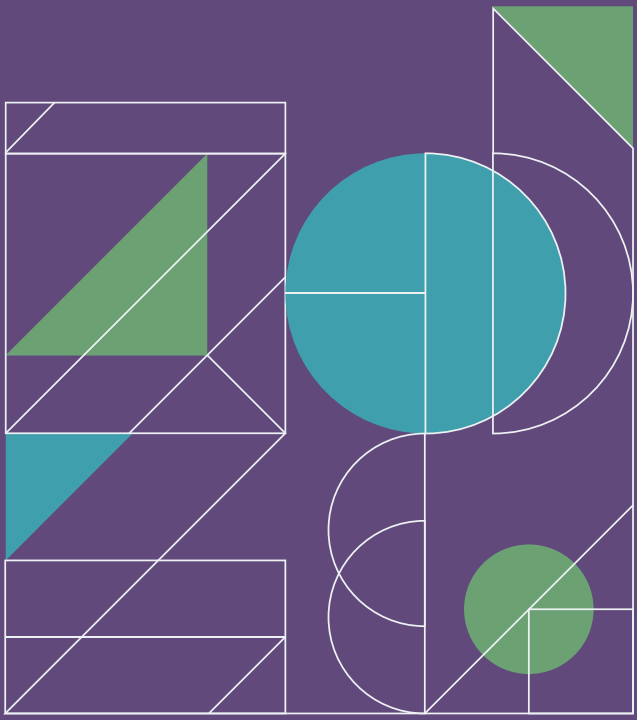
/thoughtworks

不止测试

林冰玉 著



目录

不止测试

“测试把关质量”，这是传统对质量保障工作的普遍认知。

基于业务需求设计测试策略、制定测试计划、编写测试用例、执行测试，对于测试有问题的地方提交缺陷并对缺陷进行管理，这几乎就是测试人员的全部工作内容。

随着业务形态多样化、数据复杂度增加、技术架构演进、基础设施发展，软件所处的生态越来越复杂、不确定性增多，质量面临的风险也更加不可预测；同时，软件市场的竞争越来越激烈，对软件质量的要求有了更高的呼声。

在这样的新形势下，交付高质量的软件面临着更大的挑战，光靠传统的那些质量保障工作无法实现。

“质量不是检测出来的。”著名质量管理专家戴明先生的这句名言告诉我们，光靠开发完成后的测试是没法保障质量的，质量需要团队成员一起负责，需要从软件开发的整个生命周期给予关注：

- 需要左移，关注业务的真正价值，要以业务价值驱动开发和测试；

- 需要右移，关注和利用生产环境的数据和信息，对线上问题进行深入分析，以优化和改进上线前的开发和测试工作；
- 需要关注整个交付过程，关注计划安排、团队协作等多个方面。

这，对测试人员的工作寄予了更高的期望，也赋予了更大的价值。

测试人员，已经不能止步于单纯的测试工作，对于质量相关的各方面都责无旁贷；除了自己身体力行参与质量相关工作外，测试人员还需要对团队不同角色进行质量赋能，承担赋能者的职责。

为了应对高质量交付所面临的挑战，测试需要实现从传统到适应新形势的转型; 为了胜任新形势下的质量工作，测试人员需要提升技能，探索适合自己的职业发展之路。

测试人员做的工作不止测试，需要承担质量分析者、协调者以及倡导者职责，也因此有了另一个更为合适的名字——QA（质量倡导者，Quality Advocate）。

林冰玉
质量赋能专家

用业务价值驱动测试

我们知道，敏捷交付价值，敏捷测试要以业务价值驱动，要以优化业务价值为目标。

> **“TA 主要关注系统的操作上，对业务价值关注太少。”**
>
> **“要多关注业务价值，多从业务价值的角度去思考。”**

这是我们最常见的给敏捷测试人员的反馈和建议。

可是，业务价值到底是什么？

业务价值可以简单理解为：

- 帮助企业盈利
- 满足企业业务发展要求
- 能够带来业务价值的产品需要满足用户需求、让用户使用方便

了解了业务价值，接下来我们来看测试如何关注业务价值、业务价值驱动型的测试及其相关落地实践。

01 测试从哪些维度关注业务价值

我们需要从四个不同维度来思考和组织相应的测试活动，以实现优化业务价值，如下图示：

1. 从终端用户角度进行测试

从终端用户角度进行测试是基本的测试思维，是测试人员必备的技能要求。

在思考、设计测试用例并执行测试的时候，不能简单的套用用例设计方法去机械地进行，而是要考虑用户可能的行为习惯、使用场景等。

下面来看两个例子：

- 场景一：我们在使用移动设备的时候，一般会设置自动锁屏。当测试某个移动 App 时，测试人员觉得

移动设备的自动锁屏功能会导致测试很不方便，于是关掉了自动锁屏功能，结果后来在一台测试人员自己使用的设置自动锁屏的手机中，发现了一个缺陷，就是在每一次锁屏并开启后，充值选项会新加载一遍，可以导致多个重复的充值选项存在。

- 场景二：测试移动设备断网的情况，通常会采用调成飞行模式或者关闭数据网络的方式，这两种方式对于 iPad 或者 iOS 7 版本以下的 iPhone 通常都需要离开所测的 App 去设置，这样的操作会导致某些功能不可测或者很多问题不能被发现。如果在移动中测试，也就是说打开 App 测试过程中，测试人员移动到一个没有网络的地方（比如电梯），将有可能发现很多意想不到的缺陷。

上面两个典型的用户场景都是我们不容忽视的，根据这些真实场景去测试，把自己想象成终端用户，就能容易发现用户可能遇到的问题。

除此之外，还要考虑终端用户的体验，比如说页面的布局、配色、易操作性、页面加载时间等都是在测试过程中需要考虑的因素。这样有助于交付一个增强用户体验

的系统，对优化业务价值是有帮助的。

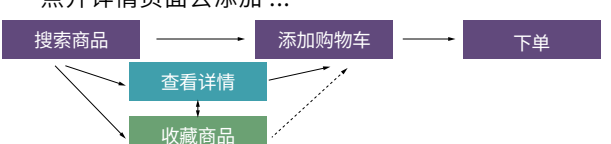
在敏捷开发生命周期的需求分析、特性启动（Feature kickoff）、用户故事评审、用例设计、故事验收和故事测试等环节，都可以体现出从终端用户角度考虑的价值。

2. 以业务为重点的测试

单个终端用户的操作可能只是业务流程的一部分，除了从终端用户角度去测试，还需要看得更高一点，那就是业务流程的合理性、流畅性和完整性。这就要求能够真正理解企业的业务，以业务为重点来测试。

下面来看一个业务流程的简单例子：

- 某购物网站提供搜索、查看详情、收藏商品的功能，用户可以将看中的商品添加购物车，并下单购买。最初的时候，只能从搜索的结果列表和商品详情页面添加购物车，而收藏是非常重要的一个功能，用户如果想把收藏的商品添加购物车还必须得一个个点开详情页面去添加 ...



这个例子中的搜索、收藏、查看详情和添加购物车本身的功能都没有问题，但是从整个流程来看就会发现缺失掉的场景：从收藏列表添加购物车。

单独的一个用户故事，通常只是覆盖某一个小的功能，但对于用户故事的分析、评审、开发和测试，就需要把所有相关故事串起来，甚至需要跟系统其他关联的特性或者第三方系统集成起来看，从整个业务流程角度去思考，以保证整个业务流程的合理性和相关功能的正确性，确保真正满足企业业务需求。

对于复杂业务情况下要发现缺失的路径可能没前面例子那么容易，需要敏捷团队各个角色都能加强这方面意识的培养。QA 和 Dev 都能在敏捷开发生命周期需求分析阶段介入，尽早接触需求，有利于更好的理解业务流程。深刻理解业务流程、以业务为重点的测试，能够更容易发现关键业务流程中遗漏的点或相应的缺陷，测试带来的业务价值又上了一个台阶。

3. 映射业务影响

保证了每个业务流程的流畅性，还需要综合企业多个业

务来看业务的优先级，能够区分哪些是关键业务和外围业务，要能理解每个业务对于企业的影响、给企业带来的价值。

举例说明：

- 某公司的系统为税务和移民两块业务服务，在报税忙季，跟报税相关的业务最关键，优先级最高；而在淡季的时候可能报税业务的优先级就要比移民相关业务优先级低。当我们在测试该系统的时候需要关注这种优先级的变化，测试策略上需要有相应的调整。

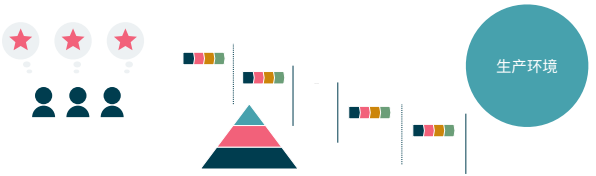
- 某公司的系统提供一种通过观看录播视频进行咨询的功能，但是由于制作视频比较费事费力，公司并没有准备马上把这个功能投入使用。这个功能目前并不完善，还有很多要改进的地方，但是由于不是当下的关键业务，它的优先级低了不少。测试人员在测试过程中，也不用投入太多精力去关注这个功能。

在测试过程中需要把系统的质量状况跟业务紧密关联起

来，需要能够识别出系统缺陷对业务带来的影响，包括可能影响到的用户数、具体的影响是什么、有没有可以绕过的方法不至于中断业务的开展等。如果影响到最关键的客户，那么相同严重性的 bug，对于只影响少量不怎么重要的客户来说，优先级也是完全不一样的。

下面几个方面可以帮助我们开展测试活动的时候更好的考虑业务影响：

- 定期跟 PO 等关键业务人员沟通，以保证大家对业务优先级的一致认识，测试策略、测试计划要考虑基于业务优先级来制定。
- 对于任何代码变更，不能是盲目的无重点的回归测试，要基于业务风险来测。
- 对于自动化测试，需要权衡成本和收益，更需要基于业务风险来考虑，不能盲目覆盖、事倍功半，要以最小业务风险实现最佳测试覆盖。
- 利用生产环境下的 QA 技术（请见下篇文章），收集生产环境数据并分析，我们可以更好的理解业务关键性和优先级。



4. 关联业务指标

除了前面三个维度，在度量项目 / 产品的测试或质量的时候，不能仅局限于项目 / 产品范围，要跟企业的业务指标相关联，跟踪、量化测试活动对业务指标的影响。这个维度是非常重要的。

测试不再是考虑发现尽量多的缺陷，而是要基于如何快速将产品投递到市场的战略，尽快让企业能够借助产品盈利。

做到这个维度，需要：

- 从相关干系人那里确认业务指标，定期跟各干系人沟通，收集对于业务指标的反馈并更新。
- 收集和分析生产环境的真实数据和模式，反馈到开发过程去更精确地预防缺陷，提高内建质量。

02 业务价值驱动的测试跟传统测试的对比

业务价值驱动的测试和传统测试在以下几个维度的关注点都有不同：需求的关注、计划与执行、应对缺陷、测试人员角度、目标、生产力、指标的关注、有效性，详细对比如下表。

维度	传统测试	业务价值驱动型测试
需求的关注	功能需求	业务需求
计划与执行	获得最大的测试覆盖率	基于风险考虑在最优时间内获得最大覆盖率
应对缺陷	不做缺陷预防，而是响应式的机制	分析历史数据和模式来精确预防缺陷
测试人员角度	测试	业务分析（BA） 和 质量分析（QA）
目标	高效发现缺陷	快速投递到市场
生产力	基于测试用例编写或执行的数量	分析历史数据和模式来精确预防缺陷
指标	面向项目的指标	面向业务的指标
有效性	实现测试的目标	通过健壮的测试实现业务目标

从表中可以看到：

1. 业务价值驱动型测试要关注业务的需求，而不仅仅是功能需求；

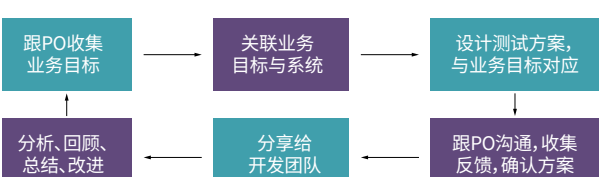
2. 业务价值驱动型测试以追求快速高质量的交付价值为目标，单纯的测试覆盖率和缺陷数量不再是考核的因素。

