第 36 卷 第 2 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.2 Computer Engineering January 2010
·软件技术与数据库· 文章编号：1000—3428(2010)02—0042—03 文献标识码：A 中图分类号：TP311

可视化业务模型的设计与实现

陈海波，董建明
(浙江理工大学理学院，杭州 310018)

摘 要：针对 xUML, OCL, ASL 等模型驱动的执行工具过于侧重语义完整，不够直观，使得设计人员难以理解的问题，在 MOF 第二层基础上改造 UML 元模型元素，建立一种支持人机交互的元模型可视化动作模型(VAM)，在此基础上实现一个 VAM 执行引擎。模型的执行过程和实际代码的执行过程相似，提高了模型驱动开发在需求验证和测试中的可用性。

关键词：模型驱动架构；模型元素；可视化动作模型

Design and Implementation of Visuable Business Model

CHEN Hai-bo, DONG Jian-ming
(School of Sciences, Zhejiang Sci-Tech University, Hangzhou 310018)

【Abstract】 For the question that the executable tools of Model Driven Architecture(MDA) such as xUML, OCL, ASL are too emphasis on the semantic integrity to observe and understand for designer, a human-computer interaction supported meta model Visual Action Model(VAM) is proposed by reconstructing UML model element in MOF2, and an executable engine of VAM is developed, which makes the process of the model code and the process of actual implementation code very similar, so as to enhance availability of requirement validation and test in model-driven development.

【Key words】 Model Driven Architecture(MDA); model element; Visual Action Model(VAM)

1 概述

模型驱动[1-2]开发的核心目标是将模型语言作为主体编程语言，而模型定义技术成为模型驱动开发是否能被业界实际采用的关键因素。模型在理论上被看作是一种形式化语言[3-5]，具有完备的语义描述，能够在语义不变的前提下进行不同模型元素之间的转换。同时，模型还作为需求分析、系统设计、验证和测试的工具。然而，模型的这 2 个作用之间存在着不可避免的鸿沟，前者将模型看作数学对象，后者将模型看作业务领域的对象、关系和活动。两者的结合需要实现业务领域内容的形式化，这是一个工程浩大且极其复杂的工作，尽管这方面已经有了一些相关的理论、方法和工具[6]，但是还远没有达到实用的程度，从而阻碍了模型驱动在工业界的推广和普及。

可以考虑暂时绕开如何将现实世界的各种实体和复杂关系全面形式化这一问题，将形式化模型中的模型元素与可视的人机交互界面元素关联起来，不追求形式化验证的自动化和完整性，设计者可以自己操作、干预模型的执行，改变执行流程，自己发现错误并修正模型。

UML2.0 标准中模型元素在执行过程中与人机交互界面的关联有一定困难，主要有以下几点原因：(1)UML2.0 中没有直接表示交互界面的模型元素，界面与普通的类同等对待，这给模型执行过程中界面的事件处理带来困难。(2)交互界面本身不是模型的内容，模型执行过程中不需要对人机交互过程进行验证，需要将交互界面与领域模型中的元素区别开来。

因此，在 MOF 第二层的基础上对 UML 模型元素进行了改造，建立了新的可视化的业务执行模型，称为 Visual Action Model(VAM)，并在 VAM 基础上实现了一个模型执行引擎。引擎对模型的执行过程与实际代码执行过程很相似，设计者可以通过不断干预模型的执行过程来验证模型。

