软件构造过程中的对象规范和实现*

李 京　　冯玉琳　黄 涛

（中国科技大学计算机系，合肥 230027）　（中国科学院软件研究所，北京 100080）

摘要　面向对象的软件构造过程中的一个关键技术是对象的描述. 本文介绍的ADL语言就是用于描述软件对象的基本特征. 一个ADL对象定义分为两部分：其一是对象规范，它在抽象数据域上描述对象的语义特征，用逻辑公式规定对象的状态和行为约束. 另一是对象体，它由开发者选用相应的某种实现语言和算法给出对象规范的具体实现，同时给出从抽象数据到具体数据的映射关系.

关键词　软件构造，对象规范，对象实现.

面向对象设计方法是一种与传统功能方法完全不同的方法，它以基于数据抽象的对象为中心，使得按面向对象方法开发的软件具有较好的可扩充、可维护和可重用性.

我们于1991年提出SOP软件构造模式[1]，将软件视为对象模块的复合，而软件设计则是对象模块经过过程控制进行复合的构造生成，表示为：

软件(Software)＝对象(Object)＋过程控制(Process Control).

在SOP模型中，对象和过程控制程序是两种不同的功能单元. 对象代表了软件设计活动中所涉及到的各种软件产品和工具，其中最主要的一种是软件构件(software component). 对象的结构和行为是由SOP模型的数据定义语言ADL描述和实现的. 而过程控制程序是由一系列可以顺序或并发进行的过程控制活动(activity)组成，它模拟了人们管理、选择构件并组装软件产品的活动. 活动是由SOP模型的过程控制语言PCL规范和实现的.

本文主要介绍ADL语言，PCL语言将另文介绍. 下面首先介绍SOP采用的对象模型和ADL的设计原则，然后结合例子讨论ADL的主要功能和特点，最后给出ADL与常见OO语言的比较.

1　SOP对象模型和ADL语言的设计原则

在软件设计过程中，SOP模型视对象为一种数据抽象，作为软件构造的基本单元. 对象

* 本文1993－11－22收到，1995－01－10定稿

本项目得到国家863高技术和中国科技大学青年基金资助.作者李京，1966年生，讲师，主要研究领域为软件工程环境. 冯玉琳，1942年生，研究员，中国科学院软件研究所所长，主要研究领域为系统模型和语义理论，软件设计方法学，软件工程环境. 黄涛，1965年生，博士，主要研究领域为面向对象软件构造，软件设计方法学，语义理论，软件工程环境.

本文通讯联系人：李京，合肥 230027，中国科技大学计算机系

由数据及其所提供的服务组成，对象的内部状态是用一组属性记录来表示，它对外界不可见，外界只有通过发送消息来获得对象的服务或改变对象的内部状态. 理解对象可通过对象的语法接口和语义描述来进行，使用对象无需了解对象的内部细节. SOP 视对象为一动态演变的实体，对象具有其固有的对象标识. 对象除了受静态约束外，其演变历史还受动态行为约束规定.

对象可以是聚合的，一个对象可由一些子对象构成，父对象和子对象间体现了 part—of 或 client—server 关系. 从而提供了直接从简单对象构造复杂对象和支持对象共享的机制.

对象类将具有相同结构和操作特征的对象组织在一起集中加以描述，单个对象是类的实例(instance). 继承(多重)、多态性、操作重载和动态联编也是面向对象的重要特征.

SOP 还支持类的模板，即含参类和类属类(generic class). 根据不同参数对类属类进行实例化(instantiation)，可以得到一组类定义.

作为一种高级规范语言，SOP 模型中的对象定义语言 ADL 除了支持上述所有面向对象的特征外，还应支持软件设计的需求. 众所周知，自开展软件工程和程序设计方法学[2]研究以来，人们总结了许多好的原则方法. 为较好体现这些原则和方法，ADL 还应满足如下一些要求：

· 形式化的软件规范

ADL 不仅可以象通常 OO 语言一样描述对象的语法界面，而且还可以使用逻辑公式来描述对象的语义，使得 ADL 可以较好地支持软件的形式化开发.

· 两级抽象

ADL 语言应能较好地体现对象规范和对象实现相分离的原则. 对象规范在抽象数据域上描述对象，提供了实现对象、理解对象和形式推理的基础. 而对象实现用某种程序设计语言按照 ADL 的约定来具体实现规范规定的对象. 一个规范可以对应多个不同的实现，即对应不同语言和算法的实现.

2　ADL 语言

为支持对象规范和对象实现相分离的原则，ADL 将一个对象的定义分为两个部分，即抽象数据定义部分(**ClassSpec**)和实现部分(**ClassBody**).

(1) 对象的抽象描述

对象的抽象描述就是在抽象域上定义对象类的属性和操作，并提供形式化语言来描述对象的语义，即对象的静态结构和动态行为特征.

