计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(9):87−94 [doi: 10.15888/j.cnki.csa.007540]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 DevOps 的轻量级持续交付方案①

王红蕾

(青岛科技大学 信息科学技术学院, 青岛 266061)
通讯作者: 王红蕾, E-mail: victy_wang@163.com

摘　要: 如何快速向用户交付可靠的产品, 是近年来持续交付研究和应用的热点问题, 传统的软件敏捷方法在交付过程中缺乏团队协作和标准化的构建流程, 大公司 DevOps 框架因体系复杂在中小规模企业应用时会产生迭代速度和产品质量相互制约的矛盾, 本文提出一种基于 DevOps 轻量级的持续交付框架, 在角色叠加、迭代频繁的项目场景中, 以脚本形式自动实现项目代码获取、测试、构建和部署, 完成项目持续交付. 通过行业调查、企业实践验证方案不仅可以缩短项目周期、提高交付质量, 还能够实现交付过程可视化, 促进软件质量不断改进.

关键词: DevOps; 项目迭代; 持续交付; 流水线脚本化; 问卷调查

引用格式: 王红蕾.基于 DevOps 的轻量级持续交付方案.计算机系统应用,2020,29(9):87–94. http://www.c-s-a.org.cn/1003-3254/7540.html

Lightweight Continuous Delivery Solution Based on DevOps

WANG Hong-Lei

(Information Science and Technology Academy, Qingdao University of Science Technology, Qingdao 266061, China)

Abstract: In recent years, how to deliver reliable products to users quickly has become a hot issue of continuous delivery research and application. Traditional agile software methods lack of team cooperation and standardized construction process in the delivery process. Due to the complexity of the system, the DevOps framework of large companies will produce the contradiction between iteration speed and product quality when applied in small and medium-sized enterprises. This study proposes a method based on DevOps, which is a lightweight continuous delivery framework. In the project scenario with overlapping roles and frequent iterations, it automatically achieves project code acquisition, testing, construction, and deployment in the form of script, and completes the continuous delivery of the project. Through industry survey and enterprise practice, the scheme can not only shorten the project cycle and improve the quality of delivery, but also realize the visualization of delivery process and promote the continuous improvement of software quality.

Key words: DevOps; project iteration; continuous integration; pipeline as code; survey interview

随着互联网市场竞争日趋激烈, 市场逐步细分促使产品的个性化需求不断增加, 众多互联网产品如何在激烈的市场竞争中获得用户青睐, 能够快速适应市场需求变化并能实现高质量持续交付是关键. 为适应不断变化的市场需求, 产品的持续交付需要重复整合需求分析、设计、编码、测试、部署和维护各阶段工作, 大量重复、缺乏协作的工作严重制约团队效率和产品质量提高, DevOps 模式下的 pipeline 流水线可以实现从版本控制库到交付用户过程的自动化[1], 成为互联网产品开发与管理的可行途径.

