计算机科学1998Vol.25№5

基于可重用构件库的软件重用技术

On RCL-based Software Reuse Technology

何志均　王少锋

（浙江大学人工智能研究所　杭州310027）

摘　要　In this paper, the RCL-based software development model is divided into six steps: identifying, qualifying, classifying, retrieving, customizing, and composing. According to these steps, we give a survey of the technology used in the RCL-based software development paradigm in detail. We also discuss the relationship of OO, Internet and software reuse technology.

关键词　Software reuse, Reusable component, OO, Internet

一、引言

1968年 McIlroy 在 NATO 软件工程会议上首次提出了软件重用的思想。1983年 Freeman 又进一步拓广了软件重用的概念，指出可重用的构件不仅可以是源代码片断，还可以是模块、设计结构、规格说明和文档等，而且不仅可按组装方式重用，还可按模式重用。目前软件重用已成为软件工程领域中的一个研究热点[1,2,3,4,5,6]。

采用软件重用技术至少具有以下几点好处：1）提高软件生产率；2）提高软件质量；3）降低开发风险；4）减少开发时间和费用；5）开发的软件系统易于维护和理解；6）增加系统的可靠性；7）易于提供文档资料。Schach 也认为软件重用不但能降低开发费用，而且能大幅度降低维护的费用，且效果更明显，一般说来，采用软件重用技术后，在维护阶段节省的费用几乎是开发阶段节省的费用的二倍[7]。另一方面，采用软件重用技术对软件质量也有很大的提高，Lenz 等人报告在功能测试时，每行平均错误数比不使用重用的少9倍，在部件和系统测试时大约要少4.5倍[8]，W. Tracz 认为软件重用技术在大的应用项目中比在小的项目中效益更明显，花费更省[9]。

二、基于可重用构件库的软件开发模式

采用软件重用技术会对传统的一些软件开发模式产生深刻的影响，目前软件重用所使用的技术很多，但使用得比较多的还是基于可重用构件库 RCL (Reusable Component Library)的方法。目前已提出了多种基于 RCL 的软件开发环境，如 RESOFT 原型系统[10]，该系统用 Eiffel 实现，运行在 UNIX 环境下，利用了 Eiffel 中的 OOP 和数据永久性特性，Moineav 等人介绍了 ESF-ROSE 项目[11]，该项目的目的是为了建立一个通用的、可扩充的、支持软件重用的开发环境，其它的还有 ROSE-2[12]，PROTEGE-II[13]，REBOOT[14]等，这些系统虽然采用不同的开发步骤，但基本上都是以 RCL 为中心的开发模式，如图1所示。

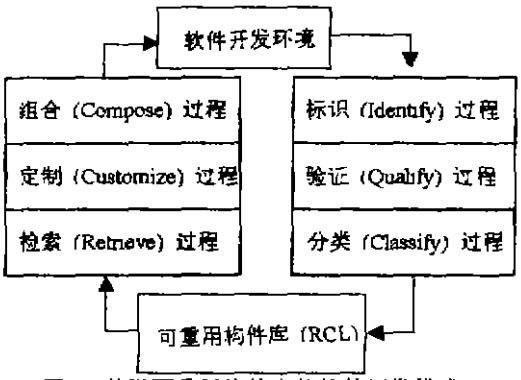

图1　基于可重用构件库的软件开发模式

一般地，可重用构件 RC(Reusable Component)在入库前须经过标识(Identify)、验证(Qualify)和分类(Classify)三个过程，而从库中提取可重用构件则

何志均　教授，博士导师，主要研究领域为人工智能、软件工程等。王少锋　博士生，主要研究领域为软件工程、智能化软件开发环境、软件重用。

须经过检索(Retrieve)、定制(Customize)和组合(Compose)三个过程,在软件开发过程中需不断地对RCL进行存储、检索、提取等操作。本文以下部分即按这六个过程对基于RCL的软件重用技术作具体的论述。

