计算机科学1997Vol. 24№. 1

在构件库支持下用过程控制生成应用软件系统

顾 明　　仲萃豪

（深圳高等职业技术学院计算机系　深圳 518055）（中国科学院软件研究所　北京 100080）

摘 要 本文提出了一个由过程控制生成应用软件系统的理论模型，并描述了与它相关的定义，为了使该模型能实际应用到应用软件的开发之中，给出了基于此模型的过程控制语言。

关键词 构件库，过程控制，应用软件系统，生成模型。

1 引言

在应用软件的开发中，有各种形式的构件，通过这些构件的组装可以生成应用软件系统，我们提出的过程控制模型是指在已开发好应用软件的各个组成构件后，如何通过组装控制来生成应用软件系统。本文描述了由事件驱动的过程控制模型和支持该模型的过程控制语言。

2 由事件驱动的过程控制模型

2.1 扩充的下推自动机模型

由事件驱动的过程控制模型是一个经过扩充的下推自动机（图1），它的形式化描述是八元组：EPDA＝(Q，Σ，Γ，I，Z，Q_0，Z_0，F)。其中--Q：过程状态的集合；$Q_0 \in Q$：是过程初态；Σ：构件标识的集合；Γ：事件标识的集合；$Z_0 \in \Gamma$：第一个事件；$F \in Q$：过程终态；I：信息指针标识的集合；$Z: Q \times \Sigma^* \times \Gamma \times C \rightarrow Q \times (\Sigma \cup C) \times \Gamma^*$。说明：符号 Σ^* 表示字母表 Σ 上所有的字符串的集合。

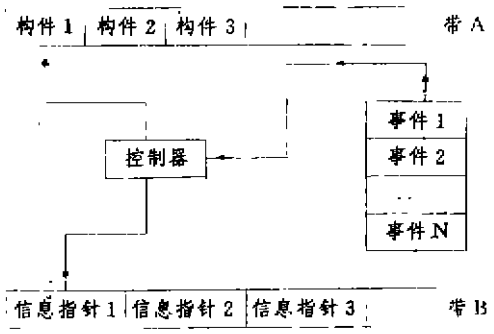

图1 扩充的下推自动机

对下推自动机的扩充之处是以下几点：

(1)八元组中的 I 是对下推自动机的扩充元组，反映在图1中就是带 B，扩充的下推自动机由栈中的事件驱动，通过控制器来读取带 A 或带 B 的内容，在某一时刻是读带 A 还是读带 B，取决于事件的类型。

(2)因为扩充了 I 元组，状态映射也就与下推自动机不同了。

(3)对控制器的读头，在一个事件的驱动下，在带 A 上一次可以读多个符号，即读多个构件；但在带 B 上，一次只可以读一个符号，即读一个信息指针，因为，在过程控制生成软件系统时，组装集成的基本上是多个构件，而其它的组装信息一次事件只有一个，如指定包含(Include)文件，说明查找路径等。

(4)带 B 上用信息指针是因为组装集成时的信息可能是写在一个系统文件中，或数据库表格中等，对这些信息的存取将通过指针来进行。

(5)EPDA 的过程终态只有一个，即最后生成所需的软件系统时的状态。

2.2 相关的定义

定义1 过程状态(Process States)是反映下推自动机控制情况的记录，每一个事件的驱动使自动机处于一个状态，这些状态称为过程状态。当第一个事件还未发生时自动机的状态为过程初态，最后生成软件系统时的状态是过程终态。

