白皮书 | 2016 年 3 月

DevOps 与测试人员

测试专家兼顾问 Paul Gerrard 在一系列文章中探讨各种测试主题。他在本文中从测试人员和测试的视角讨论 DevOps 的采用。DevOps 是各组织用于频繁而优质地交付软件的总体方法的一部分。成功实施 DevOps 的最明显成果是缩短软件变更从想法过渡到生产的时间。

Paul Gerrard
Gerrard Consulting

DevOps 对测试人员意味着什么？

背景

在本文中，我想从测试人员和测试的视角讨论 DevOps 的采用。DevOps 运动（由于缺少更好的标签）正在迅速进行。与该行业已采取的许多其他行动一样，采用速度的增加比该运动的定义本身更快。DevOps 仍未得到明确定义，而文化的细微差别、新技术突如其来的能力和（大多为成功的）案例研究的范围意味着手头上的问题仍然广受争论。[1]

根据您谈话的对象，DevOps 可以是问题的解决方案或者本身就是目标。在某些企业中，目标是“变为数字化”，DevOps 是频繁而优质地交付的总体方法的一部分。这是我在本文中假定的上下文。但在 DevOps 相关技术和服务的营销中，此目标可能会被掩盖。成功必需的文化变更（或者更具体地说，行为变更）的挑战常常被低估。

我将要做的另一个假设是参与 DevOps 并受其影响的测试人员对整个想法并不熟悉。我会将本文塑造为向这些测试人员介绍 DevOps 以及讨论其对测试实践的影响。如果您是经验丰富的 DevOps 从业者，我希望您仍能发现本文的有用之处。如果您不是测试人员，您将至少了解测试人员的视角。

对门外汉而言：DevOps 是什么？

单纯地说，DevOps 是给开发和系统运营团队更密切合作的概念贴上的标签。在所谓的交付管道中，从源代码提交到生产中的运营，开发人员调整某些运营活动并将其自动化。运营对开发人员的活动具有更多的可见性并会对这些活动产生一些影响。此过程的动机主要是加快软件的开发和实施。将 Ops 和 Dev 更紧密地联合起来 — 高效地组成一个敏捷团队 — 会实现可能被称为“敏捷运营”的东西。

成功实施 DevOps 的最明显成果是缩短软件变更从想法过渡到生产运营的时间。开发人员表示软件变更“已完成”时，向生产用途的过渡会在普遍自动化的协助下进行。自动化工具和流程用于系统配置、构建流程、测试、部署到测试、准备和生产环境、部署后监控、评估和运营。

所以，DevOps 只和工具有关？

在一个层面上，DevOps 的目标是消除交付管道到自动化中的瓶颈。但分阶段流程的自动化仍然需要监管。大多数自动化流程并非真正自主 — 在没有人为干预的情况下，它们无法完成维护或处理异常的任务。不考虑人为因素，完全自动化的 DevOps 流程就毫无意义。尽管工具承担了大量的繁重工作，但运行流程并使其运转或失效的是人。

那么，DevOps 仅仅是 Dev 和 Ops 人员在工具的协助下更紧密地合作吗？

不，也不是这样。自动化流程之间的切换往往会涉及其他流程 — 通常为一种或另一种流程的测试。自动化测试需要由开发人员和测试人员来创建。这些测试的结果集中于为其他流程或人员提供足够的信息，以便在管道中的各阶段之间过渡。进行测试的测试人员和开发人员提供 DevOps 流程成功而可靠地交付的保证。

“真让人头疼，DevOps 到底是什么？”我只能说，它是一种不断发展的、突如其来的准则。这里有篇精彩的帖子提出这个问题并进行详尽的讨论。[1] 这一争论发生在撰写本文仅仅数周之前。因此，您可以发现 DevOps 的定义仍然悬而未决。也许永远都无法解决。

这对测试人员意味着什么？这意味着仍然没有“唯一正确的方式”，并且您在不断发展的 DevOps 体制（每个体制都在不断发展）中的角色尚未固定。您可以做出两种主要贡献：

1. 您需要注意那些造成痛苦的事情，并努力使其不那么令人痛苦。
2. 您需要确定将为 DevOps 流程增加价值的机会和干预。

如果有一句最能描述 DevOps 的推动因素的口头禅，那就是“如果令人痛苦，就多做”。这可能有点老生常谈，但我会将其用作实施和改进 DevOps 测试实践的上下文。

