基于 UCM 的持续改进型软件配置管理①

廖丽玲，邱金来

(中国移动深圳有限公司，深圳 518048)

摘 要：先简单介绍了持续改进型软件开发的特点和配置管理需求，然后阐述了基于统一变更（UCM, Unified Change Management）配置管理的原理和方法，最后给出持续改进型软件的 UCM 应用的实现。

关键词：软件配置管理；持续改进型；统一变更（UCM, Unified Change Management）

Continued Improved Software Configuration Management Based on UCM

LIAO Li-Ling, QIU Jin-Lai

(China Mobile (ShenZhen) Ltd, Shenzhen 518048, China)

Abstract: This article describes the characteristics of software improved continually and configuration management needs, and then set out to change based on the UCM（Unified Change Management) configuration management principles and methods, and finally given the the realization of UCM for Software improved continually Configuration Management.

Key words: software configuration management; software improved continually; UCM (Unified Change Management)

1 引言

随着软件市场的日益成熟，软件开发团队的不断增大，为了提高工作效率、缩短开发时间、降低成本，大部分的开发团队都采用并行开发。为了最大限度减少并行开发中引起的混乱，大型软件项目都需要行之有效的配置管理过程，加强软件开发过程的配置管理是十分重要的。

本文先简单介绍了持续改进型软件开发的特点和配置管理需求，然后阐述了基于统一变更（UCM, Unified Change Management）配置管理的原理和方法，最后给出持续改进型软件的 UCM 应用的实现。

2 软件配置管理概述

2.1 软件配置管理定义

软件配置（Software Configuration）指一个软件产品在软件生存周期各个阶段所产生的各种形式（机器可读或人工可读）和各种版本的文档、程序及其数据的集合。

软件配置管理（SCM，Software Configuration Management）是整个软件生存周期中管理开发过程和软件产品的方法和规程，它标识、定义系统中软件项并指定基线；控制软件项的修改和发行；记录和报告软件项的状态和修改申请；保证软件项的完整性、协调性和正确性；以及控制软件项的储存、装载和交付。

2.2 软件配置管理主要活动

软件配置管理活动的目标就是为了标识变更、控制变更、确保变更正确实现并向其他有关人员报告变更，主要活动包括配置项识别、变化控制、状态记录和报告、配置审计等 4 个活动。

3 持续改进型软件配置管理

3.1 软件开发的模式

软件开发根据产品功能特性来源、开发过程、发布方式和发布时机的不同可以分成多种开发模式，主要模式分类如下：

3.1.1 新产品项目开发

软件产品功能特性来源于本次产品需求，开发过程集中，发布是根据既定项目要求有计划的整体发布。

3.1.2 软件升级项目开发

软件产品功能特性首先是继承系统原有功能，然

① 收稿时间:2011-07-20;收到修改稿时间:2011-09-02

后会对系统做框架性调整和优化，同时还会根据原系统日常业务需求同步新增功能特性。这种软件开发过程集中，软件发布是根据项目既定要求有计划的整体发布。

3.1.3 软件维护期开发

软件产品功能特性来源于在线运行的缺陷或业务新需求。软件开发过程是临时性的，没有严格的计划，大多根据市场新增需求灵活开发。软件发布方式可以分为阶段发布型和灵活模块发布型。软件发布的时机根据市场需求的不同，有些可以有计划的阶段性发布，有的则根据市场需要即时灵活发布各个变更。

3.2 持续改进型软件开发特点

持续改进型软件同时包含软件升级项目开发和软件维护期开发。一般产品投入运行后，在线运行的软件由于市场需求或软件缺陷需要进行各种变更，同一时期可能会有多个变更同时在进行，变更可大可小，变更之间的耦合可能松也可能紧，变更发布可能是有计划的阶段发布，也可能是市场要求任务结束后即时发布。同时软件运行一段时间后会进行升级优化项目，此时软件需要同时进行项目和生产维护开发，两者之间需要进行变更的同步。对于持续改进型软件开发，并行开发具有两层含义，一是多个开发人员同时修改同一软件产品的同一个文件，二是同一时刻同一个变更并行在两个软件产品开发中实现。

