基于通用框架 ICETIP 应用集成模型*)

智永锋　张　骏　万海东
(西北工业大学自动化学院　西安 710072)

摘　要　如何在复杂多变的环境下实现对应用系统软件的快速设计和升级，已成为软件开发所关注的基本问题。提出了有效的应用系统设计模型，它能够将应用系统抽象成表单流、数据流和事物流，由这些"流"可配置出不同的应用系统软件，并且该模型可以根据业务变化进行整个应用系统功能重组而不需要编写一句程序。因此，利用该模型设计应用系统软件只需两个步骤：需求分析和系统部署。

关键词　XML，中间件，软件工程

An Application Integrated Model ICETIP Based on General Framework

ZHI Yong-Feng　ZHANG Jun　WAN Hai-Dong
(College of Automation, Northwestern Polytechnical University, Xi'an 710072)

Abstract　Fast design and upgrade of software are the fundamental issues in the Application System as well as processing in a rapidly changing complicated environment. In this paper an effective design model is present. It can abstract the Application System into form flow, data flow and event flow, which can configure and reconfigure different internet applications. Once the business logic is changed, the model can be adjusted according to new requirements without any programming. It is concluded that when using this model to develop specific software of the Application System, it only needs two processes: requirement analysis and application deployment.

Keywords　XML, Middleware, Software engineering

1　引言

如何在异构和分布式环境下，满足对扩展性、重用性、移植性、安全性、维护性等方面的要求，已成为企业应用系统开发所关注的焦点之一。但是应用系统软件的设计处于传统思想影响之下，变化的需求难以快速的反映在软件功能之中。在传统软件工程技术无力突破的情况下，应采取什么样的新策略呢?

本文提出了企事业信息综合事务处理系统 ICETIP(Information-Centered Enterprise Transaction-Integrated Processing System)模型，它是信息处理的应用核(AppCore)，具有业务逻辑可配置，系统功能可重组特性。在应用配置时实现了可见即可得，在使用时实现了所得即所需。ICETIP 这种基于业务可重组和可配置的特征，使得该应用核(AppCore)能够适用不同行业和不同业务逻辑的事务处理，重要的是在不同行业和不同业务的应用部署过程中，不需要任何程序修改和增加新的功能模块。

ICETIP 模型突破传统应用系统的设计模式，从项目开发模式转变为技术服务模式，它无须编程就能重组整个应用系统。采用 ICETIP 开发特定的应用系统时，只有需求分析和系统部署两个过程，减少了应用系统开发所需要的概要设计、方案详细设计、编程、测试和系统维护等多个繁杂过程，同时保证了应用系统运行的可靠性。经过一段时间后如果需要扩展新的业务，只需在后台应用服务器中，通过配置新的功能模块，经过适当的权限控制，在不改变客户端应用程序情况下，就可以运作新的业务，大大缩减了应用系统开发时间，提高了经济效益。同时，在某种程度上促进了软件的标准化发展。

2　设计思想

在应用系统中，可以把所有的业务逻辑抽象成表单流、数据流和事件流，如图 1 所示。这样不同功能的应用系统可以定义为：应用系统＝表单流＋事件流＋数据流。

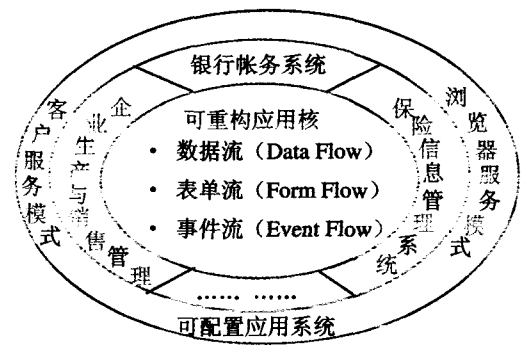

图 1　应用系统抽象

表单流是业务逻辑处理的执行界面，一系列的执行界面切换就构成了业务流程。事件流是用户在某个执行界面中操作而触发的一系列业务数据的传递和处理，它可以触发表单流、数据流和其它事件流的动作。数据流是伴随着事件触发产生的业务数据的存储、转移、更新等和由此引起的一系列活动。每种"流"中按需要又可以抽象出若干个"功能豆"，并且

