语言的抽象、封装与变换型开发方法*

张乃孝[1] 郑红军[2] 裘宗燕[1]

[1](北京大学数学科学学院信息科学系 北京 100871)

[2](北京大学计算机科学系 北京 100871)

摘要 该文提出了一种系统的软件开发方法——面向模型的变换型软件开发方法.这种方法把数据抽象的思想提高到语言抽象层次,把一类专用软件的规范抽象为语言的规范;把这类软件的实现抽象为语言的归约变换;用"规范+变换"抽象软件开发过程.为支持这种方法,提出了语言的一种抽象与封装机制Garment,以此定义语言中各成分的语法和语义,描述语言间的继承、屏蔽和扩充关系.最后,以语言知识库为核心,简要介绍了支持用Garment进行变换型开发的系统结构和工作流程.

关键词 形式方法,软件模型,变换型方法,语言抽象,语言封装,语言族,语言知识库,软件重用,Garment.

中图法分类号 TP311

科学研究中形成的学科语言(如数学语言,化学语言等)面向专门领域,可用于方便地描述所研究领域的对象和处理过程.从这种意义上讲,一个领域的专用软件应能处理(实现)其领域语言,为该领域提供一个虚拟现实的研究环境,从而使之成为该领域的抽象计算机:提供模拟领域对象及其处理的强有力手段,使之成为一种能够方便地建立领域具体问题求解模型的程序设计环境.这种软件实现了一整套在新层次上的计算功能,为原计算机系统披上了一件外衣(Garment),改变了原系统与用户之间的界面.最早成功的专用软件应该算程序设计语言系统.高级语言是计算机科学家为程序设计领域建立的抽象模型,语言系统为程序员提供了一个良好的程序设计环境.

在这种专用软件的开发与使用方面,软件开发和程序设计的划分非常清楚,就象在实验科学中建立一个实验室和用这个实验室做某项实验,也类似于在理论研究中建立公理系统和在这个系统中证明具体定理的关系.

我们认为,根据目前计算机科学的发展水平,有必要也有可能把软件理论与程序理论在一定程度上分离,以便能集中精力研究软件本身的性质和软件开发的自身规律.特别是有必要把语言理论与软件理论的现有成果结合起来,寻找一种统一的概念,以统一的观点和模型研究软件和软件开发过程.[1,2]

1 面向模型的变换型软件开发方法

从以上的基本观点出发,我们提出一种系统的软件开发方法,称为"面向模型的变换型软件开发方法".[3]这种方法把程序开发、软件开发和软件开发环境的开发视为3种不同层次上的活动,它们之间的关系可用图1来表示.

环境开发者从软件的理论出发,为软件的开发建立一个统一的软件生成环境,包括为软件开发者提供一个有效的软件开发语言和这个语言的解释器(或编译器).软件开发者以学科领域的抽象机为背景,使用软件开发语言描述所要开发的领域语言及其实现方法,通过开发语言的解释器生成领域语言的编译器.程序开发者则从学科领域实际问题出发,使用该领域专用软件所实现的语言描述实际问题的求解模型,通过该语言的编译器将其转换成计算机可运行的,能够解决实际问题的程序.

面向模型的变换型软件开发方法发展了VDM的模型概念,把软件理论的一般模型、学科领域的抽象模型和实际问题的具体模型加以区分,分别研究它们的计算机实现;并根据面向对象的原理[4],发展了数据抽象概念[5],把抽象数据类型作为语言定义的基本单位,把程序设计语言作为软件理论研究的基本对象,把一类专用软件的规范抽象成语言的规范,强调了软件理论与语言理论的结合,在此基础上探讨语言间的继承、屏蔽和扩充关系,把数据抽象升华到语言

* 本文研究得到国家自然科学基金资助.作者张乃孝,1942年生,教授,主要研究领域为程序设计方法学及新语言.郑红军,1969年生,博士生,主要研究领域为程序设计方法学及新语言.裘宗燕,1952年生,副教授,主要研究领域为计算机语言及符号计算.

