ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1317−1327
DOI: 10.3724/SP.J.1001.2008.01317

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

元建模技术研究进展*

刘 辉[1,2], 麻志毅[1,2], 邵维忠[1,2+]

[1](北京大学 信息科学技术学院 软件研究所,北京 100871)

[2](北京大学 高可信软件技术教育部重点实验室,北京 100871)

Progress of Research on Metamodeling

LIU Hui[1,2], MA Zhi-Yi[1,2], SHAO Wei-Zhong[1,2+]

[1](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Key Laboratory of High Confidence Software Technologies of the Ministry of Education, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: shaowz@sei.pku.edu.cn

Liu H, Ma ZY, Shao WZ. Progress of research on metamodeling. *Journal of Software*, 2008,19(6):1317−1327.
http://www.jos.org.cn/1000-9825/19/1317.htm

Abstract: With the popularity of UML (unified modeling language) and MDA (model driven architecture), models are becoming the core artifacts of software development and maintenance. As a result, modeling languages and meta-models which are used to define modeling languages, become more and more important. Software development may cover quite a few domains, and different domains may require different modeling languages and their supporting modeling tools. But it is very expensive to develop modeling tools manually for every domain. Metamodeling is one of the technologies to facilitate the design of domain modeling languages and the development of modeling tools. In the approach of metamodeling, people design domain modeling languages according to domain request by metamodeling. And then, metamodeling tools automatically generate modeling tools, which support the designed domain modeling languages. As shown by experimental results, metamodeling, combined with MDA, can increase productivity of software development. This paper makes a survey of the current research on metamodeling, compare metamodeling tools, and discuss further directions of metamodeling and their supporting tools.

Key words: metamodeling; metamodel; UML(unified modeling language); MDA(model driven architecture)

摘 要: 随着UML(unified modeling language)与MDA(model driven architecture)的兴起和流行,模型已经成为软件开发的核心制品,而模型重要性的提升使得建模语言以及定义建模语言的元模型逐渐成为软件开发中的一个核心要素.软件开发往往涉及多个领域,而不同的领域往往需要不同的建模语言及其建模工具.但是,手工地为不同的建

* Supported by the National Natural Science Foundation of China under Grant No.60473064 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA010301, 2005AA112030 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321805 (国家重点基础研究发展计划(973)); the Key Technologies R&D Program of China under Grant No.2003BA904B02 (国家科技攻关计划); the National Key Technology R&D Program of China under Grant No.2006BAH02A02 (国家科技支撑计划)

Received 2007-04-12; Accepted 2007-08-03

模语言开发建模工具代价高昂.元建模技术是解决这个问题的方法之一,通过元建模,可以根据领域需要定制合适的元模型以定义领域建模语言,进而自动生成支持该建模语言的建模工具.大量的工程实践表明,与领域建模以及MDA相结合,元建模可以大幅度地提高软件开发效率,基于元建模的MDA比基于通用建模语言的MDA更具潜力.在最近的几年中,元建模及其相关技术发展迅猛,不但在技术上取得了长足的进步,而且在产业界也开始出现大规模的商业应用.总结了元建模的现有研究成果,分析和比较了现有元建模工具,探讨了元建模的可能发展方向.对元建模中存在的问题进行分析,并指出了可能的解决途径.

关键词: 元建模;元模型;UML(unified modeling language); MDA(model driven architecture)

中图法分类号: TP311 **文献标识码:** A

统一建模语言 UML(unified modeling language)[1,2]和 MDA(model driven architecture)[3]的流行,使得模型成为软件开发的核心制品,提升了软件开发的抽象层次,从而提高软件开发效率和软件的可维护性[4].但是,因为 UML 作为一种通用建模语言而过于通用、复杂,现在还难以从 UML 模型生成完整的应用系统.到目前为止,UML 建模工具,包括最常用的 Rational Rose[5],IBM Rational Software Architect[6],Enterprise Architect[7],JBOO[8]等,所能生成的代码都还是程序框架而已,与生成完整应用系统还有相当的差距.其次,UML 作为通用建模语言,其所包含的大部分概念(比如类、对象、属性和操作等等)都来自面向对象编程语言,而不是应用系统的问题域.这导致 UML 中的概念与问题域中的概念存在一定的差距 [9].再次,UML 虽然是通用建模语言,但是它也并非万能,不同的领域可能需要不同的建模语言.所以,人们通常根据应用系统(领域)的需要建立自己的建模语言及其建模环境[9,10].这种建模语言称为领域建模语言(domain specific language,简称 DSL).

手工为不同的建模语言开发建模工具代价高昂,所以人们需要某种技术以降低建模工具的开发成本.元建模技术是解决这个问题的方法之一,其核心思想是由领域专家按需定制领域建模语言,然后通过代码生成技术自动获得支持该领域建模语言的建模工具.大量的工程实践表明,基于元建模的领域建模要比基于统一建模语言 UML 的效率高出 10 倍[9].采用 DSL 也使得软件的维护变得更为简单[11],所以,基于元建模的 MDA 比基于单一建模语言 UML 的 MDA 更有潜力.基于元建模的领域建模之所以如此有效,主要得益于如下了两个优点:首先,领域建模语言贴近问题域.对领域内的设计人员而言,领域建模语言更容易学习和使用;其次,因为领域建模语言被限定于特定领域,所以,从领域模型自动生成程序代码相对比较容易.

到目前为止,元建模主要有图 1 所示的两种框架结构(framework).

