软件构件表示与检索形式化的研究与进展

徐正权[1] 王家兵[1] 王能超[2]

(华中科技大学计算机学院 武汉430074)[1] (华中科技大学并行计算研究所 武汉430074)[2]

A Survey of Formal Methods of Software Component Representation and Retrieval

XU Zheng-Quan[1] WANG Jia-Bing[1] WANG Neng-Chao[2]

(Computer College[1], Institute of Parallel Computing[2], Huazhong University of Science & Technology, 430074)

Abstract Software reuse has been claimed to be the realistic means to increase the productivity of programmers and improve the quality of developed software. Software reuse approaches can be classified into two closely related categories: development for reusable component and development with reusable component. System development with reusable components involves in a series of related woks: component representation, component retrieval, component composition, component modification, etc. Because formal methods have rigorous theoretical foundations and can characterize computational semantics of a component, they has been intensively researched in software engineering domain. This paper surveys the formal methods of component representation and retrieval, introduces current status and outlines future trends.

Keywords Software reuse, Component representation, Component retrieval, Formal methods

1 引言

软件复用被认为是提高软件生产效率和软件质量较为现实的途径[3,13,27,30,37,41]。尽管软件复用的思想已经提出了30多年,但软件复用的现状离人们最初的设想仍然相距甚远。阻碍大规模软件复用的技术与非技术因素很多,主要的技术因素有构件分类、构件表示、构件检索、构件更改及构件库的管理与维护等[30,41]。其中,软件构件表示与检索是软件复用获得成功的重要前提[30]。

随着计算机硬件和软件在规模及功能方面的不断增长与提高,其复杂性及出错的可能性也在不断增加。软件工程研究的主要目的之一是使开发者在这种系统复杂性不断增长的情况下仍然能构造出尽可能可靠的系统。达到这个目的的一个途径是形式化方法的应用[9,18]。形式化方法是以数学为理论基础(主要为逻辑与代数)构造程序设计语言、技术及工具来规约和验证系统的方法。

形式化方法的使用并不是确保系统正确性的先决条件,但是形式化方法具有以下非形式化方法不具有的优点[9,15,18]:

(1)及早发现及排除系统中隐含的不一致性、不完备性及歧义性等错误;

(2)具有严密的数学基础,支持对构件属性的自动推理;

(3)能够完整刻画构件的计算语义,能够使用户清楚地理解该构件的功能。

2 国内外研究现状

Frakes[11]和Mili[30]分别对现有的构件表示和检索方法进行了分类。前者从构件表示出发,将现有方法分为人工智能方法(Artificial Intelligence Methods)、超文本方法(Hypertext Methods)和信息科学方法(Information Science Methods);后者则按照复杂度和检索效果的递增将其分为基于文本的(Text-Based Encoding and Retrieval)、基于词法描述子的(Lexical Descriptor-Based Encoding and Retrieval)和基于规约的编码和检索方法(Specification-Based Encoding and Retrieval)。本文以H. Mili提出的分类法进行讨论。

由于自然语言固有的模糊性、不完备性及不一致性,基于文本的表示及检索方法既不可靠也不完备;基于词法描述子的方法也有与基于文本的方法同样的弱点:对构件的编码没有直接反映出该构件的计算语义。从反映构件的计算语义的角度来看,基于规约的构件表示与检索方法与这一目标最为接近[30]。

关于基于文本和词法描述子的构件表示与检索方法可参考文[6, 11]。本文主要介绍国内外在构件表示与检索形式化方法研究方面所做的工作。现有的形式化方法大致可分为基于规约的方法、基于人工智能的方法及其它方法。当然这种分类并不严格,因为基于规约的方法也经常用到人工智能的有关理论。

2.1 基于规约的方法

基于规约的构件表示与检索方法一般需要一种规约语言,构件的表示用该规约语言描述。构件检索时,用户首先写出需求规约,然后提交。构件检索的实际过程一般由一证明器根据构件规约与需求规约间的某一偏序关系进行检索,构件库中的构件一般按照构件规约间的偏序关系来组织。这样,在构件检索时检索历程可以大大减少需求规约与构件规约比较的数目,从而提高检索效率。检索效果与所采用的规约语言的子集有关,有的检索方法只采用基于基调(signature-based)的检索,有的采用基调与前/后件(pre-/post-conditions)联合检索。一般而言,若只采用基于基调的检索方法,对于检索回

