针对敏捷开发的测试模式

摘要：本文对于常用的软件开发模式和测试模式进行比较后，着重介绍了针对新型的开发模式——敏捷开发的测试模式，并对于这种测试模式的优缺点进行分析及本测试模式在实际项目中的运用，并介绍了自动化测试管理工具 TD（Test Director）和自动化功能测试工具 QTP（Quick Test Professional）在新型测试模式中的应用。

关键字：敏捷开发、测试模式、TD、QTP

(一) 前言

敏捷开发是一个迭代的、增量的、需求频繁变更的过程，客户每周都希望看到新变化，而这些变化最直观体现给客户的就是产品功能的增减，这种非常短的循环，使终端客户可以及时、快速地看到他们花钱构建的软件是一个什么样的结果。因此直接做成可以工作的软件比大量的描述性文档更有效。

正因为敏捷开发的特殊性，敏捷开发的测试无法像传统测试那样：项目初期根据详细的需求文档、设计文档来设计测试用例。因为最初的需求只是一个雏形，甚至连界面也没定，做成什么样子，有什么样的功能开发也不能确定，只能一步步的与客户确认和修改才能定下来；因此用例设计只能通过产品写用例；即先根据简单的需求写出用例的框架，然后拿到产品后，一边添加测试用例一边进行测试。同时敏捷开发的测试模式与传统测试模式不同的还有一个方面，那就是包括 PM、开发人员等也都会一起投入测试。提高系统质量是个 Team work，在开发过程中每个成员都有责任提交高质量的软件交付物（需求、代码、设计文档...），尤其我们团队的“敏捷开发”的项目中，我们还面临人员缺乏、项目多而分散的背景，更加需要整个团队都必须积极投入到测试过程中，PM、DEV、测试都需要积极参与测试和项目的质量保证工作。

(二) 介绍新型的测试模式

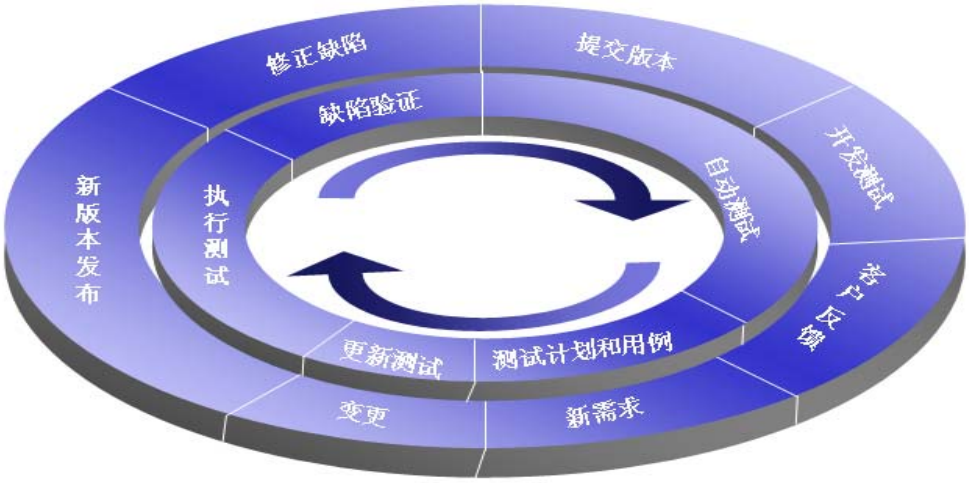

图 1 敏捷开发测试模式

从图 1 不同的环中表示不同的敏捷实践，外环主要是开发的角度，内环为测试的

角度。

测试计划和用例：在测试初期，主要是审核用户需求（FRD）定义的完整性、严密性和功能设计的可测性，然后根据用户需求设计测试计划。测试计划主要说明，（1）测试资源：在测试过程中需要的软件（包括操作系统，补丁版本，数据库版本，被测软件版本，还有诸如打印机、扫描仪等外部信息）和人力资源；（2）测试内容：测试的功能点，测试的类型（功能测试，性能测试，安全性测试，压力测试等）测试的方法，哪些可以自动化测试，哪些需要手动测试。如果是后期，主要是设计测试用例，包括性能测试用例，功能测试用例以及界面测试用例。同时完成配置库的配置，平台的搭建等内容，如一些管理工具（CQ, TestDirector，SVN）以及用户分配等。这个阶段主要输出：测试计划和测试用例。

