Broadview
www.broadview.com.cn

测试实践丛书

持续集成实践

兰洋 温迎福 编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

测试实践丛书

持续集成实践

兰洋 温迎福 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书分为三大部分，第一部分介绍一些行业的持续集成解决方案，持续集成的特点及适用范围；第二部分介绍持续集成如何进行一键式部署等；第三部分介绍持续集成与主流 Web 测试工具 Selenium 和 TestNG 的结合应用。

本书的预期读者主要为项目经理、CTO、开发经理、测试经理等。

图书在版编目（CIP）数据

持续集成实践 / 兰洋，温迎福编著. —北京：电子工业出版社，2015.7
（测试实践丛书）
ISBN 978-7-121-26238-8

Ⅰ. ①持… Ⅱ. ①兰… ②温… Ⅲ. ①软件质量－质量管理 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2015)第 120333 号

策划编辑：李　冰
责任编辑：郑柳洁
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：15.25　字数：366 千字
版　　次：2015 年 7 月第 1 版
印　　次：2015 年 7 月第 1 次印刷
印　　数：3000 册　　定价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

我，没有专家的光环，没有博士的高帽，没有国外知名软件公司的背景；我，就是那芸芸众生中的一分子；我，就是那众多读者中的一员。相信很多读者都曾这样想过：出书的人离我们生活的世界很远，很远……但一本好书对读者而言，离自己的心灵，真的很近，很近……

我相信很多读者都碰到过同样的问题：得知某某教授，某某专家，某某成功人士，出了某某书，于是我买，我搜索，我参考，希望能给工作中遇到的困难或瓶颈予指导帮助，但通常的结果是，理想很丰满，现实很骨感。此时我多么希望，能找到一些好的资料、好的案例，真正为有需要的读者提供帮助。

我及我们这个时代的人，以及我们的前辈，很多人都是这样痛苦地走过来的，只能靠自己琢磨钻研，但，我不希望我们的后辈，刚刚进入职场的新人，重新走我们的老路。在这里，我希望可以用我工作上的一些经验、积累和沉淀，帮助他们少走一些弯路。

我想起一件亲身经历的事情，2009 年，我在一家金融公司做关于 OPENSSL 的性能测试，当时解决方案是封装 DLL 用 LR 调用。一年后，我和在另一家公司工作的朋友聊天时，发现他们也在做 OPENSSL 的性能测试，而且在找解决方案，我当时直接把我之前做过的脚本发给了他，他告诉我，参考我的脚本至少使他提前一个月完成任务。也许 2009 年我在做一件事情的时候，在某个地方，有一个或多个和我做一样事情的人，但我们不知道彼此；而我做过的事情，也许未来的某一天，某一个人或某些人会用到，我的及时分享，能帮到他们。

于是，我有了一个出书的种子埋在心里。2011 年，我在一家美国企业工作，那时我刚开始接触持续集成，我们可以快速构建、快速部署环境、快速测试。那时候的我们，尝到了甜头。原来构建可以如此简单，环境搭建可以如此快速，测试可以如此方便，原来懒人可以这样去玩。研究是在快乐和皱眉中前进的，当时除了看官网的帮助文档，就是看官网的帮助文档。当时我多想看到持续集成分享的实例啊，想告诉自己，自己不是一个人在战斗。

现在，我们的测试团队非常成熟地使用持续集成做测试，帮助开发人员快速构建，缩短项目周期。真心希望我们的实践经验能帮助更多的人，给更多的人提供我们的干货。

感谢我的好友陈能技，是他帮我联系出版社，使我从一个出版菜鸟了解了如何出书；感谢我的测试团队，是我们一起努力将持续集成应用到我们的项目团队中，一起解决问题一起前行；感谢电子工业出版社，给我们这样一个好的分享平台；最后，感谢家人和朋友的大力支持。希望本书能真正帮助到在持续集成道路上有困惑的人。

关于本书

持续集成，我相信这个概念大家并不陌生，但关于持续集成的实践，并没有想象中那

么多。

本书分为三大部分，第一部分介绍一些行业的持续集成解决方案，持续集成的特点及适用范围；第二部分介绍持续集成如何进行一键式部署，如何配置，如何结合当前主流的工具进行应用；第三部分介绍持续集成与主流 Web 测试工具 Selenium 和 TestNG 的结合应用。

