基于过程蓝图的程序环路复杂性度量方法*)

刘建宾[1,3] 李建忠[2,3] 余楚迎[4] 杨林邦[3]

(北京信息科技大学计算机科学与工程系 北京 100101)[1]

(韩山师范学院数学与信息技术学院 潮州 521041)[2]

(汕头大学工学院计算机系 汕头 515063)[3] (汕头大学物理系 汕头 515063)[4]

摘 要 提出一种基于过程蓝图的程序环路复杂性度量实现方法。将传统基于程序控制流图的度量信息抽取变为对过程蓝图的实现层表示-抽象实现结构图的信息抽取,避免程序源代码的语法分析和控制流图的构造,简化度量过程和实现,并提高度量处理的效率。

关键词 环路复杂性,McCabe 度量,过程蓝图,抽象实现结构图,控制流图(CFG)

A Software Cyclomatic Complexity Metrics Method Based on Procedure Blueprint

LIU Jian-Bin[1,3] LI Jian-Zhong[2,3] YU Chu-Ying[4] YANG Lin-Bang[3]

(Dept. of Computer Science and Engineering, Beijing Information Science and Techology University, Beijing100101)[1]

(Institute of Mathematics and Infrmation Technology, Hanshan Normal College, Chaozhou 521041)[2]

(Department of Computer Science, Engineering Institute, Shantou University, Shantou 515063)[3]

(Department of Physics, Shantou University, Shantou Guangdong 515063)[4]

Abstract In this paper, a method is proposed to measure the software cyclomatic complexity based on the procedure blueprint, in which the program information required by the metrics is abstracted from the abstract implement structure diagram, the implementation representation of the procedure blueprint, replacing the traditional information abstraction based on the control flow graph. This technique can avoid the process of analyzing the syntax of the program and constructing the control flow graph so that the process of the measuring is simplified and it is easier to be implemented and more efficient than the traditional technique.

Keywords Cyclomatic complexity, McCabe metrics, Procedure blueprint, Abstract implement struture diagrams(AISD), Control flow graph(CFG)

1 引言

软件复杂性问题是影响软件质量的重要因素。大量的学者从不同角度提出各种度量各种软件复杂度的方法,目的是通过复杂度的计算来评价软件的设计合理性、结构化程度和模块复杂程度等,从而解决设计问题、重构定位问题和细化设计模块,使代码更易于维护和理解,以提高软件的稳定性、健壮性和可维护性,在整体上提高软件产品的质量。

软件度量是一种将软件工程中各种影响产品最终质量的因素转换成量化表示形式,以利于客观评价的科学方法[1]。在软件工程中提出了下列度量模型:需求分析度量模型、设计策划度量模型、源程序度量模型、测试度量模型及维护度量模型。

上述各种模型中,以设计阶段度量模型和源程序度量模型评价的指标最客观,并在这方面取得不少的成果[1]。以结构化方法为例,有著名的 McCabe 方法(环路复杂性度量)、Halstead 方法、Henry-Kafura 方法等[2]。

2 McCabe 度量方法

McCabe 度量方法又称环路复杂性度量,是一种定量度量程序复杂度的有效方法。该方法在程序的理解、软件测试和程序维护等方面得到广泛的应用,如重复代码检测[3]等。

环路复杂性度量是一种同程序图的拓扑性质有关的、关于程序结构复杂性的度量[2]。它可依据程序的控制流图(CFG)[4]来对程序的结构复杂性进行度量。结构复杂性可按下式计算:

$$V(G)=E-n+2 \tag{1}$$

其中,E 为控制流图的边数,n 为控制流图 G 的节点数。

采用式(1)表示环路复杂性,不仅可以反映控制流图中线性无关的路径数目,还避免了使用"虚边"加 1 的问题[1]。

McCabe 度量方法为软件系统的可靠性和可维护性提供了一个有效的定量度量,但这种方法的缺点也是明显的:如复杂性不能严格区分,对于分支结构,循环结构效率一样,1000 行的顺序程序与 1 行语句的复杂性相同等。因此,实际应用中可与其它度量法综合使用。