徐正权 教授,研究领域为软件工程、软件复用等。王家兵 博士研究生,研究领域为自动推理、软件复用的形式化、并行计算等。王能超 教授,博士生导师,研究领域为数值算法设计与分析、并行计算等。

叫率(recall)没有影响,但准确率(precision)不高;若采用基调与前/后件联合检索,则准确率将会提高。

Mili 等人使用关系规约和"精化"偏序关系在构件库的组织、管理与维护,构件表示与检索、构件更改等方面做了大量的工作[2,24,32,33]。文[32]研究了可复用构件库的组织及基于偏序的构件检索问题。首先利用关系规约表示构件,然后利用一种规约间的精炼序(Refinement Order between Specifications)检索构件。一个规约用二元组(S,R)来表示,其中 S 是一个集合,称为规约空间(specification space);R 为 S 上的一个关系,称为规约关系(specification relation)。S 由命名变量的数据类型构成(如 int, real, char 等),R 由规约者认为正确的所有输入/输出对构成。如果状态空间 S 从上下文已知时,也简单地把一个规约简写为 R。当一个程序所处理的变量为空间 S 时,称该程序为定义在空间 S 的程序。给定一个程序 p,p 的功能抽象定义为[p]={(s,s')|程序 p 从状态 s 开始执行,以状态 s'结束}。因为 R 是 S×S 的一个子集,因此可以定义为 S×S 的一个二元谓词,即 $R=\{(s,s')|P(s,s')\}$。L 表示 S 上的普遍关系,即 $L=\{(s,s')|s,s'\in S\}$;R^* 为关系 R 的补,即 $R^*=\{(s,s')|(s,s')\notin R\}$。$RR'=\{(s,s')|\exists\ t(s,t)\in R$ 且 $(t,s')\}$表示关系 R 与 R'的关系积。如假设规约空间 S 为实数,一个以非负实数为参数的平方根函数的规约可表示为 $R=\{(s,s')|s\geq 0$ 且 $s'^2=s\}$。规约关系 R 为关系 R'的精炼当且仅当 $RL\cap R'L\cap(R\cup R')=R'$。当两个关系 R,R'满足条件 $RL\cap R'L=(R\cap R')L$ 时,则两个关系 R,R'具有最小上界(lub)$R\vee R'=(R'L)^*\cap R\cup(RL)^*\cap R'\cup R\cap R'$;任意两个关系 R,R'具有最大下界(glb)$R\wedge R'=R'L\cap RL\cap(R\cup R')$。这样定义的格操作形成一个规约格结构。给定一个规约格,一个程序 p 称为与格中元素 R 是相符的当且仅当 p 的功能抽象[p]是 R 的一个精练。如果一个程序 p 是与规约 R 相符的,R 是 R'的一个精练,则程序 p 与 R'也是相符的。文献中定义了两种检索策略,即精确检索与近似检索。对用户提交的一个查询规约 K,称一个构件 C 与 K 精确匹配当且仅当[C]是 K 的一个精练。给定一个规约格及查询规约 K,称格中元素 R 为关于 K 的最大元素,当且仅当对所有格中元素 R',有 $R'\wedge K\subseteq R\wedge K$。一个构件 C 称为与 K 近似匹配当且仅当 C 是与格中元素 R 相符的,其中 R 为关于 K 的最大元素。文献中实现了一个构件库,并进行了实际检索研究,结果显示该方法具有较高的查准率和较快的响应时间,但查全率较低。

文[19~23]在构件库的构造、管理、构件规约与检索方面做了一系列的工作。构件库是一两层的层次结构。构件的规约是用 LARCH 语言[14,40]完成的,完成构件的规约后,用包含性测试算法(subsumption,一个自动定理证明中的概念,参见文[5, 28])来构造构件的一个层次结构。其基本思路是利用包含性测试算法来定义一个叫做构件通用性(generality)的概念。构件 A 比构件 B 通用,当且仅当对构件 B 的每一个方法 M_b,存在构件 A 的一个方法 M_a,使得 M_a 比 M_b 通用。M_a 比 M_b 通用,当且仅当 $Pre(M_b)\supseteq_{clause} Pre(M_a)$且 $Post(M_a)\supseteq_{clause} Post(M_b)$。这里,$Pre(M_b)$表示方法 M_b 的前件,$Post(M_b)$表示方法 M_b 的后件,$\supseteq_{clause}$表示子句包含关系(subsume)。当确定各个构件之间的通用性以后,按构件的通用性把所有构件组织成层次结构。构件通用性越高,则该构件将处于的层次越高。

