基于进程代数的 DSS 模型系统建模与实现①

孙　晶, 赵会群
(北方工业大学信息工程学院, 北京 100041)

摘要:把进程代数(process algebra, PA)引入 DSS 模型系统建模,提出一种新的 DSS 模型建模理论体系——DSS 模型代数,它可以描述模型之间激发、选择、重复、同步和并发等行为. 建立了基于 DSS 模型代数的模型系统与组件模型的对应关系,提出了基于组件技术的模型系统实现方法,并通过一个实例给出实现方法的具体应用.

关键词:决策支持系统; 模型; 进程代数; 组件技术
中图分类号:N945.12　　**文献标识码**:A　　**文章编号**:1007-9807(2003)02-0023-06

0 引　言

决策支持系统(decision support system, DSS)是模型驱动的. 由于决策领域复杂, DSS 模型系统实现有相当大的难度, 难点之一是模型的表示和操纵.

Blanning[1]在实体关系模型基础上提出一种基于虚关系(virtual relation)的模型表示方法, 该方法通过模型查询语言对关系表进行操作来完成模型的选择和集成.

Groffrion[2]提出了一种结构化构模语言 SML, 该语言通过引入基本实体、复合实体、属性实体、变量实体、函数实体和测试实体表示模型. SML 可显示模型实体间的相互依赖关系, 通过转换生成可以运行的模型.

Huh[3]、Lenard[4]、黄梯云[5]等对面向对象的模型表示法进行了较深入的研究, 对 SML 方法中模型部件与方法部件, 模型部件与数据部件的独立带来的特性不匹配加以改进, 在一定程度上完善了面向对象的模型表示方法. 面向对象的模型表示能很好地支持构模、集成、共享与重用, 所以得到广泛关注.

文献[6,7]等提出了一类专用构模语言的模型表示方法, 此类方法的最大特点是用户可以用类似程序设计语言的构模语言构造自己的模型.

本文把进程代数引入 DSS 模型系统建模与实现, 提出一种基于进程代数的 DSS 模型建模方法, 并应用到 DSS 的实际开发, 从理论和实践两个方面检验提出方法的有效性.

1 基础知识

进程代数 PA 是从 CCS(calculus of communicating systems)发展起来的一种形式化的分布式系统描述语言. 它把系统看成由独立实体组成的集合, 实体又称代理(agents), 各代理的执行进程是原子活动(atomic actions)集合[8], 它用于描述系统各代理之间的并发(concurrence)、同步(synchronization)等行为. 作为一种工具, 它被广泛地应用于计算机系统和通信系统表现行为描述和评价, 业已证明 PA 与 Petri 网等价, 且结构更优[9]. PA 定义的运算如下[8]:

1) 顺序组合(sequential composition)'·'　(a, r)·A 表示进程 A 中的活动顺序执行, a 是 A 中的活动, r 是活动执行时间.

2) 选择(selection)'+'　A + B 表示两个进程

① 收稿日期:2001-10-24; 修订日期:2002-07-26.
作者简介:孙　晶(1968—),女,硕士,讲师.

可选择其一执行.

3) 并发(concurrence)‘‖’ A‖B 表示两个进程中活动独立执行.

4) 协同或同步(cooperation/ synchronization)‘Θ’ AΘB 表示两个进程中的同类活动协同操作,H 表示同类活动集.

5) 隐藏运算(encapsulation)‘/’ A/ a 表示进程对活动 a 的屏蔽.

运算 3)是运算 4)的特例,即当同步运算中的 H 为空时为并发运算.

2 模型代数

2.1 模型与模型运算

狭义上讲,DSS 模型可以是一个数学公式,一个统计分析过程的描述等;广义上讲,DSS 模型可以是一个可以用计算机程序段求解或模拟的任何一个决策活动. DSS 模型可分为基模型和复合模型. 在 DSS 中模型的求解算法称为方法,模型是解决决策问题的模式,而方法是具体处理过程的实现. DSS 模型与方法是一对多的关系(1:N),即一模可以多法. 本文把 DSS 模型理解成一个数据单元或计算机单元,定义如下.

