

一种基于MDA的容错软件开发工具框架设计

崔红军, 邵培南, 严少清

(华东计算技术研究所, 上海 200233)

摘 要: 针对软件的高容错性、可靠性等要求, 该文给出一种基于模型驱动体系结构开发技术的软件开发工具的框架结构, 阐述了如何在工具中扩展对容错软件开发的支持。使用该工具可减少软件容错性描述的困难, 提高软件开发的效率。

关键词: 模型驱动体系结构; 容错软件; 统一建模语言

Design of Framework for Fault-tolerance Software Development Tool Based on MDA

CUI Hong-jun, SHAO Pei-nan, YAN Shao-qing

(East-China Institute of Computer Technology, Shanghai 200233)

【Abstract】 Aiming at the requirement of software such as high fault-tolerance and high reliability, this paper presents a framework for software development tool based on MDA, and describes how to extend the support to fault-tolerant(FT) software development. This tool can reduce the problems of the description of software fault-tolerance properties, and increases the efficiency of software development.

【Key words】 model driven architecture(MDA); fault-tolerant(FT) software; unified modeling language(UML)

当今软件变得越来越复杂, 开发人员发现软件开发仅使用代码的方法使得理解系统的关键特性及与维护系统的团队沟通变得更加困难, 并且生成的软件质量也不高。特别是对具有高容错性、可靠性等要求的软件开发面临着更大的困难。

随着现代软件工程技术的发展, 特别是统一建模语言(unified modeling language, UML)及一些支持 UML 的建模工具如 Rational Rose 的出现, 软件开发者能够在更高级别的抽象上完成工作, 通过这些工具工作在设计级别的团队向工作在实现级别上的团队提供设计模型, 实现团队转换这个抽象的设计成为详细的设计, 直至使用编程语言实现软件。但后来大家发现如果没有良好的纪律, 抽象模型和实现模型常常因为团队步调不一致而结束, 代码为了应付新增加的需求和新产生的想法而不断变化, 模型却一直停留在原地不动。为应对这些挑战, 以模型为中心的开发已成为当今软件业最热门的话题之一。

1 模型驱动体系结构

模型驱动体系结构(model driven architecture, MDA)是对象管理组织(Object Manage Group, OMG)提出的新方法学, 是一种新的系统开发方法, 它强调整个系统开发过程由对软件系统的建模行为驱动, 完成系统需求分析、架构设计、构建、测试、部署和运维工作^[1]。与传统的UML模型相比, MDA能够创建出机器可读和高度抽象的模型, 这种模型通过转换(transformation)技术可自动转换为代码、测试脚本、数据库定义以及各种平台部署描述。从此, 建模语言不仅仅是分析设计语言, 更可作为一种高级编程语言。

MDA通过抽象层次的不同, 定义了计算独立模型(compute independent model, CIM)、平台独立模型(platform independent model, PIM)和平台相关模型(platform specific model, PSM)^[2]。计算独立模型类似于人们常说的业务模型和

用例模型, 是一个抽象层次较高、独立于任何实现技术的系统模型, 它着眼于操作环境中的系统以及系统需求的描述, 而不关心系统本身的结构和功能实现细节; 平台独立模型类似于系统分析模型, 它处于中间抽象层次, 关注系统的整个架构实现, 但却忽略掉与平台相关的部分。平台独立模型可以转换成多个平台相关模型; 平台相关模型则与设计模型相像, 它把业务独立模型与具体使用平台的细节相结合, 包含了具体平台的特定实现技术。

2 基于模型驱动的软件开发过程

MDA一出现就受到业界的广泛关注, 目前这方面已有很多标准规范, 有越来越多的工具厂商对其提供支持。虽然使用不同的工具进行模型驱动的开发可能有所不同, 但基本的过程都是比较类似的。

本文描述的是一种基于模型驱动的、支持构件重用和组装的模型驱动的软件开发过程, 其根本思想是使用模型驱动框架的理论概念来指导软件开发, 并提供一种灵活的构件开发、重用和装配机制来满足不同领域的应用需求。之所以引入构件重用是因为构件技术在不同领域的应用开发中使用得日益广泛, 目前业界已经存积了不少针对某一领域应用的各类构件, 两者紧密结合更能提高软件的开发效率。该软件开发过程可概括如图1所示。