1　国内外研究现状

近年来, DevOps 在国内外已有广泛的研究和实践, 并在不同领域、不同行业证实基于 DevOps 模式的项

① 收稿时间: 2019-12-05; 修改时间: 2020-01-03; 采用时间: 2020-03-11; csa 在线出版时间: 2020-09-04

目研发, 在系统质量控制和缩短客户的交付响应时间上有突出的表现[2–4], 可以显著提升产品持续交付的质量和效率[2,4,5]. 然而, DevOps 的实践没有固定模式, 受制于公司的规模、架构和文化, 呈现多样性特征. Roche 结合自己在 Amazon, Adobe 公司大型项目研发经验, 提出了一种多角色协同的 DevOps 模式, 并研究了该模式下的质量保障体系[6]; Kamuto 等研究了大型分布式项目中 DevOps 实施的各类制约因素[7]; Rajkumar 等研究了 DevOps 文化在基于云环境下的产品交付和开发的影响, 指出 DevOps 模式需要对组织机构、管理策略等进行较大的改变[8]. 而由于交付的频繁, 能够设计符合企业实际的自动化交付方案, 确保软件处于一个稳定的、可随时发布的状态, 是持续交付实践的关键. Krusche 等提出了利于持续交付工作流快速交付产品的模型[9]. Düllmann 等提出一种基于 DevOps 实践的高可靠性的持续交付流水线实现方案[10], Shah 等探讨了采用 Jenkins、GIT 等自动化工具构造持续交付流水线的可能模式[11]. 文献 [4,12] 调研了国内外众多知名企业 DevOps 实践, 实践过程中使用的自动化工具进行调查, 其中 Jenkins、GIT、Docker 被广泛应用[13]. 随着 DevOps 的日益成熟, 其逐渐成为大型互联网企业首选模式, 而中小规模企业实施持续交付时完全借鉴现有的 DevOps 模式, 难以落地. 一方面企业规模、管理模式、人员结构存在差异, 完全抛弃企业原有的辅助工具、管理模式成本较大; 另一方面随着发布频率提高, 产品维护和升级操作频繁, 身兼数职的人员难以完成工作.

针对 DevOps 模式在中小规模企业实践存在的问题, 提出了一种轻量级的持续交付方案, 将软件交付流程进行优化, 通过调查问卷的形式了解区域互联网企业的 DevOps 应用现状和特点, 以企业实践和组织成员访谈论证了方案的可行性.

2 轻量级持续交付方案设计

以灵活的自动化形式实现从版本控制库到交付用户的方案, 首先要选取相应的工具, 目前在版本控制、代码扫描、自动化测试、构建和部署等方面的主流使用工具如表 1 所示.

以开源性、可扩展性和操作简便性为原则, 本文使用 GIT[14] 作为版本控制工具以 Jenkins[13] 为持续交付服务, 集成 sonar 进行增量式代码扫描, 以 Maven 完成自动编译和打包, 在 Robot Framework[15] 框架下编写自动化测试用例.

表 1 主流工具对比

工具	应用阶段	优势	劣势
Git	版本控制	分布式版本控制, 可扩展性好, 分支管理功能强大	开发人员可克隆下整个版本库保密性较低
SVN	版本控制	集中式版本控制, 管理方便	版本库出现宕机不能进行提交、还原等任何操作
Sonar	代码扫描	可按照质量模型进行评估	需搭建独立的服务
Find Bugs	代码扫描	基于字节分析, 侧重运行时错误检测	作为插件与开发工具IDE集成, 可扩展性较差
Maven	构建	自动管理依赖, 可自动执行编译、打包等过程	对项目的框架有特殊要求
Ant	构建	按指令编译、打包	没有集成依赖管理
Selenium	UI自动化	支持Java和Python语法规范, 用WebDriver驱动UI测试	编写难度大, 对开发脚本能力要求较高
Robot Framework	自动化测试	集成Selenium和Request库, 进行UI和接口测试有封装好的关键字	需安装多个支持库, 配置过程较复杂
Jenkins	集成部署	跨平台集成, 可扩展性高, 支持分布式构建	插件与Jenkins内核时会出现冲突
TeamCity	集成部署	支持多种工具和框架	付费

新功能的开发或缺陷问题的修复产生软件版本的迭代, 开发人员将产生迭代的代码提交到版本控制库时便触发一次产品的交付. 这包括将代码集成后交付给测试人员进行人工验证以及在生产环境上线部署交付到用户使用. 由于项目的频繁更新迭代, 将记录交付测试、用户的流水线脚本和测试脚本以版本库的形式实现保证了交付过程能够持续进行[4], 一个完整的轻量级持续交付方案如图 1 所示.

(1) 无论是新功能开发完成还是线上问题修复均需要在测试环境验收通过才能交付用户, 项目交付到测试人员时, 代码编写规范、空指针、内存溢出等规范性和阻塞性问题要尽早发现, 同时保证本次提交不能对以往版本产生影响, 因此项目远程仓库更新后, 持续交付服务自动调用已编写有代码审查、单元测试、打包和回归测试的流水线构建脚本. 执行完成后以邮件的形式通知测试和开发人员, 若执行失败开发人员进行代码修复, 反之成功则将当前版本打包归档, 测试人员开始冒烟测试和系统测试, 测试通过等待上线. 交付到测试的流水线脚本结构如图 2 所示.

