QA 与 Ops 通力和打造反脆弱的软件系统

——林冰玉　ThoughtWorks　高级软件质量分析师

大家好，非常高兴来到上海见到大家。我是林冰玉，08 年的时候加入 ThoughtWorks ，到今年正好是 10 年，服务了 16 个客户的不同项目。我主要是在海外作为软件质量分析师的身份工作，同时给国内的一些客户做过敏捷测试相关的培训和咨询。

我也是 ThoughtWorks QA Community 的 leader。ThoughtWorks 没有测试部门，所有的 QA 人员都是分布在不同的敏捷开发团队的。那么我们 QA 之间怎么样分享知识，共同进步呢？于是我们成立了这么一个 Community 。另外，我还是 BQConf 中国软件质量大会的负责人，这是 ThoughtWorks 从2010 年开始主办的，大会目前在四个城市——西安、成都、武汉、北京都有举办，北京地区刚刚举办了第 34 届，大家感兴趣的话可以在官网关注这个活动。

我今天的话题和日志处理有关系，首先要讲的是软件系统的脆弱性和应对方案，然后是日志处理的常见误区及改进，最后的重点是在项目上的日志处理的一系列的故事上。

软件系统的脆弱性

随着技术架构不断的演进，如现在流行的微服务，导致了系统生态环境日益复杂；业务的扩展，使数据日趋增加，数据量级越来越大，也越来越复杂；同时我们的基础设施，像云、容器也导致了很多的不确定性。这些生态环境导致了软件系统本身的脆弱性，软件系统不可能做到 100% 的健康，同时软件可靠性和稳定性的维护成本也非常的高。

这些不确定性给软件质量保证带来了很大的挑战。首先，不可预测性是没有办法回归，通过脚本等进行回归测试，是没有办法测出不确定性的。另外，测试需要探索，光是设定脚本去测试远远不够，测试环境有限，测试需要后延到生产环境中。

这种环境下，怎么做好软件系统的质量保证？在这里我想先卖一个关子，先问大家一个问题——你们认为脆弱的反面是什么？

没错，脆弱的反面就是坚强、坚韧或是健壮。杯子扔到了瓷砖的地面上会碎，我们认为这个杯子是脆弱的；一纸团扔到了地上没有变化，我们认为这个纸团是坚强的；乒乓球扔出去后会弹出来，乒乓球不仅是坚强的，还能在不确定性的因素中受力弹回来，我们认为这是反脆弱的。

反脆弱是一本名叫《反脆弱》的书里提到的概念。这里借用这个概念，和大家聊下如何做到软件系统的反脆弱性！

提及软件系统的反脆弱性，这里要介绍一个技术—— **QA in production** ，即生产环境下的 QA 。有人知道 ThoughtWorks 的技术雷达吗？这是我们技术战略委员会经过了长期调研得出的、评估技术趋势的雷达，它分成技术、平台、语言、工具四个象限。每一个象限有四个不同的维度，分别是采纳、试验、评估、暂缓，这几个维度用来评估技术趋势可采用性。技术雷达 2015 年提出的生产环境下的 QA ，可以帮助我们打造软件系统的反脆弱性。

生产环境下的QA，我认为主要是三个部分。

第一是测试，指的是直接在生产环境下测试。生产环境有着复杂性、不可破坏性，我们并没有办法像测试环境一样在生产环境下进行测试。但有些功能还是可以测试的，比如一些只读的操作，对其进行测试不会影响其他功能使用。或者有些服务可以隔离出来进行测试，还有可以采取蓝绿部署，做出和生产环境相同的环境，去测试生产环境下可能暴露的问题和缺陷！另外，我们在微服务下提出的混沌测试、故障注入测试，也是利用生产环境做出的。

第二个部分是监控预警，是利用生产环境下的日志，以及网站分析工具做一些监控和预警，拿到生产环境下的信息，帮助我们做好质量保证的工作。

第三个部分是用户反馈。大家知道，生产环境是真实用户使用的环境，所有的用户提供的反馈都是非常的有价值的，所以搜集用户反馈及抱怨，以及向 VIP 用户定向征求的意见，都可以帮助我们做好生产环境下的 QA 。

