第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0058—03 文献标识码：A 中图分类号：TP311

支持需求追踪的版本控制机制

刘 玲，桑 楠，苏 芮，黄小红
(电子科技大学计算机学院嵌入式实时软件技术教研室，成都 610054)

摘 要：版本控制是增强软件可维护性的重要方法，但目前常用的版本控制机制缺乏对需求的可追踪性支持。该文提出一种支持需求追踪的版本控制机制，能够有效追踪功能需求、功能设计、代码间的版本关联关系，确保开发人员可以正确获取所需程度的需求追踪信息，有利于软件产品的一致性维护。基于该机制，设计并实现了一个支持需求追踪的版本控制工具 VCFQ，并对该工具实现中的一些关键技术进行了论述。

关键词：版本控制；需求追踪；可追踪性；可回溯性

Version Control Mechanism Supporting Requirements Tracing

LIU Ling, SANG Nan, SU Rui, HUANG Xiao-hong
(Embedded Real-time Software Technology Lab, College of Computer Science,
University of Electronic Science and Technology of China, Chengdu 610054)

【Abstract】 Version control is an important way to improve the maintainability of software. However, present version control mechanisms in common use lack the supporting of requirements tracing. This article presents a new version control mechanism, which can effectively trace the connection among function requirements, function design and codes and ensure that developers can get the necessary tracing information with precision. As a result, it is beneficial to the consistence maintenance of software products. Based on this mechanism, a new version control tool VCFQ which fully supports the tracing of requirements is designed and some critical technologies involved in the completion of this tool are discussed.

【Key words】 version control; requirements tracing; traceability; irreversibility

1 概述

在软件开发过程中，随着软件规模的增加和开发人员的流动，软件的维护工作变得日益困难。为了解决这一问题，研究人员已经提出了许多方法，如需求管理[1]、变更控制[2]等，其中一类十分重要的方法就是版本控制(version control)。版本控制对软件开发产品的不同版本进行标识和跟踪，记录每个配置项的发展历史，保证了版本间的可追踪性和开发过程的可回溯性。目前已提出的版本控制方法主要分成 2 大类：(1)支持构件的版本控制机制。该方法提出了一种以构件为粒度的版本管理方案，用于支持基于构件的软件开发过程中的资源管理和维护。(2)可变粒度的版本控制机制[3]。提出将配置项分为实体配置项、复合配置项的方法，用户可以根据需要合理组合配置项，解决不同规模的软件开发过程对软件配置项粒度有不同需求的问题。人们提出上述方法的目的是为了使版本控制机制能够更好地适应软件开发模式的变化，但是这些机制都没有提供对需求可追踪性的支持，这使得开发人员无法获取软件产品间的关联信息，增加了产品一致性维护的难度。

2 在版本控制中实现需求追踪的重要性

本文提出了一种支持需求追踪的版本控制机制。在版本控制中引入需求追踪技术[4]，它使开发人员在对需求文档、设计文档、代码进行版本控制的同时还可以双向追踪这 3 类软件产品间的版本关联关系。

需求的可追踪性(RT)指的是对某一特定需求在系统开发的整个过程中形成及演变跟踪的能力，既可进行前向跟踪，也可进行后向跟踪。追踪内容包括需求由谁提出，需求存在的原因，以及需求如何和其他信息进行联系，例如系统设计、程序代码以及用户文档等。

需求追踪是配置管理中的一个重要部分，忽略可追踪性、获取的需求追踪信息不足或建立无结构化的需求追踪信息，都会导致系统质量的降低以及项目的返工，从而增加项目成本和延误工期。虽然目前已有很多专业化的需求追踪工具被用于记录、管理追踪和进行更新的影响分析，但是这些专业工具也带来了新的问题。不同的项目参与人员在系统开发的生命周期中，由于目标和优先权不同，对可追踪性的需求程度也不同。对于项目开发人员而言，他们关心的仅仅是代码存在的目的、代码被修改的原因以及当一个功能需求或功能设计被修改时，哪些代码会受到影响等，因此并不需要随时获取用户需求在整个生命周期中完整的追踪信息。提出支持需求追踪的版本控制机制，既可以确保开发人员正确获取所需程度的需求追踪信息，又避免了使用专业化需求追踪工具额外增加的复杂性。

