描述

统一建模语言（英语：Unified Modeling Language，缩写**UML**）是非专利的第三代建模和规约语言。UML是一种开放的方法，用于说明、可视化、构建和编写一个正在开发的、面向对象的、软件密集系统的制品的开放方法。UML展现了一系列最佳工程实践，这些最佳实践在对大规模，复杂系统进行建模方面，特别是在软件架构层次已经被验证有效

概念

什么是 UML

UML 是 OMG 在1997年1月提出了创建由对象管理组和 UML1.0 规范草案；
UML 是一种为面向对象开发系统的产品进行说明、可视化、和编制文档的标准语言；
UML 作为一种模型语言，它使开发人员专注于建立产品的模型和结构，而不是选用什么程序语言和算法实现；
UML 是不同于其他常见的编程语言，如 C + +，Java中，COBOL 等，它是一种绘画语言，用来做软件蓝图；
UML 不是一种编程语言，但工具可用于生成各种语言的代码中使用 UML 图；
UML 可以用来建模非软件系统的处理流程，以及像在一个制造单元等.

UML 的目标

UML 的目标是定义一些通用的建模语言并对这些建模语言做出简单的说明，这样可以让建模者理解与使用。UML 也是为普通人和有兴趣的人而开发的系统，它可以是一个软件或者使用非软件，它必须是明确的。我们不将 UML 作为一个开发方法，而是随着流程做一个成功的系统。
现在我们可以明确的了解 UML 的目标就是 UML 被定义为一个简单的建模机制，帮助我们按照实际情况或者按照我们需要的样式对系统进行可视化；提供一种详细说明系统的结构或行为的方法；给出一个指导系统构造的模板；对我们所做出的决策进行文档化。

UML 概念模型

对于 UML 的概念模型，我们有以下的理解：

- 概念模型可以被定义为模型，它是由概念和它们之间的关系组成的。
- 概念模型是在绘制 UML 图之前，它帮助了解在现实世界中的各个实体，以及他们如何互相交流。
  UML 描述的实时系统，这是非常重要的一个概念模型。
  掌握 UML 概念模型可以通过学习以下三大要素达到：
- UML 构建模块
- 规则连接构建模块

- UML 公共机制

UML 面向对象的概念

面向对象 (Object Oriented,OO) 是软件开发方法，面向对象的概念和应用已超越了程序设计和软件开发。我们可以将 UML 描述为面向对象的分析和设计的继任者。

一个对象中包含了数据和控制数据的方法，其中数据表示对象的状态，类描述的对象，他们也形成层次结构模型真实世界的系统。表示为继承层次结构，也可以以不同的方式按要求相关的类。

对象是现实世界的实体存在我们周围像抽象，封装，继承，多态的基本概念，都可以使用 UML 表示。因此，UML 是强大到足以代表所有的概念存在于面向对象的分析和设计。

UML 图是面向对象的概念的表示，因此，学习 UML 之前，详细了解面向对象的概念就变得非常重要。

以下是一些面向对象基本概念：

- **对象:** 对象代表一个实体的基本构建块.
- **类:** 类是对象的蓝图.
- **抽象化:** 抽象代表现实世界中实体的行为.
- **封装:** 封装是将数据绑定在一起，并隐藏他们外部世界的机制。
- **继承:** 继承是从现有的机制作出新的类。
- **多态性:** 定义的机制来以不同的形式存在.

面向对象的分析与设计

调查可以被定义为面向对象的分析，更具体地，它是调查对象。设计是指确定对象的协作。
所以重要的是要了解面向对象的分析和设计理念。现在，面向对象的分析的最重要的目的是要设计一个系统来识别对象。这一分析也做了为现有的系统。现在，一种有效的分析是唯一可能的，当我们能够开始思考对象可以识别的方式。确定对象后，确定它们之间的关系，并最终产生的设计。
因此，面向对象的分析与设计的目的可以描述为：

- 确定一个系统中的对象.
- 确定它们之间的关系.
- 做一个设计，使用面向对象的语言可以转换为可执行文件.

```
OO Analysis --> OO Design --> OO implementation using OO languages
```

以上三点可以详细描述：

- 在面向对象的分析，最重要的目的是确定对象和描述他们以适当的方式。如果这些对象的有效识别，那么接下来的设计工作是很容易的。对象应确定职责。职责是对象所执行的功能。每一个对象具有某种类型的要执行的责任。当这些责任协作系统的目的达成。
- 第二阶段是面向对象的设计。在这个阶段的重点是要求及其履行情况。在这一阶段中的对象根据其预期的关联协作。协作完成设计也完成了。
- 第三阶段是面向对象的执行。在这个阶段，设计采用面向对象语言，如 Java，C++ 等。

