第 1 部分: 构建管理入门

概述

软件构建流程是软件生命周期一个非常关键的环节，直接决定了开发工作是如何最终交付给最终用户的，但构建流程长期以来被认为最难以结构化和规范化管理，同时围绕软件构建和发布管理的最佳实践经验也不足。但随着软件复杂度的不断提高，软件交付周期的不断缩短和交付频率的提高，软件构建流程已经逐步得到重视。

“软件构建管理在日益影响成功的软件部署，业务以及 IT 的生产率，软件构建管理正成为 IT 企业关注的一个重点。” - IDC

“如果不能提供一致、精确以及可重复的软件构建，会形成软件开发中的严重瓶颈从而造成软件开发团队不能很好地在不增加额外资源的前提下管理项目的复杂性。” - Hurwitz

IBM Rational BuildForge 是一个用于构建和发布自动化的系统，可以集中、自动化并且加速不同软件开发组织的软件开发过程。在与 IBM Rational 软件开发平台的其他软件一起使用时可以帮助企业打造用于端到端的软件交付流水线的开发环境。同时也可以更好地帮助软件开发组织遵规。对于从事多种软件应用开发的组织 BuildForge 可以提供一个灵活集成的平台，从而充分利用用户在已有工具的投资来提高整个团队的交付效率、实现交付自动化和可跟踪性。

对于异地分布的团队 BuildForge 可以帮助不同地域的团队更高效地进行协同。BuildForge 提供的跨多种应用和硬件平台的集中访问方式可以实现全球性的报告、跟踪和硬件资源使用优化。通过 Rational BuildForge，开发团队从大量重复性的任务中解脱出来，以更聪明的方式进行工作，将精力集中在构建及发布过程的度量和改进上来。

关于本系列

本系列描述了利用 IBM Rational BuildForge 进行构建和发布过程管理及自动化的方法，讨论了如何通过整合项目组、流程以及系统来改进软件开发效率，从而提高整个开发团队的效率，改进产品质量，更好地遵规。本系列首先对编译和构建的差异给出了说明，然后对文中常用的名词或术语进行了描述，接下来在“当前构建及发布过程面临的挑战”对当前应用生命周期管理中，特别是构建及发布过程面临的挑战。为了更好地帮助读者理解 IBM Rational BuildForge 的实现原理及功能，特别在介绍具体产品功能前加入了有关构建流程的一些基础知识及最佳实践经验。在“IBM Rational BuildForge 简介”部分对 BuildForge 的基本功能进行了描述，之后是基于 BuildForge 的集成构建平台解决方案介绍，以及给软件开发、质量改进以及遵规带来的好处，最后是一些用户的案例研究。

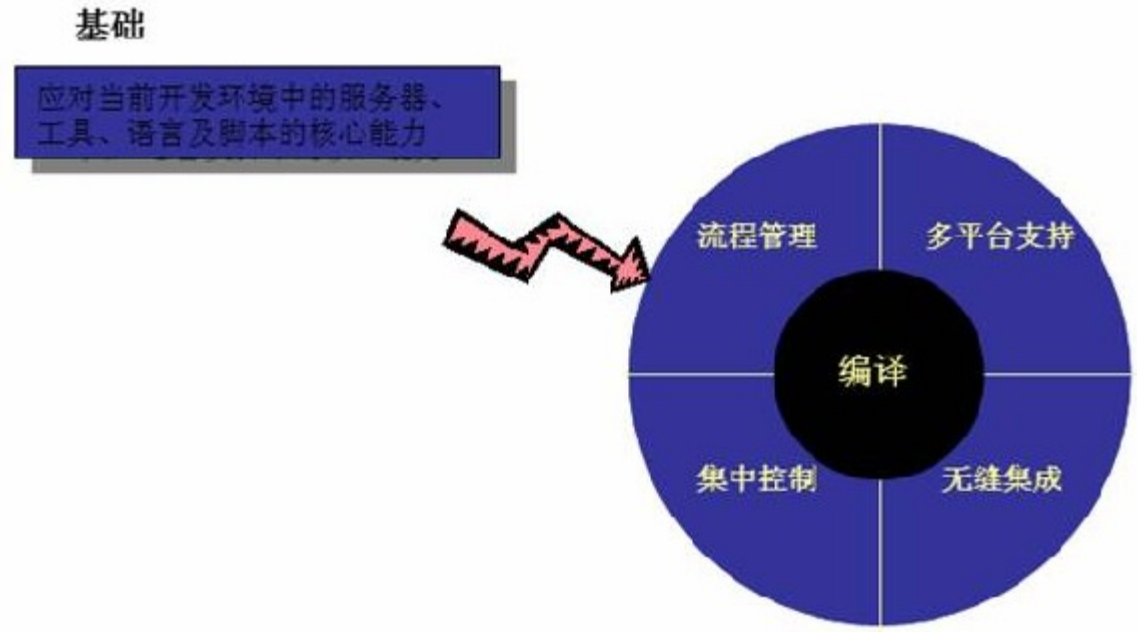
了解 本系列其他部分。

回页首

构建远远不是编译

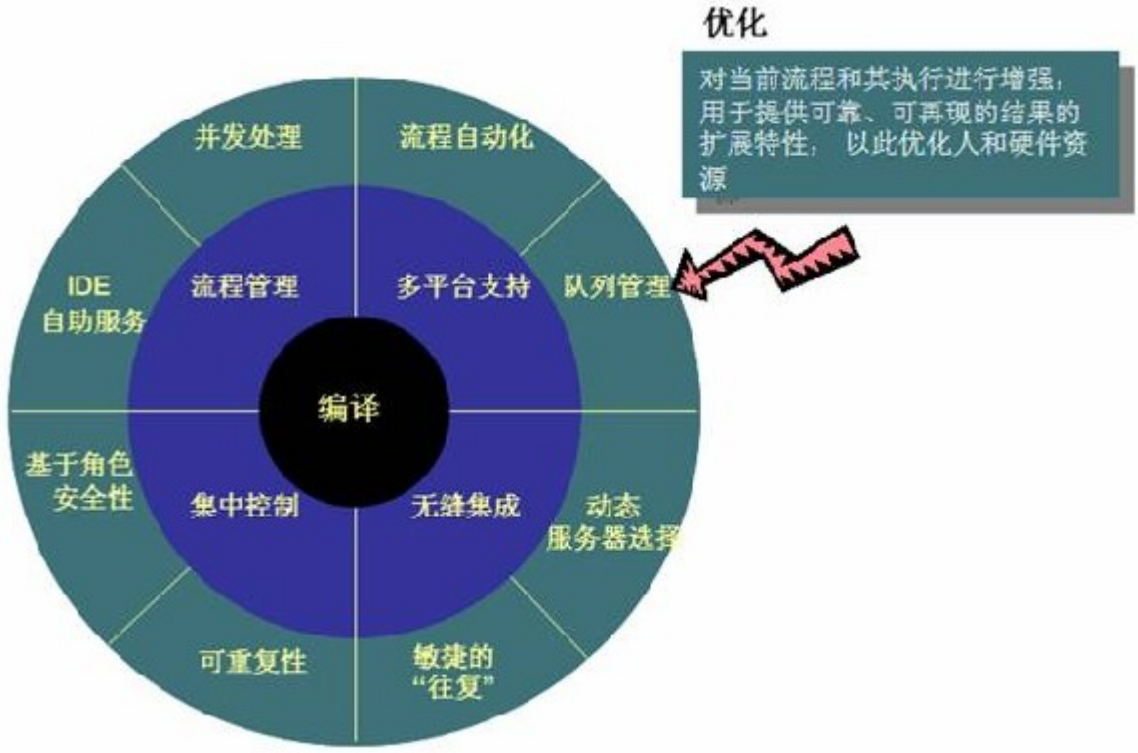
许多人都将构建（build）与编译混在一起，认为两者没有大的区别。实际上构建的内涵和外延远远大于编译。构建是指从源码到最终投产或上线的整个生命周期，而编译只是从源码到中间代码或可执行代码的过程。下面几张图解释了两者之间的差别。

图 1: 两者之间的差别



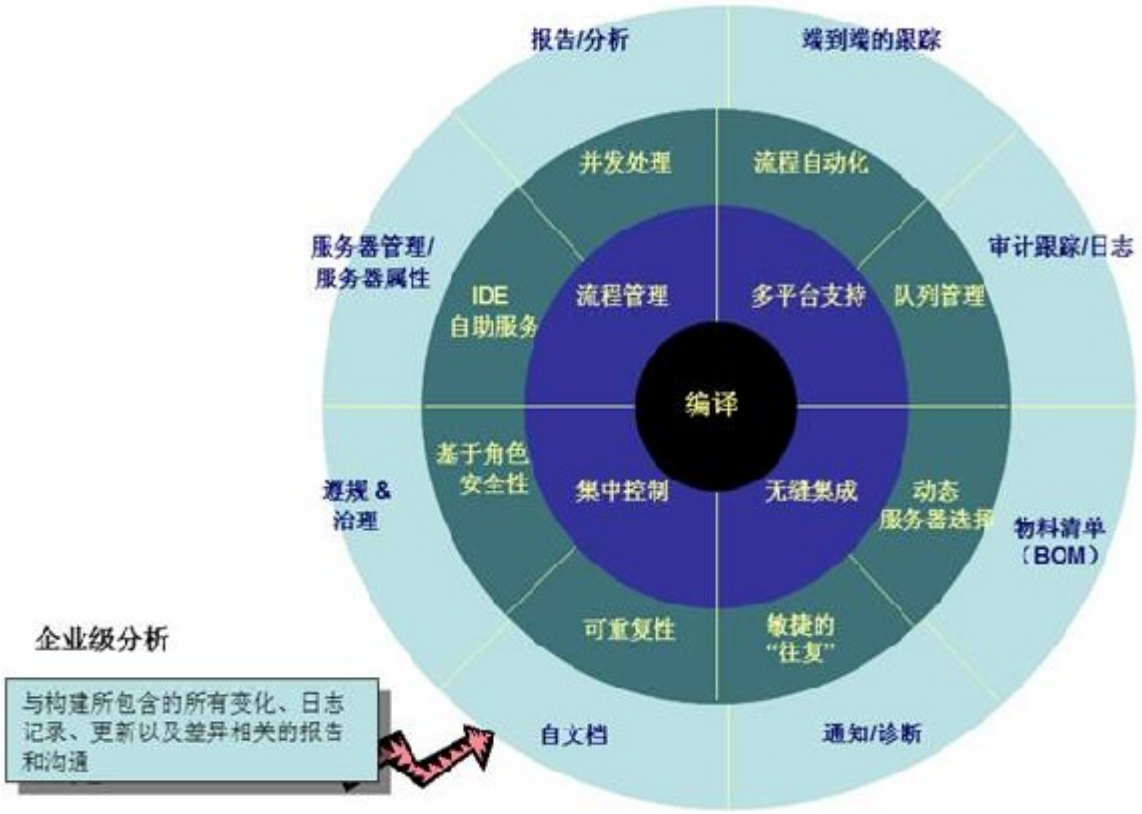
如上图所示，编译只是中间的黑色部分，从技术上看是从源码向机器码翻译的过程。而构建特别是基础构建能力应该包括：流程管理、多平台支持、构建流程/执行/结果的集中统一控制、以及与其他工具的无缝集成，这些都是处理当前开发环境中的硬件资源、工具、语言及脚本不可或缺的核心构建能力。

图 2: 在基础构建能力上可以进一步加强构建的可靠性和可再现性



在基础构建能力上可以进一步加强构建的可靠性和可再现性，从而进行人和硬件资源的优化，因此可以在核心构建能力上进一步进行扩展。如上图所示，这些扩展构建特性包括：基于角色的安全性访问控制、可重复性、对敏捷开发的支持、动态服务器选择以增强系统的可靠性和硬件利用率、构建流程自动化、构建队列管理、构建并行化处理、嵌入集成开发环境的自助构建服务等。

图 3: 构建管理的开发与验证的典型过程



进而，为了将前面提到的基础构建能力和扩展构建能力推广到整个组织，并且更好地应对审计和遵规要求，构建还应包括面向整个企业的报告和沟通能力。这些特性如上图所示可以包括：报告及分析能力、端到端的全生命周期跟踪、审计跟踪和日志记录、包含所有流程细节步骤及运行结果详细信息的物料清单（BOM）、跨团队的通知以及问题诊断、在构建执行过程中自然而然形成相关文档（自文档）、服务器静态（如操作系统类型、版本及补丁等）及动态属性（CPU 负载、空闲内存等）管理、以及遵规和治理的支持。

有关构建的名词及解释

构建（build）

广义的构建指将不同部分的软件源代码合成为最终可交付软件的整个过程。构建是软件开发过程中一个重要部分，通过重复不断的源代码收集、编译以及测试，从而确保最终软件产品的交付。狭义的构建指程序发布前的版本，通常每个构建会有一个唯一的构建号。

私有构建（private build）

开发人员在自己工作空间内进行的构建，通常用于对自己修改的代码进行校验。

集成构建（integration build）

由核心职能部门或所指定的集成人员将不同开发人员的源代码收集在集成工作空间后进行的构建。集成构建由负责集成构建的人员（通常是开发组长或来自构建部门的人员）来进行，也可以通过调度程序自动定时进行。

发布构建（release build）

由核心职能部门，通常是来自构建部门的人员执行的直接交付给内部或外部最终用户的构建。发布构建通常在隔离和受控的环境下进行。

每日构建（daily build）

基于所交付的变化而每天（通常在夜间）进行的集成构建。每日构建通常会伴随着冒烟测试或其他一些测试行为。

持续集成构建（continuous integration build）

一种广泛用于敏捷开发方法的构建方式，即不断对版本库进行监控，一旦有变化进入版本控制库就自动开始集成构建，持续集成构建通常构建时间较短（小于 10 分钟）并且伴随完整的单元测试。

构建管理流程（build management process）

有关构建的整个处理过程和管理要素。包括构建配置类型（私有构建、集成构建及发布构建）、端到端的构建流程域（版本控制、代码静态分析、编译、单元测试、数据处理、打包、功能集成测试、代码覆盖率及部署）、构建基础设施（版本控制工具、构建工具以及硬件）、构建的定义执行和报告、构建相关角色和职责、构建服务器基础设施（开发人员桌面机、构建服务器以及版本控制服务器）、构建频率等。

本文要介绍的 IBM Rational 构建管理产品 BuildForge 覆盖了上面所涉及的方方面面的构建特性。

回页首

当前构建及发布过程面临的挑战

IBM Rational BuildForge 的一大优势是将不同的过程、人员以及工具集成到一个一致而高效的软件交付系统中。它主要解决了三个关键的开发挑战：应用生命周期管理，工具标准化以及遵规。

应用生命周期管理面临的挑战

一个软件产品或交付项目从编码阶段开始一直到投产或上线会涉及一个复杂的有关人员、流程以及技术的网络，为了提高软件交付的效率就必须对它们进行整合。通常开发团队会碰到下面一些问题：