3.3 持续改进型软件开发的配置管理需求

由于持续改进型软件开发变更的多样性、并行开发多重性和市场要求第一的特点，使得这类软件的配置管理更为复杂。通常持续改进型软件项目和维护开发是由两个独立的团队负责，如何在不同的团队间准确且高效的跟踪变更，统一变更管理，保证系统功能正确稳定是这类软件配置管理的一个重要需求。

4 基于UCM持续改进型软件配置管理的原理

4.1UCM 基本概念

为了有效的管理、追踪和控制变更，IBM Rational 提出了一种基于活动对软件进行变更管理的方法，软件的统一变更管理（UCM）通过 Rational ClearCase、Rational ClearQuest 以及 Rational Suite 所提供的开发平台实现了贯穿整个软件开发周期的配置管理过程，是用于管理软件开发过程（包括从需求到版本发布）中所有变更管理的"最佳实践"流程。下面介绍 UCM 的几个基本概念：

4.1.1 元素

元素（Element）是 UCM 中的最小项，即纳入配置管理范围的软件配置项。

4.1.2 构件

构件（Component）是把一组相关的目录和文件元素组建在一起，并且把它们和 UCM 项目进行绑定。一个元素只能存在一个构件里，并且创建后不能移动。所有的项目必须有一个和多个构件，并且在项目间可以共享构件。

4.1.3 活动和工件

在 UCM 项目里对任何 ClearCase 管理的资产的变更都必须关联到一个活动。活动（activity）可以是在现有产品中修复一个缺陷或者新加一个增强功能。工件（artifact）可以是在开发生命周期中涉及的任何东西，例如需求文档，源代码，设计模型或者测试脚本等。

4.1.4 变更集（Change sets）

封装了所有用于实现该活动的项目工件。

4.1.5 UCM 流（Stream）

流类似于其他配置管理方法的分支，但是流比分支包含了更多的信息，包括基线和在此基线上的活动集。基线加上活动集决定流里包括元素的哪些版本。

4.1.6 UCM 基线（Baseline）

基线是 UCM 流的基础，代表了用于开始一个流和变基一个流的元素版本。

4.1.7 项目（Project）

项目为一组人在一个单一的项目上提供工作平台。它可以是一个完整的产品，一个完整项目的子系统，或者是集成一些产品形成一个套件。项目包含了一个集成流和零个或多个开发流。项目中元素、构件、活动、流和基线的关系如下图 1：

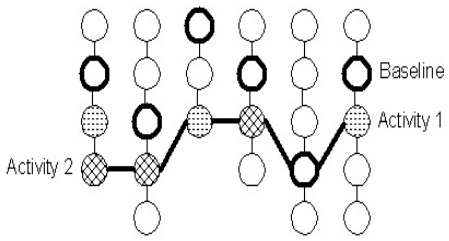

图 1 UCM 基本概念示例图

4.1.8 依赖

在使用 UCM 的过程中，当一个或多个元素的版本依赖其它一个或多个元素的版本时，用户在执行递交（即 deliver）操作会被提示发现依赖，提交操作不成功。依赖有活动依赖、变更集依赖、基线依赖等。

4.2 使用 UCM 的优点

当今有多种软件配置管理工具可供软件开发团队选择，采用基于 ClearCase 和 ClearQuest 的 UCM 和其他配置管理工具比较主要优点如下：

① 图形用户界面，简单易用；

② 项目中每个开发者拥有一个或多个开发流(Stream)，在每次项目整合之前，每个开发者有独立的工作空间 ；

③ 活动在 ClearQuest 中管理，任务清楚明了，易于项目整合 ，变更的追溯简便直接；

④ 强大的与其它工具的集成功能，可以将 UCM 集成到 Rational 的其它工具中(如建模工具 Rose、缺陷管理工具 ClearQuest、需求管理工具 RequisitePro)，也可以集成到许多流行开发工具中，如 Microsoft Visual Studio, Sybase PowerBuilder, Eclipse，JBuilder 等。

