

一种适用于独立测试组织的 持续集成测试平台

程莉[†], 韦亚军

(南京国图信息产业有限公司, 江苏 南京 210000)

摘 要:在增量开发模式下, 软件频繁迭代、持续集成是其中的关键, 传统的持续集成无法适用于独立测试团队的组织架构。根据增量迭代模式下独立测试组织的特点, 设计了一种适用于独立测试组织模式的持续集成测试平台, 实现软件从增量更新的获取、应用程序更新部署、测试执行与测试结果反馈的全过程自动化, 并应用于某大型政务服务软件研发项目。结果表明, 该平台能够极大地提升测试效率、提高测试质量, 解决了独立测试组织架构及增量开发模式下高频率交付和软件质量两者之间难以兼顾的问题, 为软件开发项目的流程优化提供了新思路。

关键词:持续集成; 自动化测试; 增量迭代模型; Jenkins

中图分类号: TP311.5

文献标识码: A

A Continuous Integration Testing Platform Suitable for Independent Testing Organization

CHENG Li[†], WEI Yajun

(Nanjing Guotu Information Industry Co., Ltd., Nanjing, Jiangsu 210000, China)

Abstract: In the incremental development mode, frequent iteration, and continuous integration of software are the keys. Traditional continuous integration is not suitable for the organizational structure of independent testing teams. Based on the characteristics of independent testing organization in the incremental iteration mode, continuous integration testing platform suitable for the independent testing organization mode is designed in this paper, which realizes the full process automation of software from obtaining incremental updates, deploying application program, executing tests, and providing feedback on test results. It is applied to a large-scale government service software development project. The results show that the platform can greatly improve testing efficiency and quality, solve the problem of balancing high-frequency delivery and software quality under independent testing organization and incremental development mode, and provide new ideas for process optimization of software development projects.

Key words: continuous integration; automated testing; incremental iterative model; Jenkins

增量迭代模型的开发模式下, 用户需求不断变化带来软件代码频繁修改, 如果不能快速测试并发现问题, 软件质量将难以保证。但如果每次开发提交更新后即开展全面的软件测试, 又会增加测试成

本, 持续的高频率交付与软件质量之间的矛盾日益尖锐。如何平衡两者成为测试团队需要重点解决的问题。持续集成就是解决这一问题的利器, 它要求开发、测试团队紧密合作, 将代码提交、代码检查

、单元测试、环境部署、系统测试全部串联起来自动化执行^[1-3]。但是在独立测试团队的组织架构下,测试和开发分属不同团队,测试团队无法强制开发环节开展代码检查、单元测试,因此传统的持续集成无法适用于这种组织架构。

设计了一种在增量开发模式下适用于独立测试组织架构的持续集成测试平台,并得到较好的实践应用,提高了测试组织的生产效率和生产质量,为大中型软件开发项目减少项目成本、提高软件质量提供了技术参考。

1 持续集成的引入

1.1 增量开发模式

历时长的大型软件项目,用户经常新增或改变需求,多采用增量迭代模型的软件开发模式。增量迭代模型(图 1)采用随着日程时间的进展而交错的线性序列,每一个线性序列被称为一次迭代,完成后产生软件的一个可发布的“增量”,第一个增量往往就是产品的核心模块或功能,它强调每一次迭代均发布一个可操作产品,并与之前的部分进行集成,从而形成更全面的软件功能,开发过程都包括分析、设计、编码、测试和交付^[4-5]。由此可见,该开发模式下每次迭代都需要进行产品的整体测试,测试完成后交付产品,测试与交付成为一项重复性工作,并且,高频率的版本交付导致测试团队需快速、高效地完成测试工作。

