

# 持续集成系统可视化设计研究

张晓帆<sup>1</sup>, 刘 宁<sup>1</sup>, 潘 帆<sup>2</sup>

(1. 中国电子科技集团第10研究所, 四川 成都 610036;  
2. 四川大学 电子信息学院, 四川 成都 610207)

**摘 要:**持续集成/持续交付是敏捷软件开发的核心实践,而持续集成/持续交付的有效实施要求团队在信息高效共享情况下的有效协作。高效的协作在中型或大型组织中始终是一个挑战,而将持续集成/持续交付过程可视化,可以大幅提升信息共享的效率和效果。文中完整提出了可视化的体系设计和参考架构,该设计将各种信息进行有序合理的组织,以最有效的方式呈现或推送给相关人员,连接并牵引相关人员采取及时的行动以提升产品交付的效率和质量。同时还陈述了可视化设计背后的思考和决策因素,使该可视化设计在不同情况下可进行适当的定制,以适应不同的应用场景。该方法已在大中型的软件组织中成功落地实践。提出的设计和架构,具有实际的工程应用参考价值。

**关键词:**可视化;持续集成;持续交付;敏捷;DevOps

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2020)02-0058-05

doi:10.3969/j.issn.1673-629X.2020.02.012

## Research on Visualization Design of Continuous Integration System

ZHANG Xiao-fan<sup>1</sup>, LIU Ning<sup>1</sup>, PAN Fan<sup>2</sup>

(1. Electronics Technology Group Corporation No 10 Institute, Chengdu 610036, China;  
2. School of Electronic Information, Sichuan University, Chengdu 610207, China)

**Abstract:** As the core practice of Agile, the success of continuous integration and continuous delivery require a well cooperated team with information shared in an efficient way. Efficient collaboration is always a challenge in large or medium-sized organizations, and visualization of continuous integration and continuous delivery is to improve the efficiency and effect of information sharing. In this study, we give the full picture of design and architecture of visualization system. It organizes the information in a right manner, and pushes the information to the right person in an effective way, so that right action can be taken to improve the efficiency and quality of product delivery. At the same time, we give the thinking behind the visualization design, so that the design can be customized in different conditions for different scenarios. The method presented has been successfully applied in large and medium-sized software organizations. The design and architecture given can be a reference for a continuous delivery system.

**Key words:** visualization; continuous integration; continuous delivery; Agile; DevOps

## 0 引 言

作为敏捷软件开发的核心实践,持续集成/持续交付向开发和交付团队提供及时有效的反馈,以持续提升开发和交付的效率和质量<sup>[1-3]</sup>。持续集成/持续交付成功实践的关键在于开发/测试/交付团队间的紧密合作。然而,不同的团队具有不同的背景和领域知识,因此紧密合作要求集成/交付相关信息高效地传递到正确的团队/人员,以便在正确的时间采取正确的行动。“高效”的沟通在中型或大型组织中始终是一个挑战,而将持续集成/持续交付过程进行“可视化”将

大大有助于信息的高效传递。

目前,对持续集成/持续交付的研究主要集中于对集成/交付步骤和各步骤相关工具的研究,对全过程可视化的研究相对较少。文献[4-5]提出了由开发集成,验收测试,部署发布构成的持续交付基本过程,并提出了对全过程进行监控的概念。文献[6]提出了由构建,单元测试,集成测试,验收测试构成的持续交付的验证步骤。文献[7]提出了 DevOps 常用的 CI/CD 工具集合。文献[8]提出了通过可视化方法,对敏捷团队中各成员的贡献进行统计和呈现。

收稿日期:2019-03-08

修回日期:2019-07-10

网络出版时间:2019-11-07

基金项目:国家自然科学基金(U173109)

作者简介:张晓帆(1979-),男,硕士,工程师,研究方向为计算机应用、软件工程、系统设计。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20191107.0910.040.html>

针对上述问题,文中提出一种持续集成(CI)/持续交付(CD)的可视化设计体系和方法,包括:哪些信息需要可视化;信息如何可视化;信息如何被动或主动地传递到合适的团队;在发现问题时,相应团队该如何采取行动。同时,提出该“可视化”系统的实现架构参考,包括由业界主流开源软件 Jenkins、BitBucket、Jira 等工具组成的系统架构。该体系设计和架构有效地将

