构件提取技术综述*)

罗 景 张 路 孙家骕
(北京大学信息科学技术学院软件所 北京 100871)

摘 要 随着软件构件技术的发展,作为一种有效的构件获取手段,从现有系统中提取构件因其成本与效率优势而成为软件复用与程序理解领域的重要交叉研究领域。针对构件提取的基本步骤,本文从系统分解和度量两个方面对构件提取的研究现状进行了综述。在此基础上本文还介绍了几个典型的构件提取系统,并对构件提取的研究前景与研究方向进行了展望。

关键词 构件提取,模块独立性度量,模块划分质量度量,遗传算法,聚类

A Survey of Techniques for Extracting Components from Existing Systems

LUO Jing ZHANG Lu SUN Jia-Su
(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)

Abstract Due to the development of the software component technology, as an effective means to acquire component, extraction of components from existing systems has become a research focus of both software reuse and program comprehension for its lower cost and high efficiency. In this paper, we analyze the current status of research on component extraction from system decomposition and its metrics. Then we present several typical component extraction systems. Finally, we outline some possible research directions of this important research topic.

Keywords Component extraction, Module independence metric, Module partition metric, Genetic algorithms, Clustering

1 引言

软件复用被认为是解决软件危机的一条现实有效的途径,它通过利用已有资源,从而提高软件的开发效率与质量[1]。作为软件复用的一项关键技术,构件获取主要存在两种途径:有目的的构件生产和从已有系统中提取[2]。基于现有软件系统的提取方法因其成本低和开发周期短的优势而逐步成为软件复用与程序理解领域的一个重要交叉研究领域。早在20世纪80年代就提出了一些可复用模块的提取方法[3]。90年代以来,构件提取主要是针对面向对象系统,出现了一系列构件提取方法以及相关的度量技术。构件提取可望成为一种高效的软件生产手段,同时它还可以辅助软件系统演化与重构[4]。

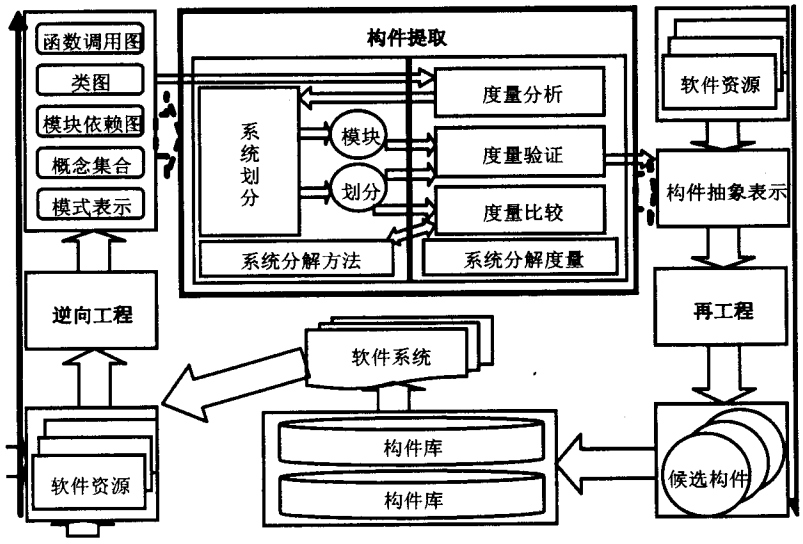

图1 构件提取流程

*)基金项目:国家重点基础研究发展规划项目计划(973计划,编号2002CB312003);国家高技术研究发展计划(863计划,编号2004AA112070);国家自然科学基金资助项目(编号60403015)。罗 景 硕士生,主要研究领域为程序理解,数据挖掘。张 路 博士,副教授,主要研究领域为软件工程,程序理解,配置管理,人工智能。孙家骕 教授,博士生导师,主要研究领域为计算机语言及编译技术,软件工程,软件逆向工程。

从软件过程角度看，构件提取基于现有的软件资源，通过逆向工程[5,6]从中抽取各种抽象形式，其中语义形式包括概念集合，模式表示，变量引用关系，数据传递关系等；结构形式包括函数调用图，模块依赖图，类图，实现实体关系图等。基于这些抽象形式，构件提取能够完成对系统的分解并在相关度量的支持下从中筛选高质量的子模块作为候选构件。然后通过再工程技术对这些候选构件进行一定修改。最后，通过对候选构件的验证、封装和入库而为新一轮的软件开发提供支持。图1给出了构件提取的基本流程。

从以上分析可以看出，构件提取一般包括系统分解与度量两个主要的步骤（见图1中黑框内的部分），二者在实施过程中相互依存，其中度量可用于验证分解结果和筛选候选构件，系统分解方法则可能需要基于度量并以之为一种构件识别手段。同时，随着构件提取的发展，为了对其进行比较，人们又提出了比较度量，它主要用于比较不同构件提取方法的优劣，以及评判现有方法的效果。

本文以这一基本过程为基础，从系统分解方法和系统分解度量两方面进行综述。由于系统分解方法与度量之间的依赖关系，本文采用如下的组织方式，首先介绍现有的系统分解度量方法，我们将其分为局部度量，全局度量和比较度量；接下来通过将现有系统分解方法分为概念匹配与结构分析两大类分别进行综述。在此基础上，我们简单地介绍了几个现有的构件提取系统，最后展望了构件提取研究的前景与方向。