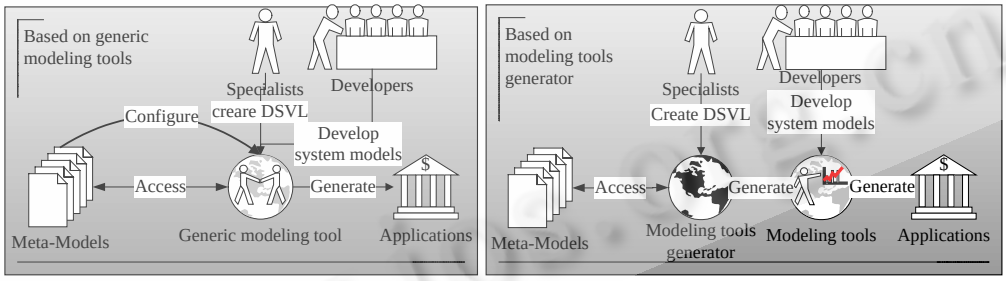

Fig.1 Two kinds of Metamodeling frameworks

图 1 元建模的两种框架结构

最早的元建模以通用建模工具为核心(如图 1 中的左图所示).首先,领域专家通过通用建模工具建立一个元模型(meta-model),用于刻画某种建模语言;然后,将建立的元模型用于配置通用建模工具,使得通用建模工具支持该元模型所刻画的建模语言.也就是说,通过元模型的配置,通用建模工具就成为该元模型所刻画的建模语言的建模工具.通用建模工具也称为通用建模环境(generic modeling environment),它可用于建立元模型(以元元模型配置通用建模工具),也可以用于建立模型(以元模型配置通用建模工具)[12].

最近兴起的另一种元建模方式则不包含通用建模工具(图 1 右边所示).元建模的第一步是通过建模工具生

成器建立元模型以刻画建模语言,但是它并不生成通用建模工具的配置文档.它直接生成支持该建模语言的建模工具.这两种方法各有利弊,详细比较参见第 2.1 节.

目前,元建模还没有一个标准的定义.从前面的描述中可以看出,元建模主要包含两个要素:① 建立元模型以刻画某种建模语言;② 提供(通过配置或直接生成)支持该建模语言的建模工具.所以,我们对元建模给出如下定义:

定义 1(元建模). 建立用以刻画某种建模语言的元模型,并提供支持该建模语言的建模工具.

元建模工具则是支持元建模的辅助开发工具.它可以用于设计建模语言,同时还可以辅助实现支持该建模语言的建模工具.

本文总结元建模的研究热点和进展,分析、比较现有元建模工具,探讨了元建模的可能发展方向.本文第 1 节按照元建模技术中的 3 个关键要素(元元模型、元模型、建模工具集成)分类介绍有关元建模技术的研究热点及其进展.第 2 节从工具的角度介绍和评述元建模技术的相关研究和应用状况.第 3 节总结、评述元建模技术,指出存在的问题并对后续研究提出建议.

1 元建模的各要素及当前研究热点

根据元建模过程中的各个活动及其所涉及的要素,对现有的研究热点进行分类介绍.现有的研究主要集中在元元模型、元模型构造和建模工具集成 3 个方面.

1.1 元元模型

描述元模型(MetaModel)需要元语言的支持,而元语言是通过元元模型(meta-MetaModel)刻画的.所以,要构造元建模工具,首先必须确定元元模型.当前,元元模型可以大致分为如下几类:

1.1.1 基于 E-R 图的元元模型

在 OO 流行之前,E-R 图是最好的建模语言之一.即使到现在,E-R 图也依然是数据库建模的主流语言.所以,早期的元建模将 E-R 图作为元元模型是很自然的事.MetaView[13]基于 E-R 图扩展提出 EARA/GE(entity aggregate role attribute with graphical extension),并以此为元元模型;ToolBuilder[14]以 E-R 图为基础提出扩展的 ER 模型 EER(extended entity relationship).AToM3[15]直接使用 E-R 模型.基于 E-R 扩展的元元模型,其优点是简单、直观,而且与当时的主流建模语言 E-R 图的特殊关系,也有利于元建模工具的推广.

1.1.2 基于 MOF 和 UML 类图的元元模型

随着面向对象建模方法逐渐成为主流,人们开始采用面向对象的方法来刻画元模型,其中具有代表性的是 MOF(meta-object facility)[16]和 UML 类图(UML class diagram).在 OMG 的 MDA 框架中,MOF 和 UML 分别处于元元模型和元模型层.从这个意义上说,UML 类图不是元元模型而是元模型.但是,UML 本身是自描述的,其中用于描述 UML 本身的一个内核经过简单扩展(增加诸如自反射等功能)就成为元元模型 MOF.所以从这个意义上说,MOF 是 UML 的一个子集,而且这个子集正好和用于描述类图的 UML 子集近似等价.

基于 MOF 和 UML 类图的元元模型已经成为主流.MOMO[17]直接支持 MOF.EMF 和 GMF 的元模型 ECore 是 MOF 的 Java 实现,它根据 Java 语言特性对 MOF 适当简化并增加 Java 语言特性,比如 Java 基本类型.MEG[18]和 GME[12]则是基于 UML 类图的代表(MEG 随后又把类图转换为类型图 type graph.但是对用户而言,类型图是不可见的).

基于 MOF 和 UML 类图的元元模型因为采用 OO 技术而自然、直观,而且表达能力比 E-R 图强.同时,采用主流建模语言(UML)的子集作为元元模型可以最大程度地减小使用者的学习负担.采用 UML 子集的另外一个好处是可以直接使用现成的 UML 建模工具来建立元模型.