2.1 控制流图的构造

控制流图是环路复杂性度量的基础,提供计算 McCabe 度量的所必需的所有信息。它是一种描述程序控制逻辑的有向图,图中的每个节点对应于一个顺序流程的程序代码块或谓词,有向边对应于程序中的转移,并且每个节点都可从入口点到达,从每个节点都可以到达出口节点[2]。

*)广东省自然科学基金项目(项目编号:032027),北京信息科技大学科学研究基金项目。**刘建宾** 博士、教授、硕士生导师,主要从事软件方法与工具、软件工程、管理信息系统的研究;**李建忠** 硕士,主要研究领域为软件方法、CASE 工具、面向对象技术等。

一个程序由许多不同类型的语句组成，这些语句可分为不影响控制流和影响控制流两类。前者通常称为表达式语句，如简单赋值语句等；后者为流程控制语句，包含循环语句、选择语句和转向语句。在符合结构化程序设计思想这一前提下，一个程序的控制流图可以由上述语句的控制流图结构组合而成。

在 McCabe 的度量中，其模型是一个控制流图，但无论是该模型还是从程序到该模型的映射关系都没有被正式定义[2]。文[3]给出了一种构造控制流图的方法。通过该方法，可以构造出各种语句(以 Java 为例)的标准结构及计算出相应的环路复杂性的值。

根据文[3]的构造方法，可容易计算出单项选择(可选)语句 if、循环语句 for；while 和 do…while 的环路复杂性均为 2；if…else if…else、switch 是多分支选择语句，具有不确定条通路，由构造方法得其环路复杂性是它们的条件分支数 n。

上述各种循环语句、选择语句的控制流图标准结构的构造及环路复杂性计算，其前提是它们的条件表达式均为简单表达式。然而，这些条件表达式有可能是通过逻辑运算符“&&”和“‖”连接起来的复合表达式(switch 语句除外)，它们会影响控制流图的构造。在没有使用 GOTO 语句的结构化程序中，对环路复杂性有影响的是逻辑运算符的数目，与它们的类型无关[2,3]。所以，对于条件表达式为复合语句的循环语句、选择语句，其环路复杂性的值是：该循环语句、选择语句的标准结构的环路复杂性的值加上逻辑运算符的数目。

2.2 控制流图的环路复杂性计算

由 2.1 节，我们已得到了所有控制语句的环路复杂性的计算方法。因此，只需知道控制流图中会影响环路复杂性的构造就可计算出一段程序的 McCabe 的值。其步骤如下：

1)遍历控制流图，找出所有会影响控制流的语句(谓词)，并记下它们出现次数之和 m；

2)McCabe 的值等于各个语句相应的环路复杂性的值之和 $-(m-1)$，即

$$V(G)=\sum_{i=1}^{m}V(G_i)-(m-1) \qquad (2)$$

这里，$V(G)$是控制流图 G 的环路复杂性；$V(G_i)$是控制流图 G 中影响控制流的语句的环路复杂性；m 是控制流图 G 中所有影响控制流的语句的出现次数之和；减去 $m-1$ 是因为所有标准结构都指向出口节点，但一个流图有且只有一个出口节点，因此 m 条语句的构造就多出 $m-1$ 条指向出口节点的边，故必须减 $m-1$。

3 基于过程蓝图的环路复杂性度量

计算不含 GOTO 语句的结构化程序的环路复杂性，需要知道代码中会影响控制流的循环语句、选择语句的数量以及逻辑运算符的数目。如果直接在源代码上计算这些语句、逻辑运算符的数量，必须进行语法分析，这是一个十分复杂繁琐的过程。

复杂性表达式，不仅可以由程序推导出来，而且可以直接从程序的各种结构图中得到，如流程图、树图结构等[2]。程序环路复杂性度量的传统方法和过程是首先扫描源程序，进行语法分析，然后将源程序变换为控制流图，之后才能进行度量计算。这种方法的度量处理过程环节多、时间长、开销大，度量系统的设计也比较复杂和困难，无论在技术上还是在实现上代价都比较高。因此，本文提出使用过程蓝图[6]来简化这一度量过程。