3. 业务价值驱动型测试不是不关注缺陷，要以预防缺陷为主，正确跟踪缺陷并进行深入分析是帮助缺陷预防的必要的手段。

4. 业务价值驱动型测试不再简单的根据数量来考核生产力，可以从多个维度评估测试的成熟度，以驱动出持续改进的方案。详情可以参考我的文章《聚焦测试，驱动卓越》

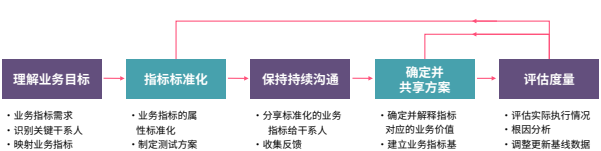
03 如何阐明业务价值

业务价值可能比较抽象，似乎不是那么好理解，况且要跟具体的测试活动对应起来更是有些难度。先给大家分享一个我们项目上的实践，我们通过下图所示的流程将业务目标、业务价值传递给团队：



1. 定期或者按需跟 PO 进行沟通，收集该时间段的业务目标、确定业务方向；
2. 将业务目标跟系统功能关联起来，确认测试方面要关注的重心；
3. 设计相应的测试方案，在方案里标明对应的业务目标；
4. 跟 PO 沟通测试方案，收集反馈，并最终确认方案；
5. 将确认的测试方案更新给团队，确保团队所有角色理解一致；
6. 方案执行后，对结果进行分析和总结，回顾做得好的和需要改进的，持续改进；
7. 把总结分析的结果更新给 PO，如此循环，一步步的优化。

每个项目的情况并不会相同，下图是一个比较通用的流程，可以根据项目团队具体情况调整：



04 优化业务价值的测试实践

业务价值有效传递给了团队，那么就需要在测试实践中关注业务价值，从而帮助优化业务价值。下面的这些测试实践，都是可以帮助优化业务价值的：

共创测试方案

测试方案在创建之前，需要跟业务人员沟通清楚对应功能特性的业务目标、业务价值，需要跟开发人员沟通相应功能模块的技术风险，基于风险的测试才是最能体现价值的。

测试左移

测试左移的目的就是为了更早的澄清和确定需求，确保团队清晰一致的理解需求的业务价值，做正确的事情。

测试右移

测试右移也叫“生产环境下的 QA”（见本书下一篇文章），利用对终端用户行为和生产环境数据的分析，帮助优化业务、优化质量内建过程，提高交付软件的质量。

精益测试

精益测试要求测试能够做到恰到好处，减少浪费，从而加速软件开发流程，快速交付。请参考 Thoughtworks 洞见文章《精益测试》了解更多详情。

渐进式的自动化测试

自动化测试要随着软件开发的进度渐进式的增加，而不是等功能开发完成再来写，可以提高投资回报率（ROI），让一开始就有自动化测试帮助提供反馈。

测试资产的管理和复用

测试资产跟产品代码一起用版本控制工具管理，尽量复用原有测试资产，不要重复造轮子。这样可以提高测试效率，加速价值交付流程。

增强的测试技术

采用智能测试技术和分析技术，提高测试的精准度和有效性，更高效的帮助业务价值的交付。

缺陷预防

利用对缺陷模式的分析，在软件开发生命周期每个环节更好的做好相应的测试工作，帮助有效的预防缺陷，降低成本，加速交付。

持续改进

整个软件开发生命周期的测试活动始终以目标驱动，实时度量，并持续改进。

测试人员能力建设

加强测试人员对于业务理解能力的培养，建立业务价值

思维，提高业务敏感度，将有利于更好的理解业务价值。

05 业务价值驱动型的测试人员

要做好业务价值驱动的测试，对测试人员的要求有：

改变认知

测试人员首先需要改变对测试目标的认知，不能再以发现更多的缺陷为目标，要培养业务敏感度，更多的关注业务价值。

领域知识

为了能够更好地关注业务价值，测试人员需要更多的获取相关领域知识。可以关注行业、领域相关文章等资料，也可以关注行业官方网站信息来增加领域知识的获取。

分析性思维

业务价值驱动的测试有很多分析工作要做，对分析能力要求较高。测试人员要注意分析能力的培养，善于分析缺陷、终端用户数据等。

沟通与表达能力

从业务负责人那里了解到业务目标，将业务价值传递给

团队，其中的沟通与表达能力是极为关键的，测试人员要注意这方面能力的培养。

06 小结

以业务价值驱动的测试，主要的是思维方式的转变，各个环节要加强对业务的关注，将业务价值融入测试流程中的每一步，帮助团队达成对业务价值的清晰理解，让团队能够做正确的事情，并且持续正确地做事，以实现真正的质量内建。

QA 在生产环境下做什么？

01 一个生产环境 bug 的应对策略

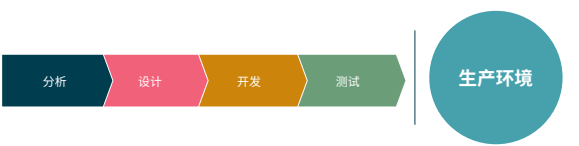
先来跟大家分享一个生产环境下的 Bug：

> 一个在线订购葡萄酒的系统，订购流程相对复杂，下单过程中后台会有随机的失败，系统采取的措施是重试，就是说顾客下单后，后台如果有错误就会不停地重试，直到成功，这个过程对顾客是不可见的。这听起来没什么问题，用户体验似乎也不错。
>
> 后来有个新版本上线了，顾客下单后还是会有随机失败，系统依然不停地重试，但这次不一样的是有的随机失败，下单却能够成功，这样就导致用户虽然只订购一箱葡萄酒，由于后台的多次重试都成功了，用户最终拿到的可能是好几百箱葡萄酒。

这是一个非常严重且昂贵的 Bug！对于这样的问题，作为 QA，你能想到的解决办法有哪些？

瀑布式 QA 流程

瀑布式软件开发模式下，测试是计划驱动的，基本是根据需求文档进行验证，测试的目的就是找到尽可能多的 bug，而且测试阶段处于开发阶段结束之后的一个相对独立的阶段，测试结束之后发布产品到生产环境。基本流程如下：

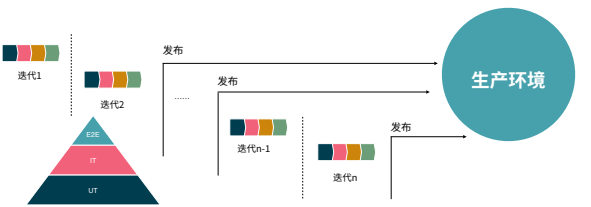


对于上述葡萄酒订购系统的 Bug，通常的办法就是加强在测试阶段的测试保证，除了验证系统功能跟需求文档的一致性以外，还可以增加一些 ad hoc 测试，确保发布到生产环境的系统尽量的稳定。

敏捷 QA 流程

敏捷测试则提倡尽早测试、频繁测试，QA 要从需求分析阶段就开始介入，QA 参与从需求到发布的整个生命周期中各个阶段。在整个 QA 的过程中，会依据敏捷测试象限，在不同阶段采取不同的测试；还会根据测试金

字塔的分层指导，加强自动化测试，制定有利于项目质量保证的测试策略，同时也会做探索性测试，尽量去发现更多不易发现的缺陷。敏捷 QA 的流程如下：



对于葡萄酒订购系统的 bug，处理方式也无非就是加强上线前各个阶段的测试，提高发布到生产环境的产品质量。

除此之外，我们还有什么处理办法呢?

上面两种开发模式下，QA 的重点都是放在发布前的预生产环境的质量保障上，而较少考虑生产环境的系统质量。随着敏捷开发和持续交付的出现，QA 角色逐渐转变到需要分析软件产品在生产环境下的质量。这需要引入产品系统的监控 , 制定检测紧急错误的警报条件，持续质量问题的确定以及找出在生产环境下使用的度量以

保证这种方式可行。

回到前面葡萄酒订购系统的那个 Bug，如果在生产环境下设置监控预警，对于一个订单购买葡萄酒数量异常的情况进行监控，将能及时发现 bug，并且比较容易在损失发生之前及时阻止产生更严重的后果。

02 生产环境的特点

为了更好的实践生产环境下的 QA，先来分析下生产环境有哪些特点：

1. 真实、不可破坏

生产环境都是真实用户在使用，是真正的支持企业业务运转的系统，不可以随意在生产环境去做测试，尤其是破坏性的测试。

2. 基础设施差异

生产环境往往有着比预生产环境要复杂和健全的基础设施，可能会出现预生产环境不能预测到的缺陷和异常。

3. 系统复杂度

生产环境需要与多个不同的系统集成，系统复杂度会比

预生产环境高很多，这种复杂的系统集成很有可能导致一些意外的情况出现。

4. 数据复杂度

生产环境的数据量和数据复杂度也是预生产环境无法比拟的，通常都不是一个数量级的数据，容易引发性能问题、或者一些复杂的数据组合导致的问题。

5. 用户行为千奇百怪

生产环境的用户分布广泛，使用习惯各种各样，导致用户行为千奇百怪，某些使用行为可能就会产生意想不到的问题。

6. 访问受限

生产环境由于是真实业务线上的系统，对安全性和稳定性要求更高，服务器的通常不是所有 QA 可以随便访问的，这种访问受限的情况对于生产环境的一些缺陷的排查带来了很大的不便。

7. 真实的用户反馈

真实用户在生产环境使用，能够提出一些真实而重要的

反馈，但是开发团队往往不能直接接触终端用户，QA 也就没有办法获得第一手的用户反馈，这些反馈常常需要通过支持团队的转述。

生产环境的这些特点决定了 QA 在生产环境不是想做什么就能做什么的，原来预生产环境那套质量保证理论和方法都行不通了。

生产环境下的 QA 有这么多挑战，听起来是不是很沮丧？不要着急，针对生产环境独有的特点，一定能找到相应的解决方案。接下来，咱们一起来看看生产环境下（不仅是 QA）可以做什么。

03 QA 在生产环境下可以做什么?

生产环境的这些特点决定了 QA 在生产环境不是想做什么就能做什么的，原来测试环境那套质量保证理论和方

法都行不通了。要实现生产环境下的 QA，有以下三个方向：

- 直接在生产环境测试
- 监控预警
- 用户反馈收集

1. 直接在生产环境测试

前面讲到不能随便去操作生产环境下的系统对其进行测试，怎么还能直接在生产环境测试呢?