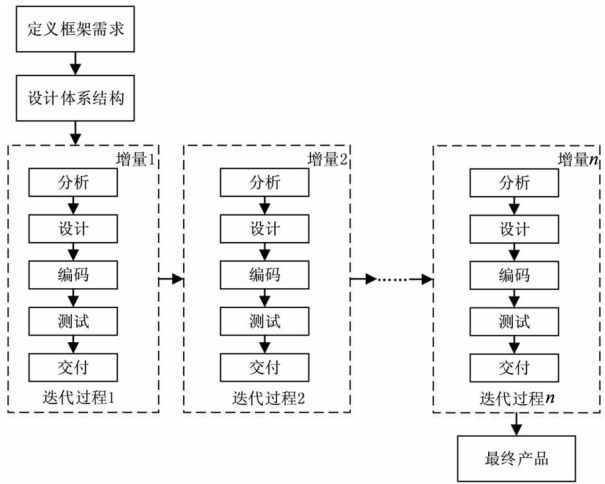


图 1 增量迭代模型

1.2 独立测试组织下的持续集成

持续集成(Continuous Integration, CI)是一种软件开发实践,它要求团队开发人员频繁地集成工作成果,通常每天至少集成一次。每次集成通过全自动化的构建(包括编译、分发、部署和测试等)来

尽快地发现潜在错误,以保证软件质量^[6-10]。通过运用持续集成技术,让整个软件交付过程处于一种全自动的代码编译、发布和测试过程,使软件的缺陷尽可能早期暴露和解决,并时刻处于可工作状态,在任意时间都可以发布和部署^[10]。

增量迭代模型下的软件开发过程频率高、重复性强,占据了大量时间和资源,如果能将此项工作实现全自动化即可解决高频率交付与软件质量之间的矛盾。于是,引入“持续集成”的概念。传统的持续集成要求从开发端开始集成,但在独立测试团队的组织架构下,开发人员和测试人员不属于同一部门,开发人员完成编码后,提交可运行的程序给测试团队,测试人员获取增量版本、更新测试环境、执行测试,测试通过后对外发布更新版本(图 2)。测试团队无法强制开发环节开展代码检查、单元测试,因此只能将测试环节的工作进行持续集成,缩短测试环节用时,实现频繁交付高质量软件的目标。

2 设计与实现

2.1 持续集成测试平台设计

持续集成是将人工频繁开展的工作转化为机器执行,因此,应用于增量开发模式下独立测试组织的持续集成测试是将测试环节的各项工作的转化为机器自动执行,全程无需人工干预^[11-12]。其过程如下:从版本管理库中获取开发不断交付的增量版本,更新至应用服务器进行应用程序自动部署,更新完成后执行自动化测试,全过程监测持续集成的运行情况,将运行异常问题和测试结果反馈给持续集成管理员。持续集成测试平台主要包括 5 个部分,总体架构如图 3 所示^[13-14]。