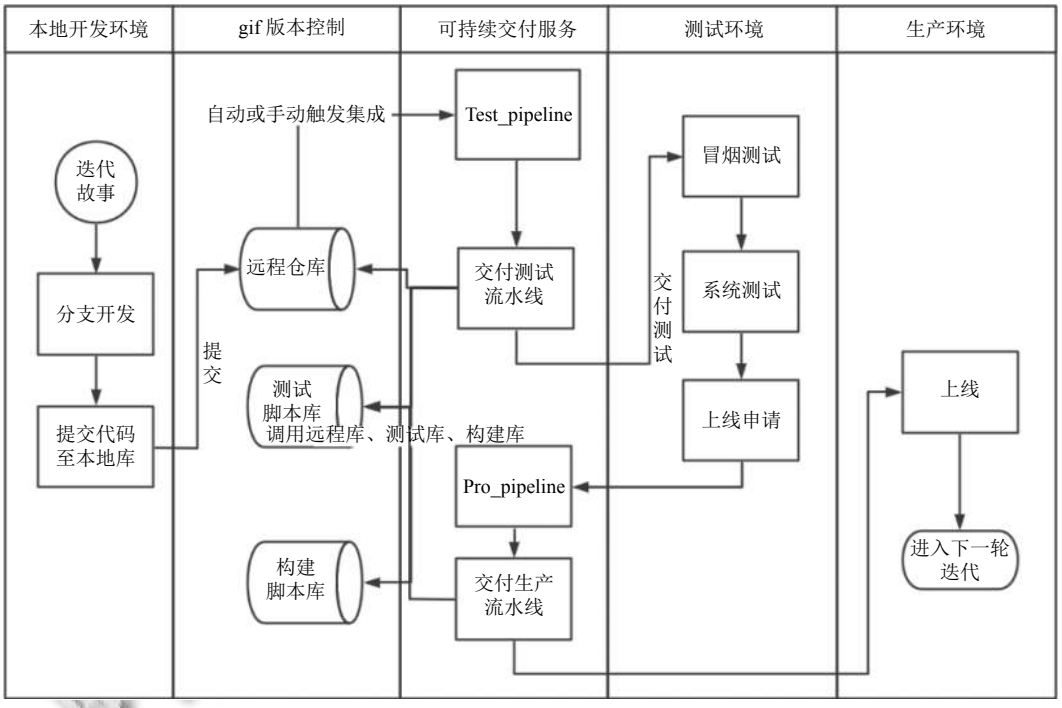

图 1 轻量级持续交付方案

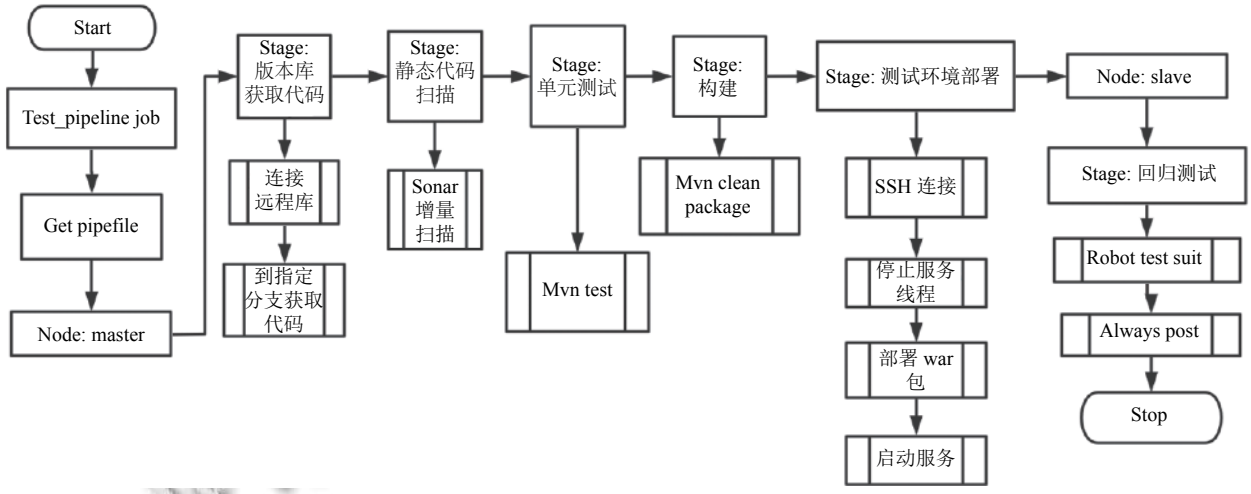

图 2 交付测试的流水线脚本结构

(2) 为了实现后续快速上线, 执行交付用户的流水线过程如下:

① 代码获取: 新代码在测试环境验收通过后手动合并到主分支, 通过 git clone 的命令获取主分支上的代码以便在后续的阶段中部署到生产环境.

② 构建: Maven 项目的构建以 mvn clean package 命令完成编译和打包的过程.

③ 部署: 部署时实行停机更新时是先停止服务的进程, 把项目打包完成后将 war 包放到指定路径启动服务.

