遗产软件系统理解方法剖析

Analysis of System Comprehension For Legacy Softwares

杜 林　赵合计　（山东大学计算机科学与技术学院　济南　250061）

摘要：本文介绍了三种系统理解模型，对具体方法起框架指导作用，没有对系统理解的已有理论进行赘述，而是立足于各种理解方法的本质、异同和适用场合来对这些方法进行归类研究，在阐明方法思想的基础上，强调的是方法间的比较、不足之处、改进的建议和继续研究的方向，这对方法间的借鉴和多种方法的综合运用很有意义。

关键词：遗产软件 系统理解模型 系统理解方法

1 引言

伴随着信息技术的飞速发展，大量的遗产软件遗留下来。所谓遗产软件是指仍在使用并且需要更新的软件系统。由于软件的多次修改导致系统结构混乱以及文档的缺乏等许多因素使得理解遗产软件是一件很困难的事情，但是开发一个全新的系统，又是一个需要耗费大量人力和物力的工程，所以遗产软件是一笔不可忽略的财富。系统理解是一个从程序中获取信息的过程，可用于系统维护、重用、更新以及文档的整理等许多方面。系统理解的任务就是建立从程序设计领域到问题领域的映射，范围包括从代码本身的模型到应用领域的模型，其目的是为了便于系统的维护、升级和再工程。

2 系统理解模型

理解一个软件系统实际上就是抽象该系统的更高层描述的过程，抽象过程的目标产品就是系统理解模型。经常使用的模型有以下三种：

（1）基于自顶向下的模型。如 Brooks 模型和 Soloway 模型。用于代码已知且对代码比较熟悉的情况下，利用已有的知识公式化假设，把系统分解为子系统，最后分解为功能独立的代码块。

（2）基于自底向上的模型。如 Pennington 模型。在对代码不熟悉的情况下，逐行理解代码，发现相似的模式和模式的集合，实现更高层次的抽象。

（3）综合模型。通过自顶向下的方式推进过程，当遇到不熟悉的代码时，回溯进行自底向上的研究。对于一个系统的理解，综合运用多种策略是最理想的方法，综合模型就利用了自顶向下和自底向上两种策略。综合模型如图 1 所示：模型分为 4 个部分：程序模型处理过程、状态模型处理过程、自顶向下处理过程和知识库，其中知识库是构造前 3 个部分的基础。

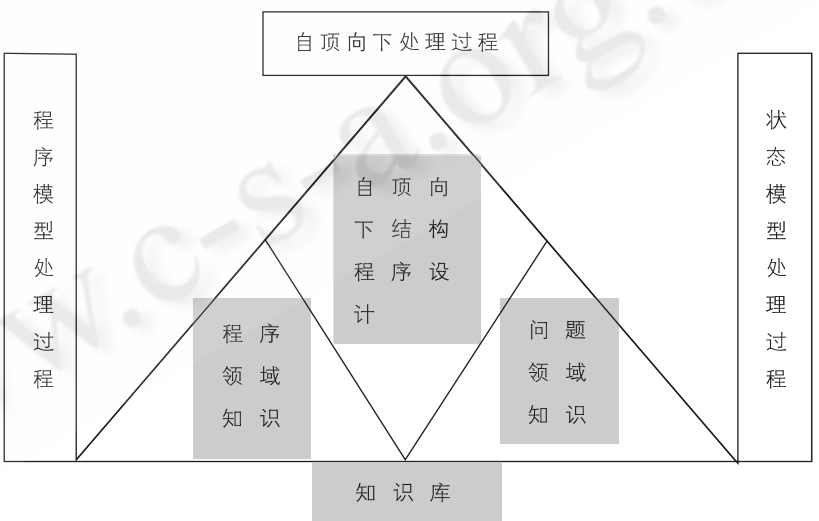

图 1　综合模型原理图

3 系统理解方法

系统理解是通过分析和抽象从软件中获取知识的过程，在更高的抽象级别上展现程序的基本模型，建立从程序设计领域到问题领域的映射。简单地说就是搞清楚程序是做什么的以及是怎么做的，需要理清程序