2 系统分解度量

在构件提取中，度量主要应用于构件标识与构件验证，前者用于从软件系统中标识候选构件，后者则用于从候选构件中筛选构件。而在系统分解方法的研究中，度量也可以应用于比较两个不同的系统分解方法，进而作为系统分解方法的比较基准。本文根据度量的应用范围将其划分为局部度量，全局度量以及比较度量三大类。其中局部度量与全局度量主要用于构件验证与构件标识，比较度量则主要用于系统分解方法的比较。

- 局部度量，从系统局部特性出发评价候选构件在语法与语义上的可复用性。
- 全局度量，针对系统的全局划分结果而评价系统整体的系统分解效果。
- 比较度量，以另一种系统分解方法为基础，来评价现有方法或者方案的有效性。

2.1 局部度量

局部度量从候选构件所在系统的局部出发，通过考察其结构与语义特性而评价其可复用性。其中结构特性通常基于类图，函数调用图等抽象形式，它反映了候选构件的内聚度与耦合度；而语义特性则一般通过统计分析方法获取基本度量，比如代码行数、继承层次、类平均方法、类引用数等，进而基于相应的度量模型计算候选构件的可复用性指标。因此在本文中我们将局部度量分为结构局部度量与统计局部度量，结构局部度量如文[7～9]，统计局部度量如文[10～12]，其中模块独立性度量(IM)[8]与构件可复用性度量模型(RMCQ)[10]是这两种形式的代表。一般而言，局部度量简单且高效，可同时应用于构件标识与构件验证，但它没有考虑候选构件对系统整体提取效果的影响。

- 模块独立性度量[8] 用于验证候选构件的可复用性，它是在借鉴QMOOD[13]度量模型中可复用性度量的基础上给出的，图2是相应的度量树。

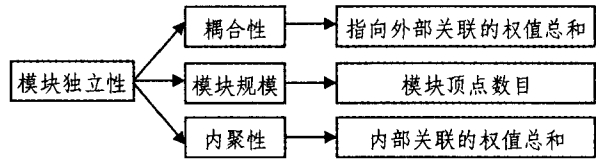

图2 模块独立性度量树

独立性度量与模块的内聚性正相关，与模块的耦合性与模块规模负相关。它存在两种表现形式，可分别应用于不同情况。

定义1 在有向带权图 $G=(V, E)$ 中，对于它的某个子图 $G'=(V', E')$，从 G' 到其补图的关系集合为 $O=\{\langle u,v,t\rangle \mid \langle u,v,t\rangle \in E$ 且 $u\in G', v\in V-V'\}$，子图 G' 的独立性 IM 定义为：

$$IM(G',G)=\frac{\sum_{e\in E'}W(e)}{|V'|\sum_{e\in O}W(e)} \tag{1}$$

$$IM(G',G)=\frac{\sum_{e\in E'}W(e)}{|V'|\sum_{e\in O}W(e)} \tag{2}$$

- RMCQ—构件可复用性度量模型 RMCQ[10]是综合REBOOT[12]、QMOOD[13]模型而提出的，它包含因子、准则和度量三个基本的层次，通过从代码资源中获取相关统计特性，比如，实例化使用数、子类数目、外部调用方法数等，然后根据度量模型进行规约，最终得到一个相对或绝对的可复用性指标。该模型在构件验证中效果良好，但其实施过程也相对复杂。

2.2 全局度量

全局度量从系统整体视图上评价系统分解方案，它能够为系统的划分度量提供支持。但由于其对整体效果的关注，使得在高全局度量指标的情况下，可能由于各个候选构件单独的可复用性都偏低而无法得到可用的候选构件。模块划分质量度量[14]是全局度量的代表。QMOOD[13]，CARE[11]等可复用性度量模型同样可以应用于全局的可复用性度量。

- 模块划分质量度量[14] 通过考察系统划分的全局内聚耦合特性而评价系统分解方案的质量。它认为如果划分方案中的模块具备高内聚低耦合的特性，则划分质量就高。

定义2 对于图 $G=(V, E)$ 的一个划分 $C=(G_1, G_2, \cdots, G_n)$，其中 $G_i=(V_i, E_i)(1\leqslant i\leqslant n)$ 是 G 的一个子图，其质量由以下公式定义：

$$MQ(C,G)=\frac{\sum_{i=1}^{n}s(G_i,G_i)}{n}-\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}s(G_i,G_j)}{n(n-1)/2} \tag{3}$$

在式(3)中，$s(G_i, G_i)$ 表示子图 G_i 的内聚性，$s(G_i, G_j)$ 表示子图 G_i 和子图 G_j 间的耦合性。由此可见，划分中每个子图的内聚性越高则划分的质量越高，而子图间的耦合性越高则划分的质量越低。

2.3 比较度量

比较度量是在已知一种系统分解方案的基础上，评价其他系统分解方案的度量技术，在系统分解方法的比较中被广泛应用，同时它也可以在特定的构件提取上，用于判断某一系统分解方法的效果。因此根据比较度量的应用范围不同，可将其分为模块划分相似度度量与比重度量，前者用于比较两种不同的系统分解方法的优劣，后者则以已有的系统分解方案为基准来评价某一系统分解方法的有效性。