这本书写给技术性读者。也许你并没有持续集成的经验，也许你有应用持续集成的想法，并想了解它可以给你的团队带来哪些好处。或者，你可能已经开始使用 Hundsun 或者 Jenkins，并且要找一些资料把你的持续集成做得更进一步。相信选择这本书可以给你带来一些有用的答案。

本书的预期读者主要为项目经理、CTO、开发经理、测试经理等，特别是做快速迭代的团队，希望本书能给他们带来帮助。

目录

第 1 部分　持续集成：介绍篇

第 2 部分 持续集成：玩转 Jenkins

第 1 部分　持续集成：介绍篇

一个程序员，可以轻松地进行编译、测试和发布，而他的痛苦之处在于他庞大的编码工作量；当多个程序员协同工作时，开发 Leader 会将程序分解，并分到每个程序员手中。每个程序员的编码工作量减少了，但所有的代码都开发完成之后需要再集成到一起进行测试，工作量同样不少。同时，由于很多 BUG 在项目的早期就存在，到最后集成时才发现问题导致开发者需要在集成阶段花费大量的时间寻找 BUG 的根源，加上软件的复杂性，问题的根源很难定位，甚至出现不得不调整底层架构的情况，在这个阶段的“除虫会议”（BUG Meeting）特别多，会议的内容基本都是讨论 BUG 是怎么产生的，最后往往发展成为不同模块的负责人互相推诿责任。就在这样的背景下，极限编程、持续集成相继诞生。

为什么我们要提到极限编程。“持续集成（Continue Integration）”一词其实来源于极限编程（Extreme Programming），作为它的 12 个实践之一出现。Matthew Foemmel 将极限编程中的指导思想转换为具体的行动，从而让我们看到了项目从少而繁杂的集成进步到如今我们轻松愉快地集成。

在本书第 1 部分里，我们提供一个简单的读书路径，我们相信读者在读书前都会带着自己的问题。

持续集成到底可以解决哪些问题？

到底什么是持续集成？

持续集成的核心价值是什么？

持续集成具体怎么使用？

如何玩转持续集成工具？

持续集成和 Selenium 在一起，碰撞出了什么火花？

接下来，我们就来看看持续集成都解决了哪些问题。

第1章

持续集成解决了什么问题

1.1 提高软件质量

如何提高软件的质量已经不再是一个纯粹的技术问题，而是一个工程问题。

自计算机诞生以来，相应的软件开发就存在。由于早期的计算机运行性能较低，软件的可编程范围也较狭窄，因此质量问题就没有那么突出。20 世纪 50 年代后期到 20 世纪 60 年代，高级语言相继诞生并得到了广泛的应用，随之而来的是软件规模越来越庞大，越来越复杂。伴随着软件得到越来越广泛的应用，软件的质量问题变得越来越突出。根据美国国家宇航局 NASA 的统计，在 20 世纪 80 年代初，软件引起的故障与硬件引起的故障比率约为 1.1:1.0，到了 20 世纪 80 年代末，这一比率已达到 2.5:1.0。20 世纪 90 年代及 2000 年以后，质量已经被所有的公司重视。软件 BUG 产生的问题会让公司损失很多客户，互联网公司就是其中的典范，它们的目标是将用户体验做到极致，让软件覆盖越来越多的用户使用场景，让用户用着爽。如果 BUG 多，那么何谈优秀的用户体验呢？恐怕连极致体验的边都触碰不到。

因此，如何提高软件的质量成为软件工程研究的一个重点。自从软件危机产生以来，出现了很多提高产品质量的理论和方法，有的从技术角度出发，例如面向对象技术的产生和推广，第四代语言的诞生等；有的从自动化工具入手，例如 CASE 工具、过程控制软件、自动化管理平台等；有的从过程模型角度出发，例如迭代模型、螺旋模型、RUP、IPD、净室软件工程等；也有从管理角度出发的，例如团队管理、绩效管理、PSP、TSP 等；也有从测试角度出发的，例如加强全流程的测试等；一些相应的规范和标准也孕育而生，例如 ISO9000 系列、CMM、QMS 等。