虽然 MOF 是 OMG 推出的标准元元模型,但是因为历史的原因(许多主流元建模工具都具有比 MOF 更为悠久的历史)以及技术原因,到目前为止,MOF 还未能结束元元模型标准之争.首先,MOF 主要考虑的是元模型的构造以及数据集成(模型交换),而不是元建模工具,所以,元建模工具实现上的许多问题它都没有考虑.比如,MOF 没有对建模语言表示法建模的能力;其次,MOF 也无法刻画建模语言的语义(UML 的语义主要靠自然语

言和 OCL 语言刻画).MOF 也没有考虑诸如图(diagram)、边(edge)与点(note)的区分,而这些信息对于工具构造却是必不可少的.这使得建模工具的自动生成尤为困难.

1.1.3 基于图的元元模型

基于可视化建模得到的元模型和模型通常都以图(diagram)的形式表示(比如 UML 元模型、UML 模型、ER 模型等),所以图可以作为表示各种模型、元模型的通用描述语言.Moses[19]以类型图(type graph)作为元语言,以带属性的图(attributed graph)来表示元模型.同样,采用图(graph)以描述元模型的 AToM3[20]呈现给用户的是一个 E-R 模型,系统会自动将 E-R 模型转换为图.基于图(diagram)的优点是具有坚实的数学理论基础,缺点是比较抽象.以图(graph)的形式描述元模型(或模型)并没有 OO 的方式直观,所以,通常图和其他(比如 E-R)形式一起使用:呈现给用户的是直观、易懂的 E-R 模型,系统自动转换为易于计算机处理的图描述.

基于图(graph)的元语言最大的好处之一是可以用通用的形式化的图转换来定义建模语言的语义[20],这为自动生成建模工具提供了极其有利的条件.同时,借助图转换理论可以对建模语言的语义进行推理和证明,以便及早发现建模语言的设计缺陷.

1.1.4 其 他

以上 3 类之外的其他元元模型一般是在开发元建模工具的时候根据实际需求(包括与实现相关的技术约束)设计的.最具代表性的是 DOME[21]和 MetaEdit+[22,23]所提出的元元模型.DOME 提供的元元模型是 DOME 工具规约(DOME tool specification).从名字就可以看出,它已经考虑到了工具实现方面(而 MOF 则几乎没有考虑工具实现的问题).在 DOME 工具规约(元元模型)中,最核心的概念是 Node(节点)和 Connector(连接子).此外,DOME 还包括 Graph(图)、Menu(菜单)和 MenuItem(菜单项)等直接与工具相关的元素.

MetaEdit+提出 GOPPRR(graph-object-property-port-role-relationship)作为描述元模型的基础.Object(对象)、Property(属性)与 E-R 模型以及 UML 中的相关概念相似.Graph(图)则对应 UML 中的 Diagram(图),表示模型的某个视图.Graph 的意义在于模型的组织与建模工具的设计:如何从不同视角刻画和展示模型.Relationship 与 DOME 的 Connector 类似.

需要强调的是,MetaEdit+的 Relationship(以及 DOME 的 connector)与 MOF 的 Relationship 是属于不同层次的概念.图 2 是 UML 元类 Generalization 的定义,其中,元类 Generalization 和 DirectedRelationship 之间的继承关系是 MOF Relationship 的一个实例,而元类 Classifier 和 Generalization 本身则是 MOF Classifier(而不是 MOF relationship)的实例.所以,如果输入这样一个 UML 元模型,则元建模工具无法确定 Generalization 实例应该是一条边(edge),而不是一个节点(note).DOME 将 UML Generalization 作为 Connector 的实例(如图 3 所示),从而在元模型层上将两者区分(MetaEdit+类似).能够自动生成图形化建模工具的元建模工具基本上都引入了类似概念以解决这个问题,比如,GME 引入 Connection.EMF 没有采用类似概念是因为它只生成树形建模工具,此时并没有边与节点的区别.GMF 则是基于 EMF 的,所以也无法采用 Connector 的概念.它采用的补救方法是在定义表示法的时候指定元模型元素是 Node 还是 Connector.

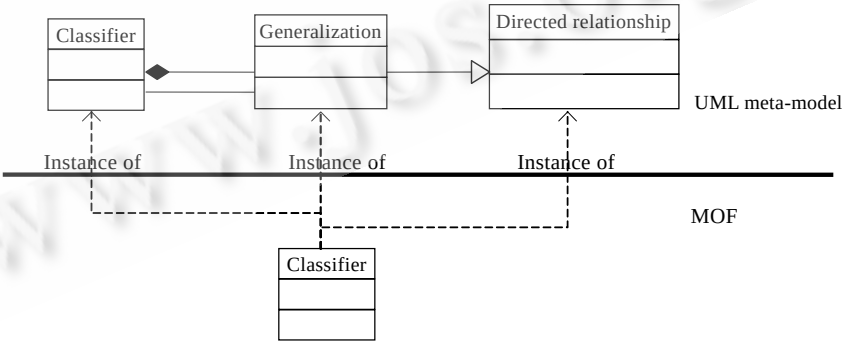

Fig.2 UML relationship and MOF classifier

图 2 UML 的 Relationship 与 MOF 元类的关系

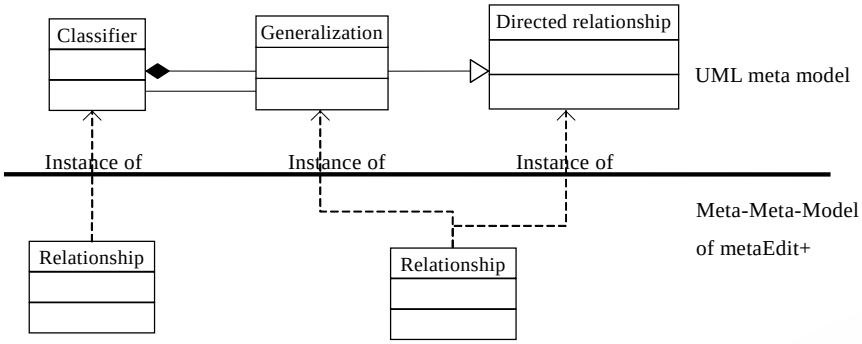

Fig.3 UML relationship and MetaEdit+ metamodel

图 3 UML 的 Relationship 与 MetaEdit+元元模型的关系

Zhu 和 Shan[24]在建模语言的结构、类型系统和建模语言对模型的一致性、完备性的定义和检查等问题上展开研究.在此基础上,他们提出了一种扩展的 EBNF 和一阶逻辑作为描述建模语言的元语言.Cesar Gonzalez-Peter 和 Brian Henderson-Sellers[25]则针对软件开发方法(software development methodologies)探讨了通过 OMG 四层体系结构方式进行元建模的一些缺陷,并提出了使用基于幂类型(powertype)的元建模框架.

