DOI:10.11721/cqnuj20130115

基于 OOZS 的 UML 类图形式化描述及其应用*

肖 岗
(重庆师范大学 研究生处,重庆 401331)

摘要:统一建模语言 UML 是一种面向对象分析和设计过程中重要的建模工具。但由于 UML 缺乏精确的形式化语义,不利于对其所描述的需求进行进一步分析和验证。这一点上,形式化方法可与之互补。基于此,本文采用一种面向对象的、基于 Z 的扩展语言 OOZS——结构化面向对象形式规格说明语言,对 UML 的类图进行了形式化描述,寻求一种在软件设计与系统建模过程中 UML 到 OOZS 的映射与转换机制,最后给出一个基于 OOZS 的 UML 类图的形式化描述实例,结果表明本文的研究工作在实践中是可行的。

关键词:面向对象方法;形式化描述;OOZS;UML;类图
中图分类号:TP311.5 **文献标志码**:A **文章编号**:1672-6693(2013)01-0067-06

1 研究背景

软件系统模型的描述目前可以分为两大类:非形式化方法和形式化方法。非形式化方法一般都是用图表来描述系统模型,如数据流图、UML 等,它们用各种图形来表示系统模型的具体含义,主要优点是其易用性、直观性、交流性强。UML 可对软件开发过程中各个阶段的需求进行可视化图形建模,是目前比较流行的软件系统模型描述方法,在工业界得到广泛应用。然而,由于复杂系统的建模往往需要严格的语义分析,而 UML 缺乏精确的语义,多视角建模视图之间也存在不一致,这使得对模型难以进行一致性检查和正确性分析,进而限制了它的有效性,所以 UML 有必要在形式化方面进行拓展[1]。

形式化方法(Formal method)本质是基于数学的方法(如集合论、一阶谓词演算、线性时序逻辑等)来描述目标软件或硬件系统属性的一种特殊技术。将形式化方法用于计算机科学领域中,是期望能够像其他工程学科一样,使用适当的数学分析以提高设计的可靠性和可信性,并能使系统具有良好的结构,使其易维护,关键是能较好地满足用户需求。形式化方法最主要的优点是具有精确性、可以验证,并且便于机器支撑和自动处理等。然而,形式化方法并没有在工业界得到广泛应用,主要原因之一是构造形式规格说明比较困难,困难主要在于难以发现有用的抽象[2];其次是采用形式化描述的规格说明不如图形化建模直观,难以被软件开发人员所理解,更不利于系统相关人员的交流。

综上,UML 与形式化方法的优势与不足是非常明显的,同时也是互补的。将二者有机结合,既可以保留 UML 的直观性,又具有形式化的精确语义。目前,国内外已有很多研究将现有的形式化技术(如 Z[3-4],Object-Z[5],XYZ/E[6])应用于 UML,为其提供准确的语义,倍受关注。

2 结构化面向对象形式规格说明语言 OOZS

OOZS 是在形式化描述语言 Z 的基础上,基于"由规格说明建立一个快速原型"的思想而形成的一种面向对象的形式规格说明语言[7-9]。

OOZS 具有表达力强、规格说明简明直观、结构清晰、支持规格说明重用等特点。基于此,本文采用 OOZS 作为工具,建立一种从 UML 模型到 OOZS 规格说明的映射。这样,可以很方便地将需求工程中建立的各种非形式化或半形式化模型转化为具有精确语义的 OOZS 规格说明,在此基础之上,不仅可以对需求模型进行精确描述,还可以进行推理和验证,也可以进一步地从需求模型自动产生测试用例。

类是面向对象模型的最基本模型元素,类图技术是面向对象方法的核心技术。UML 中的类图是由多个类以及类之间的关系组成的图示,用以描述系统的静态模型,类之间常用的关系主要有关联、泛化和依赖等。本文先使用 OOZS 对 UML 类图进行形式化描述,再给出类图的 OOZS 形式化语义,从而完成 UML

* **收稿日期**:2012-10-25 **网络出版时间**:2013-01-18 15:05
资助项目:重庆市教委科学技术研究项目(No. KJ110615)
作者简介:肖岗,男,助理研究员,硕士,研究方向为软件工程与形式化方法,E-mail:xg@cqnu.edu.cn
网络出版地址:http://www.cnki.net/kcms/detail/50.1165.N.20130118.1505.201301.67_014.html