④ 通知上线结果, 应用构建部署成功则通知相关人员并将当前版本的包归档, 若部署失败则相关人员根据构建日志进行问题修复.

交付用户的流水线过程如图 3 所示.

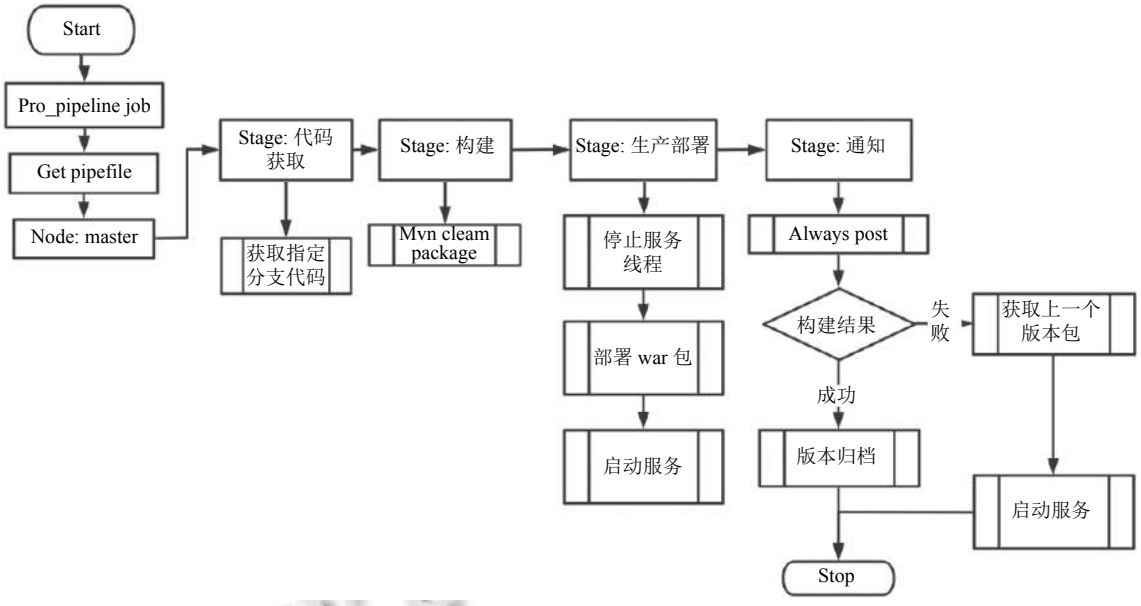

图 3 交付用户的流水线结构

3 实验

3.1 行业调查

为调查青岛市互联网企业组织现状, 反映 DevOps 应用特点, 为企业实践提供组织管理依据, 本文通过第三方调查平台发起调查问卷. 问卷设置问题均为客观选择题, 第一部分组织管理问题, 包括: 组织实施模型、团队自动化程度、员工培训投入、团队 DevOps 经验, 第二部分为 IT 组织性能调查, 问题包括项目交付周期和部署频率.