更新测试：如果是在迭代（发布不同版本）过程中，客户可以根据上次所提供的产品做出一些新的合理的变更，这时我们需要根据变更来更新测试用例。准备在下一个版本中进行测试，也是新版本测试的重点。

执行测试：这时的测试主要分为两大类：接口测试和功能测试。接口测试推荐开发人员在提交代码前直接运行自动单元测试脚本来验证。功能测试：新功能代码提交后，由测试人员单独执行，首先会验证需求文档中的基本功能，一旦出现问题，那就说明开发人员的实现违反了最初客户定义的需求，应及时向开发说明问题，并且和客户沟通，确认开发和测试理解是否一致。之后再进行其它全面的功能测试，那么测试人员要及时与开发人员沟通。同时提交缺陷并且跟踪。这个阶段输出的制品是：本版新的测试用例的执行，缺陷记录。

自动化测试：主要用于回归测试阶段。在发布周期中，随着功能的不完添加与完善，回归测试的任务也就越来越重，同时也是必不可少的。

(三) 新型测试模式在实际项目的运用

实际参与测试的敏捷开发项目是 Webpos。该项目就是采用的敏捷开发的模式：接受客户频繁的需求变更来适应变化；经常性地交付可以工作的软件，交付的间隔一般为一周。这就要求测试组在短期内迅速的对应新增或者变更的需求进行验证测试。我们采用的测试流程如图 2：

