文 章 编 号: 1000-1220(2002)09-1114-04

构件库中构件系统的模型和实现

任洪敏　王渊峰　钱乐秋

(复旦大学 计算机科学技术系, 上海 200433)

摘　要: 构件系统是具有某种关联的构件和子构件系统的集合. 构件库管理系统支持和实现构件系统具有重要的意义. 论文描述了一个构件系统的逻辑结构、物理结构和其面向对象的管理技术. 其主要特点是: (1) 构件系统的类型机制; (2) 丰富的、支持共享的构件和构件系统信息; (3) 构件、构件系统间关系的表达和存储. 因此, 利用该构件系统能够简单和强有力地表达构件库的体系结构, 全面支持软件开发生命周期各个阶段构件和大粒度构件系统的复用和构件库的其它功能, 如领域学习功能.

关 键 词: 构件库; 构件系统; 软件复用; 面向对象

中图分类号: TP311　　**文献标识码**: A

软件复用技术和其形式之一的"构件—构架"技术的研究是当今软件工程学科的重点研究方向之一, 它能够提高软件生产率和软件质量, 降低软件开发成本[1]. 软件复用需要集成 CASE 环境的支持, 缺乏复用的支持环境是阻碍软件复用的极大因素[2]. 支持复用的 CASE 环境的核心是构件库及其管理系统, 国内外的众多学者和许多著名的公司, 如贝尔实验室、AT&T 公司、北大的"青鸟系统", 皆于此做了许多研究和开发[3, 4, 5, 9, 10].

但常见的构件库管理系统存在以下不足: (1) 构件库存储的构件相关信息不能够有效地满足构件库的复用支撑功能和其它功能, 如领域学习功能[6]. 构件库是个宝贵和昂贵的资源, 必须存储构件的丰富属性, 如词汇表和复用示例, 提供复用的应用语境, 平滑学习曲线, 支持领域学习和编程风格学习等功能[4]. (2) 构件库逻辑结构和存储结构是线性结构, 存储的基本元素是构件, 在此基础上建立复杂的索引分类机制和检索方法, 其不能够把经常协同使用以完成某个功能的构件组织成粒度更大的可复用资源, 不能够很好的揭示和表达构件库的体系结构[1]; (3) 构件类型单一, 不能很好地表达完成同一功能的不同软件生命周期的构件间的相互追踪, 不能有效支持软件开发生命周期各个阶段构件复用. 事实上, 从需求分析到软件测试各个阶段的工作产品皆能成为可复用构件.

在我们研制构件库管理系统的时候, 借鉴 Ivar Jacobson 等提出的构件系统的思想[1], 对其进行丰富和发展, 从而克服了上面提出的常见构件库管理系统的不足. 论文详细介绍了构件系统的逻辑结构, 然后给出了其物理结构, 最后介绍了其面向对象的实现和结论.

1　构件系统的逻辑结构

Ivar Jacobson 等提出了构件系统的概念, 强调运用构件系统对构件进行组织, 能够表达构件间的逻辑关系和构件库的体系结构, 能够提高软件复用程度. 但他们提出的构件系统模型亦存在着很多不足, 如要求构件系统的正交划分、构件或构件系统间的关系简单单一、构件或构件系统之间不能够共享信息、包装复杂等[1]. 我们对其进行了大的改进和扩充, 丰富了构件系统的语义, 加强了构件系统的功能, 建立了一个完善的构件系统逻辑模型. 该逻辑模型给构件库管理系统的开发人员、管理人员和构件库的使用人员提供一个稳定的和简单的逻辑视图, 便于他们运用构件系统提供的功能和规范.