问卷中将组织模型分布设置为 4 类, 主要探讨不同模型在项目交付上的区别. 其中瀑布模型依次进行设计、开发、测试和运维过程, 开发周期长, 项目成果整体交付[16]; 敏捷和瀑布混合模型下交付方式视项目需求而定, 在新功能的引入和缺陷修复中使用敏捷方法[17]; 敏捷模型项目采用敏捷开发方法逐渐迭代[18]; DevOps 模型则高度引入自动化测试, 实现持续集成和持续交付.

经过统计共收到 72 份有效的问卷, 其中仅有 9% 的企业组织过程能采用到 DevOps 过程模型, 有 24% 的企业能够实施敏捷模型, 而 67% 的互联网企业仍然采用瀑布模型或瀑布模型与敏捷相混合的过程. 组织模型的分布如图 4 所示.

① 员工培训投入包括: 未对员工进行任何新技术培训; 关注员工业务知识培训; 定期组织技术分享会议. 在统计中 85% 的 DevOps 企业定期组织技术分享会议, 占整个调查的 30%.

② 自动化程度: 无自动化工具; 仅使用自动化进行问题验证; 有涉及集成和交付的自动化岗位. 在统计中仅 27.8% 的公司能够设立专职的自动化岗位, DevOps 企业均有专职的自动化岗位, 占其中的 35%.

③ 团队 DevOps 经验调查的问题包括: 无相关经验; 仅对 DevOps 有学习了解; 关键领导岗位有 DevOps 实践经验. 其中对 DevOps 有学习了解或经验的占总数的 43.05%, 而所有的 DevOps 应用企业关键领导岗位均有 DevOps 实践经验.

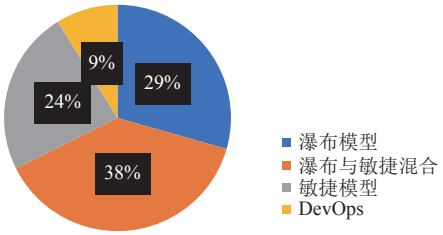

图 4 组织模型分布

结论 1: DevOps 在青岛互联网企业中应用较少. 而企业要成功实施 DevOps, 一方面要重视团队技术培训的投入提高组织自动化程度, 另一方面, 一个有较强领导力的团队领导也是 DevOps 能够成功实施的重要因素.

如图 5 所示, 在组织模型与交付周期关系的图表中, 瀑布模型或瀑布与敏捷混合的两个过程中分别有 30% 和 7.69% 的企业存在项目延期的情况, 分别有 40% 和 61.65% 的企业交付周期在 3~4 周完成交付, 仅有 30% 和 30.77% 的企业交付周期能实现 2 周完成一次交付, 无企业在这两种形式下进行 1 周完成多次的项目交付, 而敏捷模型和 DevOps 模型的企业中有 12.50% 和 66.67% 的企业能在一周完成多次的交付.