2.1 标识过程

标识是指寻找RC的候选者,它是进行软件重用的前提,W. Tracz给出了RC应具有的一些性质:遵循标准的规定、有良定义的界面、高可靠性、较好的通用性、强鲁棒性、易理解性、高内聚低耦合性等[9]。目前已提出了许多方法来对RC进行标识。G. Canfora等人提出了一个算法对程序代码中的可重用对象进行定位[15],该算法使用了统计技术,可以在过程性程序中标识潜在的RC。Caldrien和Basili也提出了一种标识可重用构件的方法,他们把标识可重用构件分为三个阶段:1)定义可重用属性模型;2)从已存在系统中抽取构件;3)根据可重用属性模型对抽取的构件进行度量[16]。W. Wieczerzycki把同一个RC的不同版本一起作为可重用对象[17],一个程序由一个程序体和一系列的程序包组成,程序体包含全局函数和数据结构,而每一个程序包由可变的局部函数组成,程序执行时只有一个程序包是活动的,具体是哪个程序包是活动的可在程序运行时动态决定。Dunn等人给出了一个基于专家系统的模型,用于标识已存在的软件中的可重用构件[18],并描述了一组工具集Code Miner,实现了一部分模型,该工具集利用了Prolog中内在的推理机制,可用来帮助程序员在C语言程序中标识RC,找到的RC再由专家作进一步的检查。

我们认为上述各种标识RC的方法虽然在某种程度上有助于发现可重用构件的候选者,但并不能从根本上解决问题,RC并不是软件开发结果的副产品,如果开发组织准备采用软件重用技术,应在开发的整个过程中强调软件重用,包括在开始开发时就要考虑构件的可重用性。Incorvaia等人在对一些大公司和机构进行的一项关于软件重用的调查表明,为了构造RC,必须在项目开始之初就要考虑重用,虽然构件的可重用性可以在构件建立之后再作翻新改进,但这样做费用非常昂贵[20]。W. Tracz也认为翻新时易于出错且费时间,因此不应鼓励[21]。

2.2 验证过程

由于RC将会在多个应用中使用,因此RC的可靠性、正确性等是非常重要的,一个RC在入库前需经过严格的测试和审查,我们把这一过程称为验证过程。

目前对RC的验证基本上还是采用手工的方法,较少有工具的支持,虽然已有一些测试工具可用,但对于特定的RC来说,这些工具的效率还不高,Caldriea和Basili把RC的验证细分为六个步骤[6],但该方法使用起来也不甚方便。

我们认为,验证过程涉及到很多管理方面的问题,为了提高RC的可靠性和正确性,可以采取管理上的一些措施,如增加专门的测试组,专职的RC开发小组,从开发的开始就进行RC设计而不是在以后才对原有的系统进行RC的抽取等,如何从管理学的角度对软件重用提供支持已有一些文献作过论述[22,23,24,25]。

2.3 分类过程

分类过程是RC入库前的最后一个过程,分类是指对RC的功能、使用方法、适用范围、接口等进行说明性的描述。对RC进行分类主要是为以后的检索过程作准备,好的分类方法应具有描述能力强、准确、易于使用、易于理解、检索效率高等特点。