·模块划分相似度度量　针对相同软件系统的两种不同构件提取方案，模块划分相似度度量在不同情形下表现出不同的效果。如果两种构件提取方案分别采用了不同的系统分解方法，则它体现的是两种方法的差异；如果两种方案采用的是同一系统分解方法，则它体现的是该系统分解方法的稳定性；如果其中一种方案是手工获取的精确结果，另一方案采用了某种给定方法，则它体现了该方法的有效性。目前针对模块相似度度量的研究有文[15～18]，其中具有代表性的度量主要有：距离度量－MoJo[16]，边相似度量—EdgeSim[17]，划分聚类相似度度量－MeCl[17]。

定义 3　距离相似性度量(MoJo)指的是将一个划分转换为另一个划分所需节点移动(Move)与模块合并(Join)的次数，它可以由如下公式定义：

$$MoJoQualit\ y(A,B)=\left[1-\frac{MoJo(A,B)}{N}\right]\times 100\% \quad (4)$$

其中 $MoJo(A,B)$ 表示将划分 A 转换为 B 时所需移动节点的次数，N 表示图的规模。

定义 4　针对两个给定的划分，分别计算它们之间的内部边与相连边，二者的交集即为边相似度度量。

定义 5　划分聚类相似度度量(Merge Clusters，MeCl)指的是将一个划分分解后重新合并为另一个划分过程中的变化程度。

在上述三种相似度度量中，第一种度量仅考虑了不同划分中各模块间节点移动的影响，后两种度量则考察了划分中模块间边的关系。实验结果表明，后两种度量往往更加稳定。

·比重度量　指的是一种系统分解方案中的某类元素在另一方案中同类元素的比重，在应用过程中，通常设其中的一种方案作为基准方案，另一种作为考察方案。我们称基准方案中某类元素在考察方案中同类元素中所占的比重为查准率，而考察方案中的某类元素在基准方案中同类元素所占的比重为查全率。根据其考察元素的区别，又可以分为：模块比重度量与相关对比重度量[18]，其中模块比重度量考察的元素为划分中的模块，该度量因其对模块要求的精确匹配使得其度量值偏低，在实际中，主要应用于一种方案为手工精确划分的情况。相关对比重度量中的考察元素是划分中的相关对。

定义 6　划分中某一模块内部的两个节点构成相关对。对于两个划分 A，B，Precision(查准率)表示 A 中的相关对在 B 的相关对中所占的比重；Recall(查全率)表示 B 中的相关对在 A 的相关对中所占的比重。

2.4　分析比较

在上述系统分解度量中，模块独立性度量用于评价划分结果中一个模块的可复用性，它保证了局部的最优性，但忽略了其对整体效果的影响；模块划分质量度量用于评价一个划分的整体质量，但它忽略了单个候选构件的可用性；模块划分相似度度量在不同情形下表现出不同的效果，主要用于系统分解方法的比较；比重度量根据一种系统分解方案中的某类元素在另一方案中同类元素的比重来进行方法的比较。表 1 给出了上述几类度量的比较。

表 1　系统分解度量比较

	代表性技术	优点	缺点	应用
局部度量	IM[8]	简单，可以获取复用性高的候选构件	缺乏全局考虑，不支持划分	在构件提取过程中用于标志，筛选构件，也可用于评价提取方案
全局度量	MQ[14]	全局特性，可以获取系统的完整划分	忽略了局部最优性	系统分解方案的实现与评价
比较度量	MoJo[16]	应用于不同系统分解方法与方案的比较	缺乏比较基准	用于比较方法或者验证方法的有效性

3　系统分解方法

3.1　系统分解方法的分类体系

从现有文献看，系统分解方法大致可分为两类：知识匹配和结构分析。知识匹配方法以软件系统的相关知识为驱动，将其与系统的构成成分进行匹配，匹配的结果即为候选构件。结构分析方法则基于某种软件系统的抽象结构，比如树或图，我们将其统称为图结构，通过对图结构的分析进而将其分解为多个子结构，其中每个子结构对应一个候选构件。图 3 展示系统分解方法的分类体系，基于我们的研究经验示意性地给出系统分解方法的研究力度与效果。

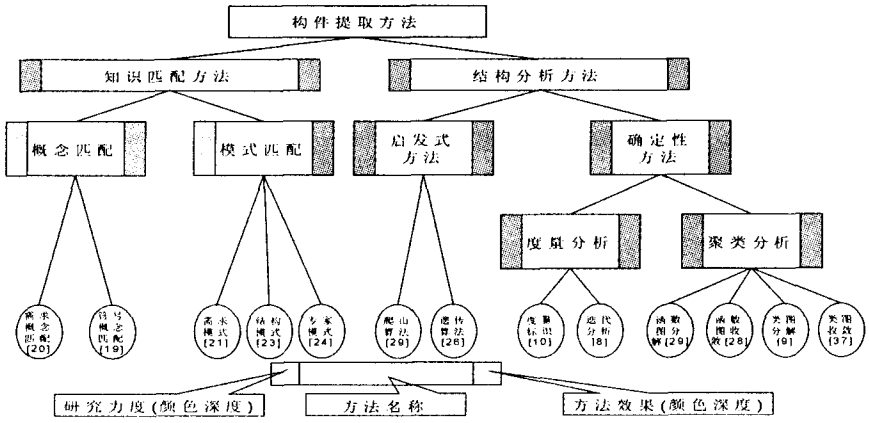

图 3　系统分解方法分类体系

3.2　基于知识匹配的系统分解方法

知识匹配作为系统分解的一类方法，它通过对软件系统中语义元素的分析而将程序中具备相同或者近似语义的实体关联起来作为一个候选构件。这类方法一般是基于信息获取、数据挖掘，关联分析，专家系统等相关技术。随着这些技术的发展，这类方法也逐步受到重视。根据其在系统分解中