1.2 元模型

作为元建模的核心制品,元模型的构造、演化和度量都至关重要.

我们关注的首要问题是元模型的构造.Tolvanen 和 Kelly[26]根据 23 个实际案例归纳出元模型元素的 4 个主要来源:领域专家或开发人员提出的概念、生成目标、待建系统的外观以及产品线变化空间.领域专家提出的概念是领域事物的一个直接映射.这是一个相对初级的阶段,表示此时还没有可以复用的代码库或者生产线产品.以生成目标定义元模型元素的典型例子是统一建模语言 UML.UML 的大部分概念都来自面向对象编程语言,而以面向对象编程语言描述的源代码正是 UML 模型的生成目标[26].如果通过待建系统的外观(比如手机的不同按键)就可以判定系统的功能,那么此时适合第 3 种方式(待建系统的外观)构造元模型.所谓系统外观,主要是指系统与用户的交互设备,比如键盘、按钮等.最后一种方式最为有效,也最为复杂:通过分析现有的产品线并估计将来可能发生的变化,总结出该领域内各个产品的变化空间并提供元模型元素以描述该空间.只有在领域相对较小、产品线比较成熟的时候,才可能由经验丰富的专家成功地预测到产品线的变化空间.

微软公司的 Pavel Hruby[27]则提出可以根据领域内的 Ontological Categories(本体类目)推导出元模型所需要的概念.

元模型演化是另一研究热点.当元模型所刻画的领域发生变化时,最简单的应对方法是重新构造一个新的元模型.为了尽可能地减少重新构造的工作量,Kalle Korhonen[28]希望可以通过分析原来的元模型以确定哪些地方是可以复用的.他把设计和实现领域建模语言及其支撑工具的工作分成 3 个层次:语言层、生成器层和构件框架层.语言层主要是元模型,复用比较简单;代码生成器是用于将模型转换为源代码的模型转换器,是 MDA 的核心所在.代码生成器的构造是开发建模工具中最为艰难的部分,而且难以复用.所以,Korhonen[28]认为代码生成器应该尽可能地简单,把大部分可以下放的功能都放到框架中去;而框架的可复用性取决于框架的模块化程度.

处理现有模型(构造在旧的元模型之上)是元模型演化的另一个难点.手工修改或者重新用新的元模型构造一个等价模型虽然不是不可行,但是当旧模型数量巨大时,手工修改就显得过于笨拙,而且手工的修改也难以保证新旧模型的一致性.更好的选择是采用模型转换的方式将旧模型自动转换到新的元模型上.Jonathan Sprinkle 和 Gabor Karsai[29]设计了一种专门用于模型演化的模型转换语言.这个基于图转换的可视化语言可以方便地描

述新旧模型之间的转换规则.之后,工具可以根据转换规则自动生成模型转换器,从而将旧模型转换为语义等价的新模型.

马浩海博士[30]根据元模型与模型的特殊关系提出,可以将 OO 度量方法应用到元模型度量上(特别是建立在 OO 之上的元模型,比如 UML).其给出的元模型质量模型包括语法质量、语义质量、语用质量、实现质量和演化质量.

1.3 建模工具的集成

元建模主要应用于领域建模,其所建立的领域语言应用范围较小.这虽然有利于提高抽象层次、有利于代码的自动生成(从而有利于 MDA 的实现),但是也给复杂系统的建模带来了一些问题.一个复杂系统往往涉及多个领域的多个方面,比如嵌入式系统需要对硬件和软件建模,需要对系统静态结构和动态行为建模.此时,有两种方案可供选择:合成一个新的综合元模型或者将现有建模工具集成.合成新的综合元模型相对比较简单,但是必须为合成的元模型开发新的建模工具以及代码生成器.而且,当某个领域语言变化时,合成的元模型也需要作相应变化.人们在合成元模型上作了许多探索,比如 Ledecz 等人[31]在 UML 中引入 3 个新关系: Equivalence(等价)、Implementation Inheritance(实现继承)和 Interface Inheritance(接口继承)以辅助元模型的合成.但总的来说,以合成元模型的方式来解决对复杂系统的建模问题依然困难重重.

DOME 则采用工具集成的方法[32]来解决这个问题.其优点是可以最大程度地复用已有建模工具,可以用不同的建模工具分别建立系统的不同视图;缺点是需要增加一个元模型来刻画系统的不同模块(视图)之间的关系.而且这种方法尚不成熟,还有许多问题需要解决.其中包括关系模型、元模型转换和代码生成等[33,34].

Gennaro 等人[35]则进一步地把软件开发过程集成到开发环境(workbench)中,以组织和集成各个领域建模工具.每个领域模型分别描述系统的一个方面,建立一个领域模型(系统在某个方面的视图)也是系统建模过程的一个步骤.但是,如何组织建模的各个步骤却无法在各个领域建模工具中得到体现.为此,Gennaro 等人[35]通过基于 UML 活动图的过程描述语言 PML 刻画软件开发过程模型.其所提出的 PML 的特色之一是可以在过程模型中应用依赖边(denpendency edge)来表示软件模型之间的约束规则(比如交互图中出现的任何参与者都必须在用况图中定义).工作台生成器(workbench generator)可以根据过程模型生成工作台,它集成了过程模型中涉及到各类模型的的建模工具,通过约束库(constrain repository)和 Xlinkit[36]可以检查各模型之间的约束.

