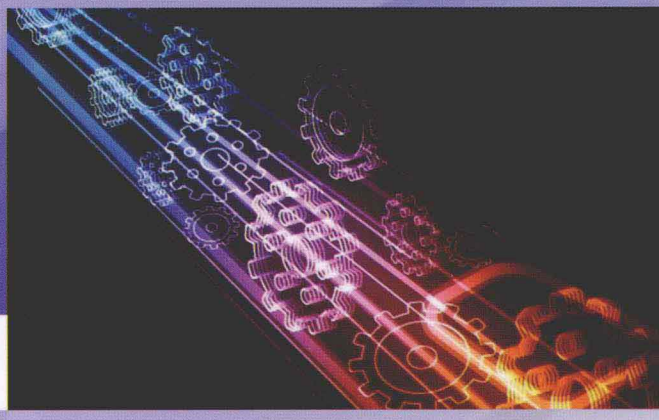

元建模与Web应用系统生成

□ 王海林 著

國防工業出版社
National Defense Industry Press

元建模与 Web 应用系统生成

王海林　著

国防工业出版社
·北京·

内 容 简 介

了解和掌握好的系统开发方法是每一个系统分析、设计和开发人员必须具备的基本技能，面向特定应用领域的元建模方法就是一种目前比较好的系统开发方法，它可以大大提高系统的开发效率。本书以模型驱动开发思想为主线，首先介绍了元建模的基本概念、基本方法、元建模工具的特点以及元建模框架，阐述了面向特定应用领域进行元建模的必要性和优越性，进而系统地阐述了元建模工具 MetaEdit + 的有关概念、元元模型、元类型工具以及其它工具的使用，然后通过几个实例，讲解了 MetaEdit + 如何在较为简单的局部应用领域中进行元建模、领域建模和生成器设计，最后通过一个综合性实例——Web 应用系统“WebShopping”，详细地讲述了 Web 应用系统的生成过程。

本书主要供信息系统分析、设计和开发人员使用，也可作为高等院校信息管理专业、计算机应用软件专业的教学参考书。

图书在版编目(CIP)数据

元建模与 Web 应用系统生成/王海林著. —北京：国防工业出版社，2014.1

ISBN 978-7-118-09264-6

Ⅰ.①元... Ⅱ.①王... Ⅲ.①软件工程 - 研究 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字(2013)第 288987 号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100048)

北京奥鑫印刷厂印刷

新华书店经售

*

开本 787 × 1092 1/16 **印张** 15¼ **字数** 379 千字

2014 年 1 月第 1 版第 1 次印刷 **印数** 1—3000 册 **定价** 38.00 元

(本书如有印装错误，我社负责调换)

国防书店：(010)88540777 发行邮购：(010)88540776

发行传真：(010)88540755 发行业务：(010)88540717

前　言

在信息系统的发展过程中，曾出现过许多著名的系统开发方法，如生命周期法、快速原型法、面向对象法、模型驱动开发方法等，其中面向对象法是目前较为流行的方法。统一建模语言（UML）的出现，为面向对象的系统开发提供了一种标准的建模语言，因而加快了面向对象的系统开发方法的流行和普及。UML 是一种通用的、可视化的建模语言，它的广泛应用大大提高了模型在系统开发中的地位，以模型为核心的系统开发思想已逐渐被人们所接受，因此模型驱动开发方法也越来越受到人们的重视。然而，UML 有时并不完全适合于模型驱动开发方法，这是因为 UML 是一种通用建模语言，它不仅庞大而且复杂，不易学习和掌握，尤其是对非计算机专业的应用领域专家，这就使得建模工作只能由系统开发人员和领域专家共同来完成，但实际建模的工作量往往很大，再加上系统开发人员对应用领域知识不熟悉，需要花较多时间去学习和了解，致使系统开发人员耗费了大量的时间和精力，造成不必要的资源浪费。

