一种基于领域模型和构件组合的软件开发框架

王晓燕, 刘淑芬, 张 俊
(吉林大学 计算机科学与技术学院, 吉林长春 130012)

摘 要: 本文提出了一种基于领域模型和构件组合的软件开发框架, 使用领域模型捕获系统业务静态需求, 描述领域内业务对象之间的静态关系, 通过领域应用框架描述系统的共性, 并在框架中提供足够多的反映领域可变性的扩展点, 在此扩展点上可以集成各种类型构件, 创建新的应用系统, 来满足领域内不同具体应用的特定需求. 实验表明, 该框架能够利用同一领域软件系统间的共性和变化性并实施有效的控制, 促进了软件复用, 提高软件生产效率和质量.

关键词: 领域模型; 构件组合; 领域框架
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2009) 03-0540-06

A Framwork based on Domain Model and Component Composition

WANG Xiao-yan, LIU Shu-fen, ZHANG Jun
(College of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China)

Abstract: A framework based on domain model and component composition is presented, in which domain models can be used to capture the static business requirement, describe the static relationship between two business objects. The domain application framework describes commonness and provides variety extension, in this extension all kinds of components can be integrated into the system. In this way, a new system can be built and the user requirements can also be needed. Results show the framework is capable of find the commonness and variety and make use of them, and the software quality can be improved.

Key words: domain model; component composition; domain framework

1 引言

在软件开发行业, 软件开发的新技术、新方法不断涌现, 这些新技术和新方法在给人们带来利益的同时, 又要人们忍受前期投资失去价值的现实, 并且新技术本身也在不断更新变化中, 不能保证完全做到向后兼容. 同时, 业务需求的变化也在一定程度上制约着软件开发的速度和质量. 新技术的不断出现和业务需求变化剧烈的融合, 导致了人们努力从实现业务需求的特定技术中分离出业务需求的模型. OMG 提出的 MDA(Model Driven Architecture) 模型驱动架构, 文[1] 提供了一种开放的、开发商中立的方法来建立强壮的可扩充的系统. MDA 使用建模语言代替编程语言来进行软件开发, 将软件系统的模型分离为平台无关模型(Platform Independent Model, 简称 PIM) 和平台相关模型(Platform Specific Model, 简称 PSM), 通过转换规则将它们统一起来, 并将 UML 作为 MDA 支持平台, 利用 UML 来建模文[2, 3] 从顶层元模型的角度比较了 UML1.1x 与 UML2.1.0 的不同, 采用声明式和命令式混合的模型框架, 给出了一种基于动作语义的 UML 模型转换方法, 并用 ASL 描述交互元模型的转换实例; 文[4] 将部分 UML 视图映射到形式语言 Z, 并用以描述面向对象的设计模式. 但这些工作大都是将部分 UML 视图形式化, 或者是针对软件设计某些阶段、某些方面的需要, 不能对整个软件开发过程提供支持. 文[5] 从基本概念、建模语言和支持工具等方面介绍了一种基于软件体系结构的、面向构件的软件开发方法))) ABC 方法. 该方法以 SA 作为系统开发的指导, 结合现在较为成熟的 CBSD 方法, 将构件运行平台作为组装和运行支持, 利用工具支持的自动转换机制, 提供了一整套从系统高层设计到最终实现的系统化的解决方案. 文[6] 给出了基于特征空间的构件模型, 研究了以特征空间分解为基础的规范化方法, 实现了多粒度、多模式构件的共存和构件间基于组合的松散耦合, 提高了构件的复用效率并降低复用成本. 构件作为一种资源, 更多的是体现了领域知识, 但是目前的构件组合技术并没有反映领域内的共性和可变性, 只是从构件组合方法上提出了不

收稿日期: 2007-12-03; 修回日期: 2008-07-03
基金项目: 吉林省重大项目/ 基于模型驱动技术的通信网络软件开发平台0(No.20060328); 国家科技攻关项目(No.2004BA907A20)

同的理论.

我们的策略是将基于构件的软件开发与领域模型集成起来, 提出了一种基于领域模型和构件组合的软件开发框架(A Frame based on Domain Model and Component Composition, 简称 FDMCC), 以支持软件开发全过程. 领域模型可以发掘业务需求中重要的领域概念, 并建立业务领域概念之间的关系, 通过领域模型, 可以清晰的表达业务对象之间的静态关系. 构件式软件系统设计方法通过构造或复用软件模块, 提供了一种自底向上的、使用构件构造系统的有效方法, 构件能够通过对自身的配置和组装来满足不同的业务需求, 极少影响到软件系统的变化, 简化软件开发过程, 提高开发效率.

