软件持续集成平台的设计与实现

###### 摘要

目前各软件公司在开发软件项目的过程中，由于软件本身的规模越来越大，设计越来越复杂，原来传统开发方式中的使用模块划分的方式，并且在模块各自开发之后再进行系统集成的做法，在现在看来越来越行不通了，不仅仅是在各模块在集成过程中会出现各式各样的问题，而且在系统集成之后存在的质量隐患也没有足够的时间和措施来排除，所以经常会影响项目的成功上线。

针对上述的痛点，各公司现在都引入了持续集成的工具，将系统集成日常化，来规避系统集成的风险。然而持续集成工具本身不提供整体性的解决方案，缺乏整体性的过程支持，本文将从平台的角度来阐述如何基于持续集成工具来建设持续集成平台，最终达到系统化持续集成的目的，从整体上把控软件的持续集成过程。

本文阐述的软件持续集成平台，是针对所在公司的系统集成现状设计研发，结合开源工具实现了过程化的持续集成系统平台，包括用户管理、权限管理、组织管理、测试管理、自动化部署、自动化测试等功能，并且在公司内部落地实践，提高了软件集成过程的效率和质量。

本文也根据持续集成的实践结果，总结了持续集成的实施建议和下一步的发展方向，相信能够为大型软件开发项目的持续集成工作带来帮助。

###### 关键词：持续集成 Jenkins 自动构建 管理平台

ABSTRACT

# 绪论

## 引言

现在的软件开发模型的不断发展，不论传统的瀑布式开发模型，还是敏捷迭代开发方式，软件过程都需要经过不同的阶段，包括需求分析、设计、编码、测试、部署、运维等活动，都面临着如何缩短项目的周期时间，提高项目的代码质量的挑战。

软件开发过程中的不同阶段，都对项目的整个生命周期产生了影响，瀑布式开发模型由于自身阶段之间的反馈沟通僵硬不畅，对于整体的结果的产出无法及时掌控，只能通过完成日期和里程碑来跟踪每个阶段的成果，无法真正对软件的质量和产出进行评测，失去及时纠正的机会，而且整个流程单一，不可逆，极度依赖于早起的需求调查。

而且软件的发展，让整个软件内部及外部都变得越来越复杂，对系统集成带来了极大的挑战，如何将系统集成的风险降低，是软件质量工作的最大难题。随着软件开发模型的发展，持续集成作为其中一项重要的革新，给软件开发带来了一种能够降低系统集成风险的机制，同时也带来了另外一个课题，如何把控持续集成整体过程。

## 研究背景及目标

“持续集成”源自于极限编程（XP），并且是XP最初的12种实践之一。ThoughtWorks公司的Martin Fowler先生在实践中发扬了这种技术，并为业界所熟悉。最开始在他参加一个大型项目期间发现了一个在软件项目重要而普遍的问题：软件集成是一个漫长而且无法预测的过程，但是这种问题可以使用一种简单的实践来避免，他给出的建议就是每人每天向一个被管控的代码库集成，其实这就是持续集成的宗旨。最后Martin Fowler先生总结了持续集成的内容：持续集成是一种软件开发实践，在实践中项目成员频繁地进行集成，通常每个成员每天都会做集成工作，如此，每天整个项目将会有多次集成。每次集成后都会通过自动化构建（包括测试）来尽快发现其中的错误。

最开始持续集成并不需要使用特别的工具来部署，只是一种开发工程师的简单实践，但是软件的规模、开发团队的人数对持续集成也提出了要求，这个时候持续集成服务为持续集成工作带来了极大的帮助。

现在基于持续集成在当前公司软件开发流程中具有的重要作用，运用持续集成平台完成软件开发过程中的集成工作已经是流程中必不可少的环节，帮助开发人员定期持续地从代码库更新代码，自动进行编译、部署以及测试，形成安全稳定运行的代码库，为项目组的软件系统质量带来保护，并且能够将所有的质量工作都集中到一个系统中进行记录管理，对软件系统的质量有全面的评测，所以持续集成平台的内容需要达到以下目标：

1、将项目集成的风险最小化，项目团队的最大风险就是项目成员的开发代码集成在一起的时候，会出现合并后的代码不能很好的工作的情况。持续集成利用不断的集成和对集成结果的反馈，促使项目开发人员积极面对问题，修复问题，让bug及早发现并远离系统，消除系统的遏制了隐患。

2、提升项目软件质量，利用持续集成对每天的代码进行单元测试、接口测试以及界面主流程的冒烟回归测试，保证每天的代码质量在一个比较高的基准线以上。

3、增强团队之间的沟通协作，持续集成平台能提供一种规范的方式让团队的每个成员都在关注软件项目的构建结果，并且构建的结果与每个人都有联系，每个人都把项目构建的成功当做最终目标，使得整个团队更加团结，沟通协作更加紧密。