各种信息呈现或推送给相关人员,提升了持续集成/持续交付的实施效率,帮助团队有效提升了开发和交付的质量和效率。

1 可视化体系设计

持续集成/持续交付系统的可视化体系设计如图 1 所示。

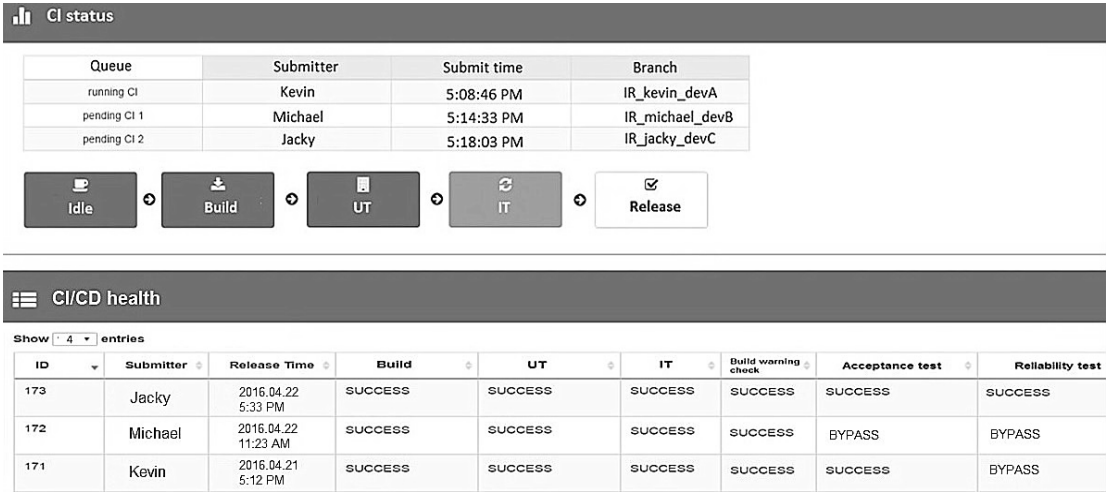


图 1 CI/CD 可视化视图

该体系由两部分构成:

持续集成(CI)状态视图:该部分主要向开发团队传递集成过程相关信息。对比文献[9-10],文中提出了 CI 排队过程和队列视图。

持续集成(CI)/持续交付(CD)健康视图:该部分主要向开发/测试/交付团队传递产品交付相关信息。文中提出了 CI/CD 核心阶段和扩展阶段的概念,并针对不同阶段的重要程度和反馈成本,提出了多种系统同步和可视化通知手段。

1.1 持续集成(CI)状态视图

该部分显示了对开发团队很重要的 CI 状态信息,包括请求排队状态和集成流水线状态。典型的 CI 过程如图 2 所示,包括排队阶段和流水线阶段。

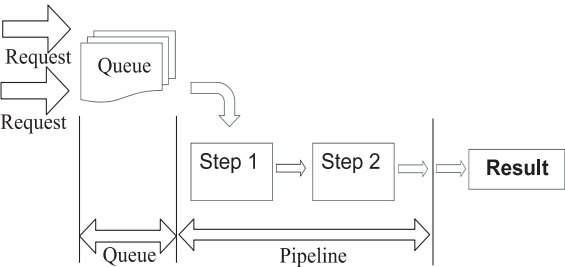


图 2 CI 排队和流水线

(1)CI 队列视图。

在中等规模的组织(约 100 个开发工程师)中,考虑每个 CI 持续时间为 10 分钟,如果每个工程师每日提交集成请求,则集成请求将不得不在 CI 系统中排队以依次处理。

CI 队列视图可以让每个工程师知道 CI 系统的繁忙程度,了解 CI 系统是否接受了集成请求,并预测何时可以处理该集成请求。因此,CI 相关的信息都应在此展示,如图 3,包括请求提交者名称、提交者时间以及为集成提交的分支。