定义 1 DSS 模型是一个计算单元或数据单元,它由模型接口和模型实现模块组成. 模型接口是模型与外部接触点的集合,即:$\langle Port_1, Port_2, \cdots, Port_n\rangle$,而每一个接触点 $Port_i$ 是一个 8 元组$\langle$ID, Publ, Exte, Priv, Beha, Msgs, Cons, Non-Func$\rangle$,其中:

ID 是模型的标识;

$Publ_i$ 是模型第 i 个接触点能提供给环境或其它模型的功能集合;

$Msgs_i$ 是模型第 i 个接触点产生消息的集合;

$Cons_i$ 是对模型第 i 个接触点的行为约束,通常包括模型运行的初始条件、前置条件和后置条件,有时为了明确表示这 3 个条件可把它写成 Cons(init, pre-cond, post-cond),init、pre-cond 和 post-cond 分别表示初始条件、前置条件和后置条件的集合;

$Exte_i$ 是模型第 i 个接触点运行所需环境或其它模型的功能集合;

$Priv_i$ 是模型第 i 个接触点私有属性的集合;

$Beha_i$ 是模型第 i 个接触点行为语义描述;

$Non\text{-}Func_i$ 是模型第 i 个接触点非功能说明,包括模型的安全性、可靠性说明等.

在定义 1 中模型接口给出了模型的形式化描述,它定义了模型的模式. 为了方便,一般用元组元素(ID)表示模型 ID 中的一类元素的集合. 如 Publ(C)表示模型 C 中各接口点 Port 中 Publ 元素的集合.

定义 2 设 A 和 B 是论域 U 中的两个模型,如果 A 和 B 满足下列条件,则称 B 是 A 的一个进化,记为 Evolve(B, A).

1) Dom(B) = Dom(A)

Dom( )表示模型的定义域

2) $Publ(B) \supseteq Publ(A)$

3) $Extn(B) \subseteq Extn(A)$

4) $Priv(B) \subseteq Priv(A)$

5) $Beha(B) \Rightarrow Beha(A)$

6) Msgs(B) = Msgs(A)

7) $(Cons(B) \Rightarrow Cons(A))$

8) $(Non\text{-}Func(B) \Rightarrow Non\text{-}Func(A))$

定义 1 和定义 2 反映了模型的可重用和可进化的特性. 除此之外,模型还应具有互操作性,下面把模型间典型的操作抽象为模型运算.

定义 3 设 A、B 是论域 U 中的两个模型,若 $\exists x \in Publ(A) \wedge \exists y \in Extn(B)$ 使得$(x \Rightarrow y) \wedge (pre\text{-}cons(B))$,称模型 A 激发模型 B,记作 $A \rightarrow B$.

$A \rightarrow B$ 仍然是一个模型,它满足下列性质:

1) $Dom(A \rightarrow B) = Dom(A) \cup Dom(B)$

2) $Publ(A \rightarrow B) = Publ(A) \cup Publ(B)$

3) $Extn(A \rightarrow B) = Extn(A) \cup Extn(B)$

4) $Priv(A \rightarrow B) = Priv(A) \cup Priv(B)$

5) $Beha(A \rightarrow B) \Leftrightarrow Beha(A) \wedge Beha(B)$

6) $Msgs(A \rightarrow B) = (Msgs(A) \cup Msgs(B))$

7) $Cons(A \rightarrow B) \Leftrightarrow Cons(A) \wedge Cons(B)$

8) $Non\text{-}Func(A \rightarrow B) \Leftrightarrow Non\text{-}Func(A) \wedge Non\text{-}Func(B)$

激发运算具有中断算子和间断算子语义.

定义 4 设 A、B 是论域 U 中的两个模型,如果 A、B 满足下列条件:

1) Dom(A) = Dom(B)

2) Publ(A) = Publ(B)

3) Extn(A) = Extn(B)

4) Priv(A) = Priv(B)

5) Beha(A) ⇔Beha(B)

6) Msgs(A) = Msgs(B)

7) Cons(A) ⇔Cons(B)

8) Non-Func(A) ⇔Non-Func(B)

则称 A、B 相等,记作 A = B.

显然,运算 →满足结合律,并可以推广到有限个模型的情形.

定义 5　设 A、B 是论域 U 中任意两个组件,若 $\exists x \in Extn(A) \wedge \exists y \in Publ(B)$ 使得 $(x \Rightarrow y) \wedge (pre\text{-}cond(B))$,这时组件 A 通过"使用"(use)组件 B 中的 Publ(B) 来实现 A 的功能需求,就称组件 A、B 进行了一次"使用"运算,记作 A ⊗B.

与运算 →不同,⊗运算需要返回结果. 从实现角度,若 A、B 是计算单元,运算 ⊗需要复制 B 中的代码到 A 中或传递 B 中代码的指针到 A 中. 例如传统意义的过程调用是"使用",所以 A、B 的实现是非独立的(dependence),组件 A 的代码与 B 的代码有关.

