TUHH
Technische Universität Hamburg-Harburg

Generation of Struts software artefacts from UML stereotypes

Project Work

as a requirement of master's programme in
Information and Media Technologies

Submitted by:
Andriy Dekhtyar
andriy.dekhtyar@tu-harburg.de
Matriculation number: 23818

Supervised by:
Prof. Dr. J. W. Schmidt
STS - TUHH

M.Sc. Miguel Garcia
STS – TUHH

Hamburg, Germany
February 2004.

Abstract

Subject domain: Modeling, Web Applications, MDA.

Problem description: This work is devoted to modeling of web applications based on Struts framework as a part of whole process of software application development. After conducting an analysis of existing approaches in web application development and tools available in the market, a given proposal of meta-model for web application modeling targets to fill the gap in modeling of web applications. An existing framework was extended and enhanced to reflect new visual syntax for modeling interactions and architecture of web application.

Applied method has proven its capabilities in modeling web applications. Its usage can considerably increase the speed of development process of web application.

Table of Contents:

List of figures

List of Acronyms:

CASE
Computer Aided Software Engineering........ 15, 20, 22, 24
DB
Database........ 6, 12, 15
DDL
Data Definition Language........ 6
ER
Entity-Relation........ 6, 15
HTML
Hyper Text Markup Language........ 7, 11, 16, 28
IDE
Integrated Development Environment........ 11, 12, 18, 19, 23
JSP
Java Server Page........ 6, 10, 11, 18, 19, 37
MDA
Model Driven Architecture........ 6-9, 22-24, 44
MDR
Metadata Repository........ 24, 26, 28, 38, 44
MVC
Model-View-Controler........ 6, 34, 37
PIM
Platform Independent Model........ 8, 9, 23
PSM
Platform Specific Model........ 8, 23
QVT
Query/View/Transformation........ 42
RDBMS
Relational Database Management System........ 6
TAS
Target Application Space........ 26
UML
Unified Modeling Language........ 6-9, 13, 15, 19, 20, 22-26, 28, 32, 33, 37, 38, 43, 44
WDM
Web Dynamics Modeling........ 29, 30, 36, 38-40, 42, 43
WebML
Web Modeling Language........ 12-18, 23, 44
XMI
XML-based Metadata Interchange........ 20, 24, 26, 28, 38
XML
Extensible Markup Language........ 6, 14
XSL
Extensible Style Language........ 16

1 Description of web application modeling.

Humans tend to be lazy, with two absolutely different consequences. On the one hand, this can lead to stagnation. On the other, it can lead to looking for ways to do something with less effort but achieving the same result. This last aspect has been key in human evolution. Concerning programming we can see the same thread all over years: make something that might be reused afterwards with as small changes as possible. That took us from procedural paradigm in software development to the object-oriented paradigm. And it can be taken one step further by adopting modeling techniques. This applies to software engineering as well as to any other of the engineering sciences. So, we do build models of software that we want to implement, with the result that good models will be reuse later. For building models we use UML that provides us with set of diagrams, each used for a special purpose.

After successfully following some approach, we want to apply the same approach again in similar situations. In the case of software, this requires that our models can be translated to implementations that are correct and defect-free. Moreover, we want this pattern to be customizable for our needs in specific cases. Thus we come to the main idea of the Model Driven Architecture (MDA). MDA is about generating executable applications from models expressed in one or more modeling languages.

1.1 Goal of web application modeling

For the usual software applications there are many tools, even non-MDA tools that can generate code based on models. For example, there are tools to generate source code from classes. The source code can later be changed, adjusted for specific needs, compiled and finally deployed and executed. But in general, there is no self-standing mechanism to generate all software artifacts that we will need for web applications. We agree with the remark that UML is not for generating any files, but for representing the dependencies between and within entities used for modeling.

The kinds of web applications I will focus on are those consisting of three layers (so called 3-tier architecture): Data-, Business- and Presentation Layers.

