计算机科学2004Vol.31№.12

OCL与Object-Z作为UML约束语言的分析比较*)

陈怡海　缪淮扣
(上海大学计算机工程与科学学院　上海200072)

摘　要　UML是目前广泛使用的标准的面向对象建模语言。为了提高建模的精确性,UML模型可以用对象约束语言OCL或者是用Object-Z规格说明语言加以补充说明。本文从多个方面分析比较了这两种语言的特点,并提出建模人员应充分利用这两种语言的特点,对系统进行精确的建模。

关键词　UML,OCL,Object-Z,分析比较

Comparison and Analysis of OCL and Object-Z as UML Constraint Language

CHEN Yi-Hai　MIAO Huai-Kou
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072)

Abstract　UML is currently a standard and an intensively used OO modeling language. To increase the precision of the model, UML can be complemented with OCL or Object-Z as constraint language. This paper provides a comparison of these two constraint languages. We propose a precise system by considering different aspects, taking advantage of OCL and Object-Z properties.

Keywords　UML, OCL, Object-Z, Comparison

1　引言

提高软件开发质量的一个重要途径是提高软件需求规格说明的正确性。UML[1]语言作为一种通用的可视化建模语言,在工业界和学术界得到了广泛的应用。UML语言使用各种不同的直觉上容易理解的图形来对系统静态属性和动态属性进行描述。已有越来越多的软件开发人员应用UML语言来对所开发的软件系统进行建模,然而在许多情况下,图形化规格说明尚不能产生一个精确的无歧义的规格说明。为了提高模型的精确性,UML提供对象约束语言OCL[2]。OCL(Object Constraint Language)是一种文本形式的叙述性约束语言,建模人员可以用它来对模型的约束信息进行描述。在需求工程研究领域的另外一个研究热点是将形式化方法和非形式方法结合起来[3,4],其研究的目的就是把UML语言的易用性和形式方法的精确性融合在一起。Object-Z[5]语言是一种常见并且广泛使用的形式规格说明语言,OCL语言与Object-Z语言都可以对UML模型进行精确的说明,本文的目的是从不同的方面对这两种语言进行分析比较。本文首先简要介绍OCL语言和Object-Z语言的背景知识及其基本原理,然后再对其作为UML约束语言的主要优缺点进行了分析,最后从多个方面对这两种语言加以对比总结。

2　OCL语言概述

对象约束语言OCL最初是由IBM公司的Jos Warmer开发出来的一种商业建模语言,在很大程度上受到Syntropy语言和方法的影响。1997年,IBM将OCL语言作为UML语言的一个组成部分提交给了对象管理组织(OMG)并被接纳为标准。期间,OCL语言与UML语言一起经历了多次的版本修订,最新的OCL语言版本是1.5版本,OCL 2.0版本正在评审中。OCL语言是一种文本性的规格说明语言,但是没有传统形式语言的复杂性。OCL语言设计的目的是在提高UML规格说明形式化程度的同时又能被编程人员乐意接受。OCL使用与C++编程语言和Java编程语言类似的表达方式,也是一种纯粹的表达式语言。OCL表达式没有副作用,也就是说它们只能说明系统的状态的改变,而不能改变系统的状态。

由于图形化的规格说明语言远不足以产生一个精确并且无歧义的规格说明,OCL设计的目的就是要消除规格说明中的模糊性,OCL可用于多种不同目的:

①用于说明类上面的各种不变式和类模型的类型;②用于说明衍型的类型不变式;③用于说明操作和方法的前置条件与后置条件;④用于描述监护条件;⑤作为一种导航语言;⑥用于说明操作上的约束。为了说明OCL语言的特点,假设我们要说明Person类的属性age的约束为:每个人的年龄应该大于零。用OCL语言可以表达如下:Person inv:

self.age>0

该OCL表达式对Person类的性质进行了说明,其中关键字inv表示不变式,而关键字self表示对象的性质,圆点后面表示具体的属性。下面的例子中我们将对OCL语言作进一步的介绍,更多的信息请参考OCL规格说明[2]。

3　Object-Z语言介绍

在万维网上的形式方法虚拟图书馆[6]上已列出了超过80种的形式规格说明语言,形式规格说明语言可基本分为基于模型和基于代数的两种类型。基于模型的规格说明以状态变量集合的方式提供了系统状态的模型。常见的面向模型的规格说明语言有Z[7],VDM,B等。基于代数的方法并不提供系统数据的模型;相反,系统的状态是用到达那个状态所需的操

