UML2.0 类图的一种形式化描述方法*)

杨敬中[1] 张广泉[1,2] 戎 玫[3]
(苏州大学计算机科学与技术学院 苏州 215006)[1] (重庆师范大学数学与计算机科学学院 重庆 400047)[2]
(暨南大学深圳旅游学院 深圳 518053)[3]

摘 要 UML 类图是根据系统中的类,以及各个类之间的关系来描述系统的静态视图。基于 UML 缺乏精确语义描述的不足,我们提出了基于时序逻辑语言 XYZ/E 来表示类图形式化语义的方法。通过对 UML2.0 类图元素及其特点的分析,找出类图元素的形式化描述规则,利用转换法实现了对 UML2.0 类图的 XYZ/E 形式化描述。

关键词 UML2.0 类图,形式化描述,XYZ/E

A Kind of Formal Method to Describe UML2.0 Class Diagram

YANG Jing-Zhong[1] ZHANG Guang-Quan[1,2] RONG Mei[3]
(School of Computer Science and Technology, Suzhou University, Suzhou 215006)[1] (School of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047)[2] (Shenzhen Tourism College, Jinan University, Shenzhen 518053)[3]

Abstract The UML Class Diagram which describes the static view shows the classes of the system and the relationships among the classes. Because UML lacks of precise semantics, a method is proposed to express the formal semantics of the class diagrams based on the temporal logic language XYZ/E. By analyzing the elements of class diagrams and their characteristics, and finding the formal regulation, transition method is taken to realize the formal description of UML2.0 Class Diagrams.

Keywords UML2.0 class diagram, Formal description, XYZ/E

1 引言

统一建模语言 UML 是一种将软件开发过程中出现的各种模型用可视化图形来描述的语言。它融合了多种面向对象开发方法的优点,在工业界已经得到广泛应用,成为事实上的工业标准。UML 具有简单、直观明了的特点,但不足之处在于其语义不够精确,难以实现逐步求精的开发过程,难以对其所表示的模型进行分析和验证,不同的开发人员对同一个模型的理解也可能出现不一致。

形式化方法是一种基于数学理论来描述系统性质的技术,有穷自动机、线性时序逻辑等都是其数学基础。这种技术能够有效地刻画和验证软件系统的性质,其目的在于开发可靠的软件产品。形式化方法的一个重要部分是形式化描述,所谓形式化描述就是用形式化的语言(具有严格的语法语义定义的语言)做描述。对系统采用形式化描述可以避免二义性、提高一致性,便于系统的细化和验证,有助于提高软件的可靠性。

尽管 UML 是一种通用的建模语言,但由于其对系统描述不能够提供精确的语义,因而十分有必要对 UML 的语义进行形式化描述。形式化方法能够帮助我们发现 UML 描述系统时所出现的语义不明确或不完整,有助于提高软件开发人员对系统理解的一致性。XYZ/E 是一种以时序逻辑为基础的形式化程序设计语言,它提供了一套形式化规范描述软件系统的规则。XYZ/E 中的包块可以很自然地表达类的概念,所以我们在对 UML2.0 类图形式化描述时,可以采用 XYZ/E 来表示其形式化语义。

2 现有 UML 形式化思路

如何对 UML 进行形式化存在两种思路,一种就是对 UML 核心语法进行形式化,使得 UML 符合形式化规范的语言。它是在元模型层次上进行的,在此基础上建立的 UML 模型具有可靠的数学基础,便于细化和验证,但这种思路实现的难度较大。英国的 The precise UML group 就在致力于这方面的工作,而且已经对 UML 2.0 做出了相关贡献[1]。另一种思路是转换法,就是利用形式化语言在不丢失或者少丢失信息的前提下对 UML 形式化,它是对每一种 UML 图形赋予形式化语义。XYZ/E 语言是一个面向软件开发全过程的时序逻辑语言,它既能表示软件系统的规范与性质,又能表示软件系统的数学模型和程序实现。文[2,3]采用 XYZ/E 很好地描述了 UML 中活动图、状态图以及顺序图的形式化语义。文[4]实现了体系结构描述语言 XYZ/ADL(是对 XYZ/E 的扩充)到 UML 的映射。基于此,本文采用 XYZ/E 形式化程序设计语言按照转换法思路以实现 UML2.0 类图的形式化描述。