建立好底层层次结构后,底层是两两不相交的构件聚类(Component Cluster),用图来表示。然后用聚类分析技术把所有构件聚类连接成一个图,得到构件层次的顶层结构。在应用构件聚类分析技术时,需要构件相似性概念(Component Similarity),构件相似性是通过构件规约时用到的子句中的原子符号的相同数来定义。再计算出每一个构件相似性值(实数),然后用一 Hash 算法构造出 Hash 表。

在检索时,用户首先写出所需构件的规约(用 Pre/Post 的形式),然后提交给检索工具。首先根据用户提出的查询计算出该查询的相似性,然后与 Hash 表查询得到与用户查询相似性相差比较小的一个构件集合,再对这个集合的每一个构件利用包含性测试算法进行检索。直觉理解,方法 M_a 比方法 M_b 通用,即方法 M_a 能完成方法 M_b 的功能,因为有以下关系成立:$(Pre(M_b)\supseteq_{clause} Pre(M_a))\Rightarrow(Pre(M_b)\Rightarrow Pre(M_a))$,$(Post(M_a)\supseteq_{clause} Post(M_b))\Rightarrow(Post(M_a)\Rightarrow Post(M_b))$,这里⇒表示逻辑蕴含。但反过来不成立,即方法 M_b 不一定能完成方法 M_a 的功能。

上述技术存在的不足是:一方面方法的前件或后件不一定能仅用一个子句(Clause)就能进行描述,更多的情形可能是需要几个子句才能完整刻画该构件的前件或后件;二是即使能用一个子句进行描述,但仅用包含性测试算法来决定方法(构件)间的通用性关系是不够的,因为子句蕴含关系与子句包含关系是非等价的[39]。有可能 $Pre(M_b)$并不包含 $Pre(M_a)$,但有$(Pre(M_b)\Rightarrow Pre(M_a))$,即 $Pre(M_a)$是 $Pre(M_b)$的逻辑结果。如果进一步还有 $Post(M_a)$不包含 $Post(M_b)$,但有 $Post(M_a)\Rightarrow Post(M_b)$,这样的话,按照包含性测试算法将会错过构件 B。

文[8]在此基础上提出了基于类比推理的构件检索方法。类比是一种人类重要的认知方法,类比推理也是人类决策的一种常用的推理方式。类比简单地讲就是运用已知的熟悉的领域知识来解决未知或知之甚少的领域内的问题。类比推理的一个关键问题是如何认识(或定义)已知领域与未知领域的某种相似性。具体到运用类比推理来检索构件,就是如何定义用户对需求构件的规约与构件库中构件规约二者之间的相似性。文[8]使用有序多种类逻辑(order-sorted logic[38])规约构件与用户的构件需求规约。因为一个构件的实际功能是由该构件的所有方法(method)来完成,因此构件规约主要是对该构件所有的方法进行规约。对构件的每一个方法,用前件与后件(即自动推理中的子句)构成,前件定义了该方法在执行前必须满足的条件,后件定义了在满足前件的条件下执行该方法后,该构件各变量所处的状态。文中定义了种类(sort)、项(term)、表达式(可以是一个单独的项,或任意一个逻辑公式)、构件方法,及构件之间的相似性概念。检索时,用户写出需求构件的规约,检索例程计算构件库中的构件规约与需求构件规约之间的相似性,并返回与需求规约相似度最高的构件。

Chen 等人[7]提出了类似 Mili 等人工作的一种构件表示与检索方法,使用抽象数据类型(ADT)的代数规约(algebraic specification)及其之间的"实现"偏序关系。把软件构件看作是抽象数据类型,然后用基调与行为公理(behavioral axioms)进行规约。但是文献中实现的检索算法仅仅使用了基调匹配而没有考虑行为公理,即通过对构件类型的重命名看与用户规约之间能否实现类型匹配。

文[43]利用基调匹配(Signature Matching)来检索构件,它研究了两种构件的匹配情形,即函数(Function)和模块(Module)的匹配。基调匹配就是利用函数或模块中的函数的