本文通讯联系人:张乃孝,北京 100871,北京大学数学科学学院信息科学系

本文1997-04-25收到原稿,1997-07-02收到修改稿

抽象的层次.[6,7]

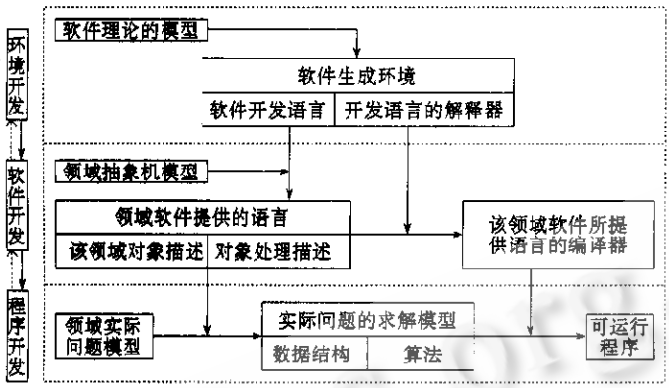

图1　环境开发、软件开发和程序开发

2　语言的一种抽象与封装机制

要实现面向模型的变换型软件开发方法,首先要提供一种语言的描述机制,用于描述专用软件所需实现的专用语言,我们称这种语言描述机制为 **Garment**. 一个 Garment 规范就是一个语言的抽象与封装. 下面将介绍 Garment 的主要语法结构.

为了描述方便,这里对 BNF 做了一些扩充:[A]表示元素内容为 A 的非空序列;[A]-s 表示以 s 为分隔符、元素内容为 A 的非空序列;{A}表示 A 可缺省;黑体字表示保留字. Garment 的结构如下.

```
garment::=garment id1 from id2          /* 首部 */
        spec                            /* 说明 */
        import inher_part;
            [type_def | struc_def ]-;
            {with op_list}
        impl                            /* 实现 */
            [trans_def]-;
    end id1
```

Garment 在形式上以保留字 **garment** 引导,分成说明和实现两个部分,分别用 spec 和 **impl** 指明. 在 Garment 中,将语言成分分为数据结构(数据抽象)和控制结构(控制抽象)两类;数据结构以抽象数据类型(简称抽象类型)表示;控制结构以语句结构表示;过程抽象包含在数据抽象中. spec 部分定义语言 $id1$ 中各成分(抽象类型和语句结构)的语法;**impl** 部分以归约规则形式描述语言 $id1$ 中的语言成分相对于其父语言 $id2$ 的归约语义[5,8],从而将语言的语义描述与实现方法统一起来,因此,语言 $id2$ 既是 $id1$ 的语义描述语言,又是它的实现语言.

具体地,继承关系说明部分 inher_part 的语法为

$$inher\_part::=[id3]\text{-},\ |\mathbf{all}\ \{\mathbf{except}[id3]\text{-},\}$$

其中 id3为语言成分名,inher_part 中的第1种说明方式是枚举所有要继承的成分名;后一种方式则仅把要屏蔽的成分列在 **all except** 之后,其他成分自动继承,包括全部继承(**all**).

spec 部分的 type_def 用于说明扩充的抽象类型,其语法结构为

```
type_def::=type id4 { '[' parameter_part ']'}
            {literal [literal_part]-;}
            operations [operations  part]-;
        end
```

其中 $id4$ 为被定义抽象类型名,parameter_part 为抽象类型的参数. 例如,抽象的栈类型可记为 Stack[α],其中的 α 是栈类型的参数,简称类型参数. 由于类型参数的行为类似于变量,故有时也称类型参数为类型变量.

语法中的 literal_part 和 operations_part 给出抽象类型的常量定义和操作的语法,操作可以采用过程、表达式、算

子和语句等方式定义.该部分所提供的信息对外部是可见的,这些信息包括操作名和操作的类型.

spec 部分的 struc_def 用于说明扩充的语句结构,其语法为

```
struc_def::=struc id5 {'['parameter_part']'}
            [struc_part]-;
```

在 **spec** 部分的最后,用 **with** 引导了一个 op_list,用以说明操作中算子的优先级和结合性.

impl 部分是由一组变换模块构成的,变换模块用于指明语言成分的归约语义,其语法如下.

```
trans_def::=transform id3 {'['parameter_part']'}
            {repr type          /* 表示不变式 */}
            rules [rule]-®
            {with [trans_decl]-;}
          end
```

一个变换模块描述语言 *id*1的成分相对于其父语言 *id*2的归约语义,所有扩充的语言成分都在变换模块中指明其语义,被屏蔽的成分不必给出其语义.对抽象类型,需用 **repr** 引入表示类型,表示类型应是父语言所能提供的抽象类型或具体类型,它用于所定义抽象类型的实现.在引入表示类型的同时,建议以注解方式给出抽象类型与表示类型之间的不变关系(Represent Invariant).**rules** 中的每个归约规则由左右两部分组成:规则左部给出应用该规则时应满足的匹配模式(语言 *id*1的程序模式);规则右部给出应用该规则进行归约时的代入模式(父语言 *id*2的程序模式).归约规则可将抽象的成分变换为更具体的成分,从而降低了该抽象成分的抽象层次,在这一点上类似于数据精化和过程精化规则.关于变换模块的具体定义、作用和变换正确性的讨论见文献[9,10].