4、增强团队人员的信心，持续集成平台引入的持续集成，能够使项目有一个明确的进度，使项目成员了解当前的项目进度及质量情况，对整个项目的信息有很大的提升。

## 国内外研究现状

目前持续集成的概念已被各个组织和公司广泛接受，并且直接或间接的参与到持续集成的研究与实践中来，在DevOps大行其道的今天，持续集成也成为了其中一个重要的组成部分，也成为了敏捷软件开发的基石。

持续集成的流行除了软件开发流程的改进，也离不开各类持续集成工具的推广使用。第一个正规的持续集成工具是于2001年所推出的CruiseControl，这是一个基于Java开发的开源软件。除了持续构建流程之外，它还提供了邮件通知、Ant以及对各种源代码控制系统的支持，并推出了支持.NET与Ruby的移植版本。CruiseControl具备了一个CI工具的基本功能，为CI过程的推广做出了很大的贡献。

而另外一个持续集成工具Jenkins后来居上，取代CruiseControl成为第一个得到广泛应用的CI工具。Jenkins的出现与发展颇有传奇色彩，它的前身是由一位来自Sun公司的开发者川口浩介（Kohsuke Kawaguchi）于2004年开发的一个基于Java的CI工具Hudson。经过三/四年的发展后，它逐渐超越CruiseControl成为了最流行的CI工具。但自从Oracle收购了Sun之后，希望将Hudson作为收费的商业工具进行开发。以川口为首的开发者社区则决定以Jenkins的名义继续免费版本的开发。有趣的是，Hudson与Jenkins的开发者各自将对方视为自己的分支版本，而将自身视为正统。在2013年后，Jenkins的发展势头已有超越之势，它的每日提交次数远远地超越了Hudson，如今已成为市面上最流行的CI工具。

除了Jenkins之外，其他一些流行的CI工具还包括由JetBrains推出的TeamCity，以及由Atlassian推出的Bamboo等等。

这些CI工具基本都提供了以下功能：

1. 对源代码控制系统的支持，例如Git、Subversion、TFS等等。可以在代码控制的主线发生代码提交时自动触发后续的一系列步骤，例如构建、测试与部署等等。
2. 对依赖管理工具的支持，如Java的Maven、NodeJS的NPM、Ruby的Gem，以及.NET的Nuget等等。
3. 对各种类型测试的支持。早期的CI只支持单元测试，即单个对象或组件的功能验证。随后加入了对集成测试的支持，即对组件之间的通信与交互进行难。尽管如此，这还不足以验证系统确实按照用户期望的方式进行工作。因此现代化的CI工具开始支持功能性测试，将原先的手工测试替代为基于Selenium等工具的自动化测试。

今年国内对持续集成的研究工作日益成熟，各大公司包括阿里、腾讯、百度、京东等互联网公司在每年的软件质量峰会上都会进行持续集成方面内容的分享，开始在实际工作中使用持续集成来提升软件开发的效率和质量。

但是针对业界的整体性的持续集成支持的解决方案还较为缺乏，如何与软件质量管理过程结合没有实际的项目时间指导，本课题希望通过对持续集成平台的整体设计和实施总结的经验和建议，为广大的软件开发及质量人员带来启发和帮助。

## 论文研究工作

持续集成现在作为一种很好的软件工程实践被很多项目团队使用，尤其在需求变化频繁的项目团队中，持续集成的实践有比较好的反馈。在实际使用中，一般会按照自身项目特点和实际需求实现持续集成的自动化。本课题研究内容也主要包括以下几点：

# 持续集成概述

## 持续集成基础理论

大师Martin Fowler先生总结的持续集成的内容：“持续集成是一种软件开发实践，在实践中项目成员频繁地进行集成，通常每个成员每天都会做集成工作，如此，每天整个项目将会有多次集成。每次集成后都会通过自动化构建（包括测试）来尽快发现其中的错误。“

持续集成这个术语源自 XP（极限编程）的一个最佳实践，但是持续集成也完全可以应用在采取非 XP 方法的项目里面。持续集成也不是一个新的概念，在此之前每日构建（daily build）提供同样的含义，微软公司的员工他们的工作其实都开始于每日零点的版本构建。