- 不同关键阶段（开发、配置管理、QA、发布和客户支持）的团队由于组织不同相互分离，使用不同的工具集，甚至分布在不同的地域。
- 每个团队有自己的流程，许多流程是手工流程，很少进行记录和文档化，结果导致非常关键的构建和发布知识驻留在少数关键人员的头脑中。
- 缺陷跟踪以及源代码控制系统经常相互分离，互不相接，形成了不同的信息孤岛，从而难以集成使用这些信息并且可能消耗大量的时间和精力。结果开发组织不能全面了解到底提交给用户的东西是什么，直到产品在用户使用过程中出现了严重问题才被动地去进行问题诊断。

随着当前对软件质量和软件交付频率的不断提高，开发团队需要一个坚实的基础来提高软件开发过程中的可重复性、可靠性和可跟踪性。因此不少开发团队都在寻求建立这样一个基础设施，目前他们面临着两个选择：

- 从单一厂商购买一个端到端的应用生命周期解决方案
- 实现一个集成框架将现有工具和流程连接起来

不管开发组织采用上面的哪一条改进线路，IBM Rational BuildForge 都可以帮助团队改进开发效率。

标准化还是非标准化？

对于想采用一整套内部已经集成好的工具集来进行软件开发管理的用户，IBM Rational 软件开发平台（SDP）是一个非常好的选择。集成了 IBM Rational ClearCase、IBM Rational ClearQuest、IBM Rational RequisitePro、IBM Rational BuildForge 以及 IBM Tivoli Provisioning Manager 的工具集提供了全面的从需求到生产的自动化控制。通过从单一可信任的厂商购买一整套解决方案公司可以获得紧密的产品集成，同时客户支持服务也比较简化。

其他的开发团队也发现采用一整套标准化工具对于他们自身环境和情况不允许或不现实。由于贯穿开发生命周期的沟通和集成非常关键，但是强行要求不同团队采用一致性的做法从而保证同其他团队的沟通和集成往往并不容易，还可能引起内部纠纷，从而使得这种集成风险较大。即使所有的团队都同意进行标准化，一些事件的发生也可能会引入不一致性的东西，例如收购了使用不同工具集的另一家企业或者将某一部分外包给使用不同工具集的第三方厂商。最终，一些开发团队需要一个灵活的架构来支持不同的工具集来适应组织未来的开发需要。IBM Rational BuildForge 通过一个灵活的框架提供了对任何工具集成、自动化和报告的能力。

IBM Rational BuildForge 灵活的框架可以与组织内现有的工具一起工作，并且随着时间的推移，它也可以和新购买的工具一起工作。这样通过 BuildForge 可以建立一个集成的开发生态系统，从而克服不同工具集成工作的难题。BuildForge 提供的一致性接口可以跨不同的操作系统和硬件平台，用户从同一个接口启动一个 Microsoft Windows 程序或者 Unix 命令，用户也可以启动一个处理过程同时在不同平台，如 Microsoft Windows、Linux 和 AIX 上进行相应任务处理。这给了开发团队极大的自由度来确定适合自身环境的解决方案。

对审计和遵规管理的支持

越来越多的组织现在正面临如何遵从政府或业界规定、标准的问题。为了能通过审计，开发团队必须要展现出他们具备可重复的过程，精细的控制，并且他们必须一致性地记录系统为什么发生变化，谁具体实施了这一变化。没有一个自动化的应用生命周期管理系统，团队可能会花大量的时间从不同应用收集数据，然后进行加工来满足审计人员的要求。

当与用户开发生态系统中的其他工具连接使用时，IBM Rational BuildForge 可以提供一个详细的有关开发过程的记录，从而不用很长的准备时间就可以提供用于审计的报告。因为 BuildForge 精确捕获到了各类详细信息，如发布了什么，什么地方被修改了，进行了什么测试等，并提供一个详细的物料清单（BOM，Bill of Materials）用作遵规和审计检查的证据。

当与 IBM Rational 其他软件一起使用时，团队可以控制、自动化和跟踪他们的所有开发遵规行为，从一开始的编码一直到生产，包括各种必要的审批、工作流以及审计追踪记录等。

回页首

构建管理流程

如何面对上面所提出的挑战呢？首先软件开发组织应该明确有关构建的基本组成部分和其运作规律，特别是构建管理流程的搭建；其次应配合一些构建及发布管理工具基于有关构建的最佳实践经验循序渐进进行改进。本节主要从构建基础设施和构建管理流程的搭建进行阐述，在阐述过程中我们将整个构建管理流程进行了分解，这样有助于读者详细了解与构建相关的各个领域，以便进行有针对性的改进。后面的章节我们将主要对构建最佳实践经验和 IBM Rational 构建管理工具 BuildForge 进行介绍，最后一章将结合案例进行分析说明。

构建配置类型

构建的具体形式和许多因素有关，例如所采用的开发语言、操作系统及其环境、开发所采用的方法论等，但总体而言构建的配置类型可以分为私有构建、集成构建及发布构建。

私有构建是开发人员在自己工作空间内进行的构建，通常用于对自己修改的代码进行校验。集成构建是由核心职能部门或所指定的集成人员将不同开发人员的源代码收集在集成工作空间后进行的构建，集成构建由负责集成构建的人员（通常是开发组长或来自构建部门的人员）来进行，也可以通过调度程序自动定时进行。发布构建是由核心职能部门，通常是来自构建部门的人员执行的直接交付给内部或外部最终用户的构建。发布构建通常在隔离和受控的环境下进行。从构建发生数量来看，私有构建数量最大，集成构建其次，发布构建最少。

尽管构建可以分为以上三种不同配置类型，但并不意味着这三类构建采用不同的构建方式，实际上推荐的做法是重用同一组构建脚本，而仅在谁、什么地方、什么原因、构建内容等方面进行区别。使用一致的构建脚本有助于在早期阶段（私有构建或集成构建）而不是发布阶段发现错误。

端到端的构建流程域

构建管理流程不仅仅是简单的程序编译，它实际涉及一系列端到端的构建流程领域，包括：

版本控制

执行如工作空间的创建和更新、创建基线以及报告等版本控制活动，建立整个构建所运行的环境，并捕获构建流程中输入和输出的元数据，从而确保构建的可重复性和可靠性。

代码静态分析

检查是否所有的开发人员遵从了与开发语言最佳实践相关的某种编程规范或标准。作为构建的一部分进行代码静态分析可以保证代码能够被开发团队的所有人员修改和阅读。

编译

编译是将源代码转换为可以直接运行的可执行文件或者中间对象。并非所有的项目都必须对代码进行编译处理，比如一些项目使用可以直接运行的脚本语言，但大部分项目都包含编译过程。

单元测试

单元测试是构建质量控制的第一道大门，用于评估是否开发人员的工作成果可以在代码单元一级工作在一起。如果测试全面并很好地进行了测试用例设计，只有通过了所有单元测试的构建才有可能成为一个高品质的构建。

数据处理

并非所有的构建都是单纯的编码和测试，某些特定应用中编译只是整个构建流程的一小部分，主要的构建工作是对数据文件进行解析和转换，从而生成最终的输出，例如多媒体游戏软件从模型到三维图像的处理过程。另外，也有不少应用系统的构建与数据库中的应用元数据的设置相关，这部分数据处理也是构建的一个重要组成部分。

打包

打包是最终产品的一个包装过程，它是将构建的输出结果打包成完整的可安装文件形式。打包后的结果有时也称之为“分发文件”，以 Java 为例打包就是将 Java 类以及库文件打包为 JAR 或 EAR 文件从而安装在服务器上。

功能集成测试

功能集成测试或者也称之为“链接测试”是构建质量控制的第二道大门，通过在已部署的应用上执行整个功能测试用例集合的一个小的核心部分，来证明该构建可以进行进一步的测试，从而给测试团队以信心。

代码覆盖率

代码覆盖率测试是单元测试或功能集成测试的一个有效补充，它可以帮助您评估已经有多少代码经过了测试，从而明确其他的功能测试应集中在哪些方面（例如，未被测试用例遍历的代码）。

部署

部署是将构建迁移到运行环境的过程。构建通常会被自动迁移到立即相关的环境（例如集成测试和系统测试环境）下。在一个安全、受控的发布流程控制下，生产环境的部署有时也可以自动完成，但通常都是由一个单独的团队手动完成的。

构建基础设施

构建基础设施包括用于源代码管理的版本控制工具（如 IBM Rational ClearCase， CVS 等）、构建工具（如用于 C/C++ 的 make，用于 Java 的 Ant 等）以及参与构建的硬件（包括开发人员桌面机、构建服务器以及版本控制服务器等）。

构建的定义、执行和报告

构建的定义也就是构建的脚本化，例如使用 GNU Make 或 Apache Ant 进行脚本编写的过程。这些脚本定义了应用的各个部分应该如何编译和链接在一起以形成最终的系统或应用。这些脚本也可能将处理过程的其他部分自动化，例如数据库的配置和安装。在构建定义阶段可以定义任何可以自动化执行的部分。

构建执行或构建控制是以受控（通常是自动化）方式进行构建脚本的执行。构建控制也包括对构建结果（成功或失败）的报告。构建报告分为不同类型的报告，如基本编译报告、单元测试报告、发布报告等。

构建相关角色和职责

从上面各节的介绍可以看到构建流程是不同用户、不同团队之间工件以及沟通的重要渠道，许多不同角色的用户，从开发人员、测试人员到构建人员、项目经理甚至运营维护团队都会受构建流程的影响。但是这里我们主要介绍一下直接与构建流程相关的核心用户角色及其职责：

开发人员

编写源代码，执行单元测试，提供构建所需的工件。

构建工程师

通过工具和/或编写脚本建立构建流程，并负责构建流程的自动化执行。

部署工程师

负责将构建输出结果迁移到运行环境中。在大型企业中部署工程师通常来自与开发部门不同的 IT 运营部门。

在一些企业上面的一些角色可能是交叉的，即一个人可能即担任构建工程师同时又是开发人员。

构建频率

构建频率是软件开发团队另外一个需要做的重要决策，即多久进行一次集成构建或发布构建。在软件开发上普遍使用了以下三种不同的构建调度策略：

每周构建

每周构建普遍用于有多支开发团队的大型软件开发项目，不同开发团队的成果或变更每周定期提交到一个系统集成区域，在集成后进行系统集成构建。

每日构建及冒烟测试

每天（通常在夜间）基于当天所提交的变更进行一次集成构建，另外在构建后进行一组冒烟测试。

持续集成构建

一种广泛用于敏捷开发方法的构建方式，即不断对版本库进行监控，一旦有变化进入版本控制库就自动开始集成构建，持续集成构建通常构建时间较短（小于 10 分钟）并且伴随完整的单元测试。

通常而言构建越频繁，也就越有可能尽早发现集成问题。因此用户应该努力尽可能频繁地进行构建。

回页首

关于构建的最佳实践经验

软件开发生命周期的其他规程，例如需求管理、配置与变更管理以及测试等长期以来总结出了不少最佳实践经验，同时形成了多种商业化工具、出版物以及覆盖这些最佳实践的分析报告等，但有关构建和发布的实践及工具仍然较少，但这种现状正在得以改变，特别是 IBM Rational BuildForge 的出现。尽管许多软件开发组织通过自己开发来进行构建和发布过程的管理，但这些方案功能一般不太完整，同时没有很好的可扩展性，很难进行维护从而满足企业需求的不断增长。

同其他 IBM Rational 产品一样，IBM Rational BuildForge 里渗透了许多关于软件构建和发布管理的最佳实践经验。由于不同用户的情况和需求不同，因此这里列出的最佳实践经验没有实施次序上的要求。不同用户在应用这些经验时可以根据自身情况，从容易见效、投入低的实践开始，逐步进行构建和发布过程的改进。

- 完整的可再现性
- 构建流程的自动化和与关键系统的集成
- 从物理资源中进行流程及元数据的抽象
- 流程优化及架构设计支持
- 构建加速技术
- 所有使用人员能够进行构建信息的集中访问和协作
- 尽早构建，经常构建
- 构建流程与部署环境的集成
- 根据业务目标进行性能度量

本文的后续部分，将介绍 IBM Rational BuildForge 的产品特性，以及如何利用 IBM Rational BuildForge 建立起的构建管理平台，改进构建流程的管理和控制效率。

参考资料

- 本系列的 第二部分 将介绍 IBM Rational BuildForge 的产品特性。
- 本系列的 第三部分 将介绍如何利用 IBM Rational BuildForge 建立起的构建管理平台，改进构建流程的管理和控制效率。

- 访问 IBM Rational 软件交付平台 V7 专题，了解 Rational V7 产品的方方面面。
- 访问 IBM developerWorks 中国网站 Rational 专区 了解更多关于 Rational 产品的信息。

第 2 部分: IBM Rational BuildForge 概述

在本文的 第一部分 我们介绍了 IBM Rational BuildForge 进行构建管理过程改进的原理和方法。本文的第二部分我们将介绍 IBM Rational BuildForge 的产品特性。

IBM Rational BuildForge 概述

IBM Rational BuildForge 是专门用于构建和发布流程管理的软件。BuildForge 始于 2001 年，在 6 年时间内帮助多家用户在构建/发布的可靠性、可重用性以及可跟踪性等多个方面进行了明显改进。BuildForge 于 2006 年 1 月完成了与 Rational 配置管理产品线（ClearCase 以及 ClearQuest）的集成，在 2006 年 5 月被 IBM 收购并加入到 IBM Rational 产品家族中。经多家客户的使用证明，利用 BuildForge 建立起的构建管理平台可以明显改进构建流程的管理和控制效率。BuildForge 通过提供一个可扩展的构建和发布框架，可以帮助用户减少软件交付成本，提高软件构建的可见性和可跟踪性，同时可以更好地进行 IT 治理和遵规，以更快的速度进行问题解决，明显提高软件质量。BuildForge 的功能结构可以结合下图进行说明。