的确不是像在测试环境那样测试，有一些限制和特定技术的采用。

对于只读操作和可隔离特性的测试

在不影响真实用户体验的情况下，可以对某些功能进行

只读操作，需要选择用户使用频率较低的时段进行，不能进行大数据了的只读操作，以免影响系统性能。

对于某些特定的特性，如果比较方便隔离的话，可以在生产环境利用独立的测试数据对其进行测试。

蓝绿部署（Blue Green Deployment）

蓝绿部署是一种通过运行两个相同的生产环境“蓝环境”和“绿环境”来减少停机时间和风险的技术，虽然不能算是真正意义的直接在生产环境测试，也是非常典型的实践之一。

在任何时候，只有一个环境是活的，活的环境为所有生产流量提供服务。通常绿环境是闲置的，蓝环境是活的。部署新的版本到绿环境，可以先进行测试，而不会给真正在使用的蓝环境带来影响。完成部署和测试以后，再进行蓝绿环境的切换。

此技术可以消除由于应用程序部署导致的停机时间。此外，蓝绿部署可降低风险：如果新版本在绿环境上发生意外情况，可以通过切换回蓝环境立即回滚到上一版本。这样就有机会提前发现新版本部署生产环境可能引起的

问题。

金丝雀发布（Canary release）

不同于蓝绿部署有两套服务器，金丝雀发布，也就是我们常说的灰度发布，只有一套服务器，是通过先部署新版本到其中部分服务器，并测试验证，没问题再推广部署到更多服务器的发布流程。

2. 监控预警

前面讲到不能随便去操作生产环境下的系统对其进行测试，但是可以通过监控的方式去获得我们需要的信息，对异常情况进行预警。提到监控预警，不得不提大家都了解或者至少听说过的日志和网站分析工具，这两者都是做好生产环境下的 QA 非常有帮助的工具。

日志

日志就像是飞机上的黑匣子，可以记录系统运行的各种信息，包括错误、异常和失败等。一旦程序出现问题，记录的这些信息就可以用来分析问题的原因；同时可以利用记录的日志设置好预警，提前预测系统可能出现的

问题。生产环境下日志所提供的监控和预警信息，将有利于：

- 阻止不良后果，避免大的损失：前面提到的葡萄酒订购系统就是一个很好的例子；
- 发现潜在的性能问题：通过对日志进行分析，可能发现还没暴露出来的性能问题，提前修复；
- 指导预生产环境的测试：通过生产环境下日志的分析，可以看出哪些功能易出什么样的错误，从而相应调整测试策略，加强预生产环境的相关功能的测试。

网站分析工具

网站分析工具根据具体工具不同，所能记录信息也有差异，但基本都可以记录如下信息：

- 用户使用的操作系统、浏览器等信息
- 用户行为习惯，包括使用的时间、关键路径、关键业务流程等
- 用户所在地区、语言等区域信息

- 用户访问量，请求的响应时间，网站性能等

利用网站分析工具收集到的以上数据，可以帮助我们调整预生产环境的测试侧重点，指导预生产环境的测试；根据用户行为习惯等信息，还可以帮助优化产品业务价值。

3. 用户反馈

介绍生产环境特点的时候提到一点就是生产环境能够收到真实的用户反馈，这是非常有价值的，要做好生产环境下的 QA 一定要利用这些反馈。由于用户行为和习惯的千奇百怪，用户提供的反馈也可能是各种各样的，为了更好的利用它们，需要一个严格的 Triage 的过程，对所有反馈进行分类并相应处理。用户反馈，可以总结为下面几类：

缺陷

用户在使用过程中，系统所出现的问题，并且能够稳定重现的，我们定义为缺陷，缺陷通常会影响到功能的使用，是需要修复的，根据优先级和严重程度，安排相应的修复计划并对其进行修复，同时还需要对修复的缺陷

增加（自动化）测试，以防被再次破坏。

除了能够稳定重现的缺陷，有时候还会有一些随机的失败（比如前面提到的葡萄酒订购系统的 bug）或者难以重现的问题，这类问题很难找到原因，但是又给用户带来很大的不爽。其实，它们通常也是一些缺陷引起的，只是根因可能隐藏的比较深，对于这种问题的处理办法就是前面部分提到的可以添加日志对相关功能进行监控和预警，利用日志协助找到问题根因并进行修复。

抱怨

用户在使用系统过程中由于行为习惯、网络环境等差异，总会有各种抱怨，一般以非功能性的问题居多，比如易用性、性能、可维护性、可扩展性等，当然也有可能是某个功能设计的不能满足真实的用户需求。需要对所有抱怨进行分类，确定是非功能的缺陷，还是新的需求。用户的抱怨有可能千奇百怪，不是所有的都能满足，需要根据分类定义优先级，确定哪些是需要改进和修复，从而采取相应的措施。

建议

除了反馈问题和提出抱怨，用户也会对系统使用方面提一些建议。但一般用户很难提出好的建议，要想收集到有价值的信息，需要提前设计好问卷进行问卷调查，有针对性的去收集；或者对一些重要用户进行访谈，或者发布一个简单的新功能收集用户的使用建议等。然后对收集到的建议进行分析，确定可行性，并将可行的建议应用到后续的系统和业务流程的改进中。

利用用户反馈，改进系统功能，可以优化业务价值，扩大产品的市场影响力，提高企业的竞争力。常被用来收集用户反馈的实践有：

- **Beta 测试：**很多有前瞻性的网站或应用会发布新功能给特定用户组（Beta 用户），收集用户使用新功能的统计数据；
- **AB 测试：**同时发布两个不同版本的系统到生产环境，并将用户引导到两个版本，统计使用每个版本的用户数据；

- **观测的需求（Observed Requirement）：**发布一个简单的功能，或者发布一个 MVP 版本，观察用户使用情况，从而引出并收集到用户的真实需求。

04 生产环境下的 QA 有什么特点

1. 不同于预生产环境的 QA

生产环境的特点决定了生产环境下的 QA 是跟预生产环境的 QA 不同的，不是简单地去测试生产环境的系统，而是通过设置监控条件，收集用户使用系统的反馈，对反馈进行分析并改进，从而让产品质量获得提高。因此，生产环境下的 QA 并不是预生产环境 QA 活动的简单后延，它有着自己独特的特点。

直接在生产环境测试、监控预警、收集和分析用户反馈并不是 QA 能独立完成的，需要与不同角色协作，QA 在整个过程中主要充当分析者和协调者的角色，对生产环境下的质量保证工作起到至关重要的作用：

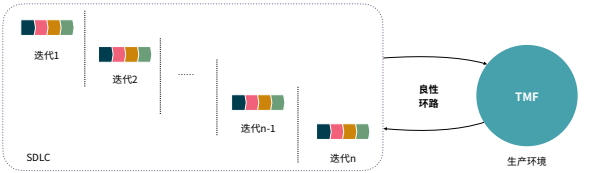
- 跟开发人员一起讨论监控标准，把日志标准化的要求融入到软件开发流程中，确保日志能够记录到真正需要记录的信息。

- 跟运维团队一起分析收集到的统计数据，指导和优化各个阶段的测试，以预防和减少系统在生产环境下的缺陷。
- 跟业务分析人员一起分析和梳理从生产环境收集到的需求相关的反馈，提炼出合理的需求，优化业务价值。

2. 不能独立存在

做好质量内建、做好正向的测试和质量保障工作是软件质量保障的基础，只有在扎实的基础上去做生产环境的 QA 才能带来更大的收益。

生产环境下的 QA 所设置的监控标准是根据系统的行为特点和在预生产环境下的表现来定义的，生产环境下各项反馈的分析结果反过来又影响着预生产环境的 QA 过程，而且这两者是相辅相成的，只有形成了良性环路，才能把生产环境下的 QA 做好。



生产环境下的QA不能独立存在

3. 有别于运维人员的工作

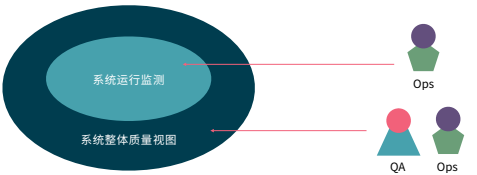
生产环境设置监控预警和收集用户反馈不都是运维团队可以做的事情吗？还要 QA 参与干什么？

传统运维人员所做的工作主要是监控生产环境应用运行的状态，而生产环境下的 QA 是关注生产环境下应用系统的整体质量视图，是收集和分析生产环境下的信息来优化开发和测试过程，提高软件产品的质量、优化业务价值。

因为 QA 有着独特的思维模式和视角，QA 的参与能够帮助更好的分析生产环境下收集到的各种反馈，并且结合对于系统的了解，能够将这些反馈更好的应用到日常的开发工作中。

生产环境下的 QA 让 QA 的工作得到了延伸和扩展，QA

的关注视角更广阔，跟运维人员形成高度紧密合作的关系，一起为生产环境下的质量负责。



生产环境下的QA不同于Ops

4. 跟 APM 的侧重点不同

可能有人会觉得生产环境下的 QA 跟 APM 有相似之处，那么这两者是不是一回事呢?

维基百科这样解释 APM:

“In the fields of information technology and systems management, application performance management (APM) is the monitoring and management of performance and availability of software applications. （在信息科技和系统管理领域，APM 是对软件应用程序的性能和可用性的监控和管理。）”

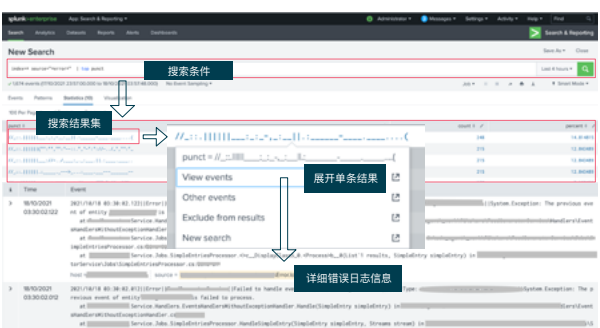
APM 更多的是从性能角度出发去管理和优化应用的可用性，可以发生在各个阶段，不一定是生产环境。生产环境下的 QA 是指在生产环境进行一系列的监控和数据收集，从系统功能、性能、易用性等多个方面进行优化，从而最终优化业务价值。因此，两者是不同的。

05. 生产环境下的 QA 在项目上的实践

我介绍的项目是一个离岸交付项目，采用的是敏捷开发模式，四周一次发布到生产环境，开发的系统包括一个内部员工使用系统和一个外部用户使用的网站，用户遍及全球，项目整个已经持续了十年有余，生产环境具备前面所描述的所有特点，并且生产环境每日的错误日志达到几千条。正是由于错误日志的数量到了一个不能忍的程度，生产环境引起了各方的关注，也使得生产环境下的 QA 迎来了试点的最佳时机。生产环境下的 QA 在这个项目上的实践主要是三部分内容: 日志分析和优化、Google Analytics 数据分析、用户反馈的收集和分析，下面逐个介绍。

1. 日志分析和优化

既然错误日志那么多，首当其冲就是从这些错误日志下手。项目使用的日志分析工具是 Splunk，这个工具功能强大、使用方便，关于工具的详细信息可以参考官网。下图所示为使用 Splunk 通过指定条件搜索日志文件得到的结果页面，结果集里四条类似乱码的东西就是代表有四种类型的错误日志，每种错误出现的数量和百分比都有统计，点击每条结果可以查看详细的错误日志信息。



关于日志的分析和优化，团队在项目上做了如下工作：

分析生产环境的错误日志

每天会有专人负责错误日志的分析，找出其中的优先级

高的问题给团队修复。QA 会协助重现、跟踪并负责测试这些重要的需要修复的问题，通过对错误日志的分析，QA 可以大概的统计出哪些功能特性比较容易产生错误，在接下来的预生产环境的测试中要重点关注。

另外，对于某些功能由于测试环境的数据等局限性，担心部署到生产环境会产生错误或者有性能问题，一般上线后会着重关注这样的功能，除了日常的监控分析之外，还要单独对该功能的日志进行检查，以防产生严重问题。

监控测试环境的错误日志

把生产环境错误日志的分析方法应用于测试环境，对测试环境的错误日志进行监控，尽量把环境无关由功能引起的错误找出来，尽早修复，不让其流落到生产环境。据不完全统计，曾经两个月内通过这种方式发现并修复了好几个潜在的 Bug。

利用日志监控不同功能的性能

Ops 在 Splunk 里设置有专门的统计和监控系统性能的 Dashboard，QA 会定期从里边拿出潜在的存在性能问

题的功能模块，创建对应的任务卡由开发团队来做性能调优。

日志记录标准化

通过对生产环境错误日志的分析，发现现有日志记录存在如下问题：

- 存储位置不统一，有些日志没有办法通过 Splunk 统计分析，造成分析成本的增加；
- 记录格式不一致，有些日志虽然可以通过 Splunk 看到，但是没有记录很有用的信息，比如没有记录错误发生时候的一些有意义的 message，或者是 Post 请求没有记录输入参数，导致排查错误日志的工作增加了难度；
- 日志级别定义混乱，有些没有到达错误级别（error）的日志也记成了错误（error）日志，这也是错误日志数量大的原因之一。