数据类型信息，利用变量重命名(Variable Renaming)，看能否通过对用户所需求的函数基调中的变量通过重命名来与构件库中一个函数的基调匹配。这类似于自动定理证明中常要用到的合一(Unification，参见文[26])概念。显然，基于基调的构件检索还只是停留在语法匹配的层次上，一个函数或构件的基调不能反映出该函数的语义或功能。如用户检索一个计算平方根的函数，用户规约为(假设只是基于基调检索)：f：real→real，即所要检索的函数以实数类型为参数，返回值为实数类型。显然，sin，cos，tg 等三角函数都满足要求，因而查准率将会较低。

文[44]在文[43]的基础上进一步讨论了行为匹配(behavioral match)。文中利用形式化规约语言 LARCH 来描述构件的行为并利用自动定理证明技术来检索构件。文中用前件/后件(Pre/Post-Conditions)来描述构件的行为，并分别讨论了函数和模块两种情形的匹配问题。文中定义了各种匹配情况，如在函数情形中定义了精确前后件匹配(Exact Pre/Post Match)、嵌入匹配(Plug-in Match)、嵌入后件匹配(Plug-in Post Match)、弱后件匹配(Weak Post Match)、精确谓词匹配(Exact Predicate Match)、通用匹配(Generalized Match)、专用匹配(Specialized Match)等概念，在模块情形下定义了精确模块匹配(Exact Module Match)、通用模块匹配(Generalized match)等概念。在从构件库中检索构件时，用户首先写出查询规约(谓词形式)，然后根据谓词匹配的定义检索构件。当没有精确匹配时，可以利用通用匹配、嵌入匹配等其它定义来检索相似构件。如对于前后件匹配，假设构件的规约为 S，用户需求规约为 Q，定义精确前后件匹配为：(Q_{pre} R_1 S_{pre})R_2(S_{post} R_3 Q_{post})，其中 S_{pre}，S_{post}分别表示构件规约的前件和后件，Q_{pre}，Q_{post}分别表示需求规约的前件和后件，R_1与 R_3为等价连接符号(⇔)或蕴含连接符号(⇒)(没有必要相同)，R_2为合取(∧)或逻辑蕴含运算。又如通用谓词匹配定义为：(S_{pred} R Q_{pred})，其中 S_{pred}为(S_{pre}⇒S_{post})，Q_{pred}为(Q_{pre}⇒Q_{post})，R 为等价连接符号(⇔)或蕴含连接符号(⇒)。精确前后件匹配定义为：(Q_{pre}⇔S_{pre})∧(S_{post}⇔Q_{post})，精确谓词匹配定义为(S_{pre}⇔S_{post})。用户构件检索时，首先写出所需构件规约，然后提交。构件检索系统用一个类似于定理证明器的程序按照具体的检索策略(即是精确前后件匹配还是精确谓词匹配等)进行检索。毋庸置疑，文献提出的这种检索方法将会具有很高的准确率，但其缺点也是形式化方法所共有的，即代价较高。

2.2 基于人工智能的方法

文[10，17，25]讨论了基于框架(frame-based)的软件制品的表示与推理问题。框架是由 Minsky 与1975年提出用于知识表示与推理的一种人工智能方法[34]。一个框架是描述一类对象属性的一种数据结构，由一组称为槽(slot)的结构构成，一个槽又可以由许多刻面(facet)组成(也可以不分刻面)，一个刻面(当把一个槽细分为刻面时)或一个槽(当槽不分刻面时)的取值范围为一个集合。框架可看作面向对象程序设计中类概念的一种扩展，有许多类似于面向对象程序设计中的概念，如子框架与父框架的继承关系、框架实例(instance)等。当利用已有软件制品开发应用系统时，由开发者首先定义需求框架 F，然后以 F 为目标在构件库中进行搜索(构件库中的构件以框架表示)，看是否存在与 F 匹配的构件框架(完全匹配或部分匹配)；若需要构件库中的几个框架来组合表示需求框架时，则利用已有的框架来创建一个新的框架。框架推理主要是利用框架匹配进行，即把用户需求的框架与库中已有框架进行比较，看是否存在能与需求框架相匹配的框架。

