维普资讯 http://www.cqvip.com

基于业务行为与业务对象约束的业务规则研究

林金娇　王海洋
（山东大学计算机科学与技术学院　济南 250061）

摘　要　研究了一类基于业务行为与业务对象约束的业务规则。从业务需求的复杂业务规则出发，探讨了这类复杂业务规则的含义，以及业务行为和业务对象、业务约束的关系。在此基础上，对此类业务规则提出了一种形式化描述。提出了此类业务规则驱动的面向对象建模方法。实例表明，该方法较好地解决了由于类之间关系改变使得应用程序频繁变化的问题。

关键词　业务规则，业务行为，业务约束，业务对象

Research on Business Rules about Business Activity-Business Object Constraints

LIN Jin-Jiao　WAGN Hai-Yang
(School of Computer Science and Technology, Shandong University, Jinan 250061)

Abstract　A kind of business rules about business activity-business object constraints is researched. In the case of complex business rules of busiess requirements, the following three objectives are taken into account: (a) to discuss the meaning of this type of complex business rules, (b) to study the relations among business activity, business object and business constraint, (c) to present a formalization of this type. A methodology of this type business rule-driven object-oriented model is also proposed. Through an example, it is proved that this methodology can solve the problem of application code changing because of the relationship among classes changing.

Keywords　Business rules, Business activity, Business constraint, Business object

面向业务规则的系统分析方法将规则和经营策略从应用程序中分离，形成“业务规则”，像管理数据一样管理“业务规则”。这样，当需求变化时，只要修改相应的业务规则，不需要修改应用程序代码，极大地提高了系统的灵活性。

目前，面向业务规则方法已经受到广泛的关注。Irma Valatkaite[3~5]提出了业务规则建模概念图方法，用 UML 的格式存储业务规则（ECA 的格式），规则执行通过主动数据库触发来实现。在此研究基础上，架构了 BR-Centric 的信息系统开发框架。P. Kardasis[6]和 W. M. N. Wan-Kadir[7]研究了业务规则的分类、业务规则的描述（用 IF ELSE THEN 格式），以及业务规则和信息系统中的软件设计的映射，提出了信息系统的 MBRM 开发框架。由于这些方法都是采用 IF ELSE THEN 或 ECA 格式来表示规则，因此存在局限性，不能表示复杂的业务规则。

本文从业务需求的复杂业务规则出发，探讨了业务行为与业务对象约束的一类复杂业务规则的含义。在此基础上，研究了这类业务规则各组分之间的关系，并对各组分进行了形式化描述，是目前业务规则研究的一种补充。

1　ACO 规则

目前，关于业务规则没有统一的定义，不同的研究者根据其研究背景提出了不同的定义。

• Business Rule Group[1]认为，业务规则是对业务中某些定义和限制的描述，用于维持业务结构或控制和影响业务的行为。

• H. Herbst[2]认为，业务规则就是关于业务如何开展的描述，如组织中有关状态转变和过程处理等的准则和限制。

本文从业务需求的复杂业务规则出发，探讨了业务行为与业务对象约束的一类复杂业务规则的含义。为了方便，我们称基于业务行为与业务对象约束的这类规则为 Activity Contrain Object 规则，简称 ACO 规则。

定义 1.1　ACO 规则是描述业务中影响业务行为对象的约束和限制。它包含两个逻辑义项。把它稍加精确化，可以表述为：ACO 规则是由业务行为、业务约束、业务对象构成的三元组，它们之间以特有的方式相互联系在一起。ACO 规则中的业务行为、业务约束、业务对象称为 ACO 规则的组分（组成部分），最小的即不能再细分的组分称为 ACO 规则的元素或要素。

义项 1 规定，ACO 规则的第一个特点是结构的完整性，ACO 规则由业务行为、业务约束和业务对象组成。义项 2 规定，ACO 规则的第二个特点是相关性或相干性，ACO 规则中不存在与其他元素无关的孤立元素或组分，所有元素或组分都按照该 ACO 规则特有的、足以与别的 ACO 规则相区别的方式彼此关联在一起，相互依存，相互作用，相互补充，相互制约。

