第 5 章

UML 概 述

统一建模语言(UML)指的是 OMG 的 Unified Modeling Language™(OMG UML®)。UML 已经是建立面向对象软件的事实标准。OMG®是一个国际非营利制定分布式对象计算领域标准的软件联盟。UML 是一种图形化的语言,用于软件密集系统要素的可视化、规范制定、对象构建和文档编写。UML 为描述软件系统的设计提供了标准,设计既包括概念方面,如业务过程和系统功能,也包括具体实现,如对象状态转换和可复用的软件组件。

5.1 UML 的作用

建造一座大楼或者装修一套单元房都是复杂的工程,需要不同的材料、不同的技术、大量的工时和精细的项目管理。例如,装修一套单元房,那么在实际施工之前,利益相关者(房主、设计师、物业)首先应该对房子有一个概念上的认识,并对装修需求取得一致。此时平面结构图是双方沟通的一个很好的模型,如图 5-1 所示。

图 5-1　平面结构图

对于水电工程师而言，水路图（如图 5-2 所示）将是改造水路的依据。在改造施工之前，一幅水路图就可以计算出成本。

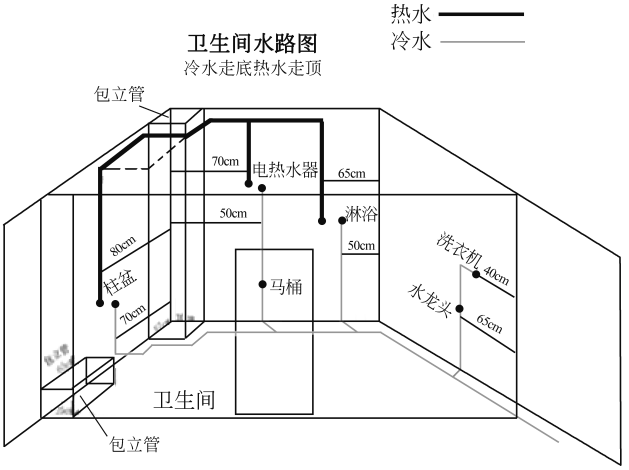

图 5-2　卫生间水路图

可以看到，在建筑装修工程中，不同的时期，面向不同的利益相关者，通过不同的图展示了一套房屋的不同方面。在这些图中，往往约定了一些图形符号表示特定含义，如墙体、门、窗、热水管、冷水管等。所以，在真正的构建活动之前，客户和装修服务供应商之间、装修设计师和施工人员之间需要双方可理解的“图”进行沟通，这些“图”展现了目标的模型。设计图的过程也就是建立模型的过程，简称建模。

建模也是开发大型软件项目的重要活动。模型在软件开发中的作用与模型在住宅楼的建筑过程中的作用是相同的。使用模型，软件开发服务的提供者可以确保自己的业务功能完整且正确，满足最终用户的质量需求，因为一旦施工（将模型转变为代码），那么将使更改变得困难且昂贵。为了保证沟通上不产生歧义和误解，利益相关者需要对模型中使用的符号的含义进行严格的约定，这就是使用 UML 的好处。UML 不但可以按照不同阶段、不同参与者的需要对软件系统进行可视化，还可以对系统的结构和行为做出无二义的描述。

UML 定义了下列四方面的模型元素和图形元素。

（1）人机交互。使用用例模型描述系统和用户之间的界定和交互，也可作为需求模型。

（2）结构。通过类模型描述构成系统的类及其关系，通过物理组件模型描述构成系统的软件（有时也包含硬件），通过物理部署模型描述物理架构与物理架构中组件的部署。

（3）行为。使用协作模型描述系统中的对象彼此之间如何进行交互以完成业务功能。

（4）状态。使用状态机描述随着时间和事件变化对象所呈现的状态和条件，活动图则描述对象行为执行的流程。

UML 也定义了一些扩展机制，如版型，以满足特别需求。

UML涵盖了整个软件生命周期的建模。在需求分析阶段,通过建立用例图等模型来描述系统的使用者对系统的功能要求;在分析和设计阶段,通过类及其关系建立静态模型,通过对象之间的协作进行动态建模,为开发工作提供详尽的规格说明;在编码阶段,把设计的模型转换为编程语言的实际代码,指导并减轻编码工作;在测试和维护阶段,以UML模型作为测试和维护依据。