- **Data layer**: Usually an RDBMS. It is desirable to create the DB schema in the same tool as the rest of the web application, e.g. by using either the ER notation or UML. Another issue is the transformation from the graphical representation to the DDL. The data for the application might be also contained in XML- based DB.
- **Business layer:** This layer I would like to subdivide into 2 sub-layers, with regards to purposes of web application:
  - **Logical layer:** the one that ensures proper functionality, consistency of data, inter-connectivity between objects, dataflow management. This layer is completely independent from the web front-end of the application. Here might be pure J2EE integration: EJB, JavaBeans…
  - **Web adapter layer:** the one necessary for adjusting the logical model to the specific needs of web application. That is the one that will be used as the “Model” in the MVC pattern that will be applied in the presentation layer.
- **Presentation layer:** The web-front-end of the web application. In this work, it is assumed to be implemented following the MVC pattern, using Struts. The main idea behind that is separation of concerns, but here as the logic we have the Web Adapter Layer, the workflow is regulated by the Struts framework with the help of Struts Actions, and the presentation is done via appropriate JSP pages that render the output of the “model” to be presented in the “view”.

These layers can be seen in the following chart:

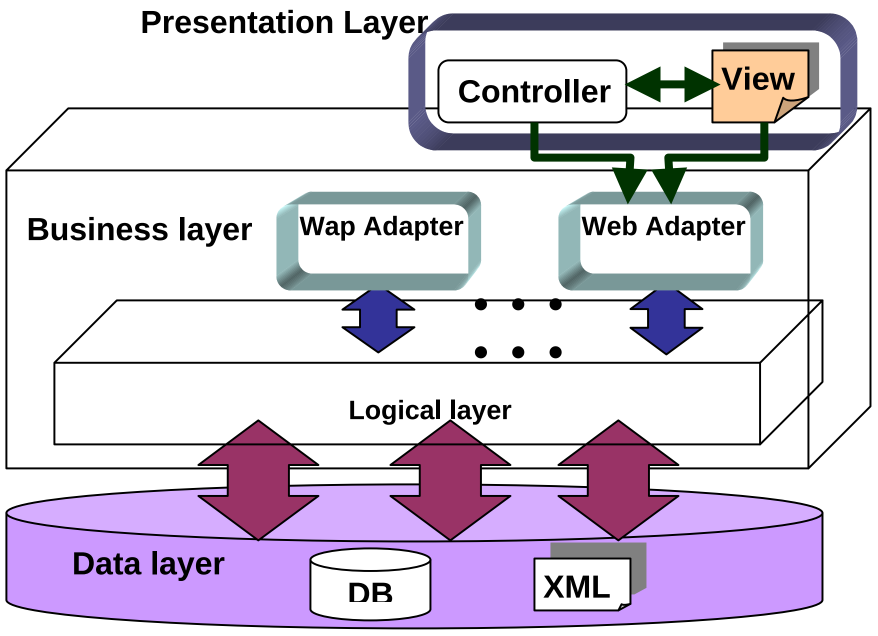


Figure 1.1 Layers in web application

Web applications consist of static HTML files, dynamically generated web pages, configuration files and so on. The UML notation clearly defines the purposes of different diagrams. But still there is no clear way to specify which diagrams, or what modeling abstractions should be used for web application modeling.

Another quite challenging issue is type of diagram to use to show the dynamics within web application. We need to represent transitions and transformations that occur to some parts of web application. Usually specifically for that purpose in UML there are Sequence, Activity or State diagrams.

This work is trying to fill that gap, and provide a possibility to model web applications and transform these models to a compilable set of files. I will use State Diagrams for showing dynamics in web application, where a "State" is a processing of some resource by a server, e.g. by a Struts action. This work uses the AndroMDA framework as a basis, adapting and extending it [AndroMDA].

1.2 MDA brief description

Model Driven Architecture is a comprehensive approach to information systems engineering that systematically addresses the complete life cycle of designing, deploying, integrating, and managing software applications using open standards. In short, MDA is a way to bring order to the increasing complex and fragmented world of enterprise computing. Introducing MDA rapidly pays for itself in reduced software development and maintenance costs, as well as improved time-to-market and quality for mission-critical applications.[FStrt]

There are quite some articles present describing benefits and pros of MDA approach in software development. Here I am providing some citation from one of them made by Middleware Company in productivity analysis of MDA based approach.