极限编程是一种开发管理模式，诞生于一种加强开发者与客户之间的沟通需求，用意在于让客户全面参与软件的设计与开发，保证需求变更能够及时响应。极限编程中有四个核心价值是我们在开发中必须注意的：沟通(Communication)、简单(Simplicity)、反馈(Feedback)、勇气(Courage)、此外还扩展了第五个价值观：谦逊(Modesty)。XP用“沟通、简单、反馈、勇气和谦逊来减轻开发压力和包袱；无论是术语命名、专著叙述内容和方式、过程要求，都可以从中感受到轻松愉快和主动奋发的态度和气氛。这是一种帮助理解和更容易激发人的潜力的手段。XP用自己的实践，在一定范围内成功地打破了软件工程“必须重量”才能成功的传统观念。

极限编程把软件开发过程重新定义为聆听、测试、编码、设计的迭代循环过程，确立了测试->编码->重构(设计)的软件开发管理思路

在极限编程中有12个实践，体现了极限编程管理的原则，具有指导型的意义，给软件过程管理带来了价值。持续集成作为12个实践之一，结合采用测试驱动开发、小版本的开发模式，通过不断集成（整体测试）的方式，随时为客户提供沟通的依据，从而可以让客户提出反馈意见。持续集成表示一个开发阶段的完成。

## 持续集成系统实现原理

持续集成系统一般分为下面几个组成部分：

1. 自动构建，包括自动化编译、测试、部署；
2. 代码版本管理；
3. 持续集成服务器；

业界普遍认同的持续集成的原理包括：

1. 需要[版本控制](http://lib.csdn.net/base/git)软件保障团队成员提交的代码不会导致集成失败。常用的版本控制软件有 ClearCase、CVS、Subversion、[Git](http://lib.csdn.net/base/git) 等；
2. 开发人员必须及时向版本控制库中提交代码，也必须经常性地从版本控制库中更新代码到本地；首先在本地机器上保留一份当前已经处于集成状态的代码的拷贝，通过代码管理系统在代码库的主线（master）上拉下（check out）一份工作代码拷贝，这个时候需要将其他开发人员的修改更新到本地并且重新构建。如果他人的修改与自己的修改有冲突，那么在本地编译或者测试阶段将会发生错误，这种情况下，需要负责修改本地代码直到与主线代码保持适当同步为止。
3. 需要有专门的集成服务器来执行集成构建。根据项目的具体实际，集成构建可以被软件的修改来直接触发，也可以定时启动，如每半个小时构建一次；构建过程将编译并链接本地代码，然后跑自动化测试。
4. 必须保证构建的成功。如果构建失败，修复构建过程中的错误是优先级最高的工作。一旦修复，需要手动启动一次构建。只有当构建和测试都没有错误时，该次构建才能算是好的构建。有了本地的成功构建，便可以考虑将修改的代码提交到代码库了。

## 持续集成主流工具

从持续集成的组成部分上来说，持续集成工具可以分成三大类：自动化构建工具、版本管理系统和持续集成服务器。

### 自动化构建工具

自动化构建工具，主要用来编译代码、打包组件、执行测试甚至对代码进行检查和部署等，一般通过命令行及插件的方式提供用户使用。常见的自动化构建工具包括 Ant、NAnt、MSBuild、make、Maven、Rake 等来满足不同开发语言的要求。

### 版本管理系统

版本管理系统，负责代码版本的控制。目前主流的版本管理系统是Git。

Git是目前世界上最先进的分布式版本控制系统之一，与SVN的集中式版本管理系统不同，Git的版本库分布每个人的本地电脑上，不仅有一个中心的服务器控制最新版本代码，而且每个开发者自己还有个本地仓库，所以在开发过程中都是先将代码提交到本地仓库再推送到中心服务器上的，这样的好处就是每个人都依赖于中心服务器来实现交互，但又不会被中心服务器限制，就算中心服务器挂了，也能很容易的找到最新版本的代码，而且我自己的工作依然可以顺利进行，提交到本地仓库，当中心服务器修复之后，再将自己仓库的东西推送到中心服务器。

### 自动化构建工具

Jenkins 是目前使用最主流的一个持续集成工具，通常用于软件开发工作。这种自动化的框架可运行重复的作业。Jenkins 可以管理和监视远程系统上的命令启动，还能执行可通过命令行运行的任何操作。Jenkins 将电子邮件、TestNG 和其他工具与支持插件集成起来。

Jenkins本身使用了master/slave架构，支持主从模式，Master提供web接口让用户来管理job和slave，job可以运行在master本机或者被分配到slave上运行。一个master可以关联多个slave用来为不同的job或相同的job的不同配置来服务，Jenkins 内置了 SSH 客户端实现，用于同远程 sshd 和从代理进行通信。还有多种在主从机器之间进行通信的方式。