A ⊗B 仍然是一个组件,它满足下列性质:

1) Dom(A ⊗B) = Dom(A) ∪Dom(B)

2) Publ(A ⊗B) ⊆Publ(A) ∪ Publ(B)

3) Extn(A ⊗B) ⊆Extn(A) ∪Extn(B)

4) Priv(A ⊗B) = Priv(A) ∪Priv(B)

5) Beha(A ⊗B) ⇒Beha(A) ∧Beha(B)

6) Msgs(A ⊗B) = (Msgs(A) ∪Msgs(B))

7) Cons(A ⊗B) ⇒Cons(A) ∧Cons(B)

8) Non-Func(A ⊗B) ⇒Non-Func(A) ∧ Non-Func(B)

与运算 →的情形类似,运算 ⊗也满足结合率.

定义 6　设 A、B 是论域 U 中的两个模型,若 $(pre\text{-}cond(A) \wedge pre\text{-}cond(B)) \wedge (pre\text{-}cond(A) \cap pre\text{-}cond(B)) \neq \Phi$,则称模型 B 与模型 A 同步,记作 A⊖B. M = pre-cond(A) ∩pre-cond(B),称为同步条件集.

A⊖B 仍然是一个模型,它满足下列性质:

1) Dom(A⊖B) = Dom(A) ∪Dom(B)

2) Publ(A⊖B) = Publ(A) ∪ Publ(B)

3) Extn(A⊖B) ⊆Extn(A) ∪Extn(B)

4) Priv(A⊖B) = Priv(A) ∪Priv(B)

5) Beha(A⊖B) ⇒Beha(A) ∧Beha(B)

6) Msgs(A⊖B) = (Msgs(A) ∪Msgs(B))

7) Cons(A⊖B) ⇒Cons(A) ∧Cons(B)

8) Non-Func(A⊖B) ⇒Non-Func(A) ∧ Non-Func(B).

"同步"运算也满足结合率.

定义 7　设 A、B 是论域 U 中的两个模型,若 $(pre\text{-}cond(A) \wedge pre\text{-}cond(B)) \wedge (pre\text{-}cond(A) \cap pre\text{-}cond(B)) = \Phi$,称模型 A 与模型 B 并发,记作 A ‖B.

模型的并发执行时,说明 A 与模型 B 同时执行,并且 A 与 B 独立.

A ‖B 仍然是一个模型,它满足下列性质:

1) Dom(A ‖B) = Dom(A) ∪Dom(B)

2) Publ(A ‖B) = Publ(A) ∪ Publ(B)

3) Extn(A ‖B) = Extn(A) ∪Extn(B)

4) Priv(A ‖B) = Priv(A) ∪Priv(B)

5) Beha(A ‖B) ⇒Beha(A) ∧Beha(B)

6) Msgs(A ‖B) = (Msgs(A) ∪Msgs(B))

7) Cons(A ‖B) ⇒Cons(A) ∧Cons(B)

8) Non-Func(A ‖B) ⇒Non-Func(A) ∧ Non-Func(B)

"并发"运算也满足结合率.

定义 8　设 A 是论域 U 中的一个模型,对 $\forall a_1, a_2 \in Publ(A)$,若 post-cond($a_1$) ⇒pre-cond($a_2$),即方法 a_1 执行后激发方法 a_2 执行,称模型 A 重复运算,记为 ·A.

·A 仍然是一个模型,它满足下列性质:

1) Dom(·A) = Dom(A)

2) Publ(·A) = Publ(A)

3) Extn(·A) = Extn(A)

4) Priv(·A) = Priv(A)

5) Beha(·A) ⇒Beha(A)

6) Msgs(·A) = Msgs(A)

7) Cons(·A) ⇒Cons(A)

8) Non-Func(·A) ⇒Non-Func(A)

定义 9　设 A、B 是论域 U 中的两个模型,若 pre-cond(A) ∨pre-cond(B),称模型 A 与模型 B 选

择执行,记为 A + B.

A + B 仍然是一个模型,它满足下列性质:

1) Dom(A + B) = Dom(A) ∪Dom(B)

2) Publ(A + B) = Publ(A) ∪Publ(B)

3) Extn(A + B) = Extn(A) ∪Extn(B)

4) Priv(A + B) = Priv(A) ∪Priv(B)

5) Beha(A + B) ⇒Beha(A) ∧Beha(B)

6) Msgs(A + B) = (Msgs(A) ∪Msgs(B))

7) Cons(A + B) ⇒Cons(A) ∧Cons(B)

8) Non-Func(A + B) ⇒Non-Func(A) ∧ Non-Func(B)

显然选择运算满足交换率和结合率.

