金融行业 DevOps 解决方案，第二部分：基于 RTC 的统一配置管理平台

Date: 24/6/2016
Type of Submission: Article

Title: 金融行业 DevOps 解决方案，第二部分：基于 RTC 的统一配置管理平台
Subtitle:

Keywords: RTC,Build Forge,构建平台

Prefix: 无需填写
Given: 蔡钦礼，刘鹤辉，张浩
Middle: 无需填写
Family: 无需填写
Suffix: 无需填写

Job Title: 咨询顾问，资深咨询顾问，咨询顾问
Email: qinlicai@cn.ibm.com, hehuiliu@cn.ibm.com
Bio: 蔡钦礼是位于 IBM （中国）软件开发实验室软件服务部门的一名咨询顾问，致力于为国内大型银行和股份制银行提供 devops 方案的咨询服务和方案落地。
刘鹤辉，现为 IBM（中国）软件开发实验室软件服务部门的资深咨询顾问，目前主要专注于 DevOps 及物联网的咨询服务工作。
张浩，现为 IBM（中国）软件开发实验室软件服务部门的咨询顾问，目前主要专注于 DevOps 及物联网的咨询服务工作。
Company: IBM
Photo filename: 照片文件名

Abstract: 本文介绍了在整个 DevOps 交付流水线工具平台中，基于 IBM RTC 的配置管理平台。本文首先根据作者经验，总结了国内金融企业开发组织在配置管理上普遍存在的挑战，并在此基础上介绍了针对这些挑战的整体方案，以及整个方案的具体实施细节，最后总结了该方案实施后所能够带来的收益。

一、国内金融业软件开发组织在配置管理上通常面临的挑战

软件配置管理作为一种标识、组织和控制软件代码修改的技术，在保证软件版本和软件代码的正确性方面起着非常关键的作用，对于大型软件开发组织来说，尤其如此。在整个DevOps交付流水线的工具平台中，软件配置管理是其最基础的能力，也是该工具平台构建的第一步。根据本文作者数年的客户现场实施经验，我们发现对于国内的很多大型软件开发组织，特别是金融行业的很多IT组织，在配置管理方面仍然面临如下若干挑战 。

1、在并行开发模式下对源代码版本管理高效性和精确性的支持

1） 任何配置管理平台，从根本上都是对现有开发模式的支持，即在企业现有开发模式下，如何实现对源代码版本管理的高效性和精确性。正如本系列文章的第一部分所述，目前国内绝大部分的金融企业都是采用多项目并行开发的模式，而这种模式下最为突出的就是如何方便快速的创建新的开发分支，如何在不同分支间进行代码合并，交付。目前比较多的企业都还是停留在靠开发人员手工拷贝代码的方式在不同项目间进行代码的共享和合并。这种做法对于一个多项目并行开发，多团队成员协作开发的团队显然是一种沉重的负担甚至是灾难。

2） 在现有多项目并行的开发模式下，大部分企业由于缺乏有效的开发过程协作机制而往往很容易导致开发过程的代码版本错乱、代码污染等问题，例如缺乏让开发人员保存代码变更到服务器上的同时又不影响团队其他成员代码的手段。

3） 代码管理精确性的有效实现途径是开发过程各产物的可追踪性，比如可以通过需求精确的获取该需求实现的所有代码，从而能够精确地获取实现该需求的代码版本。而现有大部分的企业都缺乏从需求到代码实现之间的完整追踪关系，同时也难以实现对不同目的所作的变更进行隔离；开发过程需求、任务、缺陷与源代码变更之间缺乏追踪关系，难以进行准确版本发布和任务实现跟踪。版本发布到生产环境之后，一旦有生产事故也很难追溯定位到源码版本。

2、对不同开发平台、不同开发语言的统一配置管理支持

对一个大型金融企业来说，其显然不会只使用一种开发平台，按照应用性质和类型的不同，往往会采用多种不同的开发平台，比如主机As400、Windows、Aix、Linux等等，而由于开发语言的多样化，即便对于同一平台，也往往会采用多种不同的语言，如Java、C++、C#等，不同的平台、不同的语言，往往其涉及的技术和操作方式等都会不同。从组织级的角度来说，如果对不同的平台、不同的语言采用不同的配置管理平台和方案，虽然技术上可行，但是会加大管理的成本和复杂性，也会加大整个流水线平台集成和对接的复杂度。因此，如何通过单一或者尽量少的配置管理平台统一支持不同开发平台、不同开发语言的源代码配置管理就成了大型金融企业配置管理所面临的挑战。

