

Model Based Rapid Application Development

Netta M. Shani¹

Shiri Davidson

IBM Haifa Research Lab

Abstract

Web applications become increasingly common as means for accessing online applications. These applications involve both client side and server side components. Therefore it is important to model both components in a single model i.e., introduce an application end-to-end model. Such a model should contain all four application layers: data, user interaction, flow, and business logic.

Many published models and standards approach a single layer or a subset of these layers. They do not aim at modeling a complete end-to-end application.

This paper presents a programming model that represents a complete end-to-end application. It represents an application at a high level of abstraction and thus frees the user from low level implementation related details. Therefore it is appropriate for rapid application development (RAD). A RAD environment that is developed on top of this model is briefly described.

Introduction

Programming models of Web applications and related standards are well discussed in recent years publications. This is a result of the increasing use of Web applications as means for accessing online applications. These models do not consistently cover a complete end-to-end application, but parts of it.

A few publications [1], [2] model user interface in a device independent way. The purpose of these models is to allow the development of a single user interface to be deployed on various target devices either by automatic generation of several applications, one per platform, or by the support of an appropriate runtime. XForms [3] is a W3C standard aimed at modeling Web application forms. It defines data and related form controls to describe a Web form in a platform independent manner. Java Server Pages (JSP) [4] model user interface at a very low level. They allow integration of some logic and data related code into the definition of user interactions.

The Web Application Descriptor model (WAD) [5] represents application flow. This allows centralized application flow to replace flow definition that is normally scattered in various Web pages of the application. This support is more significant when the development of business logic and the development of user interface are done by different people. Web Application Development Support Tooling (WAST) [6] is a development tool, developed on top of WAD.

The Model View Controller (MVC) design pattern defines an application as three component structure: model that encapsulates application state, view that provides model presentation and controller that handles application flow. A number of runtime frameworks, such as Struts [7], are currently emerging to provide runtime support for MVC based applications.

¹ Contact: netta@il.ibm.com

The model and the development environment that are described in this paper are currently developed at IBM Haifa Research Lab. They have evolved from previous work done in the area of tools for pervasive devices [8], [9].

This model introduces an end-to-end modeling of Web application. This model covers all application layers for both client and server side components. This distinction between the client and the server components is not explicitly represented in the model. The model reuses existing models and standards whenever adequate in order to avoid the introduction of proprietary model parts when those exist. Examples are the use of XForms in the user interaction layer and the use of RDB model in the data layer.

RAD Programming Model

The programming model that is described in this section represents an application and aims at RAD. It was kept at a high level of abstraction, simple yet powerful in order to support efficient application development. These properties make it suitable for RAD.

The model defines all application layers: application data, user interactions, application flow, and business logic. The main building blocks of an application according to this model are data sources that introduce the data model, interaction units that encapsulate user interactions, application components, and business logic units that encapsulate business logic.

The model described in this paper represents all the information that is required for an automatic generation of end-to-end complete viable application. As the model is defined to be platform independent it may be used for the generation of an application for any target platform.

Data Model

This model defines a generic data model, the main entity of which is a Data Sources. Data Source is a declarative entity that represents a data source as a set of primitive data entities and a set of actions that can act on this source.

Primitive data is represented by Fields. We distinguish between Persistent Fields that represent data mapped from persistent data source, and Transient Fields that represent variables, which are defined within the application for volatile, internal use. Each Field references a Type Descriptor, which represents its type.

Actions are represented by Action Descriptors that define the return type and parameter types of the action to be executed.

Each Data Source is mapped to a data source using data domain mapping objects. These mapping objects allow bridging between the generic model described here, to various commonly used data models. Thus when an RDB table is represented, its corresponding object in our generic data model is mapped to its representation in the RDB object model. No further modeling is done and thus there is no model redundancy.

When a Data Source represents a Relational Database, each of its persistent fields represents a table column. Additionally, a basic set of action descriptors (Action Descriptors) are defined to represent SQL statements that can be activated on the Data Source. Users may define additional Action Descriptors to operate on the Data Source.

When a Data Source is mapped to a Java Bean, its Persistent Fields represent bean data members and the Action Descriptors represent bean public methods.

User Interactions

User interactions are encapsulated into Interaction Units (that, in the context of Web applications represent Web Pages). An Interaction Unit may contain Views and push buttons. This model defines three types of Views: form, list and menu.

A view presents the data that is defined by a single data source. This data is represented by a Data Source Variable that may be manipulated by the application business logic.

The widgets that are used to construct views are XForm based and as such are kept to be platform independent. This allows a clear distinction between the application programming model and the target runtime platform that is targeted by the application.

XForms is a standard that defines the representation of form based user interactions. This model uses user interface controls, defined in XForms. Thus we define textbox, exclusiveSelect, multipleSelect, secret, output and button.

Pure visual characteristics of the displayed objects, such as layout parameters and colors, are intentionally not modeled. It is assumed that these characteristics are beyond the scope of this model and are handled by existing tools dedicated for this purpose.

Application Flow

A Component represents a component of an application and thus allows application modularity. It contains a hierarchical flow graph. Each node in this graph, according to its type, may have entry and exit points that represent the state upon which the object may be entered or exited correspondingly. The flow between nodes is represented by Links, each connecting node's exit points to node's entry point

A Component may contain other Component objects, Interaction Units and Business Logic Units.