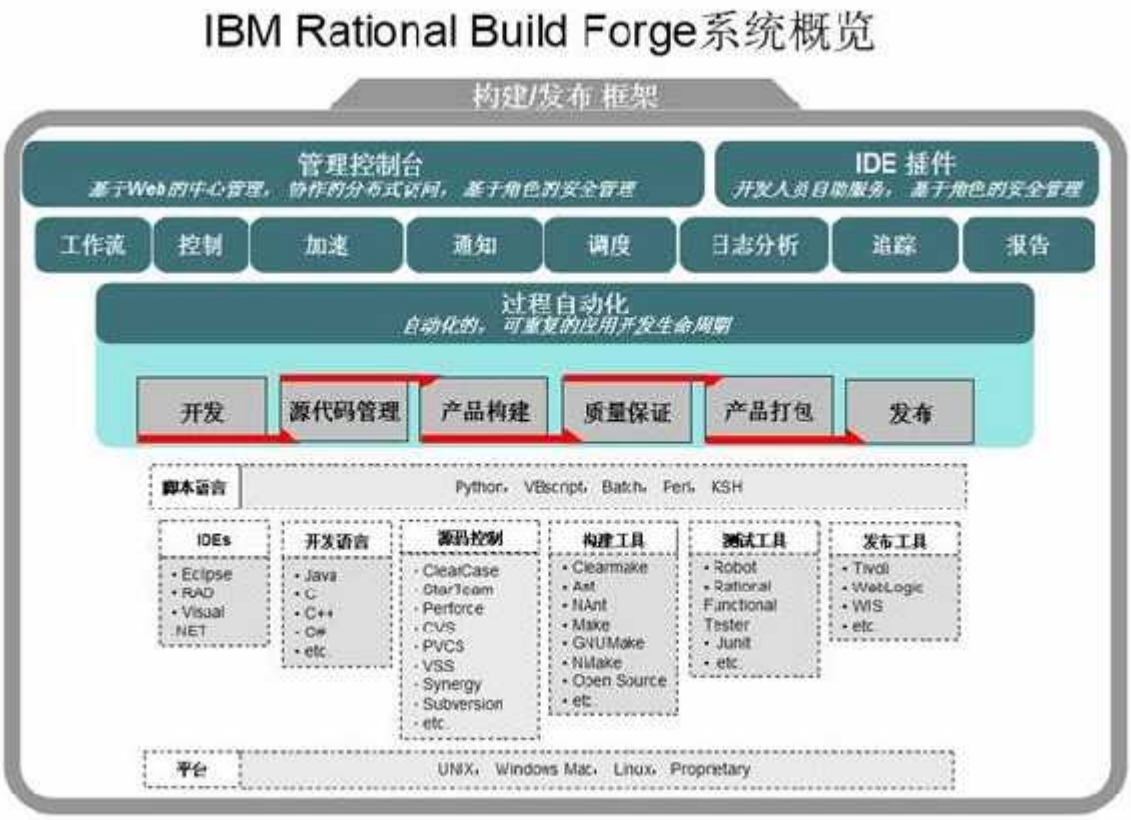
关于本系列

本系列描述了利用 IBM Rational BuildForge 进行构建和发布过程管理及自动化的方法，讨论了如何通过整合项目组、流程以及系统来改进软件开发效率，从而提高整个开发团队的效率，改进产品质量，更好地遵规。

前一部分介绍了 IBM Rational BuildForge 进行构建管理过程改进的原理和方法。本文对 BuildForge 的基本功能进行了描述。后一部分是基于 BuildForge 的集成构建平台解决方案介绍，以及给软件开发、质量改进以及遵规带来的好处，最后是一些用户的案例研究。

了解 本系列其他部分。

图 1: BuildForge 的功能结构



BuildForge 提供了一个框架来自动化“整个”端到端流程。不仅仅自动化单独的任务，而且自动化流程中不同步骤的信息传递和沟通。BuildForge 可以集成用户现有的脚本和工具，因此没有必要替代现有的资产。BuildForge 所提供的全面的应用开发流程管理方案可以提供应用开发周期完整的管理和控制。通过集成多种工具，BuildForge 可以将复杂的流程标准化、自动化并进而进行优化，从而提供一个可重复的、可靠的应用开发生命周期流程。

自动化只是 BuildForge 所做的第一步，BuildForge 还提供了诸如构建加速、自动化通知、调度、日志自动分析等多种功能。通过安全的、基于 Web 的访问机制可以管理各类任务，对各种类型团队、甚至异地开发团队进行支持。

回页首

IBM Rational BuildForge 的工作原理

在 Rational BuildForge 的核心是一个使构建和发布流程更为有效的自动化流程管理方案。Rational BuildForge 作为一个集中流程库可以在不同计算机上进行构建的运行、跟踪、调度和任务指派。本节主要描述一下 Rational BuildForge 产品可以提供的自动化范围，详细解释一下 BuildForge 的工作原理，看 BuildForge 是如何使不同工具、多台机器工作在一起的。

流程的定义和运行

在 Rational BuildForge 中，用户可以将一个流程定义为一系列的任务，在 BuildForge 中称流程为“项目”。如下面的产品截图所示，这里列出了三个项目，其中项目“Web 应用程序构建演示”的属性如右下页面所示，包括项目名称，项目的访问权限（只有 Build Engineer 组的人可以访问），允许的最大并行线程数（Max Thread），项目相关的环境定义（Environment，包含了大量的环境变量定义），项目所运行的服务器选择器（Selector，根据条件如操作系统类型、内存利用率、CPU 空闲率等可以动态进行构建服务器选择），开始运行构建的通知方（Start Notify），构建成功的通知方（Pass Notify），失败的通知方（Fail Notify），项目成功后启动另一个项目（Pass Chain，因此 BuildForge 可以支持基于多项目的构建工作流），项目失败后启动另一个项目（Fail Chain）等功能。

图 2: 在 BuildForge 中用户可以将一个流程定义为一系列的任务

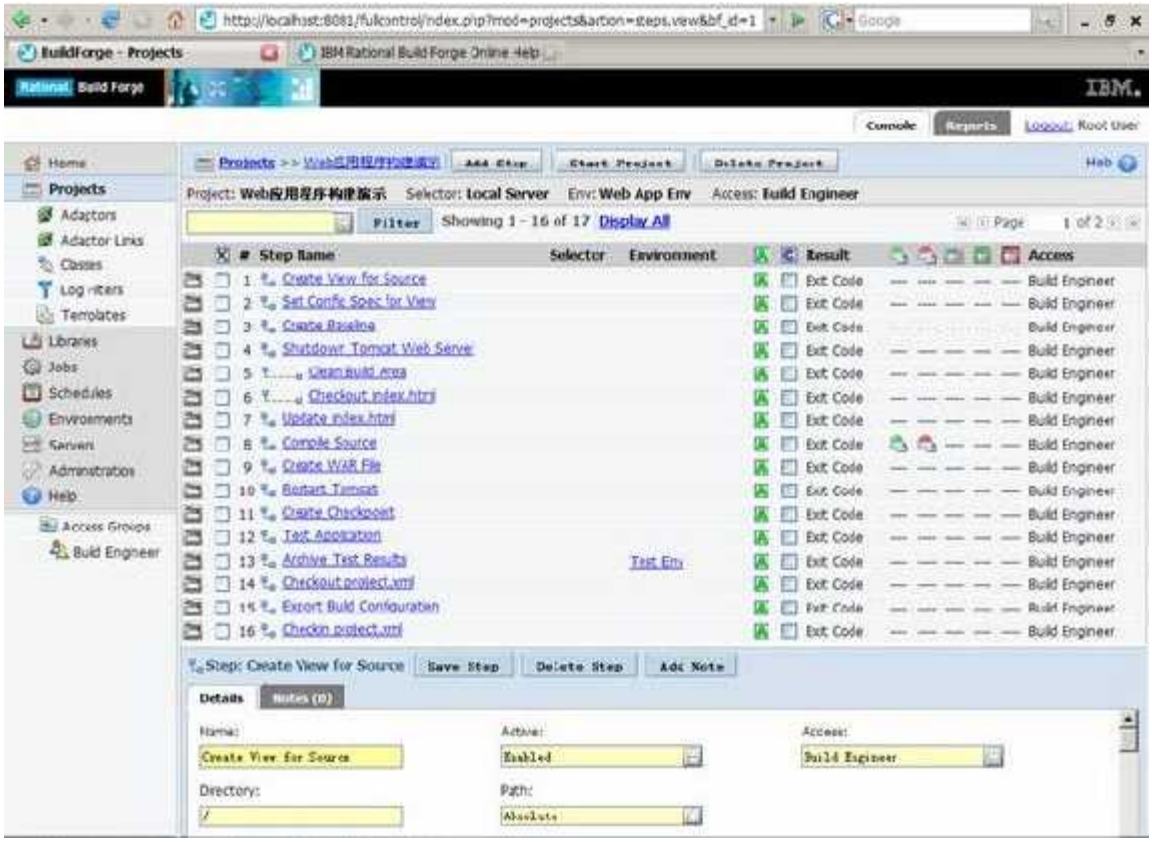
在项目属性的构建标识（Tags）页面中用户可以灵活定义构建编号规范，如下图所示。

图 3: 在 BuildForge 中用户可以灵活定义构建编号规范

上图中示出了项目“Web 应用程序构建演示”所使用的构建编号规范，该编号由四部分组成，中间用“.”分隔，这四个部分分别由四个变量表示，代表了主版本号、小版本号、补丁号以及构建序号。而且在 BuildForge 中除了可以自定义变量外，还可以分别定义这些变量的编码行为，例如 BUILD 从 38 开始，每次执行时该变量自动加 1。

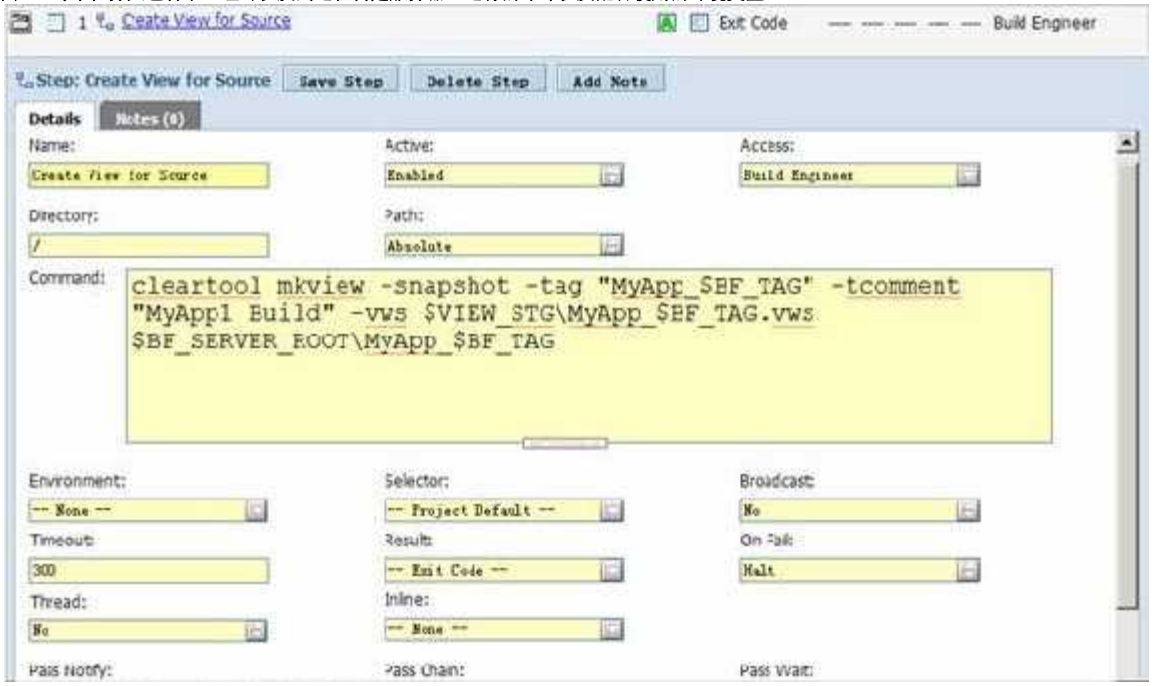
下图示出了具体的构成一个项目的所有任务，从截图中可以看出该项目包含了 17 个任务，这些任务按编号顺序执行，其中任务 5（清理构建工作区）和任务 6（从 ClearCase 中检出 index.html）可以并行同时运行，当两个任务均结束后开始任务 7。

图 4: 具体的构成一个项目的所有任务



每个任务又包含了一组可以发到远程构建服务器上运行的命令以及配合使用的环境变量。

图 5: 每个任务又包含了一组可以发到远程构建服务器上运行的命令以及配合使用的环境变量



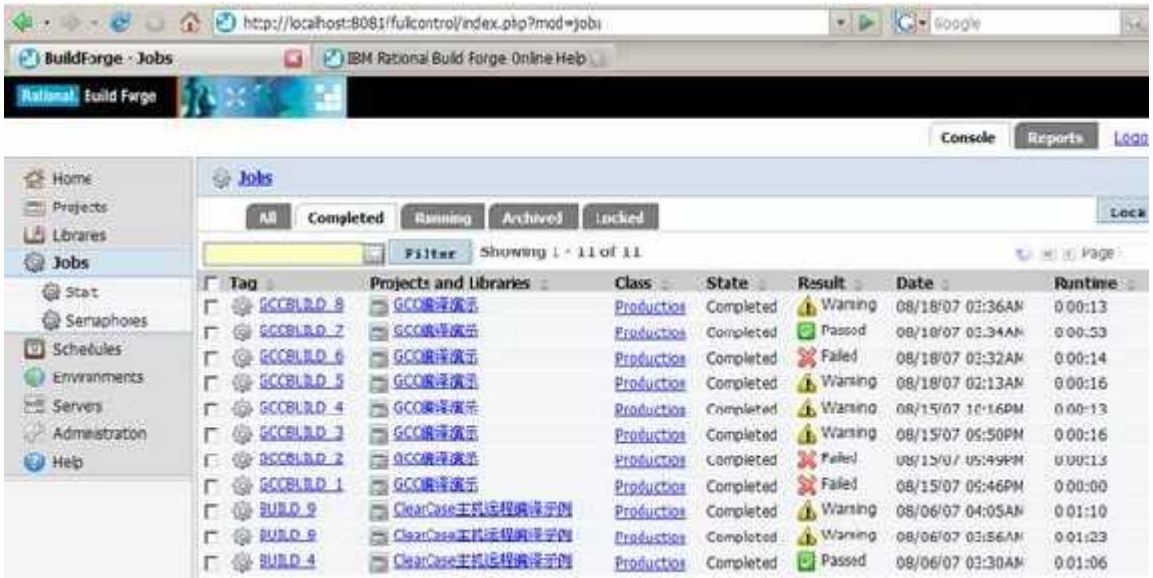
以第 1 个任务“Create View for Source”为例，如上图所示其中包含了一个创建 ClearCase 静态视图的命令，此外还可以在任务属性中定义其他内容，例如可以指定该任务使用不同于项目的环境变量、运行在不同的服务器上等等。

一旦一个项目定义完毕就可以使用 BuildForge 调度程序调度该项目到时自动运行，或者可以作为授权用户登录 BuildForge 手动启动项目。一旦项目开始运行，系统将根据所选择的构建服务器（直接指定或动态选择）在该服务器上运行特定的命令，命令执行结果将被回写到 BuildForge 数据库中。如前所述用户可以利用 BuildForge 多项目串接的特性进行构建工作流设计，从而根据前一项目的执行结果自动触发后续项目的运行。