3、从组织级别层面到项目级别层面对配置管理过程高效性和简单性的支持

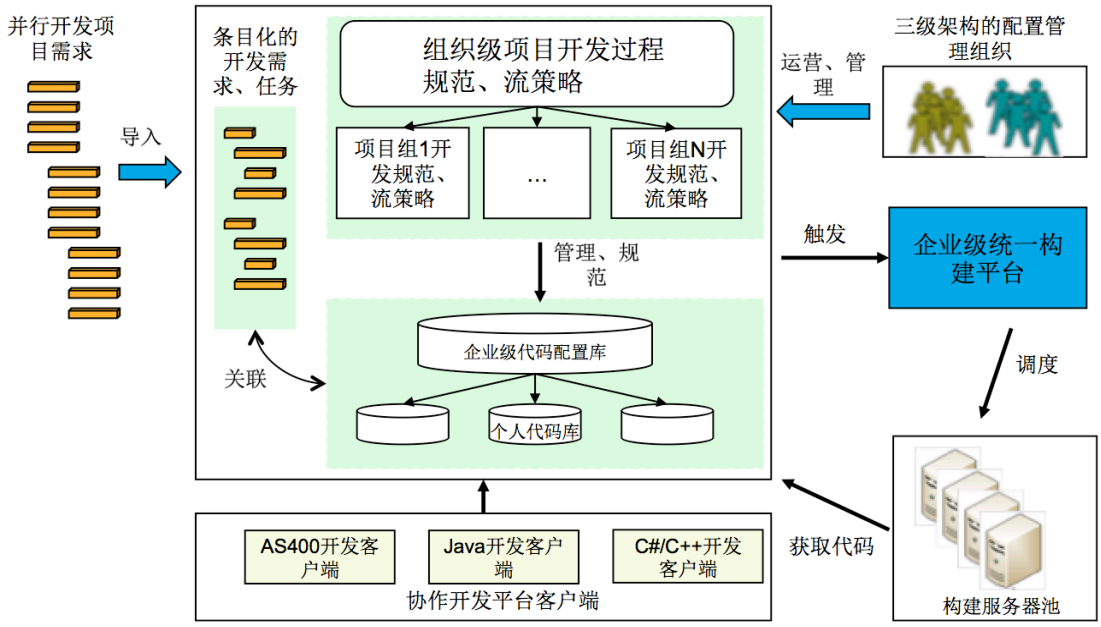
对于大型金融企业来说，为了保证所开发产品的质量，往往需要从组织层面对源代码的配置管理过程包括构建进行统一管控，而从项目团队层面来看，不同的团队，不同的平台，不同的语言，肯定会有不同的规范和需求，他们需要根据自己团队的特定进行灵活的定制。因此，在目前的大型金融企业中，往往只能将大量的配置管理及构建工作移交给项目团队，而又缺乏有效的追踪和管控，故而往往就又掉入到各开发团队各自为政，源代码版本缺乏统一管控的局面。

以上问题的存在，从DevOps流水线工具平台的源头上就导致了代码版管理较乱，效率低下，从而阻碍了整个流水线的流动。针对这些客户痛点，我们设计并实现了基于RTC的统一配置管理解决方案，从而消除了阻碍流水线流动的源头障碍。

二、方案概貌

显然，第一节中所指出的目前大部分金融企业面临的配置管理的挑战，并不单纯是技术的问题，它还包括组织保障的问题，以及流程规范的问题。要统一解决这些问题，就需要进行全方位的系统化的梳理和建设，并从组织结构、流程规范、技术平台搭建几个维度入手进行搭建。在具体工具的选择上，为了应对高度并行的项目开发模式（需要能够方便快速的创建新的开发分支），同时采用单一工具统一支持不同平台的源代码配置管理和构建管理（同时兼容主机和开发各平台），并能够兼顾组织级别的统一规范和项目级别的灵活定制，本文选用了RTC（Rational Team Concert）作为源代码配置管理工具，BF（Build Forge）作为构建管理工具，图 1 所示为本文的整体实施方案，其中关于构建方案的详细介绍将在本系列文章的第三部分中介绍。

