软件工程 程序分析 专家模型 方法学

计算机科学1996 Vol. 23 №. 6

87-90

程序分析的专家模型与方法学

刘宗田 姜 川
（合肥工业大学微机所） （合肥经济技术学院）

TP311.5

摘 要 In the paper, the research work on the area of program analysis is analyzed and compared firstly, then expert's model of program analysis is discussed from different sides, and some viewpoints about methodology of program analysis are raised according to analyzing the intrinsic nature of programs.

关键词 Program analysis, Expert's model, Methodology, Program concept.

在源代码级上分析程序是软件逆向工程的最基本的任务，其目的是从程序文本中了解程序的高层概念，这主要用于对软件维护的支持，软件可重用知识和部件的获取，以及探测已存在的软件产品等。

软件维护在软件生存周期中是最昂贵的阶段，许多人声称，习惯的软件维护活动占整个生存周期的50-90%。软件是逻辑产品，维修人员理解软件，要占用47-60%的维护工作量。没有自动支持，维护时间大部分被花在试图去理解被维护的对象上。理解的工作极大地依赖于对程序源代码的理解，这不仅对文档很差的软件如此，对于文档较好的软件也是如此，这是因为正向工程产生的文档不足以支持软件的维护活动，软件的最准确的文档就是源代码文本，因此，有的专家认为，下一代的软件工程环境应当是正反向都可用的。

目前，软件重用受到越来越广泛的重视，甚至被看作软件工业发达程度的标志。软件重用虽然在发达国家的很多公司中被采用，但无论在理论上还是在技术上，都有许多难题需要研究解决，其中可重用知识和可重用部件的获取被认为是技术难点之一。程序是提供可重用知识和部件的丰富资源。程序分析与理解是从程序中获得这些可重用知识与部件的主要途径。

程序分析还被用于探测竞争对手或军事对手的程序，检查竞争对手的程序中是否含有从我方程序中剽窃的部分，检查我方从军事对手引进的程序中是否埋藏着对我方有威胁的成分。

一、有关工作分析

程序分析方法及工具的研究至少可以追溯到70年代，这些研究可以分为四类。

第一类研究开始于70年代，而且一直持续到现在。其特点是追求程序隐含信息的显式表示和程序内部关系的可视化，例如控制流和数据流的分析、程序的各类视图的构造、各种帮助程序员观察程序的方法等等。这一类工具很多已经上市。其中 Objective-C Browser 用窗口方法显示 C 或 Object C 代码的层次、功能和继承的信息。Vifor 能将 Fortran 程序变换为有向图，并提供了一个具有选择、移动和放大缩小功能的图形编辑界面。Seela 将代码转换为程序设计的语言(program-design language)，并允许程序员编辑这种结构图。Grasp/Ada 能从 Ada 程序设计的语言或源代码产生控制结构图。Act 和 Battle Map 能显示程序的所有的控制路径和复杂性。EDSA 能让程序员跳过不重要的细节而沿控制流和数据流阅读 Ada 程序。Turing 工具采用源代码省略技术，利用省略规则使程序员跳过不重要的细节而只观察重要的部分。上述工具的售价一般在3000美元以上，有的高达29000美元。在国内，南京大学研制的 TAUS 能提供对 PASCAL 程序的多重观察。复旦大学和上海计算机软件实验室共同开发的 GAUSS 能对 C、COBOL 等语言程序进行结构分析，产生图表示。华侨大学的 CPAS-I 能将 C 语言程序变换为 PAD 图。第一类工具虽然对软件维护起到一定的辅助作用，但是，存在着三个严重的缺点：①所产生的文档或者太一般或者太详细。所产生的图描述存在迷航问题；②所产生的文档中缺乏维护所需要信息。③缺乏柔性。

第二类的研究开始于70年代后期，大量的研究集中在80年代和90年代。这一类程序分析技术与工具的特点是从源代码中提取信息，存放在通用数据库中，通过查询语言向数据库提问。有些专家把这一

