文 章 编 号: 1000-1220(2002)02-0129-04

对复用构件库体系结构的几点研究

王渊峰 朱三元 钱乐秋

(复旦大学 计算机科学系, 上海 200433)

摘 要: 复用构件库是软件复用的重要的技术支撑之一. 但由于目前的构件库体系结构主要是基于静态管理的数据仓储结构或层次结构, 忽视了构件作为其复用单元本身的变化发展特征, 所以并不能为软件开发提供真正意义上的面向复用的服务. 本文结合构件的复用性能的有关度量模型, 提出了一种新型的层次型体系结构, 从结构机制上提高了构件库的复用性能, 有利于软件复用的实践与发展.

关 键 词: 软件复用; 复用库; 体系结构

中图分类号: TP311 **文献标识码**: A

软件复用是提高软件开发生产率和保证软件产品质量的一条行之有效的途径. 但是, 经过了几十年的努力与发展, 软件复用的思想虽然已日趋完善与成熟, 软件复用的方法却并没有在软件生产中被广泛地系统化与实践化. 其在实践上的一个主要原因是由于缺少足够适用的复用构件库可以利用[1,2]. 在这里 '足够' 是指构件库的覆盖率, '适用' 是指检索到构件的精确度. 它们是软件构件库的两个重要的性能指标.

传统的复用构件库的体系结构主要有仓储型和层次型两种体系结构[3]. 它们主要的设计依据是一种静态的库结构. 即将可复用构件极其相关文档作为存储资源, 从构件相关文档中提炼出各个构件的特征轮廓(这一过程可手工也可自动化), 并在此基础上构建起相应构件库的索引体系及检索系统[1]. 从实践效果来看, 这种静态的库结构虽然便于控制与管理, 技术也相对成熟. 但由于其忽视了对可复用构件本身的可复用性、复用率等动态复用性能因子的适时的监控与管理. 使得构件库的覆盖率与检索率随着时间的发展而逐渐衰减, 从而并不能为软件的开发提供真正意义上的面向复用的服务, 失去了其作为复用构件库本身的意义.

本文在传统的复用库体系结构的基础上, 结合构件的复用度量模型, 提出了一种改进的分层的复用库体系结构. 这种体系结构根据构件的复用度量模型, 引进竞争-淘汰机制, 通过前、后端构件库的动态自适应调配, 在相同的构件库覆盖率的基础上, 能够提供更高的复用构件检索率, 从而为软件复用的发展奠定更加坚实有力的基础支持.

本文在结构上安排如下: 第一部分简要介绍了传统构件库的体系结构, 指出了其中的不足, 并提出了改进的方向与方法; 在第二部分中详细介绍了一种基于动态的改进的层次型复用库体系结构, 并着重分析了其中的关键模块及调度管理算法; 在第三部分我们给出了一组模拟复用库平台上的实验数据, 证明了这种改进体系结构的有效性与可行性; 在第四部分总结了这种改进体系结构的原理与思想, 并给出了目前的工作与进一步的打算.

1 传统构件库的体系结构

传统的构件库的体系结构, 主要分为两大类, 一类是仓储型(图 1), 另一类为层次型(图 2).

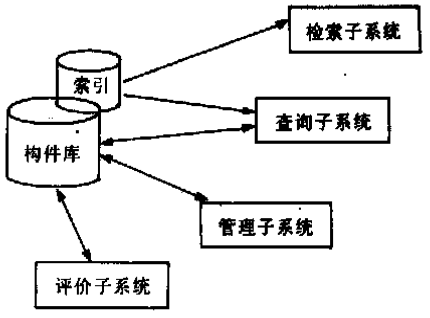

图 1 可复用构件库的仓储型体系结构

上述两种体系结构的主要区别在于各子系统边缘功能的划分及连接器的通信机制[3]. 综合来说, 整个体系结构主要包括一个复用构件库和查询、检索、管理、评价四个子系统. 在构件库中每个构件的存储数据主要包括: 构件、构件描述文档和

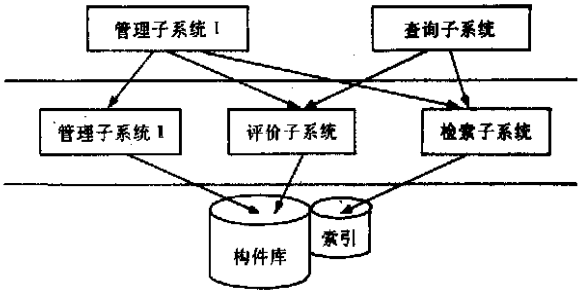

图 2 可复用构件库的层次型体系结构

收稿日期: 2000-11-06 基金项目: 国家九五攻关项目"青鸟系统"基金资助 作者简介: 王渊峰, 博士, 主要研究领域为软件工程、软件复用技术. 朱三元, 研究员, 博士生导师, 主要研究领域为软件工程. 钱乐秋, 教授, 博士生导师, 主要研究领域为软件工程. E-mail: www_yf@sina.com