The Model-Driven Architecture is a development paradigm that aims to insulate business and application logic from technology evolution. It helps you build code quickly, in a middleware-independent, well-architected, consistent and maintainable fashion.

The crux of the MDA paradigm is a development process with the following steps:

1. Secure business requirements for an application.
2. Develop UML diagrams for the domain model, independent of any particular technology (J2EE, Microsoft .NET, CORBA, etc.). This UML model represents the core business services and components. This UML model is called a *Platform-Independent Model (PIM)* because it is completely technology-independent — this UML model would be the same regardless of whether you decided to use J2EE or .NET. You develop this UML model using the UML modeling capabilities of an MDA specific modeling tool.
3. Build UML diagrams for the application, specific to a particular technology (such as J2EE, for example). This UML model will have elements that are technology-specific, such as specific J2EE design patterns. This UML model is called the *Platform-Specific Model (PSM)*. You can build this manually, or you can generate much of it using an MDA tool and hand-tune only the pieces of it that require customization.
4. Finally, *generate* the application code using an MDA tool. That is to say, instead of writing the application by hand based on the UML model, you *generate* the majority of it from the UML diagrams. In the case of J2EE, the MDA tool would generate most of the servlets, JSPs and EJBs. You would then be left to fill in any details that could not be modeled using UML, such as business logic.[CaseSt]

1.2.1 MDA vs. traditional development?

Here I will try to compare MDA approach with traditional way of creating a software product. I will do that on the grounds of the same research conducted by Middleware. MDA lets you generate code from UML models, that is true, but it was also true pre-MDA. Rational Rose, for example, can generate Java classes from a UML model. The key advancement of MDA is that it lets you go from a platform independent, high-level design all the way to platform specific code that is fairly complete. There are several particular points to note:

- MDA starts from a higher level of abstraction than other design processes. The top-level model (PIM) is very abstract; just entities and services.
- The PSM is a complete description of the application in the form of *metadata*. At that level you can enhance the design with technology specific features (e.g. custom finders for EJB entity beans) without touching Java code.
- The code generated from the PSM is close to a complete application. Many tools generate code from some kind of model (such as *Middlegen* or *XDoclet*), but they give you pieces of an application. They are not comprehensive because they do not start from a complete model of the application.
- The algorithms that generate PSM from PIM, and code from PSM, are intended to be configurable by the architect.[CaseSt]

1.2.2 Provided benefits of MDA approach.

The Middleware Company in its report “Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach” analyzes benefits that organizations gained by applying MDA way of software development:

- **Faster development time.** By generating code rather than handwriting each file, you are saving the “busy work” required to write the same files over and over again. For example, in the J2EE world, you sometimes need to write six or more files to create just one EJB component. Most of this can be automated with a clever code generation tool.
- **Architectural advantages.** When using MDA, you model your system using UML—not just by modeling Java classes, but high-level domain entities as well. This procedure

forces you to actually *think* about the architecture and object model behind your system, rather than simply diving into coding, which many developers still do. Software engineering principles have proven that by designing your system first, you'll reduce the possibility of introducing architectural flaws into your system later on in the development life cycle.

- **Improves code consistency and maintainability.** Most organizations have problems keeping their application architectures and application code consistent in their projects. Some developers will use well-accepted design patterns, while others will not. By using an MDA tool to generate your code with a consistent algorithm, rather than writing it by hand, you give all developers the ability to use the same underlying design patterns, since the code is generated in the same way each time. This is a significant advantage from the maintenance perspective. For example, developers at organizations that subscribe to an MDA approach to development will all be able to understand each other's code more easily because they will be leveraging the same design paradigm and language.
- **Increased portability across middleware vendors.** If you need to switch between middleware platforms (for example, switching between J2EE, .NET or CORBA), the Platform-Independent UML Model (PIM) is reusable. From the PIM, one should be able to regenerate the code for the new target platform. While not all of the code can be regenerated automatically, the ability to regenerate a large proportion of one application certainly would save time over having to rewrite it all from scratch. [CaseSt]

1.3 Specification of input and output data

The overall goal of MDA is not to write whole application manually, so that we want is to create a model, then "press a button" and receive a skeleton of the web application that might be used, complete it with some domain specific business logic, and receive a full fledged web application. Later, in case we want to implement some changes, we would like not to start everything from scratch, but instead to make some changes in model, then "press a button" once again and receive a changed implementation of the web application that uses the previously generated and manually changed files. We should take care about overlapping between different versions.