*)本文的研究工作得到国家自然科学基金(项目编号60373072)和上海市教委科技发展基金(项目编号02AK07)的资助人。陈怡海　博士研究生,主要研究方向是软件工程,形式化方法。缪淮扣　教授,博士生导师,主要研究方向是软件工程,形式化方法。

作组合来描述的。常见的基于代数的规格说明语言有 Larch，OBJ，LOTOS 等。描述并发式和交互式系统的形式规格说明语言则有 CSP，CCS，Petri 网等。

Object-Z[3]是 Z 语言面向对象的扩充，由澳大利亚昆士兰大学软件验证研究中心的研究人员开发现有的 Z 语言的语法和语义在 Object-Z 语言中仍然被保留。它是为了方便规格说明书写人员以面向对象的风格构造规格说明。目前，Object-Z 规格说明语言已经在学术界和工业界得到了广泛的应用。一个系统的 Object-Z 规格说明是由一系列的类模式组成的，类模式中封装了状态模式和操作模式，通过继承和多态的方式，规格说明书写人员可以用面向对象的方式构造规格说明，一个 Object-Z 类模式的语法结构如图1所示。

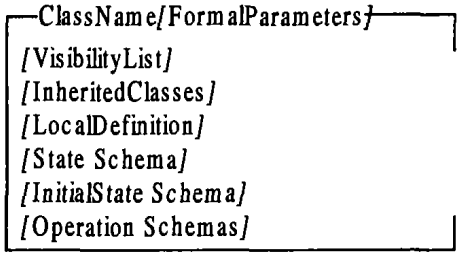

图1　Object-Z 类模式结构的语言图

其中，类定义是一个有名称的模式，也可以带有通用式参数。类的可见列表(VisibilityList)定义了类的接口。可见列表显式地定义了类的哪些属性如：常量，状态变量后，初始状态模式和类的各种操作才可以在该类的对象的环境中被引用。继承类(InheritedClasses)部分定义了类的继承特性，列出所有父类(可以继承一个或多个类)的名称，当一个类被 Object-Z 中的其他类继承时，其定义，也就是包括其局部定义，状态、初始状态模式和操作模式均被合并到继承的类中。类的局部定义(LocalDefinitio)则定义了在类中使用到的类型和常量，它们的格式与 Z 语言中的格式相同。状态模式(State Schema)定义了类的状态变量，并与局部的公理定义一起用来定义类的各种可能状态。初始状态模式(InitalState Schema)是名称为 init 的模式，它定义了该类的对象所允许的初始值。操作状态模式由多个模式组成，类的操作模式(Operation Schemas)定义了类的对象所允许的状态改变。

我们可用一个通用堆栈类[5]的 Object-Z 规格说明为例加以说明：

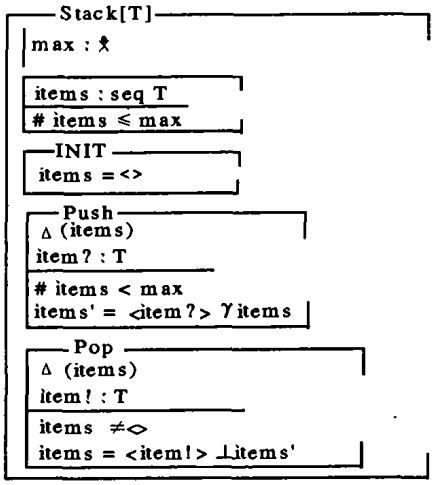

图2　通用堆栈类的 Object-Z 规格说明

该类有一个常量 *max* 和一个表示通用类型 *T* 的元素序列的状态变量 *items*。状态不变式规定序列的长度不能超过 *max*。*Push* 和 *Pop* 两个操作模式则定义了对堆栈的操作。

4　实例分析

Warmer 和 Kleppe[8]对约束给出了以下的定义：约束是指(部分)面向对象模型或者是系统的一个或多个值的约束。为了更好地比较两种语言的特点，我们以一个类图为例演示应如何使用 OCL 语言和 Object-Z 语言来得到精确的规格说明。