文[35]讨论了利用构件库的软件模块和半自动化工具进行规约精炼(specification refinements)和程序自动产生的一种方法。该文的基本思想是利用定理自动定理的有关方法。首先用谓词逻辑来描述构件，即用自动定理证明中的子句概念来描述构件的各个属性，如该构件所能完成的功能、输入输出数据类型、返回值类型、构件位置、所使用的算法等。用户利用该系统提供的工具写出所需软件构件的规约，然后由系统利用谓词演算和二阶谓词演算中的合一算法(unification algorithm)将该规约的相关部分与构件库的模块进行匹配。若匹配成功，则返回实现该规约的构件；若没有单个构件匹配，则返回能组合完成该功能的构件组合。

2.3 其它方法

文[16]描述了在一个构件库中没有单个构件满足查询条件时通过搜索构件库测试是否能通过几个构件的组合来满足用户的需求的方法。用户的需求通过输入输出对〈I，O〉来描述(I 代表输入，O 代表输出)。当构件库中没有单个构件满足条件时，则返回以 I 作为输入并以 O 作为输出的几个构件的组合。例如，假设构件库中的所有构件都是以单个参数作为输入的函数构件 f_1，f_2，…，f_n，则检索过程如下：首先对所有的1≤i≤n，测试是否存在 i，使得 $f_i(I)=O$；若没有，则对所有的1≤i，j≤n，测试是否存在 i，j，使得 $f_i(f_i(I))=O$；若没有，再继续进行类似测试。文献表明，对合成深度为 d 的合成来说，总的合成数目为 $O(n^{(n^d)})$。然而，一些技术可以用来减少合成的数目而不影响最终的结果。如类型兼容性要求就可以排除一部分组合的发生。如对某个 i，j，若函数 $f_i()$的输入参数为 int 型，而 $f_i(I)$的输出不为 int 型，则可以不用测试组合 $f_j(f_i(I))$。作者还对含有161个 LISP 函数构件的构件库进行了实验研究，发现在采用了一定的限制条件以后，组合的可能性大大减少。但该方法的一个主要限制是计算复杂性问题。文[31]表明，NP 完全问题之一集合覆盖问题(set cover problem)可以归约到此问题。再就是该方法只能应用于可执行的构件(Executable Component)。

因为文[16]提出的方法只能应用于可执行的构件，文[36]在文[16]的基础上提出了一种改进的构件检索方法，即基于采样(sample)的概念格(concept lattice)方法。该方法不需要构件执行，而是由构件的开发者对每一个构件选用一些典型的实际数据作为输入，然后得到该构件的输出数据。用输入数据、输出数据及返回类型作为构件的一个采样。设构件库中的构件集合为 C＝{c_1，…，c_n}，设采样集合为 S＝{s_1，…，s_m}，定义二元关系 R 为 C×S 上的子集，R＝{(c，s)|c∈C，s∈S 且 s 是 c 的一个采样}，则三元组(C，S，R)为一个构件库的形式化环境(formal context)。(C_i，S_i)称为一个概念，其中 C_i⊆C，S_i⊆S，且∀ c∈C_i，∀ s∈S_i，s 是 c 的一个采样。概念件的偏序关系≤定义为(C_1，S_1)≤(C_2，S_2)当且仅当 C_1⊆C_2(或等价地 S_1⊆S_2)。两个概念件的最大下界 ∧ 定义为(C_1，S_1)∧(C_2，S_2)＝(C_1∩C_2，{s|s∈S 且对所有 c∈C_1∩C_2，(c，s)∈R})；两个概念间的最小上界 ∨ 定义为(C_1，S_1)∨(C_2，S_2)＝({c|c∈C 且对所有 s∈S_1∩S_2，(c，s)∈R}，S_1∩S_2)。所有概念形成一个完备格(complete lattice)。这样就把一个软件构件库构造成一个概念格结构并可以用概念分析技术来检索构件。在检索时，假设用户提交的预期采样(即用户以选定的数据作为输入数据，并以其想要的输出作为输出数据，以及返回值类型来构造的采样)为 s_1，s_2，…，s_k，则构件检索过程将返回

