《软件体系结构与设计》

作业二

班　　级：111171

学　　号：20171000970

学 生 姓 名：董安宁

指 导 教 师：尚建嘎

中国地质大学地理与信息工程学院软件工程系

2019 年 9 月

第 2 章 软件质量属性

1.软件可管理性主要表现在哪些方面，并以一款软件为例分析其可管理性。

可管理型是指查看和修改指定系统或软件状态的能力，它描述了系统管理员管理应用程序的难易程度。软件的可管理性主要表现在四个方面：

① 系统配置：软件能够方便的安装。部署、配置和集成，并提供机制对整个过程进行有效的规划、监督和控制。

② 系统优化：通过调整软件自身参数、属性、行为以适应外界不同的计算机环境和应用需求，是软件功能和效能得到最大的发挥。

③ 系统诊断：软件出现故障或潜在隐患时，能够具备某种手段对问题进行定位，报警。并依照一定的策略在必要时采取措施补救。

④ 系统防护：当软件运行要素被无意的破坏或被恶意的攻击时，软件应能够对期价已识别和采取可能的应对方案，并及时的修复和恢复。

以 QQ 为例

① 系统配置：在官网下载 QQ 的安装程序后，只需要双击 exe 安装程序，就可以进行安装直至完成。安装方便，配置和部署都集成在一个可执行文件中。并且安装过程中可以看到安装的进度和安装文件的信息，起到了监督和控制作用。

② 系统优化：无论是 36 位系统还是 64 位系统，都用的是一个安装包进行安装，可见 QQ 在安装的时候通过收集和分析电脑的配置信息，弹性地微调安装方式，适应了不同计算机环境的需求(仅限 windows)。

③ 系统诊断：有时候会有 QQ 运行到崩溃的时候，该软件会自动退出程序并弹出窗口，发送错误报告，实现了对问题进行定位和报警。

④ 系统防护：QQ 有登录保护，以及异地登录验证等防护手段阻止账号被破坏，虽然该软件没有自动恢复的功能，但是可以去咨询 QQ 安全中心来恢复账号。

2.系统非功能需求是什么含义？与其对应质量属性有哪些？

非功能性需求定义了系统的特性，如性能、可扩展性、易用性等软件质量属

性。非功能性需求所描述的质量属性是软件主要的质量属性，是软件体系结构分析、设计、评估等工作的重要依据。对于非功能性质量从系统设计、实现、运行的实践角度出发进一步划分，从整体上将软件质量属性划分为“功能正确性”，“设计时质量属性”，“运行时质量属性”，“系统质量属性”，“用户质量属性”五大类。更细节的划分如下图所示：