Business Logic Entities

This model is aimed at keeping the representation of business logic as simple as possible and yet rich enough to be sufficiently expressive. Our purpose was to define a set of core actions to enable performing logic that is capable enough to support common applications, while keeping it as simple as possible in order to avoid introducing yet another full fledged programming language.

Business logic is encapsulated into Business Logic Units. Each unit contains a single main control block composed of control structures, and Actions. Actions are the atomic execution unit in this model. They are defined based on Action Descriptors, which represent the action signature. Actions defined within a Business Logic Unit can use expressions as well as variables (aforementioned Fields and Data Source Variable) defined within its scope (locally or in one of its containing entities).

The control structures that are defined in this model are Sequence, Condition and Iteration. Actions are of two types: core actions – a minimal set of Actions that are pre-defined in the model, and domain actions – those Actions that are introduced by Action Descriptors within Data Sources. Core actions are Set Variable, Set State, Disable Widget etc. Domain Actions introduce the major part of expressiveness into the model.

This model defines two additional business logic related entities that are contained within Component and Interaction Unit: Initiator and Concluder. An Initiator is defined to be activated when flow enters its entity and a Concluder is defined to be activated just before flow exits the its entity regardless its exit state.

Model Extensibility

An important property of this model is its extensibility. The core model may be extended in different ways in order to conveniently represent a large scope of applications. These extensions impose additional expressiveness to the core model and still leave it generic and clean.

The generic data model that maps to standard, existing data models, may be extended to additional domains such as Java beans and Web services. This is achieved by extending the mapping mechanism that is defined in the core model to support mapping of additional domains as currently done for RDB.

Another way to extend the core model is by introducing additional models that reference core model entities. An example is the definition of Roles model. Each object in this model contains Role information and references a corresponding object in the core model. This way objects of the core model may be categorized by their assigned Roles. This can be used to represent Business Process Models [10], in which roles are essential. The same Roles extension can be used to represent authentication information, different locales etc.

Additional extension can be introduced by inheritance. Domain that specializes model entities at a finer granularity compared to the core model, may add subclasses to classes that are defined in the core model. For example, domain that specializes Business Logic Units according to their context in the application, may specialize the Business Logic Unit class that is defined in the core model.

Model Based RAD Tool

Based on the model that was described in the previous section, we created a development tool that supports an easy, fast, efficient development of Web applications. This tool is developed as a plugin to IBM Eclipse framework [11]. It lets the user build end-to-end applications that utilize legacy systems from various source domains, such as Java Beans and relational databases (e.g., DB/2, Oracle). These are available to the developers via a general extension mechanism that enables the definition of new domain specific plug-ins that the environment exploits (such as EJBs and Web services). All source domains are mapped to the generic data model.

A Web application is composed mainly of Application Components, Web Pages (Interaction Units), Views (Form, List and Menu) and Business Logic Units. The user of this tool performs top-down design, starting at the design of the component that represents the complete application and recursively designing and configuring the internals of each entity. He is provided with visual design editors and with various configuration workbooks.

This environment allows the development of JDBC based Web application (DB/2, Cloudscape, Oracle, etc.) by a person that has no understanding of Web technologies (such as JSPs, Servlets, Java beans) and no understanding of the low level JDBC technologies. The developer may work at a level of abstraction that corresponds to his technical qualification and use terminology he is familiar with, while the tool automatically handles the low-level technical details.

This tool allows an enterprise to significantly reduce the level of expertise of its developers and still maintain the development of high quality applications in a short time to market. An end-to-end complete viable application is generated at the end of the design phase. This implies generation of significant quantity of high quality code.

An advanced user may edit the generated artifacts and edit them using standard development tools (Page Designer, Java IDE etc.). The modifications will be maintained in the next code generation.

Summary

This paper presents a programming model that represents an end-to-end application. This model was kept at a high level of abstraction, simple yet powerful in order to support efficient application development. These properties make it suitable for RAD.

A development environment for developing Web applications that is based on this model is briefly described. This environment allows simple application development, which yields the generation of a complete, end-to-end viable application.

References

-
- [1] Sussman, Banavar, Bergman, Generalization: A key concept in the creation of platform-independent user interfaces, Workshop on Application Models and Programming Tools for Ubiquitous Computing, Atlanta, 2001
 - [2] IBM Dharma: http://www.trl.ibm.co.jp/projects/dharma/index_e.htm
 - [3] XForms 1.0, W3C Working Draft, <http://www.w3.org/TR/xforms/>, 2002
 - [4] Java Server Pages, Sun Microsystems, Inc., <http://java.sun.com/products/jsp/>, 2002
 - [5] Tai, Nerome, Hori, Web Application Descriptor (WAD), Research Report, IBM Tokyo Research Lab, 2002.
 - [6] Hideki et al., Model-Driven Development of Dynamic Web Applications, submitted to Extreme Markup Languages, 2002
 - [7] The Struts Web application framework, Apache Software Foundation, <http://jakarta.apache.org/struts/>, 2002
 - [8] Shani, Soroker, The 2nd IBM Research Pervasive Computing Conference, NY, 2000
 - [9] Soroker, Shani, Rubin, RAD Tooling for Pervasive Applications, Workshop on Application Models and Programming Tools for Ubiquitous Computing, Atlanta, 2001
 - [10] <http://www.research.ibm.com/EnterpriseBuilder/EBhome.htm>, 2002
 - [11] Eclipse Platform, eclipse.org consortium, <http://www.eclipse.org/>, 2002