2 VAM 元模型

2.1 VAM 定义

VAM 处于 MOF 第二层，属于元模型范畴，定义如图 1 所示。

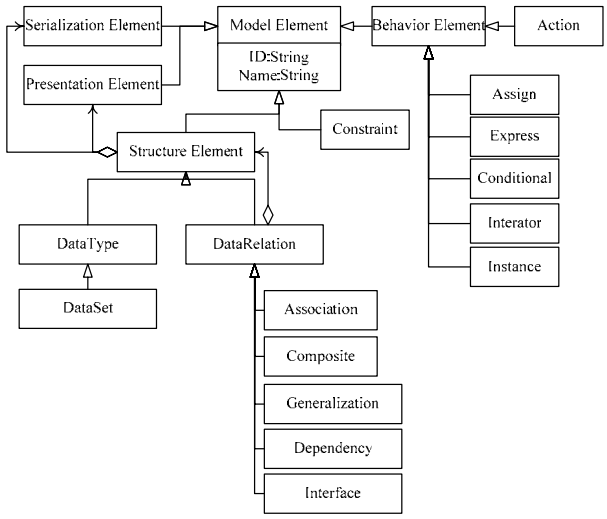

图 1 VAM 元模型

2.2 模型元素

VAM 中有 4 个基本的模型元素，分别是结构元素

基金项目：浙江省教育厅研究基金资助项目“模型驱动体系中可编译类模型的行为机制研究”(20060579)
作者简介：陈海波(1972－)，男，副教授、博士，主研方向：模型驱动，智能计算；董建明，讲师、博士
收稿日期：2009-07-30 **E-mail：**chenhb@zjip.com

(Structure Element)、行为元素(Behavior Element)、表达元素(Presentation Element)和存储元素(Serialization Element)。与MOF 标准元模型相比[2]，主要的差别是增加了存储元素和表达元素，前者用于实现模型元素的序列化；后者实现模型元素的实例的可视化。

在 MOF 模型中没有表达元素，实际上涉及 UI 的类被当作普通的模型元素的实例来处理。而 VAM 中将数据显示作为一个独立的模型元素，使得模型中每一个类和对象都能直接与可视化的界面相关联，在进行模型设计的时候能直观地看到模型执行的效果。这样做在一定程度上丧失了形式化的完备性，但增强了模型设计的实际效果。这是 VAM 和 MOF 的最大区别。

2.3 结构模型元素

结构元素是抽象元素，在实际建模中使用 DataType 作为基本的模型元素。一个 DataType 元素包含以下这些成分：(1)DataType 成份，在 VAM 中没有属性的概念，即属性也是 DataType。(2)DataRelation，说明 DataType 成份与待描述的 DataType 之间的关系，和 UML 一样，关系可以是关联、聚集、组成、依赖、范化和实现几种。(3)前两者的约束，使用 OCL 表达约束。

2.4 行为元素

行为元素由一些基本元素和动作组成，基本元素包括了赋值、表达式计算、条件分支、迭代和实例化；动作(Action)是基本元素的有序集合，动作可以被看作是通常计算机程序设计语言中的函数，其中，AsyAction 可以看作是异步执行的函数。

2.5 表达元素

模型显示的重点是表达元素。一个表达元素可以关联一个或者多个结构元素，用于描述结构元素是如何显示在界面上的。

在 VAM 中表达元素与结构元素有较大的不同，表达元素具有属性、方法，而结构元素没有；实际上，在形式化验证和求精过程中只需要 DataType 和 DataRelation 就可以了，而表达元素涉及到各种具体的 GUI 库，其设计和实现千差万别，如 AWT, SWT, WPF 等，是难以进行形式化的部分。

表达元素并不是现有的各种可视控件的元模型，因为现在的各种控件大多缺少表达元素的第 3 部分描述，即缺少表达元素与结构元素之间的内在关联。例如，.NET 可视控件 DataGridView 能够与指定的某个 Collection 相关联，但是 Collection 却没有指定固定的模型元素，因为缺少元模型层的支持。