为了让日志分析更轻松、让日志更全面的记录系统的各种情况，针对上面的问题，团队讨论并达成共识：

- 将日志输出到各个服务器的同一个路径;
- 统一日志记录格式，并且尽量记录更全面的信息帮助后期的日志分析;
- 清晰定义各个日志级别（Debug，Info，Warning，Error，Critical，Fatal），将已有的误记成错误的日志降低到正确的级别，对于新功能则严格按照所定义级别记录日志;
- QA 在用户故事（Story）的开发机验收（Dev-box Testing or Desk Check）阶段负责跟开发人员确认相关日志记录是否符合标准，并且通过测试环境的日志监控可以及时验证新功能的日志记录情况。

2. Google Analytics 数据分析

Google Analytics（以下简称 GA）是项目用来统计网站使用情况的工具，从 GA 上可以获取很多有价值的信息，对这些信息的分析是团队实践的生产环境下 QA 的另一块内容，具体做了下面几项工作:

操作系统和浏览器使用情况分析

根据 GA 数据统计分析出用户使用的操作系统和浏览器分布情况，从而相应的调整 QA 用来测试的操作系统和浏览器，尽量让测试环境跟真实用户更接近。比如，前两年客户内部员工使用的浏览器最多的是 IE9，那么我们 QA 的测试也是主要关注 IE9，而后来变成了 IE11，我们重点关注的浏览器也相应调整为 IE11（当然，鉴于 IE9 的特殊性——容易出问题，只要还有用户使用，我们还是需要关注），而对于没有用户使用的 Firefox，我们则不用来测试。

分析 QA 测试跟用户真实行为的差异，及时调整测试根据

GA 上用户访问的路径可以发现我们 QA 的测试跟一些真实用户使用习惯的差异，这样如果按照我们通常的路径去测试，可能有些问题难以在测试阶段发现。比如，QA 在测试环境通常打开“工单”的方式是这样的：

而我们从 GA 上发现真实用户使用过程中，打开“工单”最多的路径是这样的：

发现了这种差异，QA 就要在测试时候做相应的调整，让测试更接近用户的行为习惯。

提炼关键业务场景，增加测试覆盖

一个开发好几年的系统，其功能必然是比较复杂的，由于自动化测试覆盖率不是很理想，过去回归测试需要的投入比较大，而且由于功能相对复杂，又不是很清楚哪些功能对用户来说最为重要，测试很难做到有效覆盖。这种情况对于人员比例低下的敏捷团队 QA 来说，是一件非常有挑战的事情，通常 QA 们在发布前忙得不行。利用 GA 的数据结合客户业务人员对系统各功能使用情况的介绍，我们提炼出来系统最为关键的一些业务场景，并且尽量分层（参考测试金字塔）实现自动化测试覆盖，从而不需要花费太多的人力去做回归测试，同样可以保证主要的业务流程不至于被破坏，而 QA 则可以有更多

的时间和精力去做探索性测试等更有价值的事情。

发现用户较少使用的功能，优化业务流程

关键场景就是用户使用频率较高的功能，类似的，我们还可以通过 GA 发现一些用户较少使用的功能，这可能的原因是不符合用户真实行为习惯，喜欢用其他路径去完成同样的事情，或者不符合真实业务需求，也就是说真实的业务环境下根本没有要用到这个功能的地方。对于这种功能，首先从 QA 角度来讲，我们的测试基本不需要太多去关注，如果有相关功能的缺陷，它的优先级也很低很低；另外，可以跟业务分析人员一起分析下原因，在后续的功能开发过程中可以作为借鉴，从而优化业务流程。

分析用户地区分布和使用时间段分布，合理安排定时任务运行时间

系统用户遍及全球，使用时间段分布也就变得复杂，为了不影响正常使用，有些事情是需要尽量在系统没人使用的时候去做的，比如统计某类数据需要定时执行 SQL

脚本等定时任务。通过 GA 上的数据，统计出哪个时间段是相对空闲的，在不影响真实业务的前提下，把一些资源消耗较大的任务安排在那个时候执行，合理分配资源以减轻服务器的负担。

监控系统性能变化趋势，规避性能风险

QA 定期查看网站平均访问时间，监控性能趋势，同时要重点关注那些访问非常慢的页面或功能，必要的时候创建性能缺陷卡，由开发人员来调查分析并修复或优化。

确保 GA 能够统计到所有需要统计的功能

GA 数据虽然很有用，但前提是正确记录了所需要统计的数据，所以在开发过程中要确保 GA 嵌入到了各个需要统计的功能或页面，QA 在预生产环境测试的时候需要验证这一点。

下图为从 GA 拿到的用户操作行为流和页面平均加载时间的一些数据：

3. 用户反馈的收集和分析

作为开发团队的我们基本是不能直接接触到系统终端用户的，直接接受反馈的是我们客户的 Ops 团队，QA 主要通过下面几个途径去协助分析和梳理用户反馈：

跟 Ops 和业务的定期沟通会议

QA 会定期跟客户的 Ops 和业务人员沟通，了解用户对于现有系统的反馈，找出在测试中需要重视的功能特性，将预生产环境的测试关注点做出相应的调整。

培训 Ops 人员

指导和协助客户 Ops 人员利用鱼骨图（Fishbone

Diagram）的方法对收集到的用户反馈进行分析和分类，将分析结果跟现有的测试覆盖情况进行对比，找出测试过程的薄弱环节，并做出改进。比如，如果某个浏览器出现的 bug 比较多，而我们的测试则要加强该浏览器的关注；或者是因为不同用户权限导致的问题出现频率高，那就得在测试中注意覆盖不同用户角色的测试，反之则减弱不同用户的覆盖，主要测最常用的那类角色即可。

调查和跟踪生产环境 Bug

帮助重现和调查用户反馈过来的生产环境 Bug，并负责跟踪修复和验证；对于难以重现的问题，则添加日志监控，通过分析收集到的日志信息找出问题根因，从而进行修复。

协助梳理业务需求

系统要增加某个新功能的时候，客户 Product Owner（以下简称 PO）或其他业务人员跟用户会一起沟通该功能相关业务现有的使用习惯、使用场景，QA 也会尽

力参与这种会议，收集用户第一手需求信息，这些信息对于后期该业务功能开发时候的 QA 过程将非常有帮助。而还有些功能，PO 可能一时也拿不定主意要做成什么样子，我们会发布 MVP 的版本给用户使用，QA 协助业务人员分析用户使用后的反馈，梳理更具体的用户需求。

总结

- 生产环境下的 QA 将 QA 的工作范围扩大到从需求到生产环境，增加了更多的反馈来源，跟持续交付结合，可以帮助持续提高产品质量、持续优化业务价值。
- 生产环境下的 QA 给 QA 的工具箱添加了更多的工具，提供了更多评估和提高系统质量的选项，是 QA 们值得深入研究的话题。
- 生产环境下的 QA 不能走的太远，必须先做好预生产环境的质量保证，并且仅适用于持续交付实践的比较好的组织，如果连预生产环境的质量保证都做不好，而就想着去做生产环境下的 QA，那只能是舍本逐末、事倍功半。

深入分析生产环境上的缺陷

生产环境出现缺陷往往会比较麻烦，毕竟那是真实用户使用的环境，需要尽快修复，但是修复后可能涉及到部署等一系列的问题。因此，大家都希望生产环境的缺陷越少越好。但软件系统的生态环境越来越复杂，不确定性增加，还是很难避免在生产环境会有缺陷产生。那么，一旦生产环境产生缺陷，除了紧急修复以外，还能做点什么呢？

其实，利用生产环境下的缺陷数据，对其进行深入分析，根据分析的结果来指导软件开发过程，以避免同样问题再次出现，也是生产环境下的 QA 非常重要的一个实践。

01 利用 5 WHY 法分析缺陷

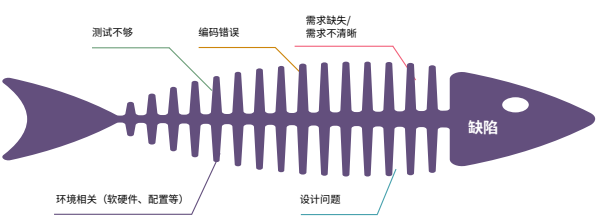
5 Why 分析法

所谓 5 Why 分析法，又称“5 问法”，也就是对一个问题点连续以 5 个“为什么”来提问，对其进行深入分析。根据实际情况，问题的数量不一定要限制在五个，可多可少，适当调整。

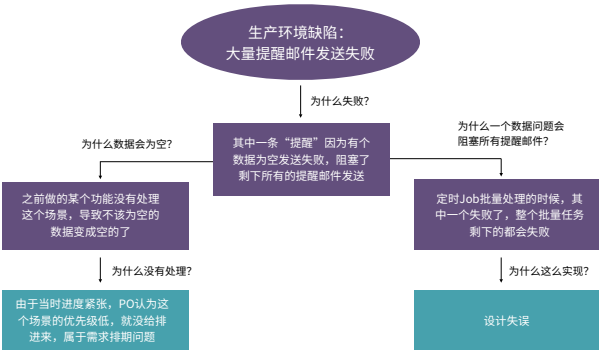
分析根因

缺陷深入分析的第一步就是找到引起缺陷的根本原因，其原因可能有下列这些方面：

- 需求缺失或者需求不清晰
- 设计问题
- 编码错误
- 测试不够
- 环境相关（硬件、软件、配置等）



利用 5 Why 分析法根据缺陷的表象，多问几个为什么，逐步分析，最终就能找出根因。下面是一个真实生产环境缺陷的根因分析过程：

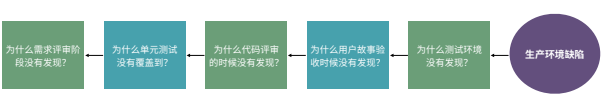


上图的根因还可以继续细分，比如为什么这么设计，可能还会有更深层次的问题；同样的，进度紧张导致的需求没有实现，也可能还有更详细的内情可以分析的。最好能一直分析，直到找出真正的根本原因。

定位阶段

找出根本原因之后，同样利用 5 Why 分析法，基于软件开发生命周期由后往前问“为什么在该环节没有发现 / 预防这个缺陷”，从而定位引发问题的薄弱环节，找出是哪个环节做的不好导致的问题。拿生产环境的缺陷为例，下面是可能（不限于）的问题列表：

1. 为什么在预生产环境没有发现这个问题?
2. 为什么测试环境没有测出这个问题?
3. 为什么 Desk Check 没有发现这个问题?
4. 为什么 Code review 没有发现这个问题?
5. 为什么单元测试没有覆盖到这个问题?
6. 为什么在需求评审的时候没有发现这个问题?
7. ……



02 缺陷分析是为了更好地预防缺陷

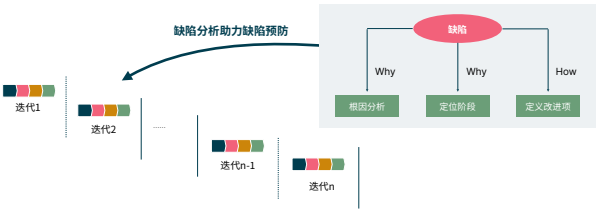
分析缺陷根因，找出引起缺陷的薄弱环节，这不是目标。在找出根因和薄弱环节之后，更关键的是进一步确定改进措施、定义改进项。

不同的根因对应的可能改进项有：

根因	改进行动项
需求缺失、不清晰	• 需求分析阶段更早地组织 QA 和技术负责人一起讨论 • 加强需求评审 • 更详细地了解现有系统的功能细节 • ……
设计问题	• 加强技术方案的讨论 • 架构师对设计方案的支持 • 团队一起学习和研究不同的技术框架 • ……
编码错误	• 更加有效的结对合作组合 • 加强代码评审 • 增加单元测试覆盖，QA 参与单元测试评审 • ……
测试用例不全	• 测试用例评审 • 从用户角度考虑使用场景 • 设计用例的时候充分了解对系统相关功能的影响 • 考虑系统遗留数据 • ……
环境问题	• 开发环境和测试环境尽量跟生产环境基础设施接近 • 统一管理各个环境的配置信息 • 尽量模拟生产环境数据来测试 • ……

