如何保证软件设计的质量

当前软件开发中，良好的软件设计方法能够大幅度提升软件设计的质量和效率。以往只有具有丰富软件设计经验的设计人员才能设计出结构均衡、质量保证的软件；而现在应用软件设计模式方法能够很好地重用优秀的软件设计方案，进而得到良好的设计结果。

1.风险分析与架构设计

软件架构设计的依据是什么？应该是需求，但架构设计中关键性的策略就是风险分析来驱动架构设计。因为一个产品设计中要考虑的问题很多，但只有在发现风险与消除风险的过程中，发现和抓住高风险的部分，才可以使思考有重点，并且针对潜在威胁有重点地提出设计解决方案，甚至改变我们的设计思想，从而设计出更加良好的产品。

1）风险关注点要明确

设计的本质是发现问题并解决问题的过程，通过分析所面临的重点问题，找到解决方案。没有解决问题的设计并不是好设计，而通过识别和分析风险，可以帮助我们发现问题。一般来说，项目的开发可以分为四个大的阶段，包括初级阶段、架构阶段、构建阶段、交付阶段。在不同阶段的风险曲线也是不一样的，二在架构阶段对软件质量的影响尤为突出。

有效的风险管理包括通过利益相关的合作和参与，并主动地进行风险标识。从一般的规律可以看出：初级阶段由于对信息掌握得不充分，项目的风险是比较大的。细化阶段的风险管理处理可能危及关键目标实现的问题，在架构的验证时风险达到最高，这是因为中间会多次改变方案，很可能重新选择技术路线，知道风险大幅下降。当风险下降到拐点的时候，也就是大部分风险都在架构阶段得到了解决，就可以进入构建阶段了。

在产品的需求中，对质量的需求可能很多。但如果设计中对每一个质量需求都同等的关注，很可能是一个面面俱到的设计。这不可能不是一个没有特色的产品，也是一个可能花费巨资但用户并不满意的产品。什么是风险，如果客户最看重最关注的东西没有达成，这样的产品客户绝不会接受。如果我们能够针对客户最关注的少数质量属性进行设计，就会形成一种独特的架构风格，并且更容易取得成功。

2）抵制前期进行庞大设计

质量风险与进度风险相关，当项目迟迟不能交付，客户频频催促的时候，软件的质量就会受到一定的影响。特别是在项目后期需求变更与频繁修改，对质量与进度都造成很大的风险。很多团队都喜欢挑战复杂的大型项目，殊不知规模越大，项目失败的可能性就越大，一般规模扩大一倍，失败的可能性往往会增加十倍。

大型项目更可能失败、更无法验证、更容易产生不必要和无用的产物。因此，无论多么诱人，都要抵制进行前期进行庞大、完整、僵化的架构设计的诱惑。要有宏伟的远景，但不要有庞大的设计。一开始就设计庞大系统毫无好处可言，其成本可能无比高昂，而且，还会招致不利的项目还值不值得继续下去的质疑。

软件开发是一种学习过程，因此环境和需求的变化不可避免，我们和我们自己的系统都要学会适应变化。在架构的这种演化过程中，由于系统体积较小，将会更容易测试，相互耦合也会更低，开发团队可以更小，协调的成本也会因此降低，系统也更容易部署。

2.使软件设计标准化

在平常的软件设计当中，通常会看到一些人员在软件设计的过程中，比如概要虽然使用了一些图、表来表示软件的架构或者各个模块以及类之间的关系，但是看着总是有点别扭，不太让客户或者开发人员能够很快的理解。为什么会造成这样一个现象呢。我个人觉得最重要的一点就是他们没有一个行业里统一的、标准的规范，以至于在交流沟通上出现了问题。就像人与人之间的交谈一样，如果他们都说的是同样的语言就会很容易交流，然而换做两个不同国家或者不同语种之间的人，他们的沟通和理解都会出现很多问题。

所以，在软件设计过程中，每个人之间，如果能够有一个统一的设计规范，并严格按照规范来实施的话，这会在很大程度上提高软件设计的质量。比如在软件开发时规定，设计阶段使用的 UML 进行建模，这样设计人员和开发人员中间的沟通难度将会很大程度的得到降低。