3 Garment 的功能

Garment 能够对已有的语言 L 进行抽象,使得抽象出的语言 L' 对其应用领域具有较高的抽象层次,Garment 对语言 L 的抽象过程可简单地记为:$L'=Garment(L)$.Garment 描述了 L' 对语言 L 的继承、屏蔽和扩充,构成了 L' 的 Garment 规范.称 L 为 L' 的父语言.Garment 就象为语言 L 披上了一件外衣(garment),改变了用户与 L 间的界面,使之表现为 L'.对 L' 的应用领域来说,L' 的抽象层次比 L 的抽象层次高.

语言 L' 中的对象和操作表示了问题领域的一个模型,它可以通过 L 中的对象和操作模拟实现,我们称此过程为归约变换.在实现语言 L' 的程序时,将重用这些变换.L' 中的对象和操作最终由一种通用的可执行语言(如 C,Pascal 等)模拟实现,称这种可执行语言为核心语言(Kernel Language).所以,选择父语言 L 时主要考虑两方面因素:是否能从语言 L 获得更多的继承,使其中更多成分的定义和实现在语言 L' 中重用;是否能方便而有效地由语言 L 实现语言 L'.

每个 Garment 封装了一个语言的定义和实现,是一种更高层次的可重用部件.这种语言的抽象与封装机制,能够使得语言的开发具有层次性、系统性,可形成以核心语言为根的树形语言族,并保存于语言知识库中.程序设计者可以根据需要,在语言族中选择适合描述其领域问题的语言进行程序设计,或者在此基础上,利用 Garment 的抽象与封装机制抽象出一种新的、更适用的语言,以提高其应用领域的程序设计效率.

以 $L'=Garment(L)$ 方式抽象出的所有语言,在语言间的继承、屏蔽和扩充关系下构成了树形语言族,语言族中的语言相对于其各自的应用领域来说,具有不同的抽象层次.Garment 提供了对不同层次抽象语言的操作,如 Garment 语法中的 **import** 部分和 **transform** 部分所示,这样可以重用对整个应用领域的分析信息,如 Garment 语法中的 **all** 所示.同时,Garment 中的变换模块提供了不同抽象层次间的联系方式与可见控制:一个层次中的成分可以对应到低层次中等价的成分.这种多抽象层次的存在正是 Garment 能力的一个主要来源.

程序设计语言从机器语言、汇编语言到高级语言的发展就是一个语言抽象过程.这些语言都具有不同的抽象层次:汇编语言是机器语言上的一层抽象,高级程序设计语言是汇编语言上的一层抽象.每个语言抽象包括抽象规范和抽象实现两部分,一个层次中的抽象规范可以是更高层的抽象实现.对于 $L'=Garment(L)$,在 L' 的 Garment 规范中,**spec** 部分为语言 L' 的抽象规范,**impl** 部分为语言 L' 的抽象实现,抽象规范到抽象实现的映射由 Garment 开发环境 Garden 完成.

事实上,Garment 提供了一种能够重用领域分析和实现的机制,表现为语言的重用,如 Garment 语法中的首部所示,语言成为更高层次的可重用部件.在 L' 的 Garment 规范中,领域分析的结果在 **spec** 部分说明,其实现方法在 **impl** 部分说明,可重用部件是语言 L,从 L' 的抽象规范到抽象实现的映射由 Garden 完成.这一点类似于编译器——编译器

的工作过程及生成的目标代码是不可见的.由于语言 L' 的 Garment 规范归纳和体现了语言 L' 应用领域的共同性,所以在创建时只实现一次,而在使用这些语言设计程序时,每次都是重用对其应用领域的分析结果.这些工作虽然没有 Garment 也可以做,但 Garment 内在的力量在于它的语言重用.[11]

4 Garment 开发环境

根据 Garment 的定义,我们设计并实现了一个系统的软件开发环境 Garden (Garment development environment),用语言知识库[12]支持建立面向不同应用领域的、具有不同抽象层次的树型语言族.