本文第 2 节概括介绍了 FDMCC 框架结构; 第 3 节介绍了领域应用框架中包含的公共特性和可变性以及领域模型; 第 4 节详细构件组合的方法, 采用构件组合方式构造整个系统框架的过程; 第 5 节总结全文.

2　FDMCC 框架结构

FDMCC 以领域模型为基础, 结合领域系统中的共性、可变性等信息, 可以为基于领域模型和构件组合的软件设计与开发提供一种系统的、半自动化的方法支持. 整个系统软件框架如图 1 所示.

从图 1 可以看出, FDMCC 分为 3 部分, 左下半部分根据静态业务需求, 抽取领域概念, 建立领域模型, 并使用模型转换器把元模型和模型存储到数据库, 同时定义系统用户和角色, 并为用户和角色分配资源权限与功能权限; 右下半部分定义领域专用的领域应用框架以及创建各种类型的构件, 最后把领域应用框架和构件使用组合工具进行组合, 从而创建一个新的系统.

采用 FDMCC 生成应用程序步骤如下:

(1) 根据领域需求, 建立领域模型;

(2) 采用图形界面设计工具设计领域应用框架, 形成初始应用程序;

(3) 在领域模型基础上, 按照一个类对应一个类型构件的原则进行映射, 类中的属性都要映射成构件中的属性;

(4) 通过构件组合工具, 在初始应用程序基础上对构件进行组合, 形成完整的应用程序.

3　建立领域应用框架和领域模型

FDMCC 提供了可视化的领域应用框架设计功能, 可以根据用户的需求, 设计静态的领域应用框架, 同时具有代码生成能力, 生成的代码就是可执行的应用框架. 该框架具有可扩展的能力, 能够添加任何类型构件.

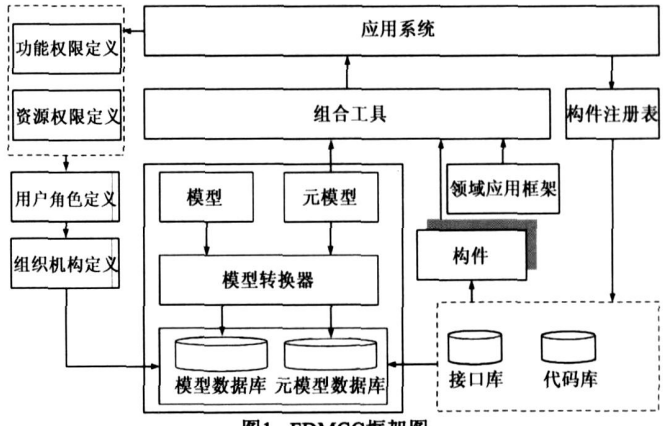

图1　FDMCC框架图

FDMCC 通过定义领域应用框架集中处理领域公共特性, 而可变性可以分为四类进行处理: 即数据的可变性、功能的可变性、界面可变性和业务过程可变性. (1) 通过在子类中定义新的属性, 并实现抽象类中的抽象函数可实现数据可变性. 数据可变性既包含在领域模型中的类模型中的类之间的继承关系, 同时也体现在具体的实现构件之间的继承关系(2) 功能可变性表现为领域应用框架可以根据运行环境的不同, 动态替换同一个功能在不同环境下不同的实现方法. 这可以通过更改 XML 配置文件来实现. (3) 界面可变性包括 2 方面: ¹ 同一应用系统的不同用户对同一操作界面要求不同展示风格 º 不同的应用系统在同一框架下展现的内容不同, 这 2 方面的可变性可以使用编织工具实现. (4) 过程可变性主要体现在需要为应用系统开发新的构件以实现框架没有提供的功能, 而且要将新构件集成到框架中, 形成新的业务过程. 新的构件可以通过框架中提供的功能集成进来. (1) 中的可变性采用领域建模技术和定制技术实现; (3) 和(4) 采用构件组合的方法实现.