每一种解决质量问题的方法都不是独一无二的，软件质量的提高应该是一个综合的因

素，需要从每个方面进行改进，同时还需要兼顾成本和进度。作为软件产品的销售人员、市场人员或维护人员，经常会受到客户的指责或抱怨，客户说：你们产品的质量太差，不稳定等。那么什么是质量呢？我们该如何衡量质量呢？质量具有三个维度：符合目标、符合需求，以及符合实际需求。目标是客户定义的，符合目标即判断我们是不是在做需要做的事情。实际需求包括用户明确说明的和隐含的需求。符合用户习惯，这其中包含了我们要引导用户如何操作或者去适应用户本身的操作习惯。

ISO 关于质量的定义如下：“一个实体（产品或服务）的所有特性，基于这些特性可以满足明显的或隐含的需要。”

注意，在这个定义中包含明显的需求和隐含的需求。我们往往会忽略隐含的需求。因此，一方面，在控制一个产品的质量的过程中必须关注这些隐含的需求，并给予应有的验证。另一方面，因为我们的产品是为客户提供服务的，因此凡是不满足客户需求的，我们都认为是失败的（failure）。所以我们的产品必须始终围绕着客户的需求进行开发和验证。

其实在一个软件的需求收集过程中需要关注客户和用户，而我们经常会忽略客户与用户之间的区别。那么谁是客户？谁是用户呢？简单来说，客户是真正能够决定是否购买你的软件的人，而用户是实际使用软件的人。了解了这个区别，你在分析需求的重要性时可以参考。同时，在产品质量验证时也可以做出不同的权衡。另一方面，我们在考虑用户需求时，往往只考虑了实际使用软件的人员，而忽略了其他人员对软件的要求或对软件造成的潜在竞争，这包括维护人员的要求，系统管理人员的要求，软件上下游人员的要求，先前版本的情况，市场上竞争对手的软件情况等。

提到质量的时候，我们经常会遇到下列矛盾，在这些矛盾中隐含着对质量的承诺：质量需要一个承诺，尤其是高层管理者的承诺。但为了得到质量，高层管理者必须和其雇用的员工进行紧密合作；许多人相信没有 BUG 的产品和服务是不可能的，但是将缺陷数控制在一定级别是正常并可接受的；质量经常和成本紧密联系在一起，一个高质量的产品同时也意味着高投入，这是设计的质量和一致性质量的一个矛盾。一个高质量的产品的需求规格说明书要足够详细，以便产品可以根据这些规格说明书进行定量分析。

目前流行的流程方法有很多种，不同的过程模型适合于不同类型的项目。瀑布模型是应用最广泛的一种模型，也是最容易理解和掌握的模型，然而它的缺陷也是显而易见的。遗漏的需求或者不断变更的需求会使该模型无所适从。然而，对于那些容易理解但很复杂的项目，采用瀑布模型会是比较适合的，因为你可以按部就班地处理复杂的问题。在质量要求高于成本和进度要求时，该模型的表现尤其突出。

螺旋模型也是一个经典模型，它关注于发现和降低项目的风险。项目从小规模开始，然后探测风险，制定风险控制计划，接着确定项目是否继续，若继续则进行下一个螺旋的反复。该模型的最大优点就是随着成本的增加，风险程度随之降低。然而螺旋模型的缺点是它比较复杂，且需要管理人员有责任心、专注，以及有管理方面的经验。RUP 工作流程示意图和 IPD 工作流程示意图如图 1-1 和图 1-2 所示。

