戴明循环在敏捷软件质量管理中的应用方法研究

孙子谦[1,2] 王雅琴[1*] 黄明明[1]

[1](山东农业大学信息科学与工程学院 山东 泰安 271018)

[2](甲骨文软件研究开发中心(北京)有限公司 北京 100193)

摘 要 敏捷软件过程凭借对变更的适应能力被越来越多的软件企业采用,同时敏捷开发对软件质量管理也提出很大挑战。针对敏捷开发过程在软件质量保证中的特点,以实际项目为例提出一种基于戴明循环的软件测试过程。这种测试过程对软件测试流程实行自反馈方式的改进,提高了软件测试的有效性,降低了资源开销,使软件测试过程能够适应敏捷开发需要。

关键词 戴明循环 软件质量 敏捷过程 软件测试 质量管理

中图分类号 TP311.5 **文献标识码** A DOI: 10.3969/j.issn.1000-386x.2016.11.002

RESEARCH ON APPLYING DEMING CYCLE IN AGILE SOFTWARE QUALITY MANAGEMENT

Sun Ziqian[1,2] Wang Yaqin[1*] Huang Mingming[1]

[1](*College of Information Science and Engineering, Shandong Agricultural University, Taian 271018, Shandong, China*)

[2](*Oracle Software Research and Development Center (Beijing) Co., Ltd., Beijing 100193, China*)

Abstract Agile software process is adopted by more and more software companies with its ability to adapt to the rapid change, meanwhile the agile development also greatly challenges the software quality management. In light to the features of agile development process in software quality assurance, taking the practical project as the example we propose a Deming cycle-based software test process. This test process carries out the improvement in the way of self-feedback on software test flow, thus increases the effectiveness of software test and reduces the overhead in resources, and this enables the software test process to be capable of adapting to the requirement of agile development.

Keywords Deming cycle Software quality Agile process Software test Quality management

0 引 言

“敏捷”是随着信息产业的发展,为了适应新的生存环境而产生的概念。上个世纪末,敏捷制造已经在制造业中为世界广泛接受[1]。随着信息技术和软件产业的发展,人们希望软件能够更灵活、更准确地适应业务环境和需求的变化。在这种情况下,“敏捷”的概念被引入到软件行业中来[2]。1998 年 Aoyama 在文献[3]中首次提出了敏捷软件过程的概念,此后逐渐成为一种潮流而被越来越多的企业采用。自引入敏捷概念以来,敏捷软件过程已经受到诸多研究人员和工程技术人员的重视。由于其思想是“面向人”的灵活过程[4],因而质量管理显得尤为重要。

戴明循环是工业产品质量管理领域得到广泛认可的管理理论。由于软件不是一种实体产品,其质量管理与工业产品有较大的差异。因而戴明循环理论并不能照搬到软件质量管理过程中。

1 戴明循环和敏捷软件质量管理

1.1 戴明循环

戴明循环又叫 PDCA 循环。在 20 世纪 30 年代,美国的 Shewhart提出了 PDCA 循环的概念。PDCA 循环将整个质量管理划分为计划、执行、检查和处理四个阶段[5]。后来,戴明对 PDCA 循环理论进行了完善,并在日本进行了大量推广实践,此后在企业质量管理中得到了广泛应用。同时,PDCA 也成为活动进行的常用工作流程[6]。