近年来一些系统开发人员尝试将面向特定领域建模语言（DSL）应用到模型驱动系统开发中，DSL 与 UML 的主要区别是其对应用领域的重视，它不要求建模工具“大而全”，而只追求其“小而专”，也就是说，它不要求像 UML 这样的通用建模语言提供很多很全的功能，只要求提供一种面向特定领域的建模语言 DSL 能快速有效地解决实际问题即可。由于 DSL 是面向特定领域的，其提供的功能仅限于本领域而且为领域专家所熟知，所以简单易学。整个建模工作可由领域专家独自完成，从而大大减轻了系统开发人员的工作量，进而提高了系统的开发效率。因此，基于 DSL 的模型驱动开发方法将是今后信息系统开发方法的发展方向。

基于 DSL 的模型驱动开发方法的核心是元建模、领域建模和生成器设计。目前，介绍模型驱动开发方法的书籍不少，但全面、系统地介绍基于某种特定的 DSL 开发信息系统的书籍却很少，这正是作者撰写本书的目的所在。

本书分为上、中、下三篇，共 9 章。书中的重点是中篇的内容，核心是下篇的内容。这两篇的内容都是作者本人近年来学术研究系列论文积累形成的最终成果，特别是下篇中所述的 Web 应用系统生成器的设计，是作者在新西兰奥塔哥大学做访问学者期间所取得的研究成果。从中篇的第 6 章内容不难看出 DSL 建模的灵活性和应用的宽广性，而这是 UML 望尘莫及的。本书使用的建模工具是 MetaEdit +，鉴于目前国内还没有一部书籍系统地介绍过该工具的使用，为了使读者更容易地学习和理解中篇和下篇的内容，本书在上篇用了一定的篇幅系统地介绍了 MetaEdit + 的使用，这样一来，读者不仅“看得懂”而且“学得会”，本书的实用性就大大增强了。

上篇为基础篇，共包括 5 章。第 1 章介绍了一些元建模的相关概念、常用的元建模工具和元建模框架。第 2 章介绍了元建模工具 MetaEdit + 的基本概念、元元模型及其元类型工具软件的使用，这一章的内容主要用于元建模。第 3 章介绍了 MetaEdit + 的编辑器和调试器，包括图形符号编辑器、对话框编辑器、生成器编辑器和调试器，这一章的内容主要用于领域建模。

第4章介绍了 MetaEdit + 的生成器定义语言 MERL 的语法和命令的使用，这一章的内容主要用于生成器设计。第5章介绍了 MetaEdit + 的 XML 导入和导出，讲解在 MetaEdit + 中导入和导出 XML 模型文件、类型文件，这一章的内容主要用于 MetaEdit + 与其它软件的集成。中篇为应用篇，通过三个实例讲解如何使用 MetaEdit + 在一个较为简单的局部应用领域中进行元建模、领域建模和生成器设计，共包括3章。第6章介绍了 LED 千足虫玩具设计，重点讲解元建模和领域建模。第7章介绍了 Windows 应用系统主窗口自动生成，重点讲解生成器设计。第8章介绍了数据库元建模及生成器设计，阐述了元建模在数据库设计中的应用，介绍了数据库生成器的设计，使用数据库生成器可以从数据库概念模型直接生成数据库物理模型。下篇为系统篇，通过一个综合性的实例——Web 应用系统“WebShopping”，详细阐述了 MetaEdit + 在复杂应用领域中的元建模、领域建模和生成器设计，共包括1章，即第9章。第9章为 Web 应用系统生成，它几乎涵盖了 MetaEdit + 的所有主要应用。

本书的撰写得到了“山西省回国留学人员科研资助项目”（项目下达文号：2013(9)晋留管办发，项目编号：2013 - 076）的资助，在此表示衷心的感谢！

作者在本书的撰写过程中还学习和参考了国内外大量的书籍和文献资料，主要的已列在本书的参考文献中，在此向有关作者致以深深的谢意。此外，山西财经大学信息管理学院的马尚才教授、贾伟教授、尚成国教授对本书的撰写给予了大力支持和帮助，常新功教授在百忙之中认真审阅了本书的初稿，在此一并表示感谢！