项目运行结束后根据项目或任务定义可以自动通知相关人员告知项目的运行结果（成功或失败），例如一个集成构建项目，如果其成功则通知测试/QA 人员在指定位置接收构建结果开始测试；如果其失败则通知相关开发/集成人员在指定位置接收构建结果对失败原因进行分析。

下图示出了已完成的构建结果，可以清楚地看出某个项目的历史构建结果记录，哪些构建成功、哪些失败、哪些成功但带有警告信息、构建时间、编号、构建用时等关键信息，点击具体某次构建还可以看到更为详细的说明。

图 6: 已完成的构建结果



一个项目内的任务可以是传统的构建任务，例如对应用程序的编译，但 BuildForge 并不局限于此，实际上在 BuildForge 中可以定义和运行多种多样的任务，从源代码的检入/检出，到源代码的编译、链接、测试以及部署，下面是一个典型的构建项目可能包括的步骤：

1. 将源代码文件从版本库取到构建服务器上
2. 对源代码进行编译并报告编译进度
3. 在成功的编译结果上执行自动化单元测试
4. 创建一个安装程序
5. 将该安装程序上传到一个站点，并通知团队成员可以使用该安装程序进行应用安装
6. 通过运行安装程序生成可执行文件
7. 运行自动化测试，对该可执行文件进行测试
8. 报告测试结果
9. 启动另一个附属项目来更新标准链接库
10. 将可执行文件和其他文件传递给 QA 部门以进行后续测试
11. 将完成的发布工件部署到生产环境，例如 Web 服务器或者发送到 CD 工厂

使用 Rational BuildForge 也可以完成其他类型的项目或任务，如：

- 将一个源文档文件（MS Word）转换为一个 PDF 格式的文件，将其复制到 Web 站点的下载链接位置，更新 Web 页面的相应文件列表，并更新网站搜索索引。
- 允许开发人员运行单个组件的构建，并每天运行一个主构建来编译整个产品，该主构建会引用其他的组件构建结果。
- 在 5 或 20 台机器上以串行或并行方式运行同一个构建流程。
- 允许项目组运行他们自己的构建，但只有授权人员才能更新 Web 服务器。

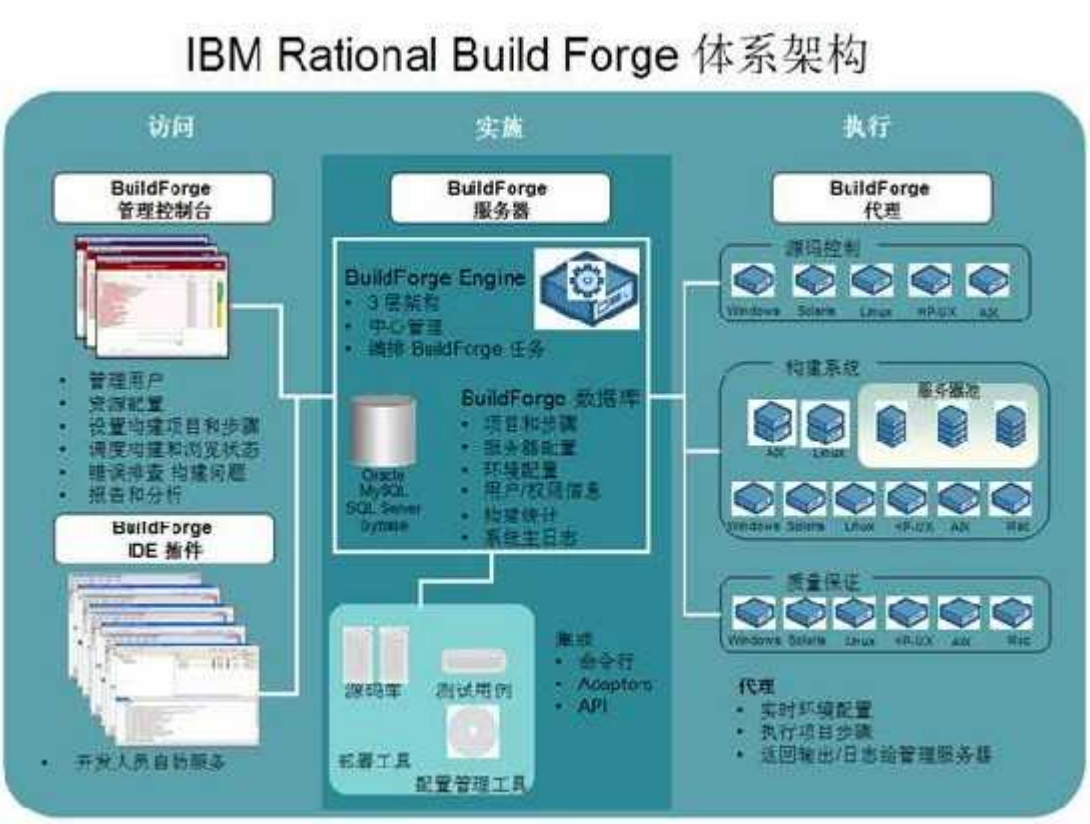
回页首

BuildForge 系统是如何运作的

IBM Rational BuildForge 的设计思路非常简单，即不管当前用户使用的是什么工具（包括配置管理、编译工具等）它都可以为开发团队提供可靠的构建发布管理系统，这一想法最终落实到了一个灵活而健壮的自动化基础设施中，从而可以为遍布全球的分布式开发团队使用。不管目标服务器的位置在哪儿，或操作系统是什么， IBM Rational BuildForge 可以使用户通过基于 Web 的浏览器界面即可在多台服务器上执行具体的命令。下图是 BuildForge 的功能架构图，其中包括：

- BuildForge 管理控制台（management console），这是一个基于 Web 的 PHP 应用，运行在一台 Apache HTTP 服务器上。通过管理控制台用户可以将命令组织到步骤和项目中来进行构建项目的编制，同时管理这些命令使用的服务器资源和环境变量，还可以进行构建运行监控、构建报告生成、构建任务调度等管理任务。
- BuildForge 引擎（engine），使用通过管理控制台输入的信息并将其存放到数据库中，同时与安装在目标服务器上的代理（agent）进行通信来执行具体的项目任务和发送通过。BuildForge 引擎根据管理控制台输入的指令来工作。
- 核心的关系型数据库，其中存放了完整的用于跟踪每次项目执行的项目信息。用户信息和系统动作也被保存在数据库中以便从中抽取数据进行分析，满足审计和报告的要求。该数据库可以使用多种数据库（如 Oracle、DB2、SQL Server 以及 MySQL 等）。
- 在分别运行不同操作系统的计算机上安装的代理（Agent），代理程序用来响应来自系统的指令。代理程序从数据库中抽取环境信息在目标服务器上为项目动态构造相应的环境，从而保证项目中的每条命令都是在完全满足命令执行的环境下执行的。
- BuildForge IDE 插件，作为开发人员可以使用 BuildForge IDE 插件直接在 IDE 环境下进行构建任务的提交，运行监控和报告。从而实现构建的开发人员自助服务。现在支持的 IDE 插件包括 Eclipse、.Net 以及 Rational Application Developer。

BuildForge 的功能架构图



一些 BuildForge 的使用场景

借助上面的 BuildForge 组件，BuildForge 系统可以象一个框架一样将用户的开发资源紧密结合在一起。下面是一些典型的 BuildForge 使用场景，这些均可以使用 BuildForge 来进行自动化和跟踪：

- 配置管理经理使用 BuildForge 管理控制台每天构建某旗舰产品的最新开发版本，而且该产品有用于多种操作系统的版本。BuildForge 系统了解该项目编译的每个具体步骤，根据规格要求在运行时 BuildForge 系统创建了对应不同操作系统的相应环境。在运行到某些公共步骤时，项目分出两个子流程，一个用于 Red Hat Linux 操作系统，另一个用于 Microsoft Windows 系统，每个子流程都从版本库中检出各自的代码在不同服务器上并行开始产品编译。
- 通过整个构建流程，配置管理团队可以对整个项目的处理流程有一个清晰的、实时的了解。如果这些流程中的其中之一出现了问题，相应失败流程会通知负责本部分程序的开发人员。一旦项目成功完成，构建得到的可执行文件被上传到内部的 Web 服务器上进行测试，同时自动通知相应的 QA 团队。
- 开发人员对该产品中属于自己工作的部分进行测试，如果开发人员通常使用的构建服务器已经宕机，系统会自动将处理流程重定向到其他可用的服务器上并记录本次运行实际使用的服务器。当流程执行结束后开发人员会收到 e-mail 通知，告知最新的构建结果存放在哪里。服务器宕机对开发人员和配置管理人员都不会造成影响，他们也不需要立即对此做出反应。
- Web 站点管理员创建了一个流程，每天数次从一台部署服务器收集所有更新过的 Web 页面并将他们复制到公共服务器上的相应位置。通过一个简单的自动化处理流程，该 Web 站点管理员能够更新文件、重启 Web 服务器并将更新文件的历史版本归档到另一台服务器上。现在当该站点管理员不在的时候，其他团队成员可以经过短时间培训即可了解该过程并帮助执行该流程。
- 项目组使用多台机器进行产品压力测试。该测试不仅需要与配置管理团队配合进行，而且需要使用隶属于不同部门的多台机器在夜间进行。由于机器分属不同部门，该项目组被授予了严格的访问控制权限，因此这些部门可以不用担心破坏服务器的环境就可以安全地共享服务器资源。任何在测试期间生成的文件可以在测试结束后自动清除，从而恢复服务器的原有状态。

IBM Rational BuildForge 的优势

IBM Rational BuildForge 完全体现和支持第 1 部分提出的 9 大构建最佳实践，其独特的设计方法展现了区别于其他产品和方法的明显优势，这些优势的结果表现在以下两个方面：

- 扎实定位在任何软件项目的真正提交件——可执行文件的构建过程及跟踪上。

- 忠实报告在开发期间到底发生了什么。

构建关键的、可靠的开发工件——可执行文件

通过第 1 部分的概述和第 3 章“当前构建及发布过程面临的挑战”可以看到，许多软件开发企业中有关构建和发布管理的科学化、自动化是比较薄弱的，绝大部分精力基本都放在在开发本身上，而忽略了从源代码到最终产品的流程优化，结果一个常见现象是配置管理人员或构建人员人手不足，而且从代码到最终产品的过程基本以手工为主，不仅容易出错，而且容易依赖某些个人的特定知识，结果这一过程中出现的问题往往会对开发做出的巨大努力大打折扣，有时用户甚至认为在产品最终交付前这方面不出问题才怪呢。但是产品开发团队的最终交付件就是可执行文件，它是面对最终用户的最终产品形态，这也是 BuildForge 关注的重点。下面通过几个例子进一步说明关注可执行文件构建管理的重要性和必要性：

- 开发人员写了几个月的代码，结果发现一个可执行文件由于不正确的构建而在用户应用系统中工作不正常。
- 您的版本控制系统告诉您一个缺陷已经修复了，但是实际上该缺陷修复分支上的代码根本没有并入最终的发布版本中。尽管配置及变更管理系统可以用于跟踪代码变化，但当用户想跟踪某个特定问题到底解决没有时，它不会告诉你在最终可执行文件中到底是否包含了该问题的修改。
- 用户的缺陷跟踪数据库指出一个问题已经被修复了，但这实际上依赖于人的录入。而可执行文件实际上才是产品的最终和最真实的记录。为了做到精确的审计，必须以从可执行文件提取数据的方式进行开发管理。当使用 Rational BuildForge 来构建产品时系统可以报告出在可执行文件中嵌入的缺陷修复。因此 BuildForge 将开发流程紧密结合在一起，从而允许用户进行逆向跟踪，从问题追溯到产生问题的代码。结果构建和发布管理流程可以和应用交付流程中的其他阶段紧密结合达成一致性管理，从而将开发可能遇到的灾难最小化。

忠实报告在开发期间到底发生了什么

Rational BuildForge 的报告是从实际运行的流程中获取到的，不管该流程是软件编译流程、自动化测试流程、Web 站点更新流程还是什么其他的流程。这与其他基于文档的开发管理系统不同，基于文档的开发管理系统需要手工输入数据。如果使用 Rational BuildForge 来打造集成的开发环境，可以得到下面的好处：

- 用户的源代码控制系统告诉用户什么代码被检入了；而 Rational BuildForge 则告诉用户在发布版本中实际包含的代码到底是什么。
- 用户的缺陷跟踪系统告诉用户哪些缺陷修复会放在本次发布中；而 Rational BuildForge 则可以告诉用户在发布版本中实际嵌入了哪些缺陷修复，并且可以执行自动化测试对这些修复进行校验。
- 当一个用户打电话报告在一个具体软件发布中出现的问题，您可以利用 Rational BuildForge 来确定在那次发布中都包含了哪些东西。
- 当用户在自己公司的 Web 站点上碰到了一个问题，Rational BuildForge 可以精确告诉用户站点都发生了哪些变化，并且帮助回滚到早期版本。

IBM Rational BuildForge 产品清单

- **IBM Rational BuildForge 标准版**
  提供了全面的构建和发布处理自动化功能，支持分布式服务器访问及流程跟踪。
- **IBM Rational BuildForge 企业版**
  在标准版的基础上进一步提供一些高级功能，如流程的并行执行，服务器配置自动发现和动态服务器分配。
- **IBM Rational BuildForge Adaptor Toolkit**
  提供了直接可用的与一些第三方软件（如 ClearCase、ClearQuest、CVS、Subversion 等）的集成以及灵活的 API 接口。为了达到更好的集成、信息共享以及源代码、缺陷、测试结果以及更多内容的跟踪，用户通过这一软件和 API 接口可以建立构建和发布环境与用户自己的第三方配置管理工具、专有工具的无缝链接。
- **IBM Rational BuildForge QuickReport**
  基于 Web 的报告定制工具，用户无需 SQL、脚本以及 API 方面的知识通过点击即可生成各类针对构建数据库的报告、图表等。

回页首

小结

本文概述了 IBM Rational BuildForge 的基本特性。在下一部分，将介绍基于 IBM Rational BuildForge 的集成构建解决方案，以及案例研究。

第 3 部分: IBM Rational BuildForge 集成构建解决方案及

在本文的前面两部分我们介绍了 IBM Rational 进行构建管理过程改进的原理和方法。本文的第三部分我们将介绍 IBM Rational BuildForge 的实施案例研究。

IBM Rational BuildForge 集成构建解决方案

经过业界许多用户的使用证明 IBM Rational BuildForge 可以帮助软件开发组织加速软件开发流程，增强产品质量，更好地支持异地分布的开发团队并满足审计和遵规需求，下面就从这三个层面来描述一下围绕 BuildForge 的集成构建解决方案。