Garden 在结构上分为两个过程:语言实现过程和程序归约过程.以 $L'=Garment(L)$ 为例,语言实现过程实现 L' 的 Garment 规范,即实现对语言 L 的继承、屏蔽和扩充,以形成语言 L' 的文本和编译器,并将其保存于语言知识库中.

程序归约过程在语言知识库管理机制的控制下,实现从抽象程序到核心语言程序的归约.程序归约是语言归约的具体化,其过程包括分析抽象程序的语法,并形成局部于该抽象程序的归约环境;类型检查和推理;属性匹配和程序生成.

语言知识库是联系和统一语言实现和程序归约的纽带,它不仅能为语言实现过程提供实现语言 L' 所需的语言知识,更重要的是,它是程序归约过程的基础.语言知识库中保存了语言族中各语言的全部信息,包括语言的语法信息和语义信息.

利用 Garden 进行系统开发的工作流程如图2所示.仍以 $L'=Garment(L)$ 为例,图2中依 Garment 的语法和语义生成的 Garment 解释器是整个环境的基础,其生成过程对应于图1的环境开发.

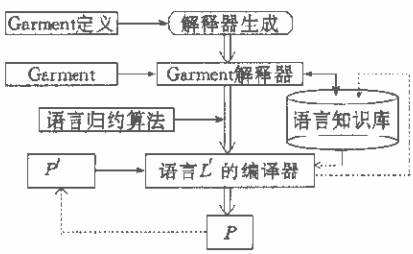

图2　Garden的工作流程

将语言 L' 的 Garment 规范(记为 Garment)经 Garment 解释器处理,与知识库中父语言 L 的信息相结合,形成 L' 的完整语言知识,与通用的语言归约算法汇合得到语言 L' 编译器,从而完成了语言 L' 的实现过程.这一过程对应于图1的软件开发.

在程序归约阶段,用语言 L' 描述具体问题,得到程序 P',P' 经语言 L' 编译器编译,得到一个与 P' 等价的语言 L 程序 P,若 L 为核心语言,则 P 就是最后的可执行程序,否则,知识库管理机制将从知识库中选取语言 L 的编译器,继续编译 P,直至得到核心语言程序为止.此过程对应于图1的程序开发.

Garden 是实现 Garment、支持面向模型的变换型软件开发方法的一个初步尝试.我们采用这种开发方法,以自应用(Self-application)方式开发了程序变换系统,在保证开发正确性方面取得了一定经验;又用变换型方法模拟开发了电话交换系统,初步验证了方法的可行性和有效性.[13]

5 结束语

本文提出了一种系统的软件开发方法——面向模型的变换型软件开发方法.这种方法把数据抽象思想提高到语言抽象的层次.为支持这种方法,本文提出了语言的一种抽象与封装机制 Garment,用以定义语言中各种成分的语法和语义,描述语言间的继承、屏蔽和扩充关系.最后简要介绍了 Garment 开发环境 Garden.

在 Garment 的设计和实现过程中,我们研究了软件理论和语言理论的已有成果,特别是类型理论、形式语义、程序变换、数据抽象、软件重用、部分实现和面向对象等方面的理论和技术;吸收了 Polya,VDM,CIP 和 Ada 语言中许多有益的成分和表示方式.与此同时,我们也对 Garment 的数学模型进行了初步探讨,为语言族中的语言建立了一个代数模型,以语言的这种模型描述了语言的继承、屏蔽和扩充的语义以及语言族的语义,并指出语言抽象是数据抽象的升华.[6]另外,我们还对 Garment 中的归约语义、多态类型系统及多态类型的 Ideal 语义[14]进行了研究.核心语言的充分性和语言扩充的一致性等问题均有待于进一步研究.

后记　软件开发形式方法的研究意义重大,但步履艰辛.10多年来,我们坚持沿着这个方向努力探索,取得了一些初步成果.本文是近年来部分工作的小结,藉以表达对我们的导师吴允曾教授的深切怀念,他离开我们已经整整10年.