PDCA 是一个自反馈式循环往复的过程,且在 PDCA 的各个阶段内部还能进行子循环,以解决该阶段的问题。

单次 PDCA 循环的示意如图 1 所示。

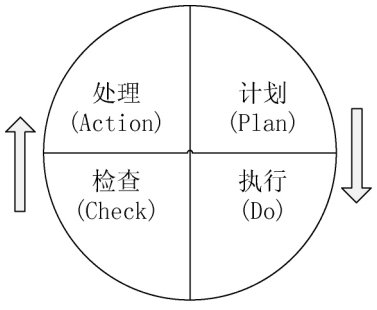

图 1 一次 PDCA 循环示意图

在一轮循环中,计划阶段包括目标的确定与过程的指定,通

收稿日期:2015-07-22。孙子谦,硕士生,主研领域:计算机网络技术,Linux 应用。王雅琴,教授。黄明明,硕士生。

常按照 5W1H 进行归纳，即做什么(What) 、为什么(Why) 、谁(Who) 、什么时间(When) 地点(Where) 以及如何做(How) 。执行阶段并不是简单地接收并执行 Plan 阶段所制定的计划，而是要按照计划进行实施，并对实施过程进行监控，使整个活动按照预期计划进行，达到 Plan 阶段的目标。检查阶段后来被戴明改为研究阶段[8] ，主要任务是根据计划与执行过程的监管结果，对执行的结果进行分析和评估。处理阶段接收检查阶段的结论，总结经验纳入标准化，并解决检查阶段所发现的问题，将检查结果传递给下一轮的计划，作为计划阶段的输入。通过上述四个阶段从而完成整个自反馈过程。每完成一次戴明循环，质量水平就得到一次改进，最终促进产品质量以螺旋方式上升。

1.2　敏捷软件质量管理

敏捷软件过程有诸多不同的类型，在国内比较广为人知的有 XP、Scrum 等[9] 。2001 年 2 月，这些不同方法的创始人成立了 Agile 联盟，并正式将该类方法统称为 Agile[9] ，即敏捷。

敏捷的价值观是[10] :

人及交流的作用胜过过程和工具;

可运行的软件胜过复杂的文档;

客户的协作胜过合同协商;

响应变更胜过固守计划。

敏捷是适应性的，而非预测性的开发方法。尽管敏捷过程在适应需求变化方面广泛受到软件领域欢迎，但由于敏捷软件开发强调“面向人”的灵活性，缺乏统一的质量标准与策略。而大多数敏捷软件过程的研究主要围绕软件的开发过程，敏捷开发中的软件测试过程没有受到足够关注，因而缺乏对软件测试过程的独立的指导。文献 [4] 中还指出，敏捷开发过程中质量监控过程不够标准化，当项目较复杂或工作量较大时缺乏统一的质量标准，难以保证进度。

2　基于戴明循环的软件测试过程

与传统软件开发过程相比，敏捷过程由于更注重适应性，因而具有更高的不可预测性，对软件质量管理过程提出了更高要求。敏捷过程中的 Scrum 方法源于对迭代式面向对象开发方法的改进，受到诸多公司的应用。本文借鉴敏捷开发实践[11] ，结合戴明循环理论，对软件质量管理过程提出如下方法。

Scrum 方法包括三个过程: 计划和设计、冲刺、交付和巩固。整个流程如图 2 所示。

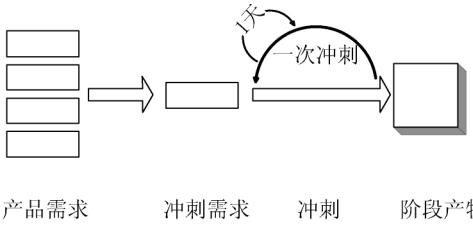

图 2　Scrum 方法过程示意图

在一次 Scrum 过程中，首先 Scrum 团队需要从产品需求中筛选出本次冲刺最需要解决的问题，形成冲刺需求。在此过程中，由于软件质量保障小组需要参与的任务不明显，软件质量保障过程应进行戴明循环的检查阶段和处理阶段。

与软件工程层次的戴明循环不同，在软件测试子过程的管理中，检查阶段不仅要对更高层次戴明循环的状况进行分析检查，更主要的是找出测试过程本身所存在的问题，分析并找到其根本原因，形成强化经验，使测试过程更符合待测软件的特性。此外，处理阶段将着重解决检查阶段发现的问题，对测试用例、测试脚本等进行改进，从而使下一轮测试能够更完备。

此后进入 Scrum 的冲刺过程。该过程中，开发团队将针对冲刺需求进行实现。此时质量保障小组需要进行戴明循环的计划阶段，根据本次冲刺的冲刺需求，以及开发团队在开发过程中的文档、进度等，参照以往测试的经验，结合上一轮测试的结果，利用 5W1H 的思想，对测试所需环境进行筹备并制定测试计划。测试计划需要具有明确的可执行性，并应当在测试计划中具有可以进行过程监控的指标。

Scrum 的交付和巩固阶段时，软件质量保障小组将按照计划阶段所制定测试计划，执行相关测试用例，对交付物进行测试。此外还需要对执行过程进行过程监控，完成测试任务并记录发现的软件缺陷，撰写所需测试报告。汇总对执行过程的监控结果并为检查阶段准备数据。