3 时序逻辑语言 XYZ/E

XYZ/E[5]是一个系列化语言族,是在直言式一阶逻辑的基础上,利用时态算子表示状态转换机制。它既能表示适应

*)基金项目:江苏省高校自然科学研究项目(批准号:05KJB520119);重庆市自然科学基金项目(编号:CSTC,2006BB2259);重庆市教委科学技术研究项目(合同号:040803)。杨敬中 硕士研究生,主要研究方向为软件体系结构与 UML;张广泉 教授,博士,CCF 高级会员,主要研究方向为软件工程与形式化方法;戎 玫 副教授,博士,主要研究方向为软件工程与电子商务。

冯·诺依曼机器体系状态转换机制的命令语言，又能表示逻辑推理特征的直言式公式。而本文则利用 XYZ/E 来描述 UML 类图的形式化语义。下面我们介绍一下 XYZ/E 的基本概念、本文所应用到的包块的概念以及相关标记。

3.1 XYZ/E 基本概念

下面列出 XYZ/E 的两个基本概念：

(1)XYZ/E 中讨论的对象主要是具有以下两种形式的条件元时序逻辑公式的集合。

$(*)LB=l_i \wedge R \Rightarrow \$O(v_1,\cdots,v_k)=(e_1,\cdots,e_k) \wedge \$O\,LB=l_m$

$(**)LB=l_i \wedge R \Rightarrow \Diamond(Q \wedge LB=l_m)$

这里 l_i 和 l_m 是标号常量，$\Rightarrow$ 代表联结词 $\rightarrow$，$e_1,\cdots,e_k$ 是不含时序算子 $\$O$ 的项。LB 是控制变量，它用来指示程序的当前标号。R 和 Q 是不含时序算子及量词的谓词公式，分别称为条件元的条件部分和动作部分。条件元(*)主要用于表示可执行的 XYZ/E 程序，条件元(**)主要用于表示程序的抽象描述。

(2)在 XYZ/E 中表示算法的语言成分为如下所示的单元：

$\square[A_1;A_2;\cdots;A_n]$ WHERE $B_1 \wedge B_2 \cdots \wedge B_m$

其中 $A_1,\cdots,A_n$ 是条件元，$B_1,\cdots,B_m$ 是单元的约束部分。符号";"表示条件元之间的合取关系。

3.2 包块

XYZ/E 语言引入了包块的概念。所谓包块是指由一组运算围绕一个数据结构封装而成的模块。包块本质上是静态的，它通过移入与移出说明来实现包块之间的信息传递。包块的主要特征在于信息封装，它所包含的各种成分的信息可以分成两类：(1)包块中定义的数据及在这些数据上定义的运算，对用户来说是可见的。(2)运行的细节往往对用户掩盖。在包块的嵌套层次结构中，要求子包块所定义的数据结构是父包块的数据结构的子结构，父包块与子包块之间具有继承性。包块的结构如下所示：

Package::=%PACK [PackageNm:]
[ImportDeclPart;]
[ExportDeclPart;]
[FatherNamePart;]
[TypeDeclPart;]
[PackageDeclPart;]
SharedVarDeclPart;
OperationDeclPart]
[WherePart]
OperationDeclPart::=ProDeclPart
PackageDeclPart::=PackageDecl{;PackageDecl}
FatherNamePart::=%FATHER NameList;

包块说明结构中的 WherePart 仅出现在规范语言或者抽象描述语言中。

3.3 相关标记

在 XYZ/E 中 ImportDeclPart、ExportDeclPart、TypeDeclPart、SharedVarDeclPart 和 ProcDeclPart 这些部分外层方括号外的标记依次分别为"%IMP"、"%EXP"、"%TYPE"、"%VAR"和"%PROC"，而 ProBody 则根据其控制结构(即 XYZ/BE,XYZ/SE 及 XYZ/PE 的不同控制结构)表示形式不同，分别以"%ALG"，"%STM"或者"XYZ/PE"作为标记。输入参量、输出参量、输入输出参量的标记分别为"%INP"、"%OUTP"、"%IOP"。

4 UML2.0 类图元素的形式化

UML2.0[6,7]中的类图是用多个类，以及这些类之间的关系描述系统的一种图示。类图是一种静态模型，是从静态角度表示系统的，是构建其他图(如状态图、协作图等)的基础。在一个类图中，类与类之间的关系通常有关联、泛化、依赖等。下面我们将根据类图元素的不同特点，分别给出它们各自形式化描述的转换规则。

