UML类图

本词条缺少**名片图**，补充相关内容使词条更完整，还能快速升级，赶紧来编辑吧！

在UML的静态机制中类图是一个重点，它不但是设计人员关心的核心，更是实现人员关注的核心。建模工具也主要根据类图来产生代码。类图在UML的9个图中占据了一个相当重要的地位。James Rumbaugh对类的定义是：类是具有相似结构、行为和关系的一组对象的描述符。类是面向对象系统中最重要的构造块。类图显示了一组类、接口、协作以及他们之间的关系。在UML中问题域最终要被逐步转化，通过类来建模，通过编程语言构建这些类从而实现系统。类加上他们之间的关系就构成了类图，类图中还可以包含接口、包等元素，也可以包括对象、链等实例。接口在类图中通过版型来表示<<interface>>，下面的介绍将主要介绍类，接口和类类似。

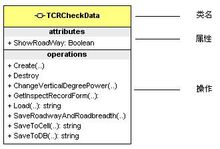
目录

1. 1 [类的UML表示](http://baike.baidu.com/link?url=v6gZ8FH5oAiH4AKWDBdEPkCqSz9D1rg4b8zDESgpwO1guJzVXDta9IE21n19vc8sCA4Ymu0rS-1KoaZbxskEBq#1)
2. 2 [类之间的关系](http://baike.baidu.com/link?url=v6gZ8FH5oAiH4AKWDBdEPkCqSz9D1rg4b8zDESgpwO1guJzVXDta9IE21n19vc8sCA4Ymu0rS-1KoaZbxskEBq#2)
3. 3 [建立类图](http://baike.baidu.com/link?url=v6gZ8FH5oAiH4AKWDBdEPkCqSz9D1rg4b8zDESgpwO1guJzVXDta9IE21n19vc8sCA4Ymu0rS-1KoaZbxskEBq#3)
4. 4 [使用类图](http://baike.baidu.com/link?url=v6gZ8FH5oAiH4AKWDBdEPkCqSz9D1rg4b8zDESgpwO1guJzVXDta9IE21n19vc8sCA4Ymu0rS-1KoaZbxskEBq#4)

类的UML表示

[编辑](javascript:;)

类的命名尽量应用领域中的术语，应明确、无岐义，以利于相互交流和理解。类的属性、操作中的可见性使用+、#、－分别表示public、protected、private。

[](http://baike.baidu.com/pic/UML%E7%B1%BB%E5%9B%BE/6842152/0/03087bf40ad162d986a4e73010dfa9ec8a13cd50?fr=lemma&ct=single)

类之间的关系

[编辑](javascript:;)

类之间的关系是类图中比较复杂的内容。有关联、聚合、组合、泛化、依赖。

关联：是模型元素之间的一种语义联系，是类之间的一种很弱的联系。关联可以有方向，可以是单向关联，也可以是双向关联。可以给关联加上关联名来描述关联的作用。关联两端的类也可以以某种角色参与关联，角色可以具有多重性，表示可以有多少个对象参与关联。可以通过关联类进一步描述关联的属性、操作以及其他信息。关联类通过一条虚线与关联连接。对于关联可以加上一些约束，以加强关联的含义。如下图所示：

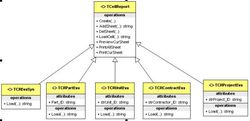
[](http://baike.baidu.com/pic/UML%E7%B1%BB%E5%9B%BE/6842152/0/42166d224f4a20a4cf86b9eb91529822720ed006?fr=lemma&ct=single)

聚合是一种特殊的关联，聚合表示整体与部分的关系。通常在定义一个整体类后，再去分析这个整体类的组成结构。从而找出一些组成类，该整体类和组成类之间就形成了聚合关系。例如舰队是由一系列的舰船组成。需求描述中“包含”、“组成”、“分为….部分”等词常意味着聚合关系。

组合也是一种特殊的关联，也表示类之间整体和部分的关系，但是组合关系中部分和整体具有统一的生存期。一旦整体对象不存在，部分对象也将不存在。部分对象与整体对象之间具有共生死的关系。

聚合和组合的区别：聚合关系是“has－a”关系，组合关系是“contains－a”关系；聚合关系表示整体与部分的关系比较弱，而组合比较强；聚合关系中代表部分事物的对象与代表聚合事物的对象的生存期无关，一旦删除了聚合对象不一定就删除了代表部分事物的对象。组合中一旦删除了组合对象，同时也就删除了代表部分事物的对象。