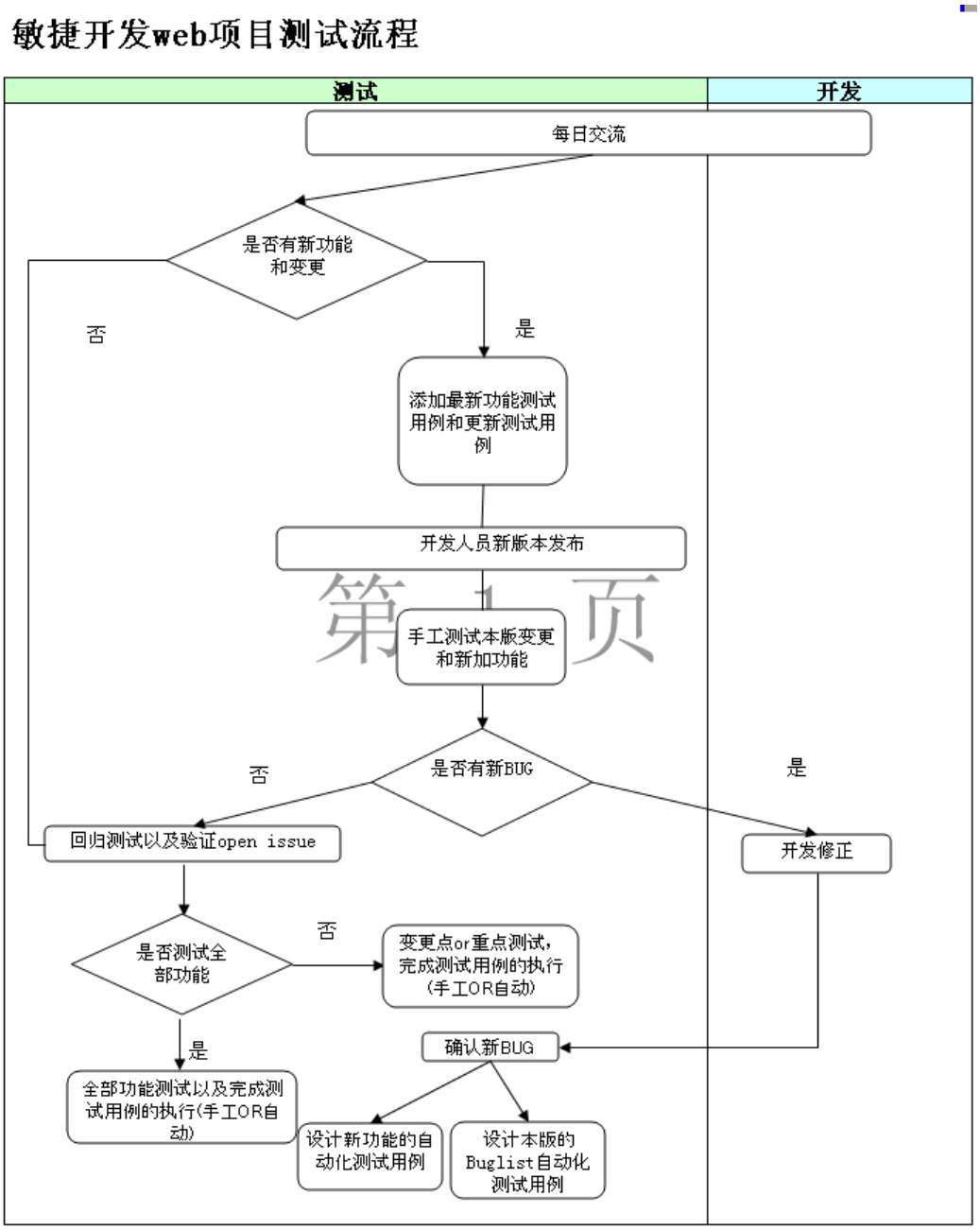

图 2 测试流程

在 WebPos 项目的测试中，贯穿全部测试的主要是新增功能测试和回归测试。测试执行的一开始可以是针对部分功能的，之后可以逐步扩展。接着开始采用迭代的过程完成测试任务，即将测试任务划分为多个周期，一开始可以做些关键的功能性测试，可以对代码中的可复用部分（组件，构件）做完整的测试。接着的迭代周期可以做边缘化的功能测试和其他测试，最后的几个迭代应该用于回归测试，和关键的性能和稳定性测试。特别是回归测试尤为重要，最好的方式就是采用自动化测试工具来提高测试效率。因此我们引入了测试管理工具 TestDirector 和自动化测试工具 Quick Test Professional（QTP）。这样在每次发布版本时，TD 上面可以建立一个新的测试集。如图 3 所示。

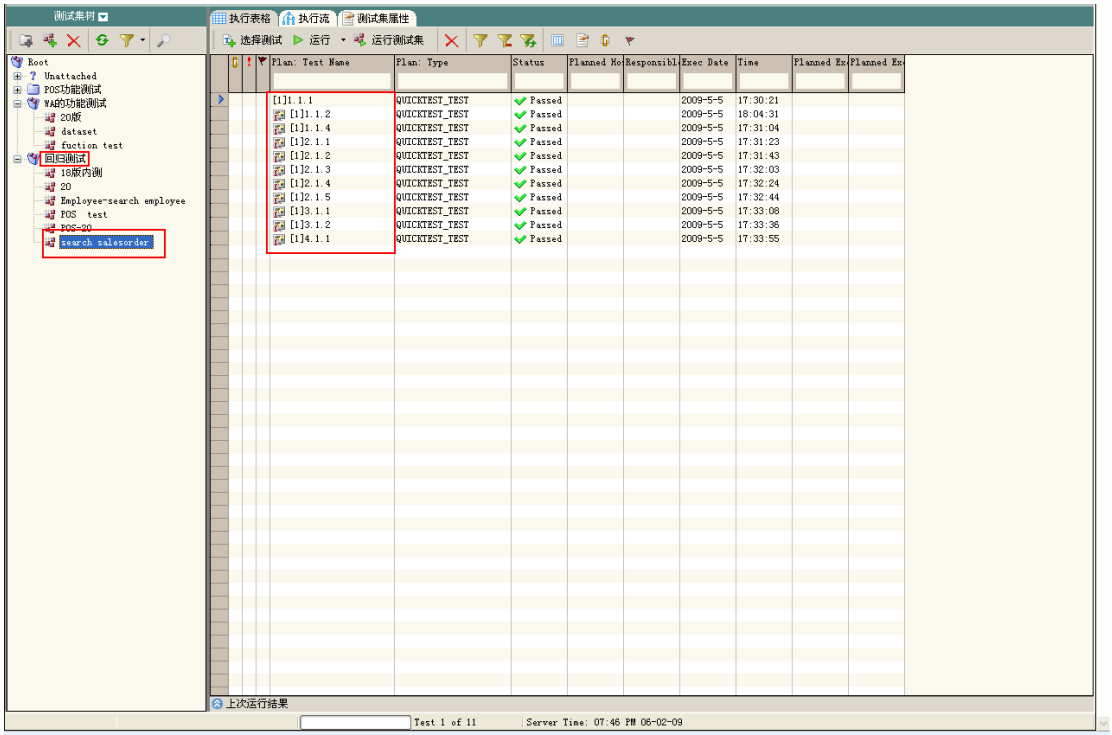

图 3 TD 测试集

导入自动化测试脚本，运行完毕后对回归测试结果进行分析。自动化测试结果如果出现错误，一般是两个原因：一是程序本身就存在缺陷，二是由于测试脚本错误的原因。测试人员可以通过错误结果分析出错的原因：程序本身的问题就提交给开发人员进行修改；若是测试脚本的问题，则由测试人员自己修正测试脚本。执行完毕后，可以将执行结果导出保存为回归测试记录。