加速软件开发流程

开发流程中的瓶颈

Rational BuildForge 提供了多种特性来解决软件开发过程中的常见瓶颈，从而加速用户的软件开发流程，这些瓶颈包括：

- 软件开发过程中从一个团队到另一个团队低效的流转
- 缓慢串行的处理速度
- 由于系统问题造成的停工
- 硬件资源的利用不充分
- 批处理过程不透明

因此本节主要描述 BuildForge 的功能及特性是如何解决上面的问题或瓶颈的。

加快流程流转，减少徒劳工作

Rational BuildForge 的自文档化（在项目创建及执行过程中自动形成相关文档）特性保证了更佳的团队协作，特别是分布在不同工作地点的团队。位于开发周期上游的团队不必担心没有将正确的信息传递给下游团队，这样他们可以将精力集中在如何正确处理他们自己的工作以及如何对系统中的信息进行封装方面，他们知道系统会对这些信息进行存储和传递。下面一些 BuildForge 的特性提供了这方面的支持：

- 物料清单（Bill of Materials）：系统将流程运行过程中相关内容的描述（包括使用检查点 checkpoint 列出发生变化的文件）组织起来并存放到一个包中。

图 1: 物料清单的一个例子

上图示出了物料清单的一个例子，从 Job Steps 中可以看出一个构建流程包含的全部步骤，这些步骤运行在哪台服务器上，用时分别多少等；还可以在 Step Manifests 看到每个步骤具体的环境变量及系统参数设置；在 Checkpoints 可以看到本次构建增加了一个新的文件 hello.exe。

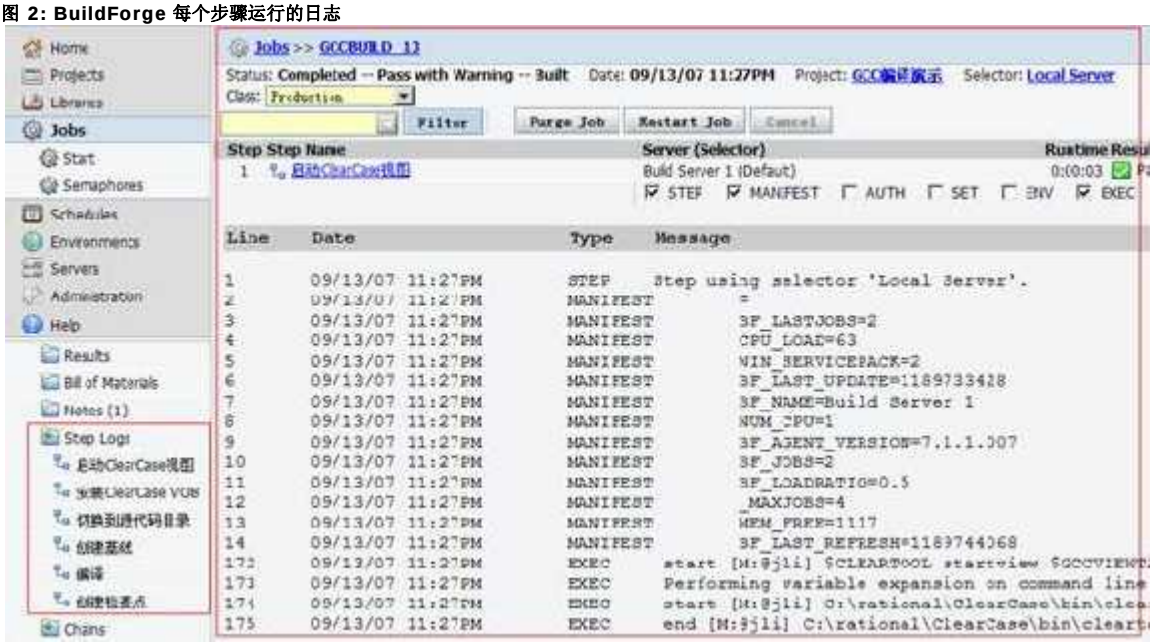
关于本系列

本系列描述了利用 IBM Rational BuildForge 进行构建和发布过程管理及自动化的方法，讨论了如何通过整合项目组、流程以及系统来改进软件开发效率，从而提高整个开发团队的效率，改进产品质量，更好地遵规。前面第 1 部分介绍了 IBM Rational BuildForge 进行构建管理过程改进的原理和方法。第 2 部分对 BuildForge 的基本功能进行了描述。最后一部分是基于 BuildForge 的集成构建平台解决方案介绍，以及给软件开发、质量改进以及遵规带来的好处，最后是一些用户的案例研究。

了解 本系列其他部分。

- 日志（Logs）：系统记录了活动以及命令的输出，并根据用户及项目的安全性设置跨组织地进行系统运行日志共享。下图示出了一个 BuildForge 每个步骤运行的日志，根据需要还可以对日志内容进行过滤。

图 2: BuildForge 每个步骤运行的日志



- 注释（Notes）：BuildForge 允许用户在项目任务、项目运行结果添加注释，增强项目任务的可维护性，项目运行结果的可读性。

图 3: BuildForge 允许用户在项目任务、项目运行结果添加注释

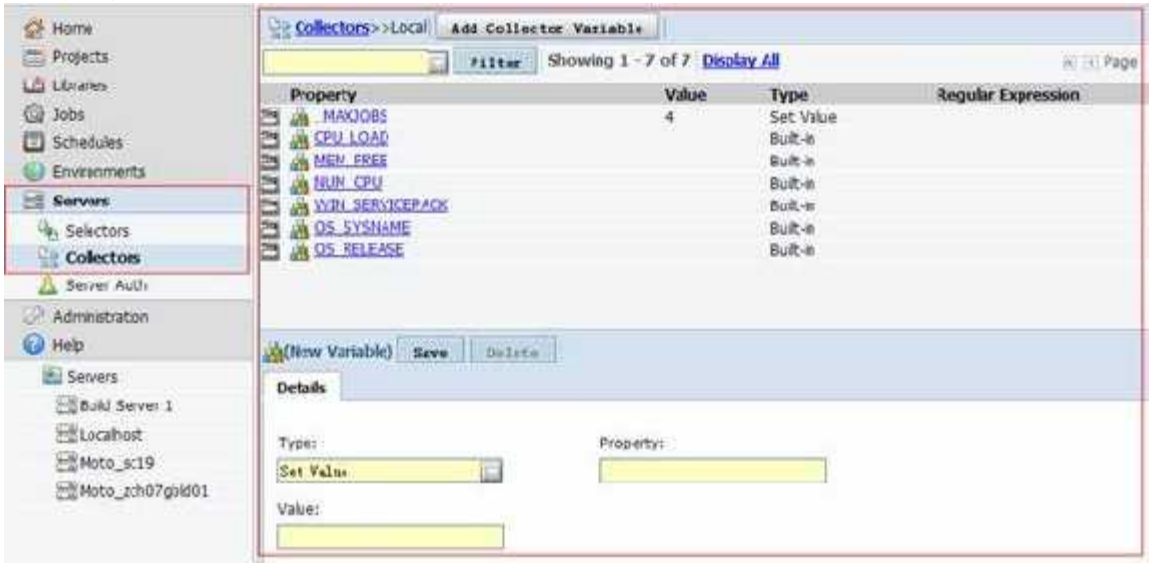
这些特性结合在一起清晰诠释了在开发期间到底发生了什么。系统可以跟踪谁在哪台机器做了什么事情，结果是什么，消耗时间是多少等。这种跟踪能力不仅加强了各类人员对于流程及其运行的理解，还有效帮助企业遵守审计规则。

加快执行速度的同时延长系统正常工作时间

使用 Rational BuildForge，用户可以通过并行处理以及服务器池（server pooling）技术加快构建速度并延长系统正常工作时间。用户可以使用这些特性来：

- 将多台机器组成一个服务器池，然后在该服务器池上运行多个项目。只要一台服务器被充分占用，系统可以自动将处理流程重定向到同池的其他服务器上。

图 4: 将多台机器组成一个服务器池



上图示出了在采用服务器池时 BuildForge 的具体做法，首先 BuildForge 使用服务器收集器（Collector）来定义收集服务器哪方面的信息，例如 CPU 负载、空闲的内存、操作系统以及操作系统版本、CPU 数量等，这些信息可能会用作服务器分池的条件。下图示出了在物料清单（BOM）报告中在项目运行过程中实际采集到的信息。

图 5: 在物料清单（BOM）报告中在项目运行过程中实际采集到的信息

然后 BuildForge 通过服务器选择器（Selector）提供了服务器池的组织方式，例如把所有 OS_SYSNAME 为 Windows XP 并且空闲内存大于 512 M 的机器组织成一个池，接着将指定某些 Windows 项目运行在这个服务器池上。这样就充分利用了服务器资源，而且大大增强了构建系统的健壮性。

- 用一个流程即可在多台机器上启动多个并发执行的任务从而更快得到结果。如下图所示，同一项目中的两个步骤 5 和 6 可以并行执行，用户甚至可以指定步骤 5 在某台机器上运行，步骤 6 在另一台机器运行，也可以交由服务器池自动调度。

图 6: 可在多台机器上启动多个并发执行的任务

- 在不同机器上启动对同一应用程序不同操作系统版本的处理。例如某个应用程序的发布包括 Windows、Linux 以及 AIX 三个版本，我们可以同时在三台不同操作系统的机器上在它们各自的环境中进行同一应用的构建和发布。
- 如果 Rational BuildForge 尝试在某台机器上运行而该服务器宕机或不可用时，处理流程会递交给同服务器池的另一台缺省指定的机器，使得流程不会被中断。
- 同一任务在服务器池所包含的每台机器上运行一次。例如，如果用户想把一个代码更新部署到所有的 Linux 服务器上，这个更新可以立即以并行方式进行部署。

使用较少的硬件达到更快的速度

在进行 IBM Rational BuildForge 实施时，可以实现将部门的硬件资源以受控方式进行共享。没有这种集中管理方式，一个部门往往按项目组分配机器，例如某台 AIX 服务器就是给某项目组用的，其他项目组无法使用，结果造成硬件资源随着时间推移利用率不高。但如果使用 Rational BuildForge 就可以使下面的使用场景成为可能：

- 一个部门需要偶尔在 Linux 服务器上运行一些流程，通过 BuildForge 无需直接进行物理访问或对系统进行修改就可以提供受控的访问权限。例如该部门只能在该 Linux 服务器上运行某些特定的任务，而且可以在这些任务执行结束后自动将产生的结果清除掉。如果这些任务的运行过于频繁还可以禁止超过限制的运行请求。
- 一个部门的服务器可以设置为另一个部门的备份服务器，如果一个部门的关键服务器宕机，系统可以自动使用备份服务器直到首选服务器重新可用。
- 可以将一组机器组织成服务器池，不管涉及的部门或项目是什么都可以在服务器池上运行任何需要它们的流程。
- 可以仅将本部门一台机器的一部分运行能力赋予其他部门。

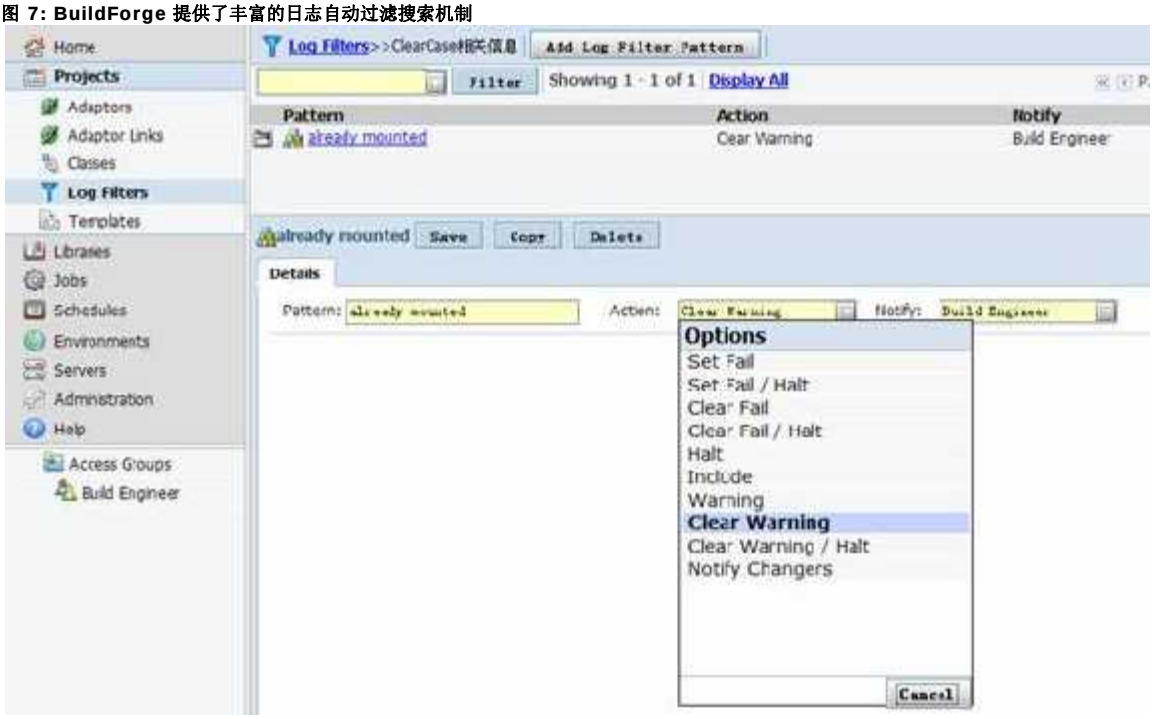
这些场景总的说来可以在提高处理速度的同时减低企业成本。

流程透明度和用于持续改进的分析

由于 Rational BuildForge 可以跟踪用户从编码到生产期间所执行的所有活动，因此在 BuildForge 中包含了相当丰富的信息，这些信息可以帮助团队提高生产效率。Rational BuildForge 可以生成一组预先定制好的报告，供用户用来分析自己的流程并持续进行改进。同使用大量的批处理文件或脚本却只能提供对流程有限的可见性相比，使用 Rational BuildForge 可以帮助用户：

- 不用搜索大量的日志记录就可以快速定位问题。如下图所示 BuildForge 提供了丰富的日志自动过滤搜索机制，用户可以抽象并定义错误或警告信息的模式，例如“ERROR”、“MYSYS_ERROR_CODE”等，同时可以定义一旦在日志或返回信息中搜索到满足这些模式的字符串后需要做什么样的处理，例如置为错误并终止运行，置为警告并继续运行，尽管返回信息为警告但可以清楚警告标志并继续运行等等。另外还可以定义需要通知哪些角色的人员。

图 7: BuildForge 提供了丰富的日志自动过滤搜索机制



- 检查每个流程甚至每个步骤的实际执行时间，从而发现运行趋势以及反常的运行。下图示出了通过 BuildForge Performance Report 对“GCC 编译演示”项目的多次运行过程进行的分析。包括时间趋势分析以及各个步骤的时长分布。