例子: 在电信资源管理平台中, 网络拓扑的构建是资源管理的重要部分, 如何采用清晰直观的方式向网络管理员展示网络拓扑图是非常重要的, 而且拓扑管理是集成其它网络管理工具的基础. 为了仿真拓扑, 需要对网络拓扑进行分析和抽象, 如对于一个典型 IP 网络, 它可能包含各种型号的防火墙、各种型号路由器(如 Cisco 路由器、Juniper 路由器、华为路由器、中兴路由器等), 各种型号的交换机(如 Cisco 交换机、华为交换机、紫光交换机等) 和各种型号的服务器(如 Sun 服务器、IBM 服务器、HP 服务器) 等. 各种设备之间也包含多种连接关系, 如路由器间通过路由器的逻辑端口进行 IP 层连接, 路由器与交换机之间通过路由器物理端口进行链路层互相连接, 交换机与交换机以及交换机和主机之间也是采用链路层互联. 路由器可以通过交换机连接主机, 也可以直接与主机进行三层互联. 而且在

网络管理中, 要求呈现各种各样的视图, 视图可以按照网络种类划分, 也可以按照地理属性划分, 也可以按照设备间的关系划分, 因此不同的视图采用不同的模型来描述.

图2 领域应用框架设计界面

网络资源管理系统的领域应用框架主要由树、拓扑、表以及菜单、工具条等构件组成, 主界面以树、拓扑和表的形式呈现网络拓扑结构, 实时反应用户操作, 实现网络资源管理. 菜单、工具条是对框架的拓展, 相当于可变性的集成点. 同时, 在此框架中需要添加监听器, 实现树、拓扑图之间的互动. 领域应用框架设计如图 2 所示.

领域模型是系统的静态视图. 领域模型由元模型和模型组成, 元模型描述被分析系统的抽象表示, 以及这些抽象表示之间的关联关系, 模型依据元模型结构建立对象和对象之间的关系. 根据例 1 描述, 图 2 给出两种元模型, 一种是根据设备的地理位置显示设备之间的互联信息, 另外一种是显示所有设备的互联关系及接口. 这两个元模型分别从不同的角度描述了领域模型中的实体信息以及实体之间的关系, 称作元方面视图, 与此元模型相对应的模型视图称作方面视图. 可以采用定制技术把方面视图定制到领域应用框架中, 即在领域应用框架中选定与树关联的元模型. 一旦选定一个元模型, 就表示在这棵树中显示与此元模型相对应的模型, 即显示方面视图.