5.2 UML的发展

UML是由著名面向对象技术专家Grady Booch、James Rumbaugh和Ivar Jacobson发起的。

Grady Booch的对象模型要素主要是封装、模块化、层次类型和并发。使用的图形文档包括类图、对象图、状态转换图、交互图、模块图和进程图六种,该方法使问题域和责任域很好地对应起来,称为Booch方法。1991年,James Rumbaugh提出对象建模技术(Object Modeling Technique,OMT),从静态建模(类图、包图)、动态建模(顺序图、状态机图、活动图)入手,试图找出问题空间与系统目标之间的关系。1992年,Ivar Jacobson提出了在面向对象软件工程(OOSE)中应用用例驱动的途径。用例一方面为建模找到了原始信息来源,另一方面将建模和需求采集结合起来。

Grady Booch和Jimes Rumbaugh首先将Booch方法和OMT统一起来,于1995年10月发布了第一个公开版本UM 0.8(Unified Method 0.8),后来Jacobson参与到这一工作中,于1996年6月和10月分别发布了两个新的版本,即UML 0.9和UML 0.91,并将UM重新命名为UML(Unified Modeling Language)。

由十几家公司组成的"UML伙伴组织"将各自的意见加入UML,于1997年1月形成UML 1.0。OMG于1997年11月正式采纳UML 1.1作为建模语言规范,然后成立任务组进行不断的修订,并产生了UML 1.2、1.3和1.4版本,其中,2000年3月发布的UML 1.3是较为重要的修订版。2005年4月,UML 1.4.2成为ISO标准。2005年,OMG发布了UML 2,这次修订采用了更严格的底层建模基础设施——OMG的元对象框架(Meta-Object Framework,MOF™)。MOF意味着UML图不仅对人类可视可理解,还能以机器可读的形式表示,从而可以对设计方案进行推理、执行一致性检查,甚至自动生成部分应用程序代码。以这种方式创建、存储和变换机器可读的模型把建模置于软件生产过程的核心,并形成对象管理组OMG的模型驱动体系结构(Model Driven Architecture®,MDA®)。

当前的版本是2017年12月发布的UML 2.5.1,支持14种图。它们可以分成两大类:结构图和行为图。结构图包括包图、类图、对象图、组件图、部署图、概要图和复合结构图;行为图包括用例图、活动图、状态机图、顺序图、时序图、交互概览图和协作图。其中,顺序图、时序图、交互概览图和协作图统称为交互图。结构图展示了系统的静态结构,包含不同的抽象和实现级别上的事物及其之间的关系;而行为图展示了系统中对象的动态行为,即系统随时间而发生的状态变化。这些图之间的泛化关系如图5-3所示。