(1)版本管理库。负责管理开发日常提交给测试团队的增量版本,保障每次集成构建能够从中轻松获取对应的程序更新文件。

(2)应用服务器。应用程序所在服务器,负责接收集成构建的应用程序更新指令,保持应用程序版本与开发提交的增量版本一致。

(3)自动化测试服务器。接受集成构建的执行测试指令,执行自动化测试,生成可读性高的测试报告。

(4)持续集成服务器。按照构建策略触发集成构建任务,检测出版本管理库中最新的更新版本,向应用服务器发送更新指令,向自动化测试服务器发送测试指令,完成集成构建。

(5)监控服务。监测集成构建过程是否存在异常,评估本次集成及测试情况,及时向持续集成管

理员推送相关信息。持续集成管理员根据反馈信息维护持续集成环境。

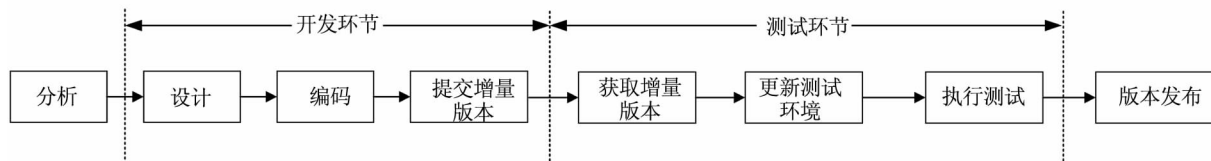


图 2 增量迭代模式下的开发、测试环节流程

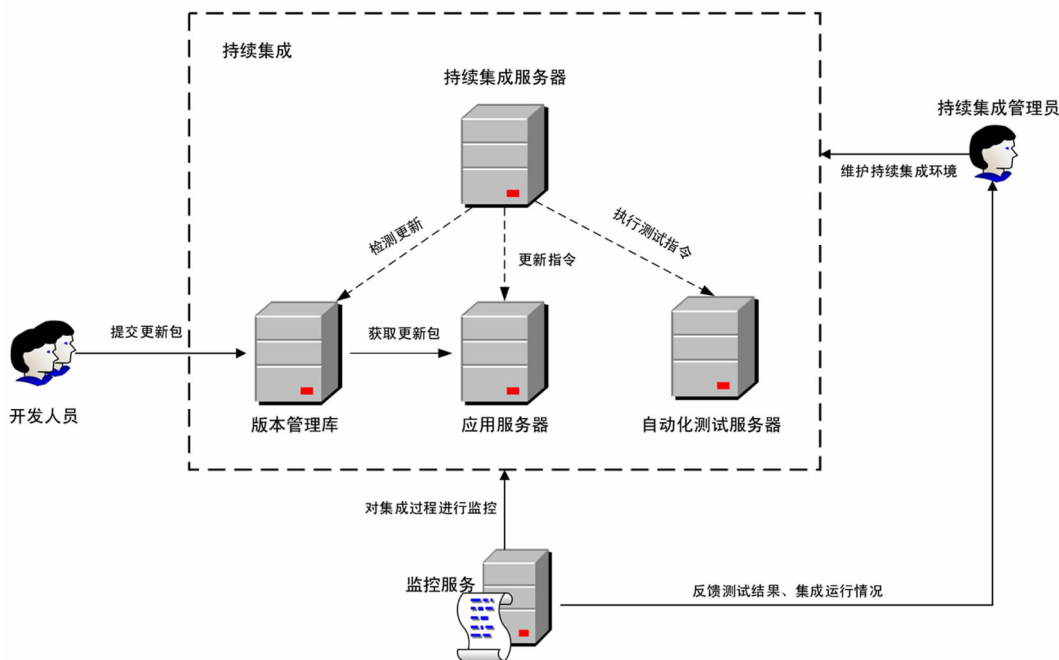


图3 持续集成测试平台总体架构

2.2 持续集成测试实现

采用了 Jenkins、SVN、UFT 等工具软件和 VBScript 脚本,研究实现了增量开发模式下独立测试组织的持续集成测试平台。Jenkins 是一款配置简单且使用方便的持续集成服务器^[15-16];SVN 是一种开源的版本控制系统^[17];UFT(前身为 QTP),是一款以 VBScript 为内嵌语言的自动化测试工具,可用于应用程序和环境的功能及回归测试^[18-19];VBScript 是一种基于 Visual Basic 的脚本语言,可以在 Windows 操作系统上的执行各种任务,具有易学易用的特点^[20]。

在本持续集成测试平台中, Jenkins 作为持续集成服务器, SVN 作为版本管理库, UFT 作为自动化测试服务器, 利用 Windows 环境下的 VB-Script 脚本编写版本检查工具、自动部署工具、测试执行工具、监控服务工具, 分别用于增量版本检查、应用程序更新部署、驱动 UFT 执行自动化测试以及构建结果通知。整个持续集成测试平台通过 Jenkins 配合各类工具, 驱动持续集成运行, 实现增量版本的获取、测试环境更新、测试执行与反馈的全过程自动化, 具体实现过程如下。

(1) 构建准备工作

开发完成代码修改,将编译好的程序包提交至版本管理库,形成一次增量版本。版本管理库可以选择 SVN,也可以采用 Git 等其他版本管理工具,其目录组织需事先约定。增量版本更新频率依据产品规模、项目应用情况可分为固定更新频率(如每天更新,每周一、三、五更新等)和不固定更新频率。根据版本更新频率配置持续集成服务器 Jenkins 的构建策略。若为固定更新频率,则构建策略设置为更新日当天固定时间定时构建;若为不固定更新频率,则构建需更加频繁,设置为工作日工作时间每小时定时构建一次。