Jenkins还提供了插件扩展功能，借助jenkins提供的功能点，可以进行定制化的插件开发，jenkins已经支持超过1000个插件，这个插件的功能包括了配置管理、自动测试、通知、报告等方面，jenkins通过单独的类加载器来加载插件，避免了插件之间的冲突，插件中的所有功能都可以无缝集成到jenkins上。

Jenkins 设计之时就已经支持了让我们通过 REST API 的方式拿到所有的对象的接口，对象的接口可以通过固定的URL进行访问查询，通过提供的接口，使用者可以根据需要来定制jenkins job的创建、设置和触发等功能，实现自身的需求。

## 持续集成的价值

根据持续集成的基本理论和工作原理，总结出持续集成的价值如下：

（1）一天中进行多次的集成，并做了相应的测试，这样有利于检查缺陷，了解软件的健康状况，频繁的集成和及时的客户反馈能够让项目小组人员积极面对软件存在的问题，及时修复，减少了集成过程的风险；

（2）减少重复的过程可以节省时间、费用和工作量。说起来简单，做起来难。这些浪费时间的重复劳动可能在我们的项目活动的任何一个环节发生，包括代码编译、[数据库](http://lib.csdn.net/base/mysql)集成、测试、审查、部署及反馈。通过自动化的持续集成可以将这些重复的动作都变成自动化的，无需太多人工干预，让人们的时间更多的投入到动脑筋的、更高价值的事情上。

（3）持续集成可以在任何时间发布可以部署的软件，保证软件项目的质量。从外界来看，这是持续集成最明显的好处，我们可以对改进软件品质和减少风险说起来滔滔不绝，但对于客户来说，可以部署的软件产品是最实际的资产。利用持续集成，您可以经常对源代码进行一些小改动，并将这些改动和其他的代码进行集成。如果出现问题，项目成员马上就会被通知到，问题会第一时间被修复。不采用持续集成的情况下，这些问题有可能到交付前的集成测试的时候才发现，有可能会导致延迟发布产品，而在急于修复这些缺陷的时候又有可能引入新的缺陷，最终可能导致项目失败。

（4）持续集成让我们能够注意到趋势并进行有效的决策。如果没有真实或最新的数据提供支持，项目就会遇到麻烦，每个人都会提出他最好的猜测。通常，项目成员通过手工收集这些信息，增加了负担，也很耗时。持续集成系统为项目构建状态和品质指标提供了及时的信息，有些持续集成系统可以报告功能完成度和缺陷率。

（5）持续集成可以建立开发团队对开发产品的信心，因为他们清楚的知道每一次构建的结果，他们知道他们对软件的改动造成了哪些影响，结果怎么样。而长期稳定的成功构建将极大的鼓舞团队的士气。

## 本章小结

本章分析了持续集成的基本概念的来历，列出了持续集成的基本组成部分，强调了持续集成在实践中的原则。对持续集成的组成部分进一步说明，分别阐述了各部分的工具支持，最后对持续集成的价值做了详细分析。

# 持续集成平台分析及设计

## 系统目标

该系统的目的是建设一个持续集成平台，能够支持目前公司开发过程中的持续集成工作，包括代码版本管理、自动构建、自动部署、自动测试、测试管理等功能的管理系统，为持续集成工作提供一套完整的解决方案。

该系统的主要用户是公司的开发和测试人员，并且与公司内部的其他研发支撑系统进行打通，成为公司研发工作解决方案中的一环，为软件开发的效率和质量带来价值。

该系统结合持续集成在软件过程管理中的使用方式，运用自动化技术将持续集成实现自动化，并参照到之前在工作中经常遇到因为集成导致的质量问题和工作效率问题，就其根本原因展开研究，将自动化测试、自动扫描、自动部署、自动反馈等技术工具应用到持续集成平台中，以实现以下目标：

1、 整合现有自动化测试工具，建立持续集成平台，该平台以规范和标准的流程把控软件开发过程，统一代码版本入口和编译构建文件，开发、测试及部署在统一标准下运行，保证整体代码质量的统一和可控；

2、 平台采用定制化的方式支持整个持续集成过程的环节，将各阶段的自动化测试工具和框架接入其中，发挥各个的重要作用，提升软件开发效率；

3、 整合质量数据，通过平台的数据质量分析，优化持续集成各环节的技术实现方式，从而推动软件开发过程的质量和效率更加优秀。

## 需求分析

## 系统框架设计

## 系统模块流程分析

## 系统数据库设计

### 说明

### 数据实体关系图

### 数据实体规范

# 持续集成平台方案实现

# 持续集成平台实践

# 总结与展望

###### 参考文献