第 35 卷 第 6 期 计 算 机 工 程 2009 年 3 月
Vol.35 No.6 Computer Engineering March 2009
·软件技术与数据库· 文章编号：1000—3428(2009)06—0047—03 文献标识码：A 中图分类号：TP311.5

基于 MDA 的 Web 应用开发方法

张玉艳[1]，黄国栋[2]，侯金奎[3]
(1. 潍坊学院计算机与通信工程学院，潍坊 261061；2. 中国人民解放军防化指挥工程学院计算机教研室，北京 102205；
3. 山东大学计算机科学与技术学院，济南 250101)

摘 要：从软件工程的实施出发提出一种基于 MDA 的 Web 应用开发方法。该方法从平台无关的高层模型描述开始，依据转换两端建模元素的语法结构和语义表达特性定义模型间的映射规则，实现模型转换和代码生成。以 ASP.NET 作为目标平台进行实验，验证结果表明该方法遵循了 MDA 开发的实质、过程和要求，能够对模型驱动开发提供有力的支持。
关键词：模型驱动体系结构；模型描述；模型转换；代码生成

MDA-based Development Approach for Web Applications

ZHANG Yu-yan[1], HUANG Guo-dong[2], HOU Jin-kui[3]
(1. School of Computer and Communication Engineering, Weifang University, Weifang 261061; 2. Room of Computer Education, Institute of Chemical Defense of PLA, Beijing 102205; 3. School of Computer Science and Technology, Shandong University, Jinan 250101)

【Abstract】 From the viewpoint of software engineering implementation, a MDA-based development approach for Web applications is proposed. The development process starts from the describing of platform independent models. Mapping relations from the source model to the target model are built according to the syntactic structure and semantic features of both metamodels, and model transformation as well as code generation can be achieved subsequently. ASP.NET is used as a target platform in the experiment which shows that this approach follows the essence, process and requirements of model-driven architecture, and thus can make an effect support for model driven software engineering.
【Key words】 model-driven architecture; model description; model transformation; code generation

1 概述

高层模型的描述以及模型间的转换是模型驱动体系结构(Model Driven Architecture, MDA)[1]中的关键技术，但目前还没有一种有效的解决方案。文献[2]提出了一种基于 UML 的框架开发方法，对 UML 的概念和语义进行了扩展，并贯穿于软件开发的整个过程中，但缺乏一个清晰的软件体系结构描述。文献[3]提出用软件体系结构扩展 MDA 中的模型架构，以改善 Web 应用开发的质量和效率，但只提出了一个扩展的 MDA 架构，并没有具体给出模型的描述方式和模型间转换的方法。文献[4]提出的聚合体系结构方法系统地阐述了以软件体系结构风格为中心的软件开发，需依次经过聚合业务对象建模、聚合模式求精、UML 求精和代码生成等过程。该方法在开发初始阶段就引入了目标平台的建模风格，影响了 PIM 的平台独立性。一些基于 MDA 的类似产品，如 OptimalJ, Rational XDE 及 AndroMDA，所建模型也都不是 MDA 意义上的 PIM。

针对以上问题，本文从软件工程的实施出发，在软件体系结构的指导下建立平台无关的 Web 应用模型描述，基于源模型语义特征在目标语义域中的重新构造，依据转换两端建模元素的语法结构和语义表达特性定义模型间的映射规则，再经过一系列的模型转换，得到目标系统，完成软件开发。

2 平台无关模型的描述方法

本文以基于 UML 的 ASLP[5]作为源端模型的描述方法，以 ASP.NET 为目标平台进行实验研究。

ASLP 模型在传统应用系统建模的基础上对 UML 进行了扩展，并加入了界面展示视图。这个界面视图不是对界面展示元素的具体形式及属性的列举，而是对界面中抽象数据及行为元素的描述，同时也描述了界面元素与展示对象之间的对应关系，从而使得界面元素与具体应用平台无关，数据元素、行为元素与具体的界面展示元素相分离。使用 ASLP 方法可以为 Web 应用建立平台无关的模型描述，作为模型转换的源端。

ASLP 模型描述包含体系结构建模和构件建模 2 个层次。在体系结构模型中，系统(System)代表构件和连接器之间组合配置的拓扑结构和约束，定义为一个四元组：<S, D, C, R>。S 是体系结构风格，D 是系统的功能描述，C 是构件和连接器的列表，R 是构件/连接器之间的关系列表。构件由构件接口和构件内部实现模块组成。连接器是实现构件与构件之间信息交换和行为联系的特殊构件。

