第4卷第2期
1993年4月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 4, No. 2
Apr., 1993

# 模型开发方法及其原型工具系统 MTOOL

丁茂顺　易章雄　　冯玉琳

(中国科学院软件研究所，北京 100080)　(中国科学技术大学，合肥 230026)

# MODELLING DEVELOPMENT METHOD AND ITS PROTOTYPING TOOL SYSTEM MTOOL

Ding Maoshun and Yi Zhangxiong

(*Institute of Software, Academia Sinica, Beijing* 100080)

Feng Yulin

(*University of Science and Technology of China, Hefei* 230026)

**Abstract** This paper sums up a method for modelling development and puts forward a tool MTOOL to support the method, according to the authors' researching practices in the fields of software development methodologies and tools during recent years. It induces the developing procedure and depicting means of the modelling method, introduces MTOOL's components and working procedure in detail, and gives the main grammar contents of a model specification language (MSL). The designing ideas and implementing techniques concerned about MTOOL are also discussed.

**摘要** 本文根据作者近年来研讨软件开发方法的工作和开发软件工具的经验，总结了模型开发方法，提出了模型方法的支持工具 MTOOL. 文章归纳了模型方法的开发过程和描述手段，详细介绍了 MTOOL 的组成和工作过程，给出了模型描述语言 MSL 的主要文法内容，讨论了 MTOOL 的设计思想和有关实现技术.

## §1. 引　言

模型开发方法越来越受到软件开发者的普遍重视，这是因为从一个可见模型出发的软件开发要比从需求描述开始容易，同时也因为基于模型的软件系统的正确性、稳固性、可理解性

本文 1991 年 1 月 5 日收到，1991 年 3 月 9 日定稿.作者**丁茂顺**，副研究员，主要研究领域为系统软件、软件工程工具环境及交互系统构造方法. **易章雄**，工程师，1989 年硕士毕业于中科院软件所，主要研究领域为软件工程技术及 JSD 方法. **冯玉琳**，教授，现在中科院软件所工作，任副所长，主要研究领域为程序变换、形式语言和软件工程方法.

和可维护性得到了改善.

模型方法认为,系统输入数据流是系统所关心的客观世界动态行为的一种信息表达.由此,模型方法根据用户基于其专业知识对客观世界行为的一种自然划分,将系统输入数据流分离为彼此独立的数据流;其中每一独立的数据流对应客观世界中的一个实体,是该实体动态行为的一种信息表达.

模型方法将软件系统输入数据流的内含——客观世界及其动态行为——称为软件系统的主题;软件系统的功能则根据系统的输入数据产生系统的输出数据.模型方法用信息处理顺序进程模拟系统主题中客观实体的行为,建立系统主题的计算机仿真模型(简称"模型"),为产生系统输出数据的软件系统中的功能进程提供所需的输入数据.模型进程、功能进程和它们之间数据通信的说明便构成了完整、详尽的软件系统说明,模型方法以此为依据形成最终形式的、可在目标计算机环境中有效、方便运行的软件系统.

传统的应用软件开发方法在软件系统说明中只包含软件系统做什么,即从什么样的输入数据产生什么样的输出数据的说明,而不包含软件系统怎么做,即系统的工作机理的逻辑说明.由于这样形成的软件系统说明不能构成一个完整的、可工作的系统,因此造成了对软件系统理解、移植和实现的困难.模型方法由于明确划分了软件系统说明同软件系统实现的界限而克服了以上缺陷.

本文下面将总结模型方法的开发过程和描述手段,详细介绍模型方法支持工具 MTOOL 的组成和工作过程,给出模型描述语言 MSL 的主要文法内容,讨论 MTOOL 的设计思想和有关实现技术.希望这些工作能为我们以后研究软件开发方法和支持工具提供帮助.

## § 2. 模型开发方法

### 2.1 模型开发过程

采用模型方法开发软件主要有三个阶段:

1. 模型化阶段

列出系统所关心的客观世界的实体和其执行或承受的动作,定义系统主题的范围,描述每一客观实体所执行或承受的动作在时间上的先后次序,将形成的系统主题描述中的每一客观实体及其动作序列模型化为顺序进程,并描述进程的算法化功能.