*)基金项目：西北工业大学研究生创新种子基金资助项目(Z200562)。智永锋　博士，主要研究领域为软件工程，大系统工程；张　骏　教授，博士生导师，校长助理，主要研究领域为软件工程，大系统工程，分布式计算等；万海东　硕士，主要研究领域为软件工程。

这些“功能豆”能够被 XML 语言描述。如：表单流中开发了若干控件，数据流中开发了“数据写入功能豆”、“数据读出功能豆”等等。其中通过对应用系统用户界面的分析，总结出了 16 种控件，在配置用户界面过程中，就是对这些控件进行拖动等的一系列操作，来布置用户界面布局，对于一些不能反映到配置界面的特殊信息，通过控件属性配置单元对这些属性进行配置和修改；根据用户操作类型差异，抽象了 9 种不同的触发事件；通过对现有应用系统软件工作模式的研究，抽象出了包括“生成新的数据变量”在内的 21 种不同类型活动事务。

用 XML 语言描述的这些“功能豆”可重配置出各种不同的表单流、事件流和数据流，再由这些“流”重构成不同的应用系统，实现了无需编程就能自动重组应用系统的功能，使得应用系统软件的开发非常快。真正做到了设计系统时可见即可得，使用系统时所得即所需，解决了传统软件工程无法解决的问题。

3 ICETIP 体系结构

ICETIP 模型基于网络应用的三层体系架构，它的整体结构如图 2 所示。主要由三部分组成：应用配置管理系统、客户程序和 ICETIP 服务器。

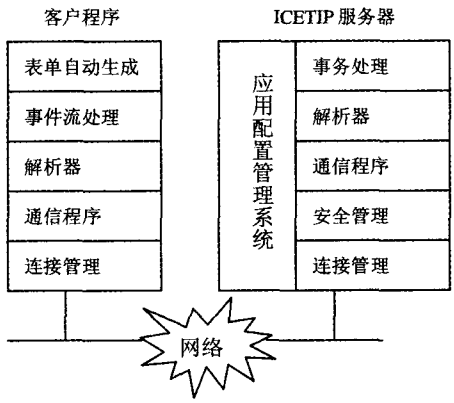

图 2 ICETIP 体系结构

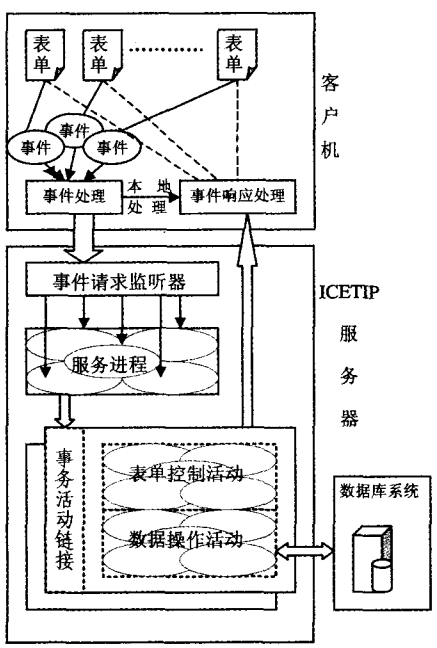

图 3 应用系统体系结构

应用配置管理系统：该模块根据不同的业务逻辑，可以把应用系统抽象成表单流、事件流和数据流，每种“流”中按需要抽象出若干个“功能豆”。在配置应用系统的过程中，首先以可视化方式拖动控件配置完成用户界面；然后通过在配置用户界面控件的触发事件时，完成对活动事务配置；最后经过系统的编译处理，把表单流、事件流和数据流转换成可以用 XML 语言描述的形式，并以特定的组织形式存入 ICETIP 系统库中，形成 XML 脚本。当用户需要升级应用系统时，只需把用 XML 语言描述的应用系统读取并解释到应用配置管理系统中，在原有的基础上经过修改、配置和编译成新的 XML 脚本，并将 ICETIP 系统库中原有 XML 脚本替换，解决了应用系统的快速升级问题。