类工具称为第二代程序分析工具。FAST 是这一类中最早的一个，是 FORTRAN 程序分析系统，它使用商用数据库管理系统 System2000存放程序结构的信息，并提供查询。OMEGA[1]是一个定义、恢复和修改类 PASCAL 程序的一些观察(配置、版本、调用图和切片)的系统，它用58条关系存放程序模型在关系数据库 INGRES 中。CIA[2]和 CIA^{++}是 C 和 C^{++}语言信息抽象系统，它们按概念模型将 C 源文件翻译成对象与关系的集合，关系信息存放在增式数据库中。OMEGA 和 CIA 被看作软件逆向工程领域中的里程碑，OMEGA 的主要贡献在于将提取器和抽象器分开，CIA 的贡献是采用了概念模型和将源与数据库隔开。DATA-tool[3]是一工具原型，能在 PASCAL 环境中做数据流分析，它用 PROLOG 字典存放对象之间的关系，并用 PROLOG 语言进行推理和查询。DATA-tool 的开发者认为他们的原型具有进化的特点，有较大的柔性。另外，含有数据库与编辑器集成的系统有 Interlisp 上的 Masterscope，Rice University 研究的 FORTRAN 环境上的 R^{n}，围绕一中心数据库而建立的系统有 Brown University 研究的 ENCORE，等等。在国内，这类型的研究很少。第二类工具由于采用了通用数据库查询，有利于信息的查找，并且由于概念模型的引入和逻辑推理的使用，使有些工具有了一定的柔性，但所产生的文挡中仍然缺乏维护所需要的信息，从源程序中所提的信息多限于程序的低层概念，例如控制流和数据流等。

第三类研究开始于80年代后期，其特点是采用知识推理等技术识别程序概念，PAT[4]是基于知识的程序分析系统，它将程序转变为语言无关的对象集合，称为事件，利用被称为规划的程序分析知识在事件库中进行推理，以产生新的事件。该系统考虑了专家对程序分析理解的模型，并采用了领域知识，能用特定领域的概念代替算法概念。KUP 是华东工学院开发的基于知识库的交互式软件理解工具。BAL/SRW[5]是从 IBM/370汇编语言程序中恢复设计概念的基于知识的逆向工程平台，它的自动代码模式理解部件能将汇编代码的连续的行与它的功能目标相联系，PROMPTER 是 IBM 日本科学院研制的一种基于知识的代码理解工具，它能产生汇编语言书写的程序的注释。68000C 反编译系统和8086C 反编译系统中分别实现各自的数据类型恢复器，类型恢复器采用基于规则的推理方法，用信息提取规则对汇编级或中间语言级的程序进行信息收集，然后对收集的信息用综合规则进行推理，得到 C 语言的数据类型表示，例如 struct、union 等，这些类型表示相对于汇编级和中间语言级是高层概念。第三类研究的优点是知识的使用和程序分析的专家模型的引入。存在的问题：①对程序分析的专家模型和领域知识的使用研究不够；②在识别高层概念时遇到困难，这是因为与高层概念相关的源代码可能分散遍布整个程序，同一概念的实现上的差别。

第四类研究开始于90年代初，这类研究的特点是采用模式匹配、程序变换等技术识别程序的高层概念，例如标准算法、数据结构等。实际上，在前三类研究中，模式识别技术也已被使用，但第四类研究中的模式识别具有新的特点，其目的在于解决识别高层概念时遇到的困难。C. Rick 和 L. M. Wills 提出一种铅板识别方法[6]，先将程序翻译成语言无关的图表示，然后在图上进行模式匹配。W. Kozaczynshi 等提出采用程序变换技术的高层概念抽象方法[7]。这类研究的目的是希望得到维护所需要的信息。更进一步的研究采用模式语言和程序变换的规范说明语言技术，使系统具有进化性，以解决概念的实现上的差别所造成的困难和应用领域知识的困难。这些研究有 D. Hildum & J. Cohen 提出的一种程序变换的规范说明语言[8]，有 S. Paul & A. Prakash 提出的模式语言和源代码查找系统的原型 SCRUPLE[9]。我们在 C 语言反编译系统 DECLER 中采用了程序变换的规范说明语言技术实现了数据类型恢复的 AB 变换器，并且对变换规则采用了四元组表示，这样可在变换中加入知识和修改知识库。这类研究的最大的优点是系统的进化性，用户参与对系统性能的改进，使系统越用越完善。但是，这类研究开展不久，有许多问题有待于进一步研究，例如程序变换的规范说明语言的研究，增式规则库和增式知识库的研究等。

这四类研究各具优点，第一类的可视化，第二类的数据库的使用，第三类的知识推理和第四类的进化性，都是自动程序分析工具所需要的。但总的看来，目前的程序分析工具对程序员理解、维护程序的支持还是很有限的，其中一个重要的原因是对专家分析程序的工作模型研究不够。