图1. 基于RTC和BF的统一配置管理及构建整体方案



对于并行开发的项目，每一个项目的需求再分解后都会以条目化（RTC中称为工作项）的方式导入到RTC中，在RTC中，通过RTC工具本身提供的可追踪关系，即可实现需求以及开发任务与源代码的关联。在RTC中，对于源代码管理，在服务器上，除了提供通常的代码库之外，其还为每个人的工作空间提供了个人的存储库工作空间，从而实现了个人代码与整个产品及团队代码的隔离。RTC同时也提供了针对不同平台的客户端开发工具，从而实现了对异构平台、多语言的统一源代码配置管理。

为了实现组织层面对开发过程规范、流策略的统一控制，本文从整个企业层面设计了一个统一的开发过程规范以及流策略；同时为了兼顾各个项目组本身的灵活性，在满足统一规范和流策略的同时，也运行各个项目增加并定制项目组特定的开发规范和流策略。所有这些规范和流策略都会通过RTC的流程模板、开发过程前置条件、后置动作等定制到工具平台中。为了保证这些规范、流策略的顺畅执行，本文也设计了一个基于三级架构的配置管理组织用于保证这些流程规范及流策略的执行。

当源代码开发完成后，对于每一个项目，不管其包含什么程序设计语言，都可以通过一个统一的企业级统一构建平台，以相同的方式触发自动化的构建，该自动化构建会根据不同的程序语言以及不同项目所需要的不同构建环境，自动调度相应的构建服务器进行编译构建。

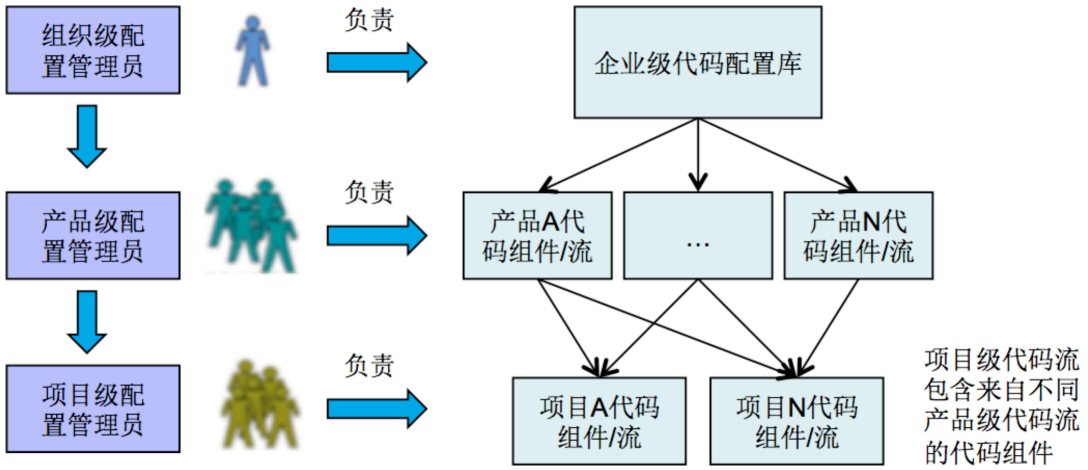
三、基于 RTC 工具的组织级配置管理方案

3.1 三级配置管理组织架构设计

任何配置管理策略的最终落地都需要有相应的配置管理组织、人员进行保证，否则，配置管理策略就只会成为一句空话。为了保证从组织级配置管理策略到项目级配置管理策略各级配置管理规范、策略的落地实施，我们制定了如图 2 所示的三级配置管理组织架构，其中，三级配置管理员分别为：组织级配置管理员，产品级配置管理员，项目级配置管理员。每一级配置管理员负责其级别所对应代码流的管理与维护，具体角色和责任分工为：

- 组织级配置管理员: 从整个企业级别维护整体的流策略和构建策略，制定并管理端到端开发过程的流策略，制定整个组织级别必须遵守的开发规范并落地到 RTC 中，维护 RTC 服务器以及用户许可证等；

图2. 三级配置管理员组织架构



- 产品级配置管理员：从产品线的角度负责管理一个开发团队所负责产品的项目区域，维护该开发团队所有产品在测试以及上线阶段的代码合并、交付， 创建并导入该开发团队所负责产品的组件和流，负责添加项目区域用户， 维护项目区域过程配置信息等；