客户程序：为了管理方便，在 ICETIP 模型中引入了角色概念，每一角色与特定的表单流相联系。用户登录采用 ICETIP 模型建立的应用系统时，根据所属的不同角色，表单自动生成模块通过通信程序与 ICETIP 服务器协商得到相应角色的 XML 脚本，经过解析器的解释说明形成适用于不同业务逻辑的用户操作界面。对于用户的请求，客户程序首先查看自己是否可以独立完成。如果可以，事件流处理模块独立完成后交由事件响应处理器并显示到用户界面，提高了应用程序的执行速度；否则，请求 ICETIP 服务器完成对用户请求的处理，ICETIP 服务器处理完成后，把结果返回给客户端的事件响应处理器并显示到用户界面，以便用户下一步操作，如图 3 所示。客户程序与 ICETIP 服务器之间的通信协议由两个类实现，一个是客户程序对 ICETIP 服务器请求协议，包括了用于定向 XML 脚本的用户界面 ID(用户界面 ID 与 XML 脚本是一一对应的)、控件 ID、用户输入操作信息等；另一个是 ICETIP 服务器对客户程序响应协议，包括了查询数据库结果、操作数据库是否成功、执行事件等信息。

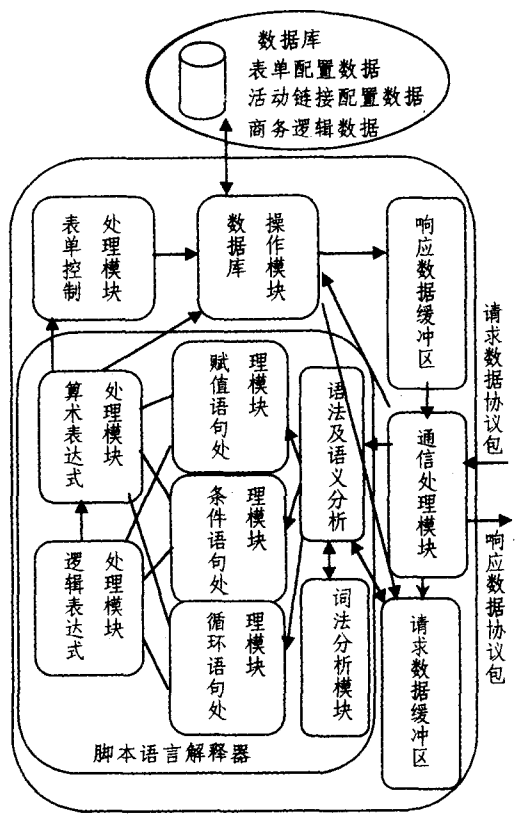

图 4 活动链接处理流程

ICETIP 服务器是整个系统的核心，主要功能是完成对客户端连接管理、数据库操作、XML 脚本解释，如图 3 所示。ICETIP 服务器与客户程序之间的连接采用的是对象池技术，它将具体功能抽象为元对象，在对象池中对元对象进行管理，并且对象池中始终维护一定数量的元对象等待系统调用，当某个元对象使用完毕后放回池中等待下次调用，通过这种方式可减少功能对象频繁创建与销毁带来的系统性能下降。对数据库操作采用的是基于 XML 数据库中间件，它不但支持 JDBC，ODBC 等接口，而且还支持数据连接池技术。通过在客户程序定义一个与用户界面一一对应的 ID 值，ID 值通过配置与 XML 脚本也是一一对应的，这样数据库中间件根据客户端请求，通过解析器完成对 XML 脚本解释，完全实现了对 SQL 语言和数据库操作屏蔽。XML 脚本解析是 ICETIP 服务器的核心，其详细的对请求数据协议包处理流程如图 4 所示，主要功能是完成对表单配置数据、活动链接配置数据和商务逻辑数据的 XML 脚本解释工作，其中 XML 脚本容括了赋值语句、条件语句和循环语句。这样 ICETIP 模型形成了与角色相链接的用户界面和特定的活动事务操作。