有关的规格说明文档、测试与演示实例等三个主要部分[5]. 在这些构件数据中可通过自动化析取或领域专家的经验分析提取出相应构件的特征轮廓, 进而构筑起分类方案、刻面和索引库[1]. 查询子系统和检索子系统主要负责给用户提供个性友好的查询手段以及快速正确的检索技术, 如: 超文本链接浏览、关键字描述检索、层次聚类检索等技术[4]. 管理子系统Ⅰ、Ⅱ主要负责构件库数据日常的管理与维护工作, 如: 用户授权验证、构件数据增减等[6]. 评价子系统主要根据各构件日常的检索、查询的情况与领域专家的打分、并结合构件评价的定量模型较适时地统计各构件的评价因数, 以协助构件库的管理与查询

综上所述, 传统的构件库体系结构基本上是建立在一种静态的组织、调度体系上的. 虽然管理子系统可以定期地通过手工配置对复用库的内容进行调整, 但这种调整往往是盲目的、滞后的. 例如, 当删去一个构件后, 也许并没有更合适的构件可以适时的填补上, 从而使复用库的覆盖率被降低, 有时填补上的也许是可复用性更低的构件, 从而降低了复用库的检索率[6]. 基于此, 本文提出了一种改进的, 基于动态的层次型复用库体系结构, 利用它可以更加合理有效地构建与配置复用构件库, 提供有效的面向复用的构件服务.

2 基于动态竞争的层次型复用库体系结构

图 3 是改进后的基于动态的层次型复用库体系结构图. 其中主要有以下四点改进: 第一是将复用库分为前端构件库与后端(候选)构件库两部分, 规定检索的结果应注明该构件的来源库, 推荐使用前端库中的构件. 第二是复用库的查询子系统应向管理子系统反馈构件的被复用情况, 例如: 构件所参与的复用项目、复用构件所做的修改量、构件被复用的次数等复用信息. 以利于构件的复用特征轮廓的丰富与评价系统的

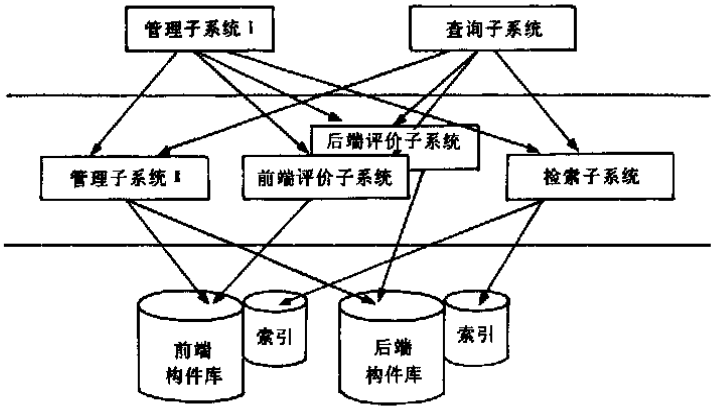

图 3 改进后的基于动态的层次型复用库体系结构

准确. 第三是在管理子系统中加入前后端调度模块, 其中的调度算法概述如下:

首先, 我们设计复用竞争因子 $g(c, t)$ 描述构件 c 在 t 时刻的可复用性, $g(c, t)$ 的定义为:

$$g(c, t) = \alpha^n + \beta h_0 + \gamma(t_1) \quad ①$$

$$\gamma(t_1) = \sum g(c, t_1) / N \quad ①'$$

其中: t_1 为构件 c 入库时间; t 为当前时间;

n_0 为构件 c 入库后被选中复用的次数;

n 为构件 c 的同类构件自构件 c 入库后被选中复用的总次数(n 包含 n_0);

N 为构件 c 的同类构件的总数

α 为时间衰减常数, α^n 称为时间衰减项;

β 为复用增益常数, βh_0 称为复用增益项;

$\gamma(t_1)$ 为 t_1 时复用库中所有构件(除 c 以外)的复用竞争因子 $g(c, t_1)$ 的平均值, 称为默认复用竞争因数, 在建库时为系统设定的常量 R;

在文献[2]中对构件的复用性能的衡量, 时间衰减项采用的是 $(t - t_1)$ 绝对时间值, 上式使用 n 相对时间值来表示复用性能时间衰减项, 既统一了①式中各项的度量, 同时也更客观地体现了复用构件的有效使用期. α、β 之间的关系为 $\beta = \alpha^N$, 这样设计使得所有构件复用竞争因子的总和基本保持不变(当在复用库中选择构件失败时, 根据式①所有该类构件的复用竞争因子应下降 α). 由于构件复用竞争因子的设计主要目的是解决构件间复用性能相对大小的比较, 所以 α 和 β 的具体数值的选取并不重要.