的数据结构、控制结构、输入输出特性、采用的算法、功能等。目前存在着许多正在使用和发展中的理解方法,下文从方法的本质和适用场合来对这些方法进行归类研究,在阐明方法思想的基础上,重点论述了方法的现状、不足之处并提出了改进的建议和需要继续研究的方向。

3.1 对象标识技术

遗产软件中很大一部分是基于面向过程的,理解并提高这些软件结构性的一个可行途径就是将系统移植到面向对象的技术中,进行对象标识是最基础的一步,然后从对象和方法出发,分析出类的层次结构,进一步完成整个系统的软件结构。对象标识一般包括以下几个步骤:

(1) 标识候选类;

(2) 标识候选方法;

(3) 通过对象标识为每一个方法选定一个最适合的类。

最典型的对象标识方法是簇分析方法和概念分析方法:

3.1.1 簇分析方法

分析对象是代码中的过程(函数)和全局变量,把局部变量直接封装在过程中,通过在数据集合中发现相关项来实现类的识别。方法是先构造反映全局变量和过程间关系的矩阵,以矩阵的行向量作为变量的坐标,并计算变量间的距离来决定之间的相似程度,用等级簇的聚合算法生成一棵系统树图,得到最终簇,每一个簇代表一个可识别类,簇中的变量为类中的属性。

3.1.2 概念分析方法

也是构造反映过程和全局变量间关系的矩阵,但是并不对变量进行分类,而是构造一个概念,一个概念表示某个过程集合使用的最大变量集合,每一个概念是一个候选类,区别于簇分析法的是它不是在距离的基础上找到最佳的组,而是利用格表示所有可能的关系。

3.1.3 两种方法的比较

(1) 分割方案。等级簇的聚合算法中,距离相近的簇可能存在多种合并的方案,只能选择一种,这样就不能显示所有可能的行为,而概念分析提供了一个概念层次结构,而不是唯一的分割方案。

(2) 过程与变量关系。簇分析只依赖于变量的距离矩阵的分析,没有体现所涉及的过程,而概念分析可以做到。

(3) 类的继承。簇分析不能为继承的依附关系提供信息,而概念分析法中,子概念的关系和类的关系是一致的,如果概念 a 有一个子概念 a1,,那么类 a1 就可能继承类 a。

(4) 简便性。概念分析法需要手动参与类的分割,而簇分析的结果是一个确定的方案。

3.2 基于语义的方法

从语义的角度对程序进行分析,常见的有切片和基于指称语义的方法:

3.2.1 切片方法

这是一种基于语义分解的方法,从语义的角度将源程序进行分解,根据一个切片标准提取出程序中影响关注点变量值的相关语句,通过这些语句可以方便地分析程序,如方法调用和对象调用等。目前的程序切片大都是以系统依赖图为基础,在其上利用图的可达性算法获得的,该方法的处理流程如图 2 所示:

3.2.2 指称语义

指称语义是用来描述程序语义属性的一种方法,通过指称语义分析,可以获得程序语义的形式化描述。

(1) 方法原理。方法中定义了一些称为语义域的数据类型来描述程序的一些含义,比如用类表域反映变量和值的关联,基于原始的语义域来定义替换函数,要确定一个变量的值是否改变,可以为变量的赋值语句解释语义函数,通过一个布尔变量来标识变量是否改变,而不需要考虑变量值的大小。

(2) 分类及比较。指称语义分为直接指称语义和接续指称语义,其中直接指称语义适合于结构化语言,接续指称语义适合于非结构化语言。指称语义的描述主要包括语法域、语义域、抽象语法和语义方程。对于同一种语言来说,两种指称语义描述的语法域和抽象语法是相同的,不同的是语义域和语义方程部分。两种语义中接续指称语义描述的难度比较大。

(3) 发展方向。如何完成两种指称语义的自动转换,是今后的研究工作,有的研究人员利用基于范畴论的 Monad 技术做了这方面的探讨。

3.3 针对某个关注点

对于大型系统而言,存在着大量的全局变量及语句间的嵌套,我们很难从整体上理解这个系统,但是可

以针对系统的某些特殊行为和我们的关注点来进行分析,下面介绍部分求值和植入技术:

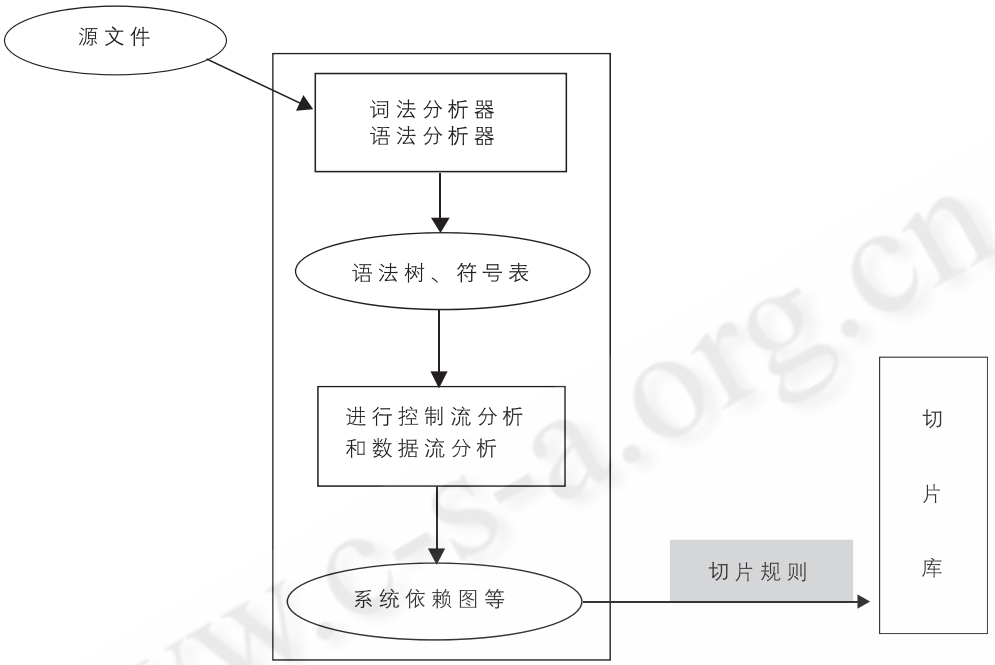

图 2　切片方法处理流程

3.3.1 部分求值

针对系统的某些特殊行为,利用部分求值技术进行分段计算。

(1) 方法原理。对于部分求值技术的研究主要集中在程序例化技术方面,这种方法将程序分为两个阶段:首先在例化阶段参照已知的部分输入数据,完成与其相关的部分计算,优化控制流,通过程序变换,将计算结果变换成程序代码,生成例化的程序,然后,运行这种例化的滞留程序完成其余计算。

(2) 存在的问题。虽然程序例化技术实现了分段计算,使例化程序获得了较高的执行效率,但是带来了代码膨胀问题。比如,在程序例化过程中,循环条件已知的循环语句将被展开为重复的循环体代码段,部分参数已知的函数将根据不同的输入参数变换为多个例化函数。这种代码膨胀可能会抵消程序例化带来的性能优化。

(3) 解决方法。鉴于以上问题,有些研究人员在例化中引入了数据例化手段[2]。数据例化通过程序分析,选择计算的部分结果,将计算分为 2 个阶段,先进行前段计算,将选出的中间计算结果保存在数据缓冲区内,在后段计算时,可以直接利用缓冲区内的中间计算结果,从而获得性能优化。

(4) 继续研究的方向。在今后的研究中,对于绑定时间的分析也需要进一步开展。绑定时间分析是一种程序预处理技术,用以确定程序中哪些计算(静态部分)在部分求值阶段完成,哪些计算(动态部分)保留在作为部分求值结果的滞留程序中。如果我们可以对绑定时间进行更加精确的分析,不仅能够分析一般变量绑定时间的上下文敏感性,而且能够分析数组变量和对象引用变量等各种变量,就可以将部分求值处理深入到复杂的数据结构内部,从而扩大部分求值的作用范围。