2 State of the art in the web application modeling

There is no clear or standardized view now in the industry about how web application modeling should be carried out. Different vendors present their own understanding of this problem and propose solutions to solve it. Some of those approaches will be discussed in greater detail in the first half of this chapter. Afterwards I will classify them from developer perspective, and after conducting appropriate analysis of their pros and cons I will formulate the requirements for web application modeling and choose one of them as a basis for the given work.

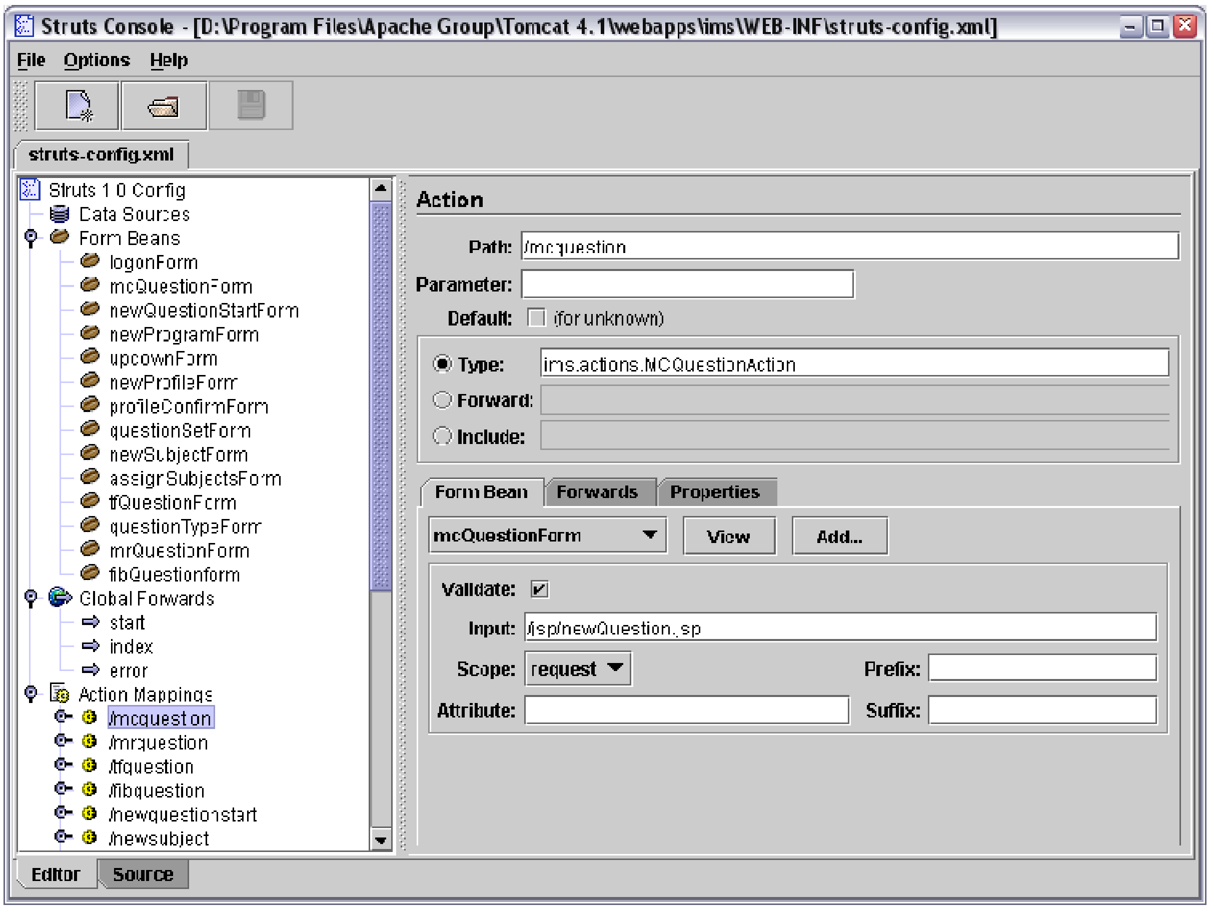
2.1 Survey of tools used for web application development