2 元建模工具的现状

从引言的介绍可知,元建模是一个过程或者是一个活动,它需要元建模工具的支持.随着元建模技术的逐渐成熟,大量的元建模工具开始出现,包括以 MetaEdit+[16]为代表的商业工具以及 DOME,GEM[37]为代表的开放的研究性工具.本节按照如下 4 个方面对这些元建模工具进行分析和比较以揭示元建模工具的现状与发展方向:元建模框架结构(framework)、元模型的内容、元建模工具的易用性以及建模工具的特征.

2.1 元建模框架结构

本节将对前面提到的元建模的两种框架结构(如图 1 所示)进行分析比较,并对每种元建模框架结构分别给出具有代表性的元建模工具.

元建模的第 1 种框架结构(如图 1 中的左图所示)以通用建模工具为核心.通用建模工具可以根据不同的元模型生成相应的配置文档.一旦载入配置文档,通用建模工具就可以成为支撑相应元模型(所描述的建模语言)的建模工具.早期的元建模工具大都采用了这种框架结构.比较有名的包括 MetaEdit+,GME,DOME 等.其优点是有利于多建模方法的集成.比如在 MetaEdit+中同时导入多个元模型到通用建模工具中,此时,通用建模工具就可以很好地支持多个元模型(所描述的建模语言),从而比较方便地实现了多方法的集成.但是在集成模型中,依然无法在不同领域的对象之间建立关联,因为在元模型层上没有定义相应元模型元素之间的关系.

在元建模的第 2 种框架结构(如图 1 中的右图所示)中,建模工具生成器不具备可配置的能力.但是建模工具生成器可以根据元模型生成相应的建模工具.其代表是 EMF 以及基于 EMF 的 GMF.生成建模工具的好处是,

可以给用户提供一个独立的工具,有利于对建模工具的定制、修改和提高.例如,如果生成的建模工具不具备某些功能(比如代码生成能力),那么用户可以对生成的建模工具进行修改以添加所需的能力.相反地,这种定制和提高在通用建模工具中是难以实现的.

2.2 元模型的内容

在元建模中,元模型是用于刻画建模语言的[2].建模语言包括抽象语法、具体语法(表示法)和语义[38,39].抽象语法一般包括构造块和通用规则[40].构造块可以通过元模型构造块表示,通用规则通常由某种约束语言表达,比如 OCL.元建模工具能够描述元模型哪些方面,是元建模工具的主要功能指标之一.

几乎所有的元建模工具都可以描述建模语言的构造块.比较简单的元建模工具,如 EMF 能且只能描述建模语言的构造块.所以,EMF 生成的建模工具除了能画出模型之外就没有进一步的处理功能.

大部分元建模工具都可以定制建模元素的表示法,但是,不同的工具对表示法建模的支持力度也各不相同.DOME 提供几种常用的图符(比如矩形、圆形)以构造表示法,所以用它可以构造一些常用的表示法,但是对一些不常见的表示法就无能为力了.GME 则以位图来表示建模元素的表示法,所以它几乎可以定制任意形状的表示法.其缺点是无法将元素的属性和表示法绑定,比如,对象名字应该显示在表示该对象的矩形的顶部.GMF 虽然提供表示法描述机制,但是直到目前为止,GMF 的表示法描述还没有比较直观的所见即所得的 GUI 支持工具,而只能以树状结点的形式描述,如图 4 所示.MetaEdit+提供了专门的 GUI 工具以定制表示法并可以在图形上绑定元素的属性.遗憾的是,当前的所有元建模工具都还无法表示不同元素的表示法之间的复杂关系,比如,UML 中的 Port 与 Classifier 的表示法之间的关系(port 必须放置在某个 Classifier 的边界上).