2.2 复合模型和模型系统

定义 10 设任意的 M_1、M_2 为论域 U 的模型,若:

1) 模型 M_1、M_2 为决策过程模型;

2) 模型 M_1、M_2 经有限次运算后是决策过程模型.

M_1、M_2 称为基模型,运算后得到的模型称为复合模型.

定义 11 设 M 为论域 U 所有模型的集合,O 是模型操作集合,把 S = 〈M,O〉称为模型系统.

模型系统有如下性质:

1) 封闭性(envelopment) 基模型与基模型,基模型与复合模型,复合模型与复合模型运算后仍是一个模型.

2) 层次性(hierarchy) 模型系统可由模型运算构成,而一个模型系统又可以再经过运算组成新的更大的模型系统.

3) 可扩展性(expansibility) 一个满足条件的模型可以通过运算加入到模型系统中.

定理 1 设 S = 〈M,O〉是一个模型系统,则 S 是一个代数系统,称为模型代数.

3 基于组件的模型系统实现

结合一个应用实例,给出基于 Java Bean[11] 的 DSS 模型系统实现方法. 当然,也可用其它组件技术,如 COM[12] 等实现. 鉴于模型激发运算是模型系统的最基本运算,仅讨论基模型和激发运算的实现方法.

3.1 DSS 基模型

1) 模型属性 DSS 模型属性是模型的特性表示. 为此,应为其它 DSS 模型提供访问这些属性的方法. 这里沿用 Java Beans 中采用的属性访问方法和命名规则.

单值属性 get〈PropertyName〉( ); set〈PropertyName〉( ). 〈PropertyName〉为属性名.

布尔属性 is〈PropertyName〉( ) 和 set〈PropertyName〉( ).

索引属性
〈PropertyType〉[ ] get < Property Name〉( )
set〈PropertyName〉(〈PropertyType〉 [ ] values)
〈PropertyType〉 get〈PropertyName〉 (int index)
set〈PropertyName〉(int index, 〈PropertyType〉 value)

2) 模型的操作和模型接口 DSS 模型的操作提供了模型的功能和服务,这些服务是通过模型接口向外界表述的,在模型的接口中定义了该模型提供的服务,但接口并不实现这些服务,它仅仅是模型功能的描述. 这些操作可以通过定义相应的类加以实现. DSS 模型的另一个特性是一模多法,可以通过定义接口的多种版本来实现.

例如,商品保本期核算模型可以有两种计算方法:

$$D(\text{保本期天数}) = F(\text{日变动费用})/P(\text{商品毛利}) \quad (1)$$

$$D(\text{保本期天数}) = FR(\text{日变动费用率})/PR(\text{商品毛利率}) \quad (2)$$

可以定义两个接口,分别定义两种计算保本期的方法. 接口定义:

```
Public interface Date_of_profit
{public int CaluculateDate ( );// 该接口定义
    计算保本期方法}
```

按式(1)求保本期的类框架为:

```
Public class Datebyvalue implements Date_of_profit
{public int CaluculateDate( );
    {dynfee m_dynfee;// 定义接口变量
    interprofit m_profit; // 定义接口变量
    int dateofprofit ,id; // 商品批号
    id = getId( );
```

```
double profit ,dynfee ;
public double dynfee = m_ dynfee. getDyn-
Fee(id) ;
public double profit = m_ profit. getProfit(id) ;
dateofprofit = dynfee %profit ;}}.
```

按式(2)求保本期的类框架为:

```
Public DatebyRate implements Date_of_profit
{public int CaluculateDate( ) ;
{int dateofprofit ,id ; / *商品批号 */
id = getId( ) ;
double pr ,fr ;
static double fr = getDynFee(id) ;
static double pr = getProfit(id) ;
dateofprofit = fr %pr ;}}. . . }.
```