- 项目级配置管理员：从单个项目的角度负责维护该项目开发过程的代码开发流，负责创建项目开发流；负责开发流代码与其他代码流之间的同步与代码交付。

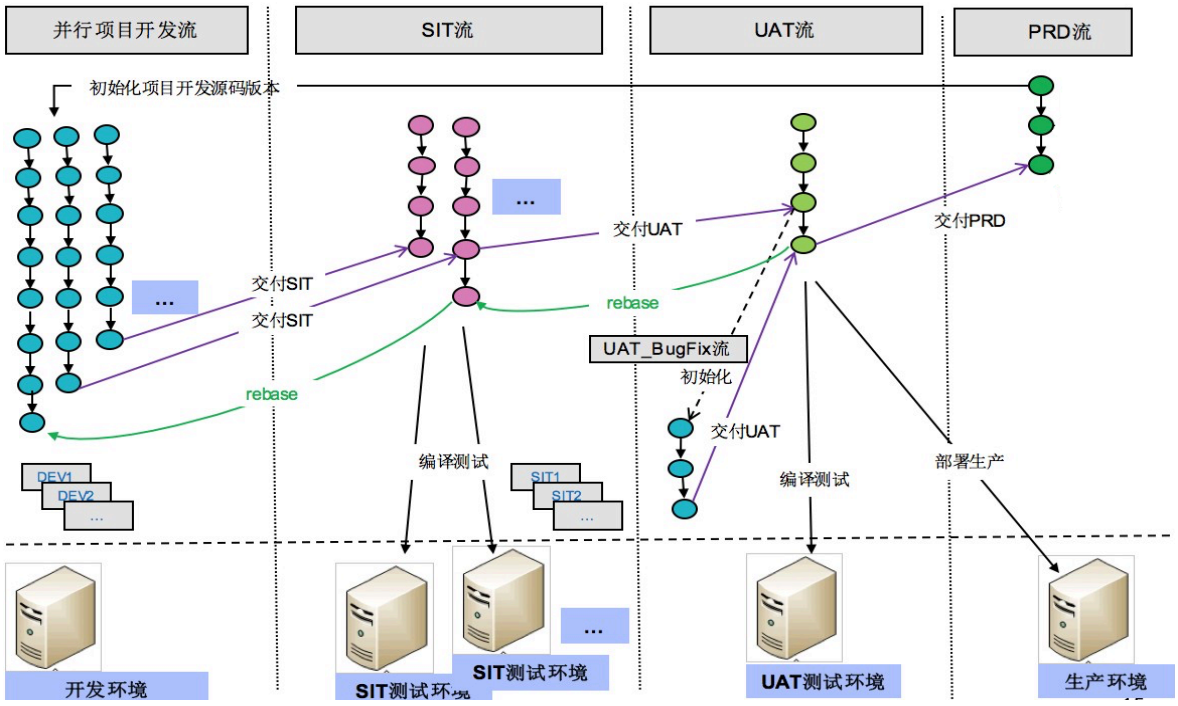
担任配置管理员角色的人员要求在岗人员比较固定，不能频繁变动，否则会增加公司的培训成本并且影响开发规范及流策略的执行效果。通过这三级的配置管理组织架构，既保证了从组织层面上整体策略及流程规范的落地执行，又通过项目级别的自主性，保证了项目层面的灵活性，即统一不死板，灵活不失控，从而保证了多项目并行开发时产品源码的版本得到有效精确地管控。

3.2 基于多项目并行开发的配置管理策略设计

正如本文第一节所述，对于目前国内的绝大部分金融企业来说，基本上都是多项目并行开发的模式，这本质上是由金融行业固有的产品性质、组织架构和需求特点决定的 。对于大部分的金融企业，其采用的主要都还是瀑布的开发模式，分别按照开发、集成测试、用户可接受测试、生产四个阶段进行代码的开发测试，不同的阶段，会有不同的环境与之对应。基于这种前提，我们定制了组织级的并行开发策略。该策略基于RTC将整个代码流分成了 5 类：分别是项目开发（DEV）流，集成测试（SIT）流，用户可接受测试（UAT）流，生产流（PRD）和用户可接受测试缺陷修复（UAT_Bugfix）流。其中，项目开发流对每个项目有一个；而用户可接受测试和生产流则会对每个产品有一个；SIT流的数量根据平台的不同会略有不同，对于开放平台，则每个产品可以只有一个，而对于主机开发，则由于不同环境的代码可能会略有差异，则可能一个产品会有多个不同的流。考虑到开发团队既要维持多项目并行，但同时在开发过程，又需要紧急修复生产环境严重缺陷的场景，我们设计了用户可接受测试缺陷修复流，用户可接受测试缺陷修复流只为紧急修复生产环境的缺陷而生成。具体的开发交付策略如图 3 所示：