具有采样 $s_1, s_2, \cdots, s_k$ 的构件。

2.4 构件检索的其它考虑

除了上述主要关注构件表示与检索的技术问题外，还有一些文献研究了构件检索所遇到的其它一些问题。如文[42]提出了应把主动信息发送服务(Active Information Delivery)集成到构件库中，因为许多程序员不知道构件的存在而重新开发构件是软件复用能得到实际应用的一大障碍。主动信息发送是通过所谓手边任务相关性(Relevance to the Task Hand)及用户相关性(Relevance to the User)来识别用户可能需要的构件并为用户发送该构件的有关信息。构件库有一个监听进程(Listener)，手边任务相关性是指当用户写好一个构件的基调或注释时，监听进程则根据基调与注释在构件库中查找是否有相同或类似基调和注释的构件。因为基调构成了程序的语法接口信息，而注释则对程序的理解及反映程序的某些概念有一定意义。用户相关性则是为每一个程序设计人员提供一个用户文件(User Profile)来记录用户曾经用过的构件，因为调查发现程序设计人员一旦复用过某一个构件以后，就倾向于用同一个构件来完成同样或相似的任务。

因为许多构件属于不同的领域，与这些构件相关的文档也形成了不同的数据源。如何访问这些数据源来获得有关构件信息，也是值得注意的问题。文[4]对此进行了研究。该文通过领域本体(Domain Ontology)建立一个屏蔽各种具体数据源的中间层(Mediator)，中间层为构件用户和构件提供者提供公共接口来方便构件检索。

2.5 实例研究

基于文本和基于词法描述子的构件编码与检索方法的实例研究可见文[12,29]。基于规约的构件表示与检索方法现在基本上仍处于理论研究阶段，实验研究也仅仅是小规模的。

3 发展方向

形式化构件表示与检索方法虽然优点是明显的，即能完整描述构件的计算语义、检索的准确率比非形式化方法高等，但其缺点也是显然的，如对程序设计人员的数学能力要求较高，程序设计人员也需要足够的耐心写出构件的规约，更重要的一点是：没有足够的证据表明写一个所需构件的规约所需要的时间比写出这个构件本身所需的时间少或更容易[30]。形式化方法究竟对大规模软件开发是否适用，虽然理论界与工业界之间对此一直存在着争议(参见文[1])，但是毕竟形式化方法在对安全性要求较高的系统(如实时系统)开发中发挥了重要作用[9,15]。就形式化方法在软件构件表示与检索中的应用而言，还可以在以下方面展开工作：新的规约语言的研究，特别是如何做到尽量使大量的数学概念与符号对用户透明，不要使一般的程序设计人员望而生畏，这一点非常重要；新的构件表示模型与检索方法的研究；面向特定领域的软件构件表示、组织和检索；现有编码和检索方法的改进；构件检索的自动化工具研究与开发；构件合成技术；实际应用和经验报告等。

参考文献

1 Astesiano E, Reggio G. Formalism and method. Theoretical Computer Science, 2000, 236: 3～34

2 Ayed R B, Desharnais J, Frappier M, Mili A. A calculus of programming adaption and its applications. Sci. Comp. Prog., 2000, 38: 73～123

3 Basili V R, Briand L, Melo W. How Reuse Influence Productivity in Object-Oriented Systems, Comm. ACM, 1996, 39(10):104～116

4 Braga R M M, Mattoso M, Werner C M L. The use of mediation and ontology technologies for software component information retrieval. In: Proc. 2001 Symposium on Software Reusability, "Putting Software Reuse in Context", May. 2001, Toronto, Ontanio, Canada. ACM Soft. Eng. Note., 2001, 26(3): 19～28

5 Chang C L, Lee R C T. Symbolic Logic and Mechanical Theorem Proving. New York: Academic Press, 1973

6 常继传，郭立峰，马黎. 可复用软件构件的表示和检索. 计算机科学，1999，26(5)：45～49

7 Chen Shicheng P S, Hennicker R, Jarke M. On the retrieval of reusable components. In: the Second Int'l Workshop on Software Reusability, Advances in Software Reuse, Lucca, Italy, Mar. 1993

8 Cheng B H C, Jeng J J. Reusing analogous components. IEEE Trans. Know. Data Eng., 1997, 9(2): 341～348

9 Clarke E M, Wing J M, et al. Formal methods: state of the art and future directions. ACM Comp. Surv., 1996, 28(4): 626～643