2. 网络阶段

描述模型与客观世界间的数据通信及模型中进程间的通信,根据开发的需要增加模型进程的内容,以便系统能产生所需的输出数据.

3. 实现阶段

考虑目标计算机环境的各种限制,如计算机的有限存储空间及计算机可提供的有限处理时间等,运用各种实现技术,如进程调度技术、进程到模块过程的变换技术、程序正文同运行状态的分离技术等,将软件系统说明转换为可在目标计算机中有效运行的软件系统.

这里实际上做了两部分工作:仿真并描述客观世界(包括做什么和怎样做)以代替客观世界的求解,是一种自底向上的设计过程;然后将描述客观世界的模型转换成软件系统,使软件系统的部分结构保持着客观世界的自然结构.

### 2.2 模型描述手段

1. 实体结构图

实体是指组成客观系统并参与或影响系统的机构、单位、部门、个人、或抽象信息对象的基本单位.它以一定的时间顺序完成一些动作,存在于现实世界中.属于同一类型的多个个体,可以有同一个实体名.

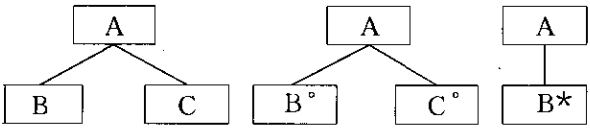


图1　控制结构

动作是与系统所要完成的目标相关的实体的行为,它完成一定的功能,如记录、统计、显示、报告、发信号等.动作发生在一个时间点,而不是在一个时间片;动作发生于现实世界中;动作不可分.动作是有时序的,可用结构化的形式来描述控制结构(顺序、条件、重复),如图1所示.动作的功能可用有限的形式化描述来说明.

2. 系统描述图

模型是由一些进程构成的可运行的系统,包括各实体活动的进程;还可能有若干功能进程,主要有三个方面:①现实世界与模型之间的数据收集和错误处理进程,收集动作的信息,只将那些合理的数据送到模型;②输出进程,从模型中抽取数据,计算输出结果;③交互活动进程,功能类似输出进程,但得出的结果不输出,而是传回到模型中去.

这些进程以通信来协调它们的运行.通信有两种基本方式:数据流和状态向量.数据流通信就是进程之间的读写消息.进程的状态向量包含了进程的局部变量.一个进程通过读其它进程的状态向量来获取其它进程的运行情况.

图2是一个简单银行业务系统(BANKSYS):来到银行的顾客(customer),最初建立帐户(invest),最后结户与银行脱离关系(terminate),在这期间他可以随便存款(pay—in)和取款(withdraw).假定不计利息,允许超支,管理员(manager)可以随时查询(enquiry)任一顾客的结余.

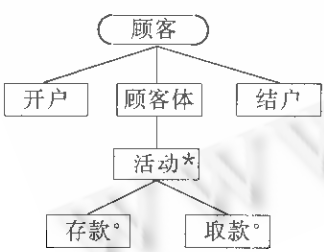


图2　简单银行系统的顾客和管理员实体结构图

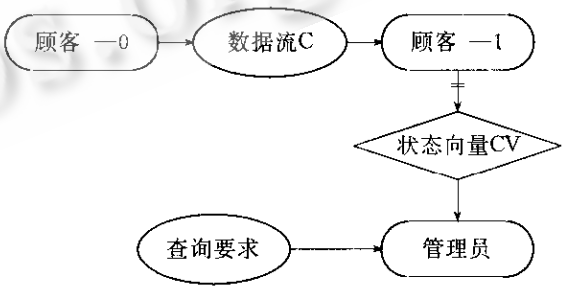


图3　简单银行系统的系统描述图

customer—0是现实世界中的顾客,通过数据流C把他们的要求送给模型进程customer—1.customer—1进程处理顾客的业务要求.银行管理员的查询要求放到enquiry input中.manager进程得到这个消息后,用状态向量CV访问customer—1的内部状态.