三个部分中，我认为**最最重要的是监控预警的部分**，这和我们今天的日志处理专场也是契合的。因为监控预警中，日志处理是非常关键和重要的。

日志处理的常见误区与改进

接下来我们就来说日志处理。日志处理有哪些常见的误区？说到日志处理，大家都会想到什么？有这么几个点：首先，是运维人员在做日志的监控、分析和处理；还有，日志的记录和分析会用什么工具、什么技术；另外，生产环境下应用的稳定性和可用性的监控，也是我们的关注点。

先说 Ops 处理日志，会有什么局限。

第一，运维人员对业务可能不是很了解，运维人员可能比较容易遗漏高业务风险的日志；还有，Ops 比较关注的可能是技术，他们对日志处理的分析和共享，其实是有所缺陷的；其次，运维人员肯定不会像 QA 那么关注系统质量，他们对系统整体质量的关注可能不是那么全面。开发团队有业务开发，他们有对系统质量的把控；Ops 人员单独处理日志的话，会形成信息孤岛。那么如果 QA 参与，会怎么样？

QA in production，会有几个优势。

第一，**QA 在质量反馈方面有优势**。QA 会特别的关注系统整体质量，会去掌握相关信息并反馈给团队。QA 时刻关注质量的思维模式，也会使得他善于发现问题。

第二，**QA 在分析优化方面有优势**。QA 对质量问题的容忍度低，他们持续改进的意愿强烈。

第三，**QA 业务敏感度高**。他们业务风险意识强烈，是最熟悉业务的，会从业务的角

度去思考分析。

围绕着优化业务，做好质量保证的工作，以上便是我所认为的 QA in production 的三个优势。

项目上有关日志处理的故事

下面分享一下项目上的故事。我们的项目是一个离岸交付的项目，客户在海外，开发团队在北京。客户那里有两个 Ops 的人员，**其他的 Ops** 是开发团队的人员来提供，项目采用敏捷开发的模式，团队人数维持在 50 到 80 人之间，每四到五周为一个开发周期。我们有三个系统，分别是企业内部使用的系统、面对客户的 Manage 系统，还有面对大量用户的用户系统，三个系统的用户分布在全球各地，也就是说这是一个全球的系统。这个项目是 09 年开始的，到现在已经有 9 年的时间了，现在还在继续。在这个过程中，我们的架构和业务演进了多次。

这期间的某一年，随着微服务的不断扩展，我们生产环境下的缺陷和错误日志爆增，而且这种缺陷很难在预生产环境测试出来。大量的错误日志，给我们团队以及客户带来了恐慌。这种迫在眉睫的时候，我们想到了日志处理，刚好同时期 QA in production 技术提出，于是我们采取了一系列的实践。

我们的日志处理是怎么做的？我会分三个阶段进行介绍。

初期被动分析

初期是非常被动的分析，大量的日志数据怎么下手都是一个问题。开始我们使用 Splunk 去查询每天重复出现的日志，并按出现的次数做排序，Ops 人员处理出现次数最多的这些日志，当天能处理多少算多少，但因为每种日志的类型都需要去分析、调查、修复等等，有时候排名前十的日志类型都处理不完。下图中的 PUNCT 是 日志分析中的会采用的一个技术，PUNCT 指的是采用日志信息的第一行，把其中所有的字母和数字去掉以后，剩下标点符号，然后是空格，这里空格是用下划线代替的，后面是 “:::|||| ” 。通过这种查询方式，字段相同的日志是会统一归为一类的，所以看不同日志类型的数量就好了。这个过程主要是我们的 Ops 人员来做处理，还没有 QA 介入。

PUNCT Punctuation，标点 第一行

2018/10/23 05:51:43.154||Error||Domain\XxxxSvc-QA||No row with the given identifier exists[Xxxx.Model.Models.ABCDStatement#74f5e429-5b87b-454f-982e-0f87fe4e1de9]|| NHibernate.ObjectNotFoundException: No row with the given identifier exists[Xxxx.Model.Models.ABCDStatement#74f5e429-5b87b-454f-982e-0f87fe4e1de9]

at NHibernate.Impl.SessionFactoryImpl.DefaultEntityNotFoundDelegate.HandleEntityNotFound(String entityName, Object id)
at NHibernate.Proxy.AbstractLazyInitializer.CheckTargetState()
at NHibernate.Proxy.DefaultLazyInitializer.Intercept(InvocationInfo info)