2.1.1 Struts Console

"The Struts Console is a free standalone Java Swing application for developing and managing Struts-based applications. With the Struts Console you can visually edit JSP Tag Library, Struts, Tiles and validator configuration files.

The Struts Console also plugs into multiple, popular Java IDEs for seamless management of Struts applications from one central development tool." [Concole]

As mentioned in the official information this tool focuses on manipulating the Struts configuration file. That is done via a visual representation GUI, which allows specifying such parameters from Struts framework as: Actions, Action Forms, Data Sources and Forwards.



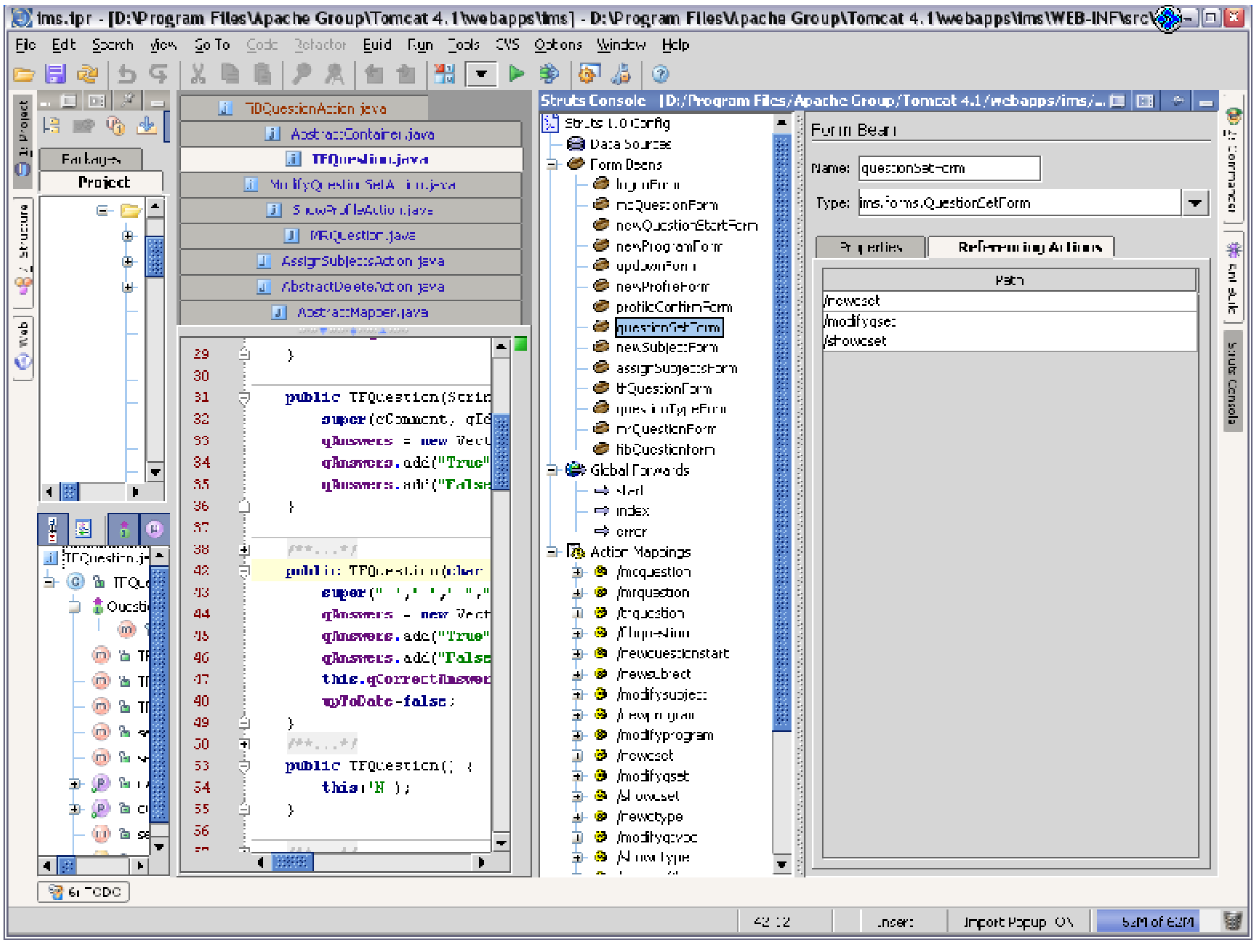
Path: /mcquestion
Parameter:
Default: ☐ (for unknown)
[✓ Type:] ims.actions.MCQuestionAction
Forward:
Include:

mcQuestionForm
Validate: ✓
Input: /jsp/newQuestion.jsp
Scope: request
Prefix:
Attribute:
Suffix:

Figure 2.1 Struts Console Application

The interface is intuitive, but implies from user good understanding of the Struts framework itself. You have to manually provide all the data that is related to the Struts specification, such as for instance full names of the classes corresponding to StrutsActions or StrutsForms. The good thing about that is the possibility to validate the file you already have for compliance.

Another great issue about the Struts Console is the possibility to use it as a plugin for the Java IDE in use. Currently supported IDEs are: Borland JBuilder (version 4+), Eclipse (version 1+), Intellij IDEA (version 3.0 +, build 668), IBM WebSphere Appl. Dev. (version 4.0.3+), NetBeans (version 3.2+), Oracle JDeveloper (version 9i+), Sun One Studio (Forte) (version 3.0+). Using this tool as a plugin enables you to concentrate on application development and not on the xml syntax bug fixing.



```
    }

    public TFQuestion(Strin
        super(cComment, qId
        qAnswers = new Vect
        qAnswers.add("True"
        qAnswers.add("False
    }

    /**...*/
    public TFQuestion(char
        super(" ", " ", " ","
        qAnswers = new Vect
        qAnswers.add("True"
        qAnswers.add("False
        this.qCorrectAnswer
        upToDate=false;
    }
    /**...*/
    public TFQuestion() {
        this('N');
    }
```