Queue	Submitter	Submit time	Branch
running CI	Kevin	5:08:46 PM	IR_kevin_devA
pending CI 1	Michael	5:14:33 PM	IR_michael_devB
pending CI 2	Jacky	5:18:03 PM	IR_jacky_devC

图 3 CI 队列视图

(2)CI 流水线视图。

如图 4,该视图让每个工程师知道其集成请求处理的状态,不同的颜色给出了 CI 流水线各步骤的不同状态。典型的 CI 步骤包括:系统空闲→构建→UT→IT→内部发布。



图 4 CI 流水线视图

当某步骤失败时,CI 系统该如何处理? 不同的团队有不同的看法,有团队建议暂停 CI 流水线并向团队发出警报,有团队建议直接回滚版本并给提交者以提示。根据笔者数年的实践经验,总结该部分的设计如下:

· 如果该系统服务于一个小团队,集成系统的吞吐量不会成为问题,团队的沟通和立即行动是可行的。在这种情况下,CI 失败时,可以选择暂停 CI 流水线并向团队发出警报。

· 如果该系统服务于中/大规模的团队,则集成系统吞吐量将成为问题。与此同时,大中型团队的沟通问题会更加严重,团队成员对集成时发现的问题的响应会更为迟缓。在这种情况下,CI 系统自动回退版本并给提交者以提示,腾空 CI 系统以使系统可以处理下一位工程师的集成请求。这种方式可以使集成系统保持高吞吐量,以提升中/大规模团队的持续集成效率。

1.2 持续集成 (CI)/持续交付 (CD) 健康视图

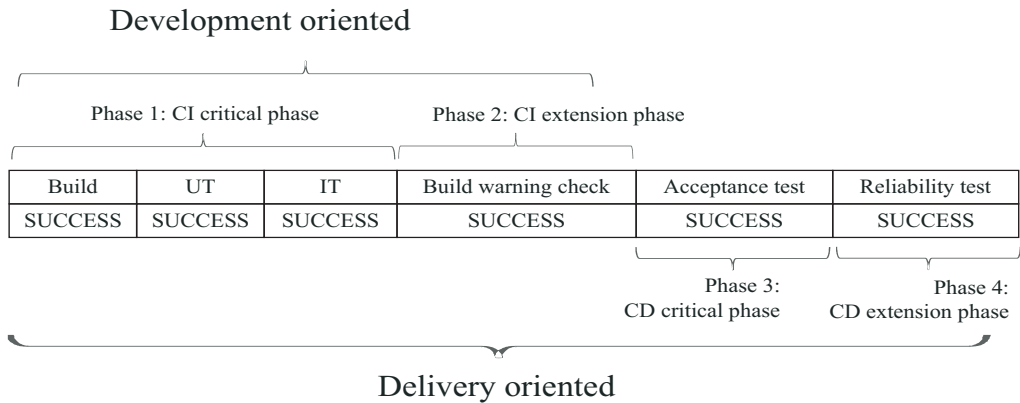


图 5 版本 CI/CD 流水线视图

(1)持续集成 (CI) 核心阶段:build/UT/IT。  
该部分呈现了 build/UT/IT 的健康状态。  
此阶段中的任何步骤失败都将导致 CI 失败。如果发生任何失败,都不会进行新版本(内部)发布,并且主干版本将回滚到最后一次成功发布的版本,并将邮件发送给提交者以指示失败原因。

(2)持续集成 (CI) 扩展阶段:代码静态检查。  
此阶段主要完成代码的静态检查,包括 build warning,或通过 lint 工具进行的代码规范静态检查,如文献[11]。如果此阶段中的步骤失败,流水线不会终止,但开发团队会收到一封警告邮件。