//_::.||||\\-||_____[...#----]||.:_____[...#----

instance, Object[] methodParameters)
at System.Web.Http.Controllers.ReflectedHttpActionDescriptor.ExecuteAsync(HttpControllerContext controllerContext, IDictionary`2 arguments, CancellationToken cancellationToken)

【z-1图】

我们来看这个阶段的处理结果是怎样的。

从分析过程来看：因为精力有限，我们没有办法覆盖全部的新增日志，分析是按从多到少的顺序进行的，我们只能处理前面的一部分；而且，从多到少的排列并不一定是非常合理的，关键的业务日志可能只有一条，但它影响很大，这种关键错误日志很有可能会被遗漏；另外，Ops 人员每天焦头烂额的处理日志，根本没有时间去做总结和分析；所有的这些日志信息全部是掌握在处理日志的 Ops 人员的手里，也没有信息的分享和传承。

从日志记录来看：在日志问题暴增后，我们发现了很多日志处理上的问题。我们的日志级别定义不清晰，一些低级别的日志也被错误地记录成了错误日志，这样每天新增几千条的错误日志，可能根本就不是错误；还有记录的信息不够用，需要用到的日志信息可能没有记上，现有的日志信息很难帮我们去解决问题；其次是日志记录的格式不一致，我们有不同的系统，也有不同的人员在开发，虽然是一个大团队，但是还没有统一日志记录的格式；甚至存储的路径也不一样，不同的系统日志会记录在不同的地方。

这个期间，QA 想去帮忙，但是也感觉到力不从心。

QA 的痛点主要在几个方面：首先 QA 没有权限去接触生产环境；QA 在北京，也没有办法去很好的了解基础设施，平常也不会接触这一块；另外，QA 理解这种日志记录也是有难度的，这时候 QA 也帮不上什么忙。

出了这些问题以后，团队引起了重视，我们开始主动出击内建日志。

主动出击内建日志

内建日志，第一是优化日志记录的问题。

结构化日志也是技术雷达提出来的。为做到日志记录的规范化、结构化，针对初期的级别定义不清晰、日志格式不一致等问题，我们把日志规范为统一的 JSON 存储格式。这种清晰的记录，QA 一看就知道相关信息了。这个规范化的过程，是我们的 Ops 人员和 QA 等人共同参与完成的。

那要怎么从流程化上去做日志内建呢？

首先是分析的阶段，需要在我们的故事卡里面大概的列一下哪些地方是需要记录日志的，这样开发的过程中就会记录这些日志。发现一些遗漏的，会再一次的加到故事的分析中。开发完毕、故事验收时 QA 加入，QA 从流程上把控、验收日志记录是否符合期待，没有被记录的日志会反馈回来，再到开发这里记录完整。与此同时，做好日志内建以后，我们同时也会做好监控预警，这时候 Ops 和 QA 等人员一起定义出来哪些业务优先级比较高，哪些是高风险的特性，哪些 API 是需要特别关注和记录的，然后通过邮件等方式发给相关人员。我们有很多监控预警的实例，每一条点进去都会显示日志的详细信息，如出了什么故障等等。

我们来总结一下这时候 QA 和 Ops 合作的主要价值体现。

第一，QA 是从流程上参与把控，做好日志内建。

第二，QA 与 Ops 一起去识别风险较高的关键特性，在识别的过程中，也能帮助 Ops 人员更好地理解业务。

第三，Ops 人员设置关键特性的监控。

第四，Ops 帮助 QA 在测试环境中也设置了关键特性的监控。

第五，QA 会负责测试环境的日志监控。QA 会定期的查看测试环境中的日志信息，以求尽早的发现问题。这时生产环境的监控分析还是 Ops 人员负责，这时不只是客户这边的 Ops 人员，会有一到二个 QA 加入 Ops 团队，一起去做分析和处理。