类图到OOZS的形式化语义的转换。OOZS的规格说明详细语法描述见文献[9]。

3 UML类图的OOZS描述

3.1 UML类的OOZS描述

1)类。在UML中,把具有相同属性、操作和关系的对象集合称为类。类名、属性和方法是类的3个基本元素。其中,类的属性有名称、可见性、类型和多重性,其中可见性可以是公共的、保护的,也可是私有的,定义可见性为

VisibilityKind::=public|protected|private

用OOZS定义类的属性如下:

UMLAttribute::=**state**

name:Name;

type:Type;

[Visibility:VisibilityKind;]

[multiplicity:**P** N;]

[initialValue:Expression;]

end state

在属性的定义中,名称和类型都是必须的,其余的可选。

操作具有名称、可见性和参数,其中参数应有参数名和类型,且参数名具有唯一性。用OOZS定义参数和操作如下:

UMLParameter::=**state**

name:Name;

type:Type;

[defaultValue:Expression;]

end state

UMLOperation::=**schema**

name:Name;

[(inParameters:**iseq** UMLParameter);]

[==>(outParameters:**iseq** UMLParameter);]

[visibility:VisibilityKind;]

[**pre** preCondition:PredExp;]

[**post** postCondition:PredExp;]

where

forall p1,p2:parameter@p1.name=

p2.name=>p1=p2;

end schema

完成属性、操作及参数的定义后,可以将类定义如下:

UMLClass::=**class**

name:Name;

[**init** attributes:**F**1 UMLAttribute;]

[**methods** operations:**F**1 UMLOperation;]

where

forall a1,a2:UMLAttribute@a1.name=

a2.name=>a1=a2

forall op1,op2:UMLOperation@

(op1.name=op2.name/\

#op1.parameter=#op2.parameter/**forall**:1..#op1.parameter@

op1.parameter(i).name=op2.parameter(i).name/\

op1.parameter(i).type=op2.parameter(i).type)=>op1=op2

end class

在类的定义中,where谓词部分表示在类中,属性的名称必须唯一;操作的名称、参数的名称都必须具有唯一性。

2)关联。在UML中,类与类之间的联结用关联(association)来表示。一般情况下,类图中两个类之间的关系是二元的,聚合、组合只是关联的特殊情况,可以用二元关联来表示,故本文只考虑二元关联,定义关联类型如下:

AggregationKind::=none|aggregate|composite

二元关联有两个关联端点,每个端点有角色名和多重性,但多重性不能为0,最多为1。用OOZS将关联端点定义如下。

UMLAssociationEnd::=**state**

name:Name;

aggregation:AggregationKind;

visibility:VisibilityKind;

multiplicity:**P** N;

[navigability:**Bool**];{% **Bool**::=true|false%}

where

multiplicity/={0}

aggregation=composite=>multiplicity={0,1}\/multiplicity={1}

end state

设e1,e2分别为一个二元关联的两个端点名称,则e1不能与e2相同,且e1或e2不能和另一端点的类中包含的属性名称相同;设e1为聚合或组合的聚合端或组合端,则另一端的聚合值应为none。因此,用OOZS定义关联如下:

UMLAssociation::=**state**

name:Name;

e1,e2:UMLAssociationEnd;

where

e1.name/=e2.name

e1.aggregation **in**

{aggregate,composite}=>e2.aggregation=none

e1. name **notin** (a: e2. attachedClass. attriutes. name}

e2. name **notin** (a: e1. attachedClass. attriutes. name}

forall a1,a2:UMLAssociation|a1/=a2@

{a1. e1. attachedClass,a1. e2. attachedClass}=

{a2. e1. attachedClass, a2. e2. attachedClass} =>a1. name/=a2. name

end state

关联继承了类和关联,联系端的聚合值为 none 且角色与属性不应重名。因此,关联类的描述如下:

UMLAssociationClass::=**class**

superclass[UMLClass,UMLAssociation];

ClassName:Name;

[**owns** attributes:**F**1 UMLAttribute;]

[**methods** operations:**F**1 UMLOperation;]

where

e1. aggregation=**none**/**e**2. aggregation =none

self **notin**{e1. attachedClass,e2. attachedClass}

{a:attribute@a. name}\{e1. name,e2. name}={}

end class

3)泛化。泛化(Generalization)描述了类与类之间的继承关系,也是对象之间耦合度最大的一种关系,使子类可以继承父类的所有结构和行为,但不允许循环继承。