ACO 规则是业务行为、业务约束、业务对象的三元组。下面分别给出各组分的定义。

定义 1.2　业务行为是指业务活动中最原子的操作，是业务活动中不可再分的操作。复杂的业务活动由这些原子行为组合而成。

定义 1.3　业务约束是指施加于业务行为上的限制和约束。

林金娇　博士生，研究方向为数据库与软件工程；**王海洋**　教授，博士生导师，研究方向为数据库与软件工程，CSCW。

维普资讯 http://www.cqvip.com

定义 1.4 业务对象是指符合业务约束条件下的行为对象。

业务约束施加于业务行为上，约束业务行为，而业务对象是业务约束约束行为对象的结果。业务对象是业务行为最终操纵的对象。它们之间的关系如图 1。

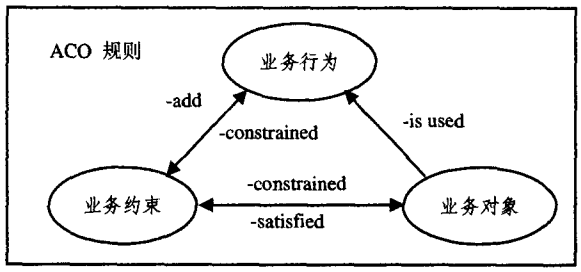

图 1 ACO 规则各组分之间的关系

ACO 规则必须有业务行为和业务对象。业务行为是 ACO 规则的输入，业务对象是 ACO 规则的输出。业务行为是一条 ACO 规则的触发点，与应用程序相对应。业务对象是 ACO 规则管理系统执行的结果，是提供给应用程序处理的数据实体集合。业务约束是施加于业务行为上约束行为对象的条件，它可以是空集。

2 ACO 规则各组分分析

根据本文的定义，ACO 规则可以表示成

$$ule=\langle Operation, Constraint, Object\rangle \quad (1)$$

1) *Operation*

定义 2.1 *Operation* 是指输入的业务行为的基本操作名或行为名。

2) *Constraint*

定义 2.2 *Constraint* 是施加于业务上的限制和约束。我们可以从行为者和行为的关系以及行为和行为对象的关系研究约束。行为者和行为之间的关系可以看作是对行为权限的约束，行为和行为对象之间的关系，可以看作是对行为所涉及对象的约束。它们之间的这种关系如表 1 所示。

表 1 业务行为组分/业务约束关系表

业务行为组分之间的关系	业务约束(Constraint)
行为者/行为	行为权限约束
行为/行为对象	对行为涉及对象的约束

这两类约束其实可以归结为一种约束，即对行为有用的对象约束。这种约束可以分为静态约束、集合势约束和设定的对象操作约束。

(1)静态约束

静态约束描述的是对象的静态属性。有如下几种：

$$\langle Object\rangle.\langle Object\ Attribute\rangle \geqslant \langle Value\rangle \quad (2)$$

$$\langle Object\rangle.\langle Object\ Attribute\rangle in(\langle Value1\rangle,\langle Value2\rangle) \quad (3)$$

$$\langle Object\rangle.\langle Object\ Attribute1\rangle=\langle Object\rangle.\langle Object\ Attribute2\rangle+\langle Object\rangle.\langle Object\ Attribute3\rangle \quad (4)$$

$$\langle Object1\rangle.\langle Object\ Attribute1\rangle \geqslant \langle Object2\rangle.\langle Object\ Attribute2\rangle \quad (5)$$

(2)集合势约束

集合势是指集合基数，即集合元素的多少，集合势的约束有如下几种：

$$Cadinality\ of\ \langle Operation\rangle.\ uses.\langle Object\rangle \geqslant \langle Value\rangle \quad (6)$$

$$Cadinality\ of\ \langle Object1\rangle.\ relates\_to.\langle Object2\rangle \geqslant \langle Value\rangle \quad (7)$$

$$Cadinality\ of\ \langle Object\rangle.\langle Object\ Attribute\rangle \geqslant \langle Value\rangle \quad (8)$$

(3)设定操作的约束

设定操作一般包括 Sum，Min，Max，Avg 等。它们可以表述为：

$$Sum\ of\ (\langle Object\rangle) \geqslant \langle Value\rangle \quad (9)$$

$$Sum\ of\ (\langle Object\rangle.\langle Object\ Attribute\rangle) \geqslant \langle Value\rangle \quad (10)$$

等。

3) *Object*

定义 2.3 *Object* 是指行为对象在约束条件下产生的实体集合。根据行为对象和约束的对应关系，产生不同的 *Object* 行为对象和约束的关系，可以是一对一的关系、一对多的关系和多对多的关系。不同的关系产生不同的 *Object*，如表 2 所示。