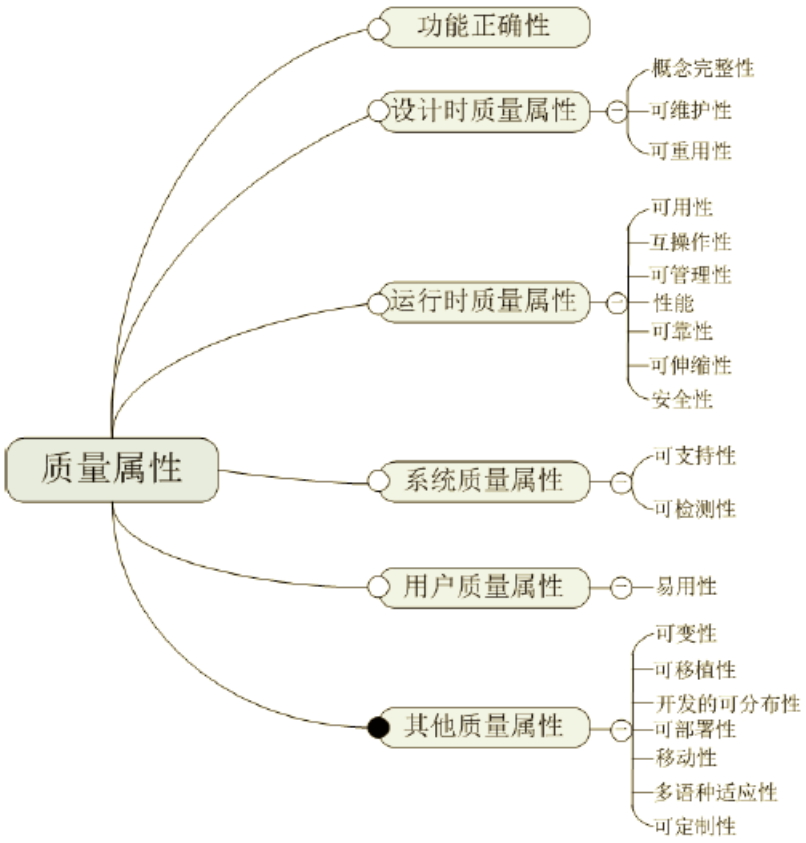

图 1 软件质量属性分类

3.功能需求、质量属性需求、约束分别对软件架构产生哪些影响？

功能性是系统完成预定工作的能力，功能并不决定系统的架构，因为可以有许多种架构来实现一种功能。尽管功能性是独立于任何特定结构的，但功能性是通过将指责分配给不同架构元素来实现的，从而形成系统架构整体结构的基础。

质量属性与功能密切相关，在进行系统软件体系结构设计时，要综合考虑多种功能性需求和非功能性需求，在实际开发中，衡量同类软件好坏的往往是肺功能特性质量。

约束是对于功能性和质量属性之间的限制。架构师对于功能性的考虑终点在于它如何与其他质量属性交互以及如何约束其他的质量属性。

4.为提高系统的设计时质量属性如概念完整性、可维护性、可重用性，您认为可采取哪些设计策略？并请举例说明。

概念完整性：

设计时首先确定关注点；酌情将其分组到设计的不同逻辑层次中；使用一致的开发过程，使用经过验证和测试的开发工具和方法学；在小组之间的沟通上考虑创建统一开发过程，并整合一些工具来推动处理流程、沟通和协作的协调统一；引入一套规范的设计和编码的指导原则与代码审查机制，确保所有人都遵循这些规则；对于遗留的难以重构的系统，应该考虑创建遗留技术的升级之路，并将应用与外部依赖隔离

可维护性：

将系统设计为定义明确的层或关注点，以分离系统的UI、业务过程和数据访问功能，降低组件和层之间的过度依赖；选择合适的通信模型、格式和协议，以及设计师县插件师架构，可以方便地进行升级和维护；尽可能使用拼台内置的功能和特性来提高重用性和可维护性，将组件设计成“高内聚，低耦合”也可以达到上述目的；最后是对文档的书写和整理，有利于将来对老软件的升级。

可重用性：

在设计的时候确定公共功能，在独立的组建中实现这些功能方便重用；对于有相似任务的功能应该使用一个方法来完成，通过参数的变化来实现行为的变化；在设计时尽量考虑通过服务接口从组建、层以及子系统暴露功能，供其它层和系统使用，以及考虑平台无关的数据类型和数据结构，方便跨平台的访问和识别。

5.为提高系统运行时质量属性如可用性、性能、可伸缩性，您认为可采取哪些设计策略？

可用性：

通过一些设计模式帮助更快的捕获错误，避免影响其他功能；设计的时候尽量多考虑可能遇到的情况，尽早完善防范策略。

性能：

优化客户端算法，缩短响应时间；设计正确的缓存策略，减少内存消耗；确保选择有效率的事务类型、锁、线程和队列的实现方式，增加数据库服务器的吞吐量；设计恰当的网络带宽使用、资源管理与数据分配策略。

可伸缩性：

考虑如何设计逻辑层和物理层的关系，以及考虑这样的设计会如何影响应用程序和数据库向上扩展和向外扩展的能力；考虑如何处理流量和负载的高峰；考虑实现存储和转发或是基于缓存消息的信息系统；

6.对于您曾经开发或目前正在开发的系统来说，最重要的质量属性需求是什么？试着使用质量属性场景描述这些需求（给出图、表中的一种）。

曾经帮同学写过一个 c 语言的学生管理系统，我认为在完成功能的正确性的基础上（毕竟作为软件工程的学生这玩意必须得过关啊），性能的可靠性是更重要的，因此我还额外考虑了一些错误使用系统的情况以及对应的解决方案，如下所示：

表 1 可靠性通用场景

场景元素	可能的值	示例
源	学生，教师	检查作业的老师
刺激	没有读入学生信息文件，不按照要求输入指令，输入错误的指令	没有读入文件就查询信息
制品	学生管理系统软件	学生
环境	等待指令输入	运行时
响应	能否识别错误并作出处理	提示没有读入文件，不能进行查询，系统正常运行
响应度量	是否会出现崩溃的现象	系统是否因此崩溃