八元组中的 Q 是由所有过程状态组成的集合，Q_0是过程初态，F 是过程终态。

定义2 过程状态的改变(Process State Change)称为过程状态的变迁。

定义3 事件(Event)是某一瞬间发生的动作，它导致过程状态的变迁(Q_1, Q_2)，Q_1和 Q_2分别是动作前后的过程状态。

定义4 事件的类型(Event Type)分为两种，直

接涉及到构件的称为类型A，不直接涉及到构件的称为类型B，类型A使自动机的控制器读带A，类型B使自动机的控制器读带B。

例如，与从构件库中查询和选取构件有关的事件是类型A，在系统组装集成时，与指定查找路径和系统初始化有关的事件是类型B。

定义5(映射) 假设 q, q_a, q_b 是状态，$a_i(1<=i<=n)$ 在 Σ 中，b 在 I 中，T 是栈顶事件，$T_a, T_b \in \Gamma$，那么映射 $Z(q,a,b,T)=((q_a,T_a),(q_b,T_b))$ 的解释是：在带A符号是a，带B符号是b，栈顶符号是T的情况下，处于过程状态q的EPDA能够进入过程状态 q_a 或 q_b，且用符号串 T_a 或 T_b 替换符号T，若进入过程状态 q_a，则控制器的读头在带A上前进若干个符号；若进入过程状态 q_b，则控制器的读头在带B上前进一个符号。

定义6 瞬时描述ID(Instant Description)形式化地描述了EPAD在一个给定瞬间的格局，它反映了过程状态的变迁，ID必须记录下过程事件栈和带A及带B的内容，定义ID是一个四元组：ID=(Q, A,B,T)。其中，Q是过程状态，A是带A已读过的最后一个符号，B是带B已读过的最后一个符号，T是事件栈顶当前的符号。

EPDA是这样工作的：栈中的事件是驱动自动机工作的动力，由栈顶的事件驱动控制器，根据事件的类型来决定是读带A或带B，读完之后，自动机进入一个新的过程状态。

我们可以把生成软件系统所需要的事件依次放入栈中，涉及到的构件放入带A，其它的信息放入文件或数据库的表中，并把这些信息的指针放入带B，这样，在理论上，自动机就能在事件的驱动下，自动完成过程控制，生成用户所需的软件系统。

以上是过程控制生成软件系统的一个理论模型。这是一个理想的情况，要想真正实现控制过程的自动化，不仅需要很多的工具和环境来支持，而且还有许多的关键点需要突破，很多情况下，自动化问题都是卡在了几个关键点上。

从目前软件研究发展的情况来看，实现过程控制的自动化还有一定的难度，在实际应用中，有很多动作不能自动完成，还需要人工来做，我们的解决办法是，提供过程控制语言，开发相应的软件支撑平台，实现计算机辅助的过程控制，生长软件系统，关于软件支撑平台，我们将另文介绍，下面将介绍过程控制语言。

3 过程控制语言

3.1 过程控制语言的基本描述

过程控制语言就是以语言的方式控制以上的动作，以实现逻辑系统到软件系统的转换，它有二级结构，一级是基本描述，另一级是构造控制，本节先说明基本描述，构件控制在下一节说明，具体地讲，基本描述是一个四元组：G=(I,S,R,P)。其中I：标识；S：查询选择；R：组装关系；P：界面包装。

(1)标识软件系统的名称和软件系统的版本

I::=〈System_name〉软件系统名称〈Version_num〉软件系统版本号

〈System_name〉::=〈Identifier〉

〈Version_num〉::=Ver(〈Integer〉('.'))*

〈Indentifier〉::=Letter(letter|Digital|'-')*

标识是由字母和数字及"-"符号组成的字母数字串。

(2)从构件库中查询选择组装集成的构件

S::=〈Select〉指定查询选择|〈Browse〉浏览查询

〈Select〉::=
Select〈X1〉目标子句 From〈X2〉范围子句 Where〈X3〉限定子句 Into〈X4〉存放子句

〈X1〉::=〈Comp_name〉

〈Comp_name〉::=(〈Identifier〉,)*

〈X2〉::=〈Base_name〉〈Comp_class_name〉

〈Base_name〉::=〈Identifier〉

〈Comp_class_name〉::=〈Inentifier〉

〈X3〉::=(〈String1 AND String2〉)|〈String1 OR String2〉|〈String1 NOT String2〉)*

〈String1〉::=〈Identifier〉

〈String2〉::=〈Identifier〉

〈X4〉::=〈Pointer〉

〈Pointer〉::=〈Inentifier〉

〈Browse〉::=〈Base_name〉|〈Comp_class_name〉

查询选择有二种形式：指定查询选择Select和浏览查询Browse。指定查询选择由四部分X1，X2，X3和X4组成，X1称为目标子句，说明要查询选择的目标构件；X2是范围子句，说明构件所属的库名和构件类名；X3是限定子句，说明查询选择时的限制条件；X4是存放子句，说明从构件库中查询选择出的构件在内存的存放位置，该位置由指针指出。其中