在介绍图1中的开发过程前, 首先要了解一下什么是元模型。开发人员使用建模工具进行可视化建模的前提是工具中已实现了其想建模的元素的元模型, 元模型就是能精确定义表示模型语言的模型。OMG提出的分层建模关系^[3]见图2。

作者简介: 崔红军(1974-), 男, 工程师, 主研方向: 软件工程, 软件测试; 邵培南, 研究员; 严少清, 高工

收稿日期: 2006-11-10 **E-mail:** cuihongjun@ecict.com.cn

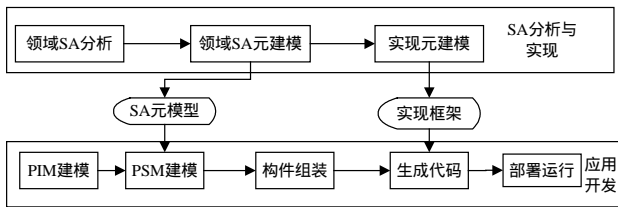


图1 基于模型驱动的软件开发过程

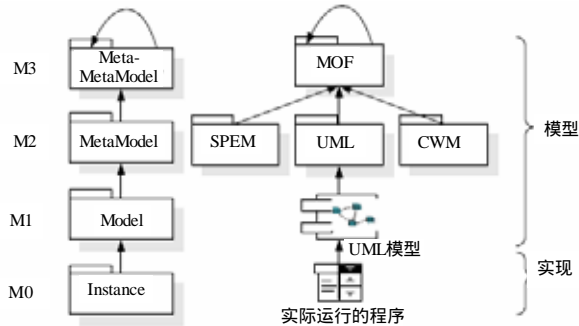


图2 OMG模型关系

从图2可看出,通常所见的Rose模型或其他UML工具创建的应用模型就是M1层。具体的应用系统,是位于最底层的M0层。而笔者所用来建模的UML规范等M2层的模型就是前面所称的元模型。位于最顶层M3层的元对象设施(meta object facility, MOF)作为一个公共语义总线是所有元模型之母,它本身是自描述的^[4]。用MOF可以创建你自己所需的元模型。MDA通过基于MOF的四层元模型架构不但能解决不同抽象层次上的不兼容的问题,而且可以在更大范围元数据的管理上发挥作用。

因此在图1中,针对不同应用领域的软件开发,首先利用对领域软件体系结构(software architecture, SA)的元建模,生成该领域体系结构元模型(如航空领域软件特有的高容错性等独有的元模型),开发人员使用标准或新构造的元模型进行领域应用的建模。建模完成后根据系统结构组成的语法和语义要求在构件库中寻找匹配的构件。当不存在符合要求的构件时,则需开发新的构件,或者将某些已有构件进行组装而得到满足需求的构件,并通过实现框架(即SA模型的实现)将模型转化为最终的应用软件,然后部署运行。

