基于复用的软件开发过程中的配置管理

张路 李欣 梅宏 杨芙清

（北京大学计算机科学技术系，北京 100871）

摘　要 Both software reuse and configuration management have the same goal, which is to improve software quality and productivity. While software reuse mainly focuses on method issues, configuration management mainly focuses on management issues. In many ways, both technologies are complementary to each other. In this paper, software reuse and configuration management are overviewed first; then the issue of how to use configuration management for better software reuse is discussed.

关 键 词 configuration management, software reuse, software engineering environment, component

一、 软件开发过程中的配置管理

配置管理并不是一个新概念，早期的软件工程环境就已经开始考虑配置管理了。现在人们越来越认识到配置管理对于提高软件质量和软件开发过程的可靠性有着重要意义[5]。

1.1什么是配置管理

配置管理是软件配置管理的简称，它是指一套管理软件开发和软件维护以及其中各种中间软件产品的方法和规则[5] [8]，配置管理通过在特定的时刻选择软件配置（一组中间软件产品及描述），系统地控制对配置的修改，并在整个软件生命周期中维护配置的完整性和可追踪性。中间软件产品和用于创建中间软件产品的控制信息都应处于配置管理控制下[6]。

1.2配置管理对软件开发的重要意义

随着软件开发规模的不断增大，一个项目中的中间软件产品的数目也越来越多，中间软件产品之间的关系也越来越复杂，对中间软件产品的管理也越来越困难，有效的配置管理则有助于解决这一问题。现在人们逐渐认识到，配置管理是适应软件开发需求的一种非常有效和现实的技术[5]。

由于配置管理对软件开发有着重要意义，配置管理在软件开发环境中有着重要的地位。软件开发环境主要有两个重点：集成和管理。配置管理是软件开发环境中管理部分的核心，有些管理功能（比如过程管理）在最初并不属于配置管理，但随着配置管理的不断

发展，也逐渐成为了配置管理的一部分。集成本是与管理相对独立的另一部分，但人们现在也逐渐认识到，配置管理对于集成也有一定帮助（见[10]）。大多数实现的软件开发环境中也都包含配置管理的功能，比如美国海军的 ALS/N[11]，Rational 的 R1000[7]，和国家八五重点科技攻关项目 JBII[12]。

1.3配置管理的主要活动

从配置管理的定义中可以看出，配置管理中的活动主要包括：识别配置、变化控制、状态记录和报告以及审计。识别配置是指找出需要管理的中间产品，使其处于配置管理的控制之下，并维护它们之间的相关关系，一般来说，这些中间产品主要是程序和文档；变化控制是指记录变化的有关信息（包括变化的内容、原因和实现者等），用以保障软件产品的质量；状态记录和报告是指通过记录各个配置的变化状态，达到记录和报告整个软件的变化过程的目的；审计是指利用配置记录验证软件达到了预期的要求。

1.4配置管理工具的主要功能

配置管理工具基本上都是围绕上述四种主要活动的，有关这四方面的功能也是配置管理的基本功能，许多配置管理工具还提供一些基于基本功能的软件开发支持，过程管理也逐渐成为配置管理的一项功能。配置管理工具的主要功能如下：

- **配置支持**

配置是一组有共同目的的中间软件产品，其中每一个中间软件产品称为一个配置项。配置管理支持用户建立配置项之间的各种关系，并对这些关系加以维护。维护这些关系有助于完成某些特定任务（比如 Build）和标识某一变化对整个系统开发的影响。

- **版本控制**

版本控制是配置管理的基本要求，它可以保证在任何时刻恢复任何一个配置项的任何一个版本。版本控制还记录了每个配置项的发展历史，这样就保证了版本之间的可追踪性，也为查找错误提供了帮助。版本控制也是支持并行开发的基础。

- **变化控制**

变化控制是指在整个软件生命周期中控制对软件的变化。变化控制系统记录每次变化的相关信息（变化的原因，变化的实施者以及变化的内容等）。查看这些记录信息，有助于追踪出现的各种问题。记录正在执行的变化的信息，有助于做出正确的管理决策。

- **Build 支持**

软件系统往往由许多配置项构成，建立整个系统是个复杂和费时的过程，配置管理工具可以记录和追踪每个配置项信息，帮助用户自动和快速地建立系统。和版本控制结合

在一起，可以有效地支持同时开发系统的多个版本。

- **过程支持**

过程详细描述了各种人员在整个软件生命周期中如何使用整个系统，过程控制可以保证每一步都按照正确的顺序由合适的人员实施。过程控制本来是软件开发环境中一个独立的部分，现在配置管理也开始提供这部分功能。目前的配置管理工具对过程的支持还很不够，而且支持方式差别还很大。许多管理只是提供一个预先定义好的生命周期模型，并保证开发的每一步都按照这个模型规定的进行。