工程师需要从 CI 获得快速反馈,需要通过 CI 在每天进行多次构建和软件测试,因此单次 CI 的时间需要缩短。对于某些不太重要,并且耗时的验证步骤,如代码静态检查,或代码圈复杂性分析,都可以将其置于扩展阶段。由于这些验证处于扩展阶段而非核心阶段,当在此阶段检查出问题时,比如引入大量新构建警告时,因为没有版本回退机制,开发团队可能会也可能不会修复这些问题。笔者在过去几年遇到了这个问题:越来越多的 build warning 被引入集成主线,而没有人采取行动。因此,在可视化设计中,将扩展阶段的结果公示到整个团队,让整个团队知道谁在哪个版本引入新警告,使得提交者有压力不要引入 build warning 或尽快修复它们。

(3)持续交付 (CD) 核心阶段:验收测试 (user acceptance test)。

该部分视图呈现每个版本的 CI/CD 健康状况,开发/测试/交付团队都应该关注该部分呈现的信息:

- 开发/测试团队应采取行动解决测试中发现的各种问题;
- 交付团队可以根据各版本的健康状况,选择合适的版本以进行交付。

每个版本的 CI/CD 过程也是一个流水线。该流水线通常有 4 个阶段,如图 5 所示。

此阶段从产品交付的角度,进行验收测试。如文献[12-13],验收测试通常自动化进行。如果验收测试通过,该版本可以进行发布。如果验收测试失败,整个 CI/CD 系统将被暂停,并向开发团队进行红色警报,以便推动开发团队及时修复在验收测试中发现的问题。

通常,从用户角度进行的验收测试会耗费较长时间,比如 1~2 小时或更长。考虑到 CI 的典型持续时间为 10 分钟,对每个通过 CI 的版本进行验收测试是不可行的。在该系统中,采用异步机制以处理 CI 过程和验收测试过程难以同步的问题:验收测试轮询最新(内部)发布的 CI 版本以开始测试,这里某些 CI 版本可能被“绕过”,如图 6 所示。

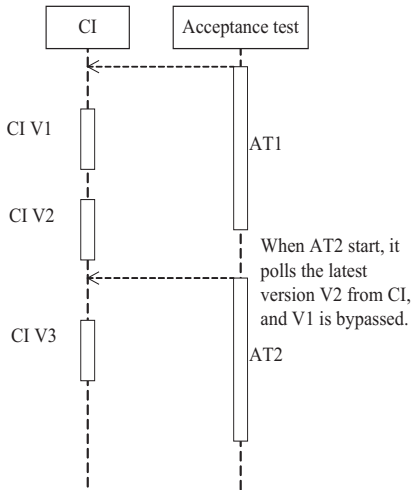


图 6 异步执行长时间的验收测试

因为 CI 和验收测试采用了异步的设计,在验收测试失败的情况下,集成流水线该如何进行?如图 6,如果 AT2 失败,系统如何处理,如何向用户进行反馈。通常解决方法有 3 种:

- 通过邮件指示失败,并且 CI/CD 系统继续处理新的 CI 请求。
- V2 失败,则回退以使用 V1 进行验收测试。
- 暂停整个流水线直到 AT2 失败的问题修复,并向整个团队及时告警通知。

根据实践,方法 1 存在开发团队没有动力/压力来解决 AT 故障的问题,因为开发团队的新 CI 请求可以继续处理。当越来越多的新 CI 请求集成入主线时,主线版本的质量持续恶化。方法 2 也有一些限制:首先,它使系统更复杂,其次,它加长了反馈周期,因为它需要长时间重新运行 V1 的 AT。方法 3 是最终使用的方法,当 AT2 失败时,系统会自动采取 3 项行动,迫使开发团队立即采取行动以修复问题:

- ①CI 系统将被暂停,开发团队不能再提新的 CI 请求,迫使开发团队必须采取行动;
- ②邮件将广播给开发团队,提示验收测试(AT)在 V1 或 V2 版本失败,CI 系统暂停;
- ③系统将可视化“红色警报”,如图 7 所示。