1、每一个项目在立项的时候，会为其创建对应的项目开发流并从生产流上获取对应的基础版本，一般是生产流最新的快照 。

图 3. 并行开发模式下的流策略设计



2、开发人员基于项目开发流创建存储库工作空间，并每日提交可运行代码到项目开发流，项目级的持续集成基于项目开发流进行，确保每日构建、自动化测试成功。

3、当项目开发完毕，进入到集成测试阶段，则由项目级配置管理员将代码从项目开发流根据不同组件所属的产品交付到对应产品的集成测试流。

4、当集成测试构建并测试通过后，则由产品级配置管理员将代码从产品集成测试流交付到产品的用户可接受测试流。

5、由构建人员或者产品级配置管理员在用户可接受测试流上为本次发布的代码创建快照，并且执行编译构建。构建得到的构建包就是本次上线到用户可接受测试环境的目标码、配置文件和数据文件等的集合 。

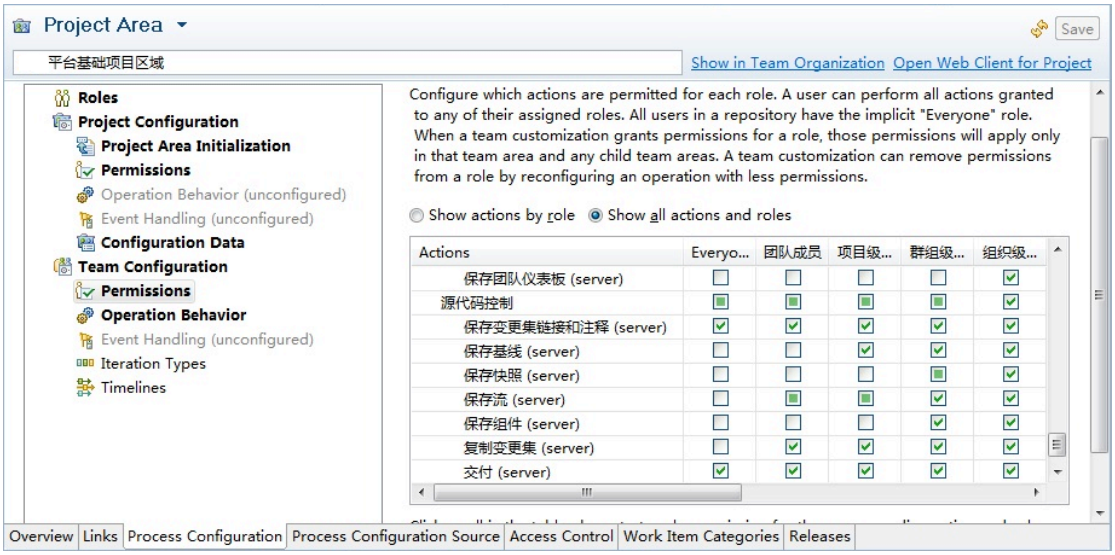
6、当用户可接受测试通过后，由产品级配置管理员将用户可接受测试流上为本次用户可接受测试创建的快照交付到生产流。此处的关键之处在于，配置管理员必须确保用户可接受测试流对于生产流只有变更集的“传出”，而不应该有“传入”，否则说明生产流上有意外修改的代码没有及时回归到用户可接受测试流，那么本次上线很可能出现覆盖之前生产环境修复缺陷的代码，导致之前出现的缺陷在本次上线后又一次出现。

7、当用户可接受测试发现有缺陷，或者生产环境发现有缺陷，需要紧急修复的，可以基于用户可接受测试流的最新快照创建一个用户可接受测试缺陷修复流，基于这个流去修复缺陷，然后编译测试上线。用户可接受测试缺陷修复流的变更集上线之后，需要及时回归（rebase）到用户可接受测试流/集成测试流。