这期间的结果是怎么样的呢？由于我们加入了业务优先级的概念，我们就可以减少遗漏高优先级、高风险的日志的可能性；另外对于关键特性的把控也好了很多；我们的日志

处理也高效了；我们还顺利的度过了业务忙季的难关。

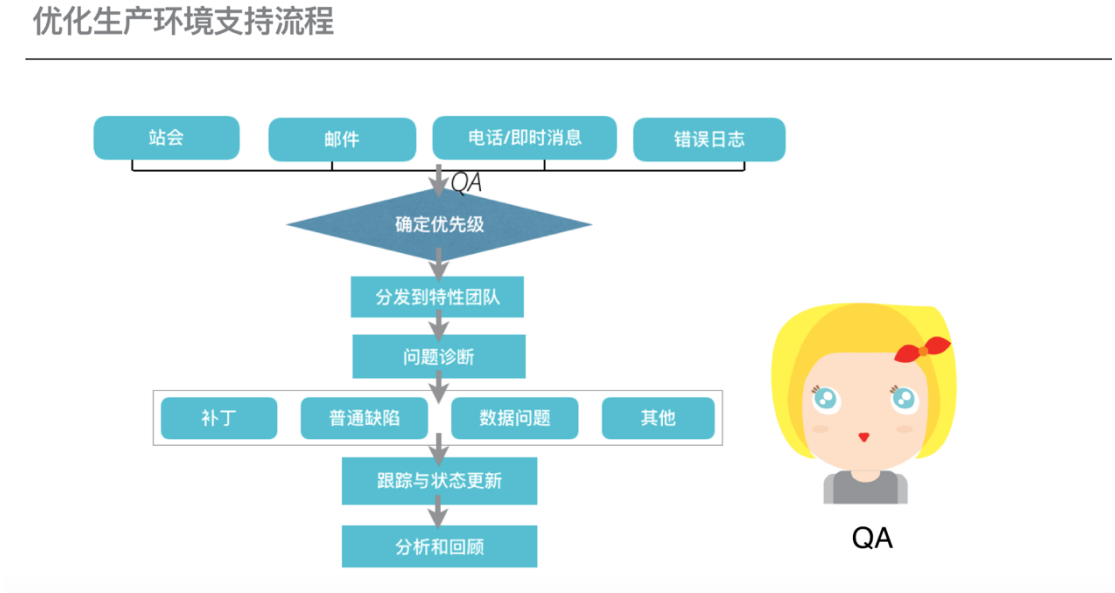
但这期间又发现了一些新的问题。

首先我们设置的 Alert 收集的信息量过多，信息没有精简，一些不是很需要的信息也被查询出来了；然后，我们设置的是一些定时的 Alert ，这在忙季很有用，但随着忙季慢慢的过去，Alert 定时发送的一些信息，在淡季可能是没有问题的；另外，随着忙季的过去，错误日志信息越来越少，团队对此的关注度不断下降，等真的错误出现时，也没有人能够及时的发现；这个时候，我们不能早于用户的反馈去发现问题。

这些新的问题出来的时候，我们该怎么办？

QA主导进一步优化

在合作的过程中需要 QA 主导，去做进一步的优化。下图是我们生产环境下支持的流程，这些是 QA 团队做的。



【Z-13图】

QA 作为主要的接口人，会去接收生产环境下的问题。之前，我们通过早上和客户的站会，或者是邮件、电话、即时消息等拿到的生产环境下的缺陷，在后续的流程中，我们把错误日志也增加进来了，也会关注错误日志所发现的一些问题。不管是哪一个方面来的问题，我们都会有统一的流程，走后续的修复、状态更新、跟踪，最后会有分析和回顾。

QA 会主导流程上的把控。前一阶段的日志处理会有很多的问题，像 Alert 查询信息

过多等等，QA 承担协调者和分析者之后，会去组织痛点的收集，然后去推动优化的发生。QA 天生就是比较关注质量的，他们会促使团队做出优化。如会把 Alert 去精简，忽略无用噪音，使真正有问题的时候才会触发 Alert。