由于水平有限，书中难免会出现缺点和错误，恳切希望广大读者批评和指正。

王海林

2013 年 10 月于山西财经大学

wang_hailin0351@163.com

目　录

上篇　基础篇

中篇　应 用 篇

下篇 系 统 篇

上篇　基础篇

本篇介绍元建模、生成器设计的基本概念和基础知识，这些基本概念和基础知识是读者进一步学习下两篇的基础。

基础篇包括5章。

第1章绪论，初步介绍与元建模相关的一些概念，介绍常用的元建模工具并对它们做了比较分析，给出两种常用的元建模框架。

第2章MetaEdit+元建模，系统地介绍了元建模工具MetaEdit+的基本概念及工具软件的使用。

第3章MetaEdit+的编辑器和调试器，重点介绍了MetaEdit+的图形符号编辑器、对话框编辑器、生成器编辑器和调试器。

第4章MetaEdit+的生成器定义语言，详细介绍了MetaEdit+提供的生成器定义语言MERL的语法和命令的使用。

第5章MetaEdit+的XML导入和导出，讲解了在MetaEdit+中导入和导出XML模型文件、类型文件。

第1章 绪 论

信息系统开发方法一直是信息管理与信息系统学科研究的主要课题之一，由于管理信息系统(Management Information System，MIS)容易受用户单位业务变化的影响，因此开发此类系统的难度相对较大，系统开发方法的选择就显得格外重要。在管理信息系统的发展过程中曾出现过许多著名的系统开发方法，如生命周期法、速成原型法、面向对象法等，每一种新方法的提出都对管理信息系统的发展具有里程碑式的意义，近年来兴起的模型驱动开发方法就是如此。模型驱动开发方法的提出不仅大大提高了系统开发的效率，而且也更便于系统开发人员之间、系统开发人员与用户之间的交流。

本章将介绍模型驱动开发、元建模的相关概念，为后续章节内容的介绍奠定一个良好的基础。

1.1 元建模的相关概念

这一节主要介绍元建模中常用的一些基本概念和术语。

1.1.1 模型在信息系统开发中的角色

目前，信息系统的开发变得越来越复杂，所涉及的领域也越来越广泛，系统开发者不仅需要掌握多种不同的技术，如面向对象技术、XML、数据库语言 SQL、脚本语言、过程定义语言等，而且还需要把来自于问题领域的需求转换成解决方案，需要对许多架构和协议深刻理解。此外，最终用户常常对系统的运行效率、易用性、易扩展性也有较高的期望。因此，信息系统的开发并不是一件容易的事。

在信息系统开发以外的一些领域，如汽车、家电(电视机，洗衣机，电冰箱)行业等，能够经常看到成本低、可靠性高的产品。在过去的几十年里，制造行业一直采用这样一种生产流程，即通过一连串复杂的、标准化的步骤来制造一种产品(如汽车或电视机)，其中有许多的步骤是完全自动化的。

系统开发者希望使用同样的原理来研制信息系统，遗憾的是，目前还没有开发出一个能够允许有效分离信息系统中关注点的系统说明语言。尽管系统开发者使用不同的计算机语言来书写应用逻辑以完成不同的开发任务，例如使用 XML 在应用组件中传递数据，使用 SQL 定义和查询数据库中的数据等，但这些语言中没有一个是直接针对最终用户所面对的业务问题。

本书将要介绍的信息系统开发技术是面向特定领域建模(Domain Specific Modeling，DSM)的技术，也称为元建模(Meta-Modeling)技术。DSM 被设计为直接面向它所要解决的问题领域。