3.1 过程蓝图

过程蓝图是具有概念、逻辑和实现 3 层外部视图和统一内部结构的程序可视化过程建模语言，通过概念层到逻辑层的控制流映射和逻辑层到实现层的数据流映射提供一定程度的程序独立性(控制流独立性和数据流独立性)[7]。过程蓝图具有具体概念结构图[8]、抽象实现结构图[8]和具体实现结构图[8]3 个抽象层次不同的外部表示形式，是一种图形化程序过程规格说明方法，是对过程源代码进行描述的模型。文[9]给出了 JAVA 过程蓝图的处理节点类型与 JAVA 编程语言的对应关系。

3.2 基于抽象实现结构图的环路复杂性度量算法

抽象实现结构图是过程蓝图的实现层外部表示形式，它是由实现节点构成的、从左到右自上而下的线型树图，其本质是一棵抽象语法树，显示包含产生程序代码的所有语义和表示信息[8]。它将顺序、循环、选择等语句抽象表示为物理实现节点，并做可视化处理。节点类型和语义说明构成了实现节点的描述与表示。实现节点可以分为带数据流的节点和不带数据流的节点。不带数据流的节点语义使用自然语言描述，带数据流节点语义使用符合目标语言句法的操作语句和表达式进行精确表示，显示提供源代码所包含的所有信息。因此，通过直接分析抽象实现结构图的节点类型和带数据流节点的表达式，就能获得计算环路复杂性度量所需的完整信息，即各类控制流语句的数量和逻辑运算符的数目。从文[9]可知 JAVA 过程蓝图的抽象实现结构图的处理节点类型与 JAVA 语言的选择、循环等控制语句存在一一对应关系。因此，容易从 2.1 节讨论的控制语句的控制流图标准结构得到这些处理节点的相应的环路复杂性的值，如表 1 所示。

表 1 抽象实现结构图处理节点类型的 McCabe 度量值

抽象逻辑结构图的处理节点	选择和循环语句	McCabe 度量值
IFT	if()…语句	2+k
IFE	if()…else|elseif…语句	2+k
SWH	Switch 多分支选择语句	分支数 n
WHL	while()…循环语句	2+k
FOR	for()…循环语句	2+k
DOW	do…while()循环语句	2+k

注：k 为条件表达式中逻辑运算符“&&”及“‖”的数目。

在过程蓝图语境下，式(2)的意义解释如下：$V(G)$表示抽象实现结构图所描述的程序的环路复杂性；$V(G_i)$为抽象实现结构图中会影响控制流的处理节点的环路复杂性；m 是抽象实现结构图中所有影响控制流的处理节点的出现次数之和。

通过遍历抽象实现结构图，从中直接提取 McCabe 度量所需要的信息，如会影响控制流的处理节点、逻辑运算符的数目等，然后根据表 1 的度量值和过程蓝图语境下的式(2)，就可以计算出程序的环路复杂性的值。直接在过程蓝图的抽象实现结构图上对程序进行分析、度量，可以避免对源程序进行语法分析以及将源程序变换为控制流图这一复杂的传统方法和过程，从而在整体上提高度量的效率，减少开销。

参考文[10]的树前序遍历算法，得到如下算法的思路：1)通过递归函数遍历抽象实现结构图；2)提取当前实现节点的信息，通过它的节点类型循环控制节点，若为真，则；3)读取该节点的内容，得到它的条件表达式(SWH 除外)，然后通过词法分析函数计算出条件表达式中逻辑运算符“&&”“‖”的数目；4)通过上面得到的信息，计算出当前节点的环路复杂性，并累加；