UMLGeneralization::=**state**

parent:UMLClass;

child:UMLClass;

where

forall g:UMLGeneralization@

(g. parent,g. child)\id(UMLClass)={}

end state

4)依赖的 OOZS 描述。在 UML 中,依赖(Dependency)是对象之间的一种临时性关系,只有一个类调用被依赖类中的某些方法时才产生依赖关系,即依赖关系是单向的。UML 中定义的依赖关系构造型比较多,但只有其中少数几个常用到类图中的依赖关系上,将其定义如下:

DependencyKind::=Bind|Derive|Friend|Instanceof

|Instantiate| Powertype|Refine|Use

UMLDependency::=**state**

source:UMLClass;

target:UMLClass;

[depConstraint:DependencyKind;]

end state

5) 类图的 OOZS 描述。UML 的类图是由类、关联、泛化和依赖组成的。在下面的 OOZS 描述中,谓词约束部分表示了类的名称是唯一的,并且关联、泛化和依赖所涉及到的类应该在同一个类图中。

UMLClassDiagram::=**state**

class:**F**1 UMLClass;

association:**F** UMLAssociation;

generalization:**F** UMLGeneralization;

dependency:**F** UMLDependency;

where

forall c1,c2:class@cl . name=c2. name= >c1=c2

forall a: association (exist cl, c2: class @ a. el. name=cl. name/\a. e2. name=c2. name)

forall g: generalization @ (g. parent **in** class)/\ (g. child **in** class)

forall d:denpendency@(d. target **in** class)/\ (d. source **in** class)

end state

3.2 类图的 OOZS 形式化语义

在 UML 类图中,具有相同属性、操作和关系的对象集合称为类,类之间的关联、泛化、依赖等关系为对象上的约束。类图的语义模型不仅要能描述与某个类相关的对象集合,而且还要表达出对象集合之间的约束。每个对象对应于一个实例,每个实例有且仅有一个标识,对象实例存在约束,用 OOZS 定义一个集合:[OId]。S 用来描述与类图(CD)相关的用户模型空间,表达对象实例和约束的集合,link 描述两个对象之间的一种关系,则有

S::=**state**

objects:UMLClass——>POId;

link:Name——>(OId<->OId);

end state

语义映射就是把 UML 类图中的语法元素在语义域中解释其含义。UML 类图的组成元素通过指派得到其相应的语义。在 CD×S 的问题域上,一个语义指派如下定义:

Interpretation::=CD&S

其中,CD::= UMLClass | UMLAssociation | UML-Generalization| UMLClassDiagram。这样,可以通过语法和语义之间的联系来完成 UML 类图到 OOZS 形式化语义的映射。

1)类。MapToUMLClass::=**state**

c:UMLClass&S;

where

forall c:UMLClass,s:S@c **in dom**(s. obejct)

end state

2)关联。MapToUMLAssociation::=**state**

a:UMLAssociation&S;

where

forall a:UMLAssociation,s:S@
dom(s.link(a.name))**sub** s.object(a.el.class)/\
ran(s.link(a.name))**sub** s.object(a.e2.class)/\
(**forall** i:s.object(a.e1.class)@#{j:s.object(a.e2.class)|(i,j)**notin** s.link(a.name)} **notin** a.e2.multiplicity)/\
(**forall** j:s.object(a.e2.class)|#{i:s.object(a.e1.class)|(i,j) **in** s.link(a.name)} **in** a.e1.multiplicity)
end state

3)泛化。MapToUMLGeneralization::=**state**
g:UMLGenrealization&S;
where
forall s:S,g:UMLGereralization@
s.object(g.child) **in** s.object(g.parent)
end state

4)类图。MapToUMLClassDiagram::=**state**
cd:UMLClassDiagram&S;
where
forall d:UMLClassDiagram,s:S@
(**forall** c:d.umlclass@c&s)/\
forall g:d.umlgeneralzation@g&s)/\
(**forall** a:d.umlassociation@a&s)
end state

4 实例应用

4.1 研究生成绩管理系统及其类图

研究生成绩管理系统主要维护研究生的成绩记录情况。在文献[10]中,使用UML对系统的需求模型、静态模型、动态模型及实现模型进行了建模。该系统功能强大,同时也相对复杂。这里,将系统内的权限管理部分提取出来,并结合成绩管理的部分功能,作为本文的一个研究实例。