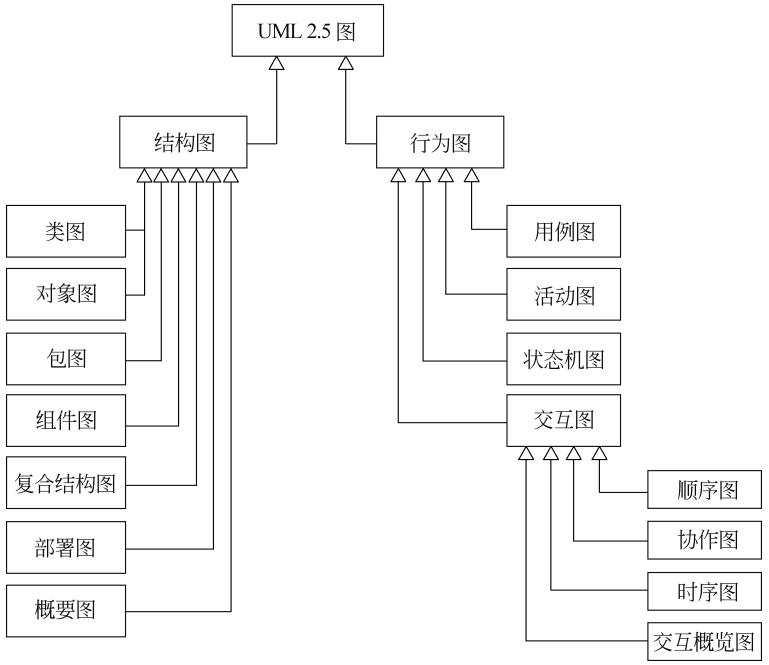

图 5-3　UML 图的分类

类图展示所设计的系统、子系统或组件的类及其之间的关系。类图中的模型元素包括类、关系、属性、操作和接待等。对象图展示类和接口的实例(对象)及其之间的链接。模型元素包括对象、链接等。包图展示包之间的依赖关系。模型元素包括包和依赖关系等。组件图揭示组件和组件之间的关系。部署图展示系统的架构,将软件工件部署(分发)到物理目标上,模型元素包括工件和节点,节点有服务器、路由器、负载均衡器、防火墙等。概要图作为 UML 标准的轻量级扩展机制,允许定义定制的原型、标记值和约束。概要允许对不同的 UML 元模型进行调整。复合结构图展示类的内部结构及相互关系。用例图描述软件系统与外部用户协作执行的一组操作,以向利益相关者提供一些可观察和有价值的结果。建模元素包括用例、参与者、主体、扩展、包含、泛化等。活动图展示协调低级行为的顺序和条件。建模元素有活动、动作、控制流、对象流、分区、对象等。状态机图用于通过有限状态转换模拟离散行为。行为状态机图的建模元素有状态、转移、事件、动作、区域等。

顺序图是常用的一种交互图,着眼于对象在生命周期内的消息交互。建模元素有生命线、消息、复合片段、交互使用、状态不变式、泳道等。状态不变式是某类所有对象状态上都为真的谓词。协作图主要目的是展示对象之间的交互。消息的顺序编码展示了消息的顺序。建模元素有三个:对象、链接和消息。时序图重点关注沿着线性时间轴在对象内部和对象之间状态的变化情况。建模元素包括生命线、状态、消息、持续时间约束、时间约束等。交互概览图提供一种概括性的视图,侧重于宏观上的交互和交互使用,建模元素有交互、交互使用以及活动图中的控制节点。

5.3　UML 的特点

统一建模语言(UML)是一种图形语言,也是一种标准。UML 是业务需求分析人员、软件架构师和程序员通用的语言,用于描述、定义、设计和归档现有或新的业务流程、软件系统工件的结构和行为。

模型驱动的体系结构(MDA®)帮助软件用户应对当下软件环境的两个至关重要的事实:多种候选的软件实现技术和面向整个软件生命周期的永不停止的对软件的改正性、扩展性和适应性维护。UML 支持创建和管理精确详细的表示应用程序结构和行为的、机器可读的模型,并独立于语言、数据库管理系统、操作系统、技术框架以及软件工程过程。

UML 独立于过程,可以应用于不同过程的上下文中。但是,它最适合于用例驱动、迭代和增量开发过程,如 Rational Unified Process(RUP)和敏捷开发。虽然 UML 独立于领域和技术平台,但 UML 提供一个轻量级用于创建 UML“方言”(称为“概要”)的定制机制。通过向标准 UML 工具中加载“概要”,就能使得 UML 建模工具支持或者增强支持特定的技术目标平台或应用领域。标准的 UML 概要用于定制 UML 特定语言,如 Java 和 XML。实时系统、容错系统、基于 CORBA 的分布式计算平台,以及面向服务的体系结构(SOA)平台都可以通过定义概要与 UML 模型进行衔接。