4.3 使用 UCM 进行软件配置管理基本工作流程

在具体的应用中，各个工程的需求和特点各不相同，如何应用 UCM 策略及 ClearCase 和 ClearQuest 的灵活性，定制一个最佳的配置管理工程实施方案，就成了配置管理人员所面临的首要问题。通过对 UCM 工程的定制和规划，可以有效处理配置管理中遇到的主要问题。实施 UCM 工程基本工作流程如下：

4.3.1 配置管理的需求调研

配置管理的需求，就是将软件开发过程所产成的什么工件进行控制，使用什么样的手段对其进行管理，如何在软件开发过程中方便有效地对其进行存取。

4.3.2 将软件工件映射为配置项

在配置管理系统内，所有受版本控制的文件或者目录都作为配置项或者配置元素进行管理，而 UCM 通过将相应的配置项聚集到一个 UCM 构件内，以便于对大量的软件工件进行管理。因此，首先应该将软件工件映射为配置项，将配置项组织为结构合理的 UCM 构件，然后使用 UCM 工程对构件进行统一管理。

4.3.3 流(Stream)的使用策略

根据工程需要不同，UCM 中流的应用策略很灵活。规划流主要考虑成员间成果的共享、结果的发布方式。比如各个开发人员相互影响比较多，可以使用多流方式，开发人员之间的工作空间保持一定的隔离性，同时，不同的开发人员又使用一个共享的集成流，当开发人员个体完成对应的变更并将工作成果提交到集成流之后，所有的成员可以看到最新的变化。集成人员在集成流上适时地创建基线，并根据项目的进展状况设置推荐基线，各个私有开发流通过复原(Rebase)操作，使得各个开发流的基线同集成流推荐基线保持一致，从而使各个开发成员在同一基础上进行变更活动。如果各个开发人员相互影响很少，可以考虑单流方式，单流结构意味着所有成员使用一个共享的流进行工作，开发人员的工作空间建立在一个集中一致的数据之上，任何一个成员的变更如果提交到 ClearCase 的流上以后，能够在其他成员的工作空间内得到反映。比如多个项目间产品架构相同，为了有效跟踪变更，可以把两个项目的流规划成同一个集成流下的子流，跟踪变更时可以在两个产品间进行代码自动同步合并。但如果项目间架构调整比较大，两者之间则不需要在同一集成流下，变更的跟踪只能通过 ClearQuest 活动手工同步。

4.3.4 基线（Baseline）的创建与维护策略

当完成 UCM 构件的组织与规划后，相应地，就要决定这些构件的基线创建与维护策略。具体可以包括： 如何根据 UCM 构件来指定一个 UCM 工程的基线、何时创建基线、如何命名基线、确定基线的级别、如何对基线进行测试。

4.3.5 ClearQuest 的集成策略

ClearCase 的活动仅仅能够记载活动的名称以及相关的注释，其间也没有任何的流程控制和历史记录，对于小型的 UCM 工程来说，管理起来并不复杂。但是对于需要通过大量变更活动来保证软件顺利演进的大型软件项目来说，不加控制和管理的变更活动会大大增加项目管理的难度，提高软件的风险。在这种情况下，就需要将 UCM 工程同变更管理系统 ClearQuest 相集成来解决这一问题。集成后的 UCM 工程中的活动（Activity）同 ClearQuest 中的变更请求相关联，通过 ClearQuest 对变更请求整个生命周期的管理和跟踪，达到控制整个软件变更过程的目的。

5 基于UCM持续改进型软件配置管理的实现

本节 2 个应用示例讨论内容均属于包含软件升级

项目开发和软件维护期（后续也称为生产）开发的持续改进型的软件工程，两者在配置管理的需求、配置项映射方面都是一致的，基本符合通用软件的情况，如源程序、目标代码、需求文档、设计文档等核心输出进行版本控制，其他输出物如测试案例、报告、会议纪要等只纳入存储范畴，不进行版本控制。下面根据软件开发过程中不同现状和需求来讨论相应的 UCM 流策略、基线策略和 ClearQuest 的集成策略，提供一个最佳的配置管理实践流程。