模型是对现实世界的人或事物的模拟和抽象，模型里的每个元素都可以映射到现实领域中的一个人、事物或概念。多年来，模型对于定义信息系统如何来保存数据一直是很重要的，

现在，模型的应用更为广泛，它可以组织数据、对数据进行操作，例如对业务过程建模。模型受欢迎是因为它能够很好地描述问题从而避免陷入技术细节中。当技术变得越来越复杂的时候，模型是提高生产效率的必要手段。模型的另一个好处是有助于系统开发人员和应用领域专家之间的沟通，它是弥合技术和业务之间缝隙的有效方法。

1.1.2 模型驱动架构 MDA

MDA 是 Model-driven architecture 的缩写，意思是模型驱动架构。MDA 是一种用于应用系统开发的软件设计方法(信息系统开发的重点是应用软件的设计)，它提供了一套软件设计的指导规范，这套指导规范是用模型来表示的。MDA 是由一个名叫 OMG(Object Management Group，即对象管理组织)的国际联盟于 2001 年颁布的。

在 IT 界，术语 MDA 一般是指在软件开发过程中使用模型。但事实上，OMG 把这个术语注册为商标，并将其引申为使用 OMG 的建模技术进行模型驱动开发的概念。

OMG 确定了以 MDA 作为对象管理组织未来的发展方向。MDA 通过使用软件工程方法和工具，为分析、设计、实现和维护信息系统提供了方法。在 MDA 中，模型不再仅仅是描述系统和辅助沟通的工具，而是软件开发的核心和主要媒介。MDA 采用标准模型描述方法来详细描述信息系统，如从业务需求分析、系统功能和体系结构设计、包含平台技术细节的系统实现等三个层次，MDA 都给出了相应的描述模型。

各模型之间的关系如图 1.1 所示。

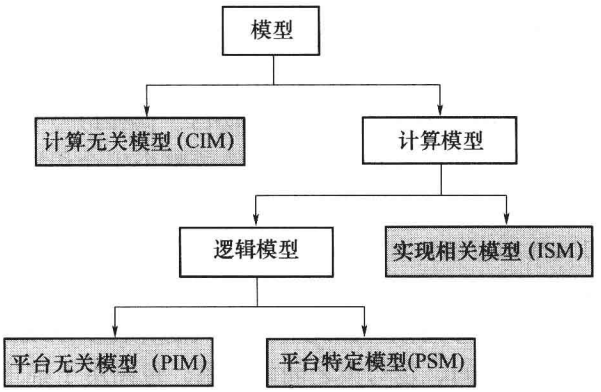

图 1.1 MDA 模型的分类

各模型及其说明：

(1) 计算无关模型(Computation Independent Model，CIM)，也称为领域模型、业务模型，是指在系统需求分析阶段从纯业务角度描述系统要完成的工作。

(2) 平台无关模型(Platform Independent Model，PIM)，从功能设计角度描述系统的体系结构，与技术细节无关。

(3) 平台特定模型(Platform Specific Model，PSM)，描述基于特定平台的解决方案。

(4) 实现相关模型(Implementation Specific Model，ISM)，也称为物理模型，是指面向最后的编程描述系统的实现细节。

模型转换是 MDA 的核心活动之一，从 CIM 到 PIM，从 PIM 到 PSM，从 PSM 到 ISM，都是通过模型转换实现的。从这个意义上讲，代码生成也可以看作是一种模型转换。从 PSM

生成信息系统的大部分代码，从而大大减少编程的工作量，提高系统开发的效率。

MDA 是由多个标准组成的，如 UML、MOF、XMI 和 CWM 等，在下面两个小节中分别介绍 UML 和 MOF。

1.1.3 统一建模语言 UML

UML 是 Unified Modeling Language 的缩写，称为统一建模语言，它是美国 Rational 公司设计的一种通用的、图形化的建模语言。UML 于 1997 年 11 月被 OMG 批准作为面向对象开发的行业标准语言。MDA 颁布以后，它成为 MDA 的众多标准中的一个标准。