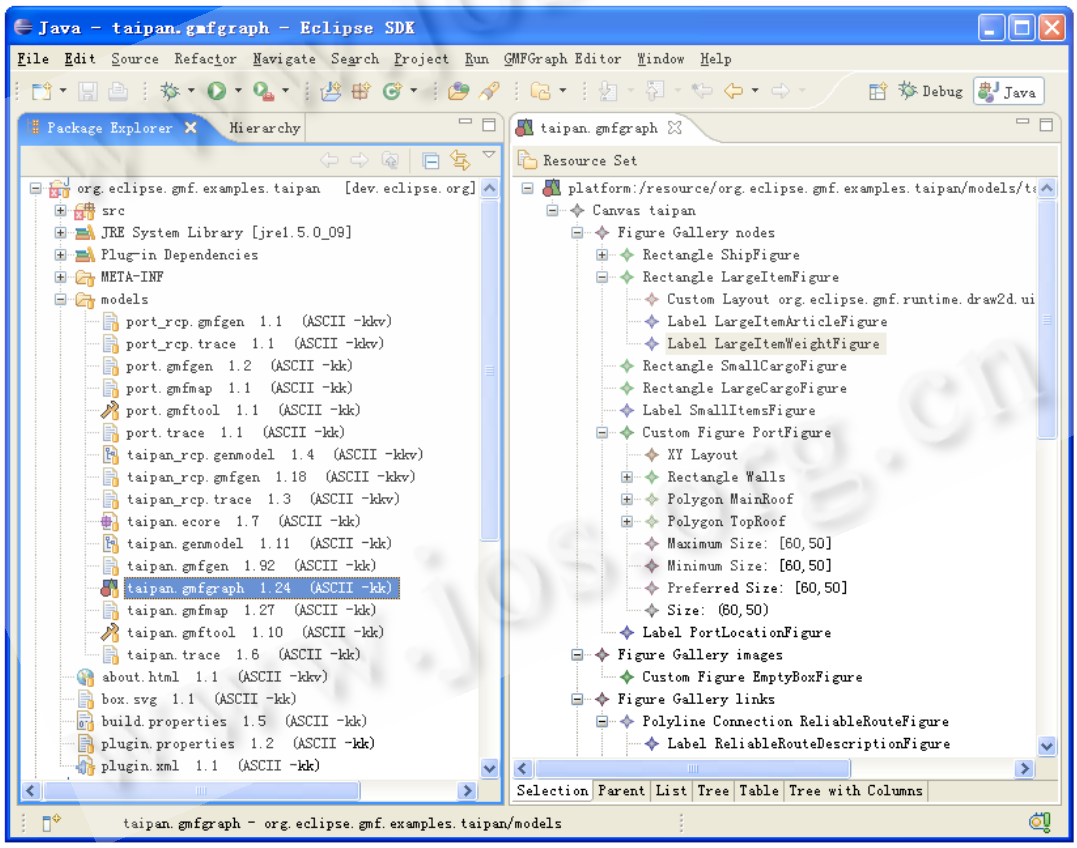

Fig.4 GMF notation

图 4 GMF 表示法定制

通用规则一般借助某种形式语言来描述.GMF 采用 OCL,这主要得益于 Eclipse 建模框架技术(eclipse modeling framework technologies,简称 EMFT)[41]下的 OCL 设施.GME 采用的是基于 OCL 扩展而来的通用断言表达语言(universal predicate expression language)[37].相对而言,DOME 对通用规则的支持略显单薄:其所支持的约束语言 Alter 的表达能力差强人意.随着 QVT[42]标准即将颁布,相信以 OCL 为约束描述语言的工具将会越来越多.

建模语言的语义描述主要有两种方式:文本描述以及转换规则.自然语言描述不利于机器理解,所以也就无法将语义嵌入生成的建模工具.定义模型语义的另一方式是通过定义转换规则把模型转换到某个具有严格语义的模型(比如状态图或者可执行程序).MetaEdit+提供了脚本语言,以根据模型的属性产生相应的报表,用户也可以使用这个语言来定义代码生成器.GME 则通过图转换来刻画元模型的语义[43].

元建模工具对元模型内容的描述能力的总结见表 1.

Table 1 Comparison of capabilities of metamodeling tools

表 1 元建模工具对元模型的描述能力一览

MetaMo leling tools	Buil ling blocks	Generic rules	N otations	Sen antics
Met ıEdit+	Yes	Constraints on ports and co ınections	GU I drawing	Report generatic n language
C ME	Yes	Universal predicate expressi(n language	Images	Graph transfo mations
D( )ME	Yes	Alter	Cor ımon icons	Alter
E MF	Yes	No	No	No
C MF	Yes	OCL	GEF	No

2.3 元建模工具的易用性

元模型工具的易用性是工具的主要衡量指标之一.EMF 提供的编辑器相当原始,它以元模型树的形式,而不是所见即所得的形式构建元模型.所以,大部分时候人们使用 UML 类图编辑器(比如 JBOO,Rose)等图形化建模工具来构造 ECore 模型,然后导入 EMF 作进一步处理.GMF 与 EMF 一样采用 ECore 作为标准元元模型,但它本身并不提供元模型抽象语法编辑器.GMF 依赖于 EMF 所提供的 ECore 编辑器.MetaEdit+也没有提供所见即所得的元模型编辑器,它采用的是简单的对话框模式.这种方式虽然简单,但是非常不直观.许多主流的元建模工具(比如 DOME,GME 等)基本上都提供了与主流建模工具类似的所见即所得的元模型编辑器.

对表示法的建模主要有 3 种模式:选择、树视图和矢量图.DOME 的表示法只能使用预定义的几种图形,所以很自然地采用了选择模式,用户只需通过下拉框选择即可.支持矢量图的 MetaEdit+则专门提供了一个类似画图的工具用于定制表示法.GME 使用的位图文件虽然也是通过画图工具编辑得到的,但是它的显示内容无法与元素的属性相绑定.相比之下,MetaEdit+的表示法构造形式最可取.GMF 则介于 DOME 与 MetaEdit+之间,它的功能比 DOME 强,但没有 MetaEdit+直观.它提供的是如图 4 所示的树形编辑器.

2.4 建模工具的特征

建模工具是元建模工具的输出,所以,从生成(或配置得到)的建模工具的特征可以间接地反映一个元建模工具的能力.这不但反映了元建模描述语言对建模语言的刻画能力(建模工具生成器的输入),也反映了建模工具生成器的能力强度.建模工具生成器是元建模工具的核心部件之一,它根据输入的元模型生成建模工具(或者通用建模工具的配置文档).其作用类似于建模工具中的代码生成器(根据模型生成应用系统).

自动生成的建模工具通常只有模型创建、浏览和修改的功能,而没有模型检查、模型转换(代码生成)等进一步处理的能力.这主要取决于元建模工具是否支持建模语言的规则和语义的刻画.如果不能刻画语言规则,就不可能进行模型检查;如果没有严格的语义模型,也难以生成建模工具的模型转换器(代码生成器).而 GME,DOME 等同时具有规则和语义描述机制的元建模工具,其所生成的建模工具就会有模型检查和模型转换的能力.