5.1 示例 1：

5.1.1 现状和需求：

① 软件维护期开发变更需要向项目同步；

② 各个开发人员开发模块独立且协作，相互影响很小；

③ 项目可以阶段性提交测试；

④ 软件维护期变更发布是有计划的阶段发布型；

⑤软件升级项目改造内容比较少，和生产维护期产品在软件架构上基本类似。

5.1.2 UCM 策略：

① UCM 流策略（如下图 2）

在这里每条流的使用方式采用的是复用流形式。该方式的最大特点是共享一个开发流的多个开发人员在检入文件时可以看到彼此的修改结果，从而在多个开发人员之间实现了集成的最大化。该流策略项目和软件维护期的变更管理可以通过 UCM 工程统一管理，软件维护期的生产开发流定期提交变更并发布到集成流，集成流发布到生产系统上线。在变更集上线成功后 rebase 到项目开发流，开发流提交变更测试，最终项目完成后项目测试流发布到集成流，集成流发布到生产系统。项目完成后再 rebase 回生产，保证生产以当前运行软件为基础进行后续活动。通过这种循环，不停的在项目和生产之间进行版本的开发和管理，有效实现了并行开发和变更的统一管理。

② 基线策略

配置管理员根据项目或软件维护期的生产计划阶段性的创建基线和发布基线。

③ ClearQuest 的集成策略

在 ClearQuest 中定制开发流程，包括开发、测试、入库、缺陷、活动同步管理，通过固化的流程保证生产和项目的统一管理。

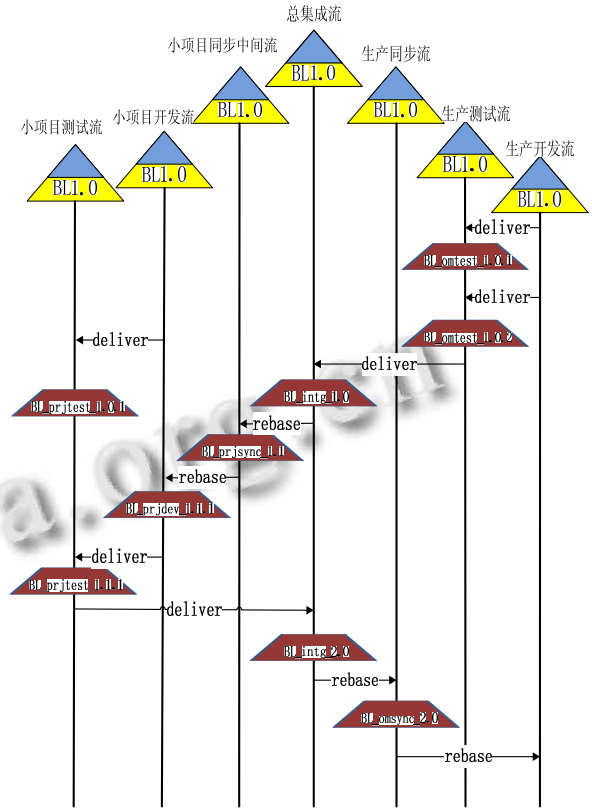

图 2 UCM 流策略示例 1

5.2 示例 2：

5.2.1 现状和需求：

① 软件维护期开发变更需要向项目同步；

② 各个开发人员开发模块独立且协作，相互影响很小；

③ 项目可以阶段性提交测试；

④ 软件维护期变更发布是适应市场需要灵活模块发布，基本上是以单个开发活动为单位进行发布上线；

⑤ 软件升级项目进行架构调整，与维护期产品在软件架构上差异很大。

5.2.3 UCM 策略：

① UCM 流策略（如下图 3）

在这里仍然采用复用流的方式，但由于软件升级项目进行架构调整，与维护期产品在软件架构上差异很大，两者如果用 delivery/rebase 代码同步需要人工干预的频率很高，管理反而混乱，因此两者在流策略上采用完全独立的模式。同时由于软件维护期的生产要求按活动灵活发布，为了避免基线依赖，采用了 2 层流方式，由测试流直接做为发布流，但发布时只取本