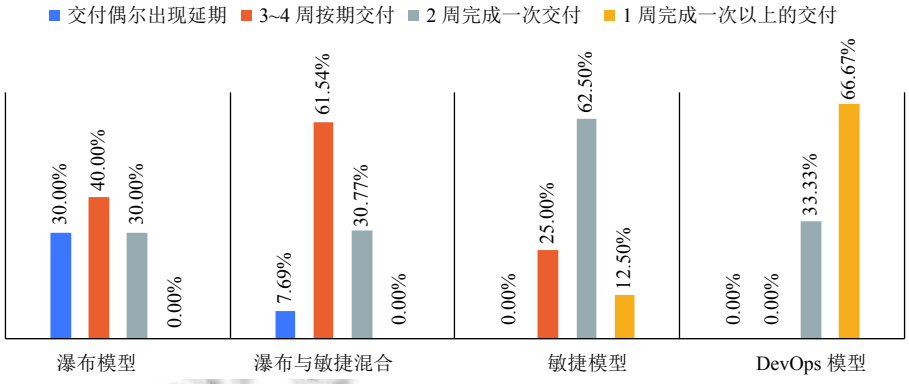

图 5 组织模型与交付周期

如图 6 在组织模型与部署频率关系的图表中, 采用瀑布模型的企业 70% 的部署频率为每月部署一次, 无企业的部署频率在一周部署多次; 瀑布和敏捷混合的组织模型中相比较瀑布模型的部署频率更多的达到每 2 周部署一次, 仍然无企业实现 1 周部署多次的情况; 在敏捷模型过程中 75% 的企业能实现 1 周部署一次, 无企业部署按月进行, 相比较前 2 种的频率有所增高, DevOps 模型中首次出现 1 周部署多次的情况, 所占比例为 33.33% 但大部分即 66.67% 的部署频率在一周部署一次, 相比较前 3 者的部署频率最高.

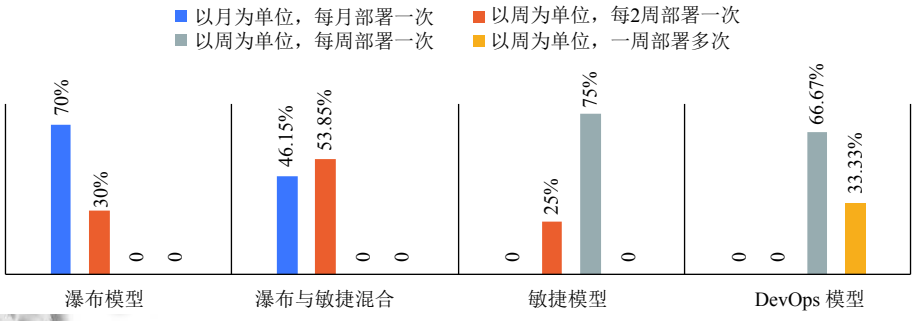

图 6 组织模型与部署频率

结论 2: 随着企业采用的组织模型由瀑布模型向敏捷模型、DevOps 模型的过程转变项目交付周期逐渐缩短, 相对应的部署频率也就越高, 这与敏捷模型和 DevOps 模型提倡项目迭代和持续集成的目标是一致的.

3.2 轻量级持续交付方案企业应用分析

自 2019 年 1 月, S 公司开始在原有敏捷开发的基础上采用轻量级的交付方案, 并在持续交付服务基础上研发了持续交付管理平台, 如图 7 所示的流水线管理模块, 在新建一条流水线后可灵活选择验证过程和回归测试用例, 对交付流水线进行修改、查看详情、执行流水线以及删除操作. 如图 8 所示的项目迭代管理是对上线功能的版本管理同时进行发布.

图 7 流水线管理模块

图 8 项目迭代管理

经过一段时间的实践在项目交付能力和交付质量上有较明显提升. 如图 9 所示, 以交付周期和部署频率体现项目的交付能力, 敏捷方法的交付周期和部署频率在 10 天左右, 采用轻量级方案后交付周期和部署频率有下降到 6 天作用, 交付能力得到提高.

项目变更失败比例是指在项目上线时需要进行系统修复的比例, 包括代码回滚、服务重启、缺陷修复等现象, 以项目变更失败比例来衡量项目交付质量. 如图 10 所示, 采用流水线方案前失败比例在 8.5% 上下浮动, 而采用方案后交付过程进行标准化操作, 降低了人为操作失误, 同时测试更加充分, 失败比例逐步降至 4% 左右.