- **团队支持**

团队支持是指对多个开发人员同时开发一个软件系统的支持。由于大多数软件系统都需要多个开发人员参与，有效的团队支持对开发人员来说是很有用的。团队支持主要包括：工作区管理、并行开发管理和远程开发管理，某些配置管理工具还包括对开发人员交互的支持，团队支持的基础是版本控制和版本合并。工作区管理是指为每个开发人员提供独立的工作区，开发人员可以互不干扰地进行工作，也可以选择某个时机向其它开发人员提供自己的最新修改结果或接受其它开发人员的修改结果；并行开发管理是指多个开发人员同时进行的修改可以进行合并，并行开发管理可以尽可能地自动解决合并中可能出现的冲突；远程开发管理实际上是并行开发管理的特例，是指在广域网上并行开发的管理，许多适合于局域网的方法可能不适合广域网。

- **报告/查询**

配置管理可以向用户提供配置库的各种查询信息，主要包括：依赖关系报告、变化影响报告、Build 报告、配置项状态报告、版本差异报告、历史报告、访问控制报告、冲突检测报告。实际上，在许多配置管理工具中此项功能是分散在各种相应的功能中的。

- **审计控制**

配置管理通过审计控制来验证配置管理过程，以保证配置库中所有配置项的完整性。简单的审计控制是记录配置管理工具执行的所有命令，复杂的审计控制还包括记录每个配置项的状态变化。

- **其它功能**

除了以上的主要功能外，配置管理还可以提供权限控制、人员管理和配置库管理等功能，这些功能主要是为配置管理实现以上功能提供保障。

二、 基于复用的软件开发

随着软件规模的不断扩大，人们对软件生产效率和软件质量的要求越来越高，长期以来，研究人员一直致力于提高软件生产率和软件质量。目前，人们逐渐认识到，软件复用是解决上述问题的现实可行的途径[3]。

2.1 基于复用的软件开发模型

2.1.1 复用的基本思想

软件复用的思想最早是在 1968 年 NATO 的软件过程会议上提出的，在过去的几十年里，软件复用得到了越来越广泛的应用。软件复用的目的是在软件开发中避免重复劳动，按照[9]，软件复用是指重复使用“为了复用目的而设计的软件”的过程。复用的对象多种多样，虽然人们对此的看法并不一致，但普遍都认为，复用的对象应包括软件开发过程中所需的和开发出来的各种资源[3]。

基于复用的软件开发主要分为两个方面：开发可复用构件和利用可复用构件进行开发。开发可复用构件是指将过去的软件产品改造成或直接开发出可复用的构件，这些构件应具有较高的质量和较好的可复用价值；利用可复用构件进行开发是指在已有的可复用构件的基础上构建应用系统，此时应保证复用已有构件比重新开发更有好处。

2.1.2 开发可复用构件

大多数软件人员进行软件开发时并没有考虑到复用，他们生产出的产品可复用性很差。例如，大量的程序只能运行在特定环境中，程序模块存在着多种多样的调用关系，没有独立性和封装性，没有标准的接口，使得以后的项目开发很难将其复用。因此，基于复用的软件开发的重要一个方面，就是将复用性较差的软件产品改造成具有较高可复用性的构件。

领域工程在构件开发中有着重要的地位。领域工程的主要思想是对应用于某一领域的已经存在的多个系统进行分析，通过相似系统之间的比较，寻找到该领域中存在的实体、关系和操作[3]，并将它们进行抽象，最终改造为可复用的构件。构件开发出来后必须对构件进行测试，以保证构件的可复用性。

为了方便复用者找到需要的构件，对构件要进行分类，并将其存入构件库。构件的分类和构件的提取是密切相关的，这部分内容将在构件提取中讨论。

2.1.3 利用可复用构件进行开发

基于复用的软件开发者与传统的软件开发者的工作有很大的不同，他们不必一切从零做起，如不必编写所有的程序代码。他们的主要任务是使用已存在构件库中的构件构造满足用户需求的软件系统。用构件进行开发包括下述环节：