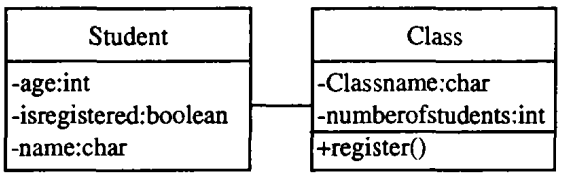

图3　类图的例子

对于类图而言，有三种不同类型的约束：

·不变式：一个不变式是类、类型或者是接口的所有实例都必须满足的条件的约束。不变式可以用如果该不变式满足则为真的表达式来描述，不变式必须在任何时候都为真。

·前置条件：一个操作的前置条件是指在操作被执行前必须为真的约束。

·后置条件：一个操作的后置条件是指在操作执行完成后必须为真的约束。

4.1　OCL 规格说明

对于图3中的类图，它没有给出对于类的属性的约束和执行类中操作所需要满足的条件的说明，例如，对于一个班级而言，只有注册人数大于25人方可开课并且由于教室大小的约束，人数又不能超过80人，对此，在上图中无法进行精确说明，下面我们进一步演示怎样使用 OCL 语言来对该类图加以更精确的描述：

首先考虑如何对类的不变式进行说明，使用 OCL 可以对班级人数的属性加上如下的约束：

班级的学生人数必须大于25人并且小于80人。

```
Context Class
inv:self.numberofstudents>=25 and self.numberofstudents<
  =80
```

每个学生必须经注册后方能听课

```
Context Student
inv:self.isRegistered=true
```

对于操作而言，可以通过前置条件与后置条件加以辅助说明

```
Context Class::register(p:student)
pre registerPre:class->excludes(p)
post registerPost: class->includes(p)
```

上述的 OCL 规格说明表示当学生完成注册以后，学生成为班级的成员。

4.2　Object-Z 规格说明

上一节中我们演示了应如何使用对象约束语言来表达一个对象中的约束和消除规格说明中的模糊性，下面再来演示应如何使用 Object-Z 规格说明语言给出精确的规格说明。对于 UML 类图，我们可以应用以下的转换规则得到对应的 Object-Z 规格说明。

1. UML 的类名可转换为 Object-Z 类模式的类名；

2. UML 类中的属性可在 Object-Z 类模式的状态模式中声明；

3. UML 类中属性的约束可在 Object-Z 类模式的状态模式中的谓词部分声明；

4. UML 类中的操作可转换为 Object-Z 类模式的操作模式，操作的名称可转换为操作模式的名称。操作中的参数可在操作模式的声明部分声明；操作中的前置条件与后置条件可在操作模式的谓词部分声明。

应用上述规则，我们可以得到 Class 类的 Object-Z 规格说明。

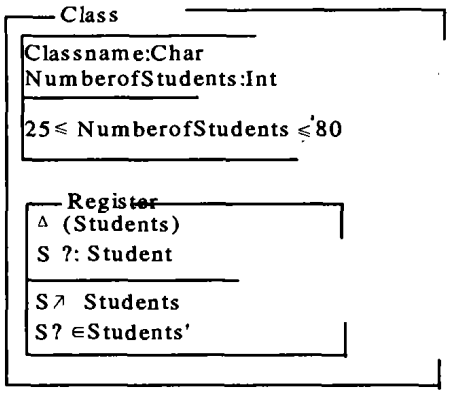

图4 Class 类的 Object-Z 规格说明

5 OCL 语言和 Object-Z 语言的分析比较

在简单介绍了 OCL 语言和 Object-Z 语言的历史背景，并给出实例说明以后，本节再具体比较一下二者的异同。规格说明语言可以从多种角度进行比较：表1是 OCL 与 Object-Z 主要特点的比较。

表1 OCL 与 Object-Z 主要特点的比较

特点	OCL	Object-Z
数学基础	·集合论 ·三值 Kleene 逻辑	·集合论 ·一阶谓词演算
逻辑基础	三值逻辑（True，False，Undefined）	二值逻辑（True，Flase）
可说明的系统特性	只能对系统的约束属性进行说明	·系统的功能 ·顺序程序和并发程序设计
不能支持的系统设计方面	·用户界面 ·系统时间约束 ·空间约束	·用户界面 ·系统时间约束 ·空间约束
可读性和易理解性	好	中等
形式语义	待定	有完备的形式语义
文档外观	使用了大量的关键字	E 型框和模式
状态变化的规格说明	没有明确的表示方法	前状态：不加修饰的变量 后状态：带撇的变量
输入和输出变量的辨别	没有明确的标记	输入：变量名后以?结尾 输出：变量名后以!结尾
前置条件和后置条件的表示方法	·用关键字 Pre 明确表示 ·用关键字 Post 明确表示	没有明确给出前置条件和后置条件
国际标准	有	没有
是否改变系统状态	不会改变	改变
生命周期的覆盖	仅适合于规格说明阶段	支持从规格说明描述到详细设计