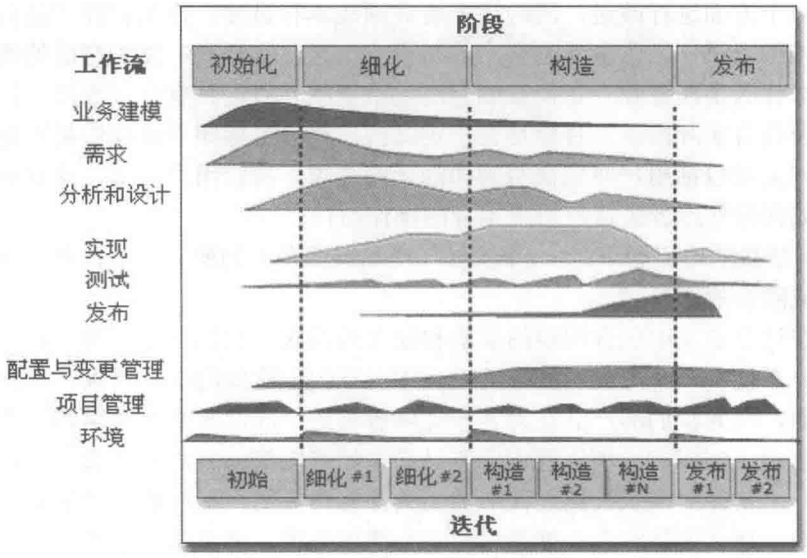

图 1-1 RUP 工作流程示意图

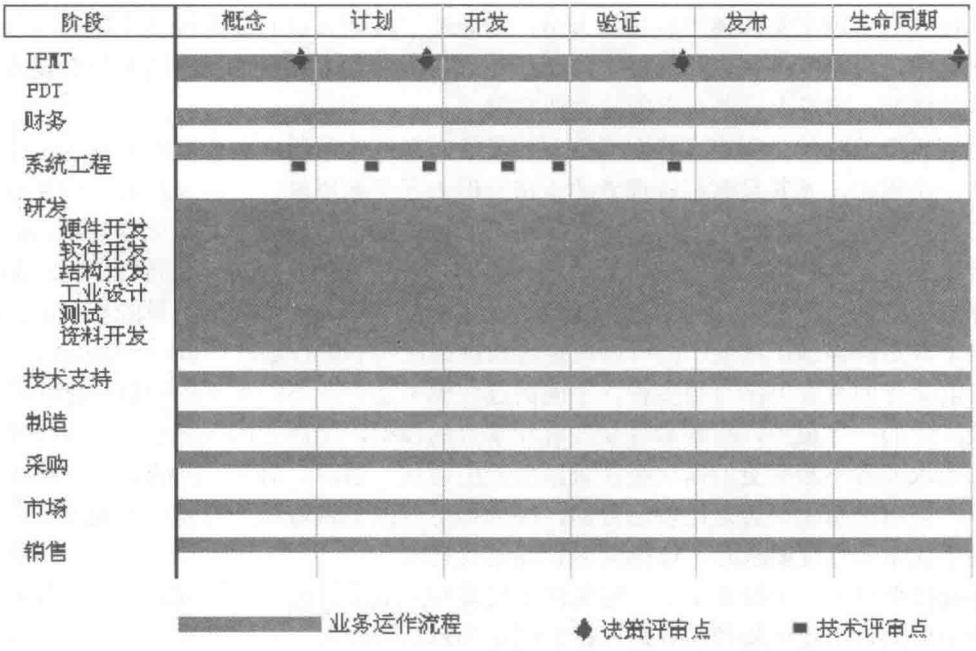

图 1-2 IPD 工作流程示意图

从以上两个模型中我们不难发现，所有的流程图例中都有开发一环，开发都有构建这一环，而在构建及发布环节上，持续集成帮你节约的时间，以及你根据你的实际项目需要而设定的构建频率，可以很好地帮助你在不断解决自己缺陷的同时，不影响团队其他人的代码质量。

BUG 的增加与时间并不是线性增长的关系，而是和时间的平方成正比，两次集成间隔

的时间越长，BUG 增加的数量就越多（超过你的预期），解决 BUG 的工作量也越大，而你越觉得工作量大，你就越想推迟到以后去集成。你甚至企图到最后一次性解决问题，结果 BUG 产生得更多，导致下一次集成的工作量更大，你越觉得集成痛苦，就越将集成的时间推后，最后形成恶性循环。

高效的持续集成模式，即在分批提交代码的时候，就已经进行了测试，测试一直持续到程序开发完毕，一直延续到功能开发完，测试完，再开发完，再测试完，避免了在代码累积到一定量进行集成时，造成的代码质量低下的风险。

如果出现问题，项目成员马上就会被通知，问题会在第一时间被修复。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时才被发现，这有可能导致延迟发布产品，而在急于修复这些缺陷时又有可能引入新的缺陷，最终可能导致项目失败。有人不禁会问，没有持续集成，我们的项目不是一样交付么？这么想当然没错。如果持续集成可以让你交付的代码质量更高，你，何乐而不为呢？

1.2　节约时间，缩短项目发布周期

快速开发和上市一个新产品，并快速取得预期的投资回报是每个企业孜孜以求的目标。但是事与愿违，很多新产品项目因盲目地追求开发进度而中途夭折，有些项目即使按期上市了也并未取得预期的投资回报。那么，如何在保证项目质量的前提下，尽可能地加快项目进度呢？有效的方法有很多，制作如图 1-3 所示的项目计划示意图就是其中一种。