分类方法在计算机的许多领域中都有应用,而且研究得也比较多。经典的分类法有关键字法、树形分类法等,但用关键字法检索存在二个问题,第一是该方法虽然对于有经验的用户来说效果较好,但对于初级用户来说,由于不清楚哪些关键字是适当的,哪些是不适当的,故可能会使用一些同义词或其它相关的词,因此效果明显下降;第二是采用关键字法有"关键字障碍"现象[26],即检索的性能到达某一阈值后将不能再提高。CodeFinder采用树形层次结构对RC进行分类[27,28],Wang和Brereton采用了一阶谓词逻辑的方法来描述RC[29],他们把软件实体作为二元组(Software_Item,Item_Descriptor)来处理,其中Software_Item是RC的具体物理实体,而Item_Descriptor是表示该构件与重用相关的信息集合,London用Z语言来描述建立在Smalltalk类库上的RC[30],London认为采用Z语言说明作为RC的规格说明部分有助于理解RC。Prieto-Diaz和Freeman提出了刻面(facet)分类法[31],该方法用多个刻面描述给定构件的特征,其中每一个刻面刻画构件的一个性质。C. Gacek对刻面分类法又进行了一定的扩充和改进[32],使每个刻面和多个属性相联系,并且根据不同的应用领域定义了各刻面之间的优先级。Chou等人采用基于行为的语义网络对可重用的OO规格说明进行分类[33],根据规格说明的行为抽取出它的语义网络,然后再根据语义网络对其进行分类。Maarek等人是采用给自然语言描述加索引的分类方法[34],Girardi和Ibrahim也使用自然语

言(英语)描述的方法来检索 RC[35]。

分类的主要目的是对检索提供支持,一般地,RCL 中包含的 RC 越多,分类方法的好坏越重要,评价一种分类方法的好坏应以其使用的效果来考虑,任何一种分类方法只有在特定的系统中,在特定的检索过程中使用才能表现出它的优势,但目前还没有一种分类方法能在所有的系统中都表现出很好的性能[36],往往是某一分类方法对某些检索方法的支持较好,而对另一些检索方法的支持就不令人满意了,今后的研究方向将是支持多种检索方法的 RC 分类方法。

2.4 检索过程

检索过程是软件重用活动中最重要的一个过程。目前软件重用技术还没有得到广泛的推广应用,虽然其原因很多,但缺乏有效的检索方法及工具的支持是其中的一个重要原因。好的检索工具可以大幅度提高重用效果[20],Woodfield 等人发现软件开发人员有70%阈值现象[37],即重用的工作量若高于常规方法工作量的70%,则软件开发人员会使用常规方法(直观上阈值是100%),因此,好的检索方法对于成功的软件重用是极为重要的,若检索过程太低效,或检索得到的 RC 非所需要的,必会使软件开发人员放弃重用的努力。

Prieto-Diaz 和 Freeman 用伪代码算法来说明检索 RC 的过程[31]:

```
begin
  search library
  if identical match
    then terminate /* 已找到可用构件 */
    else
      begin
        collect simular component as a set
        for each component in the set
        computing degree of match
        end for
        rank and select the best component
      end
  end if
end
```

目前已提出的检索方法很多,S. Henninger 做了一个工具 CodeFinder 来帮助检索 RC[27,28],即使用户对 RCL 不熟悉而给出的是非良定义(ill-defined)的检索要求,CodeFinder 也能通过增量式的求精过程得到相关的 RC。C. Fernandez-Chamizo 等人用 Reuse Assistant 来帮助检索 RC[38],Reuse Assistant 由两个子系统组成,分别支持基于统计方法的检索和基于知识库的检索方法,基于统计方法的检索使系统有一定的可扩充性,并可使用自然语言界面,而基于知识库的检索方法使系统有一定的推理能力。Ostertag 等人在构造 AIRS 系统时[39],用一系列的二元组(feature,term)来描述 RC,每个 feature 表示一个分类准则,由多个 term 定义,检索时,根据查询要求和二元组计算每个 RC 的相似度,相似度表示了 RC 和目标的匹配程度。Faustle 等人也提出了通过计算相似性来检索 RC 的方法[40]。Jeng 等人提出了一种形式化的软件可重用构件的匹配方法,他们认为采用形式化的规格说明有利于确定可重用性,因为能更精确地表示软件的功能,并且良定义的语法能使处理过程易于自动化[41],该方法克服了二义性、不完整性和不一致性的问题,但它本身也有不易于使用和理解的问题。Zaremski 和 Wing 采用了签名匹配(signature matching)方法,他们把从构件中抽取的特征信息称为签名,检索时把用户的查询要求和构件的签名相匹配[42,43]。London 则把 Z 语言用于可重用的规格说明的匹配[30]。