所采用的策略不同，又可以分为：概念匹配方法[19,20]，模式匹配方法[21~23]。该类方法在形式化描述、知识获取与表示上存在难度。

• 概念匹配　采用已有概念与程序中信息进行匹配，通过这种匹配可以获取与特定概念相关的软件构成成分，并认为这些构成成分具备良好的独立性与单一的功能性。通过这种方式得到的软件构成成分可以作为候选构件。目前这类方法存在的一些问题是：(1)概念的可获取性问题；(2)必须假定待分析的程序具备良好的语义特性，也就是它的标志符号具有实际的含义。

目前，这方面研究工作的代表有文[19,20]，其中文[19]首先通过专家系统获取概念，然后标识与这些概念关联的软件成分。文[20]则是直接从需求信息中提取概念，然后采用信息获取(IR)技术在程序中匹配与特定需求对应的成分。

• 模式匹配　随着模式匹配以及构件特性研究的深入，出现了一类以模式语言为基础的系统分解方法，这类方法根据构件的结构特性，应用环境以及需求特性为构件模式提供描述，进而根据这种描述，从程序中识别构件。目前这种方法的代表有文[21~23]，其中文[21]定义了一种模式语言为构件的需求应用环境提供描述，并以这种描述作为系统分解的方法，文[23]为体系结构提供一种描述语言，然后基于该语言完成体系结构的恢复。除了定义模式语言的方法外，有的方法还采用了专家系统[24]为基于模式的系统分解提供支持。模式匹配方法存在的主要问题是模式的可获取性与准确性难以保证。

3.3　基于结构分析的系统分解方法

3.3.1　结构分析方法的分类　目前，结构分析方法是系统分解的主流方向，它以软件系统的某种抽象结构为基础，进而通过对该结构的划分或聚类而实施系统分解。这种抽象结构可以是类图、函数调用图、文件引用图、模块依赖图以及概念格等，我们将之统称为图结构。在目前的研究中，已经证明图结构的最优划分或聚类问题是 NP 难的，因而针对它也就不存在一个普适的确定性最优解决方案。基于目前结构分析方法的现状，我们将其分为：启发式分析与确定性分析。为了提高系统分解效率，启发式方法一般采用次优解算法，根据系统分解度量获取一个可以接受的解决方案，它是一种高效的普适性算法，但系统分解的可用性差。而确定性方法一般基于软件系统的某种结构特征，试图从中发现针对特定视图的最优解决方案。

3.3.2　启发式方法　是一种基于最优搜索的近似算法，它以符合给定质量特性的系统分解方案为目标。该类方法一般能快速地发现满足给定质量的方案，但随着质量指标的提高，其效率将急剧下降。同时，启发式方法的稳定性差，即同一软件系统采用同一方法在不同时期得到的结果不一致。目前在系统分解中广泛采用的启发式方法有：爬山算法[14,25]与遗传算法[14,26]。

• 爬山算法。其基本思想是基于当前状态搜索更优解。在系统分解中，它首先随机地选取一个初始划分，在该划分的基础上，通过某种迭代策略不断地获取更优解。文[29]中的系统分解方法采用的即是以邻接划分为迭代策略的爬山算法，邻接划分指的是移动原始划分中一个模块的某节点至另一个模块而得到的新划分。图 4 展示了邻接划分的实例。图 5 描述了爬山算法的基本过程。

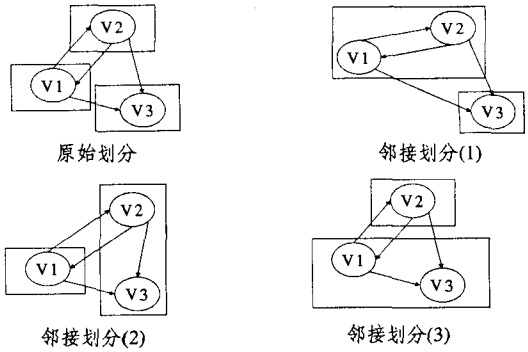

图 4　邻接划分示例

输入参数： 抽象图结构Graph

返回结果： 抽象图结构划分Partition

算法描述：

```
Partition Modularization(Graph g)
begin
    Partition P=getInitialPartitionByRandom(g)
    //P={M1,M2,M3,M4,...,Mn}, Mi是划分P中的一个模块;
    Partition NP=getBetterNeighbourPatition(P);
    //基于P找到一个质量更高的邻接划分NP
    while(NP!=null)  //条件：能够找到一个质量更高的邻接划分
    begin
        P=NP;
        NP=getBetterNeighbourPatition(P);
        //基于P找到一个质量更高的邻接划分NP
    end
    return NP;
end
```

图 5　爬山算法

为了提高爬山算法在系统分解中的质量特性，文[25]中采用了多爬山算法，它通过对某一软件系统同时进行多个爬山进程，而提高系统分解的效率与质量。

• 遗传算法。其基本思想是基于 Darwin 进化论和 Mendel 的遗传学说，它认为在进化过程中经过存优去劣的自然淘汰，适应性高的基因结构将得以保存下来。文[26]中将系统分解看成一个不断优化的进化过程，它基于模块依赖图(MDG)，以结点为单位进行基因编码，一种基因结构表示一个相应的划分实例，也称之为个体，通过对个体执行变异，交叉，选择的进化操作而获取高质量的系统分解方案。图 6 描述了遗传算法的系统分解过程。