采用 ICETIP 模型配置应用系统前，客户程序、ICETIP 服务器和它们之间的通信协议已定义完成。建立应用系统时，仅仅需要利用抽象出来的“功能豆”以可视化的方式配置完成应用系统的用户界面、系统动作和活动事务。

4 ICETIP 设计与实现

4.1 表单自动生成模块

表单自动生成模块是一种面向软件可重构、可配置、开放实现、软件反射和支持用户界面自动生成的模型。系统利用任务模型获得登录用户的权限信息和对应权限信息的任务，利用数据模型获得客户端的数据信息，利用功能模型获得系统的功能需求，然后依据交互模型的描述，利用表示模型建立内部对象和外部显示元素的对应关系，在界面模板技术的支持下，建立界面构成和布局，实现客户端页面的自动生成，模块结构如图 5 所示。在启用应用系统前，首先根据用户需求利用应用配置管理系统配置出用户界面，经过编译系统生成 XML 语言中间代码并存入 ICETIP 系统库中。

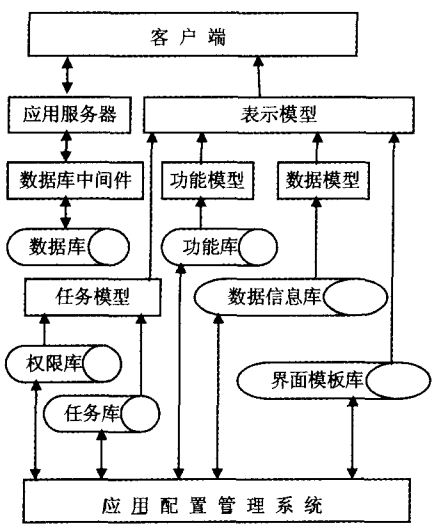

图 5 客户端应用自动处理系统

启动 ICETIP 平台配置的应用系统后，首先需要提交用户名和密码到 ICETIP 服务器验证用户信息合法性，ICETIP 服务器根据用户角色读取相应的流配置信息，按照流配置的顺序依次把活动事务中的每个活动装载到客户端表单自动生成模块中生成用户初始界面。然后 ICETIP 服务器就可以根据不同的输入，完成不同的动作，当客户端产生访问后台数据库的请求后，由客户程序封装成数据包发送到 ICETIP 服务器，通过数据库中间件完成操作。客户程序接收到处理结果，并显示到用户界面后，等待用户进一步操作请求。

4.2 数据库操作模块

数据库操作模块是对数据库操作的中间件，包括了客户数据源解析器、SQL 执行器和数据库数据源封装器三部分，其中 XML 脚本解释器是重要辅助部分，模块结构如图 6 所示。