一般说来,根据某一查询要求得到的 RC 往往不止一个,因此系统还要有帮助用户进行评估的功能,虽然这种评估的结果主要取决于用户的主观因素,但也已有一些具体的度量方法,Prasad 采用了包容度(subsumption)和紧密度(closeness)来对软件构造过程中的功能组合和功能修改等进行建模,这些概念及相应的度量方法被用来对检索到的 RC 进行评估[44],Gacek 根据各构件刻面预定义的优先级来计算各构件的匹配度[32]。

近几年,Internet 的迅速发展也对软件重用中的检索技术产生了很大的影响,一些机构专门在 Internet 上开发了一些检索工具,如 Onik[45]、GAMS[46]等,而 Poulin 等人采用 HTML 语言和 WWW 上的浏览器 Mosaic 来开发检索 RC 的工具[47]。

判断一个检索方法的好坏的标准很多,如易使用性、可扩充性、效率等,但据 W. Frakes 和 T. Pole 的一项调查结果显示[36],没有一种检索方法能满足所有用户的所有检索要求,较好的解决方案是在一个系统中同时提供多种检索方法以供用户选择,这将是软件重用技术今后的一个重要研究方向。

2.5 定制过程

一般地,检索得到的 RC 和用户的需求可能还有一定的差别,定制过程的主要工作是对检索得到的 RC 进行一定的修改、裁剪以符合用户的要求。有关 RC 的定制方法很多,针对不同类型的 RC 有不同的解决方法。Novak 等人认为 RC 的界面是重用的一个障碍[48],因此用 GLISP 实现了一个界面转换工具,该工具用二个方法解决 RC(子系统)的界面问题:自动生成程序把数据转换成 RC 所需的格式,或重写 RC 的界面以符合数据的要求。

对RC进行定制的工作量除了和所使用的方法有关外，还与RC本身的设计及目标系统有关，一些有经验的设计者可以开发出很易使用的RC，同样，若目标系统的通用性较高，则在其内部使用RC就越容易。

2.6 组合过程

组合过程是指经定制后的RC和其它的软件模块集成在一起，形成新的应用系统。组合过程的关键是构件的接口标准，有了这些标准，软件开发者或厂商就可以独立地开发构件、增值构件或集成构件，而应用系统的开发人员就可把主要精力放在应用系统本身的研究上，显然这种软件构件集成方式将使软件系统的开发周期大大缩短，软件质量大大提高，开发费用进一步降低，而且软件也更易维护，这将是软件开发的主流技术。Sugiyama利用了UNIX中make工具的思想，开发了一个Object Make工具，该工具可把多个RC组合在一起，构造出新的应用系统[49]，而Mary Shaw认为不同的应用应该有不同的RC组合方式，可以用软件体系结构作为对RC进行组合的框架结构[50]，C. Gacek采用了特定领域体系结构的方法DSSA来达到重用的效果[52]，首先根据不同的领域要求建立领域模型，然后由领域模型选择特定的软件体系结构，最后根据体系结构的规格说明对RC进行组合，而J. Purtilo使用的程序结构已预先定义了程序中模块的描述，模块连接方法等[51]，S. Bhansali用体系结构设计驱动的重用方法，该方法同时利用了代码构件和程序综合技术[52]，Tan和Ling通过OO规格说明的体系结构把多个构件组合在一起，从而达到支持构件重用的目的[53]，这种方法适用于大数据量的商业系统中，D. Libes用程序设计语言Expect把多个交互式的程序组合在一起[54]，用Expect组合成的应用可在网络上运行，可同时处理多用户、多作业的复杂程序，而不需要进行对原程序作核心修改等一些复杂的工作，Guha和Lenat给出一个通讯语言CycL语言，该语言可把多个Agent组合在一起[55]。