由于设计领域模型时并不考虑具体的数据库设计和实现, 因此, 领域模型能够分离数据模型, 指导后续

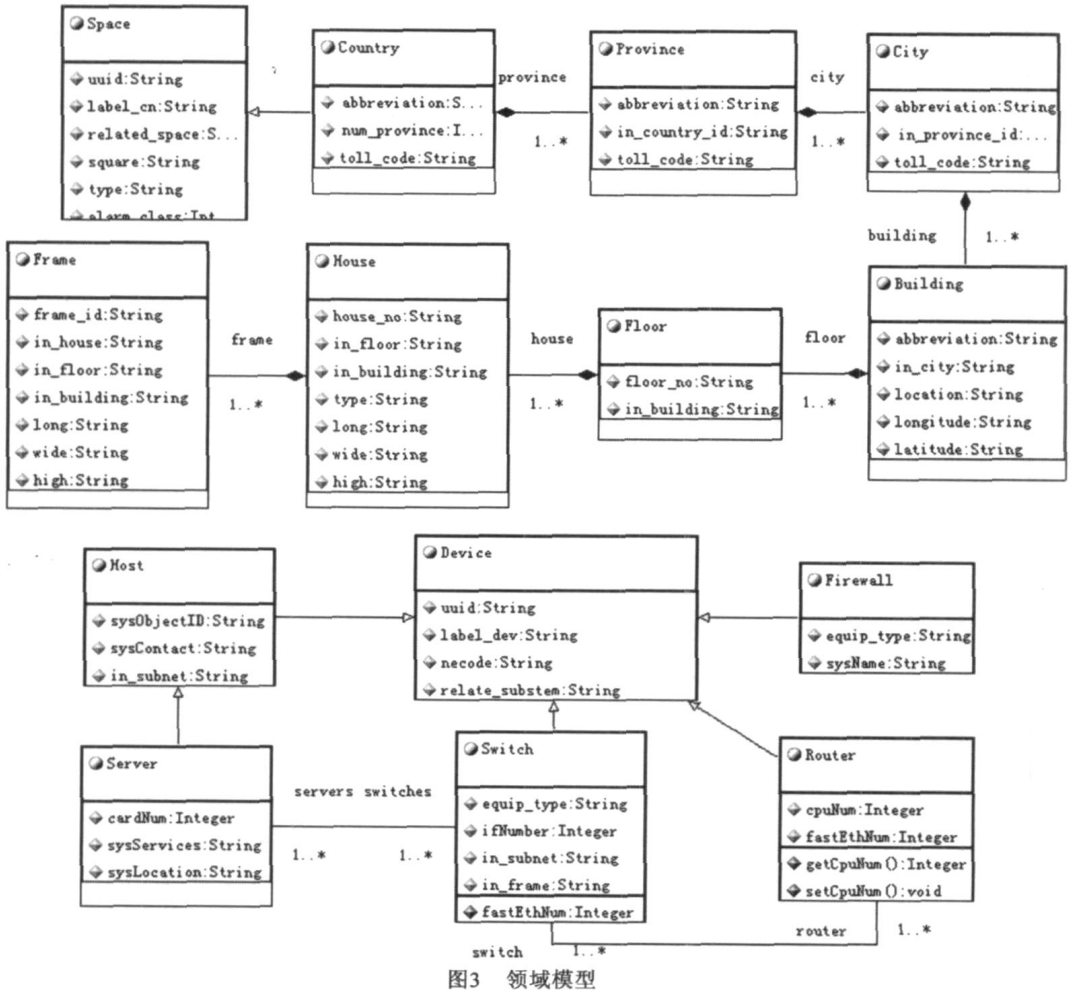

图3 领域模型

的数据库设计和开发.图 3 中的两种元模型可以采用模型转换技术存储到数据库中[7,8].

4　构件组合方法

构件组装的本质是,在构件之间建立关联,根据这种关联协调它们的行为,从而把它们组织成为一个有机的整体.为了更好描述构件组合过程,首先对构件以及构件的接口和功能给出形式化定义,部分定义参考文献[9].

定义 1　构件是系统的组成单元,它由构件接口和构件实现模块组成: Comp= (Intf, M, Attr, M Impl, Init),其中: Intf 是构件所有接口的集合;M 是领域模型中的类; Attr 是 M 中已声明类型的所有属性集合; MImpl 是 Intf 的实现,接口 Intf 中描述的操作的输入/输出参数是 Attr 中的元素; Init 是构件的初始状态.

定义 1 表明构件接口使用类中的属性作为输入/输出参数,从而使构件和领域模型进行了关联.构件接口可以直接使用类中的某个属性,也可以通过映射规则把接口中的参数映射成多个类中的属性计算后的结果.在 FDMCC 中,把构成领域应用框架的构件称为通用构件,而把与领域模型中类相关的构件称为类型构件.

通常一个构件包含一个接口的集合 Intf= ($intf_1$, $intf_2$, , , $intf_n$),每一个接口 $intf_i$ 是独立的,其内部属性和行为与其他接口无关.

定义 2　构件的接口定义为一个五元组 $intf_i$ = (ID, $Prov_i$, Req_i, $Behv_i$, Msg_i),其中 ID 是构件的标识; $Prov_i$ 是构件的第 i 个接口提供给环境或其他构件的功能的集合; Req_i 是构件的第 i 个接口运行所需环境或其他构件功能的集合; $Behv_i$ 是构件的第 i 个接口行为语义描述; Msg_i 是构件的第 i 个接口所产生消息的集合.

定义 3　构件 C 对外提供的功能集合

$PG\{Prov_i \mid \forall_i \# Prov_i \in Intf_i \cap Prov_i \subset Intf_i \in Intf\}$

构件 C 对外请求的功能集合

$R= G\{Req_i \mid \forall_i \# Req_i \in Intf_i \cap Req_i \subset Intf_i \in Intf\}$

在面向对象程序设计范例中,构件是作为一个对象来设计和实现的,每个构件都是一个可以单独部署的黑盒实体,它对外提供一些功能,同时也对外请求一些功能.这些功能之间的相互作用就是构件的组合问题.目前,构件之间的组合有 2 种方法:采用构件连接器和构件之间的调用.

定义 4　构件连接器实现了构件间的连接与交互,是一个四元组

Con= (ID, JoinPoint, Behv, Msg),其中: ID 是连接器的标识; JoinPoint 是连接器与构件的交互点的集合; Behv 是连接器行为的语义描述; Msg 是连接器的各 JoinPoint 产生的消息的集合.

定义 5　构件调用　两个构件 P 和 Q 若满足条件: $Prov(P) \cap Prov(Q) = \varnothing$, $Req(P) \cap Req(Q) = \varnothing$,且 $Prov(P) \cap Req(Q) \neq \varnothing$ 或者 $Prov(Q) \cap Req(P) \neq \varnothing$,则称它们是可调用的, Call(P, Q)= (Prov(P) ∩ Req(Q)) ∪ (Req(p) ∩ Prov(Q)).所谓的构件调用就是一个构件的对外请求正好是另外一个构件的对外提供的功能.