改进项的执行需要把职责落实到人，需要有人来负责这个事情的发生，并且后续还需要持续检查改进项的执行情况，以及通过统计缺陷重复发生情况来验证改进项的适用性。这样就可以较为有效地减少同类问题再次发生

的可能性，帮助做到缺陷预防。



当然，通过缺陷分析来预防缺陷只是实践之一，还有非常关键的正向实践不可忽视，比如：测试左移、每个环节的持续测试等。

大规模团队 QA 的合作

在规模较小的敏捷团队，敏捷实践开展起来会比较容易，QA 的工作以及不同角色间的合作会顺畅很多。对于几十个人组成的大团队，合作起来可就没那么简单。

我曾经历过一个项目，80 人左右的团队规模，大家共同开发一个产品，工作在同一个代码库上，项目根据负责开发的不同特性将大团队划分成多个特性小组，每个小组是一个敏捷开发团队。

QA 都被分散在每个特性小组，不像在小团队那么简单，沟通的成本要高很多。如果没有及时沟通，将会造成信息不对称，要补救所花费的额外精力是比较大的。

另一方面，QA 在特性小组基本都是单兵作战，沟通协作不够的话，对于自身的成长也不是很有利。

为了解决知识共享和能力提升的问题，我们决定在项目组内部成立 QA 社区，这是我们首次开启的 CoP（Community of practice）的尝试，事后证明还是很有效的，并且后面成立了其他几个不同的社区，取得了不错的效果。

01 知识共享

知识共享首要的是要解决项目上下文缺失的问题，我们在项目组先是尝试了如下两个实践：

- 集体参加用户故事的启动（Kickoff）和验收（Desk Check）

不管哪个组有用户故事的启动和验收，各组的 QA 都一起参加这样基本能搞清楚各个团队都在开发什么功能。

这个实践在团队规模还只有两三个特性小组的时候就有开展，小组少的时候，这个方法是勉强行得通的。但随着特性小组的增多，在高峰期的时候，QA 可能一天都在这两个活动中切换，精力几乎都耗费在这上面了。因

此，通过这种方式熟悉业务上下文并不是一种高效的方式，只好放弃。

- 集体参加各组的特性启动会议（Feature Kickoff）

一个发布周期每个组开发的特性数量不会太多，一般是一到两个，这样，QA 参加特性启动会议的次数也不会太多，感觉是可行的。

但尝试一段时间发现，由于各组的发布周期是一样的，各组的特性启动会议基本都发生在上个发布周期末期或者新发布周期的初期，而这个时候 QA 正是忙着上验收测试的收尾阶段，同时也在忙着参与自己所在特性小组需求分析（用户故事评审等）的时候，一般都比较忙，随着特性小组数量的增加，QA 也顾不上去参加所有其他组的特性启动会议了。

前面两项实践后来变成了根据自身情况自愿参加，但事实上就变成了因为手头工作忙不过来都放弃参加其他组的了。不过，大家始终坚信信息共享的重要性，于是 QA 社区在不断摸索中找到以下两种实践。

- 定期沟通例会

每周一次的 QA 定期沟通例会，时长为一个小时，要求所有 QA 合理安排手头工作，务必参加。由于这个例会是长期固定时间的，小组内的其他会议都需要避开这个时间，以免冲突影响 QA 的参会。

例会上每个 QA 给大家介绍自己组内的特性上下文，更新状态，把遇到的困难、可能的风险等拿出来跟大家一起讨论，找到解决方案或规避举措。同时，对一些需要 QA 社区一起完成的事情，找到专人负责推进，下次例会的时候检查执行情况。

- QA 内部功能演示

在每个发布周期，每个 QA 在 QA 社区内部做一次功能演示，共享组内新开发功能特性。这样不仅锻炼了 QA 做好功能演示的能力，同时也给其他小组的 QA 共享了业务功能信息。

这两项实践开展以来，QA 们感觉到的收益还是蛮大的，除了知识、信息共享之外，也增进了 QA 间的相互了解，给大家一起合作带来很大的帮助。

02 能力共建

除了项目上下文共享之外，QA 社区还有一个更重要的任务是 QA 的能力建设。主要有以下几个方面：

明确目标

QA 社区根据项目需求，结合技术发展趋势，确定季度或者半年的能力提升重点。在大方向下，每个 QA 根据自己的兴趣，可以有自己的关注点。

项目实践为基础

能力提升能够在项目实践的基础上实现是最理想的。

以满足项目需求为主，在做好项目质量改进的过程中，提升自己的能力。QA 们可以独自或者结对的方式来负

责某些改进举措，社区会统一跟踪每个举措的进展，利用例会的时间交流遇到的问题、一起寻找解决方案，或者在例会的时候分享实践过程中的一些收获等。

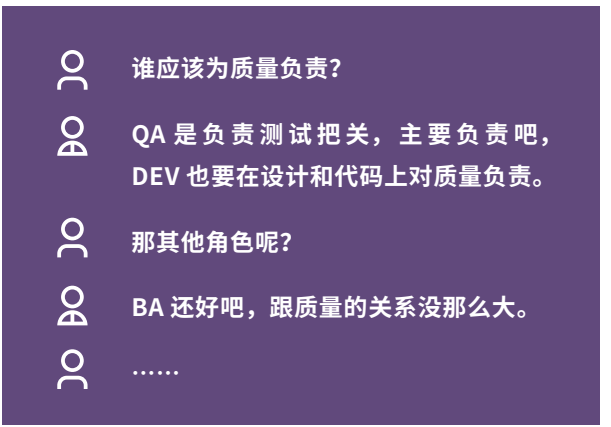
技术调研及分享

在开展某个新实践之前，往往需要对相关方案、技术进行调研，另外，除了能在项目实践的技术外，还有一些趋势性的技术也是需要了解和学习的。因此，将这些调研或学习结果给其他 QA 分享成为一项非常重要的事情，是大家一起进步的关键。

社区外的连接

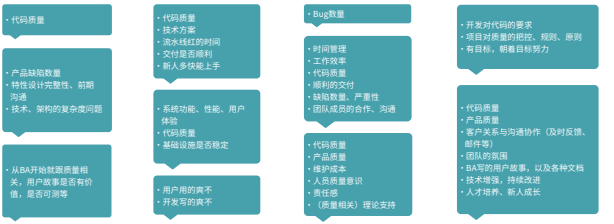
成立某个社区，很容易变成社区单打独斗，自己玩得很投入，忽略了跟外界的沟通，这是不理想的。因此，保持跟社区外的连接非常关键，也就是需要加强跟项目上其他角色的沟通和讨论。我们定期开放例会邀请非 QA 角色参加，或者把 QA 社区内的成果在项目级别上进行分享，这样，不仅扩大了 QA 的影响力，同时也获得了更多来自外部的反馈，更有利于 QA 团队的成长。

团队为质量负责



曾经在某个项目，感觉大家对质量的关注意识有些欠缺，我们 QA 在项目上的不同角色、不同工作年限的人之间采样做了一次访谈，以更好的了解团队质量意识欠缺的原因，上面这个问题就是其中访谈的问题之一。有同事曾提醒我说这种题就是送分题，肯定不会有人回答不出。可是，事实并非如此……

为什么会这样呢？另外一道题给了我们答案，那就是“大家理解的质量是什么？”同事们的回答如下图所示：



因此，大家对质量的理解不一致，在没有搞清楚什么是质量的前提下，当然也没有可能理解到底谁该为质量负责。

因此，我们来看看质量到底是什么？

01 质量是什么？

> **“产品质量不是检测出来的，从产品生产出来后质量就已经在那了。”**
>
> **——著名的质量管理专家戴明**

在讲什么是质量之前，我们有必要区分两个不同的概念：测试只能检测、发现缺陷，而质量要通过缺陷预防来实现。 测试与质量不可同日而语，以后再也不要说“上

线这么多问题，测试是怎么测的”这种话了。

那么，质量到底是什么？对于不同的角色、不同的利益相关者，质量有着不同的含义。总的来说，可以分为外部质量和内部质量两种。

1. 外部质量：

顾名思义，外部质量就是软件呈现给用户的外部形态，是否有缺陷、是否稳定、是否有性能问题等。也就是最终用户在软件使用过程中的各种体验，包括软件可学习、高效、不易出错、有用、难忘等特性，都属于外部质量范畴。外部质量也可以称为使用质量，主要是从使用软件的用户角度来看的。

外部质量能够被用户直接感知到，直接影响用户的使用，因而显得特别重要，客户 / 用户一般也比较容易为外部质量买单。

2. 内部质量：

同样的，内部质量就是指软件系统内部的质量状态，包括代码的效率、结构、可读性、可扩展性、可靠性和可

维护性等。内部质量主要从开发人员角度来看，也称为代码质量。

内部质量不会被最终用户感知到，不容易被客户 / 用户买单，也常容易被团队忽略。但是，内部质量会影响外部质量，需要团队引起重视，加强设计、开发等环节的质量把控。

3. 内建质量：

质量不能被检测出来，要提高软件产品的内、外部质量，都需要通过质量内建（或质量内嵌）的方式，做好每个环节的质量保障工作。质量内建包含自动化测试和手动测试：

- 自动化测试：从多个层次（单元、组件、端到端等）写自动化测试，并将其作为部署流水线的一部分来执行，每次提交应用程序代码、配置或环境以及运行时所需要软件发生变化时，都要执行这些测试。同时，随着项目业务和技术架构的演进，自动化测试策略也需要随之调整、不断改进。

- 手动测试: 手动测试也是很关键的部分，如需求验证、

演示、可用性测试和探索式测试等，在整个项目过程中都应该持之以恒的做下去。

另外，质量内建不仅要考虑功能测试，对于非功能测试同样是需要做到内建的，比如安全内建、持续性能测试等。

软件构建过程中多大程度上做到了质量内建，有多少缺陷做到了提前预防，这是“内建质量”。内建质量虽然跟内、外部质量不在一个维度，但也是体现质量好坏的一个方面，在此也把它作为衡量质量的一个维度，测试或使用过程中发现的缺陷数量可以作为度量指标。

因此，我们可以从这三个维度来度量软件产品的质量，可以通过以下方式来度量：

外部质量：用户反馈、用户报告的问题数量

内部质量：代码评审、结对编程、静态代码质量检查

内建质量：测试环境、生产环境缺陷，生产支持的反馈

了解了三个维度质量的含义，我们不难理解：

✗ 质量不是零缺陷，不是百分百的测试覆盖率，也不是没有技术债；

✓ 质量是快速反馈，任何改变能够快速验证，并且快速修复；

✓ 质量是把精力都集中在正确的事情上；

✓ 质量是团队在代码、设计和交付上有信心做出改变；

✓ 质量是团队对任何改变负责。

02 容易忽视的质量

从前面对质量的定义，广义的质量其实包括软件产品交付流程中的方方面面，每个环节的一点疏忽都可能对软件质量造成不同程度的影响。下面列举一些从项目上经历的对质量关注有所欠缺的内容：

- 需求分析过程仓促，或者参与人员角色比较单一，导致业务上下文了解不够，关键场景缺乏考虑等；
- 忙于交付更多的 feature，忽略了对代码质量的关注，该重构的没有重构，在原本不太健康的代码基础上继续增加更多的代码，导致混乱，筑起高高的技术债；

- 没有足够的测试覆盖，导致新增代码容易破坏已有功能；
- Dev 提交代码后，就投入新代码的开发，对所提交代码缺乏关注，CI pipeline 红了不能及时修复，可能影响后面 QA 的测试进度；
- 大面积的重构发生在 release 前夕，无法全面回归，带来很大的风险；
- 项目初期只考虑少量用户的场景，随着业务的发展，系统功能难以扩展，导致严重性能瓶颈；
- 技术选型失误，开发困难，没有及时改进，一错再错，最后问题严重到无法弥补；
- 第三方组件评估不够充分，导致线上环境无法承载等；
- 开发缺少对异常情况的处理，测试过程缺乏探索，只覆盖到主干路径，边角 case 可能引发问题；
- 开发或者测试都只考虑当前功能模块，缺乏更广范围的考虑；