Select，From，where，Into，AND，OR，NOT 都是关键字，AND，OR，NOT 是程序设计概念中的“与”，“或”，“非”操作；在浏览查询中，或者按构件库浏览，或者按构件类浏览，这种方式的主要目的是浏览，并不选择具体的构件。

(3)定义组装关系

R::=〈Path_name〉路径名〈Include_name〉包含的系统库、包等的名字〈Parameter_name〉命令行的参数名〈Form_name〉构件通讯时的 Form 名〈Env_val〉环境变量

〈Path_name〉::=(〈Identifier〉,)*

〈Include_name〉::=(〈Identifier〉,)*

〈Parameter〉::=(〈Identifier〉,)*

〈Form_name〉::=(〈Identifier〉,)*

〈Env_val〉::=(〈Identifier〉'='(〈Identifier〉|Integer〉),)*

路径名指构件库，包含的系统文件或库，与系统初始化有关的文件或数据库表等在系统中的存放路径；

在组装集成时，有一些系统的库或软件包等需一起组装，此处给出它们的名称；

在组装软件系统时，有一个主控程序，该程序可以有参数，如 C 语言中的 main 函数可以带参数一样，此处给出实参的值；

要组装成一个软件系统，构件之间除了构件类语法中的引用关系外，构件之间还可能有联系信息，例如，二个构件要使用同一张数据库的表，这是因为目前的 MIS 是要依赖关系数据库的支持，尽管在构件描述时，我们强调信息隐蔽，即构件的数据是封装的，只能通过操作界面来引用构件，但关系数据库又使得数据和操作分离，无法封装在一起，在关系数据库之上，我们开发了构件库，减少了构件之间直接通过关系数据库 FORM 的联系，但在组装软件系统时，由于要联结系统库和已有的软件包，构件之间直接通过关系数据库 FORM 的联系也是不可避免的，所以，从开发的实际要求出发，我们通过 FORM 来建立构件之间在组装时的通讯关系。

对于一个可运行的软件系统来讲，需要一些与整个系统有关的环境变量，在组装时应该给出环境变量的设置。

(4)用户界面包装

P::=〈Menu_structure〉〈form_describe〉

(4.1)设计菜单结构

〈Menu_structure〉::=

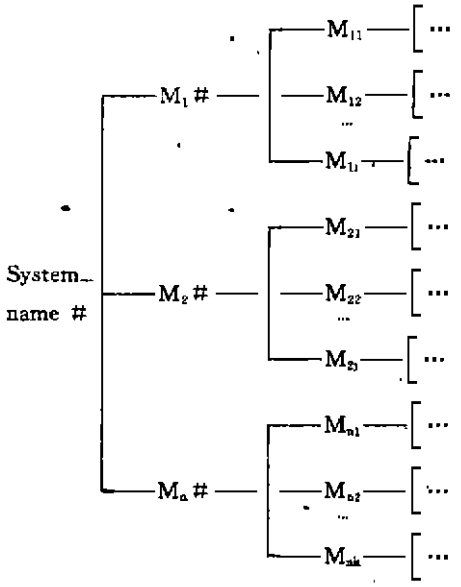

图2 菜单定义的层次结构

(#)::=X|P|S

〈Mt·〉::=String(t' 表示 M 的下标可为任意长度的正整数串)

〈String〉::=(〈letter|Digital〉,)*

Mi(1<=i<=n)是第一层出现的菜单项的名称，Mn(1<=i<=n，1<=t<=max(i,j,k,))是第一层第 t 个菜单项的子菜单，通过以上的图2可以定义出菜单的层次结构，符号“#”可取三个值 X，P 和 S，X 表示该菜单的子菜单是下拉式的子菜单，P 表示该菜单的子菜单是 POP Menu，即可随机出现在屏幕的任何位置，S 表示该菜单的子菜单是换屏显示方式，如 Informix 等的菜单一样，Mt· 可以表示图2中的任何一个菜单项，对 Mt· 可以有文字说明来描述该菜单的功能，这个文字说明出现在屏幕的提示行。

(4.2)描述 FORM

〈form_describe〉::=

〈Form_name〉表格的名称〈Format_explain〉格式说明〈Attribute_explain〉属性说明〈Form_relation_explain〉表格关系说明

