敏捷开发与敏捷测试

敏捷开发：1.敏捷型方法是“适配性”而非“预设性”。重型方法试图对一个软件开发项目在很长的时间跨度内作出详细的计划，然后依计划进行开发。这类方法在计划制定完成后拒绝变化。而敏捷型方法则欢迎变化。其实，它们的目的就是成为适应变化的过程，甚至能允许改变自身来适应变化。2.敏捷型方法是“面向人”的(people-oriented) 而非“面向过程”的 (process-oriented)。 它们试图使软件开发工作顺应人的天性而非逆之。它们强调软件开发应当是一项愉快的活动。

我认为以上两个特点很好的概括了敏捷开发方法的核心思想：适应变化和以人为中心。

敏捷开发其实借鉴了大量软件工程中的方法。迭代与增量开发，这两种在任何一本软件工程教材中都会被提到的方法，在敏捷开发模式中扮演了很重要的角色。再向前追溯，我们还也可见到瀑布式与快速原型法的影子，也许还有多。 改善，而非创新。敏捷开发可理解为在原有软件开发方法基础上的整合——取其精华，去其糟粕。因此敏捷开发继承了不少原有方法的优势。“在敏捷软件开发的过程中，我们每两周都会得到一个可以工作的软件，”Fowler 介绍，“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。”

敏捷开发提出了以下遵循的原则：

· 我们最优先要做的是通过尽早的、持续的交付有价值的软件来使客户满意。

· 即使到了开发的后期，也欢迎改变需求。敏捷过程利用变化来为客户创造竞争优势。

· 经常性地交付可以工作的软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。

· 在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。

· 围绕被激励起来的个体来构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。

· 在团队内部，最具有效果并富有效率的传递信息的方法，就是面对面的交谈。

· 工作的软件是首要的进度度量标准。

· 敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一个长期的、恒定的开发速度。

· 不断地关注优秀的技能和好的设计会增强敏捷能力。

· 简单是最根本的。

· 最好的构架、需求和设计出于自组织团队。

· 每隔一定时间，团队会在如何才能更有效地工作方面进行反省，然后相应地对自己的行为进行调整。

敏捷测试流程总结：

在敏捷方法中，XP 方法强调测试在整个项目开发过程中的重要性。针对敏捷开发方法的敏捷测试不同于以往针对传统开发模式的测试,在敏捷团队中,测试是整个项目组的“车头灯”，它告诉大家现在到哪了，正在往哪个方向走。测试员为项目组提供丰富的信息，使得项目组基于这些可靠的信息作出正确的决定。不仅是测试员要保证质量，而是整个项目组的每一个人都要对质量负责。测试员不跟开发人员纠缠错误，而是帮助他们找到目标,共同为达到项目的最终目标而努力。敏捷测试也需要高度迭代工作、频繁得到客户的反馈，需要动态调整测试计划、测试的执行。并且，敏捷测试人员参与到了更多的敏捷生产活动中，积极的影响了团队做出的决定和计划。

根据公司项目目前采用的敏捷开发模式，相应的敏捷测试建议采用以下流程：

1. 验证需求和设计

需求和设计具体来说一般包括：（1）由项目经理根据需求文本而编写的功能设计文本（Functional Design Specification）；（2）由开发人员根据功能文本而编写的实施设计文本（Implementation Design Specification）包括(Architecture Document, Project Scope Statement, Use Case)。作为测试人员，审核重点是检查文本对用户需求定义的完整性、严密性和功能设计的可测性.

在测试初期，测试人员要学会做静态测试，做好需求分析，做好对设计逻辑的分析。测试人员要更多的思考需求的可实现性，将自身作为第一用户积极参与项目和系统的需求分析，设计和开发。积极地参与前期工作，并迅速反馈给设计和开发其静态测试结果。要尽早的开始测试，不要等待到功能完全做好才开始。

产出物:测试需要提交评审结果文档,可以让测试更多的参与 DB Design,框架的评审中来

2. 测试计划，测试用例

2.1 编写计划、测试用例

在敏捷开发的过程中由于是根据每个 user story 来估算时间的。开发人员将对本次迭代所需要的完成的 user story 进行评估。开发人员可以和客户直接沟通，来确定每个 user story 的优先级。