3 总结与展望

在 MDA 的背景之下,元建模逐渐成为软件工程领域的又一研究热点.经过几年的研究与实践,元建模工具

自动生成的建模工具在功能、易用性等各方面都已接近主流商用 UML 建模工具.大量工程实践案例的成功经验[44]不仅有利于元建模的提高和改进,更重要的是,它向人们展示了元建模的可行性和优越性,为元建模的推广起到了巨大的促进作用.相信在不远的将来,元建模技术以及基于元建模的 MDA 会有更大的发展.另一方面,元建模及其相关技术还不够成熟,还有许多有待研究和改进的地方.

首先,没有一个统一的元元模型阻碍了不同元建模工具之间的数据(元模型)交换.虽然 OMG 推出了标准元模型 MOF,但是因为历史原因以及 MOF 本身的不足,MOF 未能成为事实上的标准.在工程实践方面,MOF 需要扩展以描述建模语言的具体语法和语义.我们的研究小组在分析和比较现有元元模型的基础上,推出了一个兼容 ECore 和 MOF 的扩展模型,并以此为基础开发实现了元建模工具 PKU Meta-Modeling Tool(PKU MMT).

建模语言的语义刻画非常困难.到目前为止,刻画语义的主要方式还是模型转换.而模型转换主要靠脚本语言、模版或者图转换.但是,这些手段都有其自身的不足,难以生成完整的应用系统,难以进行逆向工程(从源代码生成系统模型).而且利用这些手段描述建模语言非常昂贵,其开发成本大约是整个领域可视化语言(DSVL)开发成本的一半左右[26].人们迫切需要更为高效、直观的描述手段.可视化的标准模型转换语言 MOF QVT[42]可以在一定程度上解决这个问题,但是,MOF QVT 的定义不够严谨,难以对建模语言的语义进行推理和证明.特别是对 OCL 的使用几乎没有任何约束,从而增加了对模型语义和特性进行推理和验证的难度.图转换语言也是描述建模语言语义的手段之一,但是传统的图转换语言在表达能力上稍有欠缺.通过扩展图转换语言的描述能力,同时结合受限的 OCL 约束将是一个比较有可能取得突破的研究方向.

复杂系统的建模需要用不同的领域建模语言来描述系统的不同侧面,然后将各个侧面关联起来构成系统的完整描述.跨领域支持目前还处于探索阶段.到目前为止,我们只看到 Honeywell 的 DOME 在尝试支持跨领域建模.Salerno 大学的 MetaCASE Workbench 虽然可以集成不同建模工具,但这种集成仅限于检查不同模型之间的简单约束,如何集成不同侧面以生成完整的应用系统却还没有有效的解决方案.更有效的方式应该是结合建模语言的语义描述来支持跨领域建模工具的生成.

References:

[1] Object Management Group. UML 2.0 infrastructure specification. OMG Adopted Specification ptc/03-09-15. 2003.

[2] Object Management Group. UML2 superstructure final adopted specification. 2003.

[3] Object Management Group. MDA guide version 1.0.1. OMG/03-06-01. 2003.

[4] Shao WZ, Yang FQ. Object-Oriented System Design. 2nd ed., Beijing: Tsinghua University Press, 2007 (in Chinese).

[5] 2007. http://www-306.ibm.com/software/awdtools/developer/rose/index.html

[6] 2007. http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html

[7] 2007. http://www.sparxsystems.com.au/ea.htm

[8] Ma ZY, Zhao JF, Meng XW, Zhang WJ. Research and implementation of jade bird object-oriented software modeling tool. Journal of Software, 2003,14(1):97−102 (in Chinese with English Abstract). http://www.jos.org.cn/1000-9825/14/97.htm

[9] MetaEdit Inc. Domain-Specific modeling with MetaEdit+ 10 times faster than UML. White Paper, 2005.

[10] Cook S. Domain-Specific modeling and model-driven architecture. In: Frankel D, Parodi J, eds. Proc. of the MDA Journal: Model Driven Architecture Straight from the Masters, Chap. 3. Meghan-Kiffer, 2004. http://www.davidfrankelconsulting.com/MDAJournal.htm

[11] van Deursen A, Klint P. Little languages: Little maintenance? Journal of Software Maintenance: Research and Practice, 1998,10(2): 75−92.

[12] Ledeczi A, Bakay A, Maroti M, Volgysei P, Nordstrom G, Sprinkle J, Karsai G. Composing domain-specific design environments. IEEE Computer, 2001,34(11):44−51.

[13] Sorenson PG, Tremblay JP, McAllister AJ. The metaview system for many specifcation environments. IEEE Software, 1998,5(2): 30−38.

[14] Ebert J, Säuttenbach R, Uhe I. Meta-CASE in practice: A case for KOGGE. In: Olive JAPA, ed. Advanced Information Systems Engineering, Proc. of the 9th Int'l Conf. on Advanced Information Systems Engineering (CaiSE'97). LNCS 1250, Springer-Verlag, 1997. 203−216.

[15] de Lara J, Vangheluwe H. Using AtoM3 as a meta-CASE tool. In: Proc. of the 4th Int'l Conf. on Enterprise Information Systems (ICEIS 2002). 2002. 642−649. http://www.iceis.org/iceis2002/

[16] Object Management Group. Meta object facility (MOF) 2.0 core specification. OMG Adopted Specification ptc/04-10-15. 2004.

[17] Bichler L. Tool support for generating implementations of MOF-based modeling languages. In: Proc. of the Domain-Specific Modeling Workshop at OOPSLA 2003. 2003. http://www.dsmforum.org/events/DSM03/papers.html

[18] Costagliola G, Gravino C. Constructing meta-CASE workbenches by exploiting visual language generators. IEEE Trans. on Software Engineering, 2006,32(3):156−175.