4.1 类的形式化描述

表 1 XYZ/E 包块及其相应类的语义关系

XYZ/E 语句	相应的包块语义	相应的类语义
ImportDeclPart;	从外界移入信息如：变量、类型、操作等	类引用的属性、操作
ExportDeclPart;	外界可用的信息如：变量、类型、操作等	类可以被引用的属性、操作
FatherNamePart;	父包块名称	父类名称
TypeDeclPart;	新的类型及其名字	新的类型及其名字
OperationDeclPart;	包块的操作	类的操作
SharedVarDeclPart;	包块内的过程、进程、与程序体所允许使用的公用变量名字及类型。称为共享变量。	类中定义的属性

类在概念上是一种抽象机制，它是对一类对象的存储和操作特性的抽象。在 UML2.0 中，类有名称、属性和操作。属性是用来描述该类的对象所具有的特征，操作则说明了该类所能做的工作，类将数据和对数据进行处理的操作封装起来，形成一个完整的整体。而根据上面我们所给出的包块概念，能够很容易看出 XYZ/E 中的包块可以用来表示类的概念。所以我们采用 XYZ/E 中的包块来对 UML2.0 中的类进行形式化描述，而对于对象的处理[8]，XYZ/E 中引入了代理机构(Agent)，实际上由包块和与之相匹配的进程组成，这与对象是类实例化的关系是一致的。在表 1 中，具体给出了 XYZ/E 语句、包块、类之间的语义对应关系。

4.2 泛化的形式化描述

泛化是两个类之间的继承关系。所谓继承是一种机制，一个类的所有信息(属性和操作)能被另一个类继承，继承某个类的类中不仅可以有属于自己的信息，而且还拥有了被继承类中的信息。对于泛化关系，在 XYZ/E 中，父类和子类被定义为父包块和子包块，而在子包块的父包块名列表中，即在 FatherNamePart::=%FATHER NameList 语句中只要注明父包块的名称就可以表示它们之间的继承关系了。如果一个类是顶层的父类(即该类没有父类)，或者这个类和其他所有的类之间不存在这种继承关系，那么该类在被定义成的包块后，它的 FatherNamePart 语句部分就为空。如果父包块名列表(FATHER NameList)中父包块的名字不是唯一的，则表示这是多重继承关系。

4.3 关联的形式化描述

关联是类之间的一个连接，也是涉及此关联的那些类的对象之间的语义连接。如果一个类与另一个类关联，并且关联双方相互知道对方，那么这是双向关联。如果类与类之间的关联是单向的，则称为单向关联。在用 XYZ/E 对关联形式化描述时，只要在类所定义成的包块的 SharedVarDeclPart 语句中将对方定义为成员变量就可以了。如果是双向关联，则互相将对方定义为自己的成员变量；如果是单向关联则只要将另一方定义为自己的成员变量就可以了。如图 1 所示的单向关联：A 类是判断给定平面上的 4 个顶点是矩形还是菱形，并计算出各条边的长度。B 类是判断给定的 4 个顶点是否是正方形，并计算出它的面积。这里 A 类、B 类设计为单向关联，B 类中有一个 A 类的成员变量。在用 XYZ/E 描述时，先将 A 类、B 类定义成 A 和 B 两个包块，然后在 B 包块的 SharedVarDeclPart 语句中，将 A 包块定义为 B 包块的成员变量。在这里只给出 B 类的详细描述。