3)定义模型事件　Java 提供了丰富的事件类型,但这些事件类型主要与可视化的窗口构件或 I/O 设备有关,对具体事务处理中的事件没有涉及.所以如果事件类型不够,还可以扩充 AWT 中的事件.在构造 DSS 模型时要为每一个 DSS 模型定义事件集,DSS 模型的事件集可以建立特定的 API 或定义相应的事件类来实现.下面以保本期计算为例说明一个具体事件类的定义.在保本期计算中,如某一商品的保本期到期,应产生保本期到期的事件,并向其它模型发布该消息.为此,应定义一个保本期到期事件.

```
Import java. util. EventObject ;
Public class ExpiredEvent extends EventObject
{private int m_ expiredtype ;
public final static int EXPIRED = 0 ;
public final static int UNEXPIRED = 1 ;
public ExpiredEvent(Module module ,int ex-
piredtype) / *事件类的构造函数 */
{super(module) ;
m_ expiredtype = expiredtype ;}
public Expired getexpiredtype() / *属性的
访问函数 */
{return m_ expiredtype ;}}.
```

3.2　DSS 复合模型

根据 DSS 复合模型的定义,构成 DSS 复合模型中模型 M_1 ,M_2 , …,M_n 有两种情形:一是由纯粹基模型构造,并且它们之间不存在着激发运算;二是基模型之间存在激发运算.第一种情形可以通过组件容器把基模型组合成复合模型或通过编写程序的方法把基模型复合成复合模型.第二种情形讨论如下:

两个基模型存在激发运算意味着两个模型间有消息的传递和方法的调用.可以采用在两个模型之间建立连接器的方法解决消息的传递和方法的调用,连接器与基模型的关系如图 1.

图 1　DSS 模型调用模型

这种 DSS 模型的调用关系与组件对象模型的调用关系不同.在组件对象模型的调用中,M_i 通常为事件源,M_j 为事件接受者,并在接到事件后执行相应的操作.而在 DSS 复合模型中,M_j 为事件源,M_i 为事件的接受者,M_i 在接收到事件后执行相应的操作或模型 M_j 回调 M_i 中的方法.可以借鉴 Jave 1.1 事件模型来实现 DSS 模型激发.

1)模型 M_j 设计　在设计事件源模型 M_j 时,必须为事件的接受者 M_i 提供从事件源接收事件进行注册和注销的方法.对每一个事件类型应确定是否支持多点传输或单点传输语义.多点传输语义允许一个事件类型和任意数目接收程序相关联,而单点传输语义允许一个事件类型只注册一个接收程序.多点传输可用下述规则:

```
public void add〈ListenerType〉(〈Listener-
Type〉 list) ;
public void remove〈 ListenerType 〉(〈 Lis-
tenerType 〉 list) ;
```

单点传输可用下述规则:

```
public void add〈 ListenerType 〉(〈 Listener-
Type 〉 list)
throws java. util. TooManyListenerException ;
public void remove〈 ListenerType 〉(〈 Lis-
tenerType 〉 list) ;
```

其中,〈ListenerType〉为接受者类型.

2)连接器设计　连接器建立了事件源与事件接收者之间的联系.它允许事件产生的消息从源模型流到目标模型.连接器的本质是把目标模型实现的回调对象交给源模型;这个源模型在事件发生时向目标模型回调.连接器是通过目标模型的接口来实现的,在该接口中定义需回调的操作.如在保本期计算中一批商品的日变动费用的求解

是通过一个独立的模型实现的. 设该模型为 M_i, 计算保本期模型为 M_j, 当计算保本期时需要调用 M_i 中的 getDynFee( ). 为此, 连接器接口可定义为:

```
Public interface dynfee
{public double getDynFee(int id );
{. . . //该接口定义了计算日变动费用方法}}.
```

3) 模型 M_i 设计　模型 M_i 实现连接器中定义的方法, 如在保本期计算中事件源触发的事件是一个时间触发器事件, 时间触发器定时启动, 产生计算日变动费用的请求, M_i 接到该请求后更新计算日变动费用. 其模型框架为:

```
Public class DateDynFee implement dynfee
{public double getDynFee(int id );
. . . / * 计算日变动费用, 其中 fee1. . . fee3
分别为保管费、场地费、利息. */
    {return fee1, fee2, fee3;}}.
```

4) 子模型求解次序　复合模型设计中另一个需要解决的问题是构造复合模型中各子模型的执行次序, 这一次序反映决策过程和解决问题的步骤, 这也是 DSS 模型与一般组件模型的区别之一. 解决的方法是定义一个存放复合模型中子模型执行次序的数组, 该数组中存放子模型的模型名. 使用者可以通过在程序中设立一组存取器方法建立子模型的执行次序. 具体的作法是在建立激发运算 $M_i \rightarrow M_j$ 时, 在 M_j 回调 M_i 的方法语句后加入次序数组的设置语句.

```
Public Datebyvalue implements Date_of_profit
{. . . public int CaluculateDate( );
    {. . . private Vector m_operateoderls; // 执行次序数组
        dynfee = m_dynfee. getDynFee(id);
        addOperateOrder("getDynFee");. . . }
addOperateOrder(Orderlistener operateorderls)
{m_operateorderls. addElement(operateorderls);}. . . }.
```

上面给出了基于组件的 DSS 模型实现方法. 子模型求解次序也可以通过索引属性定制编辑器, 系统通过和用户交互与决策者共同完成决策过程.