利用 QTP 进行自动化测试：开发人员完成本版新增功能或者需求变更的编码工作，遗留版本的缺陷修改后，会给客户提交一个相对比较完善的版本。这时测试人员需要对功能进行测试。对于新增的功能，由于测试时间的紧迫，一般是以小时来计算的，无法在短时间内添加新增功能的自动化测试脚本，所以当时采用手工测试，等功能稳定后再添加自动化测试用例的方式；同时对于一直稳定的功能，进行回归测试。如图 4, 建立功能测试流。

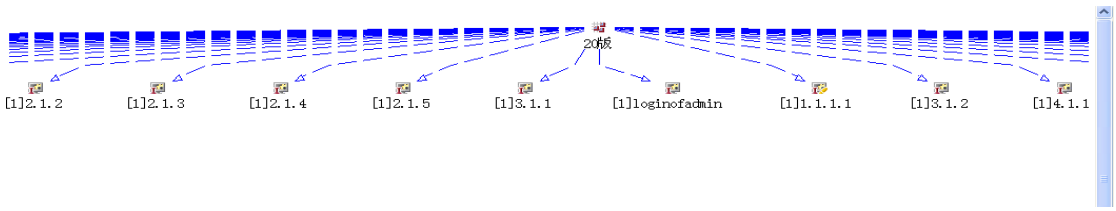

图 4 功能测试流

自动化测试用例也是通过 TestDirector 来管理，可分模块和功能来管理，不同的测试人员也可以随时添加和更新测试用例的脚本。在执行测试阶段中，测试人员需要对已有的测试用例进行及时的维护。通常以下两种情况下要新增一些测试用例：一是对于当初测试设计不周全的领域，二是对于外部的 Bug（比如从客户报告来的），没有被现有测试用例所覆盖。当产品的功能设计出现更改时和现有的功能需求同步。如

下图 5.

