第33卷 第18期 计 算 机 工 程 2007年9月
Vol.33 No.18 Computer Engineering September 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)18—0270—03 文献标识码：A 中图分类号：TP311

一种领域建模工具的研究与实现

庞世春[1,2]，刘淑芬[2]
(1. 空军航空大学基础部，长春 130022；2. 吉林大学计算机科学与技术学院，长春 130012)

摘 要：讨论了领域建模工具GMT的设计和实现方法，对GMT支持的面向对象建模规范和软件开发范型作了简介，对GMT的系统结构和功能进行了详细的描述，并提出了一种面向对象的软件建模工具的实现方法。对GMT优点和缺点进行了总结，同时提供了可能的改进办法。

关键词：面向对象；元模型；建模工具

Research and Implementation of Specific Domain Modeling Tool

PANG Shi-chun[1,2], LIU Shu-fen[2]
(1. Base Department, Aviation University of the Air Force of China, Changchun 130022;
2. College of Computer Science and Technology, Jilin University, Changchun 130012)

【Abstract】 The research and implementation of GMT (generic modeling tool) is discussed. The object-oriented modeling standard and software development paradigm which the modeling tool GMT supports is explained briefly, the system structure and functions of GMT are expatiated, and a method of implementing object-oriented software modeling tools is presented. The advantage and disadvantage of GMT is discussed and a way of mend is provided.

【Key words】 object-oriented; meta model; modeling tool

1 概述

自20世纪90年代以来，面向对象的方法在计算机科学技术领域已经占据了主流地位。到1994年，已经出现了50多种面向对象方法和建模语言，开始了面向对象领域内的“方法之战”。3位著名的方法论大师Jacobson, Booch和Rumbaugh相互合作，综合各种方法的优点，实现了一种工业标准的建模语言——UML(unified modeling language)[1]，并且成为了事实上的面向对象建模语言国际标准，基本上结束了“方法之战”。但是UML在使用的过程中也暴露出来一些缺点。文献[2]中指出了这些缺点，包括：UML元模型很大且划分不良；对视点支持不佳；与业界“组件和模式”的发展潮流不同步；对关系的含糊；profile的限制。UML的这些缺点限制了它对问题的描述能力，比如：由于UML的元模型和表示法没有提供对视点的直接支持，无法允许建模者指明哪个或哪些视点包含了某个特定的模型元素，也就很难支持面向方面的特性，对于安全问题、日志、同步等横切关注点无法以标准方式进行建模，甚至大部分支持UML的建模工具根本就不支持视点建模。对关系的含糊体现在对聚合以及一般的整体局部关系的实际语义说得很少，它的姐妹标准MOF(meta object facility)不得不定义这些语义以定义对不同技术的映射[3]。事实上，语义的不清晰在UML中是普遍的。这就使得不同的建模者在使用UML建模工具建模特定关系时结果往往不同，妨碍了语义的表达。UML的标准扩展机制profile在扩展UML时受到很多的限制，由profile定义的新语言结构的类型机制以及定义这些结构之间的关系的能力非常有限，它既不能修改UML的元模型也不可以去掉UML元模型中的约束[4]。虽然UML在某种程度上支持定义新的建模结构，但是，当面对一个需要定制元模型的需要时，只能借助于MOF。基于MOF的元模型扩展机制在处理元模型时没有限制。MOF是定义建模语言的语言，包括定义UML。MOF还提供了访问和交换用建模语言定义的模型的方法。MOF弥补了UML的很多不足。事实上，基于UML还是基于MOF可以将目前流行的建模工具划分为两类。前者包括Rational Rose、MagicDraw UML和北大青鸟的JBOO等，后者包括GME等。基于UML的建模工具在支持面向方面的开发、支持元模型扩展、支持模型转换等方面能力不足。基于MOF的模型工具具有先天的优势，可以进行元模型的重型扩展，并且很好地支持了对视点的定义。GME基于MOF可以对建模语言进行建模，为建模语言指定清晰的语义，同时对视点的建模使得GME支持面向方面的开发。但GME仍然被定位于分析、设计的工具，而不是开发的工具，因而不支持模型可执行，模型的转换，模型的定制,这些特性是OMG提出的模型驱动框架(model driven architecture, MDA)的重要特征[5]。基于UML到基于MOF的模型工具的变化体现了UML的发展方向：抽象化，自动化，可执行化。