表 2 *Object* 行为对象和约束的关系表

行为对象	约束	对象
一个	一条	实体集合
一个	多条	多个实体集合的交集
多个	多条	各个实体集合的集合操作结果

三种关系所产生的对象，形式化可以表示成：

· 一条约束条件约束，一个行为对象

$$Business\ object=\langle Entity\ collection\rangle \quad (11)$$

· 多条约束条件约束，一个行为对象

$$Businessobject=\langle Entity1Collection \cap Entity2Collection \cap \cdots \cap EntitymCollection\rangle \quad (12)$$

· 多条约束条件约束多个行为的对象

$$Businessobject=Collection\ Operation\langle Entity\ 1Collection, Entity2Collection,\cdots,Entityn\ Collection\rangle \quad (13)$$

3 ACO 规则管理系统(ACOBRMS)

基于业务行为与业务对象约束的业务规则管理系统(ACOBRMS)是一个软件平台。其功能包含有：

· ACO 规则库定义

包括 ACO 规则逻辑结构定义、存储结构定义、保密定义，以及相应的信息格式等。

· ACO 规则库管理

包括系统控制、规则存取、更新，以及规则完整性、安全性控制等。

· 规则执行引擎

包括规则的搜索和执行。

· ACO 规则库的建立和维护

包括规则库的建立、规则库的更新，以及规则库结构的维护、规则库恢复等。

ACOBRMS 常常包括三部分：

· 规则库描述语言及其翻译(编译)程序；

· 规则操纵语言及其翻译(编译)程序；

· 规则库管理程序。

应用系统提出一个业务行为的请求给 ACOBRMS，ACOBRMS 从 ACO 规则库里提取相关规则，规则执行引擎执行此规则集合，输出业务对象给应用系统。

维普资讯 http://www.cqvip.com

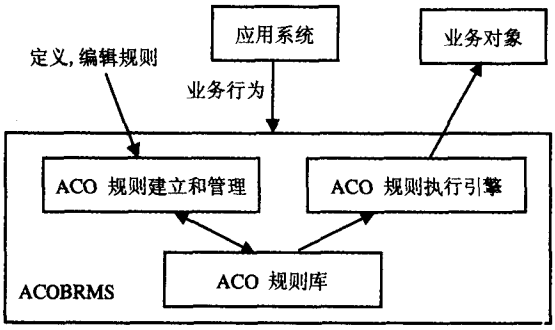

图 2 ACO 规则管理系统和应用系统的关系

4 ACO 规则驱动的面向对象建模方法

传统的面向对象建模方法中,类的描述由类和它们之间的关系组成[8]。行为类的描述是由行为类和行为对象类以及行为对象类之间的关系组成,这些关系最终都在各个类的属性和方法中体现。假设类 A 是行为类,则类与类之间存在如图 3 的关系。

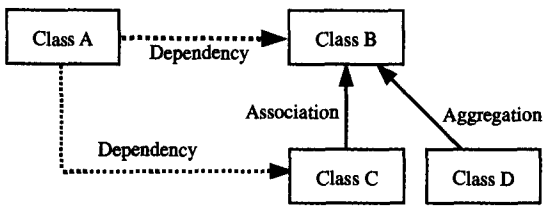

图 3 类与类之间的关系

类 A 的执行要依赖于类 C,而类 A 同时也依赖于类 B,类 C 和类 B之间存在关联关系,类 D 和类 B 之间是聚合关系等等。这些类之间的关系都会出现在各个类的代码中,而且类 A 是行为类,则和它相关的这些类在类 A 的属性和方法中可能都会出现,这是由系统对行为的约束引起的。当这些关系变化时,就会要求重新定义相应的类或修改相应的类定义。

ACO 规则驱动的面向对象建模方法是对行为类描述建模的改进。将行为类与行为对象类之间的关系和行为对象类之间关系从原有的面向对象模型中分离出来,作为 ACO 规则存储在 ACOBRMS 里单独管理,ACOBRMS 给应用提供一个此行为所要操纵的对象基类。这样,新的应用模型中,行为类的描述只有对行为对象基类的调用描述和行为操作的定义描述。当类之间的关系发生变化时,即 ACO 规则发生变化时,不再需要改变类的定义,行为类的属性和方法中不再存在其他行为对象类的状态等属性和方法。这种应用模型能更深刻地揭示系统内部组成的动态交互特性,并且简化了类代码,如图 4 所示。

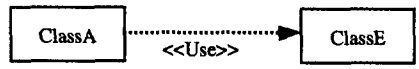

图 4 业务行为类和业务对象类之间的关系

其中类 A 是改进后的行为类,类 E 是需要 ACOBRMS 提供的行为类所涉及的对象的基类,类 A 中有相应的接口可以调用类 E。

ACO 规则中的业务行为对应的是行为类,业务约束对应的是行为类和行为对象类之间以及行为对象类之间的关系,业务对象是符合业务约束的行为对象基类。

总的来说,ACO 规则驱动的面向对象方法是把行为类和行为对象类之间以及行为对象类之间的关系从原来的面向对象模型中分离出来单独管理,达到对对象的动态控制,然后把符合业务约束的行为对象抽象成一个业务对象,供应用程序调用。