图 8: “GCC 编译演示”项目示例

- 跟踪问题发生的具体位置，属于跨部门的问题还是某个部门内的问题。
- 随着时间推移跟踪服务器的利用率。
- 发现代码变化最频繁的部分并定位错误发生的热点地区，改进项目计划。
- 使用 Rational BuildForge 公开的数据库模式产生用户自己的报告。从 BuildForge 7.0.1 开始加入了一个新的可选模块 BuildForge QuickReport，通过 QuickReport 可以基于 BuildForge 数据库简单地通过点击即可生成基于 Web 的报告、图表。下图分别示出了一个简单的对各项目构建成功率进行统计的报告的设计界面及最终结果。

图 9: 一个简单的对各项目构建成功率进行统计的报告的设计界面

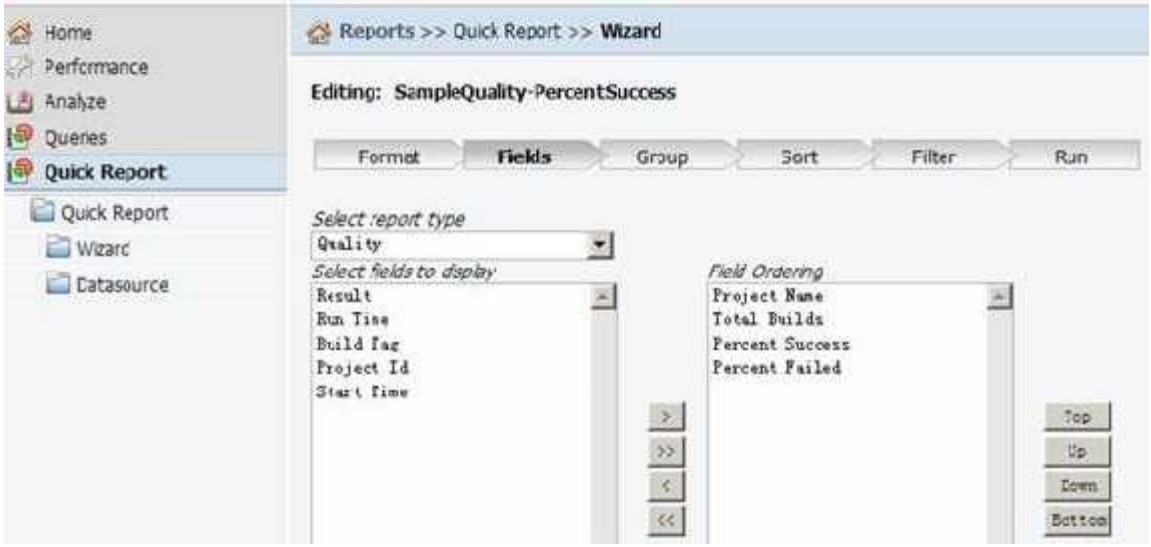


图 10: 一个简单的对各项目构建成功率进行统计的最终结果

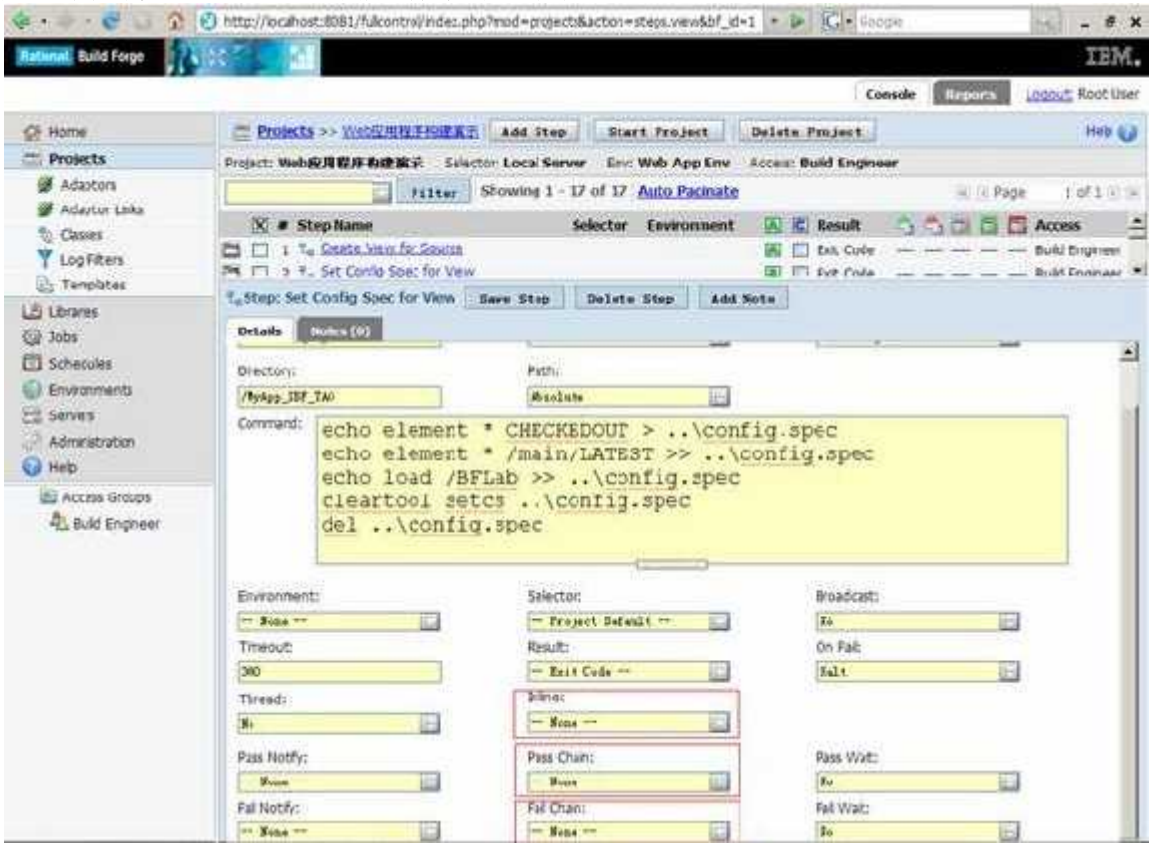
改进产品质量

当然，单纯加快软件开发速度是不够的，如果在组织中实施了 Rational BuildForge 用户还可以从拥有了统一的构建和发布自动化系统、标准化的开发流程以及更好的团队沟通而得到多种与质量改进有关的好处。

可重复的流程

通过 IBM Rational BuildForge 每次都是以一致的方式运行构建和发布流程。Rational BuildForge 可以根据用户需要多次重复运行一个项目，例如使用上一次构建的环境反复运行后续构建， BuildForge 还可以显示用户在运行时定制的值。BuildForge 解决方案非常适合重复已有的流程，反复进行测试或问题诊断的开发方式，如迭代化开发、敏捷开发等。甚至用户可以创建流程中的共享片段，使该片段成为一个共享项目，当其他项目需要该共享项目内容时直接调用共享项目即可。这样通过标准化流程提供了有价值的质量控制，减少了新项目的设置及启动时间。

图 11: 某个流程步骤可以通过 Inline 调用另一个共享的项目流程片段



如上图所示，某个流程步骤可以通过 Inline 调用另一个共享的项目流程片段；根据本步骤的结果（成功或失败）可以通过 Pass Chain 或 Fail Chain 分别启动另一个不同的项目。

知识保持

人员变动是所有组织不可回避的事实，但 Rational BuildForge 可以帮助企业克服这一难题。BuildForge 的统一知识库自动保存了企业基础设施中的流程信息，保护了信息投资并且大大降低了流程的学习曲线。团队成员可以在 Rational BuildForge 知识库中：

- 查看当前流程的定义
- 查看流程上一次运行的结果
- 随着时间推移查看流程定义的变化
- 查看注释中关于流程为什么变化的解释原因

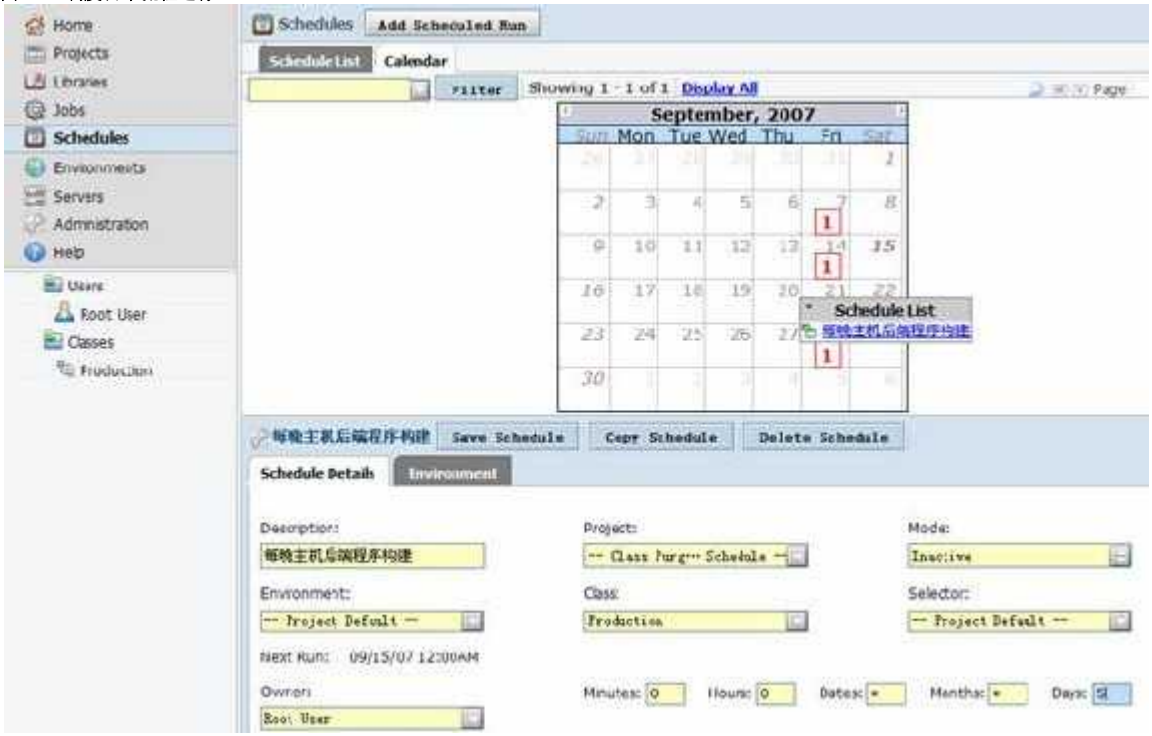
BuildForge 系统创建了一个关于流程的知识库，并且自动进行流程的归档从而流程以标准化、一致和可重复的方式执行。

通过自动化减少人为错误

Rational BuildForge 为用户提供了实施全方位构建和发布自动化系统的工具。在设计好一个流程后，用户可以轻松将其自动化，然后将该自动化流程共享给组织的其他部门或人员。通过 BuildForge 可以实现下列内容的自动化：

- 调度日常流程运行

图 12: 调度日常流程运行



- 根据一次运行或一次运行的一部分的成功或失败决定下面的活动流向
- 基于源代码的变更决定活动流向
- 基于流程运行的成败自动交付报告
- 每次运行自动生成物料清单（BOM），由此得知为什么要运行以及谁做的变更。
- 通过这些特性，每个团队都会变得更加有效率。
- 开发团队可以进行更频繁的代码构建，从而立即得到关于他们代码效果的反馈。
- QA 团队可以获得每次发布的内容以及是否可用的细节信息，这样就可以从经过自动化测试后的构建结果开始工作，评估发布中的问题。通过自动化测试可以帮助检查许多常见问题，从而将精力集中在其他更为复杂的测试上。
- 配置管理团队在日常操作方面可以花更少的时间，把更多时间用在开发流程优化上。
- IT 团队可以利用 IBM Rational BuildForge 的报告帮助评估当前系统的需要以及未来的系统需求。

集中控制分布访问

团队的沟通问题经常是阻碍发布及时性的一个关键因素。如果没有一个高效的构建和发布自动化系统，配置管理团队对于请求新构建、新测试以及新项目的响应能力会对开发流程产生不利影响。另外，与构建流程脱节的开发人员经常会因为等待构建结果而丧失效率。类似地，如果一个项目不可被重复并且没有很好地进行文档化，会很难确定产品不同版本的信息，重现客户的问题，结果重现客户问题经常变成整个问题修复过程中最花时间的部分。

Rational BuildForge 通过统一控制及基于角色的安全访问机制可以支持地域分布的开发团队。用户可以将来自许多部门的流程集中在一个统一系统中，然后可以将对这些流程的访问以及这些流程生成的信息扩展到整个组织中任何期望扩展到的范围。统一的配置管理组可以先创建并测试流程，然后将流程的使用权限授予其他相应组的成员，即使这些人员不都属于同一部门。不属于正式配置管理组的个人可以创建他们自己的流程并将其在公司范围内共享。同时管理人员拥有管理控制权，决定谁可以运行某个自动化流程、阅读某个报告或定义一个新流程。

通过 Rational BuildForge 开发人员可以启动构建项目，及时得到反馈并对新特性或缺陷修改进行测试，但开发人员只能启动那些有权限启动的项目。这种自助服务使得开发人员获得更高的生产效率，减少配置管理团队在响应这类请求方面花的时间。实际上如前所述，这种思路并不局限在构建及发布方面，用户可以将其扩展到想自动化的任何流程。

