

支持 MDA 的 Web 报表系统开发模型及其应用

张玉艳¹, 冯文堂², 黄国栋³, 侯金奎⁴

(1. 潍坊学院计算机与通信工程学院, 潍坊 261061; 2. 山东济南润丰农村合作银行, 济南 250001;
3. 中国人民解放军防化指挥工程学院, 北京 102205; 4. 山东大学计算机科学与技术学院, 济南 250101)

摘要: 依据模型驱动体系结构(MDA)的开发理念, 并遵循应用程序与用户界面设计相分离的思想, 以系统功能和界面描述的高层次抽象为基础, 提出一种支持 MDA 的 Web 报表系统开发模型。以 ASP.NET 为目标平台进行实验验证, 表明该方法遵循了 MDA 开发的实质、过程和要求, 能较好地与系统应用开发模型组合在一起, 提高 Web 报表系统的质量和开发效率。

关键词: 模型驱动体系结构; Web 报表系统; 模型转换; 代码生成

MDA-supported Development Model for Web Report System and Its Application

ZHANG Yu-yan¹, FENG Wen-tang², HUANG Guo-dong³, HOU Jin-kui⁴

(1. School of Computer and Communication Engineering, Weifang University, Weifang 261061;
2. Shandong Jinan Runfeng Rural Cooperative Bank, Jinan 250001; 3. Institute of Chemical Defense of PLA, Beijing 102205;
4. School of Computer Science and Technology, Shandong University, Jinan 250101)

【Abstract】 According to the theory of Model-driven Architecture(MDA) and following the idea of the separation between application descriptions and UI designs, a development model for Web report systems is proposed systematically based on the high-level abstract description of system function and user interface. ASP.NET is used as a target platform in the experiment which shows that this approach follows the essence, process and requirements of MDA. Great improvement of quality and efficiency of Web report system development can be obtained, for the model can be well combined together with application development models.

【Key words】 Model-driven Architecture(MDA); Web report system; model transformation; code generation

1 概述

报表系统在商务应用系统中是一个很重要而且使用非常频繁的子系统。采用手工方式开发嵌入式报表系统存在开发任务重、产品生命周期短等缺陷, 研究报表系统的自动生成可以减少系统开发的工作量, 使系统能够适应复杂多变的应用环境。当前基于 Web 的应用软件正逐步取代传统的 C/S 模式软件, 成为应用软件的主流, 报表系统也迫切需要适应这一趋势。现有的报表工具大多不具有学习功能, 通用性不强^[1], 在同一次运行时不能生成不同风格的报表, 无法满足根据用户需要实时生成报表的要求。当报表格式变化较大时, 难以满足用户动态添加新报表的要求。

模型驱动开发已成为软件工程技术的热点和发展趋势, 它通过模型和模型技术的应用, 提升抽象层次来应对软件开发的复杂性。OMG 提出的 Model-driven Architecture (MDA)^[2]为模型间的自动转换提供了理论支持。本文在深入了解传统报表生成方法和报表基本要求的基础上, 提出了一种支持 MDA 的 Web 报表系统开发模型。它能很好地与系统应用开发模型组合在一起, 同时具备报表生成简单直观、目标代码自动生成等特点。

2 支持MDA的Web报表系统开发模型

在前期模型驱动开发研究中所提出的 ASLP 模型^[3]的基础上, 依据 MDA 的开发理念^[4], 提出一种支持模型驱动开发的 Web 报表系统开发模型, 如图 1 所示。

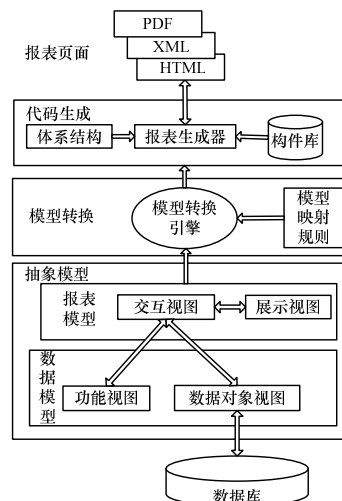


图 1 模型驱动的 Web 报表系统开发模型

开发模型主要包括数据模型、报表模型、模型转换引擎以及代码生成器 4 个部分。数据模型是整个报表模型框架的基础, 它将报表模块与实际的数据来源分离, 降低了报表模

基金项目: 山东省科技发展计划基金资助项目(2006GG2201009)