输入参数： 抽象图结构Graph g

返回结果： 抽象图结构划分Partition

算法描述：

```
Partition GenericAlgorithm(Graph g)
begin
    Population population=CreateAPopulation()\
    //随机的生成一组个体，每个个体代表一个划分
    for(int num=0;num<maxIterationNum;num++)
    begin
        Population newPopu=CreateANewPopulation();
        //通过选择操作与生产形成一个新的群体
        population=crossOver(population,newPopu)
        //对原群体以及新产生的群体进行交叉操作，生成一个子代全体
        population=mutation(poplation);
        //通过变异操作形成一个新的子代全体
    end
    return findBestPartition(population);
    //在群体population中选取一个最好的个体作为候选划分
end
```

图 6　遗传算法

将遗传算法应用于系统分解存在两个需要关注的问题：适应度函数的确定与进化操作的选取。适应度函数用于度量个体的适应程度，在系统分解中，它将采用前述的某种系统分解度量，文[26]即以全局度量 MQ 为适应度函数。而进化操作作为控制遗传算法执行的关键元素存在三种基本操作：选择，交叉与变异，在系统分解中，选择操作以适应度函数为基准选取更优划分，交叉操作则是通过综合两个划分方案以发现更优解，而变异则通过局部的改变当前划分以期获得更优结果，前述的邻接划分策略即可以作为一种变异操作，文[29]也给出了一种基于邻接划分的遗传算法。

3.3.3 *确定性方法* 在时间维上对同一软件系统的多次系统分解结果是一致的。它也是目前系统分解所主要采用的方法，通常它以函数调用图[27~34]，类图[35~39]为基础。根据分析策略的不同，可将其分为度量分析与聚类分析。度量分析方法一般基于度量进行系统分解，它通过对抽象结构的度量分析或候选构件的度量验证达到系统分解的目的；聚类分析方法则是基于聚类策略进行系统分解，它通过将软件抽象结构中联系紧密的结点集合归为一类而达到系统分解的目的。这种分类方式基于分析策略的不同侧重点。实际上，度量分析方法也可能需要采用聚类的思想，而聚类分析可能也需要基于某种系统分解度量。

输入参数： 抽象图结构Graph

返回结果： 抽象子图结构集合ModuleSet

算法描述：

```
ModuleSet RBCLAlgorithm(Graph g)
begin
    SetInfluenceFactor();
    //为各种关系(继承，组装，实例连接，消息连接)设定影响因子
    SetWeightForEdge(g);
    //对两个顶点之间的每种关系，计算其权值
    ModuleSet moduleSet=new ModuleSet();
    while(hasMoreEdge(g))
    //条件：如果图中还有边
    begin
        ReduceEdgeWeight(g,wd);
        //将图中所有边的权值都减去一个参数值wd
        ArrayList moduls=getModulsByRemoveEdge(g)
        //删除权值小于0的边并获取规模介于给定阈值之间的模块
        moduleSet.add(moduls);
        //加入新的候选模块
    end
    return moduleSet;
end
```

图 7 RBCL 构件标志算法

·度量分析。度量分析系统分解方法基于度量识别候选构件。其中文[40]采用了函数调用图的度量分析从结构化软件中识别抽象数据类型与抽象数据对象；文[8,10]则采用类图的度量分析在面向对象软件中识别类树或类簇。基于度量的可复用构件获取[10]与基于有向带权图的迭代方法[8]是度量分析方法的典型实例，它们基于不同层次的度量模型进行系统分解。这类方法的提取效果较好，但相对而言也更为复杂。

基于度量的可复用构件获取[10]采用了类间关系分析构件标志算法—RBCL 从面向对象遗产系统中挖掘构件，图 7 给出了它的基本过程，它是一种以 RMCQ(构件可复用性度量模型)为验证基准的系统分解算法。

有向带权迭代分析方法[8]采用前述的模块独立性度量 IM 考察图中小规模的连通子图，它通过选取度量值高于阈值的连通子图作为候选构件，同时将这些候选模块表示的子图抽象为一个节点，而形成一个规模更小的新图，如此迭代分析直到得到新图规模小于给定阈值。其中独立性度量考察的是软件系统的类图结构，在该方法中表现为一种构件识别手段。图 8 给出了它的基本过程。

输入参数： 抽象带权图结构Graph G

返回结果： 模块集合ModuleSet

算法描述

```
ModuleSet RBCLAlgorithm(Graph G)
begin
    ModuleSet Candidates=Φ
    While (sizeOf(G)≥s)//条件：图的结点数大于给定阈值s
    begin
        subGraphList= getConnectedSubGraphs(G, k)
          //获取G中所有规模小于k的连通子图
        newCandidates=getIndependentSubGraphs(subGraphList, d)
          //计算每个子图的独立性，获取所有独立性高于d的子图。
        Candidates=Candidates∪newCandidates
          //加入获取的候选子图
        G=constructNewGraph(G, newCandidates)
          //收敛选定的子图为结点，重构新图
    end
    return Candidates;
end
```