一个 FORM 描述由四部分组成，表格名称，格式说明，属性说明和表格关系说明。

〈Form_name〉::=〈Identifier〉

〈Format_explain〉::=

〈Screen_format〉屏幕显示格式〈Print_format〉打印格式

〈Screen_format〉::=〈Form_bead〉表头〈Form_tail〉表尾〈Fields〉表中字段〈Form_body〉表体

〈Form_head〉::=string

〈Form_tail〉::=string

〈Fields〉::=(〈Identifier〉,)*

〈Form_body〉::=〈Line integer〉〈Row_integer〉表体的行值和列值

〈Print_format〉::=〈Form_head〉〈Form_tail〉〈Fields〉〈Form_body〉

格式说明由二部分组成：屏幕显示格式和打印格式，这二部分从格式上都由四部分组成，即表头，表尾，字段和表体，其中表头和表尾可以用字符串来描述，字段可以用标识符来描述，表体用行整数和列整数来描述。虽然该二部分格式上的描述结构相同，但属性上是不同的。

〈Attribute_explain〉::=〈Screen_attr〉屏幕录入属性〈Print_attr〉打印属性〈Query_attr〉查询属性〈Store_attr〉外存属性〈Memory_attr〉内存属性

〈Screen_attr〉::=〈Object_s〉〈Color〉

〈Object_s〉::=〈Screen_format〉

〈Color〉::=〈高亮度〉|〈红〉〈绿〉〈兰〉…

〈Print_attr〉::=〈Object_p〉〈Word_des〉〈Word_size〉

〈Object_p〉::=〈Print_format〉

〈Word_des〉::=〈宋体〉|〈仿宋体〉|〈楷体〉|〈黑体〉…

〈Word_size〉::=〈Integer〉

〈Query_attr〉::=〈Keyword〉〈Sort〉〈Condition〉

〈Keyword〉查询的关键字::=〈Identifier〉

〈Sort〉按递增或递减排序::=〈Increase〉|〈Decrease〉

〈Condition〉查询条件::=((〈String1 AND String2〉)|
(〈String1 OR String2〉)|
(〈String1 NOT String2〉))

〈Store_attr〉::=〈Fields〉〈Field_des〉

〈Field_des〉::=Character|Intéger|Date|Float|Money…

〈Memory_attr〉::=(〈Record〉,)*

属性说明由五部分组成：屏幕录入属性，打印属性，查询属性，外存属性和内存属性。其中屏幕录入属性由说明对象和颜色组成，说明对象是格式说明中的某一部分，颜色是EGA和VGA彩显可以支持的颜色之一；

打印属性由说明对象，字体和字的大小组成，说明对象也是格式说明中的某一部分，字体和字的大小是一般打印机可以支持的范围之内；

查询属性由查询关键字，排序类型和查询条件组成，查询关键字是某一字段，排序类型分递增和递减二种，查询条件由字符串和关系运算符AND，OR和NOT组成。

外存属性是FORM在数据库中的存放属性，由字段和字段类型组成，字段类型是关系数据库中支持的常用类型。

内存属性指FORM从外存读入内存后的存取形式，这里设计成结构的形式。〈Form_relation_explain〉::=〈Form_name〉表格名

〈Form_name〉::=(〈Identifier〉,)*

表格关系说明是描述该FORM同其它哪些FORM有关系，例如有些表格要从其它表格中取数据，有些表格是通过其它表格产生的等，在表格关系说明中只需指出有关系的表格名称。

3.2 过程控制语言的构造控制

以上的描述是过程控制语言的基本部分，像Shell语言一样，有一批基本的操作，它们可以用C或其它的编程语言来编写，另一批操作是通过基本操作构造出来的，称为脚本，在这点上，过程控制语言同Shell语言类似，先有一些基本的描述，为了控制这些基本描述之间的关系，我们设计了一些构造算子，通过这些构造算子，来协调基本描述之间的关系。下面通过一些定义来说明过程控制语言的构造控制。