作者简介: 张玉艳(1976—), 女, 助理实验师, 主研方向: 软件工程; 冯文堂, 硕士; 黄国栋, 讲师、硕士; 侯金奎, 博士研究生

收稿日期: 2008-06-29 **E-mail:** houjk@mail.sdu.edu.cn

块与实际数据源的耦合度，并保证了具体报表定义知识的延续性，使得报表定义信息不再依赖于实际的数据库表。报表模型描述实现了用户对报表内容和报表格式的定制，为目标代码的生成提供全部的参数信息。模型转换引擎实现抽象模型到特定平台模型的转换，代码生成器根据其输出的结果生成最终的报表系统源代码。

2.1 数据模型

数据模型包括功能视图和数据对象视图两部分。功能视图通过对用户的需求分析，确定报表界面对内部模型的需求及界面之间的关系。数据对象视图对领域概念进行建模，描述了应用系统中使用到的类，它从静态方面描述了报表所需的数据对象以及它们之间的关系。

有关功能视图的介绍参见文献[3]，主要使用扩展的UML用例图来进行描述，并通过分析需求，确定界面功能、界面框架、界面之间的关系以及界面展示对内在模型的需求。

ASLP模型中的静态视图是从抽象的计算关系上描述对象的组成和行为，这样的描述不能满足建立界面结构的需要。为了满足对报表界面定制的需求，对其进行了扩充，建立了数据对象视图。除考虑对象的一般组成和行为外，还增加了服务于界面展示及代码生成的对象属性成分，并尽量减少对具体交互实现的约束。数据对象视图的扩充主要包括：(1)为提供界面上交互单元的信息，对属性扩充了UI类型、缺省值、标签和计量单位，并扩展了数据类型的描述；(2)为表现UI对象间的相关性和关联性提出了导出关系和联动关系；(3)为满足用户的心理和视觉特性提供了属性分组的描述；(4)使用对象间的关系表示界面的导航。

在以下的形式化描述中，用{}括起来的部分表示可选，用<>括起来的部分表示关键字，关键字后跟一个*号表示这个对象可能有0到多个，+号表示有1到多个，符号“:=”表示“定义为”。从而，类和对象的结构定义如下：

$\langle \text{Class} \rangle := \langle \text{ClassName} \rangle (\langle \text{Group} \rangle + | \langle \text{Attribute} \rangle +) + \langle \text{Method} \rangle * \langle \text{AttribLink} \rangle *$;

$\langle \text{Object} \rangle := \langle \text{ObjectName} \rangle : \langle \text{ClassName} \rangle$

其中， $\langle \text{ClassName} \rangle$ 为类名； $\langle \text{ObjectName} \rangle$ 为对象名； $\langle \text{Group} \rangle$ 是组； $\langle \text{Method} \rangle$ 是方法； $\langle \text{AttribLink} \rangle$ 是属性间的联动关系。

扩展的属性描述形式如下：

$\langle \text{Attribute} \rangle := \langle \text{AccessType} \rangle \langle \text{DataType} \rangle (\langle \text{AttribName} \rangle \{ \langle \text{Default Value} \rangle \} \{ \langle \text{ValueRange} \rangle \} \{ \langle \text{Unit} \rangle \} \{ \langle \text{DataSource} \rangle \} \{ \langle \text{UIType} \rangle \} \{ \langle \text{Label} \rangle \})$

其中， $\langle \text{Attribute} \rangle$ 表示属性； $\langle \text{AccessType} \rangle$ 表示运行时的可见性； $\langle \text{DataType} \rangle$ 表示数据类型； $\langle \text{AttribName} \rangle$ 为属性名； $\langle \text{Default Value} \rangle$ 为属性的缺省值； $\langle \text{ValueRange} \rangle$ 为取值范围； $\langle \text{Unit} \rangle$ 为计量单位； $\langle \text{DataSource} \rangle$ 为值来源； $\langle \text{UIType} \rangle$ 为可视对象类型； $\langle \text{Label} \rangle$ 是可视对象所附带的标签。

笔者还在模型中增加了属性分组的概念来支持界面生成，以综合考虑界面的布局信息。同一组的属性可以使用Frame框组合起来作为一个整体。在复杂的情况下，分组内部还需要划分子组，形成嵌套关系。属性分组是子组和属性的聚集，因此用下面的方式表示：