体系结构设计完成后，需要为每个构件制定或开发相应的实现构件。已存在的构件要尽量复用，不存在的构件要进行建模、开发。在体系结构的约束下，按照演化目标的要求，使用规范的 UML 描述构件和连接器：构件和复杂连接器映射为包(Package)；端口映射为抽象类(Abstract Class)；功能描述(Functional Specification)映射为接口(Interface)；将构件和构件之间(或构件与复杂连接器之间)的关联映射到类(调用者的接入点类和被调用者实现接口的类)之间的关联关系。ADL 到 UML 的映射关系如表 1 所示。

基金项目：山东省科技发展基金资助项目(2006GG2201009)
作者简介：张玉艳(1976－)，女，助理实验师，主研方向：软件工程；黄国栋，讲师、硕士；侯金奎，博士研究生
收稿日期：2008-08-17 **E-mail：**houjk@mail.sdu.edu.cn

表 1　从 ADL 到 UML 的映射关系

ADL	UML
Component	Package
Complex Connector	Package
Functional Specification	Interface
Entry Point	Abstract Class

构件的模型描述由功能视图、工作流视图、静态视图、行为视图和界面展示视图组成，这些视图以工作流视图为核心，分别描述了系统的不同方面，各个视图之间的关系如图 1 所示。功能视图描述了体系结构模型中构件的功能、系统与外部以及系统的各组成功能间进行信息交互的情况，用 UML 用例图描述。工作流视图用于为各种个体的行为建模，或者定义实体之间的交互和协作，使用 UML 中基于状态机的活动图表示。静态视图是 UML 中包图和类图的综合，主要描述功能视图中每个用例的分析类及其之间的联系。行为视图使用扩展的 UML 协作图对一系列对象的行为进行更细致的描述。界面展示视图是对行为视图中用户与系统的交互行为和边界对象的直观展示，提供了界面展示元素与行为视图中可见元素的绑定关系。其具体实现方法在文献[6]中有全面的介绍。

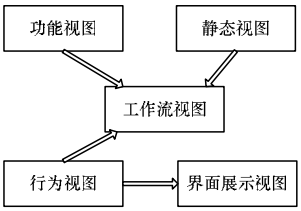

图 1　构件模型各视图之间的关系

3　模型转换和代码生成

3.1　目标平台分析

ASP.NET[7]是一个用于 Web 应用开发的框架，本文所述实验以 C#作为目标代码语言。基于 ASP.NET 平台的目标应用模型宏观结构如图 2 所示，它包括：(1)Web 页面(.aspx)，包含用户界面元素和基础代码；(2)代码隐藏文件(.cs)，实现代码和 Web 页面中用户接口的逻辑分离；(3)web.config，是应用程序所使用的基于 XML 的配置文件；(4)global.asax，是全局的应用程序文件，可以定义全局变量并对全局事件作出反应；(5)bin 目录，存储应用程序使用的.NET 程序集以及编译后的代码；(6)Web 工程文件(.csproj)以及可选的 Web 服务(.asmx)和用户控件(.ascx)。