活动涉及的变更集。项目采用三层流模式，即集成流，测试流，开发流，由配置管理员定期提交测试流通过的代码到集成流。生产变更到项目的同步采用 Clear quest 活动人工同步的方式。项目开发测试完成后，上线生产的时候，通过将原生产流隐藏的方式，将项目的流变成新的生产的流，版本发布在新的生产的测试流上进行。

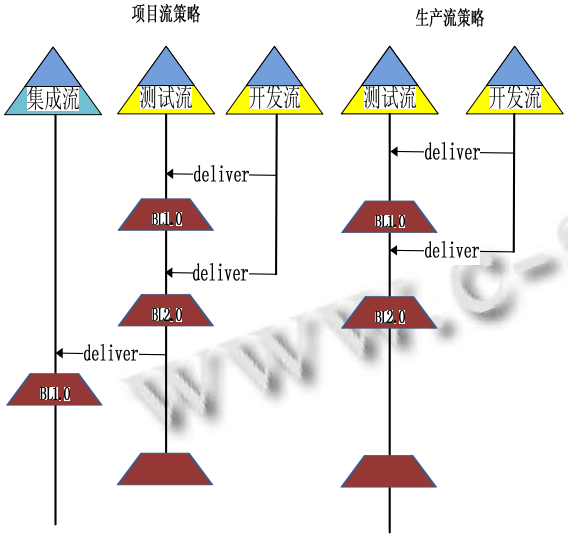

图 3　UCM 流策略示例 2

② 基线策略

在软件升级项目工程中配置管理员根据项目计划阶段性的创建基线和发布基线，在软件维护期的生产系统中根据活动的发布需求每个活动创建基线和取基线差异发布。

ClearQuest 的集成策略

在 ClearQuest 中定制开发流程，包括开发、测试、入库、缺陷、活动同步管理，通过固化的流程保证生产和项目的统一管理。

6　总结

持续改进型软件各个软件工程间如何同步管理变更是个复杂而艰巨的任务，使用 IBM Rational 提出的基于活动对软件进行统一变更管理（UCM），可以有效解决同步管理的问题。同时 UCM 是个灵活的工具，实施人员需要对 UCM 的思想进行深入的掌握，才能根据项目细节定制的对应方案和策略，达到最佳的实践。

参考文献

1 IBM.软件配置管理.IBM Rational 技术白皮书.V1.1.

(上接第 192 页)

含有交叉项，可以很好的体现语音信号的特征；CWD 是通过核函数来抑制 WVD 所存在的交叉项，故其识别性能优于 WVD。

4　结论

由于 MFCC 不能体现语音信号的动态特性，以及语音信号自身的非平稳性，提出了将时频分布与 MFCC 相结合的说话人特征提取方法。对语音信号进行时频分析得到相应的时频分布然后将时频域转换到频域再提取 MFCC+ΔMFCC 作为特征参数，通过支持向量机进行说话人识别研究。实验表明，说话人识别性能得到了很大的提高。

参考文献

1 刘丽岩.基于 MFCC 与 IMFCC 的说话人识别研究.哈尔滨:哈尔滨工程大学,2008.

2 余建潮,张瑞林.基于 MFCC 和 LPCC 的说话人识别.计算机工程与设计,2009,30(5):1189–1191.

3 Do CT, Pastor D, Goalic A. On the recognition of cochlear implant–like spectrally reduced speech with MFCC and HMM–based ASR. IEEE Trans. on Audio, Speech, and Language Processing, 2010,18(5):1065–1068.

4 张贤达,保铮.非平稳信号分析与处理.北京:国防工业出版社,1998:12–49.

5 Zhen B, Wu X, Liu Z. On the importance of components of the MFCC in speech and speaker recognition. Acta Scicentiarum Naturalum Universitis Pekinesis, 2001,37(3): 371–378.

6 Ferras M, Leung CC, Barras C, et al. Comparison of speaker adaptation methods as feature extraction for SVM-based speaker recognition. IEEE Trans. on Audio, Speech, and Language Processing, 2010,18(6):1366–1378.

7 Manikandan J, Venkataramani B. Design of a modified one-against-all SVM classifier. Proc. of the 2009 IEEE International Conference on Systems, Man, and Cybernetics. San Autonio,2009:1869–1874.