本文介绍了一种领域建模工具(generic modeling tool, GMT)及使用该建模工具进行软件开发的规范。GMT是支持MOF的面向对象建模工具。它主要吸取了GME的两阶段建模和视点建模的优点，形成了领域建模的架构。同时，配合通用数据采集工具(generic data collectiontool, GDCT)和电信通用对象呈现平台(telecom generic object platform, TGOP)可以进行基于进化的领域生命周期模型的网络管理软件的开

基金项目：国家科技攻关项目(2004BA907A20)；吉林省重大科技项目(20040304)
作者简介：庞世春(1975－)，男，博士研究生，主研方向：计算机网络，面向对象建模；刘淑芬，教授、博士生导师
收稿日期：2006-09-28 **E-mail：**psc1975@126.com

发。这种开发模式使得 GMT 同一般的进行分析、设计的面向对象建模工具区别开来。同时，GMT 提供的模型类似于 UML 的类图，描述的是系统的静态信息，将静态的模型通过定制工具关联到图形组件上，使得静态模型可以直接执行，这是朝着模型可执行方向作出的努力。

2 GMT 支持的概念与软件生命周期模型

GMT 面向对象软件开发规范通过两个阶段的建模：元建模和建模对系统进行描述。所产生的元模型和模型共同构成了系统模型。其中元模型包括 4 个视图：ClassDiagram 视图，Visualization 视图，Constraint 视图，Attribute 视图。每一个视图描述了系统的不同方面。见表 1。

表 1　GMT 开发中的视图及其构成

Class View	Visualization View	Constraint View	Attribute View
Class Diagram	Visualization Diagram	Constraint Diagram	Attribute Diagram

元建模由建模人员和领域专家共同参与。类图包含领域内的概念、关系的建模。图中可见性包含对于视点的建模，通过为元模型中的概念指定多个方面，使得建模时这个概念能够以多个指定的视点来呈现，并且提供了特定视点的访问控制。约束图中可以为特定的操作和约束建模，可以完整定义对模型的查询操作，还可以表达业务逻辑，包括动态触发规则、可以建模不变式、先验条件、后验条件。属性图中包括属性的建模以及为特定模型元素指定属性的过程。

在建模阶段，建模人员使用元建模时建立的领域语言来建模。由于领域语言中包含领域知识，GMT 在建模阶段强制验证领域约束，保证了建立的模型遵循领域知识。一个在领域内可重用的领域元模型和针对领域内一个特定系统的模型共同构成了对系统的完整描述。

基于 GMT 的软件开发采用了一种进化的领域生命周期模型。如图 1 所示。

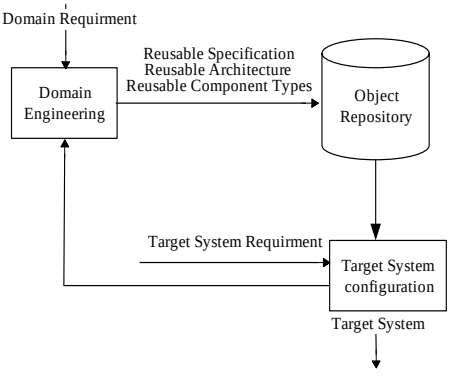

图 1　进化的领域生命周期模型

在领域工程阶段，领域工程师和领域专家通过对领域需求的分析，对领域知识进行元建模，形成领域语言，基于 TGOP 开发领域内可重用的框架、组件。TGOP 可以开发用于电信领域对象呈现的框架和各种组件，包括树组件、拓扑组件、表格组件等用来呈现各种不同的对象、属性、拓扑结构。 在应用工程阶段，应用工程师使用领域建模语言对具体应用进行建模生成模型，模型和模型的元模型构成具体应用的完整描述。在应用定制阶段，使用 GMT 的定制功能，应用工程师根据目标系统的需求，将底层数据采集工具 GDCT 采集来的数据形成的数据源定制到不同的对象模型、模型属性或者是模型关系上，再将模型定制到框架中的不同组件上，从而形成了一个数据从设备采集到形成数据源，由数据源到模型，由模型到界面上的不同组件这样一个完整的数据流路径。