3 执行引擎设计

3.1 体系

执行引擎是按照模型中预定义的流程对模型进行仿真的软件系统。VAM 执行引擎的作用主要是系统仿真和需求改进。图 2 是 VAM 执行引擎的内部结构示意图。

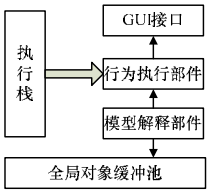

图 2 VAM 执行引擎结构

VAM 执行引擎使用 C++开发，由 3 个执行部件和 2 个关键数据结构组成。其中，模型解释部件负责读取和分析模型文件的格式和内容；行为执行部件负责按照模型中定义的动作依次执行，执行过程中需要显示用户界面的部分，则调用 GUI 接口部件，访问操作系统。在行为执行过程中所有在行为外部创建的对象将放置在全局对象缓冲池中。实际上全局对象缓冲池中的对象状态就是整个系统的状态。执行栈则记录行为执行的过程与状态。

3.2 人机交互实现

行为执行部件依次负责解释执行模型中的每一个 Action，当 Action 中行为涉及表达元素的属性和方法的时候，行为执行部件调用 GUI 接口产生实际的 GUI 控件，目前的 VAM 执行引擎只支持.NET 可视控件。为了简化引擎的实现，在实际建模过程中，表达元素的实例都与.NET 可视控件的属性和方法一致。GUI 接口通过类反射机制调用相应的.NET 控件。

4 应用案例

4.1 应用背景

下面给出一个 VAM 的实际应用案例，以说明 VAM 的可用性。健康评估系统是一个与病人之间的人机对话系统，并将对话的结果反馈给医生，从而得到健康状况的一个综合评价。一般来说，人机对话的内容包括选择性提问、填空性提问、教育(html 格式的文本)、病情指标监测(通过联机的医用设备进行)。由于评估过程的复杂性，人机对话的内容往往不是固定的，即一个对话执行完毕后，需要根据病人的回答情况决定执行下面哪个或者哪几个对话(类似程序设计的分支语句)；如果病人前面的一些对话自相矛盾，还需要反复进行前面已经执行过的对话(类似程序设计的循环)。

4.2 模型定义

根据以上这些情况，可以使用 VAM 进行评估系统的模型设计。模型包含以下内容：

(1)对话类、结果类和任务类

模型中静态部分的核心类是对话，它是一个结构型元素的实例，描述一次医生与病人之间的人机对话；任务类是 DataSet 的实例，任务类是对话类的集合；结果类记录人机对话的结果。

(2)对话及任务的执行

模型中动态部分核心类是对话的执行，它是一个 Action 模型元素的实例，这个 Action 输入一个对话，输出一个结果。

健康评估过程实际上就是任务的执行，被定义成 TaskAction，任务的执行被分解为对话的执行，C#语法表示的 Action 代码如下所示：

```
//任务执行动作，它将改变全局缓冲池中 gResult 的状态
void TaskAction()
{//取得任务的第 1 个对话
Dialog dlg = gTask.root;
do{
//满足动作的触发条件，则执行 DialogAction
if(Trigger(dlg))
  gResult.last.Next= DialogAction(dlg);
}while(dlg = gTask.next);//取任务的下一个对话
}
```

TaskAction 没有输入和输出元素，其内部结构是若干个行为型元素的实例组成，包括 3 个 Assign 元素、1 个 Iterator 元素和 2 个 Action 元素。其中，gTask 是全局对象缓冲区的任务对象，表示当前任务；gResult 是全局对象缓冲区的一个

DataSet 型的类的实例，表示当前任务执行结果的集合。

```
//对话执行动作
Result    DialogAction(Dialog dlg)
{
// FormView 是表达型元素的实例，而 view 是 FormView 的实例
FormView view = new FormView(dlg);
view.Show();
view.waitUser();
return new Result(dlg,view);
}
//对话的编辑视图
Dialog DialogEdit()
{
// FormEdit 是表达型元素的实例，而 editor 是 FormEdit 的实例
FormEdit editor = new FromEdit();
editor.Show();
editor.waitUser();
return new Dialog(editor);
}
```

(3)执行的可视化

执行引擎可以根据模型中结构型元素与表达型元素之间的关联自动显示 GUI 界面，同时显示结构型元素的内容。本例中一个对话类(Dialog)关联到 FromView, FormEdit 这 2 个表达元素，分别用于对话的显示并等待用户输入和对话内容的编辑。DialogAction 动作中会显示 FormView，而 DialogEdit 动作负责对话内容的编辑。

