

敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

五、敏捷设计

1、问题的提出

为了应对需求变化,设计应尽力避免以下问题:

- (1) **僵化性 (Rigidity)**: 是指难于对软件设计进行改动,即使是简单的改动。例如:如果一个改动会导致有依赖关系的连锁改动,那么设计就是僵化的。期间,必须改动的模块越多,其设计就越僵化。
- (2) **脆弱性 (Fragility)**: 是指在进行了一个改动时,程序的许多地方就可能出现问题的地方,即使出现问题的地方与改动的地方并没有概念上的关联,即设计易于遭受破坏。



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

- (3) **粘固性 (Immobility)**：是指在一部分的设计中包含了对其它部分有用的成分，但要想把这些成分分离出来就要付出很大的努力并具有相当大的风险,即设计难于复用。
- (4) **粘滞性 (Viscosity)**：存在2种表现形式：
- ① **软件粘滞性**：是指当面临一个改动时，若想保持系统一致的设计方法是一件很困难的事情,很易采用一些破坏设计的方法，即难于做正确的事情。
 - ② **环境粘滞性**：是指环境的迟钝和低效。例如编译时间长。
- 也是难于做正确的事情。



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

(5) **不必要的复杂性** (Needless Complexity) : 是指设计中包含了当前没有用的成分, 一方面可使软件变得复杂, 另一方面可使软件难于理解, 即过分设计。

(6) **不必要的复制** (Needless Repetition) : 是指滥用“剪切”和“粘贴”等鼠标操作。

尽管“剪切”和“粘贴”操作也许是有用的文本编辑操作, 但对代码编辑来说, 该操作却是灾难性的:

- ① 往往使开发人员忽略了抽象, 从而使系统不易理解;
- ② 软件中的重复代码, 使系统的改动变得更加困难, 不易系统的维护。



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

(7) **晦涩性 (Opacity)**：是指模块难于理解。并且由于代码随时不断演化，往往会模块变得越来越晦涩。一般来说，若能持续地保持代码是清晰的和富有表现力的，那么就可以减少代码的晦涩性。



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

2、防止软件腐化的基本途径

简言之,以变应变,尽力避免出现以上设计问题。

进一步说:

- (1) 团队几乎不进行预先 (**up-front**) 设计, 因此不需要一个成熟的初始设计;
- (2) 团队通过多次使用单元测试和验收测试, 支持系统的设计尽可能的干净、简单, 使设计保持灵活性和易于理解性;
- (3) 灵活、持续地改进设计, 以便使每次迭代结束时所生成的系统具有满足那次迭代需求的设计。



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法

3、采用敏捷设计的方法 包括：

使用一些设计原则，以保持软件是灵活的、可维护的；
掌握一些**设计模式**，以便针对特定问题权衡这些原则。

模式作为一种对经验的总结，
针对软件开发过程中一些反复出现、具有共性的问题
给出的良好的解决方案

例如：MVC (Model View Controller)
设计模式



敏捷设计

- 设计中应避免的问题
- 防止软件腐化的基本途径
- 敏捷设计方法



3、采用敏捷设计的方法

敏捷设计是一个应用原则、模式和实践的过程，其间不断改善软件结构,保持系统设计在任何时间都尽可能的**简单、干净** (主要是指边界清楚, 结构良好)和**富有表现力**，即：

- ① **它的功能** 对于用户来说，通过直观、简单的界面呈现出恰当特征的程序；
- ② **它的内部结构** 对于软件设计者来说，通过简单、直观的划分，使其具有最小耦合的内部结构；
- ③ **创造过程** 对于开发人员来说，每周都会取得一些重大进展，并生产出无缺陷代码的具有活力的团队过程。