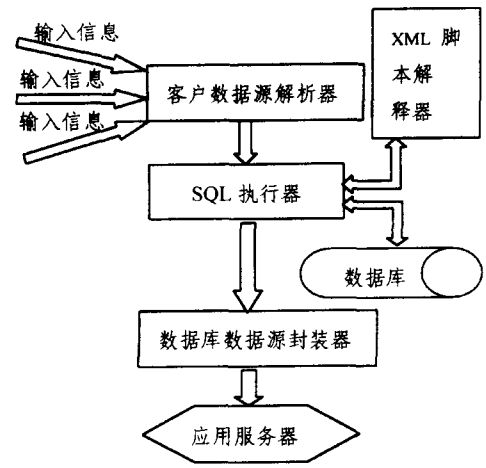

图 6 数据库中间件体系结构

它提供了两个操作接口：开放接口和功能接口。启用应用系统前，首先需通过开放接口配置完成数据库操作方式和 XML 脚本。用户操作应用系统时，客户程序把用户的操作提交到 ICETIP 服务器后并通过调用模块功能接口，由数据库中间件完成对数据库的操作。比如：ResponseEnv rEnv = dbOp. DBhandle(env)，其中 ResponseEnv 为数据库数据源封装器中封装的对象，dbOp 为数据库中间件定义的变量，DBhandle 为数据库中间件提供调用的功能函数，env 为客户数据源封装器中封装的对象定义的变量，使得对开发人员完全屏蔽了 SQL 语言。数据库中间件处理完成后反馈给客户程序必要的信息，以便客户端进行有效的操作，其中响应客户程序请求的数据包格式如下：

动作：6Byte	状态：2Byte	行数：6Byte
一维结果集	二维结果集	

动作表示对数据库进行操作，共包括 insert，delete，update，select，TwoSelect 等 5 种动作，其中 TwoSelect 表示嵌套查询；状态表示对数据库的操作是否成功；行数表示对数据库操作影响的行数；一维结果集为 select 查询的结果，它存放于一个哈希表中；二维结果集为 TwoSelect 查询的结果，它存放于两个哈希表中。

4.3 脚本表达式计算

本编译系统涉及编译和解释两部分，既语义分析之后，将得到的语句翻译成中间代码 XML 语言，并存入系统库。利用 ICETIP 模型建立的应用系统启动时，把这些中间代码加载入内存。用户请求某项操作时，XML 脚本解释器选择其中要求操作的脚本进行表达式的计算，替换掉 XML 脚本中的未知参数。比如：在活动事务 UPDATE 类型中，XML 脚本

中的伪 SQL 语句 Update Plan_Fee set Plan_Status=replace (Number,String)中 Plan_Status 的值是未知参数,需要利用客户端传递过来的实参把其中未知参数 replace(Number, String)替换,解释成可以执行的 SQL 语句,这样系统就能够实现对数据库操作。

ICETIP 模型的表单也需要编译和解释,与 SQL 语句的编译原理基本相同,区别在于表单是在应用系统启动时就已经解释完成,而 SQL 语句是在用户需操作数据库时选择其中的 XML 脚本进行解释。编译器与解释器中的表达式的运算和整个系统有两个接口,首先系统调用第一个接口得到表达式的翻译结果,系统处理后,再调用第二个接口进行表达式计算,得到计算结果,表达式计算流程如图 7 所示。

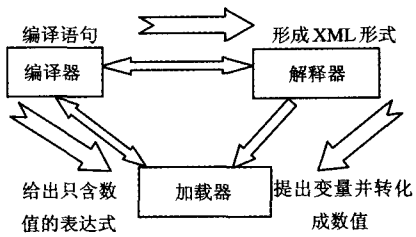

图 7　表达式计算流程

4.4　ICETIP 系统库

ICETIP 系统库包含了 95 个系统表。比如:ISTPKeyValue 主键维护表,ISTPActiveTransXML 脚本维护表,ISTPActiveTrans_ScriptXML 脚本内容维护表,ISTPFirstLevelInfo 系统一级菜单维护表,ISTPSecThirInfo 系统二三级菜单维护表,ISTPTable_Type 活动事务管理表,ISTPControl 控件维护表,ISTPForm 表单维护表,ISTPForm_AT_Rel 表单与 XML 脚本对应表,ISTPForm_Attr 表单属性表等等,其中一部分系统表的相互约束关系如图 8 所示。ICETIP 系统库包括了用户、角色、权限、活动事务、界面等应用系统软件涉及到所有信息。