构件调用反映了构件接口之间的直接关系,它可以在同一个应用程序中的同一层次的构件相互访问时使用,也可以在相邻层次的构件间互相访问时使用.而构件连接器则反映了构件接口之间的间接关系,通过消息的传递实现构件组合,需要开发人员按照某种需求或逻辑重写构件之间的连接关系.

根据构件组合方法的不同, FDMCC 组合构件时分为两种情况:一种采用领域应用框架与构件之间的组合实现为领域应用框架添加新的构件与功能;第二种是采用构件连接器实现权限控制与不同系统间的协同.

11 领域应用框架与构件之间的组合

领域应用框架从宏观上描述了领域中的通用构件以及它们之间的位置关系,但是它不能描述通用构件之间的互动关系.在 FDMCC 中,要求树、拓扑和表之间有互动关系,即点击树的相应节点,拓扑图和表中应当显示树中该节点所包含的子节点的信息.同理,如果选中拓扑图中的某个节点,则树中和表中的相同节点也应该被选中.

A1 通过领域模型找到包含的展现构件

从领域模型集合 modelSet= (m_1, m_2, , m_n)中依次取出一个领域模型 m_i ($i= 1, 2, \ldots n$),读取该领域模型包含的各种关系和类;依照类选择类型构件,不同的类型选用不同的构件;在关系表中按照领域模型中描述的关系类别查找,如果对象之间是包含关系,则把这两个对象设置为父子;如果对象之间是关联关系,则需要在这两个对象之间生成新的链路对象;

B1 把类型构件中的属性与领域模型中类的属性进行映射,即把领域模型中类的属性通过接口操作赋值给类型构件中的某一个属性;并采用直接调用的方式,在各个通用构件(树、拓扑、表)中直接生成各种类型构件,然后把生成的类型构件放入到通用构件的各个数据源中;

C1 设置各个通用构件之间的互动关系.由于通用构件都放在一个框架下,因此 FDMCC 采用基于框架的协同模式,即通过框架来协同各个通用构件,每个通用构件都需要在框架上注册监听机制,并且在任意一个构件上的操作都会由框架发送给其他的通用构件,这

些通用构件根据监听到的动作进行相应的动作, 实现协同.

通过以上步骤, 就可以生成一个应用系统.

21 采用构件连接器实现构件组合

(1) 采用横切关注点进行构件组合, 实现权限控制

在网络管理拓扑呈现时, 不同的用户有不同的权限, 网络管理员拥有对所有对象的增删改查的功能, 而普通的用户, 对不同的对象有不同的权限, 有的对象可以查看, 有的对象不可以查看, 不可以查看的对象将不在拓扑图上显示, 有的对象还可以进行修改和删除. 因此在对每一个对象进行操作时, 都需要检查用户是否具有相应的权限. 由于对权限的判断涉及到各个类型构件, 因此我们称权限的判别问题是一个横切方面的关注点, 可以采用面向方面的方法解决. 本文采用面向方面的编程语言 JAsCo[10] 解决这个问题, 如图所示. 在 JAsCo 中, 需要定义一个 Aspect Bean 和一个 Connector, Aspect Bean 可以看成是对传统的 Java Bean 的扩展, 它具有可重用的特点, 相当于构件连接器的语义描述 be2hv, 不同之处就是添加了一个或者多个 hook 定义, 它相当于构件连接器中的 JoinPoint, hook 的构造函数在 Con2nector 实例化该 hook 时与具体的方法绑定. Connector 指定了 Aspect Bean 的运行环境, 是一个连接器. 在图 4 中, 当用户想要创建一个对象时, 需要查看该用户是否具有相应的权限, 如果有, 则继续执行, 否则抛出异常.

```
class AccessManager {
  PermissionDb pdb= new PermissionDb( );
  User currentuser= null;
  hook AccessControl {
  String exceptionmessage= / General Access Exception0;
  AccessControl(method(. . args)) {
    execution( method) ;
  }
  isApplicable( ) {
    return ! pdb. isRoot(currentuser) ;
  }
  around( ) {
  if( pdb. check(currentuser, thisJoinPointObject) )
    return proceed( ) ;
  else
  throw new AccessException( exceptionmessage) ;
  }
  void setExceptionMessage( String aMessage) {
  exceptionmessage= aMessage;
  }
  }
  }
    connector PrintAccessControl {
  AccessManager. AccessControl control=
  new AccessManager. AccessControl( void CreateDS. addElement( Vector v) ) ;
    control. setExceptionMessage( / Printer Access Exception0) ;
    control. around( ) ;
    }
```

图 4 使用 JAsCo 代码示例

同理, 用户在删除和修改一个对象, 也需要应用此方法.