- 缺乏跨功能需求的关注，导致严重的安全或者性能问题；
- 对线上环境了解不够，而且没有足够日志信息记录，线上问题难以定位，导致宕机时间过长；
- ……

这样的问题还有很多很多，无法一一枚举。每个角色，每个环节都有可能出现纰漏，导致产品质量问题。

那么，谁该为质量负责是不是已经很清楚了？

03 谁该为质量负责？

快速启动	迭代 0	迭代 1~n	上线后
用户故事	完成的定义（DoD）	验收条件（AC）	生产环境下的 QA
质量策略		测试驱动开发（TDD）	生产环境支持（support）
		探索式测试	用户测试
		结对编程	
		用户故事启动	
		用户故事验收	
		持续集成 / 交付（CI/CD）	
		演示（Showcase）	
		回顾会议	

在敏捷模式下，质量是贯穿项目整个生命周期的非常关键的部分，质量保障工作自然也是需要在每个环节有所体现。

上表列出的各个实践活动都与质量有关，每个活动都要求多个不同的角色同时参与。

下面从敏捷团队三个主力角色 BA、QA 和 Dev 的不同视角来看看各自怎么为质量负责。

BA：Busines Analyst，业务分析师

BA 主要负责业务需求的分析工作，要理解客户的业务，并将业务分解成便于 Dev 和 QA 理解的功能点，同时，还要能够帮助用户验证开发完成的软件系统功能，并展示给客户。需求作为软件开发的源头，是极为关键的。

BA 视角的质量，主要是需求分析的准确性和清晰度，要帮助团队对需求有一致的认识，从用户旅程的角度关注交付给最终用户的产品是否真的带来了价值。

Dev：Developer，开发人员

Dev 作为软件系统实现的主力，对质量内建是至关重要

的。从需求的理解、整洁的代码实现、测试覆盖率的保证、频繁的代码提交、持续的集成、对生产环境的关注、运维的支持等方面，都有着不可替代的职责，每个环节都不可忽略。

因此，Dev 视角的质量不仅是按照需求实现功能的开发，还要把功能高效的交付给最终用户。

QA：Quality Analyst，质量分析师

QA 作为软件质量的倡导者，是唯一一个全流程都要参与的角色，从需求到上线后的支持，每个环节都不可缺。清晰理解需求、制定质量保障策略、做好各个环节的测试工作（手动、自动化、探索式、跨功能、非功能测试，以及生产环境的 QA 等）、关注项目整体质量状态、及时反馈质量信息给团队、识别业务风险和优先级、帮助优化业务价值，这些都是 QA 的职责。

三个主力角色中的 BA 一般都会有从用户旅程的角度去考虑，其实 Dev 和 QA 也需要同样的思维模式，不能把 story 或者 AC（Acceptance Criteria）割裂来看，而是要从整体的用户旅程的角度、端到端的去考虑需求的

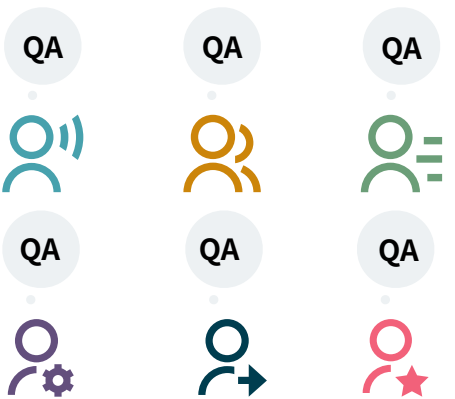
实现和测试工作。

除了三个主力角色，团队还有其他角色也都是要对质量负责的，比如：架构师要负责项目架构的健康，基础设施负责人要管理和维护好基础设施以保障开发和运维工作的顺利开展，PM 要管理好团队的交付节奏、团队成员的工作状态、客户的满意度等，这些都是跟质量相关的。

04 敏捷团队需要专职 QA 吗

既然团队整体对质量负责，肯定有人会有疑问：那敏捷团队还需要专职 QA 吗？

这是个有意思的问题，业界对于“去 QA 化”的讨论也是一波又一波的。甚至很多 QA 听到这个很是担心，觉得自己前途渺茫了。



其实，并没有必要那么纠结是否需要专职的 QA，因为 QA 在团队所需要做的事情始终都是要做的，不管是不是有专职的 QA，从团队的角度来讲，有人做那些事情不就 ok 了？我们可以把 QA 理解成一顶帽子，不管哪个角色的人戴上这顶帽子，能完成 QA 的工作就可以了。

但是，是不是什么角色的人都换个帽子戴上就能做 QA 的事情了呢？这个我相信大家还是比较清楚的，术业

有专攻，肯定是专业的 QA 会做的更好。如果没有专职 QA，对团队其他角色的要求就要高很多，不是每个团队都可以做到的。

05 能够整体对质量负责的团队有哪些特征

除了前面讲到的清晰的质量目标、注意容易忽视的质量、团队成员的充分合作，要做到团队整体对质量负责对团队本身也是有要求的。从我的经验来看，能够整体对质量负责的团队通常有如下特征：

信息共享

团队成员要能够为质量负责，首先得知道质量和质量目标是什么，而质量目标会受到众多因素的影响，每个项目每个阶段都会有不同的目标，及时将一些重要的信息分享给团队的每个人是至关重要的，包括客户的业务动态、生产环境系统的状态、终端用户的反馈、团队的工

作重心、持续改进的状态、交付的压力和进度等等。

信息共享，有利于团队成员更好的发挥主观能动性，更好的真正为质量负责，这是健康团队最关键最重要的一个特征。

反馈机制

对于软件系统的质量，我们提倡持续集成、持续测试以获取及时的反馈；为了团队的健康，需要不断优化团队成员间的沟通与合作，同样需要有相应的反馈机制或渠道。团队需要营造反馈的文化，成员间应是平等的关系，任何人可以对任何人或事提供相应的反馈。

自动化测试提供的反馈有助于软件系统更加健壮，团队内的反馈机制则可以促进团队的健康发展，有助于团队整体对质量负责。

回顾会议

回顾会议是敏捷软件开发中非常有价值的一个实践，但有些团队的回顾会议流于形式，并没有带来什么效果。

好的回顾会议应该做到对事不对人，大家都能就日常工

作中的事情发表自己的意见，做得好的方面继续发扬，需要改进的方面共同商讨应对策略，持续改进。

对于回顾会议讨论出的改进行动，一定要是可落地执行的，并且有相应的人员负责跟踪，不能光有行动的想法，但是并没有实际行动发生。

仪式感

“仪式感就是使某一天与其他日子不同，使某一时刻与其他时刻不同。” 日复一日的例行工作需要仪式感来适当的刺激，以增加团队成员的动力。

项目顺利上线、质量提高了一个档次、某事受到客户的表扬、解决了一个极为关键的生产环境故障等，都可以有一些庆祝的仪式让团队所有成员共享喜悦。

团队活动

除了有仪式感，团队还需要有定期的团队活动，比如一起出去吃午饭、一起去按摩、一起桌游等等，形式可以有很多，就算工作、生活太忙聚不全，一起简单吃个下午茶、聊聊八卦也是挺好的。

这种团队活动的目的主要是为了增进成员的凝聚力，一定要是轻松、愉快的形式，最好能够所有成员都参加。

06 责任流程模型

“敏捷测试强调团队为质量负责，那质量变成是团队的事情，可能有测试人员认为个人不那么负责问题不大，毕竟天塌下来有团队在。”一位转型中的测试经理表达了这样的担忧。

这跟传统职责分明的做法有关，“我”管“我的”，“你”管“你的”，各自做好自己的本分就可以了。现在，既然说要团队来负责，那么“你”就可以多帮“我”负责一点，“我”少负责一点也没问题。听起来似乎还挺合情合理的。

显然，这种想法是有问题的，把个人跟团队割裂开来了。

良好的团队应该有一种“我们”的态度，而不是“我”的态度：“我如何帮助团队解决问题？我可以为别人做些什么？”，而不是“我不知道这是什么问题，这不是我的问题。”

事实上，有前面测试经理提到的这种想法的人不是个例，团队并不是那么容易做到真正为质量负责，因为责任感的培养并不是一件简单的事情。

为什么责任感这么难以培养呢？ Christopher Avery 的“责任流程模型”很好的解释了这个问题。

图片来自：https://responsibility.com/responsibility-process/

上图即为责任流程模型，从下到上需要经历几种不同的心理状态：否认、指责、辩解、羞愧、放弃、义务，然后才能到达最上层的“责任”，才能培养责任感。

指责（LAY BLAME）

当问题发生时，指责心态的人会直接把责任推到别人身上，“这是他的错”。

比如，在用户故事验收时候，发现某个小功能点的实现跟需求不符，这时候开发人员可能会说“那是因为需求没写清楚，是业务分析人员的问题。”

辩解（JUSTIFY）

当发现无法指责别人的时候，就会把问题归因于我们无法控制的外部环境，通过这种方式来为自己辩解：“这就是经济”，“我们的文化就这样”，“这就是团队的管理现状”。

再比如，在软件开发中当同一个问题被多次出现的时候，可能会听到这样的辩解：“我们的遗留系统代码太乱了，出现这样的情况实属正常，很难保证代码每次都能写

对。”

指责和辩解属于从外部去找原因，从而推卸自己的责任。

羞愧（SHAME）

当意识到外部环境必须改变才能使生活变得更美好的时候，再出现问题就会归因于自己的错，会感到内疚。常会有这样的想法：“我应该做的更好”，“我怎么又做成这样”，“这是我应得的后果”。

比如，当某个复杂功能上线出现问题时，他们会想“这个点我之前应该想到的，怎么会漏掉没测呢，我真是个白痴”。

感到内疚和羞愧是好的，但是光是这样不会解决问题，下次还是会有同样的问题出现。

义务（OBLIGATION）

义务不是责任，通常是一些承诺过需要做的事情，是自

己不得不做但不一定喜欢做的事情。这种情况下会对自己的承诺感到抓狂，感觉到没有其他选择了，觉得生活就是一个负担。是一种非常消极的态度。

比如，在开发进度特别紧急的时候，测试人员会有测试压力巨大、根本测不完的情况，但测试人员还是会认为测试就是自己本分该完成的任务，是自己的义务，累的要趴下的时候只会抱怨工作太累了，不会积极的想办法解决根本问题。

羞愧和义务开始从自己身上去找原因，但是还达不到责任的高度。

放弃（QUIT）

有时候，我们会在羞愧和义务之间感觉到非常的抓狂和沮丧，然后就会想逃离，逃离羞愧带来的痛苦和义务带来的负担，也就是放弃。有放弃想法的人会觉得“只要不去关心问题，它们就会自行消失”，其实问题并不会消失，而且沮丧会一次又一次地袭来，从而非常的焦虑。

比如，某个开发人员开发的代码总是出现 bug，但是又找不到改进办法的时候，可能就会觉得无能为力，产生放弃的念头。

放弃是非常消极而负面的态度，很不利的。

责任（RESPONSIBILITY）

经历前面这几个阶段，并没有放弃，最后就会承担起责任，培养责任感，来到责任流程模型的最高层次。

进入这个阶段，会意识到自己是有能力和力量去解决某个问题。遇到需要解决的问题后，会直面问题本身，分析根因，想办法解决问题，而不是去应对自己的沮丧感觉。

在软件缺陷出现以后，解决了缺陷并且对其根因进行分析，找出以后避免出现同样问题的改进办法，并实现改进，做到缺陷预防。这就是责任，就是为软件质量负责的表现。

否认（DENIAL）

还有一个状态是模型右下角的“否认”。否认通常是因为没有意识到有问题，而忽略问题的存在。这是责任流程模型的最低层次。

比如，测试人员跟开发人员说发现了一个 bug，开发会潜意识的反应说：“我代码没问题，我自己测过了。”

否认属于负面心态，但这种心态更多的是因为能力或认知有限所致。

07 培养责任感的三把钥匙

责任流程模型形象地展示了培养责任感需要经历的几个阶段，我们可以清晰地看到培养责任感的不易。Christopher Avery 为此也给出了培养责任感的三把钥匙：动机、意识、面对。