已知传统的程序设计语言只要具备顺序、选择和循环这三种控制结构就是具有计算完备性的程序设计语言，因此，我们的构造算子就应该包括这三种控制结构。

定义1 构造算子是控制过程控制语言基本描述之间关系的运算符，由关键字和逻辑表达式或算术表达式组成。

以下定义2——定义5所定义的算子均是构造算子。

定义2(顺序算子 Series 和 End) 使出现在Series和End之间的基本描述，均按出现的先后次序顺序执行。

定义3(条件算子 If X_1 Then X_2 Else X_3 End) 当条件X_1满足时，执行X_2的描述，否则，执行X_3的

描述。

定义4(循环算子 While X_1 Do X_2 End) 当条件 X_1 满足时，执行 X_2 描述，当条件 X_1 不满足时，跳过 X_2 描述，执行 End 后的描述。

定义5(分情形算子)Case X_1

While Y_1

Z_1

While Y_2

Z_2

.....

While Y_n

Z_n

End

当 X_1 的值为 Y_1 时，执行 Z_1 描述；当 X_1 的值为 Y_2 时，执行 Z_2 描述，当 X_1 的值为 Y_n 时，执行 Z_n 描述。

定义6(无条件转移算子 Goto X_1) 当遇见该算子时，无条件执行 X_1 描述。

虽然 Goto 算子会带来控制结构上的不清晰，但有时少量使用，会使控制比较方便，所以，我们还是提供了该算子。

通过以上的五个算子，可以使基本描述进行适当的组合，这就为组装软件系统带来了极大的方便性。也可以使用这些算子，把基本描述组装成构件，放入构件库中，为以后重用提供支持，例如，把菜单生成组装成一个菜单管理构件，把系统初始化有关的描述组装成系统维护构件等。

结束语 本文提出的过程控制语言，已在我们开发的一个实际 MIS 项目中进行了初步试验，该项目是财政部的"八五"科研项目，其中的一个子系统已在试点单位投入运行，并又推广应用到另外两个单位。我们相信，该语言经过完善后，一定会有很好的实用价值。

参考文献

[1]K. Brockschmidt, Inside OLE2, Microsoft Press, 1994

[2]Richard M. Adler, Emerging Standards for Component Software, IEEE on Computer, March 1995

[3]Meyer B., Object-Oriented Software Construction, Prentice Hall, Englewood Cliffs, N. J., 1988

[4]L. Osterweil, Software Processes are Software Too, Proc. of the Ninth Intl. Conf. on Software Engineering, Monterey, California, April 1987

(上接第40页)

有灵活性，应能支持应用根据自己的特点在实时和同步之间选择偏重的一方或者采取折衷的方案。这方面的研究非常活跃。

结束语 通过 CSCW 系统与一般分布式系统的比较，说明了 CSCW 对分布式系统在技术和某些概念上的继承性，提出了 CSCW 系统的特征和要求，这些特征和要求反映在上述几个方面。从中我们得出结论：CSCW 是对一般分布式系统的继承和发展。

对各个部分的详细描述超出了本文的范围。另外文中没有提到对 CSCW 的网络支持结构以及 CSCW 体系结构，由于 CSCW 还是一个迅速发展的领域，通用的体系结构远未定型，但这方面的尝试非常多。

CSCW 被认为是一个多学科的研究领域，除了计算机科学，它还要吸取组织学和心理学的知识，这反映了 CSCW 是一个高度突出人的因素的系统。

主要参考文献

[1]Tom Rodden and Gordan S Blair, Distributed systems support for computer supported cooperative work. Comput. Communi., 15(8)1992

[2]杨德元(编译)，分布式数据库管理系统概论，清华大学出版社，1987

[3]Hiroshi Ishii et al., Iterative Design of Seamless Collaboration Media, CACM, 37(8)1994

[4]Sara A. Bly et al., Media Spaces: Bringing People Together in a Vidio, Audio, and Computing Enivronment, CACM, 36(1)1993

[5]Ting-Peng Liang et al., When client/Server Isn't Enough; Coodination Multiple Distributed Tasks, COMPUTER, 27(5)1994

[6]Sujata Banerjee et al., Performace analysis of the send-on-demand: A distributed datebase concurrency control protocol for high-speed networks. Comput. Commun., 17(3)1994

[7]Richard Bentley et al., Architectural Support for Cooperative Multiuser Interfaces, same to [6]