1.1　构件、构件系统的概念

构件库中不仅存储构件, 还应存储构件系统, 构件和构件系统在本系统中的概念如下:

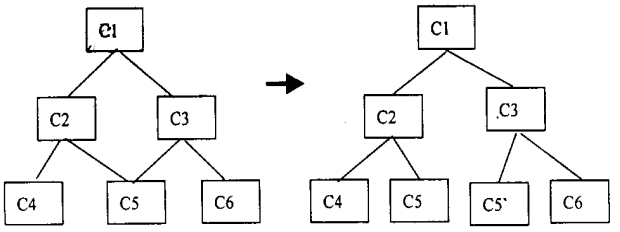
1.1.1　构件　构件是用于复用的各个软件生命阶段的工作产品(workproducts), 包括需求构件、分析构件、设计构件、实现构件、测试构件.

构件是构件库存储的最小单位和基本单位, 内部结构对构件库系统透明, 其粒度大小由构件生产者决定, 但是为了广泛地复用和灵活地与其他构件进行装配, 构件粒度通常不宜太大[7]. 并且复合构件和其组成元素构件无论在逻辑上和物理上都应分开, 不能包装成为一个整体, 因为一个元素构件可能参与多个复合构件的合成, 也可能单独使用.

1.1.2　构件系统　构件系统是具有某种关联的构件和子构件系统的集合, 如能够把密切关联、经常一同复用以实现某个常用功能的构件组织成为一个构件系统, 它是构件库存储的又一基本单位. 构件系统自身称为父构件系统, 作为其构成元素的构件或构件系统称为子构件或子构件系统. 构件系统不是复合构件, 复合构件是一个构件, 反映其组成元素间更加紧密的功能耦合关系, 但构件系统反映的是其组成元素间的一种相对松散的关系, 并且不仅仅限于功能耦合关系, 也可能是其他耦合关系, 具体语义由构件系统的生产者决定, 但对这种关系的反映和表达具有重要的意义.

收稿日期: 2001-03-20　基金项目: 国家"九五"重点科技攻关项目(98-780-01)基金资助　作者简介: 任洪敏, 博士研究生, 主要研究领域为软件复用. E-mail: RenHongmin@263.net. 王渊峰, 博士生, 主要研究领域为构件库管理系统. 钱乐秋, 教授, 博士生导师, 主要研究领域为软件工程、软件复用技术.

构件系统和其子孙构件系统构成一个树型结构. 但一个构件或构件系统能够成为多个父构件系统的组成元素, 故构件和构件系统间的包容关系是一个图结构, 但我们借鉴 UNIX 系统文件共享的思想, 通过构件或构件系统在其多个父构件系统间的透明共享, 而成为一个树型结构, 便于用户使用和操作, 对该树型结构的层次数目没有限定(见图 1). 支持共享的目的是因为一个构件或构件系统完全能够加入多个构件系统和参与多个复合构件的合成, 具有重要的意义.



图例: 图中节点表示构件或构件系统, 连线表示包容关系. 通过共享, 网络结构变成为一个树型结构

图 1　通过共享, 构件、构件系统间的包容关系

引入构件系统概念的目的, 是为了把构件组织成为构件系统, 揭示构件间的耦合关系和构件库的体系结构, 提高复用的粒度和相关构件的复用, 同时构件系统中的组成元素也能够共享复用语境信息. 因此, 应该鼓励把构件组织成为构件系统. 构件系统自身亦作为一个构件看待, 它具备和构件相同的属性, 如标识符、词汇表、文档等属性, 也用刻面对其进行描述, 从而便于整个构件库系统的统一管理. 但存储结构上, 构件系统与其组成元素是分离的, 它们之间只具备逻辑的包容关系.

1.2　构件、构件系统的类型机制

引入构件和构件系统类型的概念, 一是丰富构件系统逻辑模型的语义, 易于构件和构件系统的复用, 二是表达构件或构件系统之间的关系, 如追踪关系, 即需求构件(系统)与其对应的分析构件(系统)、设计构件(系统)彼此间的双向追踪关系.

1.2.1　构件的类型　构件对应软件开发生命周期的五个阶段, 分为五种类型: 需求构件、分析构件、设计构件、实现构件和测试构件.