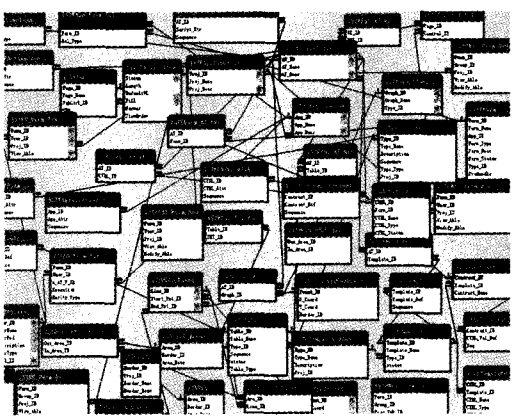

图 8　ICETIP 系统表关系

5　实例

ICETIP 模型改变了现有应用系统开发、维护和升级的传统模式,创造性地提出了全新的运作和管理模式,在一定程度上促使网络应用日趋迈向新的标准化。下面通过一个简单的例子,说明采用 ICETIP 设计应用系统的方法。

学校登记学生某科考试成绩时,需要输入学生的学号、科目和成绩。用户操作界面包含 3 个 Eidt 控件和一个 Button 控件。应用系统设计人员首先拖动控件形成用户操作的界面,然后通过对控件属性、触发事件和活动事务配置,编译生成如下所示信息录入界面的文件结构,即上面所说的 XML 脚本。

```
〈? xml version="1.0" encoding="gb2312" ?〉
〈Form〉
  〈FE type="Edit" ID="56" name="学号"〉
  〈Attri pos="227, 73, 20, 10" default=" "/〉
  〈/FE〉
  〈FE type=" Edit" ID="57" name="科目"〉
  〈Attri pos="227, 85, 20, 10" default=" "/〉
  〈/FE〉
  〈FE type=" Edit" ID="58" name="成绩"〉
〈Attri pos="227, 97, 20, 10" default=" "/〉
〈/FE〉
〈FE type="Button" ID="59" name=""〉
〈Attri pos="305, 115, 20, 10"default="提交"/〉
  〈Actions ID="60" Act="Click"〉
  〈Action AType="Put" SQL="insert into Sch_Report (id, subject, report) values (replace (56, Int), replace (57, String), replace (58, Float))"/〉
〈Action AType="Clean" CtrlID="56, 57, 58"/〉
  〈/Actions〉
〈/FE〉
〈/Form〉
```

用户登录系统后,脚本语言解释器通过对 XML 脚本解释,客户程序把 4 个 FE 按照各自的 pos 属性部署到界面的相应位置,这时用户就可以看到生成的用户界面了。用户在 Edit 中填写完相关信息后,通过单击 Button,应用系统首先会查找到 Actions 属性中 ID 为 60 的事件,同时检查相应事件的触发类型(Act)是否为 Click。如果不是,应用系统提示出错信息,否则此事件立即被激活,送到脚本语言解释器中进行处理。首先进入解释器中的是 Put 动作,此时 insert 语句并不是真正意义上可以执行的 SQL 语句,因为这时 SQL 语句包含未知参数。解释器找出 insert 语句中未知参数 replace (XX,YY),其中的 XX 是控件的 ID,YY 是输入信息的数据类型。例如 XX 是 57,YY 为 String 类型,解释器就会把 ID 为 57 的控件上的值取出,同时把此值用单引号括起来替换 insert 语句中的 replace(57,Stirng),其它的未知参数 replace (XX,YY)作同样处理。这样经过解释器后,insert 语句成为一个可执行的 SQL 语句。这条语句交与执行器执行后,就完成了一个 Put 动作。紧接着下一个动作 Clean 进入解释器,Clean 是一个控件间交互的动作,作用是把指定的控件上的内容清除。在解释器中,通过把 XML 脚本中的 CtrlID 值"56,57,58"解析出来,成为一个单独的控件 ID 的名称,即 56,57,58。把这些控件的 ID 名称依次送到执行器中,执行器根据这些控件 ID 名称把内容清除,这两个动作完成后用户输入操作也就结束了。这时数据库增加了一条新的记录,同时刚刚在 Edit 上输入的内容被清空,等待下一次输入。

结束语　实践证明,采用 ICETIP 模型设计应用系统的思想和理论是可行的。以我们所得的知识,在以前的论文中没有类似的模型提出。主要研究成果和结论如下:

(1)能够实现完全无须编程就能重组整个应用系统。

(2)突破了传统软件工程的设计思路,提出了全新的应用软件设计模式。

(3)ICETIP 平台使得应用系统的开发者和使用者在开发过程中能够有效地交流与协作。开发者根据用户的初始需求很快配置出可以试运行的样本模型,让用户提出修改意见,并很快根据修改意见调整系统,整个应用开发过程实际上是开发者与用户的交流与协商的过程,也是应用系统需求分析的过程。

(下转第 267 页)

lae. In CAV, 2000. 248～263
5 Gastin P, Oddoux D. Fast ltl to bautomata translation. In: CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification, London, UK, 2001. 53～65
6 Dill D L, Hu A J, Wong-Toi H. Checking for language inclusion using simulation preorders. In: CAV '91 Proceedings of the 3rd International Workshop on Computer Aided Verification, London, UK, 1992. 255～265
7 Etessami K, Wilke T, Schuller R A. Fair simulation relations, parity games, and state space reduction for Büchi automata. In: ICALP '01 Proceedings of the 28th International Colloquium on Automata, Languages and Programming, London, UK, 2001. 694～707
8 Henzinger T A, Kupferman O, Rajamani S K. Fair simulation. Inf. Comput., 2002, 173(1): 64～81
9 Andersen H R. Model checking and boolean graphs. Theor. Comput. Sci, 1994, 126(1): 3～30
10 Henzinger M R, Henzinger T A, Kopke P W. Computing simulations on finite and infinite graphs. In: FOCS '95 Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95), Washington, DC, USA, 1995. 453～462
11 Klarlund N, Kozen D. Rabin measures and their applications to fairness and automata theory. In LICS, IEEE Computer Society, 1991. 256～265
12 Jurdzinski M. Small progress measures for solving parity games. In: Horst Reichel and Sophie Tison, eds. STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science. Proceedings, volume 1770 of Lecture Notes in Computer Science, Lille, France, February 2000. 290～301
13 Bustan D, Grumberg O. Simulation-based minimization. ACM Trans. Comput. Logic, 2003, 4(2): 181～206
14 Bustan D, Grumerg O. Applicability of fair simulation. Inf. Comput, 2004, 194(1): 1～18
15 Gurumurthy S, Bloem R, Somenzi F. Fair simulation minimization. In: CAV '02 Proceedings of the 14th International Conference on Computer Aided Verification, London, UK, 2002. 610～624
16 Juvekar S, Piterman N. Minimizing generalized Büchi automata. In: Thomas Ball, Robert B. Jones, eds. CAV, volume 4144 of Lecture Notes in Computer Science, Springer, 2006. 45～58

(上接第 259 页)

UML 元模型，目的是方便建立互转换机制，以实现 MDA。它与 UML 之间有以下区别：

(1)裁减。取消了许多 Java 中不需要的 UML 元类，而保留了 Java 必需的元类：Element、Classifier、Type、Class、Interface、Feature、BehavioralFeature、Parameter 等。

(2)改变与扩展。把 UML 中类似元类改变为 Java 名称，如 Field、Method、Constructor；根据 Java SE5 的新语言成分，扩展了枚举 Enum、诠释 Annotation、可诠释元素 AnnotatableElement、诠释实例 AnnotationInstance 等，图 2 中也扩展了类属。注意本文仅对诠释部分进行了详细描述。

本文提出的三个诠释元类及两个图符对于 UML 扩展诠释具有指导意义。UML 允许自定义构造型来扩展已有元类，以支持新的建模元素。关于如何扩展 UML 来支持诠释建模，作为另一种技术途径另文再述。

结论与进一步工作 诠释相对于注释，其特征是基于类型、静态实例化、关联目标的实例化、实例不变性。为达到诠释的可视化、规范化建模的目的，本文以 MOF 为规范，参照 UML 元模型，扩展已有 Java 元模型，提出一个新的 Java SE5 元模型，并确定了诠释的图符以支持可视化建模。该元模型能反映诠释的特征，可支持诠释的建模和编程。该元模型具有一致性、规范性、简单性。