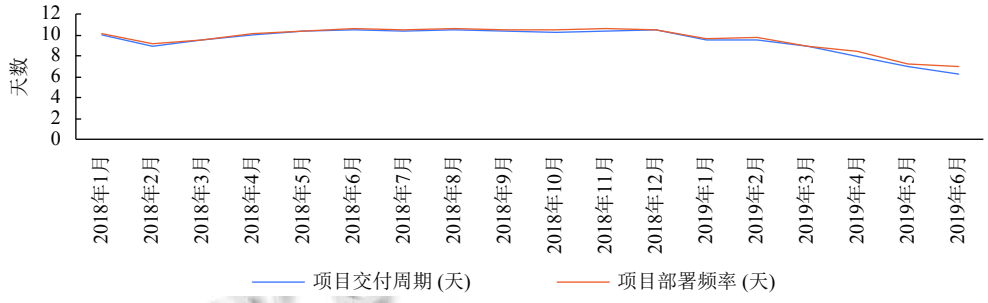

图 9 项目交付能力对比

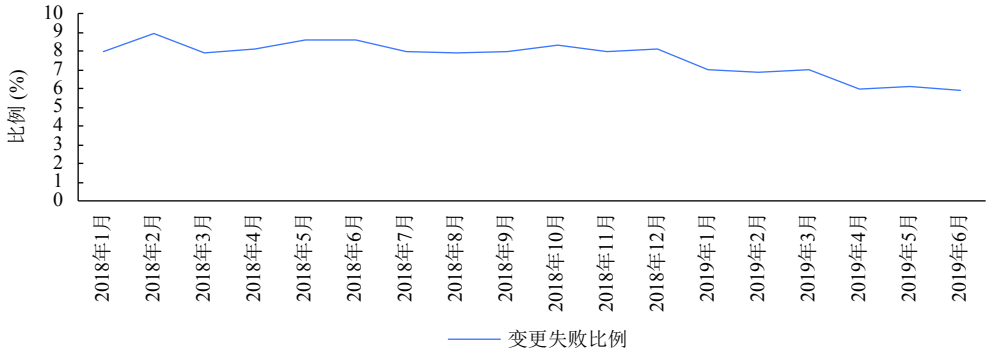

图 10 项目变更失败比例对比

修复问题的过程是开发人员修改完代码后交由测试人员进行验证, 验证通过交由运维部署上线, 如图 11 所示, 原敏捷过程通过查找代码提交记录和运维记录可粗略统计平均在 3.5 小时. 而采用流水线形式之后自动化回归验证通过可直接上线, 用时逐步下降到 2.5 小时.

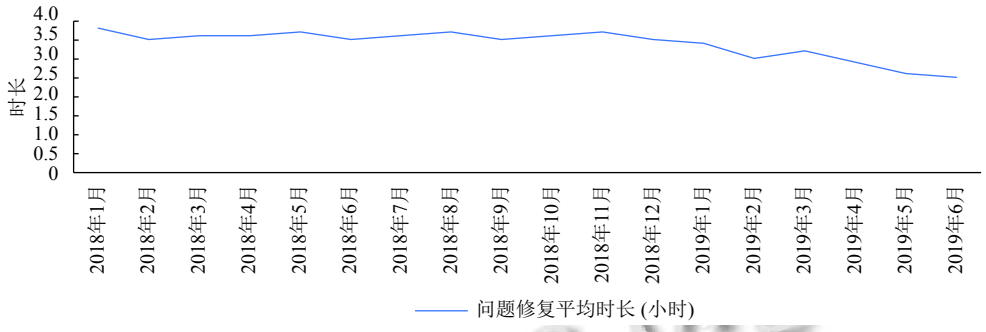

图 11 问题修改平均时长对比

4 总结

本文在敏捷过程的基础上将项目交付中重复度高的工作利用自动化工具, 使用流水线形式实现, 通过问卷调查明确企业要实施 DevOps 需要进行的组织管理改进, 对存在角色重叠的互联网企业使用流水线前后的组织性能包括项目交付能力和交付质量进行对比, 为企业实施 DevOps, 实现持续交付提供了参考.