1.2.2　构件系统的类型机制　构件系统分为三大类: 过程构件系统、领域构件系统和附属构件系统.

过程构件系统是软件开发某个过程的构件产品构成的集合, 从纵向对某个领域的构件进行分组. 过程构件系统分为需求构件系统、分析构件系统、设计构件系统、实现构件系统和测试构件系统, 它们都由相同类型的构件和子构件系统组成, 如设计构件系统只能容纳设计构件和子设计构件系统.

领域构件系统从横向对构件库中的构件进行划分, 其构成元素只能全为子领域构件系统或者全为过程构件(系统), 如果全为子领域构件系统, 则表示对该领域构件系统的进一步划分; 如果全为过程构件(系统), 则表示对该领域构件系统进行纵向划分, 因此, 该情况下, 各种类型的过程构件(系统)最多只能有一个, 如只能有一个分析构件或分析构件系统, 当然, 该分析构件系统可以包含若干子分析构件系统对其进一步划分.

附属构件系统是附属于某个构件的构件系统, 反映该构件与其附属构件系统之间存在的某种关系. 附属构件系统分为两类: 引用构件系统和特化构件系统. 引用构件系统反映构件间的引用关系, 用于对复合构件的支持, 一个构件如果是复合构件, 则其所引用的构件在附属的引用构件系统中. 特化构件系统反映一个构件的可变性, 特化构件系统中的构件是其所属构件的变体, 能够对该构件中的可变点进行客户化.

1.3　构件、构件系统间的关系及表达

构件、构件系统间的关系揭示构件库的体系结构, 有效地支撑构件库的复用功能和其它功能, 如领域学习功能. 构件库中构件间的关系纷繁复杂, 复杂多样, 但我们不必关心特定具体的关系, 具体关系的语义由用户确定. 通过对构件、构件系统间关系进行抽象和提炼, 定义了构件、构件系统间的四种关系及其对应的表达方式, 简单但却具备强的表达能力和高度的抽象能力. 在用户界面实现的时候, 结合语境和图标, 一目了然. 具体如下.

1.3.1　分组关系　分组关系表达一个构件和构件系统的集合按照某种准则划分成为若干个子部分. 应用相同种类的构件系统的嵌套结构表达分组关系, 如领域构件系统嵌套多个子领域构件系统, 分析构件系统嵌套多个子分析构件系统. 因为构件、构件系统间的共享, 构件、构件系统间的分组与划分并不要求正交. 分组关系, 能够把构件和构件系统组织成为粒度更大的复用单位, 和激发同组相关构件的复用.

1.3.2　追踪关系　追踪关系表达完成某一特定功能的不同种类的构件或过程构件系统间的相互追踪. 其表达方式是把可相互追踪的构件或过程构件系统组织在一个领域构件系统之下, 亦即如果一个领域构件系统的元素是构件或过程构件系统, 如为一个需求构件、一个分析构件系统、一个设计构件系统, 则表示它们之间具备追踪性.

1.3.3　引用关系　引用关系反映构件间的引用层次关系, 即合成构件和其组成元素间的关系, 即反映一个构件在实现的时候调用其他构件. 引用关系由构件附属的引用构件系统表达.

1.3.4　特化关系　特化关系反映一个构件与其所属的变体间的关系, 支持对构件进行客户化. 特化关系由构件附属的特化构件系统表达.

1.4　构件和构件系统的扩充属性及其共享机制

1.4.1　构件和构件系统的扩充属性　学习曲线是构件复用的主要障碍[2], 且构件库是个宝贵和昂贵的资源, 应对其充分利用, 构件库除了常用的软件复用功能之外, 还应具备其它功能, 如软件工程人员通过学习构件库中的构件系统, 能够熟悉和掌握编程语言、编程风格、应用模式、应用领域知识等[4].