3 系统结构与功能

GMT 由一系列的子工具组成，它们提供了绘图、导航、存储、多用户支持、一致性维护、约束的强制检查等功能，支持领域工程和应用工程，如图 2 所示。

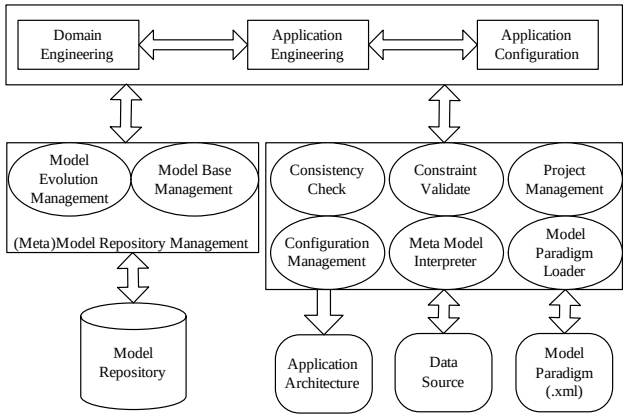

图 2　系统的功能结构

GMT 主要完成 3 大功能：元建模，建模，定制。元建模对领域知识进行抽象，建立领域元模型，形成领域知识。领域知识包括领域概念、概念间关系、领域约束。建模建立遵循领域知识的模型。定制包括将数据源定制到模型和将模型定制到领域框架和组件。这些功能是通过一个松散耦合的工具集来完成的。

(元)模型仓库管理工具负责维护元建模和建模阶段生成的元模型和模型，完成模型管理和模型演化管理。模型的管理包括按照(元)建模语义将模型入库和从库中提取模型。模型演化管理包括元模型更新时，模型也要作相应的更新以遵循新的元模型。

约束验证工具负责对建模过程中的约束和一致性问题进行验证。一致性问题包括：对命名空间、访问控制、结构关系等进行检查。约束验证在建模时验证建模过程是否遵循元建模建立的约束。工程管理工具以工程的方式维护元模型和模型。元模型解释器解释元模型为模型范式，范式装载器负责将模型范式加载到建模环境中，建模要遵循该模型范式，通过这种方式，实现了模型遵循元模型。

模型范式装载器负责将(元)模型范式装载到建模环境，元模型解释器负责将元模型解释为模型范式，元模型范式是手写的一个范式，用来约束元建模行为。这样就形成了元模型范式约束元建模，元模型解释为模型范式，模型范式约束建模的完整的两阶段建模过程。

定制管理工具负责将模型定制到领域工程建立的组件和应用框架以及将数据源定制到模型，数据源包括文件、数据库以及从端口采集的数据。

4 系统的实现

GMT 是一个多用户分布式建模工具，采用标准的 3 层 J2EE 架构实现两阶段建模。如图 3 所示。

定制管理工具负责将数据源定制到模型和将模型定制到领域框架及组件。在模型中，数据以属性的形式存在，将数据源定制到模型的过程就是指定从端口采集的数据、从文件或数据库中读取的数据与某个对象的某个属性的对应关系。领域框架是领域工程时创建的，可以在领域内重用。领域框

架中包含一系列的组件，包括树组件、表组件、拓扑组件。树、表、拓扑是 MVC 结构中同一个 model 的多个 view，3 种组件可以相互导航。从模型到框架和组件的定制包括指定模型中一个对象呈现时的图标，指定在拓扑中图标间的连线与某种关联关系的对应，指定树组件的根节点与聚合关系树中的某一点开始的子树的根的对应关系。通过这种方式，可以实现静态信息的呈现，具体的动作采用 Java 中的 Reflection 机制在图标上挂接一个字节码文件(或其他的可执行文件)来实现。

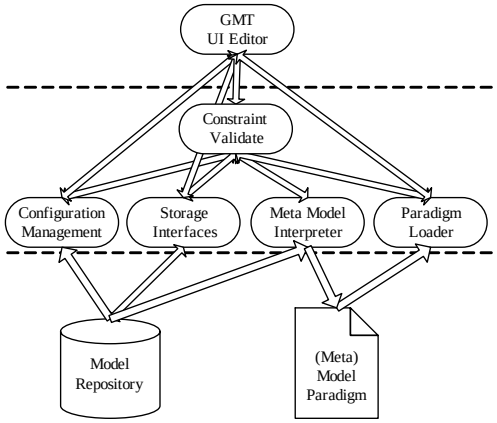

图 3　系统体系结构

元模型范式是通过 MOF 对 UML 的扩展得到的，通过 MOF 的扩展可以使用 MOF 提供的面向对象建模的全部语义能力，比如可以在构造型间声明新的关联。在 GMT 的元模型范式中，共扩展得到了 15 种模型元素，22 种模型元素间关系和若干种属性。这些模型构件可以描述元模型中所有面向对象的建模语义，包括建立类和类间的关系如关联、聚合、继承关系，还可以描述元模型中的视点建模以及属性建模。图 4 是通过 MOF 扩展 UML 元模型得到元模型范式的片断。