二、程序分析的专家模型

专家分析程序一般使用非形式的方法，不仅没有人使用形式验证方法，而且也没有人严格地作数据流和控制流的提取。专家根据他自己分析程序的

动机，跳过他认为不重要的部分而只关心他认为重要的部分。而且，认为程序分析专家在分析程序之前对被分析的程序一无所知，这是不现实的。专家正是充分利用已知的知识使程序分析尽量的简化。下面我们从几个侧面分析专家的工作模型。

1. 专家分析程序的动机。专家分析程序的动机不同，他所采用的工作模型也不相同。分析动机可概括为四种：①查错，在运行中发现程序的错误，需要通过分析找出引起错误的根源；②确认，是非行形式的验证方法；③理解，通过程序分析抽象和概括程序的功能描述。①查找，查找程序中的特殊碎片。各种实际的应用都可看作这几种动机连同程序改动等动机的组合，例如，修改是理解、改动和确认的组合。

2. 被分析的程序的状况。根据程序对分析的支持程度可分为：①具有良好的注释；②无注释或注释不好；③可动态调试；①不可动态调试。专家在分析程序时尽量使用程序文本中的显式信息或动态调试得到的信息。

3. 程序分析的维向。专家分析程序，由于动机的不同，可沿三个维向简化对程序性质的分析[10]：①模块化，分析的复杂性通过分割而减小；②选择，分析被限于被认为比其它更重要的性质或步骤；③抽象，省略细节和关心更一般的性质。

4. 程序概念。程序包含许多类型的概念，一般分为语言概念和抽象概念。语言概念包括变量、宣称、模块和语句等语法实体，它们由程序设计语言的语法规范说明定义。抽象概念描述计算和问题求解方法的语言无关的思想，它们进一步被分为：①程序设计概念，它包括一般的编程策略、数据结构和算法；②体系结构概念，它们与执行环境有关，例如操作系统、网络和数据库等；③领域概念，它们是由代码实现的应用领域的事件，例如学生成绩管理程序中"求数学平均成绩"等。

三、程序分析方法学

程序分析虽然是非形式的过程，但包含有一些形式的子过程，而且专家模型只能成为程序分析方法学的研究参考，而不能替代程序分析方法学。

1. 程序分析的基础

程序的性质决定了程序分析的基础。程序的性质和对程序分析的影响如下：

①程序的两面性。程序是用高级语言编写的，是既面向机器又面向程序员的。因为要面向机器，为适应机器的冯·诺依曼风格，过程性是其基本特征，这就是程序在计算世界中被理解。又因为要面向程序员，要求程序在现实世界中被理解，这样就出现了矛盾。但是，程序员开发程序实质上是问题从现实世界向计算世界的映射。程序分析是从计算世界向现实世界的反向映射，尽管反向映射可能不是一对一的映射，但这种反向映射关系是存在的。如图1所示。

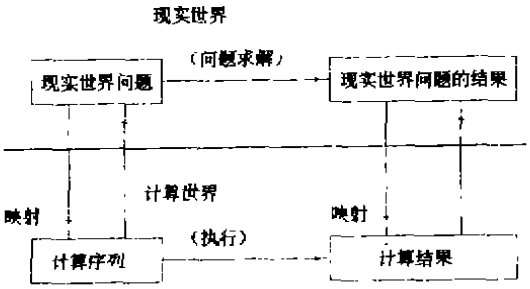

图1　现实世界和计算世界之间映射

②程序的分析复杂性。程序的分析复杂性不只是由于程序的语句的量引起的，而主要是由于程序设计语言的语义的颗粒度过细造成的。人脑所能容许的复杂度是相当有限的，据心理学试验表明，人脑的短期记忆的最大单元容量为7～9。解决这一矛盾的方法有二种，一是降低被管理对象的复杂性，二是用工具或方法辅助大脑管理，以驾驭复杂的局面。这两种方法是程序分析的主流方向。

③程序的执行性。从现实世界角度看，程序是一串符号，赋以现实世界的语义，是对现实世界问题的描述。但是，它是与领域无关的描述，因此是过分详细的。而从计算世界角度看，程序也是一串符号，赋以计算世界的语义，是对计算步骤的描写，因此，它是可执行的。程序的可执行性决定纯粹的静态分析不是最有效的，应当静态和动态分析相结合。