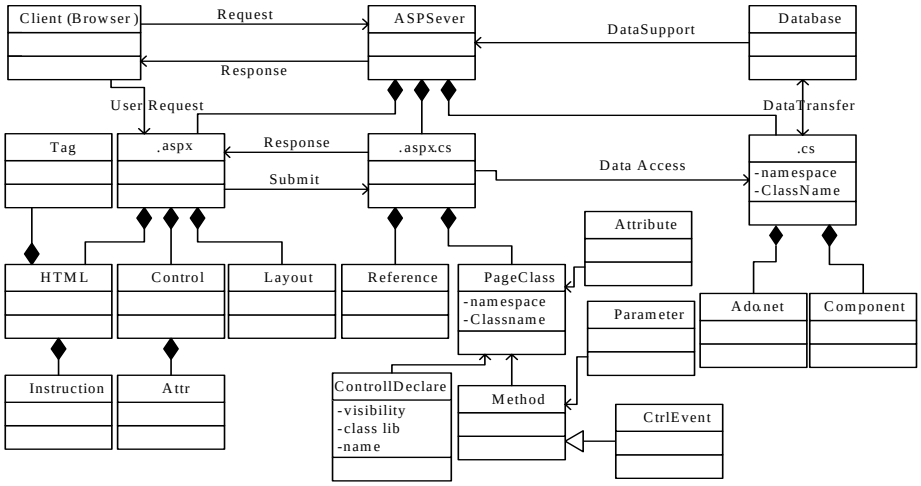

图 2　ASP.NET 目标模型结构

3.2　模型映射关系

模型间的映射可以看作是源模型语义域在目标语义域内的重新构造，即从源模型出发，通过对源模型的观察和推导得到有关的属性值，然后查寻这些属性值对目标模型定义要求的可满足性。设 C 表示目标模型中的模式概念集，$C=\{c_1, c_2, \ldots, c_n\}$；$O_1$ 表示显式观察的属性，即源模型中可直接观察到的属性；O_2 表示隐性观察的属性，这些源模型的属性需要由模式概念的语义上下文推出，即这些概念元素的必要条件$\{N_c|\ c\in C\}$；R 表示目标语义域中属性值的分类规则，即目标域分类的充分条件$\{S_c|\ c\in C\}$；M 表示模式之间的映射关系。那么映射问题可以表示成：为源模型概念元素 c_s 在目标域内找到 $c_i\in C$，使下式成立：

$$O_1\wedge O_2\wedge R\Rightarrow M(c_s, c_i) \quad (1)$$

式(1)形式地表达了映射问题，即给定一组源模式概念集 $C^S=\{c^S_1, c^2_1, \ldots, c^S_m\}$，使用目标语义域概念集 $C^T=\{c^T_1, c^T_2, \ldots, c^T_n\}$对其进行分类映射，这个映射过程由双方的语义特征所决定。源方提供对源模式的观察($O=\{N_c|\ c\in C^S\}$)，目标方提供目标模式和分类规则($C=C^T$, $R=\{S_c|\ c\in C^T\}$)。这样，一个模式概念元素就可以从源语义环境通过寻找满足式(1)的目标类 c^T_i 映射到目标语义模型中。

按上述原则，依据转换两端元模型的语法结构和语义表达特性定义模型间的映射规则，实现模型转换。基于 ASLP 的源模型到 ASP.NET 项目模型的映射关系如图 3 所示。

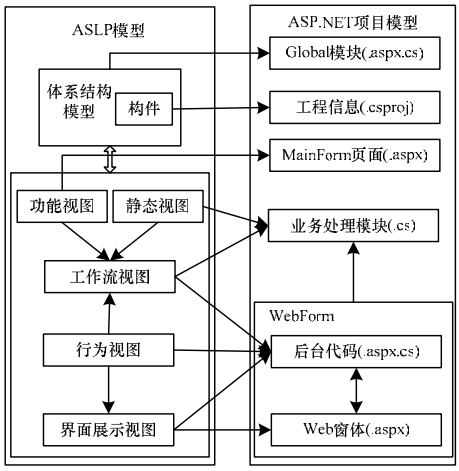

图 3　源模型到目标模型的映射关系

体系结构模型中的构件配置以及构件之间的关联信息映射至 Global 文件和部件的全局变量、公用函数中。体系结构模型中的工程部件映射为一个 ASP.NET 项目，构件中的工程信息与其他模型中的相关信息共同组合映射至项目的工程信息(*.csproj 文件)中。对象视图映射为所生成 ASP.NET 项目的业务处理模块，并提供对交互视图和展示视图的相应支持。功能视图中的复合用例映射为项目的功能选择项。交互视图结合展示视图中的模块处理信息映射为由展示视图生成的 Web 页面的后台处理代码(*.aspx.cs 文件)。交互视图中的用例映射为界面上的菜单、按钮或超链接等操作控制对象所对应的处理方法。方法调用关系映射为操作中对相应对象方法的调用。界面导航关系映射为对所导航页面的展示操作。展示视图中的模板对象映射为一个 Web 页面，模板对象中的信息映射为 Web 窗体展示元素的类型、位置、大小、颜色等属性信息(*.aspx 文件)。