5 实例

本文提出的方法已经在某高校的教务管理系统开发过程中采用。考虑到该方法的研究主要以系统开发中占据重要地位的类为背景,我们以系统中的"选课"行为类为例说明该方法的应用。

行为类:选课

行为对象:学生,课程

对行为对象的约束:

- 对学生的约束:本校学生;
- 对课程的约束(是在实例化学生以后):

(1)学生以前没有选过的课程;

(2)课程的已经注册人数小于课程的最大容量;

(3)学生已经通过本课程的前置课程。

图 5 描述了传统面向对象开发方法的"选课业务"协作图。

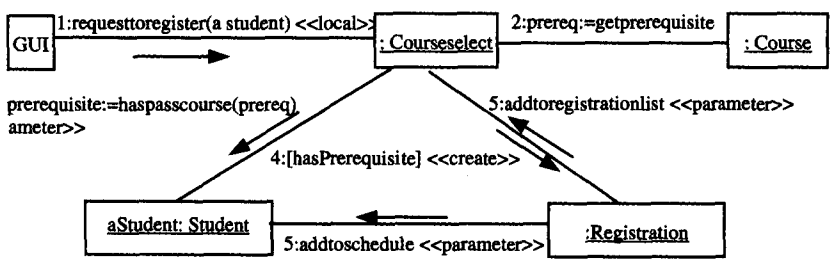

图 5 传统面向对象开发方法的"选课业务"协作图

图 6 描述了利用本文提出的方法处理过的"选课业务"协作图。

在图 6 中,图 5 中的对象之间的约束关系被抽取出来,作为 ACO 规则放在 ACOBRMS 里进行统一管理。也就是说,在图 6 中不存在对象之间的约束关系。这样,当 ACO 规则变化时,不需要修改应用程序代码。具体地说,在图 6 中,Courseselect 类被用来注册课程,ACOcourseselect 类是 ACOBRMS 提供的,包含一个学生表和一个课程表。其中,Courseselect 类包含接收注册和注册课程两个方法。接收注册使用学生表,只要学生在该学生表中就可以进行注册;注册课程使用课程表,只要是表里的课程就直接注册。由于我们在行为类中定义了行为操作,当约束学生和课程这两个对象的约束条件变了,选课类不需要修改或重新定义。同样,学生和课

(下转第 267 页)

维普资讯 http://www.cqvip.com

因为在操作 Leave 谓词部分中，有 items = ⟨item!⟩⌢items'，其后状态变量 items'的值要求原序列 items 移去第一个元素以后得到的序列，所以可以用函数 tail 来表示，这就经过了等价变换。

由(1)、(2)，我们可得队列类 Queue[T]相应的参数化主进程为：

Queue = Value[init_variables]; Queue[(Items, ⟨ ⟩), (Size, 0)]

Queue[(Items, items), (Size, size)]= (Join[(Items, items), (Size, size)][]Leave[(Items, items), (Size, size)])

结论 Object-Z 是形式规格说明语言 Z 的面向对象扩充，基于严格的集合论与数理逻辑，能够很好地描述大型复杂系统的算法与结构，可以对其规格说明进行推理。进程代数 CSP 能够方便地描述并发系统，可以利用进程迹来进行推理。Object-Z 和 CSP 各有本身的局限性，即 Object-Z 不能描述并发系统及其行为，而 CSP 不能很好描述系统的结构与算法，因此，目前 Object-Z 与 CSP 相结合是一个热点。结合后可以模块化描述并发的大型面向对象系统，它们的语言可以混合在一起，因而求精与验证对它们结合后的规格说明需要分别进行处理，因为 Object-Z 与 CSP 有不同的表示形式与语义，这样将带不方便。本文提出了一个方法，把 Object-Z 规格说明转化为 CSP 规格说明，对结合后的规格说明可以按 CSP 规则与方法一致来进行求精与验证。特别地，Object-Z 一般不能进行模型检查来验证它，而 CSP 有一个模型检测工具 FDR，经过转化后的 Object-Z 形式规格说明可以用 FDR 来进行模型检查。