3.3.2 植入技术

这是获取系统运行时表现出来的动态信息的一种方法,用于对关注点的研究。

(1) 方法原理。利用代码的静态结构信息,依据固定的规则,将软件触发器添加到代码中。软件触发器是在源程序中关注的位置添加的一些不影响原程序语义的代码,运行时由这些代码按特定协议将动态信息传递到指定的动态信息收集设施,可获取对象之间的消息传递等信息。软件触发器的植入就是对被植入的原程序的修改,目前的一些方法是在工具支持下进行的交互式修改,如 SCED 和 ISVIS 中的包装器方法就是实现了自动植入。

(2) 存在的问题。目前存在的自动植入技术是直接干预目标系统源代码,植入代码和目标代码处于同一计算层次,存在的问题是:

①植入过程控制不当,可能会破坏源代码的正常运行;

②为了确定植入点,需要对目标代码进行除编译

分析之外的额外的程序分析,没有充分利用编译阶段的代码分析过程。

(3) 解决方法。为了解决以上问题,可以考虑将植入的软件触发器和被植入的应用程序作为两个计算层次来处理。有研究者提出了反射植入的方法[3],植入时,利用反射思想和开放编译技术,将植入的软件触发器和被植入的应用程序作为两个计算层次来处理。反射原理是将一个系统看成相互依赖的层次化计算体系,改变下层的计算逻辑,会影响上层的计算逻辑,可以部分地改变整个系统,从而影响自身的计算。开放编译技术是将编译过程用元对象协议 MOP 向用户开放,允许用户自己通过编制实现元对象的程序片段来干预编译过程,以达到特定的应用目的。这样,应用程序的植入过程是在编译时进的。在应用程序交付编译之后,由这组元对象对应用程序进行软件触发器的植入,再透明地向常规编译器提交,链接时链入所需的运行时支持机制,这就达到了将软件触发器和被植入的应用程序作为两个计算层次来处理的目的。基于反射机制,充分利用编译阶段的代码分析过程,就可以规范地完成程序植入。

3.4 文字编程

这种方法立足于系统的理解,而不仅仅是系统的实现,要求程序员不仅要从机器的角度书写代码,更重要的是从便于人的理解角度书写文档,强调与人的交流,而不限制于编译器的表达方式。以陈述问题及解决方案为中心,把程序代码看作给编译器的注释。

(1) 方法原理: 如图 3 所示。

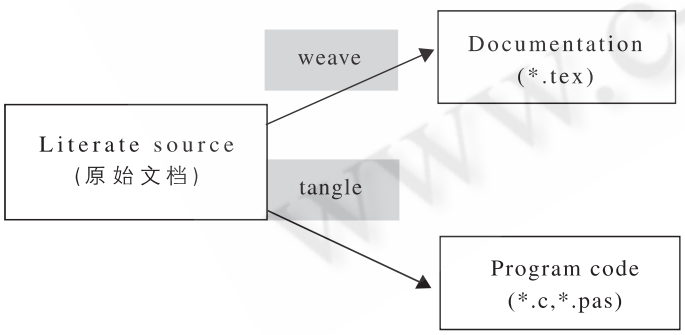

图 3　文字编程系统模型

系统通过 weave 和 tangle 两个过程把原始文档处理成供编译器理解的代码和供人理解的文档。把代码和文档放在一起,代码修改要伴随着文档的修改。文字编程系统同时集成了计算机编程语言(如 Pascal, C 等)和文字排版语言(如 TEX, LATEX, HTML 等)。也就是说书写文字程序要同时熟悉编程语言和排版语言。目前已经存在一些开发环境,如 CWEB、noweb、nuweb、FunnelWeb、Clip、Interscript。

(2) 存在的问题。通过文字编程,程序员可以像阅读文章一样阅读程序,但是并不能准确把握程序的整体行为,原因在于阅读描述性文档只是容易理解局部代码的功能,很难从整体上理解程序的结构。