3 基于模型驱动的软件开发工具框架设计

为实现图1中基于MDA的开发过程,本文提出了一个基于模型驱动的软件开发工具的设计框架,见图3。

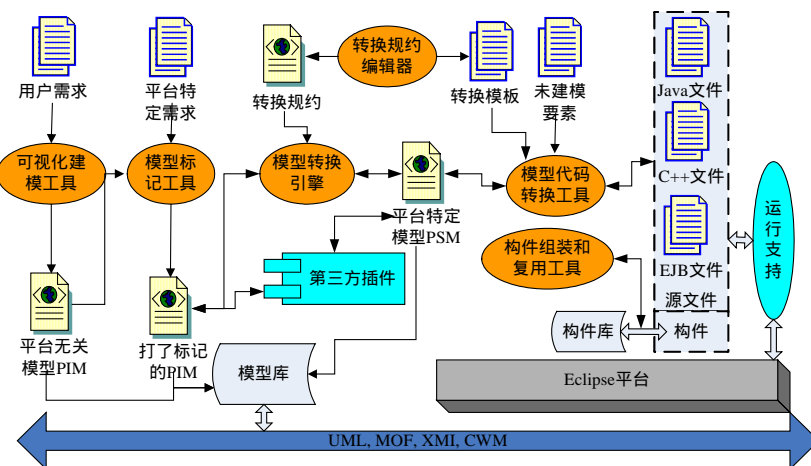


图3 基于模型驱动的软件开发工具框架

从图3中可以看出,MDA基础规范的UML、MOF、XML元模型交换机制(XML Metamodel Interchange, XMI)、公共仓库元模型(common warehouse metamodel, CWM)等作为工具的语义总线,工具产生的所有模型均保存在模型库中,模型的定义描述符合以上规范。保存在其中的模型可以通过XMI进行查询使用。

为提供最终的运行支持及动态的部署插件,工具选择运行在开放的、可扩展的Eclipse 3.0平台之上,以提供灵活的插件扩展机制。Eclipse的核心是动态发现插件的体系结构。作为一个集成开发工具,它能够将各种各样的开发工具集中到一个平台下。

工具其他部件及运行机制说明如下:

(1)可视化建模工具为模型的编辑器,模型可以在此构建和编辑以完成系统的需求分析和逻辑设计,生成平台无关的PIM。

(2)模型标记工具将平台无关的PIM根据平台特定的需求进行标记,为转换成PSM进行准备。

(3)PIM到PSM的转换可通过模型转换引擎或第三方提供的插件进行。使用转换引擎进行转换之前必须存在PIM到PSM转换的映射规则,工具会自动带有一些规约。如用户需要对规约进行变更可通过转换规约编辑器进行,转换定义可以在此构建和修改。

(4)PSM代码生成后通过模型和代码转换工具可从模型库中读取模型文件并生成最终的源代码(Java/C++/EJB)和构件,反之该工具也可读取文本形式的代码文件并将其以模型的形式保存到模型库中,供其他工具使用。代码完全由模型自动产生是不太可能的,生成代码时还要结合未建模的元素一起由人工辅助进行。

(5)如需要集成已有的构件或开发新的构件均通过构件复用和组织工具完成。构件、源文件产生后就可直接利用Eclipse平台提供的运行支持进行编译连接成可执行系统进行调试。

由于该工具涉及的关键技术非常多,特别是基于MDA软件开发的通用技术,如模型的定义方法,不同模型间的转换映射等技术;以及构件的定义、集成、组装技术等。这些技术在其他工具或资料中都有比较详细的描述,本文就不再重复。下面重点介绍该工具中如何针对容错性软件对元模型进行扩展来满足该领域软件开发的需要。在模型驱动的软件开发工具中除了前面的底层通用技术外,元模型的质量是工具成败的另一个关键。

4 工具对容错软件建模的扩展支持

标准UML规范里提供的建模元素都是很通用的,如用户想表达容错性方面的建模思想仅使用通常的元模型是不够的,所以必须对元模型进行扩展,即在MOF的定义基础上构造出容错技术相关概念的定义,这样用户在建模时才能够直接利用工具内提供的容错元模型进行建模,以弥补通用建模规范的不足。容错技术常用的概念包括:

(1)错误探测。不同的技术、工具支持不同的错误探测类型,如运行时检测、看门狗检测等;

(2)组。软件元素复制时需要标识组成复

制块的元素组;

建模工具可由 UML Profile 来定制一种适用于某个特定应用的语言。它是一个功能强大的扩展 UML 工具的基本元素。Profile 中定义的语言是 UML 的子集,看上去很像 UML。Profile 由一组版型、一组相关的约束和带标签的值构成。容错相关的 Profile 可包括 4 个元模型包来支持上述容错概念:

(2) 错误探测元模型包。包括错误探测的不同解决方案。

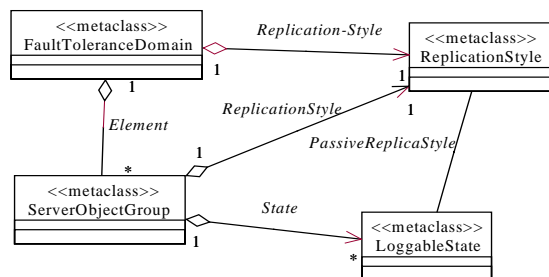
(4)复制类型元模型包。主要有主动、被动两种复制类型，不同的类型有不同的监控和状态检查方法。

```

graph TD
    A["<<metamodel>>  
Fault Tolerant Core"]
    B["<<metamodel>>  
Fault Detection"]
    C["<<metamodel>>  
Replication Styles"]
    D["<<metamodel>>  
Object Group Properties"]
    A -.-> B
    A -.-> C
    C -.-> D
  
```

由于错误探测、对象组属性相关概念描述比较简单，下面重点介绍容错核心元模型包、复制类型元模型包的设计。

容错核心元模型包的类图如图 5 所示。



对图 5 中的各类简要说明如下：

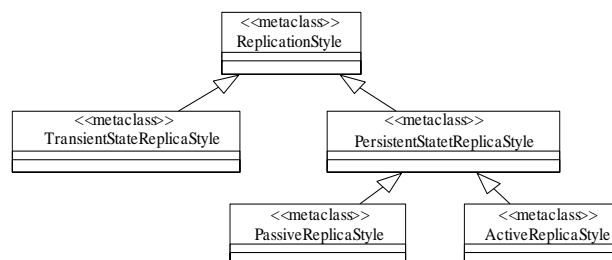
FaultToleranceDomain：该元类用于大型或复杂软件的容错处理，这种软件一般包含几台以上的主机，甚至更多。它决定了其下各对象组的缺省容错策略。

ServerObjectGroup：每一个容错对象的处理可能需要几

ReplicationStyle：该元类表示每个复制类型相关的信息。

LoggableState : 该元类定义了对象组内复制的重要状态。这些状态主要用于主复制、备份复制间的同步。

复制类型元模型包的类图如图 6 所示。



对图 6 中的各类简要说明如下：

TransientStateReplicationStyle :该元类代表了不会使对象发生永久的状态改变，只在短时间内有效地复制类型。与其对应的是 PersistentStateReplicationStyle 类型的复制，它又分为：

(1)PassiveReplicationStyle：该元类代表的复制类型就是大家平常较熟悉的主复制，即容错需要时唯一激活的复制类型。

(2)ActiveReplicaStyle：该元类代表了对同一个对象会同时激活几个复制的类型。

基于 MDA 的软件开发技术是当今非常受关注的研究领域, 本文不仅给出了一种基于模型驱动的支持构件重用和组装的软件开发工具的总体框架设计, 还重点讨论了该工具支持容错软件开发的扩展机制。使用该工具会大大减少用户在软件容错性建模方面的困难, 并能提高软件开发的效率。

- 1 刘发贵, 胡耀民. 基于 MDA 的模式化软件设计方法与应用[J]. 计算机应用, 2005, 31(4).
- 2 利用模型驱动开发降低复杂性[Z]. (2005-05). <http://www.ibm.com/cn/zh>.
- 3 OMG. MDA Guide Version 1.0.1[Z]. 2003-06.
- 4 OMG. Meta Object Facility (MOF) 2.0 Core Specification[Z]. 2003-10.
- 5 OMG. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms[Z]. 2004-06.

(上接第 277 页)

参考文献

- 1 Bergsten H. JavaServer Faces 交互式网站界面设计[M]. 福州: 东南大学出版社, 2006.
- 2 Kito D. Mann, Java Server Faces in Action[M]. USA: Manning Publications Co., 2006.
- 3 黎活明. EJB3.0 实例教程[EB/OL]. (2007-08). <http://www.foshanshop.net/EJB3.pdf>.
- 4 Mukhar K, Zelenak C, Weaver J L, et al. Beginning Java EE 5: From Novice to Professional[M]. U.S.A: Apress, 2006.
- 5 Keith M, Schincariol M. Pro EJB 3 Java Persistence API[M]. USA: Apress, 2006.