instance (ob2)・∃set2={o| o ∈ o2. r1 ∧ o. qualifier=o2. qualifier} ∧ count (set2)>=0 ∧ count (set2)<=m1)] qualifier 为限定词。
一般	% system systemname==[…] where [∀o1: Instance (ob1)・count (o1. r2)>=0 ∧ count (o1. r2)<=m2 ∧ ∀o2: Instance (ob2)・count (o2. r1)>=0 ∧ count (o2. r1)<=m1] 其中角色 r1 和 r2 转换为对象的操作，获得相应角色的对象实例。Instance 和 count 是系统操作，获得对象类的实例和计算元素的个数。

如图 2 中的关系 2)(Class2,Class1,1,1,-,-,泛化)可映射为

% COMPONENT Class2==[

% PORT port1fromClass1==datatype; □[portbehavior;]

…

% PORT portnfromClass1==datatype; □[portbehavior;]

]

关系 3)(Class2,Class4,1,1,-,-,依赖)可映射为

% COMPONENT Class2==[

% PORT pname==datatype; (% IOP o: Class4. type) □[portbehavior;]

]

而这两个关系转换后可以合并为

(下转第 269 页)

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