4.3 模型的执行和验证

上述定义的模型可以直接执行。可以在模型编辑器中直接执行，图 3 分别是模型中对话执行的视图和对话编辑的视图。这些视图是在模型设计阶段产生的，而不是最终系统的执行结果。但是可以看出，模型的执行过程已经十分接近实际系统的执行了。

(a)对话的编辑视图 FormEdit

(b)对话的执行视图 FormView

图 3　模型可视化视图

设计者可以通过这种可视化的模型自己验证模型的正确性，例如在 DialogAction 动作的执行过程中，很容易由于病人对话中的输入错误，导致了实例化可能不是一个正确的对话结果，因此需要引入一个新的动作用于校验病人对话中的输入，并引入一个新的结构型元素的实例 Regular，用于表示合法的输入，其代码如下所示：

```
Bool Verify(Dialog dlg,FormView)
{
  if(gRegular.Match(view.input.Text))
    return true;
  return false;
}
```

5　VAM 与 xUML 的比较

VAM 与 xUML 有较大的区别，主要包括以下几点：

(1)xUML 是用构造型来扩展 UML，保证了 UML 的完整性和一贯性，但是在实际应用中，必须把所有 GUI 的组件全部形式化后才有可能在模型执行过程中显示界面；VAM 则是直接引入了新的模型元素：PresentationElement，并通过执行引起将模型元素的实例直接映射到.NET 的可视组件库中，从而直接实现了模型执行过程的可视化。

(2)xUML 中模型元素包括属性；VAM 中没有属性的概念，属性就是 DataType 元素，例如一个对话动作类中不会有“属性 ID”、“类型是 int 型”这样的描述，而是对话类动作中有一个子 DataType 的实例，类型是 ID。这样的描述方式大大简化了执行引擎的实现，因为结构型元素实际上就是其他结构型元素以及它们之间关系所构成的集合。

6　结束语

从侧重模型执行的可视化效果，增加模型设计阶段设计者自我验证和修正能力的角度出发，本文提出了一种可视化的业务元模型 VAM。用 VAM 设计的实际系统模型是一种可以由 VAM 引擎执行的模型，且执行的过程不再是抽象的形式推导过程，而是有直观的用户操作界面，与最终系统的执行界面十分相似，使得设计者提前仿真了系统，通过仿真过程不断修正自己的模型设计。

VAM 模型的缺点是与 UML 不一致，各种基于 UML 的模型转换工具不能使用，缺少从模型到自动代码产生的过程，如何弥补这一不足是下一阶段的工作目标。

参考文献

[1] OMG Architecture Board ORMSC. Model Driven Architecture (MDA)[EB/OL]. (2001-07-01). http://www.omg.org/cgi-bin/doc?ormsc.

[2] Jouault F, Kurtev I. Transforming Models with ATL[C]//Proc. of the Model Transformation in Practice Workshop. [S. l.]: Springer-Verlag, 2005.

[3] Whittle J. Transformations and Software Modeling Language: Automating Transformations in UML[C]//Proc. of the 5th International Conference on Unified Modeling Language. Dresden, Germany: Springer, 2002: 227-242.

[4] Mens T, Demeyer S, Janssens D, et al. Formalizing Refactorings with Graph Transformations[J]. Journal of Software Maintenance and Evolution: Research and Practice, 2005, 17(4): 247-276.

[5] Song Dan, He Keqing, Liang Peng, et al. A Formal Language for Model Transformation Specification[C]//Proc. of the International Conference on Enterprise Information Systems. [S. l.]: Springer, 2005.

[6] Mahony B, Jin Song Dong. Blending Object-Z and Timed CSP: An Introduction to TCOZ[C]//Proc. of the 20th International Conference on Software Engineering. Kyoto, Japan: IEEE Computer Society Press, 1998.

编辑　顾逸斐