通过更频繁的测试周期提供敏捷性

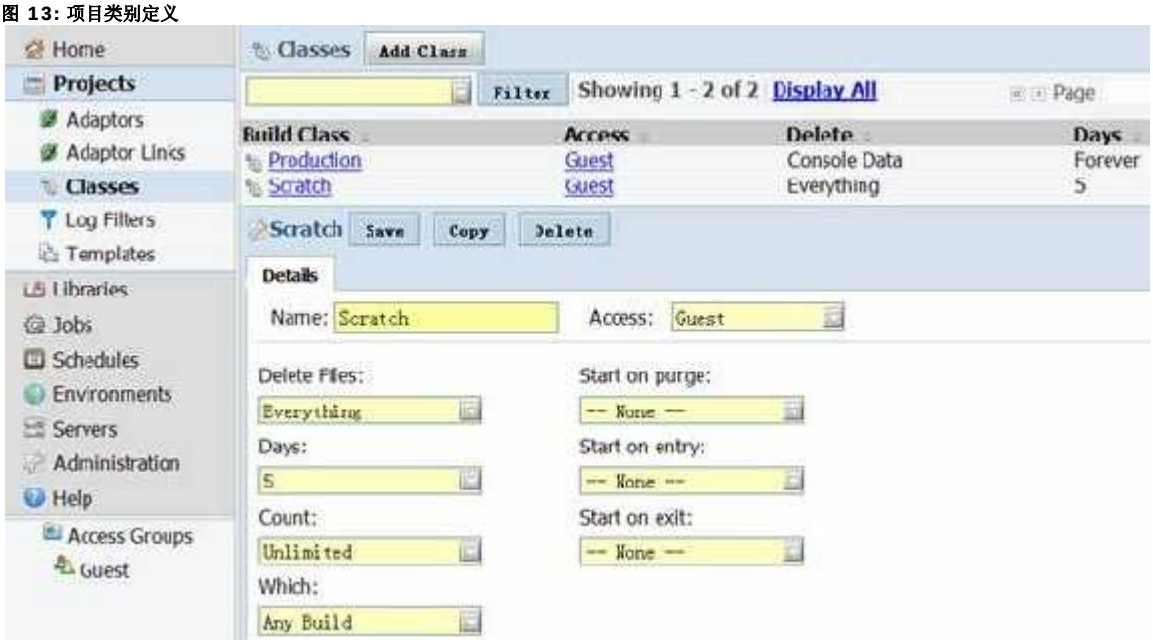
尽管自动化测试永远不能替代 QA 团队的所有工作，但 Rational BuildForge 可以将有效的自动化测试嵌入到整个构建和发布流程中，并将测试结果在整个组织范围内共享。进而 Rational BuildForge 的自文档特性可以帮助使每个测试周期更加有效，因为每次构建包含了此次构建的过程以及最终的可执行文件所包含的内容等信息。任何迭代化开发流程，例如敏捷开发及其他方法论都要求不断进行的“编码－构建－测试”活动来给开发团队和测试团队提供频繁的反馈。Rational BuildForge 提供了强大的自动化、调度以及源代码集成能力，帮助团队实现持续集成。通过 IBM Rational BuildForge 测试人员可以在许多有关产品的细节信息都已明确的基础上开始测试。

- 通过自动化测试检测到的问题可以在 QA 人员得到发布版本之前就可以反馈给开发团队，这样手工测试就可以在自动化测试通过后才开始。
- 以前碰到的问题可以很快加以重测，从而加速和改进回归测试流程，以便开发人员和测试人员较少担心引入新问题，而是将精力集中在新问题的修复上。
- 可以自动执行压力测试和负载测试。
- 通过 Rational BuildForge 的体系结构决定了它可以在所有目标机器上启动测试活动并收集测试结果。
- Rational BuildForge 定制过滤器可以对测试结果进行预处理，用详细的警告或失败列表来加强报告能力。
- 物料清单（BOM）报告可以详细报告构建中包含的缺陷修复，因此可以对每个缺陷修复进行测试或者和自动化测试结果进行逐一匹配。
- Rational BuildForge 可以与用户的源码控制系统和缺陷追踪数据库进行集成。

测试团队还可以将 IBM Rational BuildForge 用于：

- 执行冒烟测试。在每次构建后启动一个基本测试集，依据该测试集的成败来确定本次构建是否满足基本要求，是否可以继续后续测试。
- 自动将完成的构建安装在多台目标机器上。可以使用 Rational BuildForge 来配置用户的测试环境并通知测试人员这些机器已经准备好运行测试脚本了。
- 确定执行哪些测试。Rational BuildForge 系统中的项目都有不同的类别（class）标记，用户可以使用项目类别来决定执行哪些测试。下图是项目类别定义画面，在 Start on entry 可以决定一旦项目类别转为此类后就开始执行的项目，例如一旦一个项目从开发类转为测试类后就开始执行一个测试项目。

图 13: 项目类别定义



- 如果某个开发人员只想执行一个快速构建可以选择跳过测试步骤。
- 对于自动调度执行的构建，执行一个冒烟测试以确保主要部分没有问题。
- 对于一个生产构建，运行一个完整的测试集合，可能包括扩展性或负载测试。这可能需要非常大的系统资源。

以上这些特性使得在整个流程中建立更有效率的测试成为可能，因此也就将更高的品质融入到最终产品发布中。

为集成而进行的产品设计

从本质上看 IBM Rational BuildForge 实际上是一个将不同应用组合在一起以完成复杂开发任务的解决方案。借助 Rational BuildForge 用户可以轻易地将项目中的不同应用、操作系统命令、批处理文件或外壳脚本混合在一起。

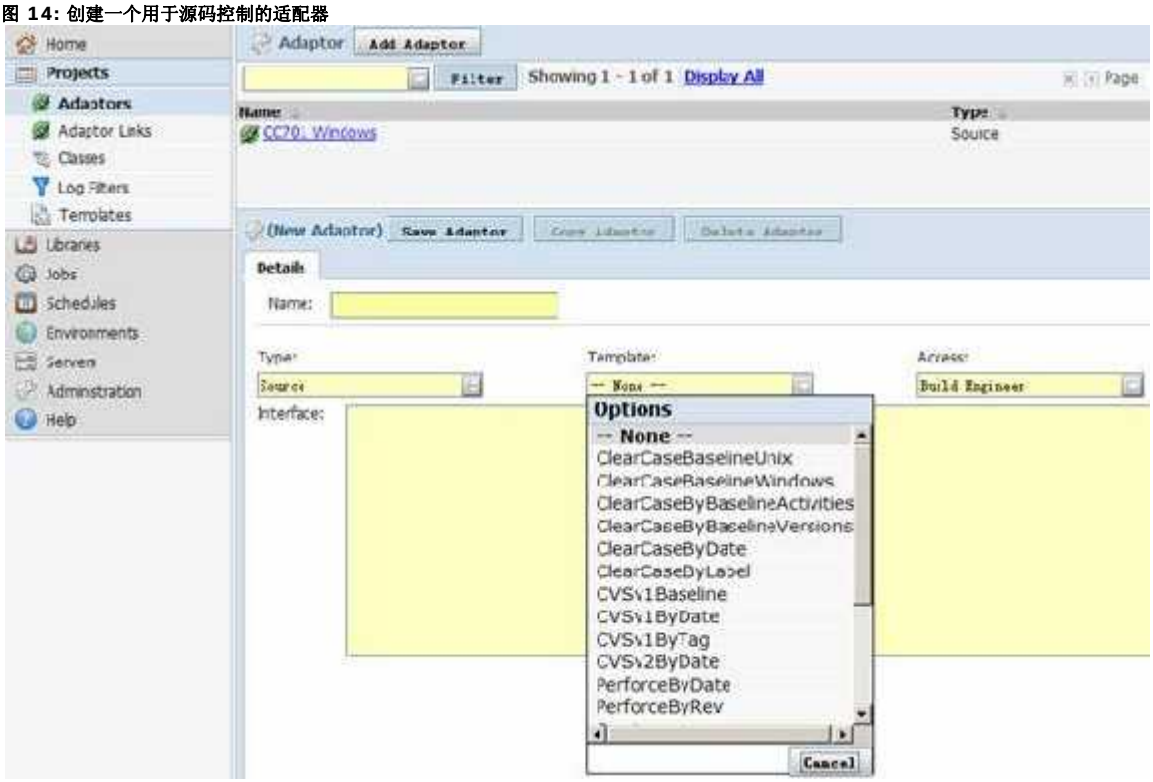
通过其独特的命令行封装能力，Rational BuildForge 可以对在目标系统上执行的每条命令在执行前创建适合该命令的必要环境。并且用户可以为与系统其他部分集成的命令定义所需的环境。

除了将不同应用链接在一起这一基本功能以外，Rational BuildForge 也可以与其他解决方案进行集成。

- 可以通过轻量级目录访问协议（LDAP）标准与现有的用户数据进行集成，这样就避免了为新的信息系统创建额外的用户登录帐号。

- IBM Rational BuildForge 为常用的源码控制工具、缺陷跟踪工具以及测试自动化系统提供了开箱即用的适配器，这样就可以通过开发活动来驱动信息的更新。例如，BuildForge 和 Rational 配置管理工具 ClearCase 的集成可以在编译前调用 ClearCase 相关功能来扫描版本库的变化，只有版本库有变化时才进行编译。下图示出了创建一个用于源码控制的适配器界面，利用随产品提供的源码控制工具适配器模板可以快速创建一个同源码控制工具（如 ClearCase）的接口（适配器），将该接口与具体的项目链接在一起后就可以在项目活动开始前调用 ClearCase 的功能（如扫描上次编译时间以来版本库内发生的变化）了。

图 14: 创建一个用于源码控制的适配器



- 通过 IBM Rational BuildForge 的 API 接口用户可以在自己的信息系统中从其他应用程序调用 BuildForge 的功能，用户可以进行细粒度的流程控制。

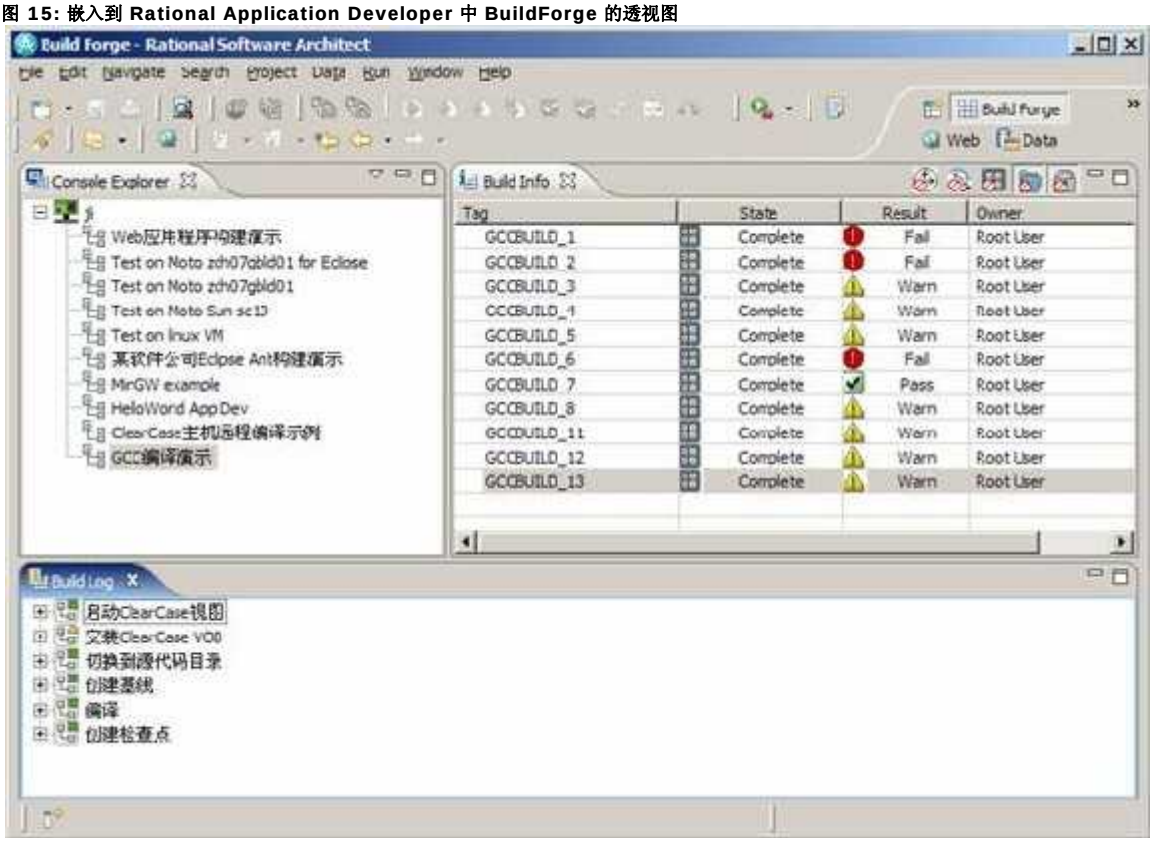
借助上面这些功能，Rational BuildForge 可以成为用户开发环境中的关键组成部分，同时成为企业信息系统中的重要成员之一。

开箱即用集成的价值：IBM Rational 软件开发工具

IBM Rational BuildForge 标准版和 IBM Rational BuildForge 企业版都可以配有开箱即用的与 IBM Rational 软件开发平台（SDP）的紧密集成，可以快速给客户带来价值。通过集成使用这些产品，公司可以管理、自动化和跟踪从需求到生产整个开发生命周期。这些集成包括：

- IBM Rational BuildForge 标准版和 IBM Rational BuildForge 企业版都可以配有开箱即用的与 IBM Rational 软件开发平台（SDP）的紧密集成，可以快速给客户带来价值。通过集成使用这些产品，公司可以管理、自动化和跟踪从需求到生产整个开发生命周期。这些集成包括：
- IBM Rational ClearQuest。Rational BuildForge 自动检测用于特定构建过程的 ClearQuest 活动，一旦一个构建成功完成，Rational BuildForge 可以自动将相关联的缺陷状态置为已解决状态，并在 ClearQuest 中将这些已解决的缺陷关联到构建记录上，这样就可以轻松在 ClearQuest 中查出某次构建所包含的缺陷修复。Rational BuildForge 还可以通过与 ClearQuest 的接口创建一个部署单元（deployment unit），用于传递给使用 IBM Tivoli Provisioning Manager 的部署工程师。
- IBM Rational Application Developer。通过在开发人员使用的 Rational Application Developer 集成开发环境中提供对构建流程的受限访问，Rational BuildForge 打破了开发与配置管理之间的隔阂，从而允许开发人员在进行小组级构建之前即可在自己使用的集成开发环境中运行预先定义好的构建流程并立即得到结果，从而达成更高质量和更及时的发布。下图是嵌入到 Rational Application Developer 中 BuildForge 的透视图。

图 15: 嵌入到 Rational Application Developer 中 BuildForge 的透视图



同第三方应用的集成

前面第 6 章产品清单中已经提到 IBM Rational BuildForge Adaptor Toolkit 允许用户将第三方软件，如源码管理工具、缺陷跟踪管理工具以及测试自动化工具与 BuildForge 集成起来，从而建立软件配置管理系统和构建环境之间无缝的链接，提高效率并且可以跟踪源代码、缺陷以及测试的变化。用于第三方的适配器包括 CVS，Perforce SCM，Borland StarTeam，Microsoft Visual SourceSafe，Subversion 以及 Bugzilla。用户也可以为满足特定要求修改这些适配器，或者创建与自己开发的工具或者与其他第三方工具集成的适配器。

Rational BuildForge Adaptor Toolkit 提供了许多集成选项来自动化并优化开发流程，包括：

- 监控源代码变化。IBM Rational BuildForge 适配器提供了持续监控第三方源码库的能力，并且当源码库发生变化时自动执行构建。系统直接监控源码变化并收集度量信息，为每次构建提供详细的源码库状态跟踪。使用 Rational BuildForge 管理控制台，可以看到每次构建中从版本检入注释到实际的文件内容都是哪里发生了变化。
- 自动化缺陷跟踪和测试。Rational BuildForge 缺陷跟踪适配器可以帮助团队跟踪缺陷并在物料清单中报告指定缺陷的状态，例如“已校验”或“已关闭”。自动报告的缺陷可以帮助用户了解某次特定构建中都修复了哪些缺陷。
- 自动化测试用例运行并通过提供的测试适配器将测试结果记入物料清单。