(2) 采用消息机制实现客户端或不同系统间的协同

当有多个用户登录客户端时, 需要在这多个客户端实现协同. 因为用户的角色不同, 拥有的功能权限和资源权限也不同, 因此当有一个用户对资源信息进行修改时, 其他对该资源有查看权限的人员应当能够及时看到更新后的信息. 另外, 当故障系统采集到某一个设备故障信息后, 需要在客户端进行显示, 客户端的拓扑图构件需要根据故障信息找到相应的设备资源, 并且比较这个告警信息级别是否大于目前这个资源的告警级别, 如果大于等于, 则在拓扑图中显示此告警, 并把它添加到告警列表中, 否则只是把它添加到告警列表中. 不同客户端以及不同系统间的构件组合需要一个分布式的组合机制, FDMCC 采用消息机制实现, 通过在客户端实现消息监听可以实现它们之间的协同. 客户端把自己注册给一个消息主题, 消息主题负责把收到的消息传送到客户端, 客户端收到消息后, 根据消息类别和内容, 进行相应的动作处理.

5 总结

本文对领域软件的共性与变化性进行了探讨, 结合领域模型提出了基于领域模型和构件组合的软件开发框架. 在该框架的实现中, 领域模型使用 UML 类图表示, 用来捕获系统业务静态需求, 描述领域内业务对象之间的静态关系, 领域应用框架描述系统的共性, 同时提供反映领域可变性的扩展点, 在此扩展点上可以集成构件. 形式化的分析了构件以及构件的组合过程, 并将方面引入构件组合中, 研究了领域应用框架与构件的组合策略. 同时将构件与领域模型结合起来, 既能够利用模型的直观、方便的特点, 又能够重用已有的代码, 提高软件开发效率.

参考文献:

[1] MDA. Specifications. 2005. http://www. omg. org/mda/specs. htm# MDAGuide. [S]

[2] Mellor S J, Balcer M J. Executable UML: A Foundation for Model2Driven Architecture [M]. Addison2Wesley Professional, 2002.

[3] 陈秀红, 何克清, 何璐璐. 一种基于动作语义的 UML11X2210 模型转换方法[J]. 软件学报. 2006, 17(8): 1698-1706.

[4] Kim S K, Carrington D, Duke R. A metamodel2based transfor2mation between UML and object2Z[A]. In: Proc of the IEEE Symp. On Human2Centric Computing Languages and Environ2ments (HCC 2001)[C]. Stresa: IEEE Computer Society, 2001. 112- 119.

[5] 梅宏, 陈锋, 冯耀东, 杨杰. ABC: 基于体系结构面向构件的软件开发方法[J]. 软件学报. 2003, 4(4). 721- 732.

[6] 王忠杰, 徐晓飞, 战德臣. 基于特征的构件模型及其规范化设计过程[J]. 软件学报. 2006, 17(1): 39-47.
[7] OMG ptc/05-10-02, MOF QVT Final Adopted Specification. [S]
[8] An open source model transformation tool implementing the MOF 2.0 QVT-Operational language. http://smartqvt.elibel.tm.fr/index.html
[9] 刘静, 何积丰, 缪淮扣. 模型驱动架构中模型构造与集成策略[J]. 软件学报. 2006, 17(6): 1411-1422.
[10] Vanderperren W, Suvee D, Cibran M, Verheecke B, Jonckers V. Adaptive programming in JAsCo[A]. In Proceedings of AOSD 2005[C]. USA: ACM Press, Chicago, 2005.

作者简介:

王晓燕　女, 1977 年出生于吉林长春, 博士生, 讲师. 研究方向: 模型驱动技术, 构件技术, 形式化方法.
E-mail: wangxy@jlu.edu.cn

刘淑芬　女, 1950 年出生, 教授, 博士生导师, 研究方向: 计算机支持协同工作、软件体系结构、基于模型驱动的软件编程方法、网络管理技术.
E-mail: liusf@mail.jlu.edu.cn

张　俊　男, 1981 年出生于吉林省吉林市, 博士生. 研究方向: 模型驱动架构和系统架构.
E-mail: mail_of_zhangjun@163.com