4 结 论

与结构化构模语言等建模方法不同, 模型代数不仅考虑模型之间的调用(激发运算)、循环(重复运算)和选择等基本模型连接方式的建模, 还考虑对模型之间的同步、并发等通信特征的建模, 从而具有描述集中式系统和分布式系统建模的双重能力.

本文把进程代数引入 DSS 模型建模, 提出一种能够描述模型之间激发、选择、重复、同步和并发等行为的 DSS 建模理论体系, 该理论不但可以支持集中式(单一决策者)系统建模, 而且可以用于分布式(群体决策)系统建模. 建立了基于 DSS 模型代数的模型系统与组件模型的对应关系, 提出了基于组件技术的模型系统实现方法.

参 考 文 献:

[1]Blanning. An entity-relationship approach to model management[J]. Decision Support Systems, 1986, (2): 65—72

[2]Groffrion Arthur M. The SML language for structured modeling: Level 1 and 2[J]. Operations Research, 1992, 40(1): 38—57

[3]Huh Soon-Young. Modelbase construction with object-oriented constructs[J]. Decision Sciences, 1998, 24(2): 342—356

[4]Lenard. An object-oriented framework for model management[J]. Decision Support Systems, 1995, 13: 133—139

[5]黄梯云. 智能决策支持系统[M]. 北京: 电子工业出版社, 2001

[6]陈世福. 智能决策支持系统 NUIDSS 的设计与实现[J]. 软件学报, 1994, (6): 23—28

[7]史忠植. 智能决策支持系统开发环境 DEIDS, 人工智能与智能计算机[M]. 北京: 电子工业出版社, 1991

[8]Hillston J, Ribaudo M. Stochastic process algebra: A new approach to performance modeling[A]. Modeling and Simulating of Advanced Computer Systems[C]. Gordon Breach: 1998. 235—255

[9]Donatelli S, Hillston J, Ribaudo M. A Comparison of Performance Evaluation Process Algebra and Generalized Stochastic Petri Net [R]. PNPM'95, 1995. 248—256

[10]Baeten J C M, Bergstra J A. Mode Transfer in Process Algebra Rapport CSR 00-01[R]. Vakgroep Informatica, Technische Universiteit Eindhoven, 2000

(下转第 46 页)

Study on developing strategies of B2B electronic commerce platform in buyer's market

CHEN Xiang , ZHONG Wei-jun , MEI Shu-e
School of Economics and Management , Southeast University , Nanjing 210096 , China

Abstract : Electronic commerce is a new model of commerce based on the economy of network. B2B electronic commerce platform provides a marketplace for transactions between enterprises. In this paper , the buyers and seller's benefit model of buyer-pull B2B electronic commerce system is established according to the principle of maximizing benefit in buyer's market. Buyers' strategies , including setting the level of substitute , the rate of restoration and the level of subsidy , are studied to induce sellers to join the electronic commerce platform. Finally the effects of the development of technology and the economies of scale on electronic commerce are analyzed based on the model.
Key words : electronic commerce ; buyer-pull ; strategy of inducement ; substitute ; restoration ; subsidy

(上接第 28 页)

[11] Javasoft : Java Beans Specification 1.1 [EB/OL]. http://splash.javasoft.com/ .
[12] DCOM whitepaper [EB/OL] . http://www.microsoft.com/com/wpaper/default.asp.

Modeling and implementing of DSS model system based on process algebra

SUN Jing , ZHAO Hui-qun
Department of Computer , North China University of Technology , Beijing 100041 , China

Abstract : By introducing the process algebra to the work of describing DSS model system , a new paradigm , called DSS model algebra , for modeling DSS model system is proposed. There are some properties for model communication such as invocation , selection , repeat , synchronization and concurrence involved in the new paradigm. The implement technology is also discussed. By comparing the DSS model system based on model algebra and model system of components , it also gives an implement example for model system based on components.
Key words : DSS ; model ; process algebra ; components