3 支持需求追踪的版本控制总体结构

基于提出的支持需求追踪的版本控制机制，本文设计了一个以文档为管理粒度的版本控制工具 VCFQ。该工具通过

基金项目：国家“863”计划基金资助项目“嵌入式构件的语义模型及其运行支撑技术的研究”

作者简介：刘 玲(1982－)，女，硕士研究生，主研方向：软件配置管理；桑 楠，教授；苏 芮、黄小红，硕士研究生

收稿日期：2008-05-12 **E-mail：**lu_linger0201@163.com

将软件开发过程中相关数据的集中存储，可以记录、维护每个文档的演化历史及文档间的相互关联关系。VCFQ 将原本独立的 2 个部分版本控制与需求跟踪结合在一起，一方面保证了在软件开发过程中可以根据需要随时恢复任意一个文档的任何一个版本；另一方面还实现了功能需求、功能设计、代码间的双向跟踪，使开发人员可以随时获取三者间的版本关联关系，利于软件的维护。

支持需求追踪的版本控制工具的主要结构如图 1 所示。

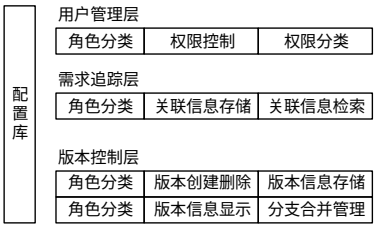

图 1　支持需求追踪的版本控制工具

支持需求追踪的版本控制工具 VCFQ 主要包括版本控制层、需求追踪层、用户管理层和配置库 4 部分。

版本控制层：针对以文档为粒度的配置项进行版本控制管理，此部分是 VCFQ 的核心部分，主要包括以下功能：

(1)版本创建及删除：完成配置项新版本的创建及删除。版本的创建分 3 类情况：

1)创建新配置项时，版本就为最初版 1.0。

2)配置项被检入时，版本可能朝纵向或横向演化。

演化规则是：

①如果配置项是被互斥写检出，版本朝纵向演化。

②如果配置项是被共享写检出，版本朝横向演化。

③如果配置项是被只读检出，版本不演化。

3)在分支中，配置项的版本朝横向演化。

(2)版本表示：给配置项的某一版本指定一个唯一的标识符。

(3)版本信息存储及显示：及时记录版本的改动信息，并详细地显示出来供其他使用者查询。

(4)版本比较：当用户想比较同一配置项的不同版本时，VCFQ 提供版本比较功能。VCFQ 可以自动列出 2 个版本之间的共同行以及第 1 个版本和第 2 个版本的不同行，以供用户观察变化。

(5)分支合并管理：在软件开发过程中可能会产生很多分支，各分支都会实现一些不同于其他分支的新特性，但有时需要将这些特性都合并到一个版本中。配置项不同版本的合并采用如下策略：不同版本配置项中新添加的内容可以有选择地添加到合并的版本中。具体实施通过版本比较来进行。

需求追踪层：在文档版本创建及变化的过程中，自动记录文档间的关联信息，同时引入需求追踪能力(联系)链，提供对功能需求、功能设计、代码三者间双向版本追踪的支持。

需求追踪能力(联系)链使开发者能追踪一个需求使用期限的全过程，即从需求源到实现的整个生存期。图 2 说明了 4 类需求追踪能力链。

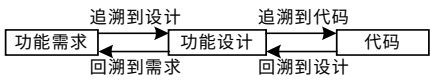

图 2　需求追踪能力链

功能需求可向后追溯到设计与代码。一旦某个功能需求变更(被删除或修改)后，通过需求追踪能力链，能够确保正确的变更传播，并将相应的设计与代码作出正确的调整。同时开发人员可以根据需求追踪能力链掌握各需求的实现情况，确保没有需求被遗漏，并在测试出错时找出最可能有问题的代码文件及版本。

代码可向前回溯到设计与需求。开发人员可通过需求追踪能力链了解每行代码存在的原因及同一代码文件不同版本与不同需求间的关联关系，利于代码的重用。在进行缺陷处理时，开发人员可明确知道修改任一代码将会影响到的功能需求与设计，以确保三者间的一致性。