## §3. 原型工具系统 MTOOL 的组成和工作过程概述

### 3.1 MTOOL 的组成

为了支持模型开发方法，我们设计并实现了一个原型工具系统 MTOOL. 它为使用模型方法的软件开发者提供一整套系统化的软件工具：

① 编辑器：编辑器帮助用户交互式地构造实体结构图（ESD）和系统描述图（SSD），描述两个图中诸对象的有关正文说明，从而抽象出模型. ② 转换器：转换器把编辑的结果（图形和正文）自动转换成 MSL 程序，或者反过来，由 MSL 程序自动生成实体结构图和系统描述图. 图形直观、形象，是用户习惯的描述模型的方式，但不适合计算机处理；MSL 是用结构化正文形式描述模型的语言，易于计算机处理. ③ 解释器：解释器直接扫描实体结构图和系统描述图，模拟模型进程的行为，接收外部事件，完成动作的功能，针对用户的设计给用户以快速反馈. ④ 编译器：编译器考虑给定环境下的资源配置和系统要求，主要是处理机的数量、内存空间的限度和系统响应时间的要求，采用一些实现技术把 MSL 程序翻译成 C 语言代码.

MTOOL 的构成如图 4 所示：

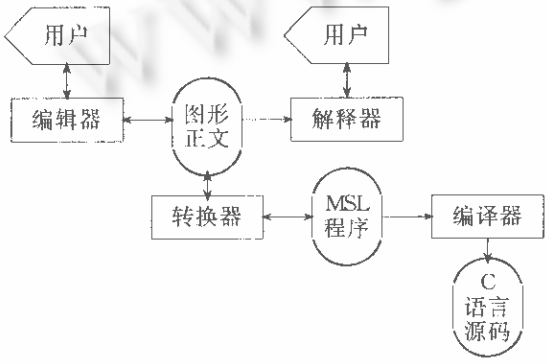


图 4　MTOOL 的构成

### 3.2 MTOOL 的工作过程

用模型方法开发软件的思想是：获取原来人工系统的方案，然后计算机化，按用户指定的工作方式（如何做）来实现在计算机上工作的软件系统. 它有两个阶段：①仿真人工系统以构成模型系统；②从模型系统映射到软件系统.

MTOOL 采用如下几个步骤的工作过程，支持模型方法，完成一个软件的开发.

第一步：抽象出模型系统.

用户通过对客观系统的理解，用实体、动作和通信等基本元素来抽取客观系统的模型. 用实体结构图描述实体所完成的动作的时序关系，用系统描述图指出这些实体之间的通信联系. 用户采用编辑器构造实体结构图和系统描述图及图中对象的正文说明，这些图和正文说明构成了模型系统的描述.

第二步：解释运行模型系统.

解释运行模型系统的首要任务是将模型系统的描述解释成计算机上的运行机制，包括：模型系统的内部状态解释成数据结构；实体解释成进程、模块；动作解释成操作序列、规则；通信解释成消息发送、过程调用等. 其次是提供描述报表、屏幕录入和样板查询机制，及多窗口和菜单等用户接口界面等. 解释运行的结果就是用户抽象模型系统的反馈，它指导用户的进一步设计工作.

第三步：生成 MSL 程序.

当用户的设计达到一定程度时，可用转换器将图形和正文形式的模型描述转换成 MSL 程序. 当然，如用户对客观系统和 MSL 语言很熟悉，他也可以直接书写模型的 MSL 程序. 反过来，用户可以从 MSL 程序得到模型的实体结构图和系统描述图，在这个基础上进行再设

计.

第四步:形成软件系统.

MSL 程序独立于具体的机器和环境,用户采用编译器将 MSL 程序翻译成指定机器和环境下的 C 代码软件系统.翻译过程中可以采用一些实现技术,如程序正文和数据分离技术,进程合并和内部缓冲区技术等.这样形成的软件系统占内存少,运行效率高.

