## 各种输入输出:

1. 需求分析

* 产出:问题描述,用例模型图,术语表,补充说明

1. 分析设计:

* 输入:用例模型,术语表,补充说明
* 输出:设计模型,结构文档,数据模型说明

1. 架构分析

* 输入:用例模型,词汇表,补充说明,项目详细指南,参考框架
* 输出:部署模型,设计模式,软件架构文档

1. 用例分析

* 输入:用例模型,词汇表,补充说明,项目详细指南,软件架构文档
* 输出:用例实现,分析模型,分析类

1. 确定设计元素

* 输入:分析模型,补充说明,如那件架构文档,项目详细细节,分析表
* 输出:设计模型

1. 确定设计机制

* 补充说明,分析表
* 软件架构文档,设计模式

## UML图:

UML图一共有10种:

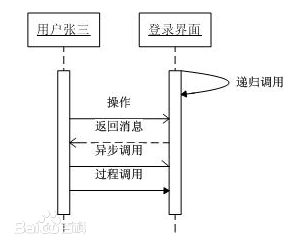
|  |  |  |
| --- | --- | --- |
| 1 |  | 用例图  描述角色以及角色与用例之间的连接关系。说明的是谁要使用系统，以及他们使用该系统可以做些什么。一个用例图包含了多个模型元素，如系统、参与者和用例，并且显示了这些元素之间的各种关系，如泛化、关联和依赖。 |
|  |  | 类图  类图是描述系统中的类，以及各个类之间的关系的静态视图。能够让我们在正确编写代码以前对系统有一个全面的认识。类图是一种模型类型，确切的说，是一种静态模型类型。类图表示类、接口和它们之间的协作关系。 |
| 3 |  | 对象图  与类图极为相似，它是类图的实例，对象图显示类的多个对象实例，而不是实际的类。它描述的不是类之间的关系，而是对象之间的关系 |
| 4 |  | 包图  用于描述系统的分层结构，由包或类组成，表示包与包之间的关系。  包里面可以是任何一种UML图 |
| 5 |  | 活动图  描述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。能够演示出系统中哪些地方存在功能，以及这些功能和系统中其他组件的功能如何共同满足前面使用用例图建模的商务需求。  活动图可以使用泳道来区分活动的发出者.也可以不使用泳道. |
| 6 |  | 状态图  描述类的对象所有可能的状态，以及事件发生时状态的转移条件。状态图是对类图的补充。  状态图画的是一个状态机. |
| 7 | https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/c0%3Dbaike72%2C5%2C5%2C72%2C24/sign=ec2dc7d8c3fcc3cea0cdc161f32cbded/503d269759ee3d6dfec6cc3148166d224e4adecc.jpg | 序列图  序列图是用来显示你的参与者如何以一系列顺序的步骤与系统的对象交互的模型。顺序图可以用来展示对象之间是如何进行交互的。上面是对象,最左端一般是操作者,然后是用户界面,最后面是数据库,返回消息是虚线,操作是实线. |
| 8 |  | 协作图  和序列图相似，显示对象间的动态合作关系。可以看成是类图和顺序图的交集，协作图建模对象或者角色，以及它们彼此之间是如何通信的。如果强调时间和顺序，则使用序列图；如果强调上下级关系，则选择协作图；这两种图合称为交互图。 |
| 9 |  | 构件图  描述代码构件的物理结构以及各种构建之间的依赖关系。用来建模软件的组件及其相互之间的关系，这些图由构件标记符和构件之间的关系构成。在组件图中，构件是软件单个组成部分，它可以是一个文件，产品、可执行文件和脚本等 |
| 10 |  | 部署图  是用来建模系统的物理部署。例如计算机和设备，以及它们之间是如何连接的。部署图的使用者是开发人员、系统集成人员和测试人员。部署图用于表示一组物理结点的集合及结点间的相互关系，从而建立了系统物理层面的模型。 |

## 重点掌握类图和序列图

1. 类图

|  |  |  |  |
| --- | --- | --- | --- |
|  | 继承:  子类指向父类,  实线空三角 |  | 实现:  实现类指向接口  虚线空三角 |
|  | 关联关系  单项关联单箭头,双向双箭头或无箭头,可以标对应数量.聚合和组合都是关联关系,(在类中拥有另外类类型的属性) |  | 聚合关系:  聚合和关联从语法上无法区分,聚合是强关联,实心箭头空心菱形箭尾,由外层指向内层元素 |
|  | 组合:  组合是比聚合更强的关联,部分与整体,不分不能离开整体独立存在,实线实心箭头实心菱形箭尾,由整体指向部分 |  | 依赖:  类E使用了类D,但是在类E中没有D类型的属性,但是E访问了D的方法或属性,或者实例化过D. |

1. 序列图



## 4+1视图:

逻辑视图（Logical View），设计的对象模型（使用面向对象的设计方法时）。

过程视图（Process View），捕捉设计的并发和同步特征。

物理视图（Physical View），描述了软件到硬件的映射，反映了分布式特性。

开发视图（Development View），描述了在开发环境中软件的静态组织结构。

第五个视图一般称为场景视图,表示架构的描述，即所做的各种决定，可以围绕着这四个视图来组织，然后由一些用例 （use cases）或场景(scenarios)来说明，从而形成了第五个视图。