结合戴明循环后的整个 Scrum 过程如图 3 所示。

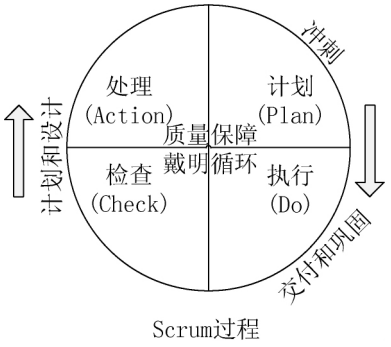

图 3　Scrum 与戴明循环结合示意图

与现存主流的软件质量管理过程不同，该过程作为整个敏捷开发中软件测试子过程，注重对软件测试过程本身的反馈和完善，提高软件测试的执行质量，进一步达到有效提高软件质量的目的。

3　应用实例

作者在甲骨文软件研究开发中心工作期间，在团队中分析并使用了该过程。

作者所在的 Solaris Desktop 团队主要负责 Solaris 系统的桌面应用程序开发及测试。整个团队采用 Scrum 的敏捷开发方法。产品每 2 周为一个冲刺，每个冲刺都有不同的桌面应用被更新。开发部门在完成更新后，会交给发行工程师将整个桌面环境进行打包。发行工程师在完成打包后，会将软件包的存储路径发送邮件给测试团队。自动化测试框架在收到邮件后会自动触发安装、测试等操作，并在完成自动化测试后将结果发送到测试团队的邮件列表。之后软件质量工程师进行结果的检查以及运行手动测试部分。

随着产品的不断更新，观察到测试的耗时在逐渐增长。在该背景下，团队开始采用戴明循环的方法进行测试过程的管理。对于整个自动化过程的首次循环过程中，更多的是将问题逐步明确。在计划与执行阶段，首先修改整个测试的测试计划，在测试计划中添加了对测试过程的监控要求，从而为检查阶段提供必要的信息获取途径。执行阶段仍按照以往的经验完成整个测试过程，并记载所需时间。检查阶段分析整个测试过程的关键路径，如图 4 所示。

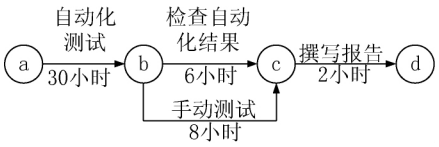

图 4 关键路径分析

从分析中发现,由于为了保障整个测试过程的完备性,自动化测试过程与手动验证、手动测试过程必须使用同一个测试环境。而整个自动化过程必须独占整个测试环境且具有最长的时间开销,因而自动化测试过程为整个过程的关键路径。进一步对比以往的自动化测试结果发现,自动化测试过程耗时还受到待测软件包的影响,并随着软件包的不断更新,自动化测试的耗时一直在增长。因此在处理阶段对自动化测试的记录进行了分析和总结,并将自动化测试过程作为下一轮循环的优化重点。在后一轮循环过程中,在原计划中增加了对自动化测试的记录,并在检查阶段发现,自动化测试的某些测试失败比例比较高,并有部分自动化测试由于环境原因无法一次完成,自动化框架会对该部分触发第二次测试。这不仅给后期的手动验证工作增加了负担,而且花费了大量额外时间开销。通过进一步分析发现,部分待测软件包由于具有细微的特性变化,导致其测试完成后对环境的更改与预期不同,而在测试脚本的设计过程中没有针对该类情况进行容错处理。

在发现上述问题后,仍采用戴明循环的方法,逐步在循环过程中修复了部分自动化测试脚本,并提高了测试脚本的鲁棒性,使整个测试逐步适应新的产品,有效降低了整个测试过程的时间成本,如表 1 所示。