泛化定义了一般元素和特殊元素之间的分类关系，类之间的这种泛化关系也就是继承关系。泛化关系是“a－kind－of”关系，定义一般元素和特殊元素之间的分类关系。下图是一个泛化关系的例子。

[](http://baike.baidu.com/pic/UML%E7%B1%BB%E5%9B%BE/6842152/0/d31b0ef41bd5ad6ef863e1c680cb39dbb6fd3c07?fr=lemma&ct=single)

有两个元素如果修改X的定义可能会导致对Y的定义，则认为Y依赖X。依赖关系可能由各种原因引起，如一个类向另一个类发送消息，或者一个类是另一个类的数据成员类型，或者一个类是另一个类的操作的参数类型等。有时依赖关系和关联关系比较难区分。如果类A和类B有关联关系，它们之间必然有依赖关系。如果两个类之间有关联关系时不用再表示出这两个类之间的依赖关系。

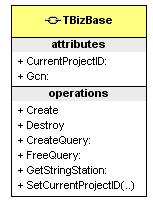
建立类图

[编辑](javascript:;)

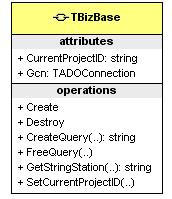
在软件开发不同阶段使用的类图具有不同的抽象层次，即概念层、说明层、和实现层。使用UML进行应用建模也应该是一个迭代的过程，所以我们应该建立一个类图的层次的概念。

概念层类图描述应用领域中的概念，这些概念与实现它们的类有联系。通常没有直接的映射关系。画概念层类图时很少考虑或不考虑实现问题，因此概念层类图应独立于具体的编程语言。下面是一个概念层类的表示。

说明层类图。此时我们考察的是类的接口部分，而不是实现部分。这个接口可能因为实现环境、运行特性等有多种不同的实现。下面是一个说明层类的表示。

[](http://baike.baidu.com/pic/UML%E7%B1%BB%E5%9B%BE/6842152/0/960a304e251f95cad60f179ac8177f3e67095227?fr=lemma&ct=single)

实现层类图才真正考虑类的实现问题，提供实现的细节。此时的类的概念才应该是真正的严格意义上的类。它揭示了软件实体的构成情况。实现层的类是最常用的，在很多的时候说明层的类更有助于人们对软件的理解。

[](http://baike.baidu.com/pic/UML%E7%B1%BB%E5%9B%BE/6842152/0/43a7d933c895d14397f9610f72f082025aaf070a?fr=lemma&ct=single)

UML的最终目标是识别出所有必须的类，并分析这些类之间的关系，类的识别贯穿于整个建模过程，分析阶段主要识别问题域相关的类，在设计阶段需要加入一些反映设计思想、方法的类以及实现问题域所需要的类，在编码实现阶段，因为语言的特点，可能需要加入一些其他的类。

建立类图的步骤：

（1）研究分析问题领域确定系统需求。

（2）确定类，明确类的含义和职责、确定属性和操作。

（3）确定类之间的关系。

类的识别是一个需要大量技巧的工作，寻找类的一些技巧包括：名词识别法；根据用例描述确定类；使用CRC分析法；根据边界类、控制类、实体类的划分来帮助分析系统中的类；参考设计模式确定类；对领域进行分析或利用已有领域分析结果得到类；利用RUP中如何在分析和设计中寻找类的步骤。

1. 名词识别法：

这种方法的关键是识别系统问题域中的实体。对系统进行描述，描述应该使用问题域中的概念和命名，从系统描述中标识名词及名词短语，其中的名词往往可以标识为对象，复数名词往往可以标识为类。

2. 从用例中识别类：

用例图实质上是一种系统描述的形式，自然可以根据用例描述来识别类。针对各个用例，可以提如下的问题辅助识别：

用例描述中出现了那些实体？

用例的完成需要哪些实体合作？

用例执行过程中会产生并存储哪些信息？

用例要求与之关联的每个角色的输入是什么？

用例反馈与之关联的每个角色的输出是什么？

用例需要操作哪些硬设备？

在面向对象应用中，类之间传递的信息数据要么可以映射到发送方的某些属性，要么该信息数据本身就是一个对象。综合不同的用例识别结果，就可以得到整个系统的类，在类的基础上，我们又可以分析用例的动态特性来对用例进行动态行为建模。