软件开发有许多利益相关者参与:分析人员、设计人员、程序员、测试人员、质量保证人员、客户等。不同角色的参与者对系统的不同方面感兴趣,例如,程序员需要根据接口设计进行实现类编码;测试人员关心如何生成测试用例。UML 涉及各种可能的视角,目标就是与所有利益相关者进行清晰明确的沟通。UML 是一种具有充分表达力的语言,使得软件项目的利益相关者都能获得自己关心的信息。UML 模型是软件服务的提供者与客户之间进行有效沟通的媒介。UML 模型为软件维护提供了准确的、易于理解的、有效的软件系统可视表示,降低了维护成本。由于软件的设计先于软件的编码,所以通过 UML 模型可以很容易地识别设计中可复用的部分,从而降低软件开发成本,提高软件可维护性。UML 模型有利于向开发团队的新成员进行有针对性的培训。

UML 展现了一系列最佳工程实践。这些最佳实践在对大规模复杂系统进行建模特别是在软件架构层次方面,已经被验证有效。UML 是一个免费可得的标准。基于这个标准形成了一个繁荣的工具供应商和开源工具产品社区。基于标准还允许来自多个供应商的工具在一个项目中一起使用。

5.4　UML 建模工具

UML 建模工具有 StarUML,Enterprise Architect,Rational Software Architect 等。

5.4.1 StarUML

StarUML 是一款灵活而且简单易用的高级软件建模工具，其特点是支持模型元素与图形元素的分离。这使得该建模工具成为软件项目模型的管理系统而不仅是一个图形编辑器。该产品的界面如图 5-4 所示。

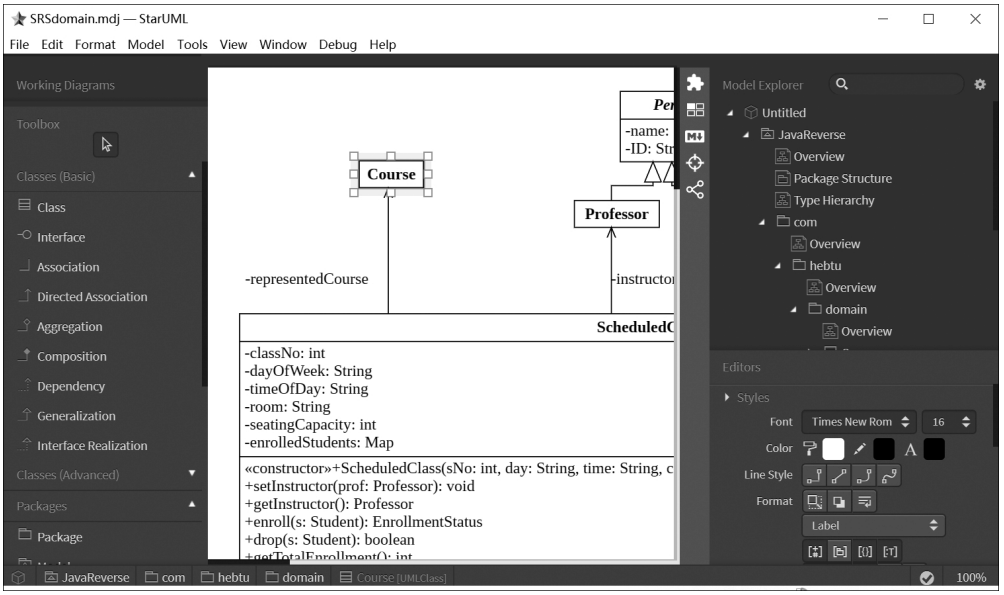

图 5-4 StarUML 界面

StarUML 支持 UML 2.x 标准模型和图：类图、对象图、用例图、组件图、复合结构图、部署图、顺序图、协作图、状态机图、活动图和概要图。除此以外，还支持创建实体-关系图(Entity-Relationship Diagrams，ERD)、数据流图(Data-Flow Diagrams，DFD)和流图(Flowchart Diagrams)。可运行于 Mac OS、Windows 和 Linux。支持 High-DPI 显示，所有图均可导出为 PNG、JPEG 等格式的 High-DPI 影像文件和以 JSON 格式存储建模数据。通过扩展，支持主流编程语言 Java、C＃和 C++ 等的代码生成和反向工程。打开和保存模型时自动执行预定义的验证规则(Validation Rules)。通过把模型转换为 HTML 文档，可以与分析师、架构师和程序员分享模型。可以把图导出为 PDF 格式实现高清晰度打印。使用标记语法(Markdown Syntax)编辑建模元素的文档，支持语法高亮和预览。