UML 的推出在软件工程界引起了很大的反响，有人认为它有划时代的重大意义。从名称上就不难看出 UML 的重要性。所谓的“统一”其实有两层含义，一方面说明了 UML 融合了多种建模语言和技术，如融合了 OMT、OOSE 和 Booch 中的概念，这使得 UML 实现了多种建模语言和技术在概念上描述的一致性；另一方面说明了 UML 并不仅仅是软件系统的“专利”，非软件行业同样可以使用 UML 进行建模描述，这使得 UML 成为一种不受应用领域限制的通用建模语言。

UML 是一种可视化的建模语言，用图形来描述模型的好处是直观、容易理解，便于系统分析人员和设计人员之间、软件从业人员与客户之间进行交流。

UML 提供了五大类(共 9 种) 图形用于建模，详述如下：

第一类是用例图(Use Case Diagrams)，用于从用户角度描述系统的功能。用例是指从系统外部可见的行为，是系统为参与者(Actor)提供的一项完整的服务。用例之间可以抽象出包含(Include)、扩展(Extend)和泛化(Generalization)的几种关系。

第二类是静态图(Static Diagrams)，包括类图(Class Diagrams)、对象图(Object Diagrams)和包图(Package Diagrams)。其中类图用来描述类、接口、协作以及它们之间相互关系的图，用来显示系统中各个类的静态结构，这种结构在系统的整个生命周期内都是有效的。类图不仅可以定义系统中的类，表示类之间的联系，如组合、聚合、关联、依赖等，而且还可以定义类的内部结构(即类的属性和方法)。对象图描述的是参与交互的各个对象在交互过程中某一时刻的状态。对象图可以被看作是 UML 类图在某一时刻的实例。对象图几乎使用了与类图完全相同的标识。由于对象存在生命周期，因此对象图只能存在于系统的某一时间段。包图由包或类组成，用来表示包与包之间、包与类之间的关系。包图用于描述系统的层次结构。

第三类是行为图(Behavior Diagrams)，描述系统的动态模型和组成对象间的交互关系，包括状态图(Statechart Diagrams)和活动图(Activity Diagrams)。状态图是对类图的一种补充，主要用于描述一个对象在其生命周期的动态行为。一个对象在其生命周期的行为表现为该对象所经历的状态序列、引起状态转移的事件，以及因状态转移而伴随的动作。在实际使用时并不需要为所有的类画状态图，仅为那些有多个状态，其行为受外界环境影响而发生改变的类画状态图。一般可以用状态机(State Machine)对一个对象的生命周期建模，状态图主要用于显示状态机和描述控制流。活动图主要用于描述业务过程和工作流，描述满足用例要求所要进行的活动以及活动间的约束关系，有助于处理并行活动。

第四类是交互图(Interactive Diagrams)，描述对象间的交互关系，包括顺序图(Sequence Diagrams)和协作图(Collaboration Diagrams)。顺序图的主要用途之一是细化用例图。一个用例常常被细化为一个或多个顺序图。顺序图主要是定义对象发送消息的顺序，同时显示对象之间的交互关系。顺序图将交互关系表示为一个二维图，纵向是时间轴，从上而下表示消息发

送的时间顺序，而横向从左到右表示消息发送到的特定对象。与顺序图类似，协作图也用于描述对象间的协作关系，但二者的侧重点不同，如果强调时间和顺序，则使用顺序图；如果强调上下级关系，则选择协作图。

第五类是实现图(Implementation Diagrams)，包括部件图(Component Diagrams)和配置图(Deployment Diagrams)。部件图描述代码部件的物理结构及各部件之间的依赖关系。一个部件可能是一段源代码、一段二进制代码或一段可执行代码。部件图有助于分析和理解部件之间的相互影响程度。配置图描述运行时处理元素的配置以及依赖这些配置的软件成分、进程和对象，它也可以显示实际的计算机和设备(用节点表示)以及它们之间的连接关系。在节点内部，放置软件成分、进程和对象以显示节点与可执行软件单元的对应关系。