建立UML类图之前,首先需要了解本实例系统主要实现的功能:

1)管理员可以添加、删除、修改、查询成绩;

2)教师可以添加、修改、查询成绩;

3)学生可以查询成绩。

由上述系统功能可以看出,系统用户主要分为3类:管理员、教师、学生,这3类用户使用该系统的权限不一样。管理员、教师和学生这3类用户所获得的系统权限级别是从高到低的;但他们操作的对象都是一样的,即"学分"。由此可以确定本系统的4个类:学分(Grades)、管理员(Administrators)、教师(Teachers)以及学生(Students)。系统所有功能均由类Grades封装提供,只是在执行操作时必须判断系统用户的权限。系统类图如图1所示。

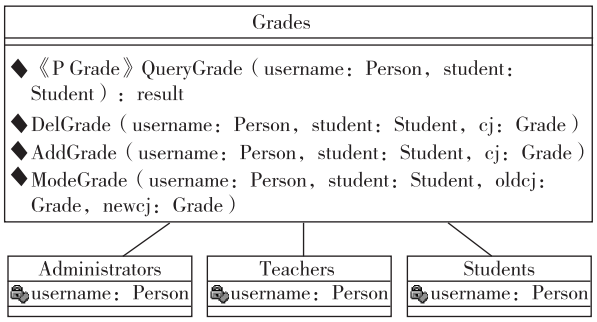

图1 研究生成绩管理系统UML类图

4.2 实例系统UML类图的OOZS描述

1)系统数据类型的声明。首先,实例中涉及管理员、教师和学生共3类用户,将其抽象为Person类型,这3类用户的权限由高到低,分别抽象为Uadmin、Uteacher、Ustudent,在执行某一个操作的时候,根据传入的用户参数(Person类型)判断该用户的权限级别,由此确定该用户能否执行此项操作;此外,系统还涉及到的对象有"学生"和"成绩",这里分别将其抽象为Student和Grade。在givenset部分作如下类型声明:

givenset[Person, Uadmin, Uteacher, Ustudent, Student,Grade];

2)成绩(Grades)类的OOZS描述。

class Grades;
owns;

在本实例中,没有需要用到的常量,因此没有定义const;要求用到的系统状态变量主要有:

state
username:**P** Person;
student:Student;
grade:Student+- >Grade;
where
dom grade **subeq** student;
Uadmin **sub** Uteacher **sub** Ustudent;
Person= Uadmin||Uteacher||Ustudent;
end state;
end owns;

在所有的操作模式中,要求传入一个Person类型的用户参数,然后在操作模式的pre部分根据约束谓词判断该用户所属权限级别,由此确定该用户是否可以执行该操作。

查询成绩的操作QueryGrade是3类用户都可以执行的,是权限级别最低的操作,只要求用户属于Ustudent即可,描述如下:

schema
QueryGrade (username?: Person; student?: Student)==>(result!:P Grade);
pre
username? **in** Ustudent;

post

result! =grade{student?};

end schema;

删除成绩的操作 DelGrade 要求的用户权限级别是最高的,它要求使用该操作的用户必须具有 Uadmin 级别的权限,描述如下:

schema

DelGrade(username?:Person;student?:Student;cj?:Grade);

pre

username? **in** Uadmin;

student? ->cj? **in** grade;

post

grade′=grade\{ student? ->cj?};

student′=student;

end schema;

添加成绩的操作 AddGrade,它要求用户至少具有 Uteacher 级的权限,属于 Uteacher 和 Uadmin 级别的用户都可以执行该操作,描述如下:

schema AddGrade(username?:Person;student?:
Student;cj?:Grade);

pre

username? **in** Uteacher;

student? ->cj? **notin** grade;

post

grade′=grade||{ student? ->cj?};

student′=student;

end schema;

修改成绩的操作 ModGrade,它要求用户至少具有 Uteacher 级的权限,属于 Uteacher 和 Uadmin 级别的用户都可以执行该操作。修改成绩的实质就是先删除原来的成绩,再重新添加一个新的成绩进去,由前面的 AddGrade 和 DelGrade 两个操作模式合并即可得到 ModGrade 的操作模式:

schema ModGrade(username?:Person;student?:
Student;oldcj?,newcj?:Grade);

pre

username? **in** Uteacher;

student? ->newcj? **notin** grade\{student? ->oldcj?};

post

grade′ = grade \ { student? -> oldcj?} ‖
{student? ->newcj?};

student′=student;

end schema;