详细的 StarUML 用法见附录 A。

5.4.2 Enterprise Architect

Enterprise Architect 是一款高效专业的 UML 软件建模工具，这款工具软件不仅实现了 UML 图的编辑，还为用户提供了软件开发需要的各种功能：需求分析、详尽的设计、测试、发布、部署等。其界面如图 5-5 所示。

Enterprise Architect 支持全方位、全生命周期建模：业务和 IT 系统、软件和系统工

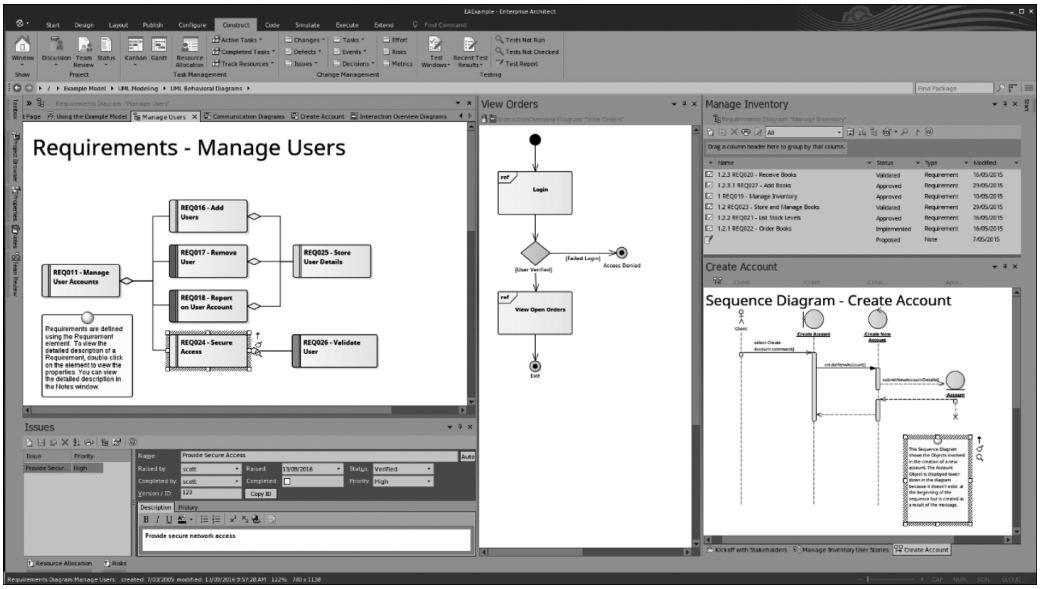

图 5-5　Enterprise Architect 界面

程、实时和嵌入式开发等。Enterprise Architect 支持遵循 UML、SysML(Systems Modeling Language,系统建模语言)、BPMN(Business Process Model and Notation,业务流程建模与标注)和其他开放标准分析、设计、实现、测试和维护模型。通过 BPMN 和 Eriksson-Penker 概要文件把业务流程、信息和工作流程结合到了 UML 模型中。

Enterprise Architect 通过关系矩阵和层次视图等功能,可以在整个生命周期内进行有效的验证,进行影响分析,以构建稳健可维护的系统。具有完整的 WYSIWYG 模板编辑器、文档生成和报告工具。内置的源代码编辑器允许在同一环境中快速从模型直接导航到源代码。代码生成模板则支持根据自定义规范生成源代码。支持许多流行语言的源代码的生成和逆向工程,包括 C、C++、C#、Java、Delphi、Verilog、PHP、VHDL、Python、System C、B.Net 和 Visual Basic 等。