3.使软件设可度量

目前很多实际项目设计和过程管理中还是采用粗放的定性方法，因此实施和判断就存在各种人为的不准确性，这种状态严重阻碍了软件项目质量的提升，因

此，定量的软件工程技术就显得尤为重要。

定性分析是重要的，它给我们提供了一种迅速的判断能力，但是，没有定量分析给定性分析提供依据，没有定量分析的支撑，定性的结果和趋势的判断，甚至决策，都成了无木之本，很可能做出错误的或者劣质的设计、判断和决策。

1）掌握管理科学的基本思路

为此，我们需要有管理科学的基本思路：复杂的事情简单化，简化的事情数量化，量化的事情专业化，专业的事情模块化。

简化的最有效手段就是分解，把一个复杂的东西分解为最基本的单元，进行单独研究。例如在项目管理中，把一个复杂点的工程分解为最基本的工作，再把这个最进本的工作研究清楚，往往使一切变得简单清晰容易控制。

量化就意味着用全世界通用的数学语言建立统一的标准，它带来的最好处就是容易比较，易于沟通。没有这样标准化的语言，人们将永远纠缠在谁比谁好的争论中。量化在管理中的另一个好处就是易于设置临界值，便于对事物的彼岸花性质和趋势做出判断。

量化的手段为抽象事物寻找共性和规律性提供了方便，而一旦事物的规律被认识到了，就可以成为专业化的行为。专业化最大的好处就是简单重复，而人类生产力进步的最大原动力，就来自于简单重复，供应链、外包、合作等等这些现代生产概念，都是建立在专业化分工的基础之上的。

把专业的行为固化为模块，等于继承了凝结前任劳动成果，它缩短了人类的学习曲线，不但能使效率进一步提高，而且可以用集成模块的方法，把复杂的决策编程简单的选择，来曾倩对付复杂事情的能力。

2）软件设计人员的度量目标以及作用

关于软件开发中，软件设计这一点，我们可以通过对每一项需求进行分析来判断他是不是一种可测量的、客观的方式来表达，比如某项需求要求系统必须是可靠的，表述方式应该为：平均无失效运行时间必须大于 15 个 CPU 小时。

我们可以测量在规格说明书、设计文档、代码和测试计划中发现的故障数，并查找产生故障的根本原因这里可以借助预期故障检测率模型，获得的信息有助于我们确定下列内容：审查和测试活动是否有效；是否可以把产品转入到下一个开发阶段。

我们可以通过对产品和过程特性进行测量，来判断自己是否已经达到了标准，满足了需求，达到了过程的目标。比如我们可以的标准：登陆的时候，在给定时间产品内，失效次数应该少于 20 次；任何模块的代码不得超过 100 行；单元测试的语句覆盖率应该达到 90%。

通过对产品当前过程的属性进行测量，可以对今后的情况进行预测。比如，对规格说明书描述的系统规模进行测量，可以预测出目标系统的规模；通过对设计文档中系统结构特性进行测量，可以预测今后的维护问题；通过对测试阶段软件可靠性测量，可以预测软件实际使用过程中的可靠性。

综上所述，测量对于软件设计人员来说是非常重要的。它可以帮助设计人员了解在开发和维护过程所发生的情况。对当前情况进行评估的同时，我们通过建立基线，来为今后的行为设定目标，也就是说，测量可以把过程和产品的各个方面更加直观地呈现在我们面前，使我们对于活动所影响的实体有更加深入地了解。测量还可以帮助设计人员对项目发生的情况进行控制。通过利用基线、目标以及对关系的了解，不但可以预测今后的情况，而且还可以通过对过程和产品进行变更，来帮助我们实现自己的目标。比如，在对代码模块复杂性进行监控的过程中，只对那些复杂性超过可以接受范围的模块进行全面评审。度量还可以帮助设计人员对过程和产品进行改进。根据规格说明书中质量的测量，以及通过对可能的设计质量进行预测，我们可以增加自己进行设计评审的次数和类型。重要的是，要对可能会利用测量结果的策略人员的期望值进行管理。数据使用者应该明白“预测的准确度是有限的”，而且必须关注测量的误差范围。我们必须注意到，许多软件度量方面的工作，都缺乏测量方面的燕歌行，这个结果很大程度上在于度量元素中人的因素占得分量很重。因此，软件度量会更关注趋势和可能性，但这种趋势应该建立在有效的度量数据上。