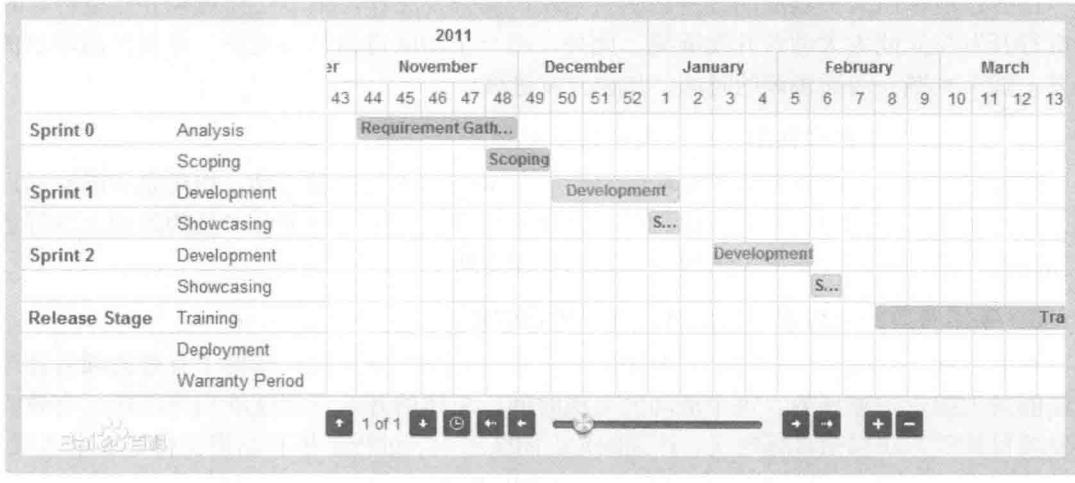

图 1-3　项目计划示意图

1．深入了解顾客需求，减少开发过程中的需求变更与返工

顾客需求是新产品开发的输入，如果输入不正确、不完整，就必然导致开发过程中大量的变更，这对开发进度影响极大。根据国际最佳实践，改进型的项目在正式开发前应该有 80%的需求被确定，突破性的新产品在开发前也至少应有 50%的需求被确定。进行顾客需求调研时最好请市场人员与技术人员参与，不同背景的工作人员能从客户方了解到更为完整的信息。

市场人员与技术人员一起进行产品定义，可以减少信息传递过程的失真，减少开发过程工作的返工。

新产品项目的顾客需求输入不应该由市场人员“提供”给技术人员，而应该请市场人员与技术人员坐在一起进行多次沟通与交流。比较复杂的产品可以采用 QFD 方法进行产品定义。清晰的产品定义能大幅减少开发过程的返工工作。

采用跨职能团队合作方式进行项目立项分析工作，以节省开发、测试、制造和上市等的周期时间。

影响新产品上市时间的不止是开发过程，测试、制造和上市准备等过程中如果出现大量反复，也会严重影响上市时机。例如，一些企业在新产品样机出来后发现开模很难、关键部件难以采购、生产工艺达不到要求等，这将大大延长新产品的上市周期，甚至导致新产品项目失败。从立项分析阶段开始就采用跨职能团队合作模式一起协同工作，能有效缩短样机出来后到上市的时间。

2．对新产品项目进行开发优先顺序排列和合理资源分配，确保重要的项目得到优先开发

在资源有限的情况下，同时开发过多项目的结果是所有项目都会延期。国际最佳实践研究表明，一个开发工程师同时进行两个项目的开发时效率最高，同时开发 3 个项目时效率开始下降，同时开发 4 个项目时效率将显著下降。所以，一个工程师最好不要同时安排两个以上的开发项目，以保证重点项目的开发进度。

3．采取跨职能团队组织模式进行新产品开发

由各职能部门人员组成的开发团队负责新产品开发工作，很多工作可以并行进行，相对串行开发模式能大大缩短开发周期。此外，由一个团队自始至终负责一个新产品项目的开发，能大大减少沟通协调的时间，加快开发进度。