图片来自：https://responsibility.com/responsibility-process/

动机（INTENTION）

首先，当事情出现问题的时候，需要以负责任的动机去面对，积极地想办法解决，而不是忽视、找内外部借口、甚至是逃离问题不予理会。

比如，当我们在软件开发中发现缺陷的时候，不管是什么原因导致，先要做的是积极想办法来解决、分析并做好后续的预防，以减少同类缺陷再次出现。

做一件事情的动机将会决定后续所采取的行动，这个非常关键。因此，端正动机，积极面对问题，是通往责任之路的钥匙之一。

意识（AWARENESS）

其次，是对责任的意识。当出现问题的时候，如果又开始找原因、找借口应对的话，要尽快把自己拉回来，增

强自己面对问题的责任意识。

敏捷团队强调团队整体为质量负责，我们作为团队的一员，都需要有对质量负责的意识，每次代码变更都要考虑是否引入新的质量问题；当发现缺陷的时候要意识到需要一起来想办法解决，作出自己力所能及的贡献。

只有意识到了，才可能采取相应的行动。毫无疑问，意识非常重要，这是通往责任之路的第二把钥匙。

面对（CONFRONT）

以积极的心态面对真正的问题，尝试去发现其中有哪些是可以学习的、哪些是可以改正的、以及哪些是可以提高的。

出错不可怕，可怕的是出错以后不能从中吸取经验教训，同样的错误频出。失败是成功之母，从失败中学习，定会有所成长。

当我们说要为质量负责，就是不管质量出现什么样的问题，我们都能积极面对，找到真正的问题所在，采取积极的应对措施，持续改进。

积极面对问题是一种成长型心态，是一种负责任的心态。积极面对是通往责任之路的第三把钥匙。

08 写在最后

质量不仅是某个角色的事情，团队每个成员都撇不开质量这个话题。团队为质量负责要求所有质量角色都将质量推向看板的左侧，从每个用户故事的开始就将质量融入其中。

软件开发生命周期的每个环节、每个实践活动都不可轻视，只有在每个点上都做好质量的工作，才能实现真正的高质量交付，每个角色对此都有着非常重要的职责。团队为质量负责，就是要做到质量人人有责。

软件测试人员的职业发展之路

测试人员的职业发展，与整个行业和领域趋势有脱不开的关系。近几年，软件测试领域新的关键趋势主要体现在以下几个方面：

- AI 的发展与软件测试
- 敏捷与 DevOps
- 自动化测试
- 环境和数据
- 成本与效能

在这样的趋势下，测试人员的职业发展之路有什么变化呢？我们先来看看测试人员的技术发展方向有哪些。

01 技术方向

基于前面提到的新趋势，测试人员的职责由单一的测试软件系统是否工作、是否满足业务需求变得更加多样化，测试人员可以全流程参与软件开发，让测试活动贯穿软件开发整个生命周期。因此，测试人员的职业发展技术

方向有：

1. 敏捷测试专家

2. 高级测试开发专家

3. 专项测试专家

4. QAOps 专家

1. 敏捷测试专家

敏捷测试强调的是尽早测试和频繁测试，测试人员需要能够从需求分析阶段开始介入，全流程参与，跟整个团队一起实现团队为质量负责。对敏捷测试专家的技能要求有：

领域测试能力：

测试人员需要丰富的业务知识、较强的业务敏感度和业务理解能力，熟悉各种不同类型的业务模式，包括新兴业务 IoT、智能服务、区块链等，能够制定相应的测试策略，有效协助团队做好质量内建，实现交付价值最大化。

自动化测试能力：

自动化测试是敏捷开展的必要条件，自动化测试技能是测试人员的必备技能。成为敏捷测试专家，要求测试人员了解不同的自动化测试框架的优缺点，能够指导项目自动化工具的选型；了解测试分层的思想，能够帮助团队制定合适的自动化测试策略；能够实现业务功能层的自动化测试，能够跟开发人员一起参与底层自动化测试（接口测试、单元测试等）的评审工作；了解持续集成工具，能够在持续集成流水线上配置和运行自动化测试。

沟通协调能力：

敏捷测试要求团队为质量负责，测试人员作为主力，需要承担起质量的分析者和协调者的角色，要求有很好的

跟不同角色沟通和协调团队合作的能力。

2. 高级测试开发专家

高级测试开发专家的必备技能要求有高级自动化测试、白盒测试、开发和平台构建能力，要求有很强的测试代码编写能力，能够自行开发自动化测试工具、搭建自动化测试框架、构建自动化测试平台和服务。

同时，最好还有 AI 应用的基础算法应用能力和自然语言处理技能，需要了解和掌握 AI 相关知识，以及 AI 知识在测试中的应用，以帮助实现自动化测试的智能化。

3. 专项测试专家

专项测试技能集包括安全、性能等跨功能测试技能，需要有扎实的计算机基础知识，了解安全问题的类型、安全测试工具的优缺点，能够提供安全测试解决方案；熟悉性能影响因素、性能测试关注点以及提供性能调优方案等。

专项测试技能也包括测试数据和测试环境的管理，要求熟悉虚拟化、云计算技术、数据匿名化等数据处理技术，

能够提供测试数据和环境管理的方案。

4. QAOps 专家

测试右移已经越来越被重视，这意味着测试活动需要右移到生产环境，需要测试人员跟 Ops 人员更紧密的合作，QAOps 专家也应运而生。QAOps 专家需要了解基础设施相关技术与实践，了解日志管理、日志监控以及日志分析技术，同时还要有用户行为分析能力，通过跟 Ops 的合作，充分利用生产环境的各种类型的信息来优化软件开发和测试流程，以实现最终优化业务价值的目标。

02 管理方向

测试类管理岗位在新的趋势下有些将不复存在，一般在

相对传统的组织架构下才会有，但是目前来看还是有相当的企业是适用的，在此也简单聊一下。根据每个公司的情况不同，测试人员直接相关的管理岗位也会有些不同，大体有如下这些：

1. 测试组长

2. 测试经理

3. 项目测试负责人

4. 测试总监

1. 测试组长

测试组长一般带几个测试工程师， 负责任务分派和人员管理等工作。除了必备的测试技能外，测试组长需要的管理技能有：

任务优先级识别能力：

需要能够识别任务的优先级，并根据当前工作合理分配给不同的人去完成。

培养团队成员的能力：

带领团队需要对团队成员进行培养和发展相应的能力，需要能够识别不同人员的自身特点，有针对性的培养相应的技能。

沟通协调能力：

要带领好团队，较强的沟通协调能力必定能事半功倍，让团队工作更顺畅。

2. 测试经理

测试经理一般是管理一个测试部门，下面可能有多个测试小组。测试经理除了需要关注技术外，还需要关注部门的发展、绩效等。需要的相应技能有：

技术洞察力：

测试经理需要对技术趋势和先进测试工具有较多了解，需要能够帮助团队确定测试技术和测试工具的研究和使用，以提高团队的工作效能。

风险识别能力：

测试经理需要能够很好的理解业务需求、识别项目风险，

负责制定测试策略和具体的实施方案，并能进行总结、报告，及时反馈项目质量状态。

培养团队成员的能力：

团队成员的能力培养非常重要，测试经理跟测试组长一样需要这个技能。

沟通协调能力：

测试经理不仅需要协调测试部门内部的各种情况，还需要横向跟公司其他部门进行沟通协调，沟通协调能力更加重要。

3. 项目测试负责人

项目测试负责人主要负责一个项目的质量保障工作，需要有跟测试经理相似的技能：技术洞察力、风险识别能力和沟通协调能力。

4. 测试总监

测试总监是测试经理的延伸，属于质量部门的最高负责人，需要负责公司所有项目的质量活动，所要求的的技能跟测试经理类似。

03 易转型方向

除了测试直接相关的管理岗位外，根据测试人员的职业特点，以下两个岗位是比较适合转型的方向：

1. 项目经理

2. 产品经理

1. 项目经理

测试人员，尤其是敏捷团队的测试人员，涉及到项目质量相关的方方面面，自然有着能纵观大局的机会，成功转型项目经理的例子非常常见。相应的技能要求有：

团队管理能力：

管理团队，包括人员风险识别、协调沟通等方面，需要掌握一定的人际关系相关的软技能。

客户关系管理能力：

项目经理除了要搞定团队，还有最为关键的是要处理好跟客户的关系，客户关系管理技能特别重要。

决策能力：

决策能力是一种综合的判断能力，即面对几个方案或错综复杂的情况，能够做出正确的判断并采取行动。

2. 产品经理

软件测试人员都需要能够很好的理解业务需求，一般都有很强的业务能力，转型当产品经理是一个不错的方向。产品经理相应的技能要求有：

用户需求挖掘能力：

产品经理需要有包括挖掘潜在用户需求、确定需求优先级、构建用户画像的能力。

多维度思考能力：

产品经理需要能够从基本维度、外在维度、核心维度和商业价值维度思考的能力。

抽象能力：

产品经理不仅要能从事物本身进行抽象，还需要能够考虑不同层次的抽象；抽象完后，还需要把抽象的对象回归到展示层面，需要有抽象回归具象的能力。

04 三个转变

测试人员要培养前面介绍的技能，首先需要实现下面三个转变：

1. 对测试的认知

测试活动不仅是验证系统功能，可以更加的多样化。比如，测试左移就包括对需要的澄清和验证，测试右移则包括生产环境的监控和信息收集等。

测试人员不是质量的把关者，好的质量意味着要交付更多的价值，而不是没有缺陷那么简单，测试人员不再是发现缺陷越多越有成就，而是要想着如何跟不同角色高

效合作，使得交付的产品能够优化业务价值。

2. 对技术的关注

由于测试活动的多样性，不能只关心测试相关技术，要把视野扩展到软件开发过程中各个环节接触到的领域知识和不同类型的技术，不同业务类型、技术架构和基础设施等都会对测试有不同的影响和要求。

3. 测试不可以独立存在

测试不能再以独立部门自居，需要跟不同的角色更多的沟通和合作。比如，需求分析阶段需要跟需求人员有密切的沟通，实现自动化测试过程中可以跟开发人员结对或其他方式的深度合作，生产环境下的测试需要跟 Ops 人员紧密合作等。

同时，测试人员对于系统所采用的技术架构、技术方案的设计思路都需要有所了解，从而更好的理解开发的工作、理解架构演进对于测试的影响，更好的开展测试工作。

跟着团队一起转型

敏捷软件开发模式已经被越来越多的企业所采用。在新的开发模式下，传统的质量保障方式需要发生较大的变化，测试团队和每一位测试人员都将面临着转型带来的挑战。如何实现测试的顺利转型，已经成为备受关注的主题，本文将跟大家一起来探讨。

01 转型给测试带来的挑战

转型让测试人员打散到研发团队，通常使得测试人员茫然无措，非常地不适应。与众多转型企业中的测试团队沟通后发现，测试转型面临的挑战主要有以下几个方面:

归属感

隶属于传统测试部门的测试团队习惯了以往的部门级管理方式，往往在加入各个不同的产品研发团队之后，找不到归属感。

职责定位

融入研发团队的测试人员面临新的工作方式，不清楚自己的职责和在团队中的定位，不清楚哪些事情是需要做的，发挥不了应有的价值，从而导致话语权低，在团队没有影响力。

团队协作

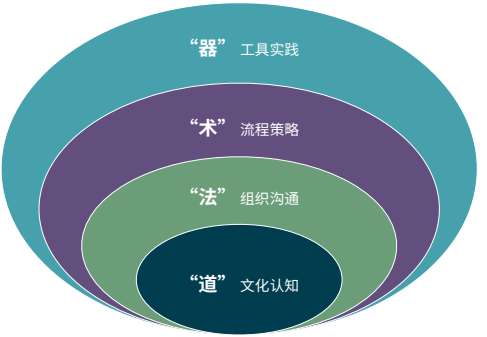
由于不清楚自己的职责定位，也不知道该怎么跟不同的角色进行协作，这种新的团队组织方式，让测试人员担忧。