(2) 版本检查

到达定时时间,Jenkins 依据构建策略开启自动化构建。版本检查工具检查版本管理库是否存在新版本,若不存在新版本且版本更新为固定更新频率,则视为异常情况,通过监控服务通知测试人员本次无更新并退出构建,若更新发布频率不固定,则直接退出构建;当存在新版本则向应用程序自动部署工具发送更新部署指令。

(3)应用程序自动部署

应用程序自动部署工具接收到更新部署指令后,从版本管理库下载增量版本程序包到应用服务器,并进行应用程序更新和启动。过程中监控服务工具检查应用是否启动成功,若失败则通知持续集成管理员应用启动失败,需人工介入本次测试与发布工作;若成功则向测试执行工具发送测试执行指令,继续执行后续的测试任务。

(4)执行自动化测试

自动化测试服务器需提前安装 UTF 工具并配置好自动化测试环境。当自动化测试服务器中的测试执行工具接收到执行自动化测试指令后,驱动 UTF 开启自动化测试,生成测试报告,测试完成后统计测试结果并通过监控服务工具发送给持续集成管理员。

(5)构建结果通知

集成构建过程中的异常情况 & 测试结果通过监控服务工具传递给持续集成管理员。当监控服务工具接收到消息传递任务时,调用即时通信软件(如企业微信)的接口或邮箱接口与持续集成测试人员建立通信连接,将构建过程中发现的异常问题、测试结果反馈给持续集成管理员。

3 应用案例

根据设计的持续集成测试平台,开展平台应用实践,在某大型的政务服务软件产品研发项目中进行了应用。该产品采用 SVN 作为版本管理库,日常需求多,增量版本更新频率为每天一次。持续集成平台应用前,开发人员每天下班前将增量版本包放置于 SVN 约定的目录中,测试人员在第二天获取增量版本包,手动更新部署测试环境、执行测试,测试完成后发布本次增量更新版本。持续集成测试平台的应用实现了增量版本获取、测试环境更新部署、测试执行全过程的自动化。

3.1 安装和配置

持续集成测试平台搭建主要包括持续集成服务器和自动化测试服务器的相关安装、部署和配置。其中,持续集成服务器上包括节点配置、构建任务和策略配置、相关指令配置等;自动化测试服务器上包括版本检查工具、应用程序自动部署工具、测试执行工具、监控服务工具的安装与配置,以及与应用服务器通信服务的安装配置。

(1)持续集成服务器

在 Jenkins 中新建自动化测试服务器节点,配置远程工作目录、标签等信息,建立持续集成服务

器与自动化测试服务器的通信。创建 Jenkins 持续日常测试任务和自动化测试执行任务(图 4),持续日常测试任务配置每晚 10 点定时构建策略,设置 SVN 版本管理库路径、用户名及密码,设置版本检查工具脚本路径,用于开启集成构建;自动化测试执行任务配置为远程调用策略,并设置测试执行工具脚本路径,用于执行自动化测试。

(2)自动化测试服务器

由于该政务服务软件部署于 Linux 操作系统中,VBScript 脚本无法直接运行,因此根据 Jenkins 测试任务中配置的工具脚本路径,将版本检查工具(表 1 为工具的部分代码)和应用程序自动部署工具放置于自动化测试服务器(该服务器为 Windows 操作系统)对应路径下,用于检测增量版本、更新部署应用程序;安装 OpenSSH,建立应用程序自动部署工具与应用服务器的通信,用于向应用服务器上传增量版本、更新并启动应用程序;部署监控服务工具,在监控服务工具中配置企业微信机器人接口地址和持续集成管理人员信息。

3.2 持续构建实施过程

持续集成测试平台搭建、安装和配置完成后,通过试运行发现并修复问题和缺陷,然后正式投入运行,测试团队配备了专职的持续集成管理员,期间做好相关日常跟踪和记录。

(1)试运行阶段,持续集成测试平台开始尝试在测试生产中应用。持续集成管理员需观察其能否满足日常更新测试的各种情况,发现并修复各种潜在的问题和缺陷。如开发是否按约定方式提交更新包到版本管理库、持续集成构建是否异常、应用程序能否自动更新并正常启动、自动化测试任务能否正常执行等。经过一段时间的试运行,将潜在问题全部解决后即可进入正式运行阶段。