Name: questionSetForm

Type: ims.forms.QuestionSetForm

Path
/newqset
/modifyqset
/showqset

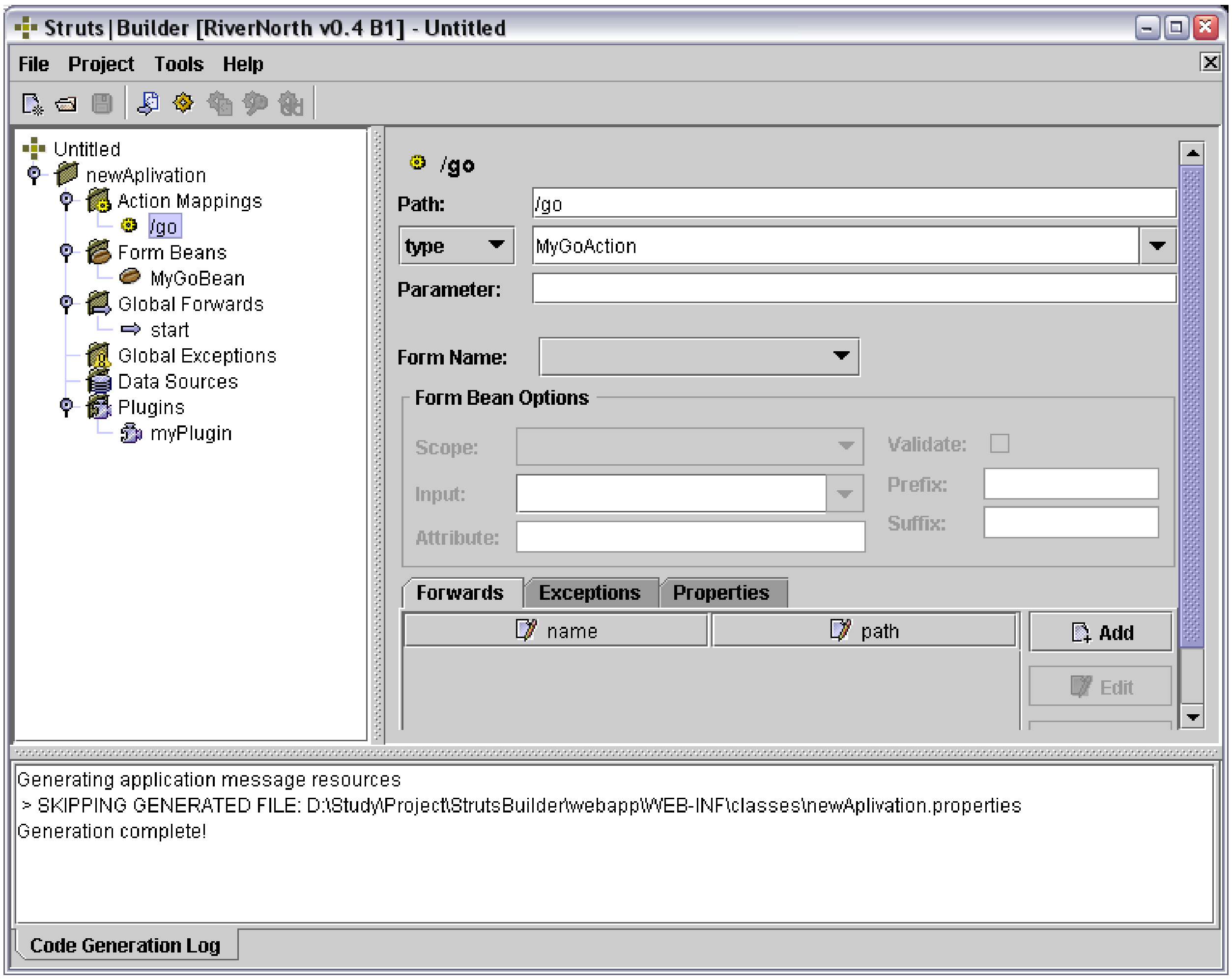
Figure 2.2 Struts Console as a plugin to IDEA

Hence, we can position the Struts Console tool for modeling on the Presentation Layer of our model of web application, with strong focus on Struts framework and experienced developer.

2.1.2 Struts Builder

“A Java Swing-based development environment to assist in the rapid creation of Struts-based (currently Struts 1.1b) web applications. This standalone application allows users to create from scratch and/or import existing Struts 1.x struts-config.xml files,”[StBlder]

This open-source project allows modification of Struts configuration file via form-based interface. Stated import feature failed on importing existing Struts project that might be caused by only Beta-version availability of the tool. Given tool can generate the shell code including JSP/HTML pages, web.xml, struts-config.xml and java source files for Actions and ActionForms classes required for an implementation. But on the other hand this is only useful in case of from scratch development of the project, with assumption that further structure will not be changed. Quite obvious that is not true for most of the applications.



Path: /go
type: MyGoAction
Parameter:
Form Name:
Scope:
Validate:
Input:
Prefix:
Attribute:
Suffix:

```
Generating application message resources
> SKIPPING GENERATED FILE: D:\Study\Project\StrutsBuilder\webapp\WEB-INF\classes\newAplivation.properties
Generation complete!
```

Figure 2.3 Struts Builder

As authors mentioned the future development of this project will be concentrated on integrating this tool to IDE available on the market. So currently we can specify this tool as the one useful for basic development of blueprint of future project that requires quite good knowledge from developer of Struts itself, and the application domain to provide the structure that will not be changed in future. In current state this tool is not applicable in the production development of the web application