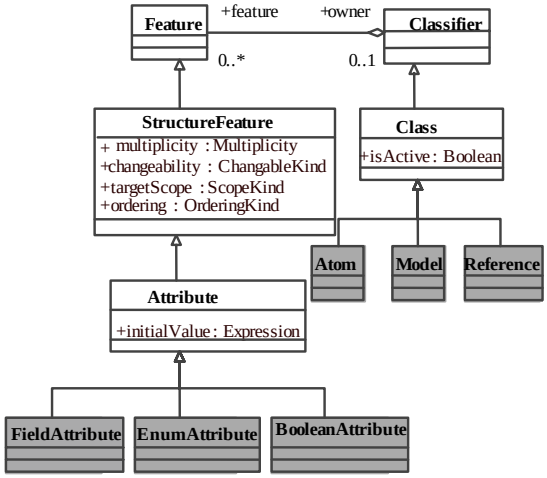

图 4　通过 MOF 对 UML 的扩展

通过继承 UML 元类 Class 和 Attribute 得到了一些模型元素和一些属性。比如 Atom、Model、Reference 等。其中一个特别的模型元素是 Aspect，通过引入该模型元素以及该模型元素同其他的模型元素之间的关系，实现了对视点的建模。

元建模人员在建模一个复杂结构时可以同时建模该复杂结构的内部视图，同时提供了对不同视图访问控制的建模。这种机制可以用来建模那些同构成系统的多个模块甚至是全部模块有关的问题(称为横切关注点)。比如安全、日志、同步等。传统的面向对象建模工具无法对这类问题进行建模。

元模型解释器解释元模型生成模型范式。这种解释过程遵循一套解释规则，这种规则保证了建立的领域词汇、词汇间关系、领域约束采用同元模型范式相同的存储结构来保存。存储接口按照元建模和建模的语义对(元)模型仓库进行存取操作。包括创建、删除、修改模型元素、元素间关系、视点、属性、约束。约束验证器会被多个工具使用以进行一致性检查和约束验证。一致性检查分为即时型一致性检查和延后型一致性检查。对于即时型一致性检查要求在建模动作发生时进行实时检查，延后型一致性检查则允许在其后的某个时间点开始进行。

5　结束语

基于进化的领域生命周期模型和 GMT，构建网络数据管理软件平台。这种软件架构为网络管理软件的开发提供了新的思路。不是把开发的关注点全部放在领域内一个具体的应用，而是首先关注领域内产品族的共性，这种共性经过抽象体现为领域元模型和领域框架及组件。领域元模型、领域框架及组件在整个领域内可以重用。基于 GMT 的定制功能使软件可以灵活适应需求的变化，在定制的基础上少编码甚至是不编码同样可以适应需求的变化。在一般的软件开发过程中，模型是分析、设计的结果，是一种辅助性的成果， GMT 将模型变成最终的软件产品的一部分，开发人员不必再像以往一样费力地去维护文档和代码的一致性，简化了开发过程。这种软件架构具有一般性，可以应用于其他的领域。

GMT 还可以作为分析设计的工具在 RUP 软件开发过程中使用，GMT 使用领域语言建模的方式使模型构件语义更丰富、更明确，防止在分析、设计过程中产生二义性。GMT 的元模型中包含领域知识，并且强制验证建模过程，保证建模过程中建立的模型是遵循该领域知识的，这种方式保证了分析、设计的结果是满足领域约束的，避免出现分析、设计上的逻辑错误。GMT 的主要问题在于：(1)模型交换的问题。GMT 的(元)模型采用私有的结构保存在数据库中。下一步考虑采用 XMI 来保存模型；(2)在模型可执行问题。GMT 中的模型可执行仅仅是静态模型的可执行，要实现真正意义上的模型可执行还有很大的距离。

参考文献

1 Booch G, Jacobson I, Rumbaugh J. The Unified Modeling Language User Guide[M]. [S.l.]: Addison-Wesley, 1999.

2 Frankel D S. Apply MDA to Enterprise Computer[M]. [S.l.]: Posts & Telecom Press, 2003.

3 OMG. MOF 1.4 Specification[Z]. (2004-10-02). http://www.omg.org/cgi-bin/doc?ptc/.

4 OMG. UML 2.0_Superstructure[Z]. (2005-05). http://www.uml.org/#UML2.0.

5 OMG. MDA Guide Version 1.0.1[Z]. (2005-04). http://www.omg.org/omg/03-06-01.