## §4. 模型描述语言 MSL

MSL (Model Specification Language)是以结构化正文形式描述系统说明的语言,它由四部分组成:

1. 事件说明

事件定义进程的行为,所有进程中的事件都必须在此说明.说明一个事件也就是定义一个消息类型,它包括事件的属性及该事件所属的进程.

如 Event

```
        invest customer—id(date amount customer—id)
    Event END
```

表明事件 invest 含有属性 date、amount、customer--id,该事件发生在进程标识符为 customer—id 的进程内.

2. 数据流说明

一个数据流说明对应于一个数据流类.一个数据流类定义给出了它的元素的消息类型,并且说明以数据流的形式连接两类进程.数据流有一个无限的缓冲区,用于存放事件说明中定义的消息,进程说明中的 read/write 语句存取这些消息.

如 DATASTREAM

```
        C (invest pay—in withdraw terminate)
        :#)—)customer—1
    DATASTREAM END
```

定义了数据流 C;它的元素的消息类型是 invest、pay—in、withdraw、terminate;它连接着外部世界和 customer—1 进程.

3. 状态向量说明

状态向量说明定义了进程之间的状态向量连接关系.它包括状态向量的名字,要引用的内部变量和存取方法.状态向量的名字是一个消息通道,传送读取一个进程的内部状态的请求消息和回答消息.存取方法指出选取进程的条件和访问进程的顺序.

如 STATEVECTOR

```
    CV(balance): customer—1)) — ) manager
    STATEVECTOR END
```

定义了状态向量 CV.每个 customer—1 进程含有内部变量 balance;manager 进程用状态向量 CV 访问 customer—1 进程的内部状态.状态向量 CV 是多对一的连接关系,即同一个 manager 进程可以同时读取多个 customer—1 进程的内部状态.

4. 进程说明

进程说明描述进程的内部结构和具体功能,其中包括对数据流和状态向量存取和使用的方法:

读语句从数据流中得到消息,并把它放到给定的变量中.

如 sread (A,rec)表示从数据流 A 中读一个消息,放到变量"rec" 中.

写语句写一个消息到一个数据流中.

如 swrite(C 'invest' 860228 100"yzx")表示写一个消息到数据流 C,消息的类型为'invest', 消息的内容为 860228,100 和 "yzx".

取状态向量语句读取另一个进程的状态向量——执行这个语句时,发送一个消息给对方进程,要求得到它的内部状态,返回结果就是它的状态向量.

如 getsv(cv "yzx" svrec)表示读取标识符是 "yzx"的进程的状态向量,结果放在变量 svrec 中.

等待语句用于实现进程之间的同步.

如 wait(ta "yzx" tgm) 表示进程一直挂起,直到与 "yzx" 进程相连的数据流 ta 中的类型为 "tgm" 的消息全部取出.

## §5. 原型工具系统 MTOOL 的设计

MTOOL 的四个子工具既相对独立,又是一个整体系统. 该系统基于一个用户接口管理系统 UIM (User Interface Manager),提供图形窗口、事件中断管理和资源管理等功能,以库函数的形式提供给应用程序.

### 5.1 编辑器的设计

编辑器将实体、动作、进程、数据流和状态向量等作为编辑的基本单位,提供相应的基本图形部件,用以构造实体结构图和系统描述图. 同时,提供正文编辑功能,用来描述图形对象的正文说明.

编辑器有两个工作状态:实体结构图编辑和系统描述图编辑. 它们分别有各自的图元,但有共同的编辑功能.

所有的图形对象均采用共同的数据结构,包括对象的标记、类型、名称和图形位置等. 这些数据在内存中用拉链链接起来,统一管理.

### 5.2 解释器的设计

解释器包括一个扫描器和一个运行器,如图5所示.

扫描器扫描实体结构图和系统描述图,检查语法和语义错误,生成如下各表:(1)事件表,存放事件说明;(2)连接表,存放数据流说明和状态向量说明;(3)附加表,存放状态向量的存取方法;(4)进程类表,存放每个进程的内部变量,进程结构和功能.