参考文献

1　张乃孝等编译.关于发展计算机科学的几点建议.模式识别与人工智能,1990,3(4),32～36
(Zhang Nai-xiao *et al* translated and edited. Proposals for the development of computer science. Pattern Recognition and

Artificial Intelligence, 1990,3(4):32～36)
2 张乃孝等编译.三十年来计算机科学的发展和贡献.模式识别与人工智能,1990,3(4):1～31
(Zhang Nai-xiao *et al* translated and edited. Scientific contributions of computer science. Pattern Recognition and Artificial Intelligence, 1990,3(4):1～31)
3 张乃孝,许卓群,屈婉玲.面向模型的变换型软件开发方法研究.理论计算机科学,Vol 2.上海:上海科学技术文献出版社,1994.54～64
(Zhang Nai-xiao, Xu Zhuo-qun, Qu Wan-ling. On the model-oriented transformational method of software development. Theoretical Computer Science, Vol 2. Shanghai: Scientific and Technological Literature Publishing House, 1994. 54～64)
4 Danforth Scott, Tomlinson Chris. Type theories and object-oriented programming. ACM Computing Surveys, 1987,20(1):29～70
5 Cardelli Luca, Wegner Peter. On understanding types, data abstraction and polymorphism. ACM Computing Surveys, 1985,17(4):471～522
6 Zheng Hong-jun, Zhang Nai-xiao. An abstract model for programming languages. In: Proceedings of Changsha International CASE Symposium'95. Oct. 1995. 69～75
7 张乃孝,郑红军.程序设计语言的抽象与语言族模型.北京大学学报,1997,33(5):650～657
(Zhang Nai-xiao, Zheng Hong-jun. Abstraction of programming languages and a model of language family. Acta Scientiarum Naturalium Universitatics Pekinensis, 1997,33(5):650～657)
8 Partsch Helmut A *et al*. Specification and transformation of programming. Springer-Verlag, 1990
9 张乃孝.程序变换在程序设计语言中的一种表示——兼论变换型语言.软件学报,1993,4(5):17～23
(Zhang Nai-xiao. A notation of program transformation in programming languages——on transformational programming languages. Journal of Software, 1993,4(5):17～23)
10 张乃孝.程序变换过程的分析与设计.计算机学报,1994,17(6),473～476
(Zhang Nai-xiao. Analyses and designs for the process of program transformation. Chinese Journal of Computers, 1994,17(6),473～476)
11 Krueger Charles W. Software reuse. ACM Computing Surveys, 1992,24(2):131～183
12 张乃孝.知识结构的三叉树表示及逻辑推理的实现.计算机学报,1990,13(1):32～41
(Zhang Nai-xiao. The trinary-tree representation of knowledge and the implementation of inference procedures. Chinese Journal of Computers, 1990,13(1):32～41)
13 屈婉玲,张乃孝.用变换型方法模拟开发电话交换系统.计算机研究与发展,1995,32(7):11～16
(Qu Wan-ling, Zhang Nai-xiao. Simulation of developing a telephone exchange system by transformational method. Computer Research and Development, 1995,32(7):11～16)
14 郑红军,张乃孝.Garment 中多态类型的 Ideal 模型.软件学报,1998,9(3):194～199
(Zheng Hong-jun, Zhang Nai-xiao. An ideal model for polymorphic types in Garment. Journal of Software, 1998,9(3):194～199)

Language Abstraction, Encapsulation and Development with Transformational Method

ZHANG Nai-xiao[1] ZHENG Hong-jun[2] QIU Zong-yan[1]

[1](*Department of Informatics School of Mathematical Sciences Beijing University Beijing* 100871)
[2](*Department of Computer Science Beijing University Beijing* 100871)

Abstract A systematic software development method named "Model-oriented Transformational Software Development Method" is proposed in this paper. In this method, data abstraction is enhanced to language abstraction; the specifying of some specific software is abstracted to language specifying; the implementation of the software becomes language reducing (transformation); the process of software development here could be described as "specification+transformation". Following these ideas, a mechanism to abstract and encapsulate languages named Garment is designed. Relationships between languages, which are classified as inheritance, shielding and extension, are described with Garment. Taking a language knowledge base as its kernel, an environment named Garden (Garment development environment) is implemented to supports system development with Garment. Finally, the system structure of Garden and some development cases are described.

Key words Formal method, software model, transformational method, language abstraction, language encapsulation, language family, language knowledge base, software reuse, Garment.