(2)正式运行阶段,持续集成测试平台正式投入测试生产。持续集成管理员根据监控服务工具发送的消息可以及时掌握软件是否存在更新版本、更新后应用程序是否启动成功、自动化测试执行结果等。针对运行过程的异常情况,测试人员需及时分析原因并解决。

(3)持续集成平台运行过程中,软件修改、环境因素等可能导致集成异常,因此需专人跟踪和维护集成平台,做好日常构建相关信息的统计,保障持续集成平台正常运行。持续集成管理员根据监控服务工具发送的消息统计构建是否成功、构建时长、测试用例执行总数和失败数,检查测试报告、分析失败原因,并将这些信息记录于持续集成跟踪表格。



图 4 Jenkins 构建任务配置

表 1 版本检查工具代码(部分)

```
连接版本更新库
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run (" net use \\ " & djDailyUpdateServerAddr& " \ IPC $ ' " & djDailyUpdateServerPwd& " ' /user: ' " & djDailyUpdateServerUser& ".")
wait(5)
Set WshShell = Nothing
读取上一次更新的版本日期
Set objfs=CreateObject("scripting.filesystemobject")
set ts = objfs.OpenTextFile("\\ "&djDailyUpdateServerAddr& "\登记 3.1 日常更新库\登记更新\lastRelease.txt", 1)
djLastRelease = ts.ReadLine
ts.Close
判断文件夹下是否有当前日期的文件夹
If objfs.FolderExists("\\ "&djDailyUpdateServerAddr& "\登记 3.1 日常更新库\登记更新\"&currentDate& ".") Then
    If currentDate>djLastRelease Then
        logFile.write("-----需要更新版本,正在更新中-----")
       重置版本号
        objfs.DeleteFile("\\ "&djDailyUpdateServerAddr& "\登记 3.1 日常更新库\登记更新\lastRelease.txt")
        Set newFile = objfs.CreateTextFile("\\ "&djDailyUpdateServerAddr& "\登记 3.1 日常更新库\登记更新\lastRelease.txt")
        newFile.write(currentDate)
        newFile.close
拷贝此文件夹到本地
If not (objfs.FolderExists(localPath)) Then
    objfs.CreateFolder(localPath)
End If
objfs.GetFolder("\\ "&djDailyUpdateServerAddr& "\登记 3.1 日常更新库\登记更新\"&currentDate& ".").Copy localPath
wait(10)
Else
    logFile.write("-----无需更新版本-----")
End If
Else
    logFile.write("-----无需更新版本-----")
End If
```

3.3 成效分析

在正式运行阶段,持续集成每晚 10 点开启构建,完成从版本更新检查、更新文件获取、环境部署、测试以及结果推送,全过程无需人工干预。持续集成自动化测试相比手工测试在测试效率方面具有明显优势,两者耗时对比见表 2。采用持续集成前该系统的日常更新测试环节全部采用手工进行,其中版本更新检查、更新文件获取、环境部署采用人工方式,若安排 1 人参与工作,则每次需花费 20 min,测

试执行 120 个核心功能用例,花费 600 min,合计 620 min;采用持续集成后,每次完整构建耗时 100 min,1 人参与测试结果分析需花费 20 min,合计 120 min。可以看出在投入相同人力的情况下,持续集成自动化测试的速度相比手动测试快了 5 倍。由于持续集成除测试结果分析需要人工参与外,其他环节完全采用机器执行,可将持续集成时间安排在非工作时间,因此,理论上采用持续集成后,每次更新迭代在人工成本方面可节省 10 人时。

表 2 持续集成自动化测试与手工测试耗时对比

测试流程	版本更新检查	更新文件获取	环境部署	测试执行	测试结果分析
手工测试时间/min	0	10	10	600	
持续集成自动化测试时间/min		100			20