5.4.3　Rational Software Architect

IBM® Rational® Software Architect(RSA)是一个高级而又全面的应用程序设计、建模和开发工具,用于实现端到端的软件交付,支持软件开发的全过程。全面支持 BPMN2、SOA 和 Java 企业版,还提供了与 IBM 的应用程序生命周期管理解决方案相集成的工具。RSA 是 IBM 公司在 Rational Rose 的基础上开发的产品,其中的 Rational Software Modeler 是一个基于 UML 2.0 的工具,它允许创建系统的不同视图,在建模视图下能够进行创建 UML 图、生成代码等操作,提供对 Java、C++、VB、Delphi、SQL、Oracle 等软件的支持。RSA 界面如图 5-6 所示。

其他工具软件还有 Visual Paradigm、Visio、Together、GreenUML、UMLet 等。

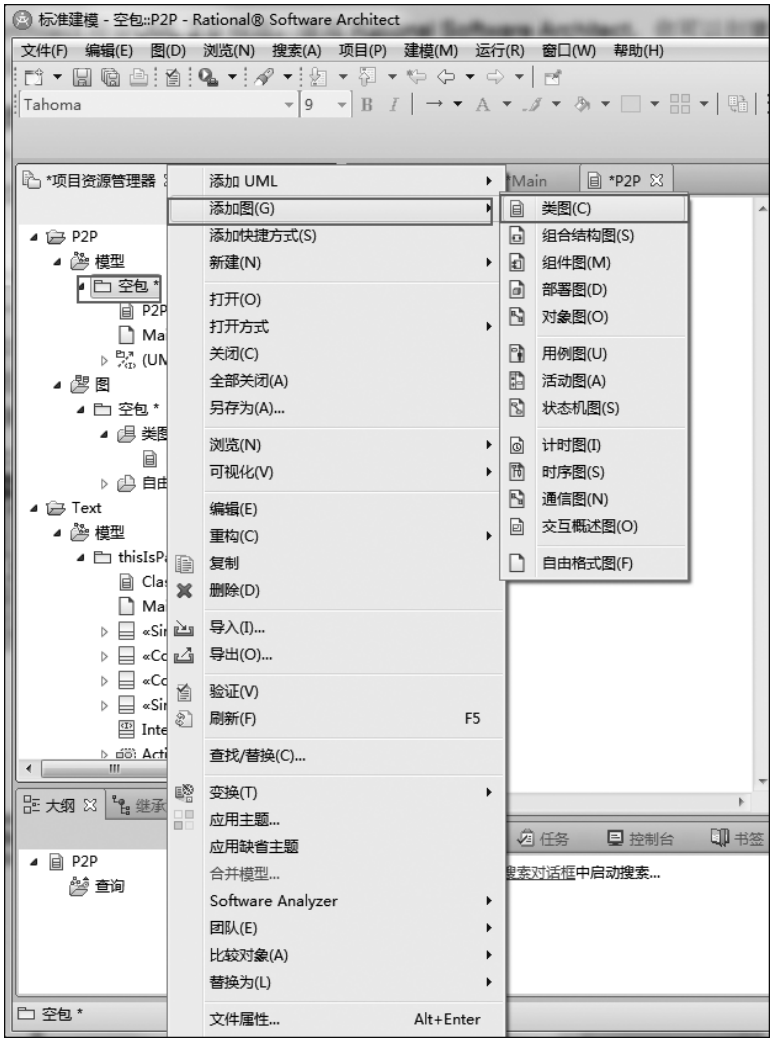

图 5-6 IBM RSA 界面

思 考 题

1. UML 2.5.1 规范中有哪些图？
2. UML 概要有何作用？
3. UML 与软件开发过程、软件开发方法和软件开发工件有何关系？

第6章 UML基本概念

模型是人们对所关心的事物的抽象的结果。通常将被建模的事物的全体称为论域。对于一个现有系统，模型是该系统特征的抽象表达；对于一个计划开发的系统，模型是对该系统如何被构造以及系统行为的规格说明。

元素(Element)是UML模型中的基本成分，每个元素都可以拥有其他元素。一个UML模型包含三种模型元素：特征集(Classifier)、事件(Events)和行为(Behaviors)。特征集是具有相同特征和行为的对象的集合。对象是具有状态的并和其他对象有关系的个体。对象特征以及相应的值构成了对象的状态。事件是一些事情的集合，这些事情的发生对系统产生了影响。行为是执行的集合，执行指的是生成事件或响应事件的一系列动作，包括可能访问或改变了对象的状态的动作。注意，UML模型中并没有对象、事情和执行，因为它们是论域中的事物。