能力建设

新的组织方式和开发模式下，测试人员的能力建设也是面临很大挑战的问题，组织和个人都不清楚测试人员的能力要求，也不清楚测试人员的能力提升路径，而对测试人员的技能要求比以往都要高都要全面。

02 测试转型的道法术器

面对前面提到的几个挑战，测试转型想要顺畅进行，想要从以下几个层面去实现转变，我把它们称为测试转型的道法术器，如下图所示：



先通过几个小例子来直观感受一下：

- **测试转型之器**——工具实践层面： 如果有两个组织分别采用全手动回归测试和全自动化回归测试，毫无疑问，是那个自动化回归测试的组织实践更好。

- **测试转型之术**——流程策略层面： 如果在那个全部自动化回归测试的组织有两个部门，一个是产品开发完成后有独立的测试阶段，另一个是测试左移到需求分析阶段，并且全流程介入，我们会说第二个做到了测试左移的部门的流程策略更好。

- **测试转型之法**——组织沟通层面： 如果在那个有测试左移的大部门分别有两条研发产品线，其中一条

的测试是一个独立的团队，而另一条的测试人员跟开发组建为开发团队，测试开发进行了融合，那么，进行测试开发融合的产品线的组织沟通方式更值得提倡。

- **测试转型之道**——文化认知层面： 虽然进行了测试开发融合，但是在文化认知层面还有一部分团队的测试人员是停留在以发现 bug 为主要目标，而另一部分团队的测试人员则以交付更大价值为目标，那么那些关注交付价值的团队一定会带来更大的价值交付。

下面更详细的介绍道法术器每个层面都有哪些内容。

1. 测试转型之道：文化与认知

文化与认知的转变是测试转型最关键的层面，只有正确的文化认知才能指导后面三个层面的正确执行，才能真正地实现转型；当然，文化与认知的转变也是最难的，根深蒂固的认知要发生转变必须以终身成长的心态来面对才有可能实现。

文化与认知主要体现在以下三个方面：

价值文化

以交付价值为目标，测试要以业务价值驱动，关注整体的质量，不再是以发现 bug 为目标，bug 要做到预防为主。请参考本书第一篇文章《用业务价值驱动测试》。

质量文化

质量不仅仅是测试人员的事情，测试不再是质量把关者，团队人人都要为质量负责，做到全面地重视质量。请参考本书文章《团队为质量负责》。

学习文化

以持续提升为目的，鼓励团队成员创新，允许试错，允许失败，只有经历过失败，才能获得更大的成功；营造学习型的文化氛围，培养团队成员的成长型心态，实现持续精进，做到终身成长。

2. 测试转型之法：组织与沟通

组织与沟通是相辅相成的，好的组织方式才能带来顺畅的沟通。组织与沟通同样体现在三个方面：

组织架构

推翻部门墙，将测试人员打散到开发团队，这种表面上的组织架构的变化，实际上是为了更充分的沟通，我们需要促进沟通的充分和顺畅进行，发挥团队整体的最大价值。

协作赋能

在新的组织架构下，测试人员需要做到全周期的参与，全方位的与不同角色进行沟通和协作，以实现对不同角色进行质量赋能。详情请参考我的网站文章《敏捷团队的质量保障赋能》。

构建社区

打散了的测试人员分属于不同的研发团队，构建一个横

向的测试社区非常关键。这个社区可以制定组织级质量保障策略，构建测试胜任力模型；一起探讨解决质量保障过程中的难题，实现质量知识的共享，让测试人员抱团共同成长。

3. 测试转型之术：流程与策略

转型后的流程与策略跟传统模式相比会有很大的区别。这里还是强调三个方面：

全流程介入

测试全流程介入的开发模式，最关键的是要做到测试左移、持续测试和测试右移。分别请参考 Thoughtworks 洞见文章《敏捷测试的核心》和本书文章《QA 在生产环境下做什么？》

轻量级策略

测试策略文档要摒弃传统的冗长文档，而是采用可视化

的一页纸测试策略，做到轻文档，重视文档背后的充分沟通和协作。详情请参考 Thoughtworks 洞见文章《一页纸测试策略》。

演进式策略

测试策略受影响的因素非常之多，不同项目团队、不同人员成熟度、不同的项目阶段，策略都会有差异，需要拥抱变化，实现策略的目标驱动和演进式发展。

4. 测试转型之器：实践与工具

测试的成功执行离不开好的实践和工具，这个层面也是大家比较容易接受和较多的采纳的。同样从三个方面来看：

实践活动

实践活动，主要是强调这么几条原则：测试要以业务价值驱动，基于业务优先级来开展测试，让测试更有价值；

关注整体质量，要从整体视角去看质量，而不是把注意力放在某些细节功能模块上；缺陷要做到预防为主，不再是关注后期发现缺陷的数量。

工具方法

工具主要有自动化相关的工具，包括测试自动化和流程的自动化；持续集成流水线以及流水线上的标准化工具也是助力质量提升的赋能利器，这块请参考我的网站文章《从技术趋势看质量赋能》；另外一个就是 DevOps 系列工具的采纳，让整个软件交付过程更加顺畅，当然质量也会更有保障。

精益测试

但是，不同的实践活动和工具的采用要特别的注意，实践活动很容易流于形式化，工具很容易被神化。我们只有认清实践背后的真正价值，才能让实践带来应有的价值，只有真正理解工具所能解决的问题，才能物超所值；而不是盲目模仿别人的实践，或者盲目的采购工具，最后得不偿失。

另一方面，要特别关注测试的有效性，做到实时、适量

和精准，也就是精益测试。请参考 Thoughtworks 洞见文章《精益测试》。

03 测试人员的能力提升

由于组织架构的调整，测试人员所属团队的负责人一般都不是专业测试人员，对测试的技能要求并不是很清楚，很难对测试人员的能力提升起到关键作用；另一方面，原来的测试负责人，由于不直接跟测试人员在一起工作，对测试人员的能力提升也是爱莫能助，感觉使不上劲。

在这样的情况下，该如何面对挑战呢？我们分别从组织角度和个人成长角度来探讨。

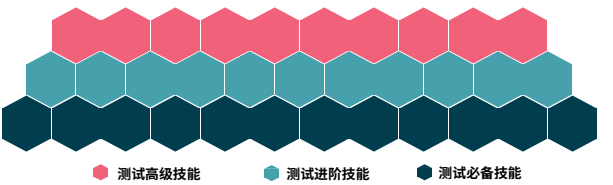
1. 组织角度

从组织的角度，测试负责人需要承担起测试人员能力建设的职责，可以从以下三个方面去开展能力建设工作：

构建测试胜任力模型

对测试技能需要进行整理和分类，构建不同级别的测试胜任力模型，以此作为测试人员的成长路径指南。胜任力需要将行业趋势和组织内部业务、技术需求相结合，

形成适用于不同层级技能的测试人员，比如有测试必备技能、进阶技能和高级技能等。



测试人员梯队建设

基于胜任力模型对测试人员进行针对性的培养，构建短期快速胜任和长期持续胜任的能力，建设健康的测试人员梯队，做到可持续的培养和发展。

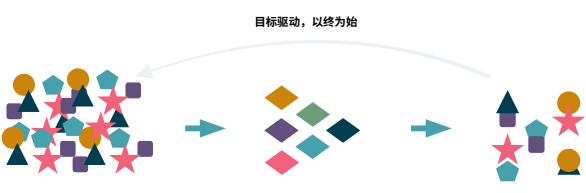
测试人员的能力培养同样需要遵循知识 -> 经验 -> 能力的提升路径，并且与测试人员的自身优势结合。创造环境，鼓励测试人员将知识应用到实际项目中，形成经验积累，以最终发展为解决问题的能力。

构建测试社区

前面测试转型之法部分已经提到构建社区，对测试人员能力建设非常关键。通过这个社区，可以实现前面提到的测试胜任力能力模型构建和测试人员梯队建设，帮助测试人员构造能力提升路径。

2. 个人成长角度

从测试人员的个人成长角度来看能力提升，我的网站有文章《BQConf 演讲：软件测试人员该何去何从？》详细讲过这方面的内容，这里着重强调三点：



当我们有明确的目标需要去解决某个问题的时候，学习的效率会加倍的提升，效果会更好。因此，找到提升的目标，以目标驱动去学习，从海量的知识里找到自己当

下需要的部分，并且进行提炼加工，应用于实际工作中，这是最佳学习路径。

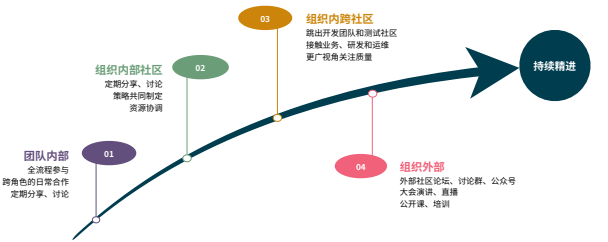
目标来自哪里呢?

目标可以来自于三个方面：

- 项目需求：项目可能正在转向微服务技术架构，需要针对性地开展微服务测试；项目可能需要加强自动化测试，需要自动化测试技能。
- 技术趋势：每年发布两期的《Thoughtworks 技术雷达》就是不错的技术趋势指南。
- 领导者的建议: 作为领导者更清楚组织的发展需求，可以跟自己的领导去寻求建议，确定学习发展的目标。

增强互动交流

增强互动交流，让知识翻倍，做到持续精进，是加速学习成长的关键。要做到跨角色、跨社区 / 部门、跨组织的沟通，从更广的视角去关注质量，提升自己的能力。



成为真正的“QA”

关于敏捷 QA 的含义，有三个层次：

1）QA = Quality Assurance 质量保障

第一个层次 QA 的要求是做好质量保障工作，确保我们交付给客户的软件产品是正常工作的。这个层次的 QA 跟传统测试人员的工作比较接近，主要在做质量保障执行层面的工作，包括理解需求，根据需求设计测试用例，并执行测试。

2）QA = Quality Analysis 质量分析

第二个层次的 QA 通过测试、数据收集的方式，分析系统的质量，识别风险，并反馈给团队，和团队所有成员

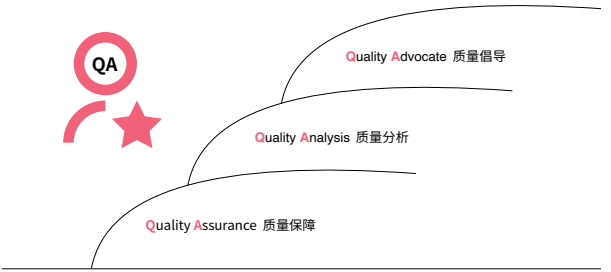
一起确保交付的质量是合格的。这个层次的 QA 除了设计并执行测试，还需要做很多的分析和协调工作，这也是敏捷团队 QA 的基本要求。

3）QA = Quality Advocate 质量倡导

第三个层次的 QA 不再局限于只关注质量，通过培养对产品和流程持续改进的思维模式、了解产品的整体质量视图和持续关注产品质量的意识，引导整个团队构建正确的产品，并且正确地构建产品。这个层次要求 QA 承担质量倡导者职责，为团队进行质量赋能。

测试人员转型的最高目标就是成为这第三个层次的 QA，通过分析软件风险并制定相应实践，确保软件在其整个生命周期中持续工作并创造价值，为业务和团队提供信心，从而为客户提供价值。

QA 的所有实践和活动都需要以价值为核心来驱动，根据不同团队的具体情况可以适当调整，且在不同项目阶段应该也是演进式的。QA 跟各个角色的沟通和协作，也是随着团队成员的了解程度、配合默契度不同而有变化，并不是要求一成不变的。



04 写在最后

转型会很难、很痛，但转型也是大势所趋，必须勇敢面对。

只有能够拥抱变化，以成长型心态面对，实现终身成长，转型才可能成功，才可能不会被淘汰。

编 辑：张凯峰
设 计：陈思容
宣 发：徐艳娇 何颖

Thoughtworks 洞见

《质量三人行》播客

林冰玉的网站