1. **快捷键**

快捷键Ctrl+Shift+P调出VS Code命令行，Ctrl+~ 调出VS Code的终端，Ctrl/⌘+Shift+E 文件资源管理器，Ctrl/⌘+Shift+G 源代码管理，Ctrl/⌘+Shift+F 跨文件搜索，Ctrl/⌘+Shift+D 启动和调试，Ctrl/⌘+Shift+M 查看错误和警告，Ctrl/⌘+Shift+X 管理扩展插件，

1. **需求的四种类型：**

功能性需求:根据需要的活动描述需要的行为

质量需求或非功能需求:描述软件必须具备的一些质量特征

设计约束:设计决策，如平台或接口组件的选择

过程约束:对可用于构建系统的技术或资源的限制

1. **高质量需求的特点**

解决冲突，需求是可测试的，正确的、一致的、无二义性、完整的、可行的、无与主要目标不相关的需求、可描绘的、可测试的

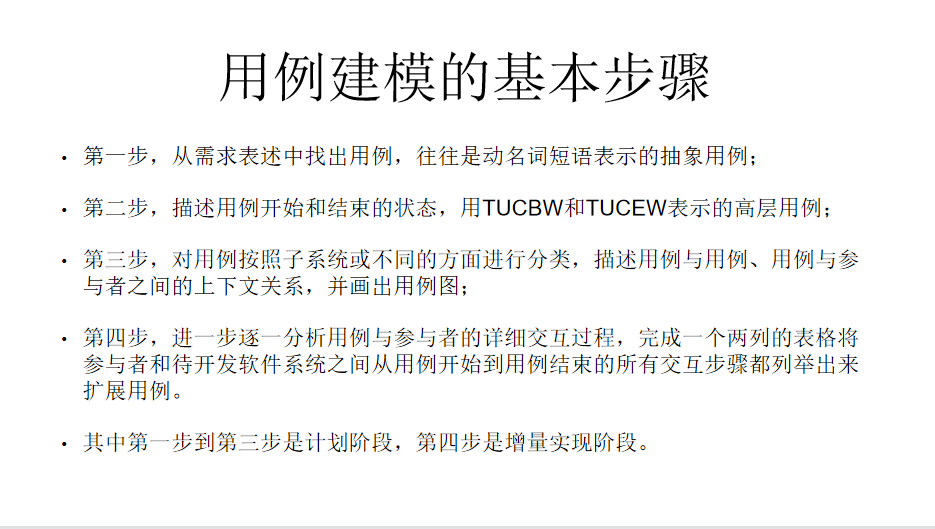
1. **需求分析的两类基本方法**

原型化方法（Prototyping）和建模的方法（Modeling）是整理需求的两类基本方法。

原型化方法：可以很好地整理出用户接口方式（UI，User Interface），比如界面布局和交互操作过程。

建模的方法：可以快速给出有关事件发生顺序或活动同步约束的问题，能够在逻辑上形成模型来整顿繁杂的需求细节。

1. **用例建模的步骤**

****

1. **用例满足的四个必要条件**

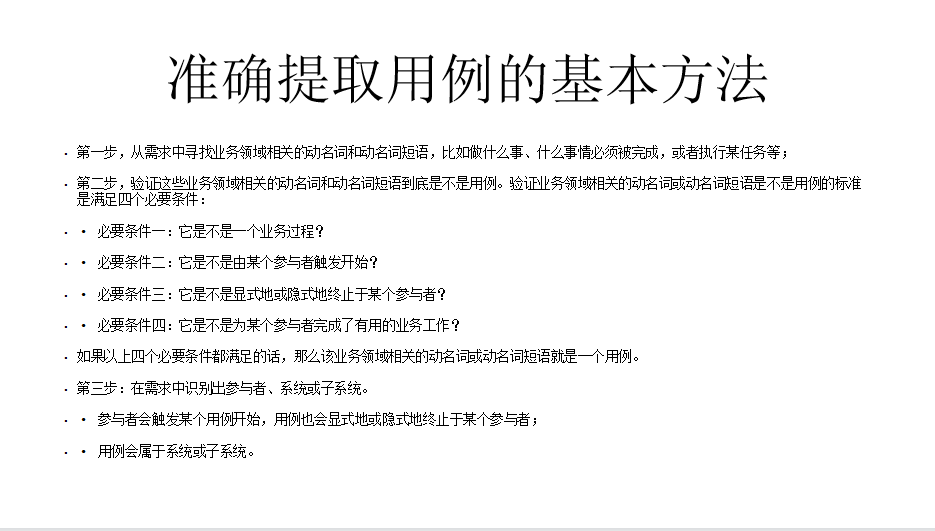
• 必要条件一：它是不是一个业务过程？

• 必要条件二：它是不是由某个参与者触发开始？

• 必要条件三：它是不是显式地或隐式地终止于某个参与者？

• 必要条件四：它是不是为某个参与者完成了有用的业务工作？

1. **准确提取用例的基本方法**

****

1. **统一过程的核心要义是什么**

统一过程（UP，Unified Process）的核心要义是用例驱动（Use case driven）、以架构为中心（Architecture centric）、增量且迭代（Incremental and Iterative）的过程。

1. **模块化的基本原理**

模块化（Modularity）是在软件系统设计时保持系统内各部分相对独立，以便每一个部分可以被独立地进行设计和开发。背后的基本原理是关注点的分离

1. **本地化外部接口是什么含义**

使用本地化外部接口来提高代码的适应能力， 将我们的代码接口分离出来，写成本地化的外部接口，能更好的帮助我们分离业务之间的关联性，使得代码开发更加高效。

1. **接口的5个基本要素**

接口的目的；

接口使用前所需要满足的条件，一般称为前置条件或假定条件；

使用接口的双方遵守的协议规范；