好处：

客户可以很清楚的了解到哪些 user story 需要花费多长的时间，以及他们的优先级。

问题：

在 user story 的时间估算上，开发人员常会估算过少。引起版本无法按时发布或者必须进行加班才能进行发布。

分析：

由于版本更新很快，任务的时间都是以小时来进行估算的。开发人员一般会忽略掉开发以外的时间，比如开发中遇到问题的时间，开会，给**其他**成员提供帮助的时间，等等。

举个例子：

开发人员估算某个 user story 编码的时间需要 1.5 天，开发人员自己估算了其他时间为半天。于是开发人员给的估算时间是 2 天。

开发阶段实际的花费时间如下，每天花费开例会的时间。在例会中项目的其他成员需要技术上的支持。于是发费了 3 个小时进行帮助。在开发的过程中遇到了一些没有预见到的问题，结果解决问题花费了 4 个小时。（也许更多）。需要处理一些公司突发性的事务等等。

所以非常建议大家在估算时间上能充分的考虑到以外的因素，某本 XP 相关的书上写到，在时间估算上最好的时间是编码时间的 2－3 倍。听起来很吓人，但是实际的过程中，的确需要这么多的时间。

测试人员根据已审核通过的需求和设计编制测试计划，设计测试用例。在前面提到的三种文本中，功能设计文本是主要依据。测试的这两个文本也要被项目经理和开发人员审核。

2.2 测试用例的审核

为使开发人员能参与到 Test Case 的 Review 中来，以保证 TC 的质量和可行性，确保测试工作的顺利进行，让开发人员迅速地了解测试的重点并给出相应的意见和建议，测试人员在出 TC 的同时，应出一份 TC_Matrix（Test Case 跟踪矩阵），其中注明 TC 已覆盖了哪些 Features，具体每个 Features 对应的 TC 的编号，这样在测试经理和 PM 对 TC 进行 Review 的时候，能够对 TC 的覆盖率一目了然，对覆盖率不足（如某个重点 Feature 的 Test Case 不够）的地方能够及时给出意见。

另外，在每天早上的 Morning Meeting 上，测试人员可以简洁地讲述一下当天测试的重点部分，以及项目中存在哪些严重的 bug，让开发人员了解当天测试的重点是什么，怎样进行测试，并提出自己的意见和建议。这样做加强了开发与测试人员的交流和沟通，使测试工作能够更加有效，更加顺利地开展。

在迭代后期测试要抽时间更新 test case。

3. 实施运行测试

在敏捷方法中，测试有两种：单元测试和接收测试。单元测试是由开发人员来完成的，接收测试是由客户代表来完成。

由于我们客户无法在现场，我们采取了，开发人员做单元测试，测试人员做验证测试，最后由客户进行接收测试。在每个版本发布给客户之前必须由测试人员进行测试，发布版本之后由客户做接收测试，提出需要修改的地方。需要修改的地方将在下一个发布完成。

· 单元测试

在 daily build 版本给测试前，开发首先要做单元测试，提前告知软件中的薄弱环节，帮助测试人员调整测试重点。Unit test

做单元测试的好处是可以提高版本质量，减轻测试的工作量，减少浅层次的 bug 的发生率，使测试人员能够将更多的精力投入到寻找深层次的 bug 上面。

· 验证测试

测试人员的验证测试从总体上说就是将上一步设计的测试用例按计划付诸实施的过程。这一阶段的测试必须在周密的计划下进行。这种计划性首先体现在开发和测试的相互协调配合，根据产品的架构和功能模块的依赖关系，按照项目的总体计划共同推进。从测试的过程来看，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。

3.1 每日提供 bug 趋势

为方便衡量项目的进度，测试可每天测试完毕后提供测试的 bug 趋势，即将每天新生成的 Bug 数和每天被解决的 Bug 数标成一个趋势图表。一般在项目的开始阶段新生 Bug 数曲线会呈上升趋势，到项目中后期被解决 Bug 数曲线会趋于上升，而新生 Bug 数曲线应下降，到项目最后，两条曲线都趋向于零。PM 会持续观察这张图表，确保项目健康发展，同时通过分析预测项目 Bug，对于每个版本的 bug，开发都应该想想为什么会出现这样的问题，特别是很低级的 bug，对于同类的 bug，是否可以避免。