因此, 构件和构件系统除了运用刻面进行描述外, 还应存储其他丰富的信息, 充分和全面支持构件库的功能. 此类信息, 称为构件和构件系统的扩充属性, 具体包括:

① 术语表: 定义构件和构件系统中使用的术语的含义、

差异和它们之间的关系及组合运用方式 一致的术语运用对构件系统的理解和交流具有重要的意义

② 学习教程: 指导软件复用者学习使用该构件或构件系统的步骤和指南

③ 复用示例: 构件或构件系统复用的典型样例

④ 用户文档段: 该文档段能够被复用者合并进入描述应用系统的用户文档中, 从而实现构件和构件系统的文档复用

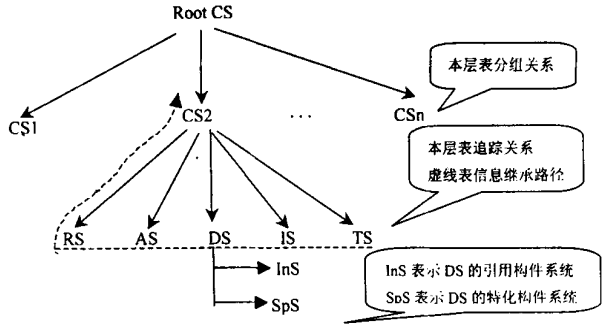
⑤ 复用反馈信息: 描述该构件或构件系统的被复用的历史, 及用户对它的反馈评价

1.4.2 构件和构件系统扩充属性的共享机制 构件和构件系统的扩充属性信息量大, 并且一个构件系统的子构件或子构件系统因为属于同一领域或者同一个构件生产商, 完全可能具有相同的或被组织在一起的扩充属性, 如相同的术语表扩充属性的共享机制一则能够方便构件和构件系统的生产者, 二则能够节省系统资源 构件和构件系统扩充属性共享原则如下:

① 生产者指定优先原则 构件和构件系统的扩充属性可由其生产者明确指定, 亦可指定为空, 即不具备该项属性否则, 按下列顺序共享其它构件或构件系统的属性

② 继承高抽象级别构件、构件系统属性原则 运用追踪机制, 查询抽象级别更高的具备追踪关系的构件或构件系统的该项属性, 查询顺序为: 测试构件(系统)、实现构件(系统)、设计构件(系统)、分析构件(系统)和需求构件(系统), 直到某构件或构件系统具备该项属性, 就共享之 否则, 遵循下面原则

③ 继承父构件系统属性原则 构件或构件系统至此尚未获得该项信息, 则查询其父构件系统和祖先构件系统的该项属性, 如有, 共享之, 否则, 为空



图例: CS 表示领域类型的构件系统 RS、AS、DS、IS 和 TS 分别表示需求、分析、设计、实现和测试类型的构件或构件系统 InS 表示引用构件系统, SpS 表示特化构件系统, 它们皆附属构件系统 图中未画出构件或构件系统自身的共享

图 2 构件库存储系统的逻辑结构、构件之间的关系、信息共享机制

④ 共享属性冲突解决原则 一个构件或构件系统可能是多个父构件系统的组成元素, 具备多条共享路径, 可能产生共享冲突 解决办法是一个构件或构件系统加入另一个父构件系统的时候, 除非特别指明或者该项属性为空, 否则, 保持不变

构件库存储系统的逻辑结构、构件间关系及表达和信息继承机制的图示见图 2

2 构件系统的物理结构

构件系统的物理结构描述构件系统的存储结构, 存储构件系统的具体数据 构件系统的存储策略应与构件库管理系统的存储策略一致 存储构件和构件系统理想的物理存储结构是面向对象数据库, 但考虑技术的成熟程度和应用的普及程度和可移植性, 及关系数据库自身具备的安全管理、事务管理、并发控制等功能, 我们决定采用关系数据库存储构件库和构件系统数据 在关系数据库上, 构筑面向对象的数据管理和操纵层, 给上层工具系统提供一个虚拟的面向对象数据库视图(见图 3).