进一步的工作包括扩展该元模型以支持 AOP(如 AspectJ5)及 AOM(Aspect-Oriented Modeling)，研究动态诠释及应用，实现元模型以开发 MDA 及相关建模工具等。

参 考 文 献

1 Gosling J, Joy B, Steele G, Bracha G. The Java Language Specification (Third Edition)[M]. Addison-Wesley Professional; 2005 ISBN 0321246780
2 许满武，严悍，张琨，李千目. Java 程序设计[M]. 北京：高等教育出版社，2006，ISBN 704019645X，普通高等教育"十五"国家级规划教材
3 Harold E. An Early Look at JUnit 4 [OL]. http://www-128.ibm.com/developerworks/java/library/j-junit4.html, 2005. 9. 13
4 Hibernate Annotations Reference Guide, v3. 2. 0 [OL]. http://www.hibernate.org/hib_docs/annotations/reference/en/html/, 2006. 4
5 Kiczales G, Mezini M. Separation of Concerns with Procedures, Annotations, Advice and Pointcuts [C]. In: proceedings of ECOOP 2005. Glasgow, UK, July 2005
6 Yan H, Kniesel G, Cremers A B. A Meta Model for AspectJ [R]. Technical Report IAI-TR-2004-3, October 2004, Computer Science Department III, University of Bonn, Germany, ISSN 0944-8535
7 Java Document. Getting Started with the Annotation Processing Tool (apt)[OL]. http://java.sun.com/j2se/1.5.0/docs/guide/apt/GettingStarted.html, 2005
8 OMG(Object Management Group), Unified Modeling Language: Specification: Superstructure [EB]. version 2.0, 2005. 7. 4
9 OMG(Object Management Group), Unified Modeling Language Specification: Infrastructure [EB]. version 2.0, 2004. 10. 16
10 Kleppe A, Warmer J, Bast W. MDA Explained: The Model Driven Architecture——Practice and Promise [M]. Addison-Wesley Professional; 1st edition (April 25, 2003) ISBN: 032119442X
11 Java Document. Mirror API [OL]. http://java.sun.com/j2se/1.5.0/docs/guide/apt/mirror/index.html, 2005
12 Java Document. Reflection API [OL]. http://java.sun.com/j2se/1.5.0/docs/guide/reflection/index.html, 2003

(上接第 263 页)

参 考 文 献

1 Maes P. Concepts and experiments in computational reflection. ACM SIGPLAN Notices, 1987, 22(12): 147～155
2 Bates J. The role of emotion in believable agents. Communication of the ACM, 1994, 37(7): 122～125
3 Maes P. Agents the reduce work and information overload. Communication of the ACM, 1994, 37(7): 31～40
4 Kruchten P B. The 4+1 view model of architecture. IEEE software, 1995, 12(6): 42～50
5 Nwana H S. Software Agents: An Overview. Knowledge Engineering Review, 1996, 11(3): 205～244
6 Magee J, Kramer J. Dynamic structure in software architectures. ACM SIGSOFT Software Engineering Notes, 1996, 21(6): 63～83
7 Richard N T. A Component and Message-Based Architectural Style for GUI Software. IEEE Transaction on Software Engineering, 1996, 22(8): 390～406
8 Wooldridge M. Agent-based software engineering. IEEE Proc. on Software Engineering, 1997, 144(1): 26～37
9 Perkowitz M, Etzioni O. Adaptive web pages: Automatically synthesizing web pages. In: Proceedings of AAAI/IAAI' 98, 1998. 727～732
10 Wermelinger M. Towards a chemical model for software architecture reconfiguration. IEEE Proc-Softw, 1998, 145(5): 130～136
11 Oreizy P, Taylor R N. On the role of software architectures in runtime system reconfiguration. IEEE Proc-Softw, 1998, 154(5): 137～144