UML 作为软件工程建模阶段的统一标准，已经有四十多年的历史，是经过相关专家学者不断地完善和改进而形成的第三代建模语言。在一个软件工程项目中，软件的设计阶段所生成的UML 文档几乎涵盖了该软件设计的全部信息，利用 UML 能够把工程的各种细节和部署描述得非常完整。经过多年的发展和完善，UML 已经展示了它在建模方面的优势和对需求分析阶段信息的反馈能力，无数次的工程实践可以证明 UML 可以对大型复杂软件系统的层次区分和构架基础进行优良建模。UML 作为一种表达能力和普适性都很强的语言，已经被多种建模工具支持。针对UML 向程序语言比如 C、C++或 java 高级语言转化的软件已经不在少数，UML 能够适应当前绝大多数的软件设计过程，同时对于面向对象设计流程中的模块化思想也能够完美表现。UML的特点可以总结为以下几个方面。

（1）对模型种类的适应性非常强大，可以支持构建任何特性的模型，即是说其中包含了对任何结构的设计，无论设计者有何种类型的构思和想法，只要对UML 规则有所了解，就可以通过该语言进行建模。

（2）对设计者来说没有约束，UML 对设计者需要构建何种类型的模型和需要反馈软件中何种方面的信息没有规定。设计者完全可以按照自己的意愿进行设计，并根据自己的设计选择不同的 UML 图例进行表达

（3）UML 的一大优点是可视化，同时其图示具有易于观察的特点，通过观察 UML 模型图，人们可以非常的了解设计者的意图及其设计的特点所在。

（4）UML 便于记录，目前支持 UML 的软件种类非常多，rational rose、Star UML、argo UML 等，既方便设计者进行软件设计，也方便需求方对设计进行了解。同时，每当设计者产生新的想法，只要在编码阶段之前就可以对当前 UML文档进行更新，具有简易便捷的特点。

（5）UML 作为软件过程中的应用语言，对于程序员也是非常友好的，编码阶段如果有新的工作人员进入，其可以通过观察 UML 文档而大大缩减工作的熟悉阶段。

2.2 UML 模型图分类

UML 建模总体来说分为动态建模和静态建模两类，但从模型图的区别上来说又可以细分为 10 种，不同的图可以从不同角度表达系统的属性。

静态 UML图主要保存了系统中的参与对象以及这些各个不同对象所参与的活动及关系，动

态结构则主要保存了系统中的运行流程与活动交互信息，这些图结合在一起就可以完整的从各个角度表达一个系统的属性和状况。下面将介绍 UML 图的十种图示。

（1）用例图：用例图中的元素包括参与者、用例以及他们之间的联系，例如一个售票系统，坐在电脑旁边的售票员和买票窗口外的购票人是参与者，而买票和收费就属于用例，而售票员和用户和收费用例都有联系。用例图一个很重要的功能就是能够让 UML 的读者明白系统中的参与者分几种，这些参与者又主要参加哪类用例活动，对系统功能模块有很好的划分作用。用例图中的主要关系分为四种

1）关联：关联主要指参与者和用例之间的关系，比如售票员和售票，购

票人和买票，其方向为从参与者出发到用例。

2）泛化：泛化主要指同类对象之间的关系，即参与者与参与者或用例跟用例之间的关系。比如公司员工可以泛化为经理和普通员工，买票可以泛化为网络购票和现金购票，而且泛化出来的用例同样包含与其基础用例相关的所有关系，也包括基础用例中的全部信息。

3）包含：包含代表用例的内部功能划分，也代表用例中的一部分。比如整个购票用例包括订票、支付、取票三个部分，则购票和支付之间就是包含关系。子用例不一定包含父用例的所有关系，比如公司员工的用例为工作，工作又包含审核员工和基本工作，其中审核员工由公司员工中的经理参与，一般员工不参与此项用例。

4）扩展：扩展同样是用例之间的关系，但扩展出来的用例相当于一种附加条件，比如购物是用例，而买一赠一为其扩展用例。

用例图作为系统中大部分功能的信息记录，对于系统模块的划分有着重要意义，人们通过对用例图的观察能够在很短的时间内就明白的系统的功能和各部分之间的关系。

（2）状态图：状态图主要表示系统中某个对象或整个系统在运行过程中的各个状态及造成状态发生转移的条件，同时也包含状态的下一步选择分支及在该分支处的判定条件。状态图主要包括初始状态、终止状态、转移警戒条件、转移结果以及子状态结构等。其中初始状态是系统或对象所处的最初状态，满足下一步转移的警卫条件时才会发生变化；子状态则指当前状态内部有更加具体的状态变化。总体来说，状态图蕴含着系统发生变化的判定条件信息和系统的生命周期信息，这对于系统的整体结构建立都有着至关重要的意义。

（3）时序图：时序图主要表示系统流程各个事件中不同对象参与时间的先后关系，同时也包含对象之间的交互信息。在 UML 规范更新至 2.0 版本之后，时序图中添加了多种组合片段，可以表示更多信息。这里介绍几个主要的组合片段：

alt：该组合片段代表转折，即此处带有判定，如果某变量的值符合当前变化条件，则 alt 片段中的对象对发出与原来不同的消息。

loop：该片段表示循环，如果此片段内某条件未成立则该对象一直循环发出消息。

seq：该片段表示顺序执行，即此片段内的消息需存在先来后到的顺序关系，先到的消息会首先得到响应。

par：代表并发，指当前到达的消息可以同时触发两个或多个操作并发执行。

还有其他结构这里不再一一介绍，时序图在本课题的研究中具有十分重要的意义

（4）类图：类图最直接的用处就是可以将其中蕴含的信息直接转化为面向对象高级编程语言中的类定义，实际上很多软件也支持将高级程序语言中的类定义及声明转化为 UML 类图。这充分说明类图在软件设计阶段处于后期设计

（5）活动图：活动图主要用来显示系统运行流程及其在运行过程中的所有节点判断条件。其主要意义是可以在软件业务功能模型中显示工作执行流程。

（6）协作图：协作图在 UML2.0 以后被重新定义名称为通信图，主要用来表示一件复杂事件中不同对象的合作关系，与时序图不同的是它更加注重消息的发出者和接收者之间的组织关系。

（7）组件图：组件图倾向于系统内部的整体关系，比如代码和库文件之间的链接，或者是源代码结构的低耦合性与高内聚性，其最重要意义是反映了设计阶段的代码规划与组织。

（8）部署图：部署图主要表示软件系统在实施阶段的硬件部署情况，同时包含了系统中的软件和硬件结构信息。

（9）包图：包图可以作为系统中的一种组织结构，当系统中类因为数量关系难以表达时，设计者加入包这种组织来进行较为简单的划分。

（10）对象图：和类图十分相似，若作为软件编码前测试用途信息时和类图基本无区别，这里不做赘述。

软件方面的验证计划

软件开发计划

软件验证计划

软件配置管理计划

软件质量保证计划

软件需求标准

软件设计标准

软件代码标准

软件需求数据

设计说明

源代码

可执行目标代码

软件验证例子和程序

软件验证结果

软件生命周期环境配置索引

软件配置索引

问题报告

软件配置管理记录

软件质量保证（SQA）记录

软件完成综述

数据追踪

参数数据项文件