$\langle \text{Group} \rangle := [\langle \text{GroupName} \rangle \langle \text{Attribute} \rangle +] [\langle \text{GroupName} \rangle \langle \text{Attribute} \rangle + | \langle \text{Group} \rangle +] +$

其中， $\langle \text{GroupName} \rangle$ 代表组名； $\langle \text{Attribute} \rangle$ 代表属性。

2.2 报表模型

报表模型包括交互视图和展示视图两部分。交互视图从

动态方面描述系统，是对用户与系统交互的详细描述。它也是用户界面行为的抽象描述，提供了用户界面行为与系统功能的内在关联。界面展示视图的功能是根据内部模型(数据对象视图和交互视图)和用户对数据的展示要求给出界面的布局和展示形式，它是对界面直观展示的全面描述，同时提供了界面元素与交互视图中可见元素的绑定关系。

交互视图在整个框架中起着承上启下的作用，如图2所示。一方面，交互视图与功能视图和数据对象视图衔接，把界面展示所需的功能和对象有机地结合在一起，形成界面的抽象轮廓；另一方面，交互视图的分类又是展示模型中界面模板^[5]检索的依据，即交互视图中对象与对象之间的交互关系决定了界面模板的类型，为用户选择合适的界面提供了约束，保证了界面生成的正确性。另外，交互视图的抽象程度比功能模型和对象模型都低，与它们相比，交互模型更接近于界面。

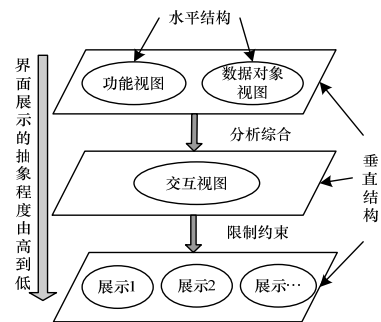


图2 交互视图的作用

交互视图中的数据对象是对象视图中定义的数据对象的实例化。对象视图中的数据对象是类的概念，而交互视图中的数据对象则是应用于具体界面的类实例。交互视图在对数据对象视图中的对象进行实例化的同时，根据其对面表达构成的作用不同又可进行进一步细分，以确定恰当的展示形式。交互视图定义为一个三元组 $\langle V(G), E(G), \varphi_G \rangle$ 。其中， $V(G)$ 是交互对象集合； $E(G)$ 是交互关系集合； φ_G 是从交互关系集合 E 到交互对象序偶集合上的函数。

交互对象是界面上可以进行交互的实体对象，包括数据对象、汇集、用户、用例、页面引用等。交互视图中的数据对象源自于数据对象视图，服务于与该交互视图所对应的展示界面。它被展现到界面后，对应的界面元素可以是各种形式的导航链接，以及各种表单，如文本框、密码文本框、单选按钮、复选框等。交互关系是交互对象之间的关联关系，它强调对象与用例间的交互行为及其对界面对象间关系的影响，包括调用、参与、消息连接和导航4种。调用关系指一个对象激发另一个对象的功能方法并返回结果的行为关系。在交互视图中，用户到用例之间是用例调用的关系，该关系反映在界面上就是用户点击按钮；用例与数据对象之间的调用，通过引发数据对象或汇集的行为建立用例与其操作数据之间的关联。参与关系是表达数据提供的一种关系，一般由对象或汇集指向用例，表示向用例提供数据或参数。消息连接关系表达对象与可见的数据对象或汇集之间的消息传递，也可以是事件触发器到部件、可视的对象或汇集及用例到部件的消息传递。导航关系指用例激发出导向某个页面的操作。

界面展示视图为界面的布局提供了一个约束展示的环境。交互视图是界面展示视图的内在基础，是对界面内在的全面描述。而展示视图是交互视图的外在描述，是在交互视

图基础上对界面外观形式的描述,主要处理界面宏观布局和表示控制,服务于图形用户界面自动化生成的目标。它依据界面的内部模型把抽象的界具体化,解决界面的整体布局问题。展示视图主要包括模板对象、区域对象、分割线对象和模板交互对象等。