图 8 有向带权迭代分解算法

·聚类分析。针对软件系统的抽象结构，构件可以看成是由紧密联系且具备相似特性的一组软件成分构成，这样系统分解就可以抽象为软件构成成分的聚类问题[9,14,15,25,26,28,35,38,39]。聚类研究在很多领域有着广泛的应用，其在系统分解中的应用可以大致分为两种情况，自顶向下的系统分解以及自底向上的结点收敛。自顶向下的系统分解一般基于完整的系统抽象结构，通过去除其中的一些元素而将系统分解为几个独立的模块。自底向上的分析一般通过将被依赖结点收敛到依赖结点而得到更大粒度的结构作为候选构件。基于聚类的系统分解研究一直以来受到了广泛的重视，本文将简要地概述函数调用图收敛方法[28,30]、函数调用图分解方法[29,31~33]、类图收敛方法[35,37]、类图分解方法[9,41,42]。

1)函数调用图收敛方法。文[28]中给出了一种针对函数调用图所表示的树结构收敛的算法，其基本步骤是：(1)选取一个活动结点；(2)选取一个收敛结点；(3)将收敛结点合并至活动结点，依此继续直到只剩下一个结点。这种方法可以针对函数调用图给出系统的层次结构，而每一层的构成元素都可以作为一组候选构件。文[30]给出另外的一种针对函数调用图的收敛方法，它通过获取抽象数据类型与抽象数据对象将结构化软件系统转化成面向对象软件系统。

2)函数调用图分解方法。针对结构化软件的函数调用图，怎样从其中获取抽象数据类型以及抽象数据对象在软件维护与演化领域受到了广泛的关注。对此，人们提出了一系列抽象数据类型与抽象数据对象的获取方法[29,31~33]，它们一般采用的是函数调用图的分解方法，通过对函数调用图的局部结构分析从中标识调用关系紧密的一组函数以及相关的变量作为可复用的软件成分。

3)类图收敛方法。基于类图的收敛方法通过将关系紧密的两个类收敛成一个结点，该方法针对类图给出了系统的层次结构，其中每层的元素都可以作为候选构件。文[37]针对类图采用了将权值小的边收缩而使两个结点合并成一个结点

的收敛策略，类似的其他方法还可见文[35]。实际上，有向带权迭代分析方法[8]也采用了类似的收敛策略，但是它的核心机制是独立性度量技术。

4)类图分解方法。类图分解方法通过将类图中某些弱边删除掉，而对一个系统进行划分，划分中不同粒度与层次的模块即是候选构件。图聚类方法[9]采用了最小图算法为图中的边确定权值，然后根据边的权值顺序不断地删除弱边而进行系统结构的划分。基于度量的可复用构件获取[10]也采用了类似的聚类策略，但该方法的核心机制是 RMCQ(构件可复用性度量模型)。

3.3.4 *结构分析方法的讨论* 目前，结构分析方法是系统分解研究的主流，通过实验发现，该类方法的系统分解效果通常优于知识匹配方法，但方法的通用性与结果的可用性仍无法满足应用的需要。结构分析方法一般基于假设：1)良构的软件系统由具备高内聚特性的模块松散地耦合在一起[8,9,14]；2)软件系统模块之间不存在交叉[17]。

众多实验研究表明，满足假设的软件系统，结构分析方法的系统分解效果良好。但实际中的软件系统往往不能很好地满足上述假设，通过对软件系统结构特性的分析，我们发现如下几种软件结构模式：模块型，分散型，公用型，聚集型是可能存在的，如图 9 所示。

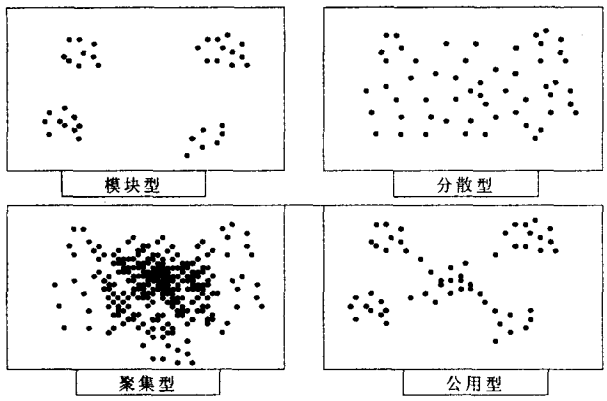

图 9 软件结构模式示例

在这几种软件结构模式中，模块型是满足假设的软件形态，它具备高内聚低耦合的系统特性且不存在交叉模块。分散型描述的是一种松散的软件结构，它不存在明显的高内聚模块，实际中很多类库结构就属于这种形态；聚集型以一个主要模块为中心，其他成分挂靠在它的周围，某些具备核心功能的软件属于这种形态；公用型结构中，存在一个或者多个的交叉成分，这种形态的软件在实际中是普遍存在的[17]。我们认为，正是由于多种可能软件结构形态的存在，所以目前的系统分解方法缺乏足够的通用性。

4 构件提取系统

随着构件提取研究的深入，人们希望通过工具的支持，它能成为一种高效的软件生产手段，对此，目前存在一系列的研究性与商业性工具。本文将概述其中几个典型的系统。

· **CodeMiner**。是 Dunn 和 Knight 等人开发的辅助程序员识别 C 语言遗产系统中可复用模块的工具[24]。它通过(1)标识多次调用的函数，(2)标识松散耦合的函数，(3)表示能够构成抽象数据类型的函数以及变量三种方式来识别软件系统中潜在的可复用成分。