3.3 代码生成

目标代码的生成算法可归纳为 4 大功能：目标代码框架的生成，构件代码的生成，展示界面代码的生成和界面(窗体)元素代码的生成，其流程如图 4 所示。

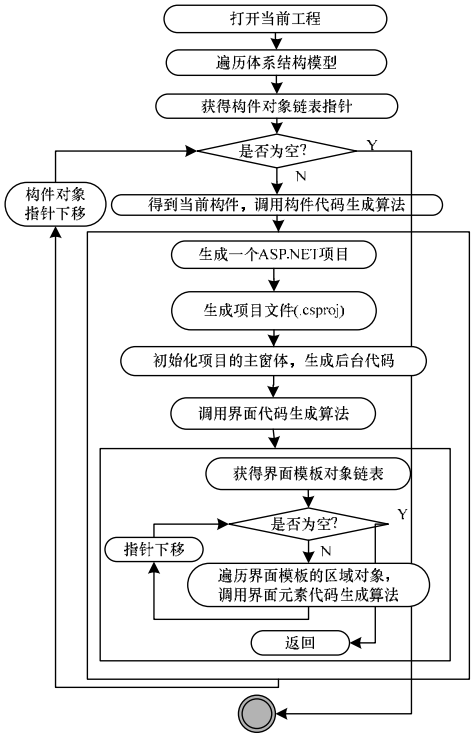

图 4　目标代码的生成算法流程

目标代码框架的生成主算法是整个代码自动生成的入口，从这里循环遍历整个体系结构中的对象，根据不同体系结构对象的类型生成不同的目标工程，然后对每个构件按照其类型调用相应的代码生成算法。

构件代码生成算法的主要功能包括：创建一个 ASP.NET 项目；根据体系结构模型生成工程的主模块及调用其他服务所需的文件；初始化目标项目的主窗体；遍历界面模板，调用展示界面生成算法等。创建一个 ASP.NET 项目的主要依据是体系结构模型，在体系结构模型中提取各个构件的信息，依据它们之间的关联关系创建 ASP.NET 项目的项目文件(*.csproj)，并生成其他项目所需的相关文件，如程序集文件(Assembly.cs)、配置文件(Web.config)、应用程序文件(Global.cs)、启动 URL 路径文件(csproj.webinfo)。对于引用的构件和连接器，如果是.NET 组件，只需在 ASP.NET 项目的项目文件中添加相应的引用。对于非.NET 组件，必须调用.NET 工具生成相应组件的包装组件，再将该包装组的引用件添加到 ASP.NET 项目的项目文件中。

展示界面的代码生成算法依据展示模型(界面模板)提供的信息生成用户界面，包括窗体框架、界面元素、界面元素布局等，同时根据界面模板对应的交互视图及对象视图的信息生成相应的后台操作代码。该算法的主要过程为，遍历界面模板的区域对象，根据每个界面展示单元展示的数据类型(如数据对象、汇集)和展示形式调用相应的代码生成算法。

界面元素代码生成算法比较繁杂。由于界面模板中的展示对象和具体 ASP.NET 环境中的展示控件之间不可能完全一一对应，因此需要根据展示形式将界面模板的展示对象映射为相应的 ASP.NET 控件。如果控件中还包含子控件，需要利用递归的方式来生成。

支持工具 AUI 的原型已经基本实现，主要功能包括：(1)图形化的体系结构建模；(2)体系结构模型到 OOD 设计模型的映射；(3)构件模型描述；(4)高层模型到目标平台模型的转换；(5)构件库的管理；(6)模型验证；(7)目标代码的自动生成。所有的程序都在 Microsoft Visual C++.NET 下实现。

4　结束语

本文提出了一种结合软件体系结构和 MDA 的 Web 应用开发方法。该方法保持了高层模型的平台无关特性，依据建模元素的组织结构和语义特性定义映射规则，通过模型间的自动转换完成软件开发。该方法遵循了 MDA 开发的实质、过程和要求，能够对模型驱动开发提供有力的支持。下一步的主要工作是对模型转换的形式化验证及对界面展示视图描述的进一步细化。

参考文献

[1] Miller J, Mukerji J. MDA Guide Version 1.0.1[EB/OL]. (2003-06-01). http://www.omg.com/mda.

[2] Yang Y J, Kim S Y, Choi G J, et al. A UML-based Object-oriented Framework Development Methodology[C]//Proc. of Asia Pacific Software Engineering Conference. Taibei, Taiwan, China: [s. n], 1998-12: 30-39.

[3] Meliá S, Cachero C, Gómez J. Using MDA in Web Software Architectures[C]//Proceedings of the 2nd International Workshop on Generative Techniques in the Context of MDA. Anaheim, California, USA: [s. n.], 2003-10: 76-82.

[4] Hubert R. Convergent Architecture: Building Model-driven J2EE Systems with UML[M]. New York, USA: John Wiley & Sons, 2002.

[5] 侯金奎, 万建成, 张玉艳. 一种支持MDA的PIM建模方法[J]. 计算机工程, 2007, 33(8): 71-73.

[6] 侯金奎, 张玉艳, 万建成, 等. 一种支持模型驱动开发的 Web 用户界面建模方法[J]. 计算机应用, 2006, 26(6): 1446-1448.

[7] Jeffrey R, Francesco B. Applied Microsoft .NET Framework Programming[M]. Washington, USA: Microsoft Press, 2003.

编辑　张正兴