(3) 解决方法。根据文字编程系统的特点,原始文档被分解成小的"文档块",文档块的定义为了便于理解可以以任何次序组织,但是文档块的引用必须以实际可执行代码的排列次序为准,所以可以根据引用次序提取文档块之间的嵌套引用关系,构造反映程序结构的图形,从而在整体上对系统有一个直观的认识。

(4) 继续研究方向。书写文字程序需要掌握编程语言和文字排版语言,开发一个更加友好的编程环境是今后的工作。

3.5 程序信息库技术

方法的思想是:把程序的信息存放于关系数据库中,通过一些分析工具从信息库中提取对用户有用的信息。

(1) 方法原理: 处理流程入图 4 所示。

程序信息库的建立一般分为以下 3 个步骤:

① 建立概念模型: 一个目标程序语言的概念模型定义了软件在指定抽象层次上的程序元素和元素间关系,概念模型是一个信息库系统的功能描述,决定了从该信息库可以获取知识的范围。

② 建立关系视图: 在建立概念模型以后,要建立一个分析器去分析程序文本,并依据概念模型建立视图关系,在这个阶段程序的文本信息被转换成了关系数据库。

③ 建立分析工具: 在建立关系视图后,通过一些分析工具从信息库中提取对用户有用的信息。

(2) 存在的问题。现在已有一些这样的系统,如用于 Pascal 的 Omega 和用于 C 的 CIA,但是这些系统实际使用价值不高。比如在 Omega 中,将程序的所有信息存放于 58 个关系中,即使是对一个小的项目也要维护大量的数据,效率很低。在 CIA 中,分析只能达到一个较高的层次如函数级,无法提供变量等具体信息。

(3) 解决方法。以上问题主要是由于概念模型的问题,模型的抽象层过高无法提供详细信息,反之则会使信息库过于庞大,关系过于复杂。基于以上问题,我们希望信息库包含充足信息,库又不要太庞大,可以考虑改变把对象和关系分开存储的传统方法,将实体的关系包含在实体的属性中。为每个程序元素赋予唯一的标识,同时每个对象有一个属性标识,用于标记其所属的对象,比如对于一个类的成员,其属性编号为该类的编号。

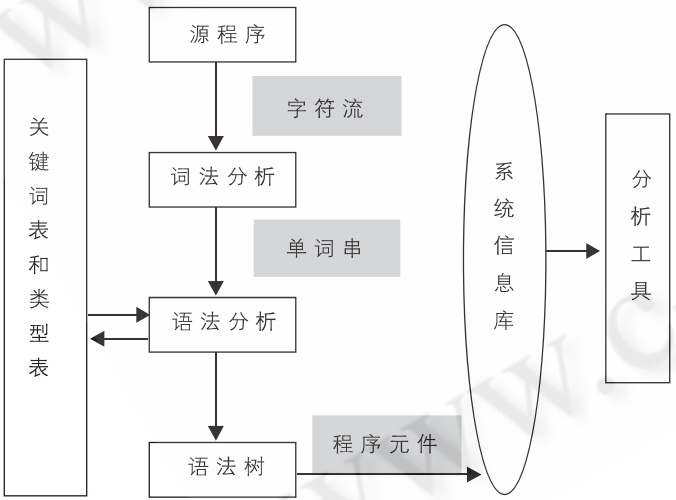

图 4　信息库技术处理流程

4　结论

基于以上研究,各种系统理解方法都存在着不足之处,都有进一步研究的必要性。如何能综合运用各种方法的成果,在多种方法间的借鉴和结合进行研究,从而对系统理解做得更加高效也是很有意义的研究领域。另外,如构件技术和分布式技术等新技术的使用也使得系统的结构发生了变化,这对系统理解提出了新的挑战。

参考文献

1　易彤、吴方君,“一种基于覆盖测试的动态切片的计算方法”,《应用科学学报》, VOL. 22. NO. 2 June. 2004。

2　廖湖声,“基于程序流程图的数据例化与程序例化”,《计算机学报》,VOL. 24. NO. 9 Sept. 2001。

3　李青山、陈平、王伟、宋海鸿,“逆向工程中反射植入的研究”,《计算机学报》, VOL. 27. NO. 4 Apr. 2004。