有了复用竞争因子的定义后, 前后端构件库的调度法则如下:

t 时刻, 构件 c 从前端库淘汰至后端库的准则为:

$$g_{前}(c, t) < \overline{g_{前}(c, t)} - \rho \cdot \sigma_{前} \quad ②$$

其中: $\overline{g_{前}(c, t)}$ 表示前端库中所有构件的复用竞争因子的平均值;

σ 表示前端库中所有构件的复用竞争因子的标准方差;

②式的淘汰思想比较直观, 它默认在复用库中构件被复用的概率服从高斯分布, 当构件 c 的复用竞争因子处于高斯概率分布的弱势区时, 认为其有被淘汰的必要. 在②式中 ρ 是一个经验常数, 我们推荐它的取值范围为 [0.8, 1.6], 因为对

于高斯分布有: P(|x- μ|> 0.8σ ≈ 0.424 和 P(|x- μ|> 1.6σ ≈ 0.110. 同时根据对传统复用库管理统计数据的模拟分析, 一般调整量为复用库构件总数的 2% 至 5% 时, 有利于复用库复用性能的较好收敛, 具体的实验数据见本文的第三部分.

同理, 当在 t 时刻构件 c 满足③式时, 认为它有必要从后端库晋升至前端库:

$$g_{后}(c,t) > \overline{g_{后}(c,t)} + \rho \cdot \sigma_{后} \qquad ③$$

当然, 也有从后端库淘汰出库外的准则, 这一般与原管理子系统的淘汰原则一样(并不赞成轻易地从整个库中淘汰构件), 此不再赘述.

由于构件 c 的复用竞争因子在短时期内可能呈震荡变化, 所以前后端构件库的调度时机定义如下:

对构件 c 从前端库向后端库进行淘汰操作的时机为:

$$n_c > \S \qquad ④$$

④式中 n_c 表示复用库连续 n_c 次选用构件时未选用构件 c, § 的定义为〔(N-1)/N〕§ < 1%, 该式认为一个理想的复用库中每个构件的复用性能应基本相同, 当选用一个构件时, 构件 c 不被选中的概率近似为(N-1)/N, 它连续 § 次不被选中的概率为〔(N-1)/N〕§, 当这个概率小于< 1% 时认为该构件复用性能很差, 应被淘汰至后端库. 同理, 我们可类似设计出构件 c 从后端库向前端库进行晋升操作的时机, 此不再赘述.

上述竞争-淘汰调度算法的描述, 主要提供原理, 至于具体的经验常数的选择、概率密度函数的模拟及复用竞争因子的定义可参考实际模型再具体设计. 在第三部分的实验中, 我们采取了实验数据分析与上述理论相结合来确定具体的经验常数值的方法, 效果理想. 在以后的具体实践中也可采用这种通过构建模拟平台来调节、选定经验常数值的方法.

该体系结构最后一个主要的改进是在复用库的检索子系统中应能够提供按复用竞争因子排序给出构件的推荐浏览, 并且当构件变迁位置后, 应及时在查询子系统中表注其去向信息, 提供良好的信息接口.

3　实　验

我们用 MathCAD7.0 和 VC6.0 相结合, 构建了一个复用库使用的模拟平台 ReuseLibrary, 分别对两种库体系结构(层次式与改进层次式)进行了实验测试(主要模拟前端库的复用管理), 得到了下两张表格数据.

实验平台模拟某一领域的构件数为 100 个的一个复用库, 并设初始的 100 个构件实际复用性能呈高斯分布, 在库评价子系统中用基于复用记录的复用竞争因子来模拟各个构件的复用性能, 并假设平均每天使用 5 个构件. 在选择构件时, 若连续选择三次都未选中合适的构件, 则记录一次 NotFinds, 这时认为库中不存在合适的可复用构件. 每一次的选择失误, 都记录为一次 FindFails. 显然, FindFails 是该复用库的检索精确率的一个度量参考, 而 NotFinds 是该复用库的覆盖率的一个度量参考. 对于传统的复用库, 每 100 天进行一次管理(如果增大管理周期天数, 实验结论会更明显), 而对于改进体系结构的复用库则使用参数集〔§ = 500、α = 0.01、β = 1、R = 5、ρ = 1.5〕进行实时的管理.