构件库检索系统、评估系统、管理系统…
面向对象封装层
关系数据库(构件、构件系统)
计算机平台

图 3 构件库存储系统的总体设计

2.1 构件系统存储结构的关系模型

系统运用 9 张关系数据库表存储构件库的物理数据, 现具体阐述如下

2.1.1 构件扩充属性存储表 构件或构件库的扩充属性因为共享, 故用 5 张表分别存储构件或构件系统的扩充属性 该 5 张表具备相同的模式, 包括 4 个字段: 属性唯一性标识符、共享计数、数据格式和二进制流存储的属性数据本身

2.1.2 构件系统存储表 构件系统存储表存储构件系统的信息 存储的内容包括构件系统的唯一性标识符、4 个扩充属性的属性标识符、构件系统类型、共享计数

2.1.3 构件系统元素表 构件系统元素表存储构件系统的组成元素 其字段为父构件系统标识符和子构件或子构件系统标识符

2.1.4 引用构件系统表 引用构件系统表存储构件附属的引用构件系统 其字段为构件唯一性标识符和表示其引用构件系统的构件系统唯一性标识符

2.1.5 特化构件系统表 特化构件系统表存储构件附属的特化构件系统 其字段为构件唯一性标识符和表示其特化构件系统的构件系统唯一性标识符

3 构件系统面向对象管理实现

为了增强构件系统的易用性和稳定性, 决定对关系数据库存储的构件系统进行面向对象封装, 给上层工具开发人员提供一个虚拟的面向对象存储的构件系统视图

3.1 面向对象封装的层次结构

采用层次结构进行面向对象封装(见图 4).

数据库操纵对象是 VC++ 提供的数据库操纵对象, 负责具体的数据库存取操作 因为共享, 能够从多条路径检索构件或构件系统, 待访问的构件或构件系统可能已经从数据库

中取出, 缓冲管理对象负责对此进行管理, 以减少数据库访问

服务层对象
缓冲管理对象、共享管理对象
数据库操纵对象
关系数据库(构件、构件系统)

图 4　面向对象封装层次结构图

次数. 共享管理对象负责对构件和构件系统的扩充属性或其自身的共享情况进行管理, 包括对共享计数进行管理和清除管理等. 服务层对象负责对上层工具系统如构件库检索系统和管理系统提供服务. 下面具体给出服务层对象的设计.

3.2　服务层对象的对象模型设计

运用组合模式(Composite)和重述器模式(Iterator)[8], 设计服务层对象模型(见图 5). 构件和构件系统统一处理, 整个系统逻辑结构简明, 对象模型亦相当简明, 给上层系统提供了一个相当简单的开发接口.

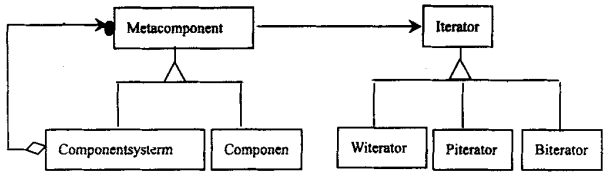


图 5　服务层对象的对象模型

Metacomponent 类封装构件和构件系统的共同属性和方法, 如构件或构件系统标识符和各种属性存取运算方发. Componentsystem 类封装构件系统特有的属性和方法, 如子构件和子构件系统对象的指针, 建立和删除子构件或子构件系统方法等. 运行时构件和构件系统对象的逻辑结构是个动态按需生成的树林结构, 采用三叉链结构存储. 重述器 Iterator 提供访问接口, 其三个子类分别提供宽度优先、先根次序和后根次序深度优先的树林周游算法.

4　结　论

论文设计和开发的构件系统模型较好地克服了常见构件库的不足. 其主要创新点在于: 支持构件和构件系统共享, 引入构件和构件系统类型机制, 能够简明地把构件组织成为大粒度可复用构件系统, 支持构件的连锁复用, 表达构件库的体系结构, 存储构件和构件系统的丰富信息并且支持此类信息共享, 从而全面有效地支持构件库的功能如复用功能和学习功能. 利用关系数据库存储构件系统并提供面向对象访问接口, 系统界面友好和易于进化.