3.3 基于RTC项目区域继承特性的开发过程模板设置

在具体实施过程中，以上所描述的配置管理策略和角色权限分工都要落地到RTC工具中，从而作为一个统一的过程配置模板。一般情况下，如果是单一团队使用RTC，则只需要在一个项目区域上进行配置。但是，当需要要支持多个团队多个项目区域的情况下，如果还把过程配置数据分别配置到各个项目区域，那么组织级的维护工作量就会变得很大。若开发过程规范有更改，那么过程配置数据也需要修改，这时候容易出现某些项目区域漏改导致过程规范配置不一致的问题。在本文的实施中，我们利用了RTC过程配置模板可被继承的特性进行设置。

图 4. 过程模板基础项目区域配置

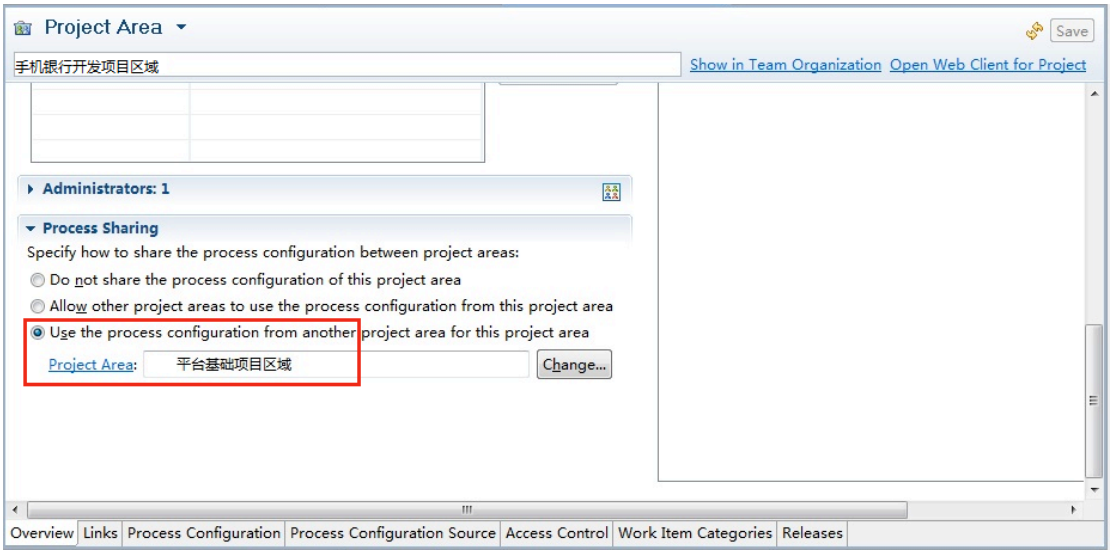


首先，由组织级核心人员根据组织级别的需求，定制一个（或者若干个）统一的过程配置模板；并将该过程模板固化到一个（或者若干个）项目区域里面，这个项目区域我们称之为“过程模板基础项目区域”，如图 4 所示的平台基础项目区域就是一个过程模板基础项目区域。

其次，在创建一个新的项目区域后，选择继承基础项目区域，并且删除当前项目区域自身的过程配置数据。具体操作为，打开项目区域的配置界面，在Overview选项卡的“Process

Sharing”选择“Use the process configuration from another project area for this project area”并选择基础项目区域，如图 5 中选择了使用“平台基础项目区域”的过程模板。

图 5. 项目区域继承设置



最后，删除当前项目区域原来的过程配置信息。选中“Process Configuration Source”选项卡，将这个页面的所有代码都删掉，然后保存本次修改。

基于项目区域之间过程配置数据的这种继承设计，组织级只要维护一个（或者若干个）过程配置文件，每一次调整或者修改，都只需要在基础项目区域修改，然后其他项目区域就会自动继承新的变更，不需要逐个修改项目区域。