用计算器记下影响度量的控制节点出现的总次数；5）最后，使用式(2)计算出程序的环路复杂性。图 1 和图 2 是用抽象概念结构图描述的具体算法。

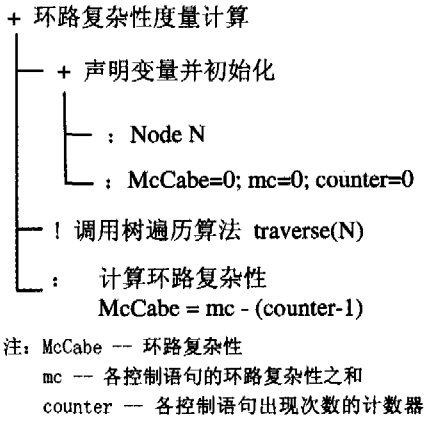

图 1　用抽象逻辑结构图描述的环路复杂性计算算法

4　实例研究

图的遍历算法是求解图的连通性问题、拓扑排序和求关键路径等算法。通常，有两种遍历图的路径：深度优先搜索和广度优先搜索。广度优先搜索与树的按层次遍历的过程类似，实质是通过边或弧找邻接点的过程。图 3 是用 JAVA 语言实现的广度优先搜索算法的源代码[11]。

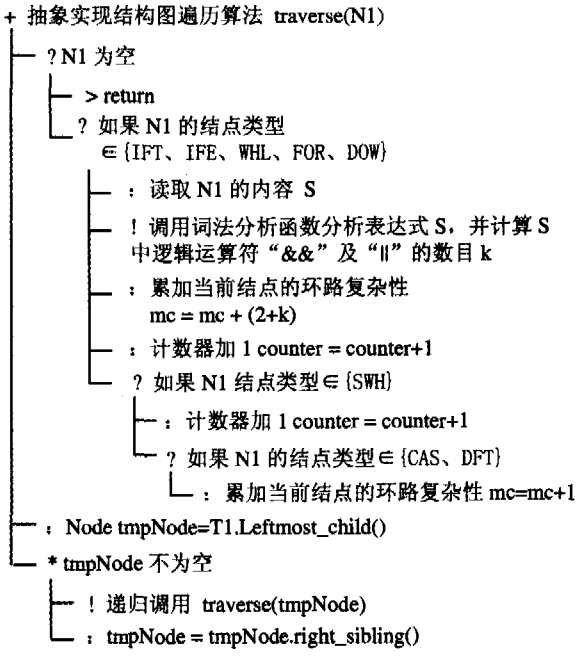

图 2　用抽象概念结构图描述的遍历算法

```
public atatic void BFS(int Vertex){
    int Pointer;//节点声明
    QueueArray Data=new QueueArray();
    int i;
    Data. AddQueue(Vertex);//存入队列中
    Visited[Vertex]=1;//已搜索
    System. out. print("|"+Vertex+"==>");
    while(Data. Front! =Data. Reat){//队列为空时,结束循环
    Vertex=Data. DelQueue();//从队列中取出数据
      for (i=1;i<VertexNum;i++){//遍历顶点
        //如果顶点未被访问
        if(Grqph[Vertex][i]==1&&Visited[i]=0){
          Data. AddQueue(i);//存入队列中
          Visited[i]=1;//已搜索过的顶点
          System. out. print("{"+i+";==>");
        }//end if
      }//end for
    }//end while
}
```