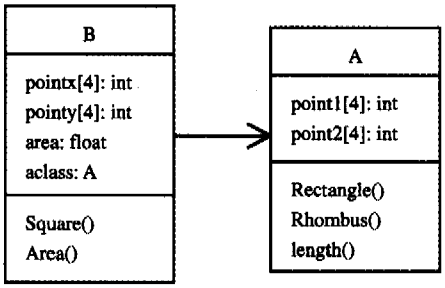

图 1 单向关联

```
B类的 XYZ/E 具体描述如下：
%PACK:[ B:
%TYPE[arr=ARRAY(int ; 4)];
%VAR[pointx, pointy: arr;
       area: float;
       aclass: A ];
%PROC[
  Square()==[
  %ALG[
    LB=START_ Square ⇒ $OLB=l1;
    LB=l1⇒aclass. Rectangle();
    LB=l2⇒aclass. Rhombus();
    LB=l3⇒ $OLB=RETURN
    ]]
  Area()==[
%LOC[len: float;
       a,b,c,d: int];
%ALG[
  LB=START_ Area⇒ $OLB=l1
  LB=l1⇒ $Oa=pointx[0]∧ $Ob=pointx[1]∧LB=l2
  LB=l2⇒ $Oc=pointy[0]∧ $Od=pointy[1]∧LB=l3
  LB=l3⇒aclass. length(%IOP a| a; %IOP b| b; %IOP c| c; %
  IOP d|d; %IOP lent|len)
  LB=l4∧ $Oarea=len * len∧LB=l5
  LB=l5⇒ $OLB=RETURN ] ]
   ]
]
```

4.4 依赖关系的形式化描述

在 UML2.0 中还提供了依赖关系，这种关系显示一个模型元素需要另一个元素来达到某种目的。在一个依赖关系中，一个类引用另一个类，引用类的改变可能会对使用类产生影响，离开了引用类，使用类的语义将不再完整。依赖关系是单向的。在用 XYZ/E 处理依赖关系时，在被依赖包块的 ExportDeclPart 语句中，列出它能被别的包块所引用的属性和操作，而在依赖包块的 IMP 语句中，则列出引用哪些属性和操作供本包块使用。在 ImportDeclPart 语句中列出以后，就可以在该包块内像使用本包块所定义的属性和操作一样使用它们。在这里，从对依赖和关联形式化描述所采取的处理方式的不同可以看出，依赖与关联是有区别的，它们的不同点在于依赖是对被依赖方属性和操作的引用，而关联则是将对方作为自己的成员变量。如图 2 所示的依赖关系：假设类 A、B 之间的依赖关系是 B 类依赖 A 类，其中在 B 类的操作中引用到 A 类的属性 width 和 length、操作 IsParallel。在 XYZ/E 描述时，先将 A 类、B 类定义为 A 包块和 B 包块。如表 2 所示：B 对 A 引用的具体情况是，A 包块的 ExportDeclPart 的语句和 B 包块的 ImportDeclPart 语句，分别表示 A 包块的移出的属性和操作，以及 B 包块从 A 包块中移入的属性和操作。在 B 包块中，对来自 A 包块中的属性 width、length 及操作 IsParallel()的使用就像在本包块中所定义的属性和操作一样使用。

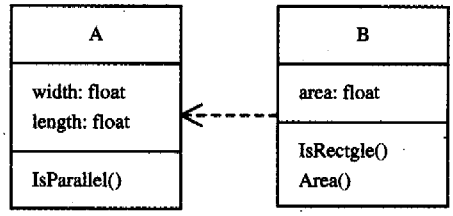

图 2 依赖关系

表 2 依赖关系处理方法

A 包块的 ExportDeclPart 语句	B 包块中的 ImportDeclPart 语句
%PACK:[A: %EXPORT[%VAR width; length; %PROC IsParallel]; .....]	%PACK:[B: %IMP[%VAR width from A; length from A; %PROC IsParallel from A]; .....]
A 包块中的属性 width、length 及操作 IsParallel()可以被 B 类所引用	B 包块将引用 A 类的属性 width、length 及操作 IsParallel()

4.5 聚合的形式化描述

聚合(Aggregation)只是关联的一种特殊情况。这种聚合关联表示类之间的关系具有整体与部分的特点，所以用 XYZ/E 对聚合进行形式化描述时，可以比照前面对关联的处理方式，首先是把类定义为 XYZ/E 的包块，然后在具有整体特点的包块的 SharedVarDeclPart 语句中，将具有部分特点的包块定义为它的一个成员变量。

4.6 接口实现的形式化描述

在 UML2.0 中，接口不能包括那些真正的对象，而是仅包含一些抽象的操作。因此，一个接口有多个操作签名，它们一起指定了一个行为，而任意元素通过实现该接口就能够支持该行为。当一个接口在某个类中实现时，则它与该类之间是关联关系，而使用该接口的类可以通过一个依赖关系与该接口相连接。依赖类仅依赖指定接口中的那些操作，而不依赖接口实现类中的其他部分，依赖类可以调用在接口中已声明的操作。因此，对于接口实现(InterfaceRealization)的形式化描述，可以采取类似依赖关系形式化描述所采取的处理方式。对于某个类要实现其接口，先把类定义为包块，然后只要在其包块的 ExportDeclPart 语句中列出一些操作，供别的包

