如何保证软件设计的质量

14126101 胡志伟

1 前言

从第三方出发，在整个软件生命周期中对软件工程过程及阶段产品进行监督和检查，来确保软件质量符合用户需求，是软件质量保证的目的所在。软件质量保证可以减少并纠正实际软件开发过程或产品与预期的不符，向软件人员提供反映软件质量的信息和数据，提高项目进展的透明度，辅助工程组取得高质量的软件产品。

目前，国内外备受关注的软件质量保证体系有ISO9000认证、CMM / CMMI认证等。ISO9000是国内任何行业参与国际化竞争的通用标准，ISO9000-3是软件行业的指南，可作为软件企业发展过程改善框架，主要面向合同环境，站在用户立场对质量要素进行控制。CMM侧重于软件开发过程中的管理及工程能力的提高，主要从软件组织内部过程的逐步改善入手，为软件企业提供一个过程管理和过程改善的基础，CMM在ISO9000-3的基础上强化了具体过程，更能够帮助软件企业改进和优化管理，实现软件生产的工程化。

在开发阶段能否体现质量保证体系的要求主要表现在编码是否规范化，如设计模式与惯用法的引入；测试阶段能够有助于质量保证体系的实现主要表现在能否进行更加完善高校的测试，如可靠性测试；质量控制过程能否做到质量保证主要取决于质量控制方法与控制技术的应用，如风险管理与目标问题度量，全面质量控制模型等。三个方面有机结合可以使质量保证体系更加系统更加完善，可以发动团队各层人员协同进行质量保证，而非仅仅是SQA的单方面努力。

2 强化规范编程,巧妙引入惯用法

规范编程是程序设计的基本要求，是有效开发的前提，是产品质量保证的基础，是协作式质量保证的首要环节。编程阶段可以充分发挥开发人员的责任感与工作热情，强化规范编码的要求，加强规范编程的培训。系统化程序设计项目，需要较高的代码可读性与代码的健壮性，充分发挥面向对象的抽象化、层次化、多态性、封装性、继承性等优势，同时要加强代码的依赖性管理，因此，规范编程不仅要从常规的规范要求做起，也要引入做好依赖性管理的规则与方法。

常规的编程规范要求可以从命名规定、注释规范、编程风格等各方面。对于代码处理好依赖性管理就要处理好设计规则与设计模式的使用。例如：面向对象的可复用设计的开闭规则(Open-Closed Principle，OCP)，该设计规则的本质在于设计一个模块的时候，通过抽象化与封装使这个模块在不被修改的前提下进行扩展。其优势在于软件系统有一定的适应性、灵活性、稳定性与延续性，这样就保证了系统的可复用性与可维护性。Factory Method定义一个用于创建对象的接口，让子类决定将哪一个类实例化。

总之，编码阶段作为软件过程的基础阶段，直接决定了软件的本质属性，作为协作式软件质量保证的第一步，需要加强编码规范的培训，处理好依赖性管理，灵活使用惯用法及设计模式。

3 加大测试力度，适量强化可靠性测试

在对软件工程进行系统的传统软件测试基础之上，基于软件质量的提高与过程的改善引入软件可靠性工程，软件可靠性工程是以保证和提高软件可靠性为目标，采取系统化的技术，通过工程化方法加以实施并对其过程进行工程化管理的过程技术，软件可靠性工程的基本内涵可概括为软件可靠性的度量，软件可靠性的分析与设计，软件可靠性的测试与验证，软件可靠性管理。

软件可靠性测试是指为了满足用户对软件的可靠性要求，通过对软件进行测试发现并纠正软件中的缺陷，提高软件的可靠性水平并验证它能否达到用户可靠性要求的一种软件测试方法。包括软件可靠性增长测试和软件可靠性验证测试。

软件可靠性增长测试目的是为了发现程序中影响软件可靠性的故障，并排除故障实现软件可靠性增长。软件可靠性验证测试目的是为了验证在给定的统计置信度下，软件当前的可靠性水平是否满足用户的要求。两者都是基于操作剖面形成测试数据展开测试。可靠性增长测试应用于软件系统测试阶段的末期。可靠性验证测试应用于软件验收阶段。