图 3　广度优先搜索算法的源代码

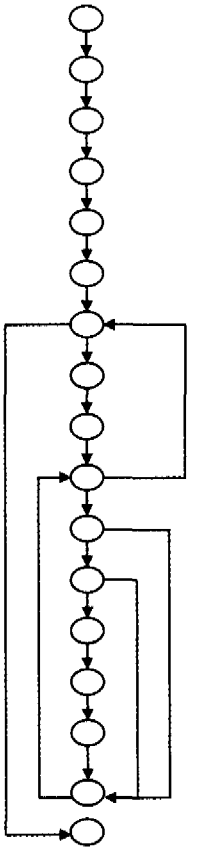

图 4　广度优先搜索算法的控制流图

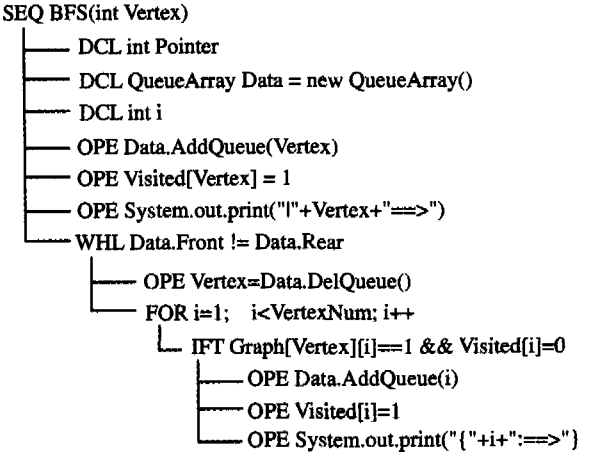

图 5　用抽象实现结构图描述的广度优先搜索算法

根据度量程序环路复杂性的传统方法与过程，需要先将广度优先搜索算法的源代码变换为控制流图，如图 4 所示。由控制流图可以得到边数 $e=20$，节点数 $n=17$，将其代入度量式(1)，可以计算出该算法的环路复杂性的值：$V(G)=20-17+2=5$。

抽象实现结构图一种图形化程序表示方法，是对源程序进行描述的模型。图 5 为广度优先搜索算法的抽象实现结构图表示，它与图 3 所示的算法源代码等价，可以从图 5 所示的抽象实现结构图出发自动生成与图 3 完全一致的代码。使用 3.2

（下转第 283 页）

例构成了复合关系，包括了动态建立的子封装，标记为{plug-ins}。因此，在给定类层次的封装只能有一个协作图。然而，DRSCDE中的AO构件可以出现在不同的框架层结构图中，也可以由若干个子AO构件组成。所以，UML-RT的协作图并没有提供像框架层这样的灵活性。

UML-RT的封装对象的协作图表示封装的分解，每个子封装是一个类而不是一个实例，所以分解的协作图表示子封装实例的一个样式(pattern)。语义上有点微弱区别的是DRSCDE的框架层通过复合关系支持AO构件的层次分解，但不是样式，表示的是构件实例。

在UML-RT中，某些标记为{plug-ins}的子类表示在封装的协作中在运行时动态引入的，和该子封装的连接协议相当于DRSCDE中的状态原语中的创建原语，但是UML-RT中的封装的动态调度关系没能显示表示，因而与DRSCDE中的状态域及状态原语在语义上不能精确匹配。

封装中带有和状态机相连的终止端口，在模型中显示的表示其动态行为，在DRSCDE中没有相应语义的建模元素。在DRSCDE中AO构件的动态行为由详细层表示，详细层定义了基于实时多任务操作系统的过程设计的抽象对象。如任务、邮箱、信号量等。由于篇幅所限，就不在此详细讨论从状态机到详细层的映射。

在UML-RT中，封装之间的通讯关系表示广泛意义的信号的接收和发送，而DRSCDE中的AO构件之间的通信关系显示表示C/S关系，语义相对较窄，因而我们在两个模型之间的通讯关系映射时作了限制，将UML-RT的通讯关系约束为DRSCDE中的C/S关系。

在UML-RT中，跨接在封装边界的端口既没有和外部其它封装的连接也没有和内部其它子封装的连接，则无法确定此端口是终止端口或是中继端口，其语义是模糊的具有不确定性，因而在语义上无法匹配DRSCDE中的服务域或状态域或是客户域，此为限制之一。另外UML-RT的端口的多重性(二元性或多元性)指示端口实例的个数，DRSCDE也不支持。

结束语 本文提出了从UML-RT的结构模型映射到DRSCDE的体系结构模型的方法、约束和限制，以解决两个模型之间高层设计的映射，通过一个实例说明了映射的具体方法。将要做的研究工作是探讨从UML-RT的行为模型到DRSCDE的AO构件的可执行模型的映射方法以及AO构件在CORBA环境中的发布方法。