用户管理层：对用户创建、删除以及系统中资源使用权限的管理。

配置库：在软件生命周期某一个阶段结束时，存放作为阶段产品而释放的、与软件开发工作有关的计算机可读信息和人工可读信息的库。

4　VCFQ 实现的关键问题

在实现 VCFQ 的过程中必须解决 2 个关键问题：(1)功能需求的凝练。通过将功能需求从需求说明文档中独立出来，对其进行版本控制，可有效获取其与下游产品功能设计、代码之间的关联关系。(2)文档间版本关联信息的自动记录。在对文档进行版本管理的过程中，系统自动记录文档间的关联信息，以便于进行功能需求、功能设计、代码间的双向追踪。

4.1　功能需求的识别

通常情况下，版本控制的最小粒度是文件，而一份需求说明文档描述了多个功能需求。当任意一个功能需求发生改变时，需求说明文档的版本就会发生变化。因此，仅仅依靠需求说明文档的版本不能直接反映出单个功能需求的版本变化情况。

所以需要引入功能需求条目文档，把功能需求从需求说明文档中独立出来，以实现对功能需求的版本管理。

功能需求条目文档内容的结构如表 1 所示。

表 1　功能需求条目文档的内容结构

名称	属性
需求说明文档名	功能需求所在的需求说明文档的名称
页范围(page range)	需求说明文档中，功能需求所在具体页的页号
行范围(line range)	功能需求内容在对应页面具体位置的行号
文档版本号	与功能需求条目文档版本相对应的需求说明文档的版本号

功能需求条目文档各名称说明如下：

页范围：需求说明文档对功能需求的描述可能集中在某一页上，也可能覆盖了多页。

行范围：如果功能需求被集中描述在同一页中，则行范围为功能需求内容起始处在该页面对应行行号与功能需求内容结尾处在该页面对应行行号；如果功能需求被描述在多页中，则行范围为对应的首页页面中，功能需求内容起始处在该页对应行行号与对应的最后一页页面中，功能需求内容结尾处在该页对应行行号。

文档版本号：当功能需求的内容发生改变时，需求说明文档将生成一个新版本，文档版本号就用于记录这个新版本需求说明文档的版本号。

功能需求条目文档记录了功能需求内容的变化过程。每一个功能需求对应一个功能需求条目文档，当需求说明文档中某一功能需求的内容发生改变时，则由修改者将其内容新的位置信息记录到对应的功能需求条目文档中。当任一开发人员需要获取某一功能需求特定版本的内容时，首先获取对应版本的功能需求条目文档，然后根据记录的需求说明文档

名和文档版本号获取需求说明文档某个具体版本，最后根据页范围和行范围即可查阅到所需的功能需求的内容。

引入功能需求条目文档，使得功能需求和需求说明文档有了各自独立的版本演化过程，为需求跟踪提供了基础。针对功能需求的下游产品——设计文档，可以采用同样的方法引入功能设计条目文档，实现对功能设计条目的版本控制。

4.2 关联关系的建立与存储

为了实现功能需求、功能设计、代码间的双向追踪，需要设计数据库表，在文档版本的建立和变化过程中，将追踪所需的文档关联信息自动记录到对应表格中。

数据库表的属性及表间关联关系如图 3 所示。

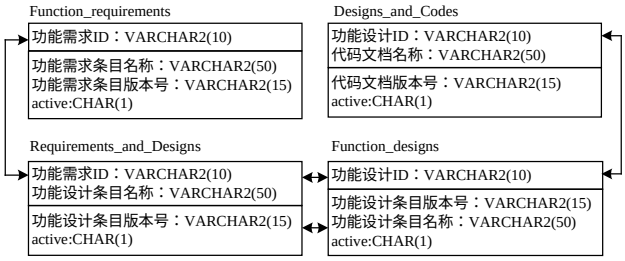

图 3　数据库表属性映射关系结构

各表说明如下：

Function_requirements 表：项目中所有功能需求的版本信息。

Requirements_and_Designs 表：功能需求与功能设计间的版本关联信息。

Function_designs 表：项目中所有功能设计的版本信息。

Designs_and_Codes 表：功能设计与代码间的版本关联信息。

系统自动记录关联信息的方式如下：

(1)功能需求的每个版本对应 Function_requirements 表中一条记录，功能设计的每个版本对应 Function_designs 表中一条记录，每个代码文档对应 Designs_and_Codes 表中一条记录。