测试需要考虑:探索性测试用例的编写

3.2 测试用例的维护

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 Beta 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（敏捷项目中功能设计的更改频繁），所涉及的测试用例也要相应地修改，使测试用例保持和现有的功能需求同步。

3.3 根据项目不断补充 Common Sense

在项目进行过程中，测试人员需要不断积累经验，不断补充、完善各类目的 Common Sense 标准。例如，由 CTTS 项目总结出的 Common Sense for USA 标准，在以后的美国项目中要严格按照它来执行测试，保证以前出现过的失误在以后的项目测试中不会再出现。在保证项目质量的同时，不断积累新的经验。

3.4 控制中间版本

为更好地保证软件质量，规避风险，必须加强对中间版本的控制。例如，客户要求或者计划周五要提交版本，则周三一定要提交一个中间过程的版本进行测试，也就是控制中间版本，避免所有的工作都压到后期最紧急的时候去完成。以前的项目中出现过项目前期很轻松，到后期 bug 越来越多，开发人员和测试人员都异常忙碌，经常加班的情况。为减少后期工作量，规避风险，建议开发进行 Daliy Build，或者按照完成一个 feature 就进行一次 build，针对这个 feature 进行测试，这样就可以有效避免后期 bug 越来越多的状况发生，后期工作量也就会相应减少，项目的质量也会更有保证。

3.5 发布版本前编写 Release Note

在每次发布版本之前，测试人员要根据待发布的版本情况编写 Release Note，使客户对发布的版本情况一目了然。Release Note 主要包括三方面的内容：Fixed，New Features，Known Problems。其中，Fixed 部分写明此版本修复了上个版本中存在的的哪些比较大的 bug；New Features 部分写明此版本新增加了哪些功能；Known Problems 部分写明此版本尚存在哪些比较大的问题，有待下个版本改善；或者列出需求不太明确的地方，有待客户给出明确答复意见，在下个版本中完成。

4. 需求管理

采用敏捷开发模式的项目中，客户对于需求的变更很频繁。因此，需求管理是十分必要和重要的工作。整个项目进行过程中，对不断变化的需求，一定要作跟踪，每次的需求变更都要有相应的历史记录，方便后期的管理和维护工作。可将每次的变更整理记录到需求跟踪文档中，并使该文档始终保持最新更新的状态，与需求的变化保持同步。

问题：

客户可能会在一个功能点上来回修改他们的需求，也许开始需要某个功能，结果做完以后又觉得不好，于是让去掉这个功能。后来又由于一些原因，有需要加上。在整个过程中可能来回修改了很多次。那一定要记录下变更的内容和日期。可能后期客户会觉得一个功能怎么会花那么多的时间，不是以前很早就做过

了吗？这时这些记录才是解决客户疑虑的最好证明。说白了，是有证据证明我们做了很多的变更。大家可能觉得，怎么会有这个问题。其实当一个项目长达半年以上，也许大家的记忆力都不可靠了(:p)

建议：

目前采用的是 vss 工具，对每天的 Email 中提到的需求变更做一次 backup，文档以当天收到 Email 的日期命名

5. 项目开发末期开展“bug 大扫除”

在项目开发的末期，可以开展“bug 大扫除”活动。划出一个专门的时间段，在这期间所有参与项目的人员，集中全部精力，搜寻项目的 Bug。注意以下要点：（1）尽管这是一个测试活动，但参与者并不仅限于测试人员。项目经理，开发人员甚至于高层管理人员都应参加，如同全民动员。目的是要集思广益；（2）要鼓励各部门，领域交叉搜索，因为新的思路和视角通常有助于发现更多的 Bug；（3）为调动积极性，增强趣味性，可以适当引入竞争机制，比如当活动结束时，评出发现 Bug 最多，发现最严重 Bug 的个人，给以物质和精神奖励。（4）可以分专题展开，比如安全性、用户界面可用性、国际化和本地化等等。