10 Devanbu P, Brachman R, Selfridge P, Ballard B. LaSSIE: A Knowledge-Based Software Information System, Comm. ACM, 34(5), 1991:34～49

11 Frakes W B, Gandel P B. Representing reusable software. Inform. Soft. Tech., 1990, 32(10): 653～664

12 Frakes W B, Pole T P. An empirical study of representation methods for reusable software components. IEEE Trans. Soft. Eng., 1994, 20(8): 617～630

13 Frakes W B, Succi G. An industrial study of reuse, quality, and productivity. The Journal of systems and Software, 2001, 57: 99～106

14 Guttag J V, Horning J J. A larch shared language handbook. Science of Computer Programming, 1986, 6: 135～157

15 Hall A. Seven myths of formal methods. IEEE Software, Sept. 1990. 11～19

16 Hall R J. Generalized behavior-based retrieval. In: Proc. of 15th Intl. Conf. on Software Engineering, Baltimore, USA. 1993. 371～388

17 Henninger S. An evolutionary approach to constructing effective software reuse repositories. ACM Trans. Soft. Eng. Meth., 1997, 6(2): 111～140

18 Hoare C A R. An Overview of Some Formal Methods for Program Design. IEEE Computer, Sept. 1987. 85～91

19 Jeng J J. Applying Formal Methods to Software Reuse: [Ph. D. Thesis]. Michigan State University, 1993

20 Jeng J J, Cheng B H C. Specification matching for software reuse: a foundation. In: Proc. of the ACM SIGSOFT Symposium on Software Reusability (SSR'95), Seattle, Washington, USA, 1995. 97～105

21 Jeng J J, Cheng B H C. Using automated reasoning to determine software reuse. Int. J. Software Eng. and Knowledge Eng., 1992, 2(4): 523～546

22 Jeng J J, Cheng B H C. Using formal methods to construct a software component library. Lecture Notes in Computer Science, 1993,717:397～417

23 Jeng J J, Cheng B H C. A formal approach to reusing more general components. In: Proc. of Ninth Knowledge-Based Software Engineering Conf. Monterey, CA, USA,1994. 90～97

24 Jilani L L, Desharnais J, Frappier M, Mili R, Mili A. Retrieving software components that minimize adaptation effort. In: Proc. of 12th IEEE Intl. Conf. on Automated Software Engineering, Incline Village, NV, USA. 1997. 255～262

(下转第113页)

(4)企业应用框架与中间件框架集成的关键。①按照CORBA标准,采用IDL/CIDL对构件进行描述,设计领域构架、领域构件、业务构件及领域知识库中反映热点变化的热点构件和基于设计模式的热点子系统,并将其纳入接口库中,统一到CORBA环境下,便于统一管理;

②针对ADL现状(没有基于构件标准的ADL),自行设计基于CCM的ADL,以便充分利用CORBA机制的优点;

③设计ADL编译器,经过ADL编译器即可将ADL变成可执行系统。

总之,体系结构是构件的管理者、组织者和领导者,它就像乐队指挥,用它那充满魔力的手,将乐队的成员——构件调动起来,按业务流程运转,完成系统的各个功能,实现系统的重构和集成。

结论 本文从代表构件技术最新发展方向的业务构件入手来研究RRIS框架,角度新颖。对RRIS框架的整体设计采用"纵向分层,横向分割"的指导思想,不同级别的抽象可以化解问题的复杂性。对企业应用框架的设计是本文的关键,"黑白"混合式的设计策略,兼顾理想和现实两方面的考虑,是一种较实际的解决方案。框架是半成品的应用程序,其设计与实现是比较复杂的,限于篇幅,本文归纳性地阐述了框架的整体与部分的关系以及框架层间的集成。

参考文献

1 Kozaczynski W. Architecture framework for business components. In: [C] 5th Intl. Conf. on Software Reuse, computer socity press, 1998. 300～307

2 Baster G, Konana P, Scott J E. Business Components: A Case Study Of Bankers Trust. Communications of the ACM, 2001, 44 (5)

3 Fingar P. Component-based Frameworks For E-Commerce. Communications of the ACM, 2000, 43(10)

4 Kobryn C. Modeling Components and Frameworks With UML. Communications of the ACM, 2000, 43(10)

5 Chan S M, Lammers T L. Reusing a Distributed Object Domain Framework. In: [C] 5th Intl. Conf. on Software Reuse, computer socity press, 1998