5 结束语

UML 与形式化方法的优势与不足是显而易见的,将 UML 和形式化方法相结合成为国内外研究人员的一个研究热点,文献[11-24]列出了部分国内外结合 UML 和形式化方法的研究工作。本文的研究工作正是建立在这些大量的基础研究之上,采用结构化面向对象规格说明语言 OOZS 对 UML 类图进行形式化描述,并将它成功地运用于实际软件的系统建模中。相对其他形式化描述语言,OOZS 除了能够支持面向对象系统分析后的用户需求规格说明的编写外,它还具有进行软件求精和程序变换的特点,也就是说它不但能够直接被计算机处理,而且还降低了规格说明语言的抽象程序,这也是下一步重点研究的工作。

参考文献:

[1] 祝义,张广泉. 基于 UML 和 Z 的软件体系结构求精方法及其应用[J]. 计算机工程与应用,2006,42(5):59-62.
Zhu Y,Zhang G Q. Software architecture refinement and its application base on UML and Z[J]. Computer Engineering and Applications,2006,42(5):59-62.

[2] 缪淮扣,陈怡海. 带 OCL 约束条件的类图到 Object-Z 的规格说明的转换[J]. 计算机科学,2007,34(1):228-235.
Miao H K,Chen YH. Transformation UML class diagrams with OCL constraints into object-Z formal specification[J]. Computer Science,2007,34(1):228-235.

[3] Woodcock J,Davies J. Using Z:Specification,refinement,and proof[M]. NJ,USA:Prentic-e-Hall,Inc,1998.

[4] 缪淮扣,李刚,朱关铭. 软件工程语言——Z[M]. 上海:上海科技文献出版社,1999.
Miao H K,Li G,Zhu G M. Software engineering language—Z[M]. Shanghai:ShangHai Scientific and Technological Literature Publishing House,1999.

[5] Smith G. The object-Z specification language[M]. Boston,USA:Kluwer Academic Publis-hers,2000.

[6] 唐稚松. 时序逻辑程序设计与软件工程(上、下册)[M]. 北京:科学出版社,2002.
Tang Z S. Temporal logic programming and software engineering(Volumes 1 and 2)[M]. Beijing:Science Press,2002.

[7] 李刚,朱关铭,童頫. OOZS:一种面向对象的 Z 语言扩展[J]. 计算机研究与发展,1997,34(10):742-746.
Li G,Zhu G M,Tong F. OOZS:An object-oriented extension to Z[J]. Journal of Computer Research and Development,1997,34(10):742-746.

[8] 李刚,朱关铭,童頫. 结构化面向对象形式规格说明语言 OOZS——设计原理[J]. 上海大学学报:自然科学版,1998,4(2):187-195.
Li G,Zhu G M,Tong F. The design of structured and object-oriented formal specification language OOZS[J]. Journal of Shanghai University:Natural Science Edition,1998,4(2):187-195.

[9] 李刚,朱关铭,童頫. 面向对象的形式规格说明语言 OOZS[J]. 计算机应用与软件,2000,17(3):1-11.

Li G, Zhu G M, Tong F. The syntax of object-oriented formal specification language OOZS[J]. Computer Applications And Software, 2000, 17(3): 1-11.

[10] 肖岗,张广泉. UML 在研究生成绩管理系统建模中的应用[J]. 重庆师范大学学报:自然科学版, 2007, 24(1): 34-36.
Xiao G, Zhang G Q. Application of UML in modeling the postgraduate grade management system[J]. Journal of Chongqing Normal University: Natural Science Edition, 2007, 24(1): 34-36.

[11] 黄正宝,张广泉. UML2.0 顺序图的 XYZ/E 时序逻辑语义研究[J]. 计算机科学, 2006, 33(8): 249-251.
Huang Z B, Zhang G Q. Semantics of UML 2.0 sequence diagrams in XYZ/E[J]. Computer Science, 2006, 33(8): 249-251.