(下转第 288 页)

P_1、P_2 上执行。假定 P_0、P_1、P_2 的故障发生率和恢复率分别为$[\lambda_0, \mu_0]=[0.001, 10]$,$[\lambda_1, \mu_1]=[0.0012, 8]$,$[\lambda_2, \mu_2]=[0.0015, 8]$。

按照上述推导经编程计算，容易求解其结果：对于处理机 P_0，$p_0(r_1, r_4, r_5)$在$(r_1, r_4, r_5)=(4,8,5)$时取得最小值，这意味着当任务 n_1，n_4，n_5 分别设置 4，8，5 个检查点时，处理机 P_0 的失效率最低；同样，对于处理机 P_1，$p_1(r_1, r_2)$在$(r_1, r_2)=(5, 6)$时取得最小值；对于处理机 P_2，$p_2(r_1, r_3)$在$(r_1, r_3)=(3, 8)$时取得最小值。

由于处理机 P_1、P_2 的失效率 $p_1(r_1, r_2)$和 $p_2(r_1, r_3)$分别随(r_1, r_2)和(r_1, r_3)的变化是三维的，图 6 和图 7 给出了其曲面图。为便于观察，图中采用的是概率 $q_1(r_1, r_2)$和 $q_2(r_1, r_3)$的曲面图$(1\leqslant r_1, r_2, r_3\leqslant 15)$。由等式(12)可知，当概率 $q_1(r_1, r_2)$和 $q_2(r_1, r_3)$分别取最大值时，失效率 $p_1(r_1, r_2)$和 $p_2(r_1, r_3)$为最低。

结论 本文针对实时分布系统中的 Out-Tree 任务图，提出了一种启发式的 Out-Tree 任务图调度算法，并通过符合实际地简化其故障模型，开发了一种多处理机上相关多任务的最优检查点策略。该调度算法能够保证任务的调度长度最小，所需处理器数目尽量少，没有处理机间通信开销；该检查点策略没有检查点全局一致性开销，可保证各处理机的失效率最低。

参 考 文 献

1 Ghosh S, Melhem R, Mosse D. Fault-tolerant rate-monotonic scheduling. Journal of Real-Time System, 1998, 15(2): 149～181
2 Ghosh S, Melhem R, Mosse D. Enhancing real-time schedules to tolerate transient faults. In: Proceedings of the 16th IEEE Real-Time Systems Symposium, 1995. 120～129
3 Pedro M A, Mosse D. A responsiveness approach for scheduling fault recovery in real-time systems. In: Fifth IEEE Real-Time Technology and Applications Symposium, 1998. 4～13
4 Pedro M A, Aydin H, Mosse D, et al. Scheduling optional computations in fault-tolerant real-time systems. In: Proceedings of the 8th International Conference on Real-Time Computing Systems and Applications, 2000. 323～331
5 Seong W K. Reliability analysis and design of real-time fault tolerant control systems under transient faults: [Ph. D thesis]. Korea Advanced Institute of Science and Technology, 2000
6 Krishna C M, Singh A D. Optimal configuration of redundant real-time systems in the face of correlated failure. IEEE Trans on Reliability, 1995, 44(4): 587～594
7 Krishina C M, Shin K G. Real-Time Systems. New York: McGraw Hill, 1997
8 Daniel P S, Robert S S. Reliable Computer Systems: design and evaluation. 2nd ed. Burlington, Digital Press, 1992
9 Geist R, Reynolds R, Westall J. Selection of a checkpoint interval in a critical-task environment. IEEE Trans on Reliability, 1988, 37(4): 395～400
10 Shin K G, Lin T H, Lee Y H. Optimal checkpointing of real-time tasks. IEEE Trans on Computers, 1987, 36(11): 1328～1341
11 Krishna C M, Singh A D. Reliability of checkpointed real-time systems using time redundancy. IEEE Trans on Reliability, 1993, 42(3): 427～435
12 Seong W K, Byung J C, Byung K K. Checkpointing strategy for multiple real-time tasks. In: Proceedings of the Seventh International Conference on Real-time Systems and Applications, 2000. 517～522
13 Park CI, Choe T Y. An optimal scheduling algorithm based on task duplication: [Technical Report]. Dept of CSE, POSTECH, August 2000
14 Ruan Y, Liu G, Li Q, et al. An efficient scheduling algorithm for dependent tasks. In: Proceedings of the Fourth International Conference on Computer and Information Technology, 2004
15 Zhang H, Hu M, Fang B X, et al. A sub-optimal algorithm on allocating a single task cluster on NOWs. Journal of Computer Research and Development, 1999, 36(9): 1076～1079
16 Colin J Y, Chritienne P. C. p. m. scheduling with small communication delays and task duplication. Operations Research, 1991, 39(4): 680～684
17 Darbha S, Agrawal D P. Optimal scheduling algorithm for distributed- memory machines. IEEE Trans on Parallel and Distributed systems, 1998, 9(1): 87～95
18 Liu Z Y, Fang B X, Zhang Y, et al. A scheduling algorithm for an Out-Tree DAG. In: Proceedings of the Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region, 2000
19 Ziv A, Bruck J. An on-line algorithm for checkpoint placement. IEEE Trans. on Computers, 1997, 46(9): 976～985