(2)Function_requirements 表、Requirements_and_Designs 表由属性域功能需求 ID 相关联，记录了与功能需求条目文档 *X* 的版本 *n* 相关联的所有功能设计条目文档名称及其版本情况；Function_designs 表、Designs_and_Codes 表由属性域功能设计 ID 相关联，记录了与功能设计条目文档 *Y* 的版本 *n* 相关联的所有代码文档名称及其版本情况。

(3)数据库表的各条记录由命令驱动生成。

命令选项与命令参数所表达的含义如表 2 所示。

表 2　命令选项与命令参数含义

命令选项	命令参数	说明
-x	…	提交一个功能需求条目文档
-s	功能需求名：版本号	提交一个功能设计条目文档
-c	功能设计名：版本号	提交一个代码文档

命令选项说明如下：

-x：系统将在 Function_requirements 表中为该功能需求文档新建一条记录，功能需求版本号和功能需求 ID 由系统自动生成，active 域的值默认为 1。

-s：命令参数用于说明与该设计条目文档相关联的功能需求条目文档的名称和版本号。系统将分别在 Requirements_and_Designs 表、Function_designs 表中为该设计条目文档新建一条记录；Requirements_and_Designs 表中的功能需求 ID 是根据命令参数查 Function_requirements 表得到，功能设计版本号由系统自动设置为 head，表示该功能设计条目文档的当前最新版本，active 域的值默认为 1。Function_designs 表中的功能设计 ID 由系统自动生成，active 域的值默认为 1。

-c：命令参数用于说明与该代码文档相关联的功能设计条目文档的名称和版本号。系统将在 Designs_and_Codes 表中为代码文档新建一条记录，功能设计 ID 由命令参数查 Function_designs 表得到，代码文档版本号由系统自动生成，active 域的值默认为 1。

操作流程：

(1)首次提交文档：根据命令选项在数据库各表中生成对应的记录。

(2)更新文档：

1)将一个功能需求条目文档 *X* 的版本由 *n* 更新到 *n*+1 的过程如图 4 所示。

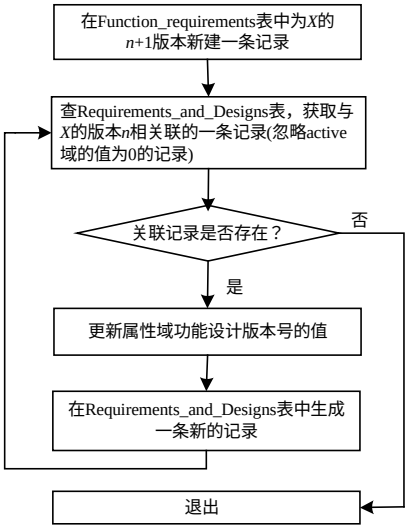

图 4　文档更新过程

①更新属性域功能设计版本号的值：若查 Requirements_and_Designs 表得到的记录中属性域功能设计条目名称为 *Y*，则将属性域功能设计版本号的值由 head 更新为 *Y* 当前最新版本的具体值。

②在 Requirements_and_Designs 表中生成一条新的记录：属性域功能需求 ID 的值设置为 *X* 的版本 *n*+1 在 Function_requirements 表中获得的 ID 号；属性域功能设计条目名称设置为 *Y*；属性域功能设计版本号设置为 head，active 域的值默认为 1。

2)将一个功能设计条目文档的版本由 *n* 更新到 *n*+1 时方法同操作 1)，被操作对象为 Function_designs 表、Designs_and_Codes 表。

(3)删除文档：将文档在表中对应记录 active 域的值修改为 0。

通过以上方法，在对文档进行版本控制的过程中，系统将自动记录文档间的相互联系。开发人员可以随时获取功能需求、功能设计、代码间的版本关联信息。如通过查询数据库表，可获得与任一功能需求的任一版本所对应的所有功能设计和代码及其版本情况。这使得任何工作产品发生改变(被删除或修改)后，借助建立的版本关联信息能够确保改变的正确传播，并将其他相应产品的关联版本作出正确的调整。

（下转第 64 页）