表 1　传统体系结构复用库的模拟实验数据

选用构件个数 (500 个/次)	平均复用竞争因子	FindFails	NotFinds	淘汰的构件个数
第 1 次	5.09	379	250	11
第 2 次	5.08	356	254	7
第 3 次	5.37	344	239	13
第 4 次	5.67	340	219	14
第 5 次	5.92	331	186	10
……	……	……	……	……
第 36 次	8.51	128	33	2
第 37 次	8.33	130	34	5
第 38 次	8.33	139	33	4
第 39 次	8.35	126	26	5
第 40 次	8.37	161	35	7
……	……	……	……	……

对照表 1 和表 2 分析, 容易得到两种体系结构在实验中库平均复用竞争因子比较图(图 3). 从图中可以看到: 传统体系结构下复用库的平均复用竞争因子呈震荡向上的发展, 但在中后期则几乎在原地震荡, 而改进后的复用库的平均复用竞争因子则能较平稳、迅速的收敛于模拟中的较理想的平均复用竞争因子值.

表 2　改进体系结构后复用库的模拟实验数据

选用构件个数 (500 个/次)	平均复用竞争因子	FindFails	NotFinds	淘汰的构件个数
第 1 次	5.17	361	247	4
第 2 次	5.24	354	242	5
第 3 次	5.36	346	238	6
第 4 次	5.74	330	197	5
第 5 次	6.04	298	160	5
……	……	……	…… ……	
第 22 次	9.43	12	3	1
第 23 次	9.45	11	2	0
第 24 次	9.45	10	1	0
第 25 次	9.45	10	2	1
第 26 次	9.46	9	1	0
……	……	……	……	……

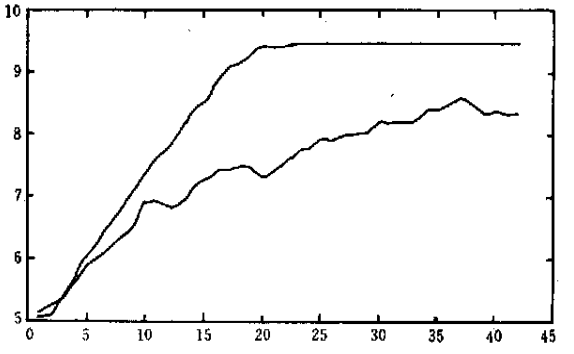

图 4　两种体系结构在实验中库平均复用竞争因子发展比较图

从表中还可以直观地发现, 随复用库的平均复用竞争因子的增大, 库的 FindFails 和 NotFinds 也相应减小, 从一个侧

面证明提高了复用库的覆盖率和检索的准确率

4 结束语

综上所述,由于在改进后的复用库体系结构中建立了前、后端构件库的基于复用性能的调度机制,并在原有的纵向层次的基础上丰富了横向层次的竞争-淘汰的管理意识,使得改进后的构件库体系结构无论在理论上还是在实验中都能够比原有的构件库体系结构提供更高的复用率与复用性.下一步我们主要的工作将是使这种改进的库结构从实验推向实践,开发出较完善的相应的软件复用库的模拟测试平台与应用平台,并使之构件化与普及化.同时进一步研究复用竞争因子和竞争-淘汰调度机制的分类细化模型.我们有理由相信,随着构件库体系结构的面向复用的逐步完善与发展,必将带来软件复用生产力的进一步的提高与发展

参 考 文 献

1 Xu Zheng-quan. Approaches to software reuse and related techniques [M]. Huazhong University of Science and Technology Press. Wuhan, China: 1998.
(徐正权. 软件复用方法与技术 [M]. 武汉: 华中理工大学出版社. 1998. 101～169)

2 Carma McClure. Software reuse techniques [M]. Prentice Hall PTR. 1997.

3 Mary Shaw, David Garlan. Software architecture [M]. Prentice Hall PTR. 1996.

4 Tomas Isakowitz, Robert J. Kauffman. Supporting search for reusable software objects [J]. IEEE Transaction on SE. 1996, 22(6): 407～423

5 William B. Frakes, Thomas P. Pole. An empirical study of representation methods for reusable software components [J]. IEEE Transaction on SE. 1994, 20(8): 617～630

6 Thomas Wappler, Kathryn P. Yglesias. What a reuse tool can do for you [J]. Obect Magazine. 1995, 1. 42～45

Research on Architectures of Reuse Library

WANG Yuan-feng, ZHU San-yuan, QIAN Le-qiu

(*Department of Computer Science, Fudan University, Shanghai* 200433, *China*)

Abstract A Reuse Library is an important element of practicing software reuse. However, present architectures of reuse library are mostly based on static repositories or layered architectures which ignore the reuse-oriented change and grow in the contents of the reuse library, so they can't provide reuse-oriented services authentically. This paper assisted with the reuse metrics of components to provides a new layered architectures which can improve reuse performance of reuse library and advance the progress of the software reuse.

Key words software reuse; reuse library; architectures