3. 使用CRC分析法：

CRC（Class,Responsibilities,Collaboration）卡的最大价值在于把人们从思考过程模式中脱离出来，更充分的专注于对象技术。CRC卡允许整个项目组对设计做出贡献。参与系统设计的人越多，能够收集到的好主意也就越多。因为CRC会议是大家全力参与的，通常只需要很少的有类名的卡片，实际上没有写出完整的卡片。CRC会议进行中，一些人模拟系统和对象交流，把消息传给其他的对象。通过一步步处理，问题很容易地被解决。它由三部分组成：类(Class)、职责(Responsibility)、协作(Collaborator)。下面是一个CRC卡的示例：

|  |  |
| --- | --- |
| 类名 | |
| 职责1 | 职责1的协作 |
| 职责2 | 职责2的协作 |
| …… | …… |

职责是类需要知道或做的任何事物。这些职责是类自身所知的知识，或类在执行时所需的知识。协作是指为获取消息，或协助执行活动的其他类。创建CRC模型需要下面的步骤。

1) 建立团队，包括客户、设计人员、分析人员和一个导引者。如果没有那么多人，那么可以是客户和你自己两个人。

2) 找出需求中存在的名词和名词词组，特别注意复数（通常是集合），他们对应的单数才是。把你第一次想到的所有概念都写在白板或纸上。不管看起来这些概念是如何荒谬，把他们都写下来。

3) 筛选。把对象分为三类，核心对象（必须首先实现），可选的（目前不能确定），以及不需要的对象。这之前最好确定一下你的项目范围。某些不属于本项目范围的对象可以使用轻量的adapter或proxy实现。这里可以加入对分析、设计模式的考虑和应用。

4) 建卡。取出CRC卡，把核心类写在每一张卡上，把可选的类和排除的类分别写在不同的纸上。

5) 角色扮演。最好是一个团队执行，一个人很难做。每个人负责几个类。对每一个Use case其中的情景。导引者指定从某一个人的类开始，某一个人看一看自己能够独立完成，如果不能完成，大家看一看手中的类，谁能完成，就站起来，宣布自己能够完成，以致继续这个过程，每个人完成自己的职责就坐下。在这过程中不断修改类的责任，并写下协作者的名字。

4. 根据边界类、控制类、实体类帮助分析系统中的类

UML中类有三种主要的版型：边界类、控制类和实体类。引入边界类、控制类及实体类的概念有助于分析和设计人员确定系统中的类。

边界类位于系统与外界的交界处，窗体、报表、以及表示通讯协议的类、直接与外部设备交互的类、直接与外部系统交互的类等都是边界类。通过用例图可以确定需要的边界类，每个Actor/Use Case对至少要一个边界类，但并非每个Actor/Use Case对要唯一的边界类。

实体类保存要放进持久存储体的信息。持久存储体就是数据库、文件等可以永久存储数据的介质。实体类可以通过事件流和交互图发现。通常每个实体类在数据库中有相应的表，实体类中的属性对应数据库表中的字段。

控制类是控制其他类工作的类。每个用例通常有一个控制类，控制用例中的事件顺序，控制类也可以在多个用例间共用。其他类并不向控制类发送很多消息，而是由控制类发出很多消息。

5. 领域进行分析

建立类图的过程就是对领域及其解决方案的分析和设计过程。类的获取是一个依赖个人创造力的过程，有时需要和领域专家合作，对研究领域进行仔细分析，抽象出领域中的概念，定义其含义及相互关系，分析出系统类，并用领域中的术语为类命名。领域分析是：通过对某一领域中的已有应用系统、理论、技术、开发历史等的研究，来标识、收集、组织、分析和表示领域模型及软件体系结构的过程，并得到结果。

使用类图

[编辑](javascript:;)

类图几乎是所有面向对象方法的支柱，应该如何使用类图呢？以下提供了一些使用类图的一些建议。

不要试图在项目的初始阶段使用所有的符号，首先应该从简单概念开始。比如类的关系等等，在需要的时候才使用。在项目的不同开发阶段，应该使用不同的观点来画类图。如果处于分析阶段应该画概念层类图，当开始着手软件设计时，应该画说明层类图，当针对某个特定的技术实现时应该画实现层类图。不要为每个事物都画一个模型，应该把精力放在关键的领域。使用类图的最大危险是过早的陷入实现的细节，为了避免这个问题，应该将重点放在概念层和说明层。