⟨data abstract⟩::=

ClassSpec ⟨class—identifier⟩[(⟨class—parameter—list⟩)]	
⟨description clause⟩	对象语义的非形式化描述
⟨category—clause⟩	对象范畴
⟨inherit—clause⟩	父类表
⟨assume—clause⟩	引用类表
⟨observe—clause⟩	对象“观察”表(包括属性定义)
⟨operate—clause⟩	操作表

〈ensure—clause〉　　操作的语义规定

〈constraint—clause〉　　对象状态约束

End 〈class—identifier〉

一个对象类的抽象定义以关键字 **ClassSpec** 开始,最多可以包括 8 个部分,前面几项一般 OO 语言均提供,下面我们仅简要介绍 ADL 的特色:对象操作的语义规范和对象状态的约束限制.

· 操作语义描述

ADL 用操作的前后断言来描述操作的语义,前断言(pre—predicate)规定了操作执行前输入参数和对象状态应满足的条件,而后断言(post—predicate)则给出了操作执行结束后一定满足的条件. ADL 中,前、后断言用多类一阶谓词逻辑公式表示,如栈类 Stack(T)的操作 push 的语义描述如下:

```
for st:Stack;t:T;
st. (Push t) Satisfies
  pre st. NumOfElement<st. Vals. Length;
  post @st. NumOfElement=st. NumOfElement+1 and
       @st. Vals. (Entry@st. NumOfElement)=t;
```

其中@var 表示变量 var 在操作执行后的值.

· 对象状态约束描述

对象状态约束用来规定对象的合法状态集合,它包括静态和动态两类约束. 对象静态约束表示对象在任何时刻均应满足的约束条件,用一阶谓词逻辑公式表示. 而对象的动态约束则规定了对象在不同时刻的状态之间的约束关系,用时序逻辑公式来表示.

例如栈类 Stack(T)的状态约束为:

NumOfElement>=0 **and** NumOfElement<=Vals. Length.

有关状态约束推理参见文献[3],SOP 环境提供有模型一致性检查工具以检查约束一致性,目前已实现静态约束一致性检查.

(2)对象的具体实现

对象的具体实现是根据对象规范选择某种程序设计语言来实现对象的功能,对象实现是真正运行在机器上的实体,为此对象实现由 **ClassBody** 引导,包括下面 4 个部分:

· 对象版本信息描述,给出了对象实现的版本号.

· 对象运行的上下文语境(context)

由于 SOP 模型支持将多种语言实现的对象组装在一起,各种语言的执行机制不同,因此必须提供对象运行的语境给组装过程,以正确地连接和运行不同的模块. 执行语境包括基语言、运行库、编译器和解释器等项内容.

· 对象的语言实现

对象类的具体实现是使用某种高级语言来完成的. 由于通常的高级语言如 C、Pascal、Fortran、4GL 等都不具备访问对象的机制,所以要为这些语言扩充相应的使用 ADL 类和访问对象的机制. 如 C 的扩充 C+是在 C 的基础上添加了对象变量定义,例如

object Stack(T) *p, stack1;

和对象访问机制(属性和操作),例如

```
p→push(item); item = stack1.pop() stack1.NumOfElement;
```

这里指针变量 p 指向 Stack 对象,变量 stack1 表示一个 Stack 对象.

实现包括结构定义和操作实现两个部分.结构定义由 instancedef 引导,例如类 Stack(T)的属性定义为:

```
instancedef {
  int Number;
  object Array(T) Values;
};
```

操作实现则是按照 C 函数的形式来定义的,只是在函数定义时允许对象变量和对象表达式出现,例如

```
void push(object T i) {……}.
```

实现语言一般只对基语言作很少扩充,不会增加用户的负担.

·抽象数据到具体数据的映射

由于对象的语义描述是在抽象域上进行的;是理解对象、引用对象、对对象行为进行约束推理以及验证对象性质的基础,而对象的执行是在具体域上进行的,因此必须提供某种机制来建立两者之间的联系,抽象数据到具体数据的映射就是为了这个目的而设置的.映射提供了抽象域上的属性和操作名与实现域上的属性和操作名之间的对应关系,如 Stack(T)的实现中有:

```
Transform
  NumOfEleent => Number;
  Vals -> Values;
```

抽象到规范的一致性由程序开发者自行保证,SOP 环境拟提供动态行为一致性检查工具.

(3)一个完整的例子

下面给出一个用 ADL 定义的含参类 Stack(ITEM),从中可以看出 ADL 的一些特点.它引用了数组类 Array(ITEM),具体实现采用了 C+语言.

```
ClassSpec Stack(ITEM)
  Assume Array(ITEM);
  Obervation
    NumOfElement: Integer;
    Vals: Array(ITEM);
  Operation
    (Init Integer);
    (Push ITEM);
    (Pop ITEM);
  Ensure
    for st:Stack;t:ITEM;len:Integer;
    st.(Init len) Stisfies
      pre len>0;
```

```
        post @st. Vals. Upper=len and @st. Vals. Lower=1 and
            @st. NumOfElement=0;
      st. (Push t) Satisfies
        pre st. NumOfElement<st. Vals. Length;
        post @st. NumOfElement=st. NumOfElement+1 and
            @st. Vals. (Entry @st. NumOfElement)=t;
      st. (Pop) return t satisfies
        pre st. NumOfElement>0;
        post t=st. Vals(Entry st. NumOfElement)andd
            @st. NumOfElement=st. NumOfElement-1;
    Constraint
        NumOfElement>=0 and NumOfElement<=Vals. Length
End Stack;
Classbody Stack(ITEM)
    Version "C version 1. 0";
    Execform
      Source C;
      Compiler: cc;
    Implementation
      Instancedef {
        int Number;
        object Array(ITEM)Values;};
      void Init(int length) {
          selfiNumber = 0;
          selfValues = Array(ITEM). Creat(1,length); }
      object ITEM Pop()
          { return self. Values. Entry(self. Number--); }
      void Push(object ITEM t)
          { self. Values. Store(++self. Number,t); }
      Transform
          NumOfElement => Number;
          Vals => Values;
End Stack;
```