UML 在面向对象设计中的作用

UML 是一种建模语言，用于示范性软件和非软件系统。虽然 UML 用于非软件系统，重点是面向对象的软件应用建模。大多数的 UML 图到目前为止讨论的用于模拟静态，动态等不同的方面，如现在各方面的构件是对象。

如果我们观察到类图，对象图，协作图，交互图，将基本上基于对象的设计。

因此，面向对象的设计和 UML 之间的关系是非常重要的理解。根据要求，面向对象的设计转化为 UML 图。在详细了解 UML 的面向对象的概念应该学会正确。的面向对象的分析与设计完成后，下一步是很容易的。从面向对象的分析与设计的输入是输入的UML 图。

UML 基本元素

- 三个基本模块：事务，关系，图。
  - 四种事务
  1. 结构事务：类，接口，协作，用例，活动类，组件，节点。
  2. 行为事务：交互，状态机。
  3. 分组事务：包
  4. 注释事务：注释。
  - 四种关系
  1. 依赖
  2. 关联
  3. 实现
  4. 泛化
  - 十种图
  1. 用例图
  2. 类图
  3. 对象图
  4. 包图
  5. 部署图
  6. 活动图
  7. 状态图
  8. 序列图

9. 协作图
10. 组件图

快速学习指南

UML 注释

UML 中最重要的建模元素是符号。
适当有效地使用符号对于一个完整的，有意义的模型来说是非常重要的。如果一个模型的目的无法正确的描绘，那么该模型是无用的。
因此，在开始学习 UML 的时候就要强调表示法的重要性，不同的符号可用于表示物件和关系。
可扩展性是 UML 的另一个重要的特点，这使得UML更加强大和灵活。

UML 核心

UML 的核心是图表，大致可以将这些图归类为结构图和行为图。

- 结构图是由像静态图，如类图，对象图等静态图；
- 行为图是由像序列图，协作图等动态图；
  一个系统的静态和动态特性是通过使用这些图的可视化。

UML 类图

类图是使用面向对象的社会最流行的 UML 图。它描述了在一个系统中的对象和他们的关系，能够让我们在正确编写代码以前对系统有一个全面的认识。
一个单独的类图描述系统的一个具体方面，收集类图表示整个系统。基本上，类图表示系统的静态视图。
类图是唯一可以直接映射到面向对象的语言UML图。因此，它被广泛应用于开发者社区。

UML 对象图

对象图(Object Diagram)描述的是参与交互的各个对象在交互过程中某一时刻的状态。对象图可以被看作是类图在某一时刻的实例。
在UML中，对象图使用的是与类图相同的符号和关系，因为对象就是类的实例。

UML 组件图

组件图是一种特殊的UML图来描述系统的静态实现视图。组件图包括物理组件，如库，档案，文件夹等。
此图是用来从实施的角度。使用一个以上的元件图来表示整个系统。正向和逆向工程技术的使用，使可执行文件组件图。

UML 部署图

组件图是用来描述一个系统的静态部署视图。这些图主要用于系统工程师。
部署图是由节点和它们之间的关系。一个高效的部署图是应用软件开发的一个组成部分。

UML 用例图

用例图是从用户角度描述系统功能，并指出各功能的操作者，用来捕捉系统的动态性质。
一个高层次的设计用例图是用来捕捉系统的要求，因此它代表系统的功能和流向。虽然用例图的正向和反向工程是不是一个很好的选择，但他们仍然在一个稍微不同的方法来模拟它。

UML 交互图

交互图，用于捕获系统的动态性质。
交互图包括序列图和协作图，其中：序列图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互；协作图描述对象间的协作关系，协作图跟时序图相似，显示对象间的动态合作关系。

UML 状态图

状态图是一个用于模拟系统的动态性质的五个图。这些图用来模拟一个对象的整个生命周期。
一个对象的状态被定义为对象所在的条件下，特定的时间和对象移动对其他状态，在某些事件发生时。状态图还用于正向和反向工程。
状态图着重描述从一个状态到另一个状态的流程，主要有外部事件的参与。

UML 活动图

活动图是 UML 的动态模型的一种图形，一般用来描述相关用例图，活动图是一种特殊的状态图。
准确的活动图定义：活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动。活动图是一种特殊的状态图，它对于系统的功能建模特别重要，强调对象间的控制流程。

UML构建模块

事物

事物是实体抽象化的最终结果，是 UML 构建块最重要的组成部分，事物的分类如下:

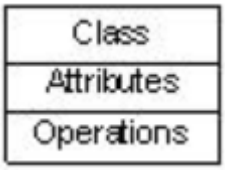
- 结构事物
- 行为事物
- 分组事物
- 注释事物

结构事物

结构事物是模型中的静态部分，用以呈现概念或实体的表现元素，是软件建模中最常见的元素，接下来是对结构化物件的简要描述：

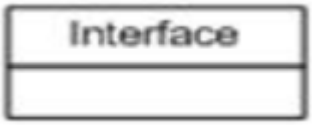
类（Class）

类是指具有相同属性、方法、关系和语义的对象的集合；



接口（Interface）

接口是指类或组件所提供的服务（操作），描述了类或组件对外可见的动作；



协作（Collaboration）

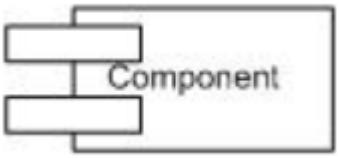
协作定义元素之间的相互作用；

用例（Use Case）

用例定义了执行者（在系统外部和系统交互的人）和被考虑的系统之间的交互来实现的一个业务目标；

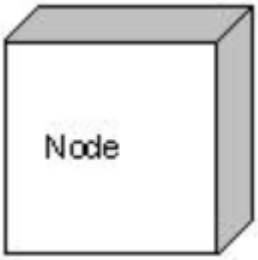
组件（Component）

组件描述物理系统的一部分；



节点（Node）

一个节点可以被定义为在运行时存在的物理元素；

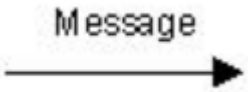


行为事物

行为事物指的是 UML 模型中的动态部分，代表语句里的 "动词"，表示模型里随着时空不断变化的部分，包含两类：

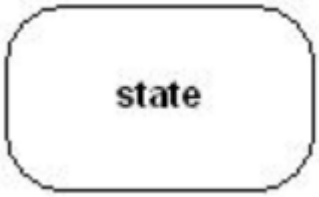
交互

交互被定义为一种行为，包括一组元素之间的消息交换来完成特定的任务。



状态机

状态机由一系列对象的状态组成，它是有用的，一个对象在其生命周期的状态是很重要的。

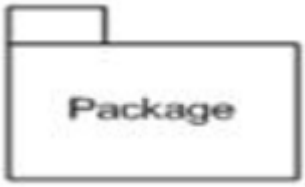


分组事物

可以把分组事物看成是一个"盒子"，模型可以在其中被分解。目前只有一种分组事物，即包（package）。结构事物、动作事物甚至分组事物都有可能放在一个包中。包纯粹是概念上的，只存在于开发阶段，而组件在运行时存在。

包（Package）

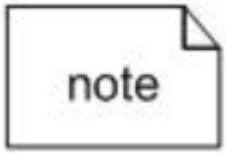
封装是唯一一个分组事物可收集结构和行为的东西。



注释事物

注释事物可以被定义为一种机制来捕捉UML模型元素的言论，说明和注释。注释是唯一一个注释事物。

注释（Note）
注释用于渲染意见，约束等的UML元素。

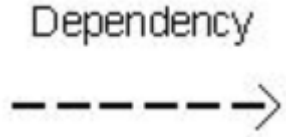


关系

关系是另一个最重要的构建块UML，它显示元素是如何彼此相关联，此关联描述的一个应用程序的功能，UML中定义了四种关系：

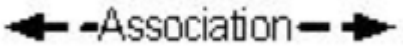
依赖

依赖是两件事物之间的语义联系，其中一个事物的变化也影响到另一个事物。



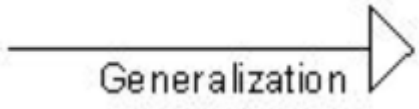
协作

一种描述一组对象之间连接的结构关系，如聚合关系（描述了整体和部分间的结构关系）；



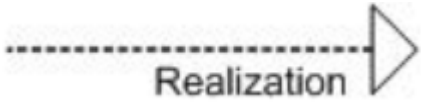
泛化

泛化可以被定义为一个专门的元件连接关系与一个广义的元素，它基本上描述了在对象世界中的继承关系，是一种一般化-特殊化的关系；



实现

类之间的语义关系，其中的一个类指定了由另一个类保证执行的契约。



UML 图

UML 图的整个讨论的最终输出所有要素，关系用于使一个完整的UML图，图中表示的系统。
UML 图的视觉效果是整个过程中最重要的部分。

图是事物集合的分类，UML 中包含多种图：

1. 类图：类图描述系统所包含的类、类的内部结构及类之间的关系；
2. 对象图：对象图是类图的一个具体实例；
3. 用例图：用例图从用户的角度出发描述系统的功能、需求，展示系统外部的各类角色与系统内部的各种用例之间的关系；
4. 顺序图：顺序图表示对象之间动态合作的关系；
5. 协作图：协作图描述对象之间的协作关系；
6. 活动图：活动图描述系统中各种活动的执行顺序。
7. 状态图：状态图描述一类对象的所有可能的状态以及事件发生时状态的转移条件；
8. 部署关系图：部署关系图定义系统中软硬件的物理体系结构；
9. 组件图：组件图描述代码部件的物理结构以及各部件之间的依赖关系；