说明如下：

·所有形式规格说明方法都是以数学为基础的。有些方法是基于集合论和一阶谓词演算。有些方法是基于时态逻辑；Object-Z 是基于集合论和一阶谓词演算，而 OCL 语言则是基于集合论和三值 Kleene 逻辑。

·从表达能力来分析，Object-Z 语言的表达能力比 OCL 语言更强，OCL 语言只能对模型的约束加以说明，而 Object-Z 语言的描述能力也比它更强。

·从语言的可读性方面比较而言，OCL 语言远优于 Object-Z 语言。毕竟 Object-Z 是一种数学语言，要求书写和阅读者有较好的数学基础，因此 OCL 语言才更易被软件开发人员接受。

·Object-Z 是 Z 语言的面向对象的扩充，它有完备的公理语义和指称语义。而 OCL 语言是一种半形式化的语言，正式的 OCL1.5版本的规格说明中尽管对 OCL 语言的语法进行了详细描述，但是没有对其形式语义进行过定义，而最新 OCL 2.0版本的草案中提出了两种语义，一种是基于元模型的语义，另一种是称之为对象模型的集合论的形式语义，已有研究人员指出这两种语义存在着不一致性和不完备性[9]。

·Object-Z 是一种基于模型的规格说明语言，系统的状态模式中的谓词会改变系统的状态，而在对 OCL 表达式进行求值时并不会改变系统的状态。

·用 OCL 语言书写的规格说明和用 Object-Z 语言书写的规格说明在外观上也不同。OCL 语言使用了大量的关键字（如 self，context，pre，post 等），其风格类似于编程语言。而 Object-Z 语言则是一种数学语言，以 E 框和模式以及大量存在的数学符号为其显著特征。

·Object-Z 语言中可区分变量的前状态与后状态、输入变量与输出变量，而 OCL 语言却没有相应的机制。

·在 OCL 语言中明确地给出前置条件与后置条件，而在 Object-Z 语言中则需通过计算才能得到。

·Object-Z 语言虽然是一种广泛使用的规格说明语言，但它目前没有相应的国际标准。OCL 语言作为 UML 语言的一个组成部分，是一种国际标准语言。

·OCL 语言只能对系统模型的约束进行说明，不能支持从规格说明的描述到详细设计的软件开发全过程。Object-Z 语言则可以支持从规格说明描述到详细设计的软件开发全过程。

6 工具支持的比较

一种规格说明语言要得到广泛的应用，强有力的工具支持是至关重要的。一般而言，规格说明语言的支持工具可以分为下面几种类型：

·可视化编辑工具：主要是用于对规格说明语言进行输入和编辑。

·语法检查工具：是规格说明语言最基本的支持工具，其功能是用于检查规格说明的语法正确性，但这种工具只能发现一些简单的语法错误。

·类型检查工具：规格说明语言均为类型语言。使用类型检查工具的目的是为了保证规格说明类型的正确性。

·语义分析工具：是用来检查规格说明的语义的正确性。

·规格说明的求精：求精是指将规格说明转换到程序代码。

·测试自动化工具：测试自动化是指从规格说明中产生

测试用例，运行测试用例和进行测试结果分析的自动化，这些工具能提高软件开发的效率并且降低软件维护费用。

· 验证和确认工具：主要是用于对规格说明进行验证和确认，以保证规格说明具有所需要的性质。验证和确认的方式主要有模型检查，定理证明和动画技术三种。

6.1 OCL 语言的支持工具

OCL 语言是一种相对较新的语言，因此目前该语言还没有足够的支持工具。大多数商业化的 UML 建模工具并没有普遍提供对 OCL 语言的支持，只有一些研究人员对 OCL 的语义和支持工具进行过研究。其相应工作介绍如下[10]：

6.1.1 *OCL 编辑工具* 目前主流的 UML 编辑工具如 ArgoUML 和 MagicDraw 等都提供了对 OCL 语言的支持，并提供了 XMI 格式的输出。

6.1.2 *词法分析和类型检查工具* 首先出现的 OCL 支持工具是 IBM 公司出品的 OCL 词法分析器，它是用 Java 和 JavaCC 词法产生器编写的。该分析器的功能包括语法分析和部分的类型检查。后来出现的 OCL 词法分析器还有 Klasse Objecten 公司出品的 OCL 词法分析器，但它只能提供词法分析。ArgoUML 工具中集成了 Dresden 提供的 OCL 编译器，它能提供完整的 OCL 语法和类型检查。在 MagicDraw 工具中也可以对 OCL 语法进行检查。