对转化后的 Object-Z 形式规格说明的模式检查与验证是我们今后的工作。

参 考 文 献

1 Smith G. The Object-Z Specification Language. Advances in Formal Methods, Kluwer Academic Publishers, 2000

2 Smith G. Reasoning about Object-Z specification. In: APSEC 1995. 489～497

3 Smith G. A fully abstract semantics of classes for Object-Z. Formal Aspects of Computing, 1995, 7(3): 289～313

4 Smith G, Derrick J. Specification, Refinement and Verification of Concurrent Systems-An Integration of Object-Z and CSP. Formal Methods in System Design, 2001, 18(3): 249～284

5 Olderog E R, Wehrheim H. Specification and (property) inheritance in CSP-OZ. Science of Computer Programming, 2005, 55(1-3): 227～257

6 Hoenicke J, Olderog E R. CSP-OZ-DC: a combination of specification techniques for processes, data and time. Nordic Journal of Computing, 2002, 9 (4): 301～334

7 Mahony B, Dong Jin Song. Blending Object-Z and Timed CSP: An introduction to TCOZ. In: Proceedings of the 20th international conference on Software engineering, IEEE, 1998. 95～104

8 Sun Jing, Dong Jin Song. Specifying and Reasoning about Generic Architecture in TCOZ. In: APSEC'02, IEEE, 2002. 405～414

9 Kim Il-Gon, Choi Jin-Young. Formal Verification of PAP and EAP-MD5 Protocols in Wireless Networks: FDR Model Checking. In: Proceedings of the 18th International Conference on Advanced Information Networking and Applications, 2004

10 Lowe G, Roscoe B. Using CSP to Detect Errors in the TMN Protocol. IEEE Transactions on Software Engineering, 1997, 23(10)

11 Hoare C A R. 通信顺序进程. 周巢尘译. 北京大学出版社, 1988

(上接第 258 页)

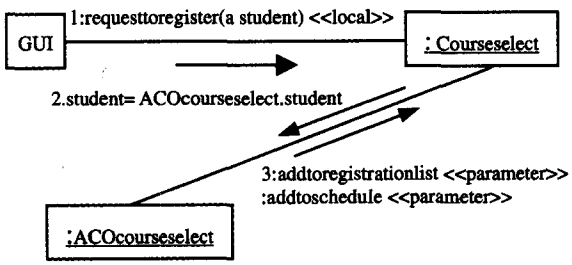

图 6 利用本文提出的方法处理过的"选课业务"协作图

程这两个类也不需要重新或修改类定义。根据我们提供的形式化方法，可以将上述规则表示成：

BR1:⟨*Courseselect*, *xh*=*student. xh*, *student*1⟩

BR2:⟨*Courseselect*, *Course* ⟨ ⟩ *student. selectedcourse*, *Course*1⟩

BR3:⟨*Courseselect*, *registrationlist. size*()⟨*Course. getminimum*(), *Course*2⟩

BR4:⟨*Courseselect*, *Course. getprerequisite*() = *student. haspassedcoures*(), *Course*3⟩，等等。执行完这些规则集，ACOBRMS 提供一个 ACOcourseselect 对象给应用，包含符合条件的学生表和课程表。

结论 快速灵活地响应需求变化是面向业务规则的信息系统分析方法的目标。本文从需求理解出发，提出了 ACO 这类业务规则，以及此类规则的形式化描述方法，并提供了基于 ACO 规则驱动的面向对象建模方法。实例表明，该方法较好地解决了由于类之间关系变化使得应用程序频繁变化的问题。

参 考 文 献

1 GUIDE International Corporation. Guide Business Rules Project: [Final Report]. 1995

2 Herbst H. Business Rules in Systems Analysis: a Meta2model and Repository System. Information Systems, 1996, 21(2): 147～166

3 Valatkaite I, Vasilecas O. A Conceptual Graphs Approach for Business Rules Modeling. Computer Science, 2003, 2798: 178～189

4 Valatkaite I, Vasilecas O. Automatic Enforcement of Business Rules as ADBMS Triggers from Conceptual Graphs Model. Information Technology And Control, Kaunas, Technologija, 2004, 2(31): 36～42

5 Valatkaite I, Vasilecas O. On Business Rules Automation: The BR-Centric IS Development Framework. Computer Science, 2005, 3631: 349～364

6 Kardasis P, Loucopoulos P. Expressing and organising business rules. Information and Software Technology, 2004, 46: 701～718

7 Wan-Kadir W M N, Loucopoulos P. Relating evolving business rules to software design. Journal of Systems Architecture, 2004, 50: 367～382

8 张维明. 信息系统建模. 北京: 电子工业出版社, 2002