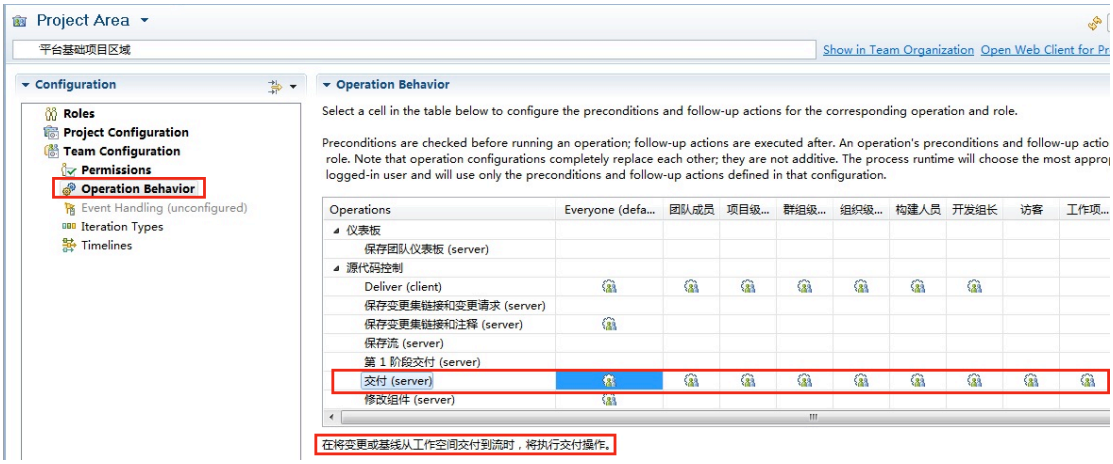
3.4 基于RTC API的扩展开发固化开发过程规范

在开发过程规范的落地实施过程中，单纯依靠RTC产品本身的内置功能并不足以支持所有配置规范的管控，所以基于RTC API的扩展开发自然变得必不可少。RTC API在项目中最常见的就是前置条件和后置动作两种类型插件的开发。其开发过程可以分成5个步骤：

1、评估业务需求，寻找控制节点。

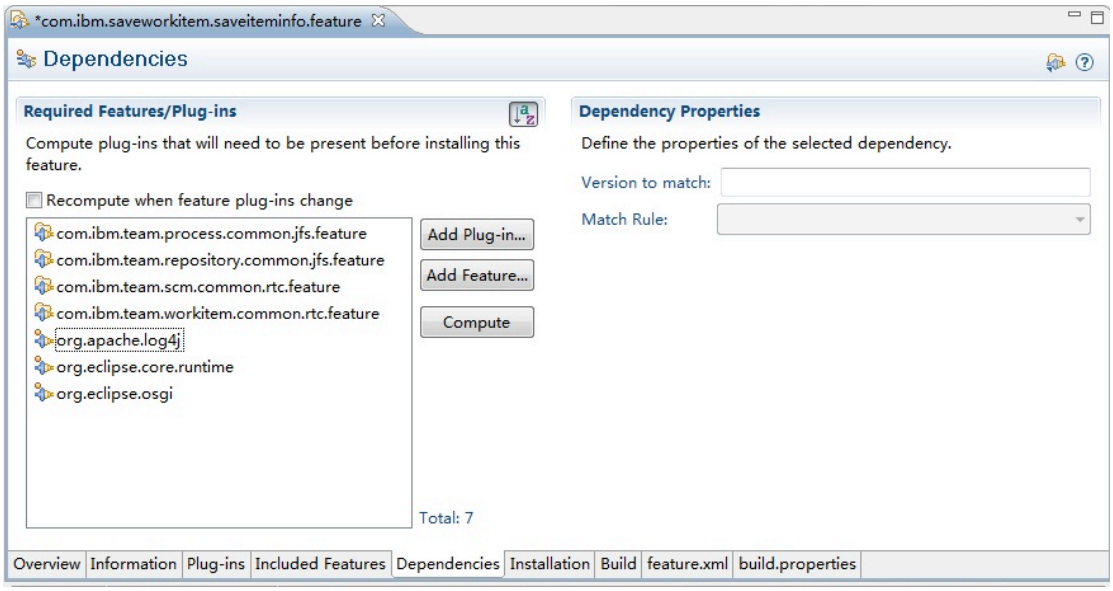
在这一个步骤中，需要根据整个端到端的开发规范，仔细分析开发过程各类角色在不同阶段所应该执行的操作以及该操作执行时的前置条件或者后置动作，然后在RTC的过程配置页面中，定位到该操作，并确定RTC的内置功能是否已经包含了该前置条件或者后置动作，如果没有包含，则需要从第2步开始，执行定制开发。例如某个客户需要执行一个这样的策略，代码需要关联到工作项，工作项上面的项目编号必须和交付变更集的开发流流名的项目编号一致才允许交付，否则不给交付。对这样的一个需求，首先通过分析RTC基础项目区域的过程配置界面上“团队配置信息”下面的“操作行为”，可以找到交付动作的控制点，如图6所示。在该交付动作中，启用前置条件，同时可以发现RTC内置的前置条件中已经包含了代码需要关联到工作的前置控制条件，但是还缺少校验项目编号的控制条件，因此，这一个控制条件需要定制开发。