接口使用之后的效果，一般称为后置条件；

接口所隐含的质量属性。

1. 通用接口定义的基本方法

参数化上下文

移除前置条件

简化后置条件

1. **git merge和git merge -no-ff的区别**

git merge –no-ff 可以保存你之前的分支历史。能够更好的查看 merge历史，以及branch 状态。

git merge 则不会显示 feature，只保留单条分支记录。

1. **正则表达式：**

工欲善其事必先利其器的ppt中

1. **依赖倒置原则的含义及其应用价值**

依赖倒置原则的原始定义为：高层模块不应该依赖低层模块，两者都应该依赖其抽象；抽象不应该依赖细节，细节应该依赖抽象（High level modules shouldnot depend upon low level modules.Both should depend upon abstractions.Abstractions should not depend upon details. Details should depend upon abstractions）。其核心思想是：要面向接口编程，不要面向实现编程。应用于：模块化设计

1. **MVC架构为什么更灵活以及MVVM架构为什么更智能**

MVC: 为了包容需求上的变化而导致的用户界面的修改不会影响软件的核心功能代码，可以采用将模型（Model）、视图（View）和控制器（Controller）相分离的思想

MVVM: View和Model之间没有联系，通过ViewModel进行交互，而且Model和ViewModel之间的交互是双向的，因此视图的数据的变化会同时修改数据源，而数据源数据的变化也会立即反应到View上。

1. **软件角度分析MVC到MVVM的发展**

随着前端页面越来越复杂，用户对于交互性要求也越来越高，jQuery是远远不够的，于是MVVM被引入到Javascript前端开发中。前端有了MVVM的设计思想后，终于实现了前后端工程及数据的彻底分离

1. **执行视图、部署视图、分解视图**

执行视图：执行视图展示了系统运行时的时序结构特点，比如流程图、时序图等。执行视图中的每一个执行实体，一般称为组件（Component），都是不同于其他组件的执行实体。如果有相同或相似的执行实体那么就把它们合并成一个。

部署视图：部署视图是将执行实体和计算机资源建立映射关系。这里的执行实体的粒度要与所部署的计算机资源相匹配，比如以进程作为执行实体那么对应的计算机资源就是主机，这时应该描述进程对应主机所组成的网络拓扑结构，这样可以清晰地呈现进程间的网络通信和部署环境的网络结构特点。当然也可以用细粒度的执行实体对应处理器、存储器等。部署视图有助于设计人员分析一个设计的质量属性