- **构件提取**

可复用构件的数量可能是庞大的，包括应用于不同领域(如数据库管理和工程计算等)的，完成不同功能的构件。为了方便复用者查找到所需的构件，必须对构件进行有效的组织和管理，不同的构件库系统提供了不同的分类和查询策略。好的分类和查询策略有助于提高查询的质量。基于关键词的分类和查询策略是一种最常见的分类和查询策略；REBOOT系统[4]采用了一种基于刻面的分类方法，根据构件的抽象层次，功能，功能对象，环境依赖(不同平台)对构件进行描述和分类，在查询和提取构件时，复用者把对构件的要求转化为对刻面的描述，从构件库中查询和提取构件；JBIII中的构件库系统（见[1]）采用了以刻面分类为主的多种分类和查询策略，这样可以结合各种策略的优点，提高查询的质量。另外，由于用户需求的不确定性，可能造成查询质量降低，因此构造库系统还可以提供模糊查询机制。

- **构件的适应性修改**

一般来说，提取出来的构件很难完全达到复用者的要求，复用者需要对这两者之间的差距进行评估，并对构件做一定的适应性修改，使之达到要求。构件的适应性修改包括三个方面：选择、修改和集成[3]，其中修改是最重要的一个方面。复用的基本要求是复用构件的代价要小于重新开发的代价，如何减少构件修改的代价对复用是非常重要的。

- **构件组装**

找到所需的构件后，就要对构件进行组装，构件的组装并不一定要求将构件直接组装成应用系统，可以先将小粒度的构件组装大粒度的构件，最后再组装成整个系统。

2.2 软件开发环境对软件复用的支持

上面描述了基于复用的软件开发模型。实际上，整个基于复用的软件开发过程都离不开软件开发环境的支持。软件开发环境将各种复用工具(如构件提取、构件组装、构件库的管理等)和开发工具集成于一个系统，使各种工具能互相调用，协调工作；软件开发环境中所提供的通讯与并发控制加强了复用者之间的协调；软件开发环境为复用者提供了简单、一致和方便的界面。

配置管理是软件工程环境的热点，在其中占有重要位置。同时，在基于复用的软件开发过程中，配置管理也起了重要作用。本文的第三部分将对此进行详细讨论。

三、 基于复用的软件开发对配置管理的要求

[2]已经指出，软件复用对配置管理提出了新的需求：配置管理应记录项目中正在使用的构件与相应的在构件库中的构件的追踪关系；针对可复用构件，配置管理应记录更多的信息。实际上，配置管理和软件复用有着更紧密的关系。复用使得软件开发分为两个阶

段：可复用构件的开发和利用可复用构件进行开发，配置管理在这两个阶段都能起到重要作用。

3.1开发可复用构件时的配置管理

可复用构件的开发也是一种软件开发，配置管理对软件开发的作用也可以在其中体现出来。提高可复用性是开发可复用构件的一个重要目标，目前主要从技术的角度研究如何提高可复用性，实际上，好的管理也对提高可复用性有着重要意义。构件的质量是构件可复用性的一个重要方面，构件使用者对构件质量的要求比对普通软件质量的要求要高。配置管理可以有效地提高构件的质量，从而提高构件的可复用性。另外，开发可复用构件时也可能会复用已有的构件，有关这方面的讨论在下面进行。

3.2利用可复用构件进行软件开发时的配置管理

利用构件进行软件开发可以分为三个方面：构件提取、构件组装和构件的适应性修改[3]。在构件提取方面，配置管理可以提高构件提取的质量；在构件组装方面，配置管理有助于在提取的同时进行组装；在构件的适应性修改方面，配置管理可以控制构件的变化，保证构件的适应性修改正确完成。

3.2.1 构件提取时的配置管理

为了较好地支持利用可复用构件进行软件开发，需要在构件库中保存大量的构件，好的构件库组织和管理可以提高构件提取的质量，配置管理对构件库的组织和管理有很大的帮助。

一个构件在构件库中往往需要保存多个版本，不同版本之间的差异可能是使用平台不同，也可能是功能或使用方式的差异。许多构件库系统都记录构件间的版本关系，比如JBIII系统中的构件库（见[1]）。利用配置管理中的版本控制，可以更好地维护构件间的版本关系。配置管理采用增量存储，可以减少这些构件在构件库中所占的空间，从而提高构件库的容量。多个构件可能会包含相同的成分，比如有两个构件，每个构件都包含多个类，其中有些类是这两个构件共有的。利用配置管理，这些类就可以只保存一份，这样可以减少冗余，保证一致性。

由于构件库使用者对构件库的查询机制不够熟悉或者难以准确表达查询要求，有时所需构件在构件库中存在，但不能查出来，或者查出来许多显然不需要的构件，这样就会影响开发的效率。构件库中的构件大多数是领域相关的，因此在开发某一个项目时，构件库中许多构件是不可能在该项目中复用的，有可能在该项目中复用的构件主要是通用构件和与该项目所处领域相关的构件。如果将这些构件的查询信息以配置的形式记录下来，使