然后用户应初始化运行环境,或者指出使用以前的运行环境. 最初,运行环境中没有消息,运行器让用户给出一个消息,如果该消息的目标进程不存在,则创建它.

运行器包括消息扫描器、进程解释器、调度中心、用户界面和一些功能程序:

1. 消息扫描器

有三类消息:数据流消息、读状态向量请求消息、读状态向量返回消息. 消息扫描器分析消

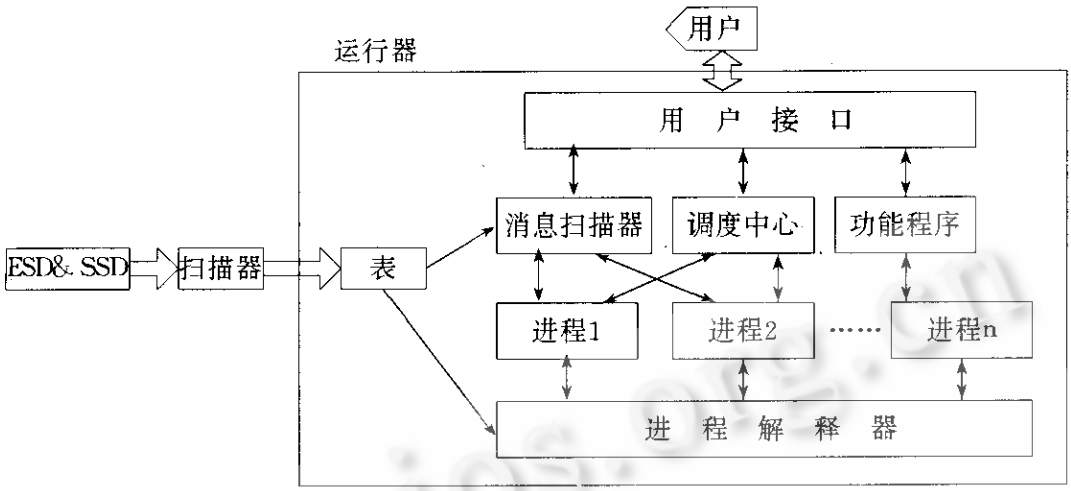


图5　解释器

息的类型,把它们送到相应的缓冲区.如果有一个数据流消息,它要送到的目标进程还没有建立,则消息扫描器创建这个进程和与它相连的相应缓冲区.

2. 进程解释器

进程一旦建立,进程解释器就执行这个进程直到遇见读语句、取状态向量语句和等待语句才挂起进程运行.当调度中心激活进程时,解释器重置进程,继续执行.

3. 调度中心

调度中心随机地查看进程,决定是否要激活进程.进程的激活条件存于缓冲区中,可以是如下三种消息之一:(1) 数据流消息;(2) 访问状态向量的返回消息;(3) 访问状态向量的请求消息.

激活情况可分为两种,一种是对上面的(1)和(2),取出缓冲区中的消息,恢复进程;另一种是对(3)请求得到进程的状态.调度中心将考虑如下特殊情况:(1) 所输入的仅仅是状态向量的进程的激活和挂起;(2) 当进程需要外部世界的状态向量时,调度中心将请求用户界面来提供.

4. 用户界面

用户界面有如下功能:(1)得到外部世界的事件,把每个事件作为一个消息送扫描器;(2)获得属于外部世界进程的状态向量的值;(3)处理调度的挂起.

5. 功能程序

功能程序的机制是:跟踪进程的状态向量和消息,跟踪数据流,显示进程的当前状态和队列的消息.

### 5.3 编译器和有关实现技术

MSL 程序独立于具体的资源配置和分配策略,为了设计在给定环境下的运行代码,从 MSL 程序到可运行代码的生成是一种实现途径,这个工作是由编译器来完成的.它采用一些实现技术,如程序正文和数据分离技术、进程合并和内部缓冲区技术等,把 MSL 程序翻译成适当的语言代码.