参 考 文 献

1 Shaw M, Garlan D. Software Architecture. 北京：清华大学出版社，1998
2 Lyons A. UML for real-time overview. April 1998 http://www.uml.org.cn/UMLApplication/pdf/umlrt_overview.pdf
3 Selic B, Rumbaugh J. Using UML for modeling complex real-time systems. March 11, 1998, http://www-128.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf
4 刘晓燕，张云生，等. 复杂实时系统软构件对象设计. 计算机工程与应用，2003,39(31)：119～121
5 刘晓燕，张云生，等. 基于构件的分布式实时系统架构图形化建模. 计算机应用研究，已录用
6 Schwarz J J, Maranzana M, Skubich J J, Martinez Y. Applicative Objects Interconnection in a Graphical Design for Real-Time Applications [C]. 20th IFAC Workshop on Real Time Programming (WRTP'95), 6-10 November 1995, Florida, USA
7 Booch G, Rumbaugh J, Jacobson I. UML参考手册. 姚淑珍，等译. 机械工业出版社，2001
8 Rational software Corporation. Rational Rose Real-time 2002. http://www.rational.com/product/rosert, 2002

(上接第269页)

节描述的算法遍历抽象实现结构图，可以得到影响程序环路复杂性的节点：WHL、FOR和IFT，它们的出现次数各为1；并且IFT的条件表达式为复合语句“Graph[Vertex][i]==1 && Visited[i]=0”，逻辑运算符“&&”的数目为1。所以 $m=1+1+1=3$。根据式(2)及表1，可计算出该算法的环路复杂性的值：$V(G)=V(G_{WHL})+V(G_{FOR})+V(G_{IFT})=2+2+(2+1)-(3-1)=5$。

通过两种不同的方法，计算得到的该广度优先搜索算法的环路复杂性的值都为5，结果一致。

结束语 对代码直接进行环路复杂性度量，需要为源程序构造控制流图。这要求先对源程序进行语法分析处理，然后再根据所提取的信息来生成控制流图。假设采用广为使用的自下而上的分析法-算符优先分析法[12]对需要进行度量的程序进行语法分析，则由文[12]可知，该方法包括算符优先文法及优先表的构造、算符优先分析算法、分析中的出错处理等，其时间复杂度至少为$O(n^2)$；控制流图实质上是一个有向图，假定采用十字链表[13]作为其存储结构，则构造控制流图算法的时间复杂度为$O(n+e)$。而3.2节讨论的基于过程蓝图的环路复杂性计算算法的时间复杂度为$O(n)$；并由文[12]有词法分析函数时间复杂度为$O(n)$。度量过程的复杂性分析和实例研究均表明在过程蓝图上对代码进行程序复杂性度量，其效率较传统方法要高。

环路复杂性度量是基于语法树的检测重复代码的Kontogiannis度量方法的主要度量技术之一[3]。因为过程蓝图本质是一棵抽象语法树，所以可以直接在过程蓝图上进行重复代码的检测，并避免源代码到抽象语法树变换这一繁复的过程。同时，应用本文所述的算法，可简化程序环路复杂性度量过程，从而提高该重复代码检测的效率，为重构自动定位提供更好的技术。

参 考 文 献

1 丁炎炎，等. 程序复杂度系统pgrmetrics的设计与实现. 计算机应用研究，2004,21(1)：167～169
2 王振宇. 程序复杂性度量. 北京：科学出版社，1997
3 van Rysselberghe F. Vingerafdrukken van code om duplicatie op te sporen：[Master Thesis]. Universiteit Antwerpen, 2001-2002
4 单永明. 一种源程序到控制流图的自动生成方法. 小型微型计算机系统，1996,17(10)
5 杜子德. 程序控制流图：一种可视化的程序设计工具. 计算机研究与发展，1995,32(12)
6 龚世生，刘建宾. 软件工程. 广州：广东高等教育出版社，1999
7 刘建宾，郝克刚. 抽象概念结构图到JAVA过程蓝图的平滑过渡及一致性. 计算机科学，2001,28(8)
8 刘建宾. 过程蓝图设计方法学. 北京：科学出版社，2005
9 刘建宾. JAVA过程蓝图. 计算机科学，2000,27(7)
10 Shaffer C A. 数据结构与算法分析：Java版. 张铭，刘晓丹，译. 北京：电子工业出版社，2001
11 黄国瑜，叶乃子. 数据结构：Java语言版. 北京：清华大学出版社，2002
12 陈火汪，刘春林，等. 程序设计语言编译原理：第3版. 北京：国防工业出版社，2000
13 严蔚敏，吴伟民. 数据结构：C语言版. 北京：清华大学出版社，1997