得开发人员对构件的查询时可以仅在这些构件中进行，构件提取的质量会有较大提高。在构件库中寻找有可能在某一特定项目中复用的构件，可以由对构件库和该项目都比较熟悉的开发人员来完成。如果把这一项工作也看作是构件提取的话，由于提取者对构件库和该项目都比较熟悉，提取的质量也会比较高。

3.2.2 构件组装时的配置管理

在进行构件组装时，组装的结果可能仍是可复用的构件，这些组装出来的构件也应该保存在构件库中，供以后使用。由于这些构件的组成部分都已存在于构件库中，再在构件库中存储这些构件不是个好办法。因此，构件库应该在不直接存储组合构件的情况下，提供查询组合构件的功能。此时，由于不能穷举出构件库中所有的构件组合，必须采取一些实际可行的办法使得要查询的构件组合的数目降下来[3]。从管理的角度看，可以把可能的构件组合以配置的形式记录下来，供使用者查询，甚至可以在构件库的使用过程逐步将新的构件组合添加到构件库中。

3.2.3 构件适应性修改时的配置管理

构件的使用者和往往不是构件的开发者，对构件不够熟悉，构件的使用者在对构件进行修改时，可能为进行很小的修改付出巨大的工作量；而且由于修改不当，还可能引入错误，造成构件复用的失败。配置管理有助于减少构件修改的工作量和保证构件修改正确完成。

在对构件进行修改的过程中，构件使用者往往不能一次找到正确的修改办法，因此需要尝试多种修改办法。利用配置管理中的版本控制，记录构件修改过程中的多个版本，可以避免多次尝试之间的相互影响，也可以尽可能地保存以前有用的工作，减少重复劳动，从而减少构件修改的工作量。

一个构件可能包含多个部分，对一个部分的修改可能会对其它部分造成影响。利用配置管理中的配置支持和变化控制，还可以在构件修改过程中维护各个部分的一致性。配置管理可以记录每次修改对整个构件的影响，有助于使修改局部化，减少修改造成的构件各个部分的相互影响，指导复用者正确完成构件的适应性修改。

四、 结束语

本文在介绍软件复用和配置管理的基础上，探讨了这两种技术的关系及其结合的方式。软件复用是目前普遍看好的一种技术，在基于软件复用的软件开发过程中提供配置管理的支持有助于更好的复用。JBIII系统中已实现了一个构件库系统（见[1]），以支持软件

复用，在下一步的工作中正在考虑增加配置管理对复用的支持。

主要参考文献

[1] Li Keqin, Guo Lifeng, Mei Hong, Yang Fuqing, An Overview of JB(Jade Bird) Component Library System JBCL, Technology of Object-Oriented Languages and Systems Tools 24, 1997

[2] NATO, NATO Standard for Software Reuse Procedures, Vol. 3 of 3 volumes, NATO contact number CO-5957-ADA, 1991.

[3] H. Mili, F.Mili, A. Mili, Reusing Software: Issues and Research Directions, IEEE Transaction on Software Engineering, vol. 21, No. 6, June 1995.

[4] Jean-Marc MOREL & Jean FAGET. “The REBOOT Environment” BULL S.A. Rue Jean JAURES, F-78340 LES-CLAYES-SOUS-BOIS, FRANCE

[5] C. Burrows, G. W. George, S. Dart, Configuration Management, Ovum Ltd., 1996.

[6] M.C. Paulk, etcl, Capability Maturity Model for Software, Version 1.1 (CMU/SEI-93-TR-24), Software Engineering Institute, Feb. 1993.

[7] Rational R1000 Development System Reference Manuals, 1987.

[8] STSC, Software Configuration Management Technology Report, Software Technology Support Center, Sept. 1994.

[9] W. Tracz, Confessions of a Used Program Salesman – Institutionalizing Software Reuse, Addison-Wesley Publishing Co., New York, NY, April 1995

[10]K. C. Wallnau, Issues and Techniques of CASE Integration with Configuration Management, (CMU/SEI-92-TR-5), Software Engineering Institute, Mar. 1992.

[11]Weiderman, Nelson, etc, Evaluation of Ada Environment, (CMU/SEI-87-TR-1), Software Engineering Institute, Mar. 1987.

[12] 杨芙清，邵维忠，梅宏，面向对象 CASE 环境 JBII型系统的设计和实现，中国科学，No. 5，1995.