表 1 CI/CD 可视化总结

环节	关键步骤	发现问题时,系统如何自动处理	如何提示团队	开发/测试团队如何响应	交付团队如何响应
持续集成(CI)核心阶段	build/UT/IT	版本回退	系统可视化前端展示,并邮件通知发现的问题	发现问题时,修复并连同原改动重新提交	不涉及
持续集成(CI)拓展阶段	代码静态检查	继续版本发布	系统可视化前端展示发现的 warning,并邮件通知失败原因	发现问题时,修复并提交修复的改动	不涉及
持续交付(CD)核心阶段	验收测试	暂停并锁住系统	系统可视化前端展示,邮件通知失败原因,并在团队高亮公示该失败状态	发现问题时,立刻修复问题,提交修复的改动并解锁系统	成功通过该阶段的版本,将是潜在的候选发布版本
持续交付(CD)扩展阶段	可靠性测试	版本继续发布,记录发现的问题	系统可视化前端展示,并邮件通知发现的问题	发现问题时,修复问题并提交修复的改动	成功通过该阶段的版本,将是高信心的候选发布版本

2 实现架构参考

可视化系统的实现架构参考如图 8 所示。

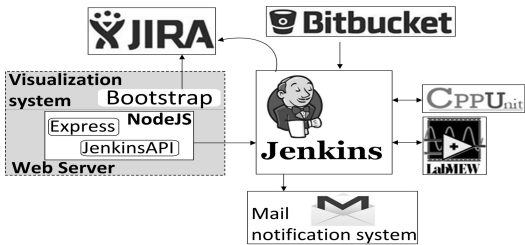


图 8 可视化系统架构设计

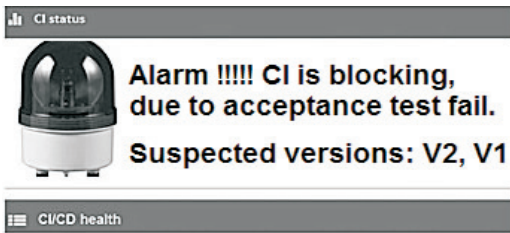


图 7 可视化“红色警报”(针对验收测试失败)

(4) 持续交付(CD)扩展阶段:可靠性测试(reliability test)。

不仅可靠性测试,其他测试,如性能测试、探索测试等也可以在该阶段进行。

如果扩展阶段的验证失败,则应在问题跟踪系统(如 Jira)中记录和跟踪问题,以便开发团队可以跟进。如果可靠性测试通过,对于交付团队,该版本将是更有信心的候选交付版本。可靠性测试的典型持续时间为 1~2 天,因此,与 CI 和验收测试的异步设计类似,可靠性测试也以异步方式运行:并非所有通过验收测试的版本都会进行可靠性测试,可靠性测试将轮询通过验收测试的最新版本以进行测试,并且可以绕过某些版本。

表 1 总结了 CI/CD 系统的可视化反馈/指示,以及各团队所需的响应。

可视化服务是采用 NodeJS 框架的 Web 服务,用于显示 CI/CD 的状态和数据。

如文献[14],CI/CD 系统由多种子系统构成,其核心为 Jenkins。Jenkins 提供了流水线功能,并提供了 CI/CD 系统可视化所需的数据。

(1)可视化服务需要查询 Jenkins 的数据,如提交者名称,时间,构建和测试结果的种类。

Node.js 提供了一个模块和接口,以查询 Jenkins 的数据。下面是数据查询样例代码片段:

```
var jenkinsapi = require('jenkins-api');
# Get last build
```



```
var jobName = 'Jenkins-build-jobs';
jenkinsapi.last_build_info(jobName, function(err, data) {
  callback(err, data);
});

# Get one specific build
var jobName = 'Jenkins-build-jobs';
var buildNum = 10;
jenkinsapi.build_info(jobName, buildNum, function(err,
data) {
  callback(err, data);
});
```

通过查询接口,如下 JSON 格式的数据将从 Jenkins 返回 Node.js 的模块。

```
"builds" :
{
  "buildNumber" : 294,
  "duration" : "4 min",
  "icon" : "blue.png",
  "jobName" : "A-Jenkins-BuildJob",
  "parentBuildNumber" : 384,
  "parentJobName" : "A-Jenkins-BuildJob-Parent",
  "phaseName" : "Build",
  "result" : "SUCCESS",
  "url" : "url for the Jenkins job"
},
```