6 Lu Jian, Li Yingjun, Ma Xiaoxing, Tao Xianping. A Hierarchical Framework For Parallel Seismic Applications. Communications of the ACM, 2000, 43(10)

7 何克清,应时,田中茂,等.业务应用软件框架的一种分析方法.软件学报,12(7)

8 李景峰,刘西洋,陈平.一种可重用构件模型——类属构件.计算机科学,1999,26(9):83～86,80

9 符进强,汪洋,钱乐秋.基于动态构件框架的构件演化.计算机科学,2001,28(1)

10 刘晓铭,刘积仁,李华天.构件化领域框架设计与实现.计算机研究与发展,1999,36(2)

11 Butler G. Object-Oriented Frameworks. http://www.cs.concordia.ca/～faculty/gregb

12 万麟瑞,李绪蓉.系统集成方法学研究.计算机学报,1999,22(10)

13 李群明,张士康,王成恩,宋国宁.可重构的企业管理信息系统. http://sy.863cims.net/lab2

14 Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall, 1996

15 沈延森.快速可重构信息系统极其关键技术研究:[南京航空航天大学博士学位论文]. 2001.5

(上接第102页)

25 Karimi J, Zand M K. Asset-based system and software system development-a frame-based approach. Information and Software Technology, 1998, 40: 69～78

26 Knight K. Unification: a multidisciplinary survey. ACM Computing Surveys, 1989, 21(1): 93～124

27 Krueger C W. Software Reuse. ACM Comp. Surv., 1992, 24 (2): 131～183

28 刘叙华.基于归结方法的自动推理.北京:科学出版社,1994

29 Mili H, Ah-Ki E, Godin R, Mcheick H. Another nail to the coffin of faceted controlled-vocabulary component classification and retrieval. In: Proc. of Symposium on Software Reusability'97 (SSR'97), ACM Software Engineering Notes, 1997, 22(3): 89～98

30 Mili H, Mili F, Mili A. Reusing software: issues and research directions. IEEE Trans. Soft. Eng., 1995, 21(6): 528～561

31 Mili H, Marcotte O, Kabbaj A. Intelligent component Retrieval for software reuse. In: Proc. of 3rd Maghrebian conference on artificial intelligence and software engineering. Rabat, Morocco, 1994. 101～114

32 Mili R, Mili A, Mittarmeir R T. Storing and retrieving software components: a refinement based system. IEEE Trans. Soft. Eng., 1997, 23(7): 445～459

33 Mili R, Raymond J. Towards a formal framework for software reuse. Inform. Sci., 1998, 110: 135～149

34 Minsky M. A framework for representing knowledge. In: Psychology of Computer Vision (Winston P H eds.), McGraw-Hill, 1975

35 Nishida F, Takamatsu S, Fujita Y, Tani T. Semi-automatic program construction from specifications using library modules. IEEE Trans. Software Engineering, 1991, 17(9): 853～871

36 Park Y. Software retrieval by samples using concept analysis. The Journal of systems and Software, 2000, 54: 179～183

37 Rine D C, Sonnemann R M. Investments in reusable software. A study of software reuse investment success factors. The Journal of systems and Software, 1998, 41: 17～32

38 Waldmann U. Semantics of order-sorted specifications. Theoretical Computer Science, 1992, 94: 1～35

39 王家兵,徐正权,王能超.RLD演绎及子句蕴涵与子句包含关系的非等价性.计算机研究与发展,已录用

40 Wing J M. Writing larch interface language specifications. ACM Trans. Programming Languages and Systems, 1987, 9(1): 1～24

41 杨芙清.软件复用及相关技术.计算机科学,1999,26(5):1～4

42 Ye Yunwen, Fischer G, Reeves B. Integrating active information delivery and reuse repository systems. In: Proc. of the ACM SIGSOFT Eighth Intl. Symposium on Foundations of Software Engineering, San Diego, Calif., ACM Soft. Eng. Note., 2000, 25 (6): 60～68

43 Zaremski A M, Wing J M. Signature matching: A tool for using software libraries. ACM Trans. Soft. Eng. Meth., 1995, 4(2): 146～170

44 Zaremski A M, Wing J M. Specification matching for software components. ACM Trans. Soft. Eng. Meth., 1997, 6(4): 333～369