[19] Esser R, Janneck JW. A framework for defining domain-specific visual languages. In: Proc. of the Workshop on Domain Specific Visual Languages, Conjunction with ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications OOPSLA 2001. 2001. http://www.isis.vanderbilt.edu/oopsla2k1/Papers/Esser.pdf

[20] de Lara J, Guerra E, Vangheluwe H. Meta-Modelling, graph transformation and model checking for the analysis of hybrid systems. In: Proc. of the AGTIVE 2003. LNCS 3062, 2003. 292−298.

[21] Honeywell, Inc. DOME guide. 1999. http://www.htc.honeywell.com/dome/

[22] MetaCase. MetaEdit+4.0 user's guide. http://www.metacase.com/support/40/manuals/index.html

[23] Pohjonen R. Metamodeling made easy—MetaEdit+ (tool demonstration). In: Glück R, Lowry M, eds. Proc. of the Generative Programming and Component Engineering, 4th Int'l Conf. (GPCE 2005). LNCS 3676, 2005. 442−446.

[24] Zhu H, Shan LJ. Well-Formedness, consistency and completeness of graphic models. In: Proc. of the UKSIM 2006. 2006. 47−53.

[25] Gonzalez-Perez C, Henderson-Sellers B. A powertype-based metamodelling framework. Software & System Modeling, 2006,5(1): 72−90.

[26] Tolvanen JP, Kelly S. Defining domain-specific modeling languages to automate product derivation: Collected experiences. In: Proc. of the Software Product Lines: The 9th Int'l Conf. (SPLC 2005). LNCS 3714, 2005. 198−209.

[27] Hruby P. Domain-Driven development with ontologies and aspects. In: Proc. of the Domain-Specific Modeling Workshop at OOPSLA 2005. 2005. http://www.dsmforum.org/events/DSM05/Papers.html

[28] Korhonen K. A case study on reusability of a DSL in a dynamic domain. In: Proc. of the 2nd Workshop on Domain-Specific Visual Languages. 2002. http://www.dsmforum.org/events/DSVL02/Papers.html

[29] Sprinkle J, Karsai G. A domain-specific visual language for domain model evolution. Journal of Visual Languages and Computing, 2004,15(2):291−307.

[30] Ma HH. Research on the meta-model quality evaluation for the UML family of languages [Ph.D. Thesis]. Beijing: Peking University, 2005 (in Chinese with English abstract).

[31] Ledeczi A, Nordstrom G, Karsai G, Volgyesi P, Maroti M. On metamodel composition. In: Proc. of the 2001 IEEE Int'l Conf. on Control Applications. 2001. 756−760. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=973959&fromcon

[32] Schloegel K, Oglesby D, Engstrom E, Bhatt D. A new approach to capture multi-model interactions in support of cross-domain analyses. Technical Report, 2001.

[33] Schloegel K, Oglesby D, Engstrom E, Bhatt D. Composable code generation for model-based development. In: Proc. of the Software and Compilers for Embedded Systems: The 7th Int'l Workshop, SCOPES 2003. 2003. 211−225.

[34] Schloegel K, Oglesby D, Engstrom E. Towards next generation metamodeling tools. Technical Report, 2001.

[35] Costagliola G, Deufemial V, Ferrucci F, Gravino C. Constructing meta-CASE workbenches by exploiting visual language generators. IEEE Trans. on Software Engineering, 2006,32(3):156−175.

[36] Nentwich C, Capra L, Emmerich W, Finkelstein A. xlinkit: A consistency checking and smart link generation service. ACM Trans. on Internet Technology, 2002,2(2):151−185. http://www.xlinkit.com

[37] Ledeczi A, Maroti M, Bakay A, Karsai G, Garrett J, Thomason IV C, Nordstrom G, Sprinkle J, Volgyesi P. The generic modeling environment. In: Proc. of the Workshop on Intelligent Signal Processing. 2001. http://www.mit.bme.hu/events/wisp2001/

[38] Atkinson C, Kuhne T. The essence of multilevel metamodeling. In: Gogolla M, Kobryn C, eds. Proc. of the Unified Modeling Language, Modeling Languages, Concepts, and Tools: The 4th Int'l Conf. LNCS 2185, 2001. 19−33.

[39] Harel D, Rumpe B. Modeling languages: Syntax, semantics and all that stuff, Part I: The basic stuff. Technical Report, MCS00-16, Weizmann Science Press of Israel, 2000.

[40] 2007. http://www.metamodel.com/

[41] 2007. http://www.eclipse.org/emft/projects/

[42] Object Management Group. MOF QVT final adopted specification. OMG Adopted Specification ptc/05-11-01. 2005.

[43] Agrawal A, Karsai G, Shi F. Graph transformations on domain-specific models. Technical Report, ISIS-03-403, Vanderbilt University, 2000.

[44] 2007. http://www.metacase.com

附中文参考文献:

[4] 邵维忠,杨芙清.面向对象的系统设计.第 2 版,北京:清华大学出版社,2007.

[8] 麻志毅,赵俊峰,孟祥文,张文娟.青鸟面向对象软件建模工具的研究与实现.软件学报,2003,14(1):97−102. http://www.jos.org.cn/1000-9825/14/97.htm

[30] 马浩海.面向 UML 建模语言家族的元模型质量评估技术研究[博士学位论文].北京:北京大学,2005.

刘辉(1978－),男,福建长汀人,博士生,主要研究领域为面向对象建模,软件重构,元建模,MDA,形式化软件工程.

麻志毅(1963－),男,副教授,CCF 高级会员,主要研究领域为软件工程,软件工程支撑环境,面向对象技术,构件技术.

邵维忠(1946－),男,教授,CCF 高级会员,主要研究领域为软件工程,软件工程环境,面向对象技术,软件复用与构件技术.