其后,可视化服务解析数据,并按设计显示 CI/CD 的状态和数据。

(2)可视化系统还提供了到 Jira 的超链接。所有团队都可以通过超链接直达 Jira,以报告反馈/问题。

作为最流行的 CI/CD 工具,Jenkins 具有强大的集成功能,可以与各种构建/测试/通知系统互通:

- 在该架构中,BitBucket 用作版本控制系统。BitBucket 是私有 Git 仓库,用于存储来自开发团队的代码,通过其 WebHook,BitBucket 可以在某个分支上发生提交时触发 Jenkins 作业。如文献[15],Jenkins 从 BitBucket 中获取代码更改,并进行本地构建。

- 使用构建输出,Jenkins 与测试系统接口。测试系统的参考是 CppUnit+labview。CppUnit 适用于 UT/IT。它可以在非目标环境中高效运行。对于验收测试和可靠性测试,由于需要真实的目标环境,labview 可用于链接和控制各种设备进行自动测试。

- 当获得测试结果时,Jenkins 可以与邮件系统连接以通知团队,并将失败作为问题记录入轻量级问题跟踪工具 Jira。

### 3 结束语

提出了一种持续集成/持续交付系统的可视化体

系设计和参考架构。该设计将各种信息在不同的视图进行合理组织,将信息呈现或推送给最相关人员,并建议了团队在各种情况下应该采取的行动。该设计连接了开发/测试团队和产品交付团队,有效在团队中共享信息,促进团队协作,高效持续集成以持续交付高质量的产品。

可视化是持续交付的重大进步,而探索永远不会停止。持续交付仍有许多机遇和挑战:CI/CD 系统在展示数据的同时,会记录大量数据。如何利用这些数据以不断改进系统本身和开发/交付过程,这是一个可以进一步探索的主题。当软件准备交付时,可以进一步为最终用户进行持续部署。然而,对于嵌入式设备/系统来说,持续部署将是一个巨大的挑战,当成百上千个设备通过有线或无线连接分布在不同位置时,可以想象将新软件部署/升级到每个设备的难度。该难题也是可以进一步探索的主题。

#### 参考文献:

- [1] RUBIN K S. Scrum 精髓[M]. 北京:清华大学出版社,2014.
- [2] 张敬周,钱乐秋,朱三元. Agile 方法研究综述[J]. 计算机应用与软件,2002,19(6):1-9.
- [3] 陈国栋,罗省贤. Scrum 敏捷软件开发方法实践中的改进和应用[J]. 计算机技术与发展,2011,21(12):97-99.
- [4] HUMBLEJ,FARLEY D. 持续交付[M]. 北京:人民邮电出版社,2011.
- [5] M-DUVAL P,MATYAS S,GLOVER A. 持续集成[M]. 北京:机械工业出版社,2008.
- [6] 张文林. 持续交付及其在大型项目中的应用[J]. 软件导刊,2017,16(10):159-161.
- [7] 牛晓玲,吴 蕾. DevOps 发展现状研究[J]. 电信网技术,2017(10):48-51.
- [8] 陈 龙,叶 蔚,张世琨. Onboard 以数据驱动的敏捷软件开发协同工具[J]. 计算机研究与发展,2016,53(12):2753-2767.
- [9] 兰 洋,温迎福. 持续集成实践[M]. 北京:电子工业出版社,2015.
- [10] 陈 刚,羌铃铃. 软件项目开发中的持续集成研究[J]. 项目管理技术,2011,9(12):103-106.
- [11] 姜 文,刘立康. 基于持续集成的 PC-Lint 静态检查[J]. 计算机技术与发展,2016,26(11):31-36.
- [12] WHITTAKER J A,ARBON J,CAROLLO J. Google 软件测试之道[M]. 北京:人民邮电出版社,2013.
- [13] 肖婵婵. 自动化测试在 DevOps 体系中的应用初探[J]. 移动通信,2017,41(22):77-83.
- [14] 林新党,穆加艳. 基于 Jenkins 的持续集成系统研究[J]. 雷达与对抗,2014,34(1):58-61.
- [15] 姜 文,刘立康. 基于 SVN 的软件配置管理和持续集成[J]. 电子设计工程,2016,24(2):1-5.