分解视图：分解是构建软件架构模型的关键步骤，分解视图也是描述软件架构模型的关键视图，一般分解视图呈现为较为明晰的分解结构（breakdown structure）特点。分解视图用软件模块勾划出系统结构，往往会通过不同抽象层级的软件模块形成层次化的结构。

1. **没有银弹的含义**

工程专家们所找到的各种方法都是舍本逐末，它们解决不了软件中的根本困难，即软件概念结构(conceptual structure)的复杂性，无法达成软件概念的完整性和一致性，自然无法从根本上解决软件危机带来的困境

1. **简述v模型**

V模型也是在瀑布模型基础上发展出来的，我们发现单元测试、集成测试和系统测试是为了在不同层面验证设计，而交付测试则是确认需求是否得到满足。也就是瀑布模型中前后两端的过程活动具有内在的紧密联系，如果将模块化设计的思想拿到软件开发过程活动的组织中来，可以发现通过将瀑布模型前后两端的过程活动结合起来，可以提高过程活动的内聚度，从而改善软件开发效率。

1. **团队强度和项目特点的关系**
2. **简述敏捷宣言的核心思想**

• 个体和互动 高于 流程和工具

• 工作的软件 高于 详尽的文档

• 客户合作 高于 合同谈判

• 响应变化 高于 遵循计划

1. **敏捷开发的宗旨**

互联网使得知识的获取变得更加容易，很多软件可以由一个小团队来实现。同时，技术更新的速度在加快，用户需求的变化也在加快，开发流程必须跟上这些快速变化的节奏。于是敏捷方法就产生了。

1. **性能优先策略带来的隐藏代价**

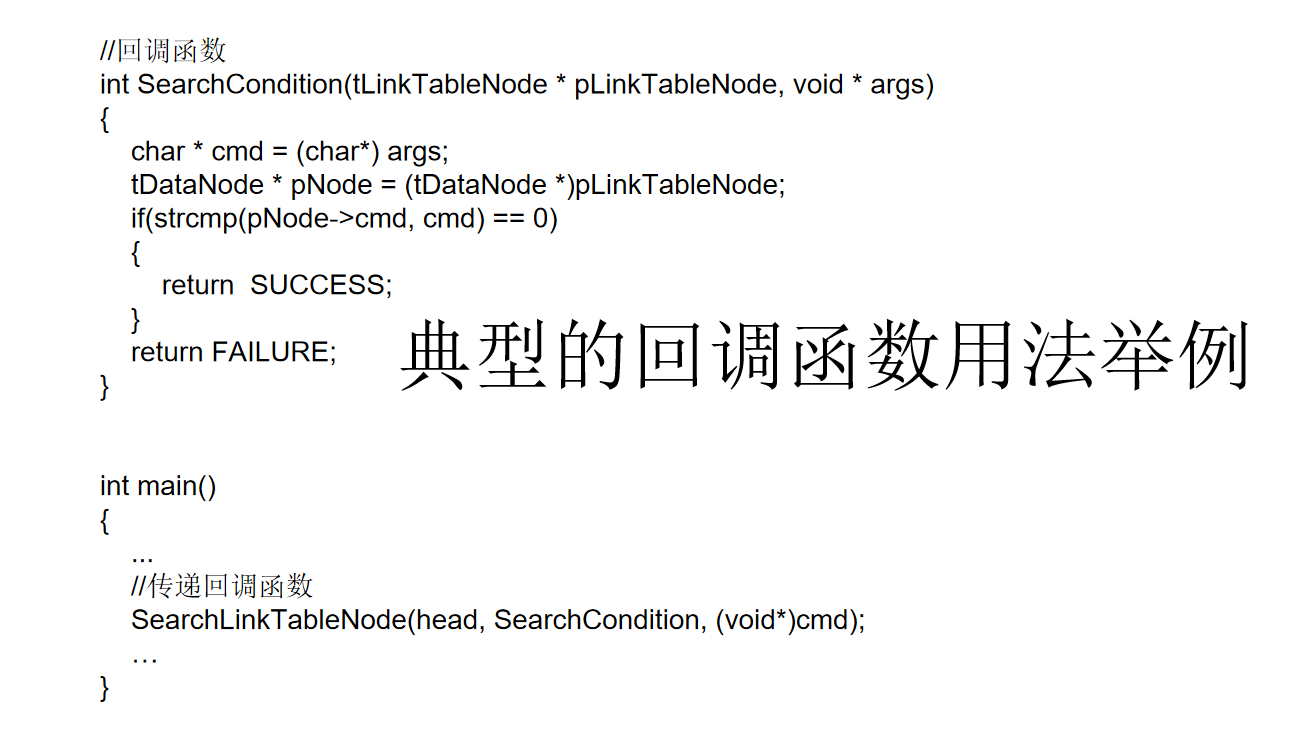
cost to write the code faster。当软件工程师的人力成本远大于所消耗的计算资源成本时，提高代码编写的工作效率将更有价值；