2.1.3 WebRatio

A great and, at the same time, restrictive tool. This tool is the only one that is focused on the whole process of web application development. You are allowed to create an application from scratch to the real running product. It takes care on all layers of abstraction from data-layer allowing to specify the Data Base schema, and create a DB as well as fill it with a test data for future testing of application, to the presentation layer, where they even subdivide this layer into 2: the SiteView Layer and PagePresentation Layer. One positive thing about Siteviews in particularly and WebRatio tool in general is that it is based on the abstract notation – WebML (Web Modeling Language). That specifies aspects specific to the web application development.[WebML][WebRatio]

The project is represented via several views on the future web application, here I will list them, giving a short description and afterwards I will describe each of them more precisely highlighting pros and cons, so:

1. ”Editing view” – Here we have an overall structure of the web application represented via
   - Data Structure – Specifying the structure of the information objects used by the application.
   - Site Composition – Site views that represent views for different users. Specifying a site view for letting users access the information managed by the application.
2. Mapping View – Creation of a data source, connecting it to the project and filling with test data for simulating the content
3. Presentation View – Specification of the site presentation via page design and generating the application code.

Test-project observation:

By providing these three views on the process of web application development vendors trying to focus on site logic as it is. That leads them to reducing the role of the business logic of the application as it was described in the model. While concentrating on presentational workflow they lose the point in the data management (retrieving, synchronization) as well as possible optimization on different levels of abstraction. All data transformation and manipulation and management till the point of presentational workflow are restricted to what is available in the tool; that on my opinion is a disadvantage. Within the process of site workflow management used WebML notation could not be overestimated, it provides a lot useful abstractions that can be used for dataflow management, such as Data/Multi Data Unit, Scroller Unit, Entry Unit, Create/Delete/Modify Unit, Login/Logout Unit several Link Units and so on. Given abstractions might be well applied for separation of the presentation to several areas, pages and transitions between them with help of links.[WebML]

Another issue about WebRatio tools is non-intuitive and inconvenient graphical interface. The one using it either should already has done several projects with this tool, or is forced to consult the manual throughout the whole process. Set of immediate accessible actions is really limited, to change some parameter in one of the diagrams one supposed to make set of actions. For instance: while creating anything let’s say entity in Data Structure view you obviously want to change the name of this entity from “Entitity4” to something more meaningful. For entity it is still quite ok, because your focus is set to the name field in the property panel, which is strictly parked to the left lower corner of the screen. If you then want to add some attributes to this entity you have to move focus either to the diagram screen or hierarchy tree, where you can access this possibility via context menu, and then once again move back to property panel, to change the name, but for an attribute you definitely want to specify something more then just name, let’s say type, but to do that you have manually with help of the mouse set focus to the field type and then edit it. Just imagine the number of mouse movements just for setting focus if you want to set several parameters related to the attribute of property such as constraint, type, value.

So let’s start walking through the WebRatio web application development process in details:

1. Editing view

Here we can actually create the model of our application in two different perspectives. Data Structure and Site Composition. This model will become a basis for future web application and for the transformations that we have to make with the model. Data Structure represents the Structure model concept from WebML that is actually a combination of UML Class Diagrams and Entity-Relation modeling concept. The fundamental elements of structure models are *entities*, defined as containers of data elements, and *relationships*, defined as semantic connections between entities. Entities have named properties, called *attributes*, with an associated type. Entities can be organized in *generalization hierarchies* and relationships can be restricted by means of *cardinality constraints*. Instances of entities are considered individually addressable by means of a unique identifier (OID). This OID is the primary key of the entity it cannot be removed or manipulated by the application directly. For the XML data store OID represents the XML ID attributes.

Site composition in the editing view allows to create several site views on the given web application specifying what information is relevant for each view. You can provide here for instance one view for the administrator of the site, that will be totally different from the one for the regularly user. Each of this *site views* represents the combination of composition and navigation model of WebML. Where on composition model you specify what is represented where in the web site, whilst on a navigation model you specify possible transitions from one page to another. More precisely composition modeling specifies which pages compose the hypertext, and which *content units* make up a page. The pages of the web site are the containers of information delivered to the user. Units are atomic content elements used to publish the information described in the structure model. Seven types of units are predefined in WebML to compose pages: *data*, *multi-data*, *index* (and its variants *multichoice* and *hierarchical*), *entry*, and *scroller*.