本文的不足之处主要体现在两个方面, 一方面在流水线方案中, 一个流水线执行完测试验证需要上线时要进行申请, 经过专人确认后方能上线, 需要申请的原因是当有多个功能需要上线时防止项目代码相互覆盖, 如何有效的解决上线等待问题将是今后的研究方向. 另一方面本文调查问卷仅对青岛市企业进行统计且企业数量相对偏少, 在今后将对不同地区互联网企业的持续关注, 不断完善调查报告.

参考文献

1 Humble J, Farley D. 持续交付: 发布可靠软件的系统方法. 乔梁, 译. 北京: 人民邮电出版社, 2011.

2 Gupta RK, Venkatachalapathy M, Jeberla FK. Challenges in adopting continuous delivery and DevOps in a globally distributed product team: A case study of a healthcare organization. Proceedings of 2019 ACM/IEEE 14th International Conference on Global Software Engineering. Montreal, QC, Canada. 2019. 30–34.

3 Punjabi R, Bajaj R. User stories to user reality: A DevOps approach for the cloud. Proceedings of 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology. Bangalore, India. 2016. 658–662.

4 Liu YH, Li CB, Liu W. Integrated solution for timely delivery of customer change requests: A case study of using DevOps approach. International Journal of U- and E-Service, Science and Technology, 2014, 7(2): 41–50. [doi: 10.14257/ijunesst.2014.7.2.04]

5 翟志军. JenKins 2.x 实践指南. 北京: 电子工业出版社, 2019.

6 Roche J. Adopting DevOps practices in quality assurance. Communication of the ACM, 2013, 56(11): 38–43. [doi: 10.1145/2524713.2524721]

7 Kamuto MB, Langerman JJ. Factors inhibiting the adoption of DevOps in large organisations: South African context. Proceedings of 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology. Bangalore, India. 2017. 48–51.

8 Rajkumar M, Pole AK, Adige VS, *et al*. DevOps culture and its impact on cloud delivery and software development. Proceedings of 2016 International Conference on Advances in Computing, Communication, & Automation. Dehradun, India. 2016. 1–6.

9 Krusche S, Alperowitz L, Bruegge B, *et al*. Rugby: An agile process model based on continuous delivery. Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. Hyderabad, India. 2014. 42–50.

10 Düllmann TF, Paule C, Van Hoorn A. Exploiting DevOps practices for dependable and secure continuous delivery pipelines. Proceedings of 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering. Gothenburg, Sweden. 2018. 27–30.

11 Shah J, Dubaria D, Widhalm J. A survey of DevOps tools for networking. Proceedings of 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference. New York, NY, USA. 2018. 185–188.
12 金泽锋, 张佑文, 叶文华, 等. 面向完整价值交付的文档 DevOps 应用研究. 软件学报, 2019, 30(10): 3127–3147.
13 黄璜, 张贺, 邵栋. 自动化工具对中国 DevOps 实践的影响. 软件学报, 2019, 30(10): 3056–3070. [doi: 10.13328/j.cnki.jos.005788]
14 Soni M. End to end automation on cloud with build pipeline: The case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery. Proceedings of 2015 IEEE International Conference on Cloud Computing in Emerging Markets. Bangalore, India. 2015. 85–89.
15 Stresnjak S, Hocenski Z. Usage of robot framework in automation of functional test regression. Proceedings of the 6th International Conference on Software Engineering Advances. Barcelona, Spain. 2011. 30–34.
16 邢斌斌, 姚郑. CMM/CMMI 与软件生命周期模型关系的研究. 计算机应用研究, 2007, 24(11): 65–69. [doi: 10.3969/j.issn.1001-3695.2007.11.019]
17 闫帅. 中小型企业使用瀑布模型与敏捷开发相结合的产品开发项目管理方法. 计算机光盘软件与应用, 2011, (1): 152–153.
18 容国平, 张贺, 邵栋, 等. 软件过程与管理方法综述. 软件学报, 2019, 30(1): 62–79. [doi: 10.13328/j.cnki.jos.005645]