表 1 应用戴明循环前后自动化测试状态比较

项目	应用前	应用后
平均耗时	约 30 小时	约 22 小时
重复执行的套件数量	2	0

我们也针对具体测试对象应用了戴明循环。Solaris Desktop 团队负责 Solaris 平台上 Firefox 的编译、测试等。在每次 Firefox 发布 ESR 版本后,我们都要对新版本进行编译与测试。计划阶段在新的 ESR 版本发行后随之开始。在该阶段,要将新的版本与上一个 ESR 版本进行对比,找出其新的特性,针对这些特性增加新的测试用例,并获取 Firefox 最新的自动化测试套件,针对 Solaris 系统进行配置。在执行阶段,将按照计划阶段的输出,运行相应的测试用例,并将测试结果和日志信息保留进行分析。检查阶段,会详细分析测试的每一个结果与测试日志。在自动化测试脚本中,对于专门针对 Windows 操作系统或者 Mac OS 操作系统的特性变更,该部分在 Solaris 系统上不适用,因此在分析结果并找出该类测试用例以及测试步骤后在处理阶段将该测试删除; 部分测试用例只有针对 Windows 或者 Linux 操作系统的描述,不能直接在 Solaris 系统环境中运行,对于该类自动化测试,会在处理阶段对脚本逐步进行改善。经过几轮循环后,针对 Firefox 的自动化测试脚本会变得更加适合团队的测试需要。

在不断的循环过程中,测试过程的有效性也得到了不断提高。定义由 Desktop 团队发现的桌面系统的故障数量为 $I1$,故障管理系统中桌面系统的故障总数量为 $I2$。其比值可以有效反映测试的有效性,即 $E = I1 : I2$。根据故障管理系统中的数据,E 的值在采用戴明循环之后出现了上升趋势,如表 2 所示。

表 2 测试有效性变化

阶段	E 值
应用戴明循环前	55.2%
应用后三个月	73.8%
应用后六个月	78.4%

在质量管理过程中采用戴明循环半年后,每一轮循环的人员开销由采用前的 12 人/天降低到 6 人/天,团队的工作效率提升了 1 倍。

4 结 语

本文对戴明循环在敏捷软件测试过程本身进行研究,并给出了在具体项目应用中戴明循环的执行策略。

由于敏捷开发更重视“面向人”的灵活性,在敏捷开发中引入戴明循环所带来的提升作用,仍一定程度上受到人的主观因素影响。

在软件质量保障中,戴明循环适用于产品功能相对稳定的项目,以及虽然产品功能持续变化,但敏捷开发的每个阶段均能够有充足时间进行软件测试的项目中。在产品特性变化非常快,以及每日构建型的敏捷项目中,由于每轮测试难以受到有效的检查以及改进,戴明循环难以得到有效的应用。

在此后的研究中,针对敏捷开发模式下产品特性剧烈变化的项目以及每日构建的项目软件质量的保障与提升仍有待研究。

参 考 文 献

[1] 沈备军,陈诚,居德华. 敏捷软件过程的研究[J]. 计算机研究与发展,2002,39(11):1456-1463.

[2] 谢东强. 敏捷软件开发的双迭代模型[J]. 计算机应用与软件,2012,29(6):176-178,198.

[3] Aoyama Mikio. Agile software process and its experience [C]//Proceedings of the 20th international conference on Software engineering (ICSE'98). IEEE Computer Society, 1998:3-12.

[4] 徐琳,陈荔,杨丽. 6 Sigma 管理在敏捷软件开发中的应用[J]. 计算机系统应用,2011,20(6):85-88,20.

[5] JingFeng Ning, Zhiyu Chen, Gang Liu. PDCA Process Application in the Continuous Improvement of Software Quality [C]//Proceedings of the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE). IEEE, 2010,1:61-65.

[6] 武占春,王青,李明树. 一种基于 PDCA 的软件过程控制与改进模型[J]. 软件学报,2006,17(8):1669-1680.

[7] 王青,李明树,刘霞. 一种支持软件过程控制和改进的主动度量模型[J]. 软件学报,2005,16(3):407-418.

[8] 黄飞雪,李志洁,孙效里. 基于 PDCA 的印度软件质量保证模型研究[J]. 哈尔滨工业大学学报,2005,37(11):1583-1585.

[9] 张敬周,钱乐秋,朱三元. Agile 方法研究综述[J]. 计算机应用与软件,2002,19(6):1-9,54.

[10] Agile Alliance. The Agile Manifesto [EB/OL]. http://www.agilealliance.org/the-alliance/the-agile-manifesto/.

[11] Agarwal A, Garg N K, Jain A. Quality assurance for Product development using Agile [C]//2014 International Conference on Optimization, Reliability and Information Technology (ICROIT),2014:44-47.