关系(Relationship)也是一种元素，这种元素表示另外两个元素之间的关系。有向关系(Directed Relationship)是源模型元素用于目标模型元素的关系。

特征集、事件和行为等UML所使用的术语称为UML元类(Metaclass)。元类的名字使用大驼峰命名法，例如DirectedRelationship；特征名字使用小驼峰命名法，例如ownedAttribute。

数学建模就是使用数学语言表达具体的问题；UML建模就是使用UML描述目标系统。UML建模总是按照一定的目的、从某一角度、从纷繁复杂的细节中抽取出感兴趣的个体事物的特征集、事件和行为。

6.1 类型和多重性

类型(Type)和多重性(Multiplicity)都是对一些值增加约束。类型定义了所允许的值(Value)的集合，这些值称为类型的实例(Instance)。包含在集合中的值的数目称为集合的基数(Cardinality)。对集合的有效基数约束称为多重性，这个约束要求基数不得小于指定的下界，也不得大于指定的上界(除非多重性是无限的，在这种情况下，上界没有限制)。例如，一门课的容量是40，那么“选课学生清单”这个集合的多重性就是闭区间[0,40]。

多重性元素(Multiplicity Element)是指定它所表示的集合的有效基数的UML元素。多重性元素的isOrdered特性用于设置是否有序，isUnique特性用于设置是否唯一。根据有序性和唯一性，一组值就可以形成四种类型的群集(Collection)，如表6-1所示。

表 6-1 四种类型的群集

isOrdered	isUnique	群集
假	真	集合(Set)
真	真	有序集合(OrderedSet)
假	假	包(Bag)
真	假	序列(Sequence)

在 UML 中,使用文本字符串记号“<下界>..<上界>”表达多重性。下界是一个整数,上界是一个整数或者“*”,“*”表示“没有限制”。如果多重性元素和记号为字符串的元素关联(例如特性),那么把多重性元素使用中括号“[]”括起来并放置在文本字符串里;如果多重性元素和记号为图形符号的元素关联(如关联的端点),则直接把该元素放置在所关联的元素附近。如果下界和上界相等,则直接使用上界。例如,“1”等价于“1..1”。下面是多重性的 BNF 定义。

```
<多重性> ::= <多重性范围> [ [ '{'<有序性> [',' <唯一性> ]}] | [ '{ '唯一性> [',
'<有序性> ] '}']]
<多重性范围> ::= [ <下界> '..' ] <上界>
<下界> ::= <值>
<上界> ::= <值>
<有序性> ::= 'ordered' | 'unordered'
<唯一性> ::= 'unique' | 'nonunique'
```

例如,对于一门容量为 40 的课的“选课学生清单”,要求记录选课的先后次序,并且不能重复选课,则其多重性可以表达为:

```
enrolledStudents: Student[0..40]{ordered, unique}
```

巴科斯-诺尔范式(Backus-Naur Form,BNF)是一种形式化的语法规则描述语言。每条语法规则有左部和右部两部分:左部是一个语法成分,右部是由语法成分和字符组成的字符串,中间以“::=”连接,读作“定义为”。具有相同左部的规则可以共用一个左部,各右部之间以直线“|”隔开,读作“或”。语法成分用尖括号括起来,字符或者字符串用单引号括起来。方括号“[ ]”表示其中内容为可选项,花括号“{ }”表示其中项目可重复 0 次或若干次。

6.2 名字空间

名字空间(Namespace)是包含一组命名元素(Named Element)的建模元素。命名元素是可通过名字(Name)标识的建模元素。包(Package)就是一种名字空间。

名字空间是命名元素的容器。这些命名元素称为名字空间所拥有的成员(ownedMember)。一个名字空间可以从另外一个名字空间导入命名元素。如果名为 N 的命名空间中有成员名为 x 的命名元素,则可以由形式 N::x 的限定名(Qualified Name)