4．建立技术平台和共用模块，缩短开发周期

有研究发现，一个新产品开发项目中平均有 40%以上的重复劳动。如果企业能够通过建立技术平台，使一些技术模块化，使一些模块标准化，在一个平台产品的基础上进行更多的同类产品开发，则能大大缩短新项目的开发周期。

5．采用有效的项目管理方法进行开发项目的管理

有效的项目管理方法是每个开发团队成员都应该熟练掌握的，掌握了有效的项目管理方法能大大缩短每项任务、每个活动的完成时间，总体的开发周期就能相应缩短。有效的开发项目管理方法包括目标定义、计划制定、团队组织、过程监控和结果交付等 5 个步骤。

采用适当的 IT 工具提升开发效率。比如应用 PDM 工具能有效管理各种产品信息，减少重复劳动，降低信息沟通成本，加快新产品的开发速度。其他如 CAD/CAM/CAE 等软件的应用也能大大加速新产品的设计和开发。

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、数据库集成、测试、审查、部署及反馈。通过 Jenkins 工具进行的持续集成可以将这些重复的动作变成自动化的，无须太多人工干预，让人们将更多时间投入到需要深入钻研的事情上。从而缩短项目总体耗时，将项目发布时间提前。

1.3　便捷部署

显而易见，部署包需要包含应用程序的所有组件，不仅仅是你自己的二进制包——EARs、WARs 之类。通常，这些包由集成构建产生，还应包含静态内容、配置文件、共享库、防火墙配置，等等。特别还应包括应用服务器的参数和设置，比如消息队列、连接工厂、数据源或类环境变量。实际上，软件包应包含软件生命周期中的所有内容，也就是那些需要一起被部署、升级和取消部署的内容，图 1-4 所示为部署包示意图。

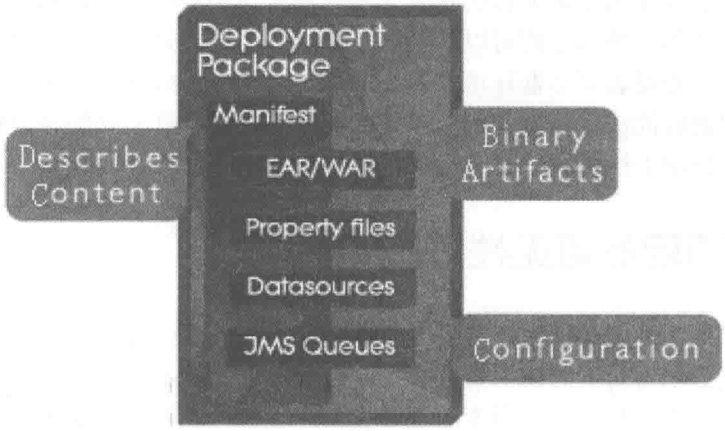

图 1-4　部署包示意图

确保软件包是“完备的可部署单元”对于一次可靠的部署来说是至关重要的，特别是在大规模环境中。指望目标中间件栈（Middleware Stack）已经用“正确值”设置好是导致部署失败的一个常见原因，这通常导致耗费很多时间找出不同环境中的细微差异，这往往是令人沮丧的过程。

部署包中只包含配置信息，比如共享“服务”，例如，为多个应用所共享的消息队列或数据源，这种情形不常见。这些配置显然应该按照有版本管理的、可部署的对象来处理，这意味着这些配置文件按照和“普通”应用程序一样严格的流程来发布。实际上，平稳、可靠的部署共享服务通常更为关键。

除了确保软件包最完整地描述了包含什么使特定版本的应用程序或配置正确工作外，软件包还需要指出它们能够运行所需的其他东西。换句话说，部署软件包需要明确界定他们的先决条件和依赖，以便在匹配的环境上部署。

准确地说，哪个部门负责部署和组织结构有很大关系。多数情况下，开发团队通常能使用持续构建工具使某些发布流程变为自动化，但发布过程还可能涉及其他团队。举例来说，让 DBA 在发布前确认 SQL 脚本，或要中间件竞争中心（Center of Competence）检查中间件配置的情况并不少见。

顺畅和可靠的部署流程从简单的第一步开始：以一种自动的、可检查校验的方法整合并发布一个结构化的、完整的部署包，该部署包定义了特定版本的应用程序中所有的组件、配置和依赖关系。这能显著减少因不合法或缺少定制值、组件或必需的服务而产生的错误。