3 面向ASP.NET平台的模型转换和代码生成

ASP.NET^[6]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。

3.1 模型映射关系

报表模型到 ASP.NET 项目模型的映射关系如图 3 所示,对应生成的目标代码主要包括工程信息、业务处理模块、Web 窗体和相应的后台代码。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)。

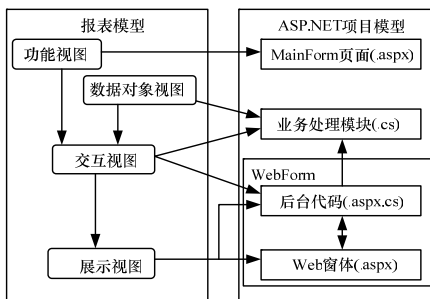


图 3 报表模型到目标模型的映射关系

3.2 代码生成

目标代码的生成算法主要包括报表框架的生成、报表界面元素的生成和报表打印及预览程序的生成。报表框架的生成算法是整个代码自动生成的入口,用来生成用户报表界面代码的最上层算法。它依据界面模板提供的信息来生成用户报表界面、报表界面元素以及界面元素布局等,同时根据界面模板所对应的交互视图生成相应的后台操作代码。该算法流程如图 4 所示。从这里循环遍历界面模板中的区域对象,并根据每个界面展示单元所展示的数据类型和展示形式调用相应的代码生成算法。

报表界面元素的生成算法是用户报表界面基本元素生成的算法,主要用来生成具体的界面控件。这是一个递归算法,因为有些控件还可能包含子控件,如 Frame、Tabstrip 等。

报表打印及预览生成算法主要用来生成报表的打印预览程序。打印预览模块是报表系统的又一个核心模块,它是整个系统的输出部分。目前在 .NET 平台下,实现报表打印预览的基本过程是:先定义好整个打印页面的布局,然后定义各个打印区(Rectangle 对象),包括打印区的尺寸、需要打印的文本、图形、图像等,最后调用系统提供的 DrawString(),

DrawRectangle()方法来输出相应的文本、图形、图像即可。

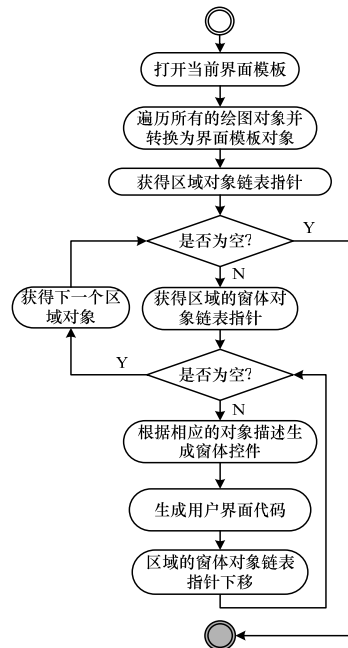


图 4 报表框架代码生成算法流程

4 结束语

本文提出了一种支持 MDA 的 Web 报表系统开发模型。该模型能很好地与系统应用模型结合在一起,为应用程序提供报表功能并具有一定的软件复用能力,能满足用户动态添加新报表的要求。与基于形式化语法和规范语言的自动生成方法相比,本文的方法直观且易于理解,因而提高了应用系统的质量和开发效率。它不仅可以减少报表系统开发的工作量,使系统能够适应复杂多变的应用环境,而且便于适应目前种类繁多、规模各异的数据库管理系统的变化,满足企业系统不断更新的需求。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。

今后的工作包括:(1)进一步完善交互视图的描述,增强其语义表述能力;(2)对界面展示模型进行全面的抽取和描述,以加强所生成页面的视觉吸引力;(3)目标平台的多样化以验证该方法的实用性。

参考文献

- [1] 刘 罡. 基于 XSL-FO 的 WEB 数据库报表实现[J]. 计算机应用, 2003, 23(5): 131-133.
- [2] Miller J, Mukerji J. MDA Guide Version 1.0.1[EB/OL]. (2003-06-01). <http://www.omg.com/mda>.
- [3] 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.
- [4] Kleppe A, Warmer J, Bast W. The Model Driven Architecture: Practice and Promise[M]. Boston, USA: Addison-Wesley, 2003.
- [5] 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.
- [6] Richter J, Balena F. Applied Microsoft .NET Framework Programming[M]. Washington, USA: Microsoft Press, 2003.

编辑 顾逸斐