· **CARE**(*Computer-Aided Reuse Engineering*)[11]。是一个针对 Ada 和 ANSI C 遗产软件系统的辅助可复用构件标识和验证的系统。它把整个分析过程分为以下两个阶段：第一阶段识别一些候选的构件，并为可能的复用部件进行包装；第二阶段由具有构件所在的应用领域的领域专家分析每个构件以决定构件所能提供的服务。最后，合格的可复用构件连同所有能得到的相关信息被存入构件库中。

· **PATRicia**(Program Analysis Tool for Reuse)。是一个针对面向对象系统基于度量的可复用构件自动标识系统[43,44]。它由自然语言分析工具(CHRiS)，C++ 程序分析工具，面向对象度量数据收集工具，构件质量和可复用性分析工具组成。

· **青鸟构件提取辅助工具**[45]。是青鸟程序分析工具集的组成部分，它以程序分析和信息抽取器为基础，能够根据青鸟软件可复用性度量模型，自动标识面向对象软件系统中的候选可复用构件，支持候选构件的手工筛选，支持可复用构件代码的包装并提供了构件夹和工作台管理，方便了构件的提取过程。

展望 构件提取尤其是基于结构分析的系统分解方法一直是程序理解与软件复用领域的一个重要交叉研究分支，它能够为软件复用，软件测试，逆向工程以及软件维护提供支持。目前的研究已经取得了相当的进展，针对其中一些方法实现了相应的系统，它能为特定的软件系统实施构件提取，在结构分析方法的讨论一节我们可以看出，构件提取还存在着广泛的研究空间。要使构件提取真正成为一种有效的软件生产手段还是一个巨大的挑战。

随着网络软件的发展，构件提取将会面临新的挑战，同时知识匹配系统分解方法的深入研究也将进一步地推动系统分解方法的发展，因此我们认为未来的构件提取研究存在如下的几个关注点：

· 研究软件系统的结构特性，发现其中的一些新的特征，以期能突破目前过于严格的假设，使得系统分解方法能够比较通用地应用于各种软件结构形态。

· 研究网络形态的软件构件提取，随着网络技术的发展，软件形态发生了巨大的变化，未来的构件提取以及程序理解应该能够适应这种新情况的变化，不仅能够从网络软件中获取构件，而且能够将其封装成网络服务形式。

· 知识匹配方法与结构分析方法的有效结合，进而提高构件提取的效率与质量，提高候选构件的可用度。我们认为以结构分析方法抽取构件实现体，知识匹配方法获取构件描述将是构件提取的一条有效途径。

· 设计模式的发现与体系结构的恢复可以认为是一种高层次的构件获取，随着软件系统规模的增长，这也将必然成为一个备受关注的研究话题。

参 考 文 献

1 杨芙清，梅宏，李克勤. 软件复用与软件构件技术. 电子学报，1999，27(2)：68～75
2 杨芙清. 软件复用及相关技术. 计算机科学，1999，26(5)：1～4
3 Belady L A, Evangelisti C J. System Partitioning and its Measure. Journal of Systems and Software, 1982, 2(1): 23～29
4 Koschke R. Atomic architectural component recovery for program understanding and evolution. In: Proc. of 20th Intl. Conf. on Software Maintenance, 2002. 478～481
5 袁望洪. 面向对象程序理解系统 JBPAS:[硕士论文]. 北京大学计算机科学技术系，1999

6 周欣,孙家骕,杨芙清. 青鸟 C++ 程序理解工具. 计算机工程,2000,26(11):80～81

7 Patel S, Chu W, Baxter R. A Measure For Composite Module Cohesion. In:Proc. of 14th International Conference on Software Engineering, 1992

8 罗景,赵伟,秦涛,等. 基于有向带权图迭代的面向对象系统分解方法. 软件学报,2004,15(9):1292～1300

9 Chiricota Y, Jourdan F, Melancon G. Software components capture using graph clustering. In: Proc. of 11th IEEE Intl. Workshop on Program Comprehension, 2003

10 周欣,陈向葵,孙家骕,杨芙清. 面向对象系统中基于度量的可复用构件获取机制. 电子学报,2003, 31(5):649～653

11 Abd-El-Hafiz S K, Basili V R, Caldiera G. Towards Automated Support for Extraction of Reusable Components. In: Proc. of 9th Intl. Conf. on Software Maintenance,1991

12 Even-Andre Karlsson C. Software Reuse: A Holistic Approach. New York: Wiley & Sons, Ltd., 1995

13 Bansiya J. A Hierarchical Model For Quality Assessment Of Object-Oriented Designs: [Ph. D. Dissertation]. Huntsville: University of Alabama in Huntsville, 1997

14 Mancoridis S, Mitchell B S, Rorres C, Chen Y, Gansner E R. Using automatic clustering to produce high-level system organizations of source code. In:Proc. of 6th Intl. Workshop on Program Comprehension, 1998

15 Anquetil N, Fourrier C, Lethbridge T. Experiments with hierarchical clustering algorithms as software remodularization methods. In: Proc. of 6th Working Conf. on Reverse Engineering. 1999

16 Tzerpos V, Holt R C. Mojo: A distance metric for software clustering. In: Proc. of 6th Working Conf. on Reverse Engineering, 1999