从系统设计角度看，当采用面向对象方法设计系统时，第一步是描述用户需求；第二步根据用户需求建立系统的静态模型，以构造系统的结构；第三步是描述系统的行为。前两步所建立的模型都是静态的，可以通过用例图、类图(包含包)、对象图、组件图和配置图等五个图形来完成，这五种图形代表了 UML 的静态建模机制。第三步所建立的模型是系统的动态模型，它可以通过状态图、活动图、顺序图和协作图等四个图形来完成，这四种图形代表了 UML 的动态建模机制。因此，UML 的主要内容也可以划分为静态建模机制和动态建模机制两大类。

1.1.4 元对象机制 MOF

为了实现 MDA，OMG 制定了一系列的标准，UML 就是其中之一。尽管 UML 并不是为 MDA 而生(UML 出现在 MDA 之前)，但是作为目前最为流行的建模语言，UML 已经占据了全球建模语言领域的大部分市场份额，成为建模语言事实上的标准，因此 OMG 将它作为 MDA 技术的基础是自然而然的明智选择。

另一个值得一提的标准是 MOF。MOF 是 Meta Object Facility 的缩写，意为元对象机制，它是比 UML 更高层次的抽象，它的目的是为了描述 UML 的扩展或者其它未来可能出现的类似 UML 的建模语言。

图 1.2 给出了 OMG 模型的层次结构，共分为四层，分别用 M3、M2、M1 和 M0 表示。

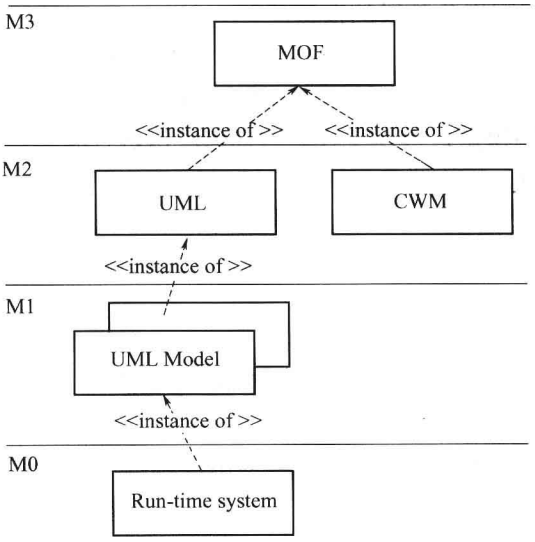

图 1.2 OMG 模型的层次结构

图中每个层次分别代表了对现实世界事物不同级别的抽象，每一层次都是对其下一层次的抽象描述，例如 M3 层是对 M2 层的抽象描述、M2 层又是对 M1 层的抽象描述。M3 层抽象级别最高，我们常把处于这个层次的模型称为元元模型(Meta-Meta Model)或元建模语言(Metamodeling Language)，MOF 就处于这个层次，所以可以通过 MOF 来创建元模型(Metamodel)；M2 层是 M3 层的实例(Instance)，我们常把处于 M2 层的模型称为元模型或建模语言(Modeling Language)，UML 就处于这个层次，因为 UML 就是一种建模语言，可以通过它创建模型；类似地，把 M1 和 M0 层称为模型层和实例层(或对象层)。

1.1.5 面向特定领域的语言 DSL

DSL 是 Domain Specific Language 的缩写，意为面向特定领域的语言，它可以是编程语言，如用于设计代码生成器的语言 MERL，也可以是用于其它目的的语言，如用于网页显示的 HTML、用于关系数据库查询的 SQL。显然 DSL 不是一个新概念，因为 SQL 已出现有 20 多年的历史了，不过我们这里讲的 DSL 特指用于建立模型的 DSL，从这个意义上讲它又是一个较新的概念。