图 6. 在 RTC 过程配置界面中寻找对应控制节点



2、 查找控制节点的API接口。在RTC的Eclipse客户端中，创建一个插件工程，在创建RTC插件工程的时候选择“Plug-In Project”类型，然后导入并设置该工程的依赖包（即RTC SDK），确保当前这个插件工程项目是可以在客户端调试代码的，如图7所示。

图 7. 创建插件开发工程

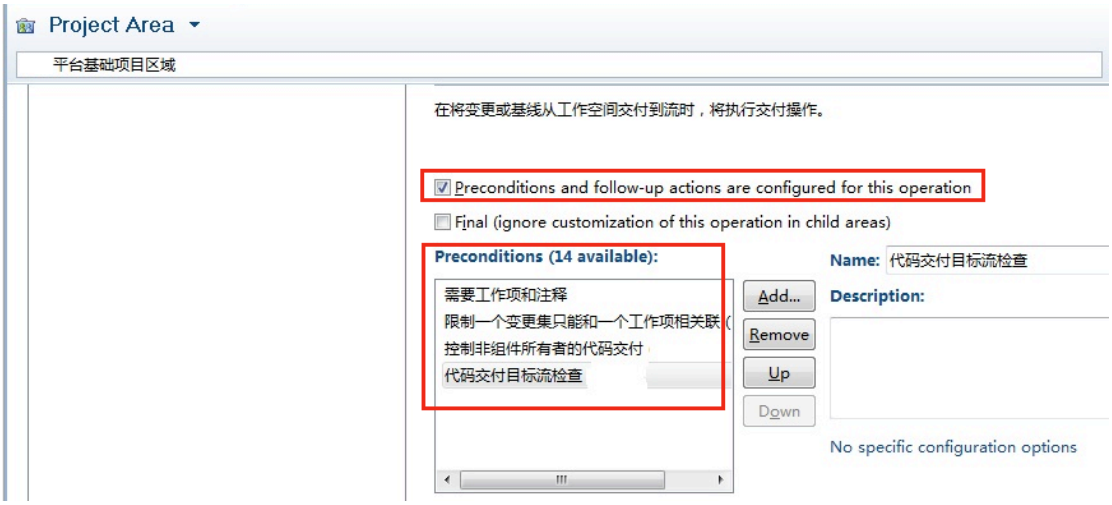


3、实现插件工程的业务管控逻辑。

4、部署RTC插件到RTC服务器。管理员登录到RTC服务器，将RTC插件包解压到RTC应用安装路径下面，该路径为<RTC应用安装路径> /server/conf/ccm/，重置应用配置并且重启应用服务器。

5、在RTC基础项目区域配置激活RTC控制插件。管理员打开项目区域过程配置界面，在“团队区域”下面找到“操作行为”的交付控制点，勾选“Preconditions and follow up actions are configured for this operation”，继而点击“Add”按钮选择刚才部署上去的RTC插件，保持项目区域配置即可，如图8所示。

图 8. 激活插件



五、实施收益总结

本文上述的方案已经在部分金融客户进行了实施，并给客户带来了积极的变化和收益。

1、影响企业文化的改变

在本文方案没有实施之前，某些客户的配置管理、开发过程和构建都没有统一的规范化，其规范和流程是“百家争鸣”，开发团队无可适从；而在实施之后，这一切都有了统一的明确的规范，井然有序，能够让开发团队更好的遵循和执行。

2、有效应对多项目并行开发的模式

使用Rational Team Concert很好的落地了多项目并行开发模式的流交付策略，让IT组织更具满足市场业务需求快速变化的能力。

3、增强开发过程管理的可追踪性和可视性

通过工作项管理，将程序开发版本与开发任务和缺陷关联起来，每次的测试和投产发布，都能通过工作项追踪到具体的程序修改，从而确保每次版本发布的可追溯性。同时，项目经理也通过工作项的管理，增强了开发过程的可视性。

参考资源 (resources)

- 通过DeveloperWorks关于软件配置管理的课程，了解关于软件配置管理的相关知识， 获取技术文档以及其他资源。
- 通过RTC产品的文档了解RTC产品信息以及操作指南。
- Jazz.net开发者网站获取RTC相关技术文档、how-to 文章、培训、下载、产品信息以及其他资源。
- RTC前置条件开发资料获取RTC插件定制开发的技术文档。