今后, 我们将在下面两个方面继续展开工作:

1 动态构件系统的支持和管理. 现在介绍的构件系统由构件生产者或者构件库管理员创建并进行维护. 动态构件系统根据用户的查询和反馈信息, 自动识别耦合的构件, 把其组织成为构件系统, 并且删除过时的构件系统.

2 分布式构件库的支持和研究. 运用构件系统的概念和分布式面向对象计算, 对分布存储的构件系统提供一个统一的逻辑结构, 即把每个站点的构件库作为根构件系统的一个子构件系统.

参 考 文 献

1 Ivar Jacobson, Martin Griss et al. Software resuse [M]. ACM Press, 1977

2 Yang Fu-qing, Zhu-bing, Mei Hong. Software resue [J]. Journal of Software, 1995, 6 (9): 525～533
(杨芙清, 朱冰, 梅宏. 软件复用 [J]. 软件学报, 1995, 6 (9): 525～533)

3 Yang Fu-qing, Shao Wei-zhong, Mei Hong. The design and implementation of an object-oriented CASE environment in jade bird 2 system [J]. Science in China (Series A), 1995, 25 (5): 533～542).
(杨芙清, 邵伟中, 梅宏. 面向对象的 CASE 环境青鸟Ⅱ型系统的设计与实现 [J]. 中国科学(A 辑), 1995, 25 (5): 533～542)

4 Rym Mili, Ali Mili. Storing and retrieving software components: a refinement based systems [J]. IEEE Transactions on Software Engineering, 1977, 23 (7): 445～460

5 Tomas I Sakowitz, Robert J. Kanffman. Supporting search for reusable software objects [J]. IEEE Transactions on Software Engineering, 1976, 22 (6): 407～423

6 Chen Xiao-qun, Shao Wei-zhong et al. Supporting project-centered object-oriented resue [J]. Journal of Software, 1999, 10 (3): 283～287
(陈小群, 邵伟中 等. 以项目为中心的面向对象复用支持 [J]. 软件学报, 1999, 10 (3): 283～287)

7 Buhr, R. J. A. Casselman, R. S. Use case maps for objected-oriented systems [M]. Prentice Hall, 1996

8 Erich Gamma, Richard Helm et al. [M]. Design Patters. Addison Wesley, 1994

9 Damiani, E. and Fugini, M. G., Bellettini, C. A hierarchy-aware approach to faceted classification of objected-oriented components [J]. ACM Transactionson Software Engineering and Methodology, July 1999, 8(3): 215～262

10 Dieter Merkl, A Min Tjoa, Gerti Kappel; Learning the semantic similarity of reusable software components [C]. Proc. of the 3rd Int'l Conference on Software Reuse (ICSR'94). Rio de Janeiro. Brazil. Nov 1-4. IEEE Computer Society Press. 1994. 33～41.

Modeling and Implementing a Component System in a Component Libary

REN Hong-min, WAN Yuan-feng, QIAN Le-qiu
(*Department of Computer Science and Technology Fudan University, Shanhai* 200433, *China*)

Abstract A component system is a set of components or subordinate component systems associated together. It is of significances to support and implement component systems in a component library. A logical structure, a physical structure and a objected-oriented implementing technology for a component system are proposed in this paper, which characterize type mechanisms of components and component systems, plentiful and sharable information about components and component systems, and representation and storage of relations between component systems and/or components. Therefore, by means of the component system, component library's architecture can be revealed and represented easily and effectively. Software reuses throughout the whole software engineering process and reuses of large-grained component systems, as well as the other functions of a component library, for example studying an application domain, can be supported thoroughly.

Key words component library; component system; software reuse; object-oriented; technology