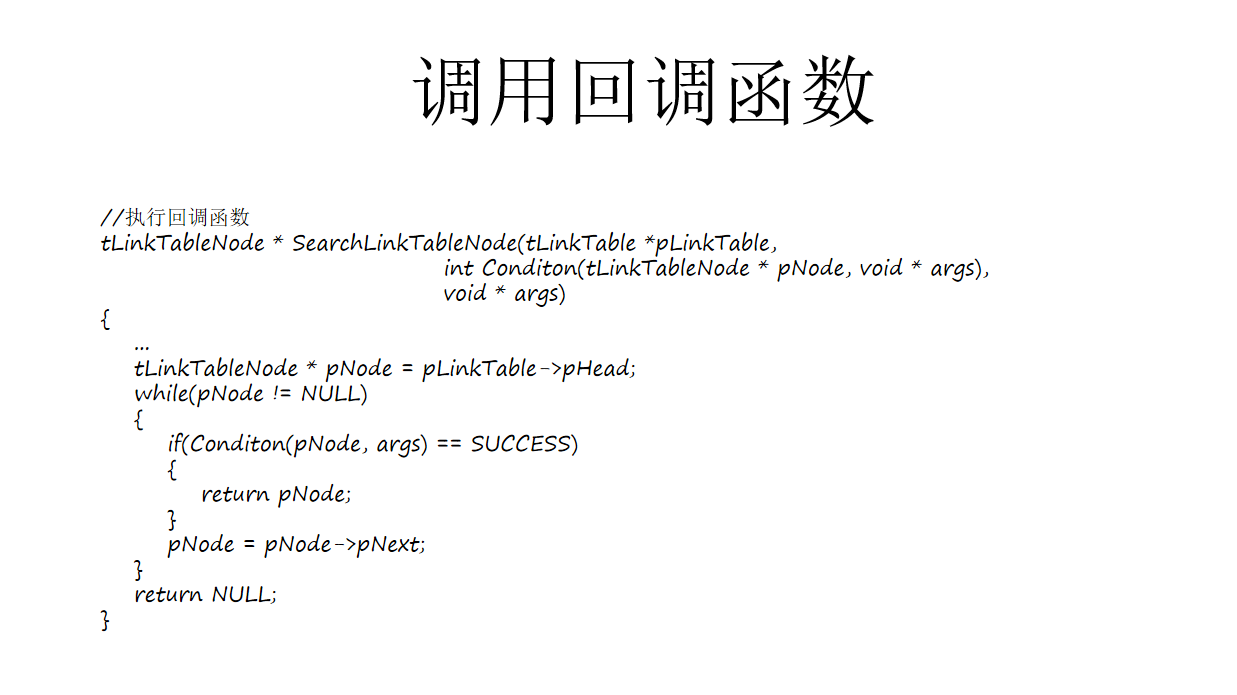
cost to test the code。质量保证的人力成本和质量保证的成效也比所消耗的计算资源成本更有价值；

cost to understand the code。性能优先的策略往往会让代码很难理解，结果需要消耗更多的工时；

cost to modify the code。面向机器的代码修改起来更困难，可扩展性差，同样会消耗更多工时。

1. **编程大题：写一个menu里面自定义查找结点的测试代码**

****

****