如果令人痛苦，就多（经常）做

我们在进行特定工作时遇到的困难或痛苦会对我们产生负面影响。如果不喜欢做某个任务，我们常常会将其搁置。我们终于开始做这个任务时，它会更加令人痛苦。看牙医、打扫车库、整合软件、测试等等都是如此。我们的经验通常是，我们执行这些任务的频率越低，真正动手的时候，任务就更加令人不快。Martin Fowler 提出频繁甚至连续执行某些任务为什么会减少痛苦的三个原因。[2]

第一个原因是更大、更复杂的任务难以规划、管理和控制 — 将大型任务分解会使其更易于解决、风险更低，而且，如果出了什么问题，更容易解决。第二个原因是许多任务（测试就是最好的例子）会提供反馈。如果在早期经常收到反馈，则意味着可以在浪费更多时间之前迅速而确定地解决问题。第三个原因，如果我们更频繁地从事某项活动，我们会对它更擅长。我们将学会如何有效率地做事。我们还可能发现以某种方式将其自动化的机会。

从测试人员的视角，此口头禅强迫我们在测试流程中更认真地对待自动化的概念。如果有人工干预（通常在 DevOps 流程中的自动化阶段之间），这些将被视为痛点 — 瓶颈、延期的原因以及流程中潜在的可靠性更低和易于出错的方面。手动测试令人痛苦。是的，您可能喜欢探索性测试；您可能担心只有作为人类的您才能发现自动化永远无法找到的这些奇形怪状的缺陷，而作为测试人员的您才是防止灾难发生的唯一值得信任的人。

信任开发人员和自动化来正确进行测试工作可能会让作为测试人员的您感到痛苦。如果令人痛苦，您必须更经常地做。

测试、自动化和信任

围绕检查与测试[3] 以及我们可以对测试人员、对检查和自动化[4,5] 给予的信任等存在着大量的争论。

我不是说我们可以将全部的信任都托付给自动化检查。我们当然需要比那更多的熟练。但是，出于本文的目的，我们至少可以将测试和测试执行活动分割为四个部分。

1. 可由开发人员作为其组件级签入和持续整合流程的一部分实现自动化的检查。
2. 可被自动化（通常由系统测试人员进行）以执行 API 级、链接或端到端事务的检查。
3. 可执行兼容性检查以展示跨浏览器、操作系统、平台的兼容性的测试。
4. 仅可人工执行的测试。

我在本文中仅就进行这些区分的方式提供一些建议 — 当然，每个环境都不相同。与本文更贴切的问题是“测试人员如何‘放开’晚期手动检查？”我在之前已经讨论过消除晚期手动检查。[6] 它需要主动措施和信任。

这些将是您的措施的主要重点：

1. 无论在何处，只要有可能，就应将可在组件级执行的手动检查推进到开发人员。作为测试人员，您可能会在配对或白板会议中提议这些测试。您可能必须自己写下来并将其包括在持续整合体制中。
2. 端到端或用户界面测试可能需要自动化。这些测试需要被减少到最低限度，因为它们常常运行缓慢、脆弱并且经常需要维护。思考它们需要在每次代码签入时运行还是可以被保留而仅用于更大、频率更低的发布。
3. 哪些仅手动测试可在尚未整合到发布候选的组件上运行？是否可在配对会议中与开发人员一起执行手动测试？此测试是否有替代方案？头脑风暴、BDD 风格原型制作是否能够提供帮助？是否可在实物模型或线框模型上执行 UI 检查？
4. 与需要为回归用途而保留并作为自动化候选的检查相比，哪些检查仅需手动运行一次？

我在上文中提到了信任的概念。查看此概念的另一种方式是在完全没有任何晚期手动测试的情况下推测如何对系统进行可靠的测试。想像一个由工具完成所有测试的环境。您的顾虑是否会受到您单纯不信任开发人员可以进行良好的测试工作这一事实的支配？将测试思维左移（如我在上一篇文章中所述）应能减少这些疑惑。如果身为测试人员的您更像探险者一样行动来确定风险并对其进行评估、选择测试并确保将其整合到开发和自动化中，则您的顾虑可降到最低限度。