与 DSL 相反的是 GPL(General Purpose Language)，即通用语言，如 C、Java、UML 等。

由于 UML 是作为一种通用建模语言而设计的，所以它过于庞大和复杂，不易学习和掌握，尤其是对那些非计算机专业的人员，如领域专家(Domain Specialists，即信息系统用户单位的业务人员)。

DSL 是专为特定领域建模而设计的语言，其功能仅限于满足该领域的要求，往往功能比 UML 少得多，因此它小巧易学，容易被领域专家掌握，这样许多建模工作就可以由熟悉业务的领域专家去完成，从而为系统开发人员节省了大量的时间，大大提高了系统开发的效率。

归纳起来，使用 DSL 的优点有：

(1) 高级别的重用。如果仅使用通用编程语言，则每次只能解决一个问题，但如果使用 DSL，它可以高效地解决一类相似的问题。

(2) 使用 DSL 的软件架构可以跨越软件开发过程各阶段之间的鸿沟，特别是通过代码生成可以很好地进行设计和实现各阶段的衔接。

(3) 让领域专家参与开发过程，不仅仅是需求分析阶段，创建领域模型阶段也需要参与。

(4) 通过参与开发过程，可以让不熟悉如何实现技术的领域专家，也能够更了解模型。

(5) 使用 DSL 表达的模型，可以在较高的抽象层次进行验证，这意味着可以在开发周期的早期，发现因为理解和表述的偏差而造成的错误。

(6) 一个模型中具备了重要的业务知识，将解决方案从一种技术迁移到另一种技术，或在同一技术的不同版本之间迁移，就变得相对容易。一般通过适当修改 DSL 生成器或解释器就可以做到。

1.2 元建模工具

元建模是指建立元模型的过程，这种元模型用以刻画某种建模语言，通常通过元建模工具来设计元模型。

元建模工具在面向特定领域建模中扮演重要角色，这一节介绍一些目前比较流行的元建模工具，并对它们提供的功能进行一些比较分析。以下是比较分析时需要考虑的一些因素：

(1) 供应商。
(2) 支持的平台。
(3) 许可证。
(4) 文档和技术支持：
① 用户指南；
② 教程；
③ 代码生成器定义说明书；
④ E-mail 支持。
(5) 元建模语言。
(6) 约束定义方法。
(7) 代码生成：
① 生成器定义语言；
② 生成器输出语言。

1.2.1 微软 DSL

微软 DSL 工具是 Microsoft Visual Studio 中的一个软件，因此微软 DSL 是一种只能运行于 Windows 平台的商品化工具。微软 DSL 工具集包括元建模环境、建模环境和代码生成功能，并提供了完整的文档支持。微软 DSL 允许开发人员自行设计专属的图形化工具，它内置了模型的相关支持，以及模型与图形之间的支持，还包括对模型的验证、规则、事务的支持。

微软 DSL 工具为建模者提供了一种图形化的操作方式。领域模型(Domain Model)是 DSL 定义的核心部分，它表示了领域的结构。领域模型定义了领域类(Domain Class)和领域关系(Domain Relationship)，即定义了组成模型的要素，并给出了将这些要素互相联系在一起的规则。

领域模型和表示是分离的，它仅定义了 DSL 所要处理的概念层上的内容，而并没有给出如何在图中表示这些概念。在 DSL 设计器中，图形和领域类的对应关系是非常直观的，有多种基本的图形和连接线类型可供选择，在定义了一组图形之后，还要定义一组图形的映射关系，也就是定义用哪个图形或者连接线来显示某个领域类或者领域关系。表示层和底层模型相分离的设计，可以在不改动模型的情况下，合理改变领域模型的表示方式。

在用 DSL 设计器完成 DSL 的定义后，将从这个 DSL 定义生成编辑和处理 DSL 实例工具的代码，然后就可通过写一些附加的代码，来增强所产生的设计器的功能。

1.2.2 GME

GME(Graphical Modeling Environment，即图形建模环境)是美国范德比特大学(Vanderbilt university)开发的元建模工具软件，它是一款只能运行于 Windows 平台的开源软件。GME 具有灵活的 DSL 配置功能，并提供了完整的文档支持。