3　结束语

我们已经在 AT&T UNIX SVR4.0 上实现了 ADL 语言的编译器(对象的实现语言可以是 C+或 ESQL/C+)和对象行为约束一致性检查器,并且用 SOP 方法(ADL 语言和 PCL 语言)开发了两个应用实例:窗口界面生成和学校健康信息管理系统 HMS,结果令人满意.

下面我们总结一下 ADL 的特色:

(1) 对象规范和对象实现分离

ADL 在两个不同范畴描述对象,与软件设计的规范与实现分离原则相一致,体现了不同层次的抽象,从高层的对象行为规范到对象由相应语言的具体实现. 对象规范在抽象数据

域上对对象的特征和行为进行描述，对象实现是对规范的具体实现，由于规范部分并不规定具体的实现语言，因此可由实现者按实现要求和自己的爱好选择语言实现对象. ADL 通过映射将抽象数据和具体数据联系起来. 与国际上著名的 OO 语言相比，ADL 的这一特征十分突出.

(2) 提供了形式化的语义描述语言

ADL 分别采用一阶谓词逻辑和时序逻辑作为描述对象静态特征和动态特征的数学工具，对象的语义可以有严格的形式化规范，为形式化开发和使用对象提供了基础. 尽管 SOP 提倡形式化的软件开发方法，但我们出于使用方面的考虑，并不要求用户提供的对象语义必须是完备的，但要求必须是一致的.

Eiffel[4]语言也提供了类似的例程的前置条件和后置条件、类不变式，但是这些断言语言仅具有完全谓词演算的部分能力，而且 Eiffel 也未将对象规范和实现部分分开.

(3) ADL 提供了集成多种语言模块的框架

我们并未将对象的实现语言局限于某种特定语言，而希望能够给开发者提供一个选择合适实现语言的机会，他们可按自己的意愿选择 C+、Pascal 或某种 4GL+来实现对象.

(4) ADL 提供了持久对象

ADL 采用对象标识符的概念，在系统一级支持持久对象(persistent object)，临时对象和持久对象的处理统一对待，区别对用户透明. 从这个意义上讲，也可称为 OO 数据库数据定义语言. 目前广泛使用的 C+和 Eiffel 语言均不在系统一级支持持久对象，因而它们只提供了复用类的可能性，对象通常是不可复用的.

下表给出了 ADL 和常见的 OO 语言的比较.

特 征	C++[5]	SmallTalk[6]	CLOS	Eiffel	ADL
强类型	Y	N	N	Y	Y
类是对象	N	Y	Y	N	Y
封装	Y	Y	N	Y	Y
多继承	Y	N	Y	Y	Y
含参类	Y	N	N	Y	Y
对象标识符	N	N	—	N	Y
动态联编	Y	Y	Y	Y	Y
持久对象	N	N	N	N	Y
断言和约束	N	N	N	Y	Y
规范实现分离	N	N	N	N	Y
对象实现语言	1	1	1	1	many

注：Y 表示有此特征，N 表示没有，— 表示不详.

参考文献

1 Li Jing, Feng Yulin. Construct software compositionally. Proc. of Urumqi Inter. CASE Symposium'92, 1992.

2 冯玉琳,仲萃豪,陈友君. 程序设计方法学,第2版. 北京:北京科学技术出版社,1989.

3 冯玉琳,李京,黄涛. 对象语义理论和行为约束推理. 计算机学报,1993,16(11):823－828.

4 Meyer B. Object oriented software construction. Prentice Hall, 1988.

5 Stroustrup B. The C++ programming language. Addison－Wesley, 1986.

6 Goldberg A, Robson D. Smalltalk 80: the language and its implementation. Addison－Wesley, Reading, 1983.

OBJECT SPECIFICATION AND IMPLEMENTATION IN SOFTWARE CONSTRUCTION

Li Jing

(*Department of Computer Science, University of Science and Technology of China, Hefei* 230027)

Feng Yulin　Huang Tao

(*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080)

Abstract　This paper deals with ADL, a language designed to describe the features of objects in the software construction. An ADL class definition includes two parts: one is the object specification which specifies the object interface and semantics by logic formulae on the abstract domain; the other is the object body which is the implementation of the specification, and it also provides a map between the abstract data and the concrete data.

Key words　Software construction, object specification, object implementation.