总之，测试阶段不仅仅要通过bug的查找与排除来提高软件质量，也要重视软件其他特性的保证，尤其软件可靠性保证，因此应该引入软件可靠性测试完善协作式质量保证的第二个环节。

4 全面质量控制，高效使用控制技术

SQA人员既要检查过程是否符合规范，还要制定质量管理计划、实 施“过程检查”与“技术评审”，参与开发与测试，跟踪缺陷，改进过程等。引入日式的全面质量控制(TQC)或美国式全面质量管理(TQM)强调对质量的全面性和全过程的质量控制。美国式全面质量管理(TQM)类似于TQC，是一种基于顾客需求与期望 驱动的管理理念，以质量为中心，建立在全员参与基础之上的一种管理方法，其目的在于长期获得顾客满意以及组织成员和社会的利益。从TQC到TQM，已经将质量管理目标从追求企业最大化利益转移到体现企业的社会责任。

用于软件质量控制的一般性方法包括目标问题度量法、风险管理法、PDCA质量控制法等。由质量管理专家戴明提出的PDCA之一种科学的工作程序，即“P(plan)计划-D(do)实施-C(check)-检查A(action)处理”这样一个循环过程。四个阶段迭代循环，没有终点只有起点。通过这样的循环能够提高产品、服务或工作质量。

对于大型软件项目的开展，全面质量管理具有举足轻重的作用，它强调以产品质量为核心，通过企业中所有部门全部成员协作，将管理方法、控制技术等引入质量保证体系，从而想用户系统满足需求的产品。

总之，软件质量控制作为质量保证的主要环节，需要从编码、测试环节入手，加强控制方法与测试方法的结合、控制技术与编码方法的协作，全面开展质量控制。

5 协作式软件质量保证

基于“编码-测试-控制”三方面的协作式软件质量保证，从软件过程出发，表现在人员协作，过程协作，方法协作。

人员协作：SQA不应该仅仅包括软件质量保证小组及部分测试人员，而应该以责任感要求项目组每位成员，做到质量第一，各尽其职，协同工作。基于“编码-测试-控制”的协作式软件质量保证可以实现人员上面的高效协作。

过程协作：ISO/EIA 12207提出，软件生命周期过程包括主要过程、支持过程、组织过程。主要过程包括获取过程、供应过程、开发过程、运行过程、维护过程；支持过程包括文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审核过程、问题解决过程等：组织过程包括管理过程、基础设施过程、改进过程、培训过程等。基于“编码-测试-控制”的协作式软件质量保证提出将以上各项过程协调开展，共同完成质量保证。

方法协作：编码、测试、控制均有其经得起验证的有效方法策略，比如编码方面的命名规则、惯用法及设计模式使用，测试方面的白盒测试技术与黑盒测试技术及可靠性测试等，控制方面的控制技术、容错技术、复用技术等。基于“编码-测试-控制”的协作式软件质量保证强调将不同方法之间的协作，全面进行质量保证。

基于“编码-测试-控制”三方面的协作式软件质量保证，可以组成一个稳固的质量保证正三角，编码、测试、控制分别为相互交叉的三条边，同时从人员、过程、方法三个维度着力，协同作业，共同保证产品的质量。

6 结束语

规范编码是基础，深入测试是关键，全面控制是保障，三方有机结合发动工程组所有成员协作开展全面软件质量保证。质量控制与质量保证不能等同，但是通过质量控制和质量度量可以很好的对软件状况进行评估与预测，同时进行控制改进。可靠性测试，是对软件可靠性度量、验证、预计的有效测试方法。规范编程从开发者的角度，再次强调代码的可读性与健壮性的重要性 ，是软件质量保证的基础，需要强化管理与要求。总之，软件质量保证体系不是一个封闭的系统，

需要编码，测试，控制等多方面的参与。通过系统化的协同工作才能够更好完成其内在的任务与职责。