同时，QA 加入到特性团队中。QA 会和开发人员一起做日志的监控和 Alert 的分析处理，有些时候 QA 会和 Ops 一起结队，一起分析日志，互相取长补短，QA 进一步的帮助我们的 Ops 人员熟悉业务，Ops 进一步帮 QA 理解并做好测试环境的日志监控。

然后，QA 会定期查看一些关键特性的错误日志。不管设置了多么完美的监控，还是有可能会漏到一些没有及时被发现的日志信息，所以每一次发布后，QA会花几天的时间查看生产环境下有没有新产生的错误日志，或者是一些高优先级的日志。我们有一个邮件的功能，是给客户的大老板用的，这个功能每一个发布周期可能只发一到二次的邮件。有一次发邮件失败了，QA 在看日志的过程中知道这个功能是特别优先的，因为是发给大老板的，一旦错误可了不得！在这个期间也体现出了 QA 业务敏感度的价值，同等情况下 Ops 人员可能不会敏感的察觉到这条日志信息的重要性。

第三个阶段以后，我们日志信息的处理更高效了。有了业务的优先级，有了 QA 在流程上的把控和推动，我们生产环境下的缺陷发现的更及时了，把日志处理融入到整个生产环境的支持中，我们后续缺陷的遗漏也没有了，客户也给予了我们好评，团队工作起来也更开心了！

三个阶段说完了，我们简单的回顾一下项目上一系列的故事。前期被动的分析阶段，是 Ops 人员独立完成的，随着问题的暴露、流程的优化，QA 参与到日志分析处理中，从流程上帮助做好日志内建、监控和预警，接下来 QA 主导，和 Ops 进一步的合作，做好日志的监控、分析和处理。这个过程中，QA 主要是帮助 Ops 更好地理解业务、把握系统质量，同时我们的 Ops 人员帮助 QA 更好地理解日志处理，以及利用日志信息优化测试过程，提升业务价值。整个的日志处理和开发过程，形成了良性的循环。

今天一共说了三个部分，首先是这种软件系统的脆弱性，我们提到了生产环境下的 QA ，利用生产环境下的信息构建反脆弱的系统，最关键的就是日志处理和监控预警。其次是日志处理有哪些误区，以及 QA 参与会有哪些优势，最后分享了我们的九年项目，也就是我们在日志处理上不断演进、优化的过程。

今天我要讲的就这些了，谢谢大家！

Q&A：

提问：您说到日志的内建，有一个流程是从分析到开发，再到验收。验收肯定是有标准的，比如说一些日志到底是不是该记，这个标准是很难衡量的，请问有没有这种规范？

林冰玉：并没有一个特别通用的规范，主要还是从业务的角度出发。在分析阶段，QA 会将一些和技术人员一起讨论出来的、特别关键的信息规范化，这些在我们的故事卡上都会提前写好。QA 验收的时候看这些关键的信息，根据验收的条件去验收就好了。

提问：你们会把一些日常的场景都写清楚，验收的时候按这个去做？

林冰玉：对。

提问：如果你们交付的项目是一些财报之类的敏感数据，数据的异动需要做一些分析，你们会不会做一些模型去监控数据的异常？比如说一些敏感的数据，会去监控是谁做的修改之类。

林冰玉：我们会根据业务上的一些要求进行日志记录。这一块做的不是很多，但是我们的审计日志是有记录的这些日志信息的。

提问：你们会借助AI做一些分析吗？

林冰玉：我们目前还没有这么的高大上！

提问：据我所知，质量管理应该是分 QA 和 QC 的，贵组织对质量管理是不是有一个明确的分工？如果有的话，按我的理解，很多工作应该都是 QC 去做，而不是 QA 。那 QA 区别于 QC ，更应该关注哪一些业务？

林冰玉：不好意思，我们的团队是只有 QA ，我们 QA 的定义是质量分析师，是从业务分析阶段开始，一直到最终线上环境的实践，都要全程参与的。

提问：那我可以理解你们的组织形式是项目性的形式吗？

林冰玉：对，我们 ThoughtWorks 的模式是敏捷团队，我们的所有的 QA 都是分散在不同的敏捷开发团队，和开发人员是一起的，所以我们并没有独立的测试部门。