(上接第 279 页)

块引用就可以了，而依赖类则在 ImportDeclPart 语句中将这些操作引入。这与依赖关系处理的区别在于 ExportDeclPart 语句中只列出一些操作而不会列出任何属性。

5 相关工作比较

目前在对 UML 类图的形式化描述方面，国内外已做出了相关工作[9～11]，这些工作大多是基于 Z 的静态描述语言如 B、COOZ 等进行形式化定义的，但它们都存在一些不足之处：一方面 Z 语言对大型系统的模块化能力不足，另一方面尽管这些静态语言可以描述系统的组成等结构性特征，但对于系统的动态行为特征描述显得力不从心。因此，若对 UML 类图采用这些静态语言来描述，则会在实用性上受到限制。而时序逻辑语言 XYZ/E 则强调静态语义与动态语义并重，它是一种广谱语言，这种语言同时具有多种级别语言的特点，既可以用来描述规格说明，也可以用来编写可运行的程序代码，而且支持混合语言。因此我们选择 XYZ/E 来对 UML2.0 中的类图进行形式化描述，相对来说具有较好的实用性。

结束语 本文在对 UML2.0 类图进行形式化描述的过程中，采用了 XYZ/E 来表示类图元素的形式化语义，并且给出了一套其形式化描述的转换规则。有了这样的一组规则，就可以在对一个 UML2.0 类图结构进行描述时，根据我们所给定的转换规则，很容易得到其精确的形式化描述。面向方面编程(AOP)是对 OOP 的继承和发展，AOP 将系统分解成核心关注点和横切关注点，对于核心关注点使用传统的面向对象机制，对于横切关注点则使用 Aspect 机制。这就要求 UML 不仅能表示出系统中的类，也要能表示其中的 aspects。我们下一步的工作是，扩展 UML2.0 以表示出系统中的横切关注点，同时为了能够得到精确的形式化描述，也需要进行 XYZ/E 对面向方面的扩充工作。

参 考 文 献

1 The precise UML group. http://www.cs.york.ac.uk/puml/uml2-0
2 朱雪阳. 软件体系结构形式描述研究:[博士学位论文]. 北京：中国科学院软件研究所，2005.2
3 黄正宝，张广泉. UML2.0 顺序图的 XYZ/E 时序逻辑语义研究. 计算机科学，2006，33(8)：249～251
4 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
5 唐稚松，等. 时序逻辑程序设计与软件工程. 北京：科学出版社，2002
6 Object Management Group. UML 2.0 Superstructure Specification: Final Adopted Specification. http://www.omg.org/docs/ptc/04-10-02.pdf
7 Hans-Erik Eriksson 等. UML 2 工具箱. 北京：电子工业出版社，2004
8 郭亮，唐稚松. XYZ/E 面向对象程序语义概述. 软件学报，2003，14(3)：356～361
9 Kim S K, Carrington D. A Formal Mapping between UML Models and Object-Z Specifications. ZB 2000, LNCS 1878, Berlin Heidelberg: Springer-Verlag, 2000. 2～21
10 Ramalho F, Robin J. Mapping UML Class Diagrams to Object-Oriented Logic Programs for Formal Model-Driven Development. http://www.metamodel.com/wisme-2004/ accept/2.pdf
11 周瑾，马应龙，李巍，吴志林. UML 的形式化及其应用. 计算机科学，2005，32(3)：136～140