人工测试由于人工的不可控性及资源时间有限性,测试质量难以保证,持续集成全部采用机器执行,测试结果更具可靠性,在提升测试效率的同时还会提高 bug 发现率,提升测试质量。对持续集成测试平台跟踪统计 3 个月,完成构建 150 次,累计节约人力 1500 人时。其间,通过持续集成共发现 26 个 Bug,占日常 Bug 总数的 24.5%,占核心功能日常 Bug 总数的 100%。从应用结果可以看出,将持续集成测试平台引入日常测试生产中可以有效节约测试成本,保证软件产品质量。

4 结 论

为解决增量开发模式下高频率交付和软件质量难以兼容的问题,引入持续集成,设计了一种适用于独立测试组织的持续集成测试平台。实现了软件测试从增量版本获取、测试环境更新部署、测试执行到测试结果反馈的全过程自动化,使测试人员从重复性工作中解放出来,从而将更多的精力投入更重要和更棘手的问题上。同时,随着测试效率的提升,软件产品中存在的问题也被及时发现,避免了问题流转,保证了产品质量。实践证明,该平台能够极大地提升测试效率、提高测试质量,为独立测试组织架构下软件开发项目的流程优化提供了新思路。

参考文献

[1] 张晓帆,刘宁,潘帆. 持续集成系统可视化设计研究[J]. 计算机技术与发展,2020,30(2):58—62.
[2] ZAMPETTI F, VASSALLO C, PANICHELLA S, et al.

An empirical characterization of bad practices in continuous integration[J]. Empirical Software Engineering, 2020, 25(4):1095—1135.
[3] 周莹,欧中红,李俊. 基于 Jenkins 的持续集成自动部署研究[J]. 计算机与数字工程,2016,44(2):267—270.
[4] 韩炜. 可信嵌入式软件开发方法与实践[M]. 北京:航空工业出版社,2017.
[5] 郭佳. 增量式开发模型在 IDC 虚拟主机网上服务系统中的应用[C]//吉林省第四届科学技术学术年会论文集. 2006:188—192.
[6] 心蓝. 构建高质量软件 持续集成与持续交付系统实践[M]. 北京:机械工业出版社,2021.
[7] GHALEB T A, COSTA D A, ZOU Y. An empirical study of the long duration of continuous integration builds[J]. Empirical Software Engineering, 2019,24(4):2102—2139.
[9] 胡鹏,常朝稳,祝现威,等. 面向持续集成的回归测试优化方法[J]. 计算机应用研究,2021,38(12):3709—3714.
[10] 陈迪. 基于 Jenkins 的持续集成系统研究[J]. 电子测试,2020(8):48—51.
[11] 郝宇. 基于 Jenkins 的自动化持续集成与持续交付[C]//辽宁省通信学会 2019 年度学术年会论文集. 2019:98—100.
[12] 桂文军,刘德民,张媛. 基于持续集成的自动化单元测试[J]. 指挥信息系统与技术,2023,14(4):91—94,100.
[13] 李德祥,祝宁. 列车中央控制单元软件持续集成及交付平台设计与实现[J]. 铁道车辆,2023,61(1):27—31.
[14] 雷建胜,苏晓,金明磊. 一种分布式可持续集成自动化测试平台[J]. 计算机与现代化,2020(4):14—18.
[15] 国建胜,张亚楠,刘晶. 基于持续集成的自动化测试框架[J]. 电脑知识与技术,2019,15(6):259—260.
[16] 王红蕾. 持续迭代的软件自动化测试管理平台设计与实现[D]. 青岛:青岛科技大学,2020.
[17] 姜文,刘立康. 基于云环境的持续集成[J]. 计算机技术与发展,2018,28(1):11—16.
[18] 徐小龙,韩丽娟. UFT12.02 在功能测试中的应用[J]. 电子技术,2015,44(9):52—54.
[19] 张伟. 基于 Web 系统的 UFT 自动化测试框架研究与应用[J]. 科技创新与应用,2019(33):159—160.
[20] 王鹏,左涛. 基于 EPS 的 VBScript 二次开发技术在不动产权籍调查项目中的应用[J]. 测绘技术装备,2019,21(3):28—30,27.