由于RC可能已经过一定的修改，且RC是应用在不同的系统中，因此，还需对RC与新开发的构件一起作测试。在这方面今后需进一步研究的内容有：RC和非RC对系统可靠性的影响的分析，如系统的错误数及错误出现在哪类构件中，使用RC和非RC的费用的比较，RC在系统中所占的百分比，对RC和非RC的维护费用的比较等，这些方面的研究对今后的系统开发，对软件工程的认识都有很大的意义，但目前有关这方面的研究还较缺乏。

三、OO技术和基于RCL的软件重用技术的关系

OO技术和软件重用技术有很密切的关系。软件重用的概念于1968由McIlroy在NATO的软件工程会议上提出，其时正是OO语言的鼻祖Simula语言刚刚提出不久[56]，第一次软件重用国际研讨会在1983年召开，当时正是OO技术引起广泛关注的时候，随着OO技术的兴起，软件重用成了软件工程领域中最重要的内容之一。

Pant论述了把OO构件转化为RC所需的工作量及结果[19]，认为该工作量要比把非OO构件转化为RC所需的工作量要少，Bauer在IBM公司建立RC中心的过程中，发现OO语言，如C++，比非OO语言在支持重用方面更有效[57]。

Basili等人研究了软件重用技术对用OO方法开发的系统质量的影响[58]，他们把系统的错误数和系统大小之比定义为密度(Density)，通过对八个用OO方法开发的系统的分析，他们发现使用RC的系统其密度值为0.125，而不使用RC的系统的密度值为6.4，相差约50倍。

虽然使用OO技术可以有效地提高重用，但OO技术对于软件重用来说既不是充分的也不是必要的[59]，两者的研究领域如图2所示。即OO技术和软件重用技术两者有相互重叠的部分，但不存在一个包含另一个的关系。

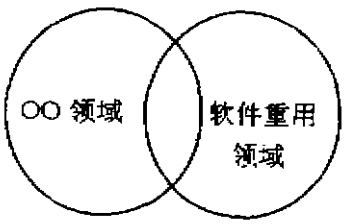

图2 OO研究领域和软件重用研究领域的关系

四、Internet技术和基于RCL的软件重用技术的关系

近几年，Internet的发展非常迅速，Internet的出现对软件开发方法，包括软件重用技术的影响很大，已引起越来越多人的重视[45,46,60]。在图1中，我们把基于RCL的软件开发模式分为六个过程，而Internet对这几个过程都有很大的影响。

我们知道，Internet的最大特点是其丰富的网上资源、方便的通讯方式、众多的网上用户等。传统

上认为，高昂的建库费用和缺乏有效的检索工具是阻碍软件重用的二个因素，但在 Internet 上这二个问题将得到缓和。首先 Internet 上有许多免费的可重用构件库，原先是由某一机构开发 RCL，RC 的分类、归档需花费昂贵的费用，当某一 RC 修改时，需重新分类和归档，这种工作极为繁琐，且易于出错，这也是软件重用难以推广的原因，而现在有许多外部的 RCL 可供使用，因此即使开发机构内部无 RCL 也可进行软件重用。其次对 RCL 的搜索也可以依赖于外部的信息提供者，Internet 上有许多方法来检索可重用构件，当在 Internet 发布了一个需求时，往往会得到许多反馈信息，这些信息往往有助于我们找到可重用的构件，其结果往往比亲自搜索还要好。