17 Mitchell B S, Mancoridis S. Comparing the decompositions produced by software clustering algorithms using similarity measurements. In: Proc. of 19th IEEE Intl. Conf. on Software Maintenance, 2001

18 Koschke R, Eisenbarth T. A framework for experimental evaluation of clustering techniques. In: Proc. of 8th Intl. Workshop on Program Comprehension, 2000

19 Biggerstaff T J, Mitbander B G, Webster D. The concept assignment problem in program understanding. In: Proc. of 15th Intl. Conf. on Software Engineering, 1993

20 Zhao W, Zhang L, Liu Y, Luo J, Sun J S. Understanding How the Requirements Are Implemented in Source Code. In:Proc. of 10th Asia-Pacific Software Engineering Conf. 2003

21 Spinellis D, Raptis K. Component mining: a process and its pattern language. Information and Software Technology, 2000, 42 (9):609～617

22 Ferreira da Silva M, Lima Werner C M. Packaging reusable components using patterns and hypermedia. In: Proc. of 4th Intl. Conf. on Software Reuse, 1996

23 Pinzger M, Gall H. Pattern-supported architecture recovery. In: Proc, of 10th Intl. Workshop on Program Comprehension, 2002

24 Dunn M F, Knight J C. Automating the detection of reusable parts in existing software. In: Proc. of 15th Intl. Conf. on Software engneering, 1993

25 Mahdavi K, Harman M, Hierons R M. A multiple hill climbing approach to software module clustering. In: Proc. of 19th Intl. Conf. on Software Maintenance, 2003. 315～324

26 Doval D, Mancoridis S, Mitchell B S. Automatic clustering of software systems using a genetic algorithm. In: Proc. of 9th Intl. Workshop Software Technology and Engineering Practice. 1999. 73～81

27 Korel B, Rilling J. Program slicing in understanding of large programs. In: Proc. of 6th Intl. Workshop on Program Comprehension, 1998. 145～152

28 Rayside D, Reuss S, Hedges E, Kontogiannis K. The effect of call graph construction algorithms for object-oriented programs on automatic clustering. In: proc. of 8th Intl. Workshop on Program Comprehension, 2000. 191～200

29 Canfora G, Czeranski J, Koschke R. Revisiting the Delta IC approach to component recovery. In: Proc. of 7th Working Conf. on Reverse Engineering, 2000. 140～149

30 Valasareddi R R, Carver D L. A graph-based object identification process for procedural programs. In: Proc. of 5th Working Conf. on Reverse Engineering, 1998. 50～58

31 Canfora G, Cimitile A, Tortorella M, Munro M. A precise method for identifying reusable abstract data types in code. In: Proc. of 10th Intl. Conf. on Software Maintenance, 1994. 404～413

32 Koschke R. An incremental semi-automatic method for component recovery. In: Proc. of 6th Working Conf. on Reverse Engineering, 1999. 256～267

33 Sahraoui H A, Melo W, Lounis H, Dumont F. Applying concept formation methods to object identification in procedural code. In: Proc. of 12th IEEE Intl. Conf. on Automated Software Engineering, 1997. 210～218

34 Yeh A S, Harris D R, Reubenstein H B. Recovering abstract data types and object instances from a conventional procedural language. In: Proc. of 2nd Working Conf. on Reverse Engineering, July 1995. 227～236

35 Mancoridis S, Holt R C. Recovering the structure of software systems using tube graph interconnection clustering. In: Proc. of Intl. Conf. on Software Maintenance, Nov. 1996. 23～32

36 De Lucia A, Fasolino A R, Munro M. Understanding function behaviors through program slicing. In: Proc. of 4th Workshop on Program Comprehension, March 1996. 9～18

37 Muller H A, Uhl J S. Composing subsystem structures using (k, 2)-partite graphs. In: proc. of 6th Conf. on Software Maintenance, Nov. 1990. 12～19

38 Wiggerts T A. Using clustering algorithms in legacy systems remodularization. In: Proc. of 4th Working Conf. on Reverse Engineering, Oct. 1997. 33～43

39 Saeed M, Maqbool O, Babri H A, Hassan S Z, Sarwar S M. Software Clustering Techniques and the Use of Combined Algorithm Software clustering techniques and the use of combined algorithm. In: Proc. of 7th European Conf. on Software Maintenance and Reengineering, March 2003. 301～306

40 Girard J F, Koschke R, Schied G. A metric-based approach to detect abstract data types and state encapsulations. In:Proc. of 12th IEEE Intl. Conf. Automated Software Engineering, Nov. 1997

41 Chen J L, Wang F J, Chen Y L. An object-oriented dependency graph for program slicing. In: Proc. TOOLS 24 of Technology of Object-Oriented Languages, Sept. 1997. 121～130

42 Mancoridis S, Mitchell B S, Chen Y, Gansner E R. Bunch: a clustering tool for the recovery and maintenance of software system structures. In: Proc. of 13th IEEE Intl. Conf. on Software Maintenance,1999. 50～59

43 Etzkorn L H, Davis C G. Automatically Identifying Reusable Components in Object-Oriented Legacy Code. IEEE Computer, 1997,30(10): 66～71

44 Etzkorn L H. A Metrics-Based Approach to The Automated Identification of Object-Oriented Reusable Software Components, The University of Alabama in Huntsville

45 陈向葵. 面向对象系统中可复用构件的提取及工具支持:[硕士论文]. 北京大学计算机科学技术系, 2000