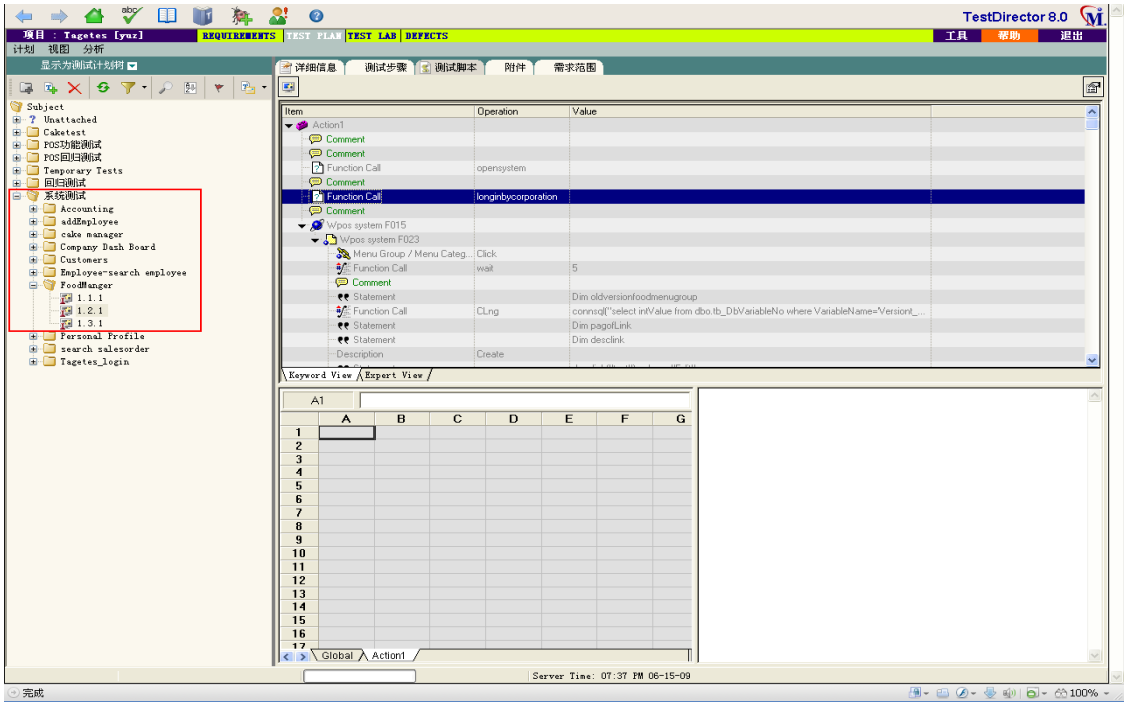

图 5 自动化测试用例

测试结果的分析同回归测试结果分析。

在每一次提交版本后，把开发人员和顾客代表包括进来，不要只是测试员自己做测试，开发也和测试人员一起测试新的版本。计划在每个迭代中探索产品，寻找 bug、遗漏的问题和改进的机会。

(四) 新型测试模式在敏捷开发中的优点

1、可以快速对应频繁的变更。

由于敏捷式开发持续地改进设计，以便于每次迭代结束生成的系统都具有最适合于那次迭代中需求的设计。那频繁的变更和版本的发布，只能通过自动化测试来保证已经确认的功能，而新功能和需求的测试成相应版本的重点。

2、自动化测试，减小人力或者重复劳动。

纯粹的自动化功能测试，而不是手工测试。即使测试用例有上千个，自动化测试也可以每天可以回归几十次。而手工测试如果要回归，将花费大量的人力和重复劳动。

也许，测试最重要的好处就是它对于构架和设计的影响。为了使一个模块或者应用程序具有可测试性，必须对它进行解耦合。越是具有可测试性，耦合关系就越弱。全面的考虑验收测试和单元测试的行为对于软件的结构具有深远的正面的影响。

(五) 新型测试模式的不足

1、频繁的变更，对自动化测试架构提出了更高的要求。

由于自动化功能测试是基本 UI 的测试，如果是频繁的变更，不便于自动化脚本的维护。那么这就对代码的架构提出更高的要求，怎么样让自动测试脚本快速对应变

更？开发人员在设计初期就应该考虑到测试先行的问题，就应该有不同于普通的系统架构方面的权衡，而不是在进行迭代后，使用自动化测试工具出现很多问题之后，才发现架构的问题，这时的人力、物力的浪费都是很巨大的，在 Webpos 项目中，就是因为没有考虑到这个问题，导致自动化测试脚本在每次版本变更后对应起来非常困难，甚至在项目中期因为可扩展性差的问题而重新进行架构设计，造成了大量的人力成本的浪费。

2、如果太多依赖客户，影响我们在客户心中的信誉度。

由于敏捷开发中频繁变更的需求都是从客户那里获取的，所以对于一些建议性方面的 bug 是否该提，测试人员的立场非常尴尬。因为开发人员不愿意为了这些建议性的问题来增加工作量，特别是这些问题客户可能并未提出。时间久了就会对测试人员产生抱怨。这样，测试人员的积极性也会被降低甚至从此不再关注此类问题；但是对于客户来说，遇到此类问题就可能会抱怨为何测试人员没能及早发现而遗留到最终用户手上。

3、由于自动化测试，对测试人员提出了更高的要求。

敏捷开发的测试人员不但要熟悉开发语言、自动化测试工具，能够编写自动化测试脚本或者用工具录制，而且要参与项目和系统的需求分析和架构设计中。但是对于目前实际的项目，测试人员只是被要求进行验收测试，并没有被要求更多的思考需求的可实现性，也没有将自身作为第一用户参与项目和系统的需求分析，设计和开发。当然这也是很多敏捷开发项目中测试人员甚至开发人员都无法达到的水平。这也是我们今后努力的方向。

(六) 改进

无论是传统开发的测试模式，还是新型开发的测试模式，目的都是为了保证产品质量，达到客户满意。对于目前实际项目来说，目前仍旧需要持续改进的就是：

1.系统架构的设计应该充分考虑测试的需求和可测试性。

2.测试人员加强主流测试工具的学习，提高测试水平，并且积极的参与到项目讨论及客户交流中去。

希望能通过我们的持续改进，使我们的团队充满激情和活力，能够适应更大的变化，做出更高质量的软件。