一些机构根据 Internet 的特点专门开发一些检索工具，Gertz 等人开发了 Onika 以在 Internet 上搜索软件模块[45]，Onika 能通过 Internet 或 WWW 检索远地站点的 RC，并能把这些 RC 和本地的 RC 集成在一起。Boisvert 用了 GAMS 系统在分布式计算环境，即分布在 Internet 上的多个 RCL 中寻找 RC[46]，作者给出了 GAMS 的实现原理，使用方法等。Poulin 等人认为用形式化的工具和技术来开发 RCL 需要很大的投资，而使用这些 RCL 还需要专门的训练。由于这些原因，要想从 RCL 中获得效益很难，即使这些 RCL 中包含了许多高质量的构件，因此 Poulin 等人采用 HTML 语言和 WWW 上的浏览器 Mosaic 来开发检索 RC 的工具，其开发、维护费用仅为采用常规方法费用（80－130人年）的 1%[47]。

目前，在 Internet 上搜索可以在短时间内搜索大量的 RCL，其 RCL 的数量是以前单个开发机构的 RCL 数量无法相比的，相差达好几个数量级。当一个系统的参数值的变化达到几个数量级时，该系统的性质往往会发生根本性的变化，Internet 也一样，巨大的网上资源可以保证网上存在我们所需要的构件，但检索这些构件的方式是和传统方法不同的。

当然，由于 Internet 的迅速发展还是近几年的事，有关的网上软件电子交易的法律，商业规范还不健全，因此 RC 的质量、安全等一些问题还无法保证，一些安全性要求极高的系统还不宜使用这类 RC，但随着 Internet 的逐步成熟，这些问题会逐步解决，可以说，Internet 的迅速发展对于软件重用技术来说既是一个机会也是一个挑战。

结束语 软件重用就好像"站在前人的肩膀上"，能大幅度提高软件生产率、降低开发成本、保证软件质量，是解决软件危机的一个很有效的方法，但目前软件重用还没有在工业界得到全面的推广应用，其原因很多，涉及到管理、心理、经济、社会文化、法律等方面的问题。从技术角度讲，软件的重用面临着通用性与专用性、构件的分类与检索、理解、修改以及其它潜在的问题，今后，领域分析、过程重用、采用软件体系结构的重用方法、大粒度重用、构件程序设计方法学、重用和软件工程的其它领域的相互关系等将会是研究的重点。（参考文献共60篇略）

（上接第126页）

常规的合取式（假设后置断言已经化为 Skolem 范式），每一个合取式再分解成一组类似 Prolog 的 Horn 子句：

$$a1 \wedge \cdots \wedge an \rightarrow b1 \vee \cdots \vee bm$$

分解成：

$$a1 \wedge \cdots \wedge an \rightarrow bi (i=1, \cdots, n)$$

然后对各组 Horn 子句进行排列组合，则形成一组类似 Prolog 的程序。如果对于前置条件这组程序中有（至少）一个经过执行后的结果为真，那么我们认为在这样的前置条件下，后置断言为真。在规范中，总是存在特殊的谓词，例如"a is father of b"，所有这种谓词必须在 Where 部分有结构化的定义，否则无法执行。为了更好地求值，我们规定这些谓词定义成类似 Prolog 式子的递归函数。对于问题(b)，只要把每次对 PRE-POST 断言的求值看成是一次过程调用，处理好参数值的传递即可。

结论 上述方法不但可以用于逐步求精技术中检查相邻的两步之间的语义一致性，还可以认为是一种基于形式语义的速成原型的新方法，因为在这种方法中，每一规范都可以看成是一个原型。近年来，使用形式语义作为速成原型的基础被认为是可行的方法。我们的方法有如下优点：(1)这种速成原型的方法直接和产生高效的可执行程序的过程相联系。(2)通过 PRE-POST 断言，这种处理方式又和验证相联系。这种把验证和速成原型相结合的结果可以看成是把动态语义和静态语义结合在一个统一的形式化框架下的产物。

但是，我们的工作还存在缺陷。对于该方法中的 PRE-POST 断言的处理，依旧存在效率的问题。分解出的 Horn 子句组很多时，对它们进行排列组合是个 NP 问题，导致搜索效率不高。如何高效地搜索解空间是我们下一步要解决的问题。（参考文献略）