GME元模型建立在UML之上，并用关系数据库来存储。GME使用的主要概念有文件夹、模型、原子(Atoms)、集合(Sets)、引用(References)、连接(Connections)、角色(Roles)、约束(Constraints)和切面(Aspects)，其中模型、原子、集合、引用和连接被称为一级对象。GME用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，但它的缺点是无法将元素的属性和表示法绑定。

GME采用OCL(Object Constraint Language，即对象约束语言)来定义各种关系约束。由于GME是基于COM的元建模工具，所以生成器设计语言可以采用C++、Visual Basic、Java、Python等。

1.2.3 GEMS

GEMS(Generic Eclipse Modeling System，即图形Eclipse建模系统)是Eclipse环境下的一个元建模工具。GEMS是GMT(Eclipse Generative Modeling Technologies)项目的一部分，并且要利用其它几个项目的软件，如EMF(Eclipse Modeling Framework)、GMF(Graphical Modeling Framework)、Draw2D和GEF(Graphical Editing Framework)。GEMS是以Eclipse插件的形式实现的，并可运行于任何Java平台。

GEMS的元建模语言是Ecore，Ecore用到的概念有实体(Entity)、属性(Attribute)、连接、继承(Inheritance)和切面。 元模型以图形方式创建，描述了实体及其它们之间的联系。

GEMS中的约束可以用Java、OCL和Prolog编写。与GME类似，GEMS也主要关注与元建模和领域建模，不过，它提供了书写代码生成器的扩展功能。GEMS也是以Eclipse 插件的形式实现的，生成器定义语言采用Java。

1.2.4 MetaEdit+

MetaEdit+是一款出现较早、使用最广的商用元建模工具，第一个商用版本发布于 1993年，目前最新版本为 MetaEdit+5.0。和 UML 一样，MetaEdit+是一个基于图形的建模工具，与 UML 不同的是它主要用于元建模。实际上 MetaEdit+是一个建模工具集，它提供了一组建模工具，如 diagram editors，matrix editor， table editor，browsers，report and code generation，method tools，API & XML connectivity，object repository。

MetaEdit+以GOPPRR作为元建模语言，GOPPRR是对GOPRR的一种扩展。元建模语言的名称来自于元类型名Graph(图)、Object(对象)、Property(属性)、Port(端口)、Relationship(联系)和Role(角色)。GOPPRR语言增加了端口概念到GOPRR语言。 对象是一种事物，可用一定形状的图形表示；联系是对象之间的连接，常用连线表示；角色定义了对象如何参与到一个联系中，常出现在联系的两端，例如用箭头表示；端口是角色连接到对象的一个特定的点，例如放大器由一个模拟信号输入端口和一个数字信号输入端口组成；图是一种由对象及其联系组成的图形。一个图中的对象可分解(Decomposition)为一个新图或通过扩展(Explosion)连接到其它的图。一个对象可以有零个或有一个分解图，但可以有多个扩展图，图的分解和扩展实现了图形模型的分层结构。

以上五种元类型都可以含有属性，属性只能作为其它元类型的一部分被访问，MetaEdit+为创建每一种元类型都提供了一个工具。例如，在Object tool中既可以创建一个新对象，又可以编辑一个已经存在的对象的属性。

MetaEdit+提供了两种元建模方法：基于图的元建模和基于表单的元建模。基于图的元模型是通过使用GOPPRR语言的可视化表示法将其设计为一幅图，图形元模型仅适用于简单的语言。基于表单的元模型是利用上面提及的创建元类型的工具创建的，它看上去可能没有基于图的元模型直观，但它更为精确、更具有可延伸性。

MetaEdit+提供了多种定义约束的方法。当在Graph tool中定义图时，联系的基数可以被定义在Graph tool的绑定页面中，其它约束可以定义在Graph的约束页面中。MetaEdit+规定了许