④程序的层次性。结构化程序设计的思想使程序在控制结构上有了层次性，面向对象的思想使程序在数据结构上有层次性，这些都是降低程序的复杂性的基础。

2. 程序分析的活动

如上面所述，程序分析主要沿两个主流方向：降低复杂性和计算机辅助管理。降低复杂性主要用抽象的方法，通过逐层抽象，得到程序的各层的概念，直到领域层的概念。计算机辅助管理主要以程序隐含信息的显性化、信息的可视化和支持信息查询为内容。信息的显性化和查询也可看作在另一方向上的降低复杂性的抽象。下面分别介绍。

①程序概念的抽象，概念抽象是以简单的高层概念代替复杂的低层概念。一个高层概念能代表低层概念群的一类。但是，程序概念的抽象不是纯省略性的抽象，而是以一定的知识的共享为背景的抽象。共享知识是实施程序概念抽象的主要难点。共享，意味着程序分析员与程序概念抽象器之间的共享，其难度可想而知。

②程序隐含信息的显性化。显性化的典型的例子是程序的控制流分析和数据流分析。这些信息是程序已有的，只是将它们显性化。显性化也是一种抽象，是纯省略的抽象。有些隐含信息隐藏颇深，使其显性非常困难，例如由多层指针变量而影响的变量的引用定值关系等。

③程序信息的可视化。可视化的作用是用实物演示弥补大脑思维中的表象演示的不足。在程序分析中可视化的主要困难是图迷航问题。

④程序信息的查询支持。查询可以看作对已显性化的信息再次使用纯省略的抽象。有效的查询依赖于信息的合理的存储和好的查询机制。

3. 结构程序代数

对于结构程序的自底向上的抽象的理论基础是结构程序代数，这个代数中的一条公理是代换公理。

定义 真程序是具有下述控制结构流图的程序：①有唯一的执行入口和唯一的执行出口；②对于每条语句，都有从执行入口到执行出口的执行路线通过。

代换公理：令 P 是 Q 的真子程序，【P】是 P 的语义，并令在 Q 中以 P' 代换 P 得到 Q'，则【P】=【P'】——→【Q】=【Q'】

定理 对于结构程序，存在一个结构程序代数。

结构程序代数是结构程序阅读、编写和验证的重要理论基础。

4. 程序注释

程序注释一般是非形式的，但形式化的注释有利于程序分析。注释分为两类：数据注释和真程序注释。真程序注释又分为状态注释和函数注释。程序证明中所使用的断言就是很好的状态注释。真程序对数据的作用相当于一个映射函数，该函数是很好的函数注释。

结束语 理想的程序分析支撑环境应当是与程序分析的专家模型相一致，从这一角度观察，现在的各类程序分析研究分别只反应了专家模型的不同的侧面。程序分析方法学提供了程序分析的方法和工具的理论依据。建立在专家模型和方法学基础上的程序分析环境将另行文介绍。（参考文献共10篇略）

（上接第86页）

对软件图形化表现技术的基本要求，我们相信它是目前程序图形化表示方面较为简单、能力较强的方法之一。它对于开发各阶段的平滑过渡、文档及程序代码的一致性维护、鼓励最终用户介入、新型 CASE 工具的研制等方面在程序表现技术方面提供了新的表示手段和基础。

目前，我们已经研制出基于抽象逻辑结构图的程序设计支援工具[6-9]，工具已能支持 foxpro 和 C++语言的程序开发。

参考文献

[1]James Martin, Carma Mcclure, Action Diagrams: Clearly Structured Specification, Programs, and Procedures, 2nd. ed. New Jersey: Prentice-Hall, 1989

[2]Leonard L. Tripp, A Survey of Graphical Notations for Program Design-An Update, ACM SIGSOFT Software Eng. Notes, 13(4), 1988

[3]S. P. Maj, Language Independent Design: An Introduction, Oxford: NCC Blackwell Limited, 1991

[4]Yaohan Chu, Software Blueprint and Examples, Lexington: D. C. Heath and Company, 1982

[5]James Martin, Principles of Object-Oriented Analysis and Design, New Jersey: Prentice-Hall, 1993

[6]M. A. Jackson, Principles of Program Design, London: Academic Press INC1975

[7]何克清，计算机软件工程学，武汉大学出版社，1983

[8]刘建宾，FOXPRO 程序设计支援工具 FPDST 的设计与实现，数据库研究与进展'95—全国第十三届数据库学术会议论文集，哈尔滨工业大学出版社，1995. 12

[9]刘建宾，C++函数开发工具 CFDST 的设计与实现，中国计算机科学技术新发展 - 中国计算机学会第九次全国学术会议论文集，西南师范大学出版社，1996. 5