6.1.3 *其它 OCL 语言支持工具* Bremen 大学开发了 USE（UML-based Specification Environment）系统工具。USE 是一种用于信息系统规格说明的验证确认工具。USE 能对系统的模型进行动画并对规格说明和用 OCL 语言表达的非形式需求进行确认。在对系统模型进行动画的过程中用户可以创建并操纵系统状态。

6.2 Object-Z 语言的支持工具

形式化方法不能获得广泛应用的最重要原因是缺乏有效的支持工具。至今作为 Object-Z 的图形化编辑工具的有 Moby/OZ[11]，Wizard 提供了基于 Latex 格式的 Object-Z 规格说明的类型检查功能。ZML 提供了 Z、Object-Z 和 TCOZ 的基于 Web 方式的可视化环境。我们课题组正在进行的研究项目旨在提供支持 Object-Z 的设计开发环境，现在已成功地开发出了 Object-Z 语言编辑工具 OZEditor[12]，该工具提供了 Latex 格式的 Object-Z 规格说明的输出以及 Object-Z 规格说明的语法检查和类型检查功能，同时还提出了从 Object-Z 规格说明中产生测试用例的算法并开发了相应的工具[13]，设计实现了从 UML 模型到 Object-Z 规格说明的转换工具[14]。Object-Z 语言的模型检查和动画工具的研究工作也正在进行之中。

小结：目前大多数 UML 建模工具提供了诸如绘制 UML 模型，代码产生以及逆向工程等功能，我们发现只有很少的工具能给 OCL 语言提供支持。而缺乏有效工具支持的重要原因是因为目前 OCL 语言正处于发展过程中，这给 OCL 语言的使用带来了障碍。相对而言，Object-Z 语言则是一种更为成熟的语言。

结束语 我们介绍了 OCL 和 Object-Z 两种规格说明语言，展示了如何利用约束语言来提高 UML 模型的精确性，并对两种规格说明语言进行了分析和比较。结果表明，OCL 语言具有易表达性和易理解性的优点，但是缺乏形式化语义和精确性及支持工具，而这些缺点正是 Object-Z 语言所拥有的优点，因此这两种语言具有很强的互补性，我们已开展了 UML 和 Object-Z 语言相结合的研究工作[4,14]，并着手定义从 OCL 到 Object-Z 的形式转换工作，以便从 UML 模型中得到更为完整的 Object-Z 规格说明，Hung 等人[15]的研究成果是我们很好的工作起点。

参考文献

1 Booch G, Rumbaugh J, Jaccobson I. The Unified Modeling Language User Guide. Addison Wesley Longman, Inc. American, 1999

2 OMG. Unified modeling Language. Specification v 1.5. http://www.omg.org

3 Bruel J M. Integrating Formal and Informational Specification. Why? How?. 2nd International Workshop on Industrial-strength Formal Techniques(WIPT'98), Boca Raton, FL, USA, 1998

4 Chen Yihai, Miao Huaikou. Integrating Object-Oriented Methods and Formal Methods For Requirement Engineering. 见：第七届国际青年计算机会议(ICYCS 2003)，哈尔滨，2003

5 Smith G. The Object-Z Specification Language. Kluwer Academic Publishers, American, 2000

6 形式方法虚拟图书馆. http://archive.comlab.ox.ac.uk/formal-methods.html

7 缪淮扣，李刚，朱关铭. 软件工程语言—Z. 上海科学文献出版社，1999

8 Warmer J, Kleppe A. The Object Constraint Language-Precise Modeling with UML. Addison-Wesley, 1999

9 Flake S, Mueller W. Formal Semantics of OCL Messages. Sixth International Conference on the Unified Modeling Language -the Language and its applications, (UML 2003), San Francisco, California, USA, 2003

10 OCL Center. http://www.klasse.nl/ocl/index.html

11 Object-Z Online Reference. http://www.itee.uq.edu.au/~smith/tools.html

12 羊冬昭. Object-Z 编辑器的分析，设计和实现：[上海大学工学学位论文]. 上海大学，2003

13 Liu Ling, Miao Huaikou, Zhan Xuede. A Framework for Specification-Based Class Testing. In: Eighth Intl. Conf. on Engineering of Complex Computer Systems, Greedbelt, Maryland, USA, 2002

14 陈恰海，缪淮扣，高如海. 基于 XMI 格式的 UML 模型的交换. 见：第十届全国计算机大会. 北京，2003

15 Ledang H, Souquieres J. Derivation Schemes from OCL Expressions to B: [Technical Report A02-R-042]. LORIA, 2002