[12] 韦银星,张申生,曹健. UML 类图的形式化及分析[J]. 计算机工程与应用, 2002, 38(10): 5-7.
Wei Y X, Zhang S S, Cao J. A formalization and analysis of UML class diagrame[J]. Computer Engineering and Applications, 2002, 38(10): 5-7.

[13] 李景峰,李琰,陈平. UML 序列图的 Z 形式规范[J]. 西安电子科技大学学报:自然科学版, 2002, 29(6): 772-775.
Li J F, Li Y, Chen P. Formal specification of the UML sequence diagram using Z notation[J]. Journal of Xidian University: Natural Science Edition, 2002, 29(6): 772-775.

[14] 燕昊. UML 建模的形式化研究[D]. 兰州:兰州大学, 2006.
Yan H. Research on the formalization of UML modeling [D]. Lanzhou: Lanzhou University, 2006.

[15] 周欣,魏生民. 基于 B 语言的 UML 形式化方法[J]. 计算机工程, 2004, 30(12): 62-64.
Zhou X, Wei S M. B-based formalization of UML[J]. Computer Engineering, 2004, 30(12): 62-64.

[16] 周瑾,马应龙,李巍,等. UML 的形式化及其应用[J]. 计算机科学, 2005, 32(3): 136-140.
Zhou J, Ma Y L, Li W, et al. A formal frameword of UML and its application[J]. Computer Science, 2005, 32(3): 136-140.

[17] 周彦晖,张为群. 形式化与可视化结合的 FDOOM 软件开发方法[J]. 计算机科学, 2003, 30(9): 179-182.
Zhou Y H, Zhang W Q. The transform between formal and visual software model[J]. Computer Science, 2003, 30(9): 179-182.

[18] 罗密,张为群. 结合形式化方法的 UML 系统开发[J]. 西南师范大学学报:自然科学版, 2003, 28(2): 203-208.
Luo M, Zhang W Q. Formal refinement for the UML development[J]. Journal of Southwest China Normal University: Natural Science Edition, 2003, 28(2): 203-208.

[19] 庞军,王云峰,郑国梁. 基于 COOZ 对 UML 的类结构的形式化[J]. 计算机工程与应用, 2000, 36(6): 86-88.
Pang J, Wang Y F, Zheng G L. Towards a formalization of UML class structures in COOZ[J]. Computer Engineering and Applications, 2000, 36(6): 86-88.

[20] Abrial J R. The B-Book: Assigning program to meanings [M]. NY, USA: Cambridge University Press, 1996.

[21] Lano K. The B language and method: A guide to practical formal development[M]. NJ, USA: Springer-Verlag New York, Inc, 1996.

[22] 李刚,朱关铭,童频. 结构化面向对象形式规格说明语言 OOZS——类型检查器[J]. 上海大学学报:自然科学版, 1998, 4(2): 187-195.
Li G, Zhu G M, Miao H K. The type check tool for OOZS specification[J]. Journal of Shanghai University: Natural Science Edition, 1998, 4(2): 187-195.

[23] 刘峰,尤飞,康亚明. 基于 UML 的嵌入式温室环境监测系统设计[J]. 电子设计工程, 2012, 20(20): 138-140.
Liu F, You F, Kang Y M. Design of embedded the greenhouse environment monitoring system based on UML[J]. SAMSON, 2012, 20(20): 138-140.

[24] 杜博. 高校实验室管理系统的设计与实现[J]. 电子设计工程, 2011, 19(15): 36-38.
Du B. Design and implementation of university laboratory management system[J]. SAMSON, 2011, 19(15): 36-38.

Formal Specification and Application of UML Class Diagram Based on OOZS

XIAO Gang

(School of Postgraduate, Chongqing Normal University, Chongqing 401331, China)

Abstract: Unified Modeling Language is a generic graphical modeling language of object-oriented technique. However, the lack of precise formal semantics makes it difficult to analyze the models they describe. In this aspect, formal methods can be used complementarily. OOZS is an object-oriented formal specification language which is an extension to Z. In this paper we discuss formally the Class Diagram with OOZS language, which provides formal basis for UML, and explores a kind of mechanism for how to map and combine it between UML and OOZS. In the end, an example is given to explain how to combine UML and OOZS to describe the Class Diagram, and to illustrate the feasibility of this study in practice.

Key words: object-oriented method; formal method; OOZS; UML; class diagram

(责任编辑 游中胜)