一般模型进程的生命期是很长的,但其真正处于活动状态的时间又很短,大部分时间都处

于等待消息的状态.出于节约处理机资源的考虑,必须有一个挂起和激活机制,使得进程处于等待状态时挂起它,当可以继续运行的条件满足时激活它.

有时同一进程类型的进程实体是大量的,当然,这些进程实体的状态有所不同,但它们的功能正文是一样的.进程的正文和数据分离,能节约处理机时间和存储空间,它让进程的若干实体对应若干份数据(即状态向量)和一个公用的子过程,输入记录和状态向量看作子过程的参数.调用子过程时,输入记录的值和状态向量一同传给子过程,子过程执行完后,返回被修改了的状态向量,调用程序负责存取和修改相应的状态向量.

进程合并,能减少处理机数目,它把读消息进程作为写消息进程调用的一个子过程.消息传递就是过程的调用,消息记录作为调用的参数.这样,就把一个由进程形成的网络系统合并成了一个由主程序 MAIN 和它的若干调用子过程所组成的层次结构.

设立内部缓冲区,可以克服系统描述图中有数据流环路无法合并问题:在环路中增加一个内部缓冲区来打破这个环路.

**结论**:模型方法揭示了客观世界问题和软件系统之间的关系,把计算机看作为人类劳动的一种替代工具.模型方法是仿真客观世界中的活动过程,并设计人工系统中所需要的各种数据信息.因此,原先人工系统能做到的,软件系统也能实现;而人工系统不能做到的,软件系统也不能实现.由此大大地提高了软件系统的稳固性、可理解性和可维护性.

我们为使用模型方法的软件开发者提供了一套软件工具.当软件工作者(用户)采用模型方法学从事软件开发工作时,这种支持主要表现在:提供基本的适用于模型方法学的图形和正文编辑功能,让用户尽可能地用图形和正文相结合的方式来表达设计思想;实现图形表示和说明语言之间的相互转换;为用户提供设计的快速反馈;为说明语言提供翻译功能.

然而,模型方法中的一些固有缺陷,严重影响了人们去尝试.主要有以下两个方面:首先,客观世界与其软件系统的概念不完整.模型方法采用实体、动作、模型进程等抽象客观世界,所有实体都对应于软件系统中的进程.这些概念不能完全反映客观世界,也难以映射到软件结构之中.其次,系统的实现过于复杂.模型方法没有考虑从模型系统映射到软件系统时的规范或算法,以及软件系统本身的结构.因此,我们在研制 MTOOL 系统的同时,也注意总结、发现新的软件开发模型、方法和技术.

## 参考文献


[1]仲萃豪、叶农,JSD 方法对软件工程研究的启示,《计算机科学》,1988.6.

[2]冯玉琳、丁茂顺,系统开发的时序模型方法,《计算机研究与发展》,1989.1.

[3]冯玉琳、桂自强、丁茂顺,系统模型开发的形式化技术,《软件学报》,1991.1.

[4]仲萃豪、丁茂顺等,应用软件的开发方法,《计算机科学》,1991.1.

[5]Boehm, B.,Improving Software Productivity, Computer,Sept. 1987.

[6]Cameron, J. R.,An Overview of JSD, IEEE Trans. on SE,Feb. 1986.

[7]Cameron, J.,JSP&JSD: The Jackson Approach to Software Development (2nd Edition),IEEE Computer Society, 1989.

[8]Ding Maoshun & Dai Guozhong,The Interactive Graphics Techniques, in the Software Engineering Environment Proceedings of the International Workshop on Software Engineering Environment,Aug. 1986.

[9]Jackson, M. A.,Principles of Program Design, Academic Press,1975.

[10]Jackson, M. A.,System Development, Printice Hall,1983.

[11]Junzo KATO, Direct Execution of a JSD Specification, The 11th COMPSAC.

[12]Ramamoorthy, C. V.,Software Engineering:Problems and Perspectives, IEEE Computer, Oct. 1984.