当然，您必须停止认为自己是质量的看门人、最后一道防线和唯一操心的人。您必须更像有远见者、风险识别者、风险管理者、探险者、促进者和教练/导师一样进行思考。

实践、监控和改进

有了降低或消除对晚期手动检查的信任的良好意图，缺陷仍然会通过测试。将软件发布到生产中时，问题就会产生。从运营的观点，DevOps 的关键准则之一是更深层次的监控。

每一层的监控，从应用程序中的组件和简单事务到整合与消息传递，当然，还有基础架构本身。监控的目标之一是发生故障时在用户受到故障的影响之前发出警报。这非常费劲，但却是终极目标。

在生产中遇到问题时，随后的任务是使用从监控中获得的分析，不仅要追踪原因并予以解决，而且要改善自动化或手动测试流程以降低未来发生类似问题的可能性。这里介绍并讨论了测试和分析在整个管道流程中的角色。[7]

您可以将 DevOps 流程中的自动化测试称为“监控”。与生产中的监控相结合，您可以说整个 DevOps 流程和生产中的监控会扩大测试的范围。因此，DevOps 不会降低测试人员角色的重要性。

结论

最近有人问我“组织中什么时候不应尝试 DevOps？”这个问题很好，但我认为在它背后是对 DevOps 是否在此停留以及测试人员是否应该予以注意的顾虑。我的回答很简单。

为什么您不想让开发人员和运营人员彼此对话？为什么您不想让更可靠的构建和部署进入测试和生产？为什么您不想让最好的技术支持更准确、高效和信息量丰富的管道？DevOps 是好东西，但并非总是容易实现。不用说，它需要文化变更，并且并非总是容易的。

对于测试人员，DevOps 在项目的早期阶段对我们产生更大影响，强迫我们更认真地思考测试、信息部署和决策制定中的自动化。测试人员需要拥护 DevOps，因为它提供在项目团队中变得积极主动并获得更多权威和尊重的机会。

关于作者

作为一名顾问、教师、作者、网络管理员、开发人员、测试人员、会议发言人、赛艇教练和出版人员，Paul Gerrard 执行过软件测试和质量保证领域所有方面的咨询任务，专门从事测试保证工作。他曾在欧洲、美国、澳大利亚和南非的各大测试会议上发表主题演讲，并且借此荣获诸多奖项。

2010 年，Paul 就读于牛津大学和伦敦帝国学院，并且荣获欧洲之星“欧洲测试卓越奖”，并于 2013 年荣获欧洲软件测试奖 (TESTA) 终身成就奖。

2002 年，Paul 与 Neil Thompson 合著“Risk-Based E-Business Testing”。2009 年，Paul 著“The Tester's Pocketbook”。2011 年，Paul 与 Susan Windsor 合著“The Business Story Pocketbook”，2014 年著“Lean Python”。

2014 年，Paul 担任都柏林欧洲之星会议的项目主席。

他是 Gerrard Consulting Limited 的负责人，TestOpera Limited 的主管，同时也是测试管理论坛的主持人。

邮件：paul@gerrardconsulting.com
Twitter：@paul_gerrard
网站：gerrardconsulting.com

有关更多信息，请访问 CA Technologies 的 **Develop & Test**。

联系 CA Technologies，网址：ca.com/cn

CA Technologies (NASDAQ: CA) 致力于开发促进企业转型的软件，为其抢占应用程序经济的先机。软件是各行各业的核心。从规划到开发再到管理和安全性，CA 正与全球各地的公司开展跨移动、私有和公共云、分布式和大型机环境的合作，以改变我们的生活、交易和沟通方式。要了解详细信息，请访问 **ca.com/cn**。

参考

1. “What is DevOps”，The Agile Admin，http://theagileadmin.com/what-is-devops/
2. “Frequency Reduces Difficulty”，Martin Fowler，http://martinfowler.com/bliki/FrequencyReducesDifficulty.html
3. “Testing and Checking Refined”，James Bach，Michael Bolton，http://www.satisfice.com/blog/archives/856
4. “A New Model for Testing”，Paul Gerrard，http://dev.sp.qa/download/newModel
5. “The New Model and Testing v Checking”，Paul Gerrard，http://blog.gerrardconsulting.com/?q=node/659
6. “How to Eliminate Manual Feature Checking”，Paul Gerrard webinar，http://blog.gerrardconsulting.com/?q=node/622
7. “Thinking Big: Introducting Test Analytics”，Paul Gerrard，http://blog.gerrardconsulting.com/?q=node/630

CS200-181159_0316