适配器技术将源代码变化、缺陷以及测试与特定的构建关联起来，这样加强了对构建组成部分的详细了解。IBM Rational BuildForge 可以捕获关键的构建统计信息，包括所施加的变更、构建时间和日期等并为了便于访问将这些信息集中存储在一个位置。

支持审计和遵规管理

如本文前面所讨论的，遵规是大多数开发团队正在考虑的一个重要问题。即使开发团队不对公司的财务系统负责，但开发团队所交付的产品或服务对于公司的收入而言很可能是至关重要的，同时开发团队应该准备应对审计方面的要求。

作为萨班斯法案的结果，许多公司已经意识到标准化、可重复和文档化的开发流程也是一个关键的业务最佳实践。关键是尽量减少收集相关审计数据的代价，从而减少开发团队的负担。IBM Rational BuildForge 可以捕获关键数据并提供从编码到生产全生命周期的可跟踪性，因此可以用作对应遵规和审计的证据。当使用 Rational BuildForge 来运行构建和发布流程时，系统自始至终自动跟踪用户的流程。将越多的流程放入 Rational BuildForge 系统，对于开发环境的了解也就越清晰。

系统保留了用户所有流程的版本信息，对于每个产品迭代，用户可以看到改变了什么，谁进行的修改以及为什么要进行这样的修改。并且系统为每个完成的流程运行提供了物料清单（BOM）。

对流程进行自动跟踪

即使是最简单的脚本只要在 Rational BuildForge 中运行，因为系统保存了每次项目运行的数据，它也可以作为一个有价值的遵规工具。例如一个普通的批处理文件或外壳脚本，它的作用是将文件从一个位置复制到 Web 服务器上，当在 Rational BuildForge 中运行它时它很快就远远不是一个普通脚本了。再看看下面这些例子：

- 系统可以通过电子邮件和基于 Web 的信息板进行项目运行成功或失败的报告。
- 尝试运行的命令和结果输出或者错误信息都保存在日志中。
- 系统可以调度脚本反复运行多次，保证标准流程每次都以同一方式得以执行。
- 系统可以发现可用的服务器资源来运行脚本，而不是依赖某一台特定机器。
- 系统会对脚本的执行人进行日志记录。

其他用户动作，如对脚本的修改也会被记录下来。另外， Rational BuildForge 注释（notes）可以帮助记录每个变化发生的原因。
当用户与现有工具进行集成时，Rational BuildForge 可以将多个孤立的信息源转换成随时可用的用于遵规的数据。

流程的版本化

当将一个流程加入 Rational BuildForge 后，就可以获取关于流程（项目）的许多信息了：项目中所包含的步骤、项目所需的环境变量、项目应该运行在哪台或哪些服务器上。不管在任何时候，只要用户对流程进行修改项目记录就可以得到更新；系统保留了当前版本的流程作为该流程的缺省指令集。每次在运行项目时系统都保存了运行信息的副本，即使项目定义已经发生了变化，用户仍然可以审查项目运行记录以便了解当时项目执行的确切步骤。进而系统可以重建先前的运行，按照当时的指令重新运行项目，而不用考虑已经发生的变化。这样任何加入 Rational BuildForge 的流程立即就可以变成可重复和可追踪的流程。如果流程的变化导致质量问题，用户可以返回到最近一次质量达标的状态。尽管 BuildForge 系统在自己的数据库中自动保存了流程记录，但用户可以将流程指令归档到自己的源码控制系统中。这样就可以将流程指令与用于创建产品的源码放在一起进行保存和管理。用户也可以将流程归档这一操作本身用 Rational BuildForge 加以自动化。

通过物料清单（BOM）建立自文档系统

IBM Rational BuildForge 包括一个可配置的物料清单，每次运行一个流程时 BuildForge 系统会自动生成一个物料清单用来提供详细的有关流程运行的信息，可以用物料清单来对流程的运行结果进行检查。同一产品（项目）由于使用人员角色不同流程往往执行方式也有不同，因此可以用物料清单作为记录流程的一种文档。
系统可以在物料清单中自动包含每次构建中一些感兴趣的信息。

- 通过使用 BuildForge 所提供的内置命令，用户可以在流程中加入一些任务来将一些额外的信息写入物料清单。例如，用户可以将在流程工作目录下的所有文件信息保存起来，然后检查这些文件在不同的检查点是如何变化的。下图是在 BuildForge 的示例流程的第 4 步中嵌入了 BuildForge 内部点命令 .scan baseline，其目的是记录编译前的所有文件状态，然后在第 5 步编译步骤之后，在第 6 步设立一个检查点（利用另一个 BuildForge 点命令 .scan checkpoint），检查文件的变化，例如新增了哪些文件、目录等。

图 16: BuildForge 的示例流程

- 物料清单就象一张从流程中的一个团队传到另一团队的工作说明书，没有物料清单构建就是一个神秘的黑箱，不知道里面装的是什么，也不知道结果是什么；而拥有了物料清单团队可以快速评估一次新的构建对他们意味着什么。
- 物料清单就象一张从流程中的一个团队传到另一团队的工作说明书，没有物料清单构建就是一个神秘的黑箱，不知道里面装的是什么，也不知道结果是什么；而拥有了物料清单团队可以快速评估一次新的构建对他们意味着什么。

总之，物料清单将流程以及流程端到端的运行结果都封装到一个直接可用的包中，适于审计或遵规校验。

回页首

IBM Rational BuildForge 案例研究

IBM Rational BuildForge 在短短 6 年左右时间内在全球获得了广泛应用，包括独立软件供应商、硬件厂商、电信、金融、软件游戏厂商等多类业界领先的用户。下面主要结合 IBM Rational 和某个世界领先的独立软件供应商为例分析他们是如何利用 BuildForge 解决各自面临的问题的。

回页首

IBM Rational 自身使用 BuildForge 的概况

自 2006 年 IBM 收购 BuildForge 并归入 Rational 软件部门后，Rational 迅速将 BuildForge 用于自身的研发管理流程中。在使用 BuildForge 之前 Rational ClearCase、ClearQuest 等产品线的 350 多名开发人员分布在全球几个实验室中，而开发的多个产品又涉及多种操作系统平台，发布工程组一直在寻找一个更为健壮的多平台支持、流程自动化并且可扩展的构建和发布环境来更好地利用人力资源和硬件资源，保质保进度地完成产品发布。在实施了 BuildForge 后 Rational 三个产品线已经建立起了一整套可扩展的构建和发布管理基础设施，明显改进了开发团队生产率、资源利用率和产品质量。而且在充分利用了现有开发工具和构建脚本的同时，团队增加了对他们对端到端的开发流程的了解程度，进而能不断提高其可靠性。

“IBM Rational 软件一致致力于提高自身软件开发实践从而改进我们自己的产品质量和开发效率，这就要求我们的端到端开发生命周期必须是可预计、可扩展的并且完全可以进行跟踪。BuildForge 的构建和发布管理平台是 IBM Rational ClearCase 开发团队生产业界领先的 Rational ClearCase 产品所使用系统的核心部分。BuildForge 的自动化和控制能力帮助我们以更准确的预判性来交付更高质量的产品。” —— Lee R. Nackman，副总裁，产品开发及客户支持，IBM Rational 软件

下面展开介绍一下 Rational 自身使用 BuildForge 后的经验和好处。

通过脚本开发自己搞构建自动化只能走这么远

在实施 BuildForge 之前发布工程团队通过脚本开发自动化了他们许多构建和发布流程，但是他们发现系统并不能扩展用来满足不断增加的新需求和复杂性。由于每个任务都是独立运行的，因此整个流程执行时不得不时时等待其他任务的完成来进行不同任务之间的协同。如果一些任务比预期时间晚完成，可能会导致整个项目失败。例如当夜间构建失败后通常会花一整天进行诊断和测试。同时他们发现内部开发的系统并不能充分利用超过 100 多台不同操作系统的机器。项目与具体的机器捆绑在一起，结果往往造成一些机器非常繁忙而其他的机器却闲着。进而为每个流程而专门书写的脚本使得团队成员很难共享工作，一个人不在别人很难接替他的工作。

痛定思痛——建立基于 BuildForge 的统一构建和发布管理系统

在对 BuildForge 进行评估后，项目组立即意识到可以用简单而更聪明的方式利用他们的资源。BuildForge 帮助发布工程团队通过一致的管理控制台集中管理原来分散的流程。BuildForge 无缝地与团队现有的脚本和工具（IBM Rational ClearCase、 IBM Rational ClearQuest、IBM Rational Robot 以及其他）集成在一起，因此可以快速基于现有做法进行自动化运行。现在他们可以建立流程之间的智能链接从而导演整个“编码－构建－测试－发布”周期。IBM Rational 软件开发经理 Dave Barr 说：“使用我们以前的系统，如果整个流程的某个环节出了问题，我们不得不停下来花很多时间（通常一天）来进行检查并重新开始。现在我们可以很快发现哪里出了问题并快速补救以免浪费更多时间。更重要地整个流程更加可靠和健壮，需要恢复的次数也减少了。”

事半功倍

通过 BuildForge， IBM Rational 软件团队变被动响应为积极主动，主管人员可以从他们的 BuildForge 控制板中即可有效监控项目的运行，并且可以快速指出错误发生的位置并在发生错误时及时解决错误。BuildForge“隐藏”了他们构建的所有复杂性，团队成员只要按一个按钮就可以执行相关的任务，这样团队成员不经过特定流程或项目的培训也可以更容易地与他人分担工作负载。现在他们差不多一个星期可以省出一天来放在其他任务关键性的工作上，同一批人还能给其所支持的其他团队以更多的服务。Barr 说：“BuildForge 使得我们可以轻松同时管理多个项目，我的团队从一个地方就可以得到他们需要的所有信息。另外，由于构建变得更加可靠，出现紧急问题的时候也少了”。

更加高效的构建和发布

BuildForge 帮助发布工程团队优化了硬件资源的使用，得到了更好的性能和更快的发布周期。现在他们众多的服务器已经组织成多个服务器池并有效地分布在不同的开发地点。通过使用线程选项，没有依赖关系的任务可以跨不同服务器、不同操作系统同时运行，这大大减少了一些 IBM Rational 软件产品的构建时间，从 1 天下降为小于 3 个小时。

通过更频繁的测试周期获得更好的质量

现在他们可以更快和更可靠地进行构建，团队可以用更多的周期来进行软件测试，QA 团队可以立即对 BuildForge 系统进行访问以获取最新构建包含了哪些内容，到底需要测试什么等。通过提高可靠性，IBM Rational 有了更多的测试时间来保证产品质量。

未来——支持开发人员得到更多好处

基于他们现有 BuildForge 的初步实施成功，发布工程团队计划将系统进而开放给开发团队。利用内置的基于角色的安全性，开发人员可以从他们自己的集成开发环境在授权范围内直接访问已经批准生效的生产流程。这样开发人员就能够在 IBM Rational ClearCase 版本库中检入变更内容之前或之后都可以对他们自己的工作进行校验，在影响夜间构建前提前发现和解决错误。开发人员由此可以对他们的代码更有信心，整个团队期望能得到进一步的生产率及质量的提高。

回页首

某独立软件供应商

业务挑战——不可预期的构建和发布，相互脱节的开发工具

作为一家全球领先的桌面出版软件供应商，拥有 50 个产品线、150 个产品开发组、2000 多名分布在全球的开发人员。但长期以来分离的构建和发布流程使得开发团队各自为政，形成了多个相互重复、自行开发的构建系统。150 个产品开发组的 150 个配置经理每人都使用他们自己的手工的、没有很好文档化的构建流程，结果构建过程时间长而且经常失败，发布也很难再现，开发成本相当惊人。另外，每个开发工具都有自己的信息，由于这些信息不能共享，因此无法得到关于发布内容的可见性。总是不断重头做同一件事但却缺乏端到端的构建和发布流程集成造成该软件供应商产品质量不高、不能按期发布以及发布成本高。因此他们感到他们需要一个可靠、集中管理的系统来减少成本，使得 50 个产品开发团队都统一到这个构建管理平台上来。

业务解决方案——可预期性和集成性

公司成立了一个专门的工程服务小组（ESG）来为构建和发布流程创建相关标准，他们想将所有的开发系统（源码控制、测试、缺陷修复、本地化、CD 制造等）全部连接在一起，形成那个一个紧密集成的面向服务的架构。在对比了多种解决方案，包括他们自己开发的解决方案后，由于 BuildForge 的灵活性、可扩展性和快速见效的特点，该公司最终选择了 BuildForg。

在采用了 BuildForge 解决方案后，通过一个跨不同团队、产品和平台的集中管理的系统，用户可以对构建和发布的最佳实践进行管理和自动化。BuildForge 可以相当容易地插入到现有的开发环境中，将分离的工具连接到一起从而创建贯穿整个应用开发生命周期的、透明的流程。通过实现自动化的、一致的流程，发布周期以及产品质量均得到了改进。

另外，BuildForge 确实帮助用户加速了他们的发布过程。用户想更多采用敏捷方法来改进质量，但他们知道他们以前的系统是不能支持这种想法的。在采用 BuildForge 之前 1 个月只能完成 18 次构建，在采用了 BuildForge 之后可以进行 360 次构建，提高了 20 倍。这使得他们的配置管理团队更有效率，更少的投入可以得到更多的回报。

业务结果——开发流程完全连接在了一起

- 由于开发流程被完全集成在一起，因此开发流程运作得更加平滑。
- 配置管理团队不用添加额外的资源就可以满足每个项目的需要。
- 开发、配置管理、QA 以及管理团队都可以访问集中管理的系统，了解每个发布的状态。
- 开发周期从每月 18 次构建变为每月 360 次构建，而且构建人员减少了一半。
- 配置管理团队的生产率提高了 90%。
- “没有 BuildForge 我们是不能以现在这种效率进行运作的，它每年为我们节省了数百万美金。” —— 高级计算机科学家

回页首

总结

本系列描述了 Rational BuildForge 进行构建和发布过程管理及自动化的方法，讨论了如何通过整合项目组、流程以及系统来改进软件开发效率，从而提高整个开发团队的效率，改进产品质量，更好地遵规。

本系列首先对编译和构建的差异给出了说明，然后对文中常用的名词或术语进行了描述，接下来在“当前构建及发布过程面临的挑战”对当前应用生命周期管理中，特别是构建及发布过程面临的挑战。为了更好地帮助读者理解 IBM Rational BuildForge 的实现原理及功能，特别在介绍具体产品功能前加入了有关构建流程的一些基础知识及最佳实践经验。在“IBM Rational BuildForge 简介”部分对 BuildForge 的基本功能进行了描述，之后是基于 BuildForge 的集成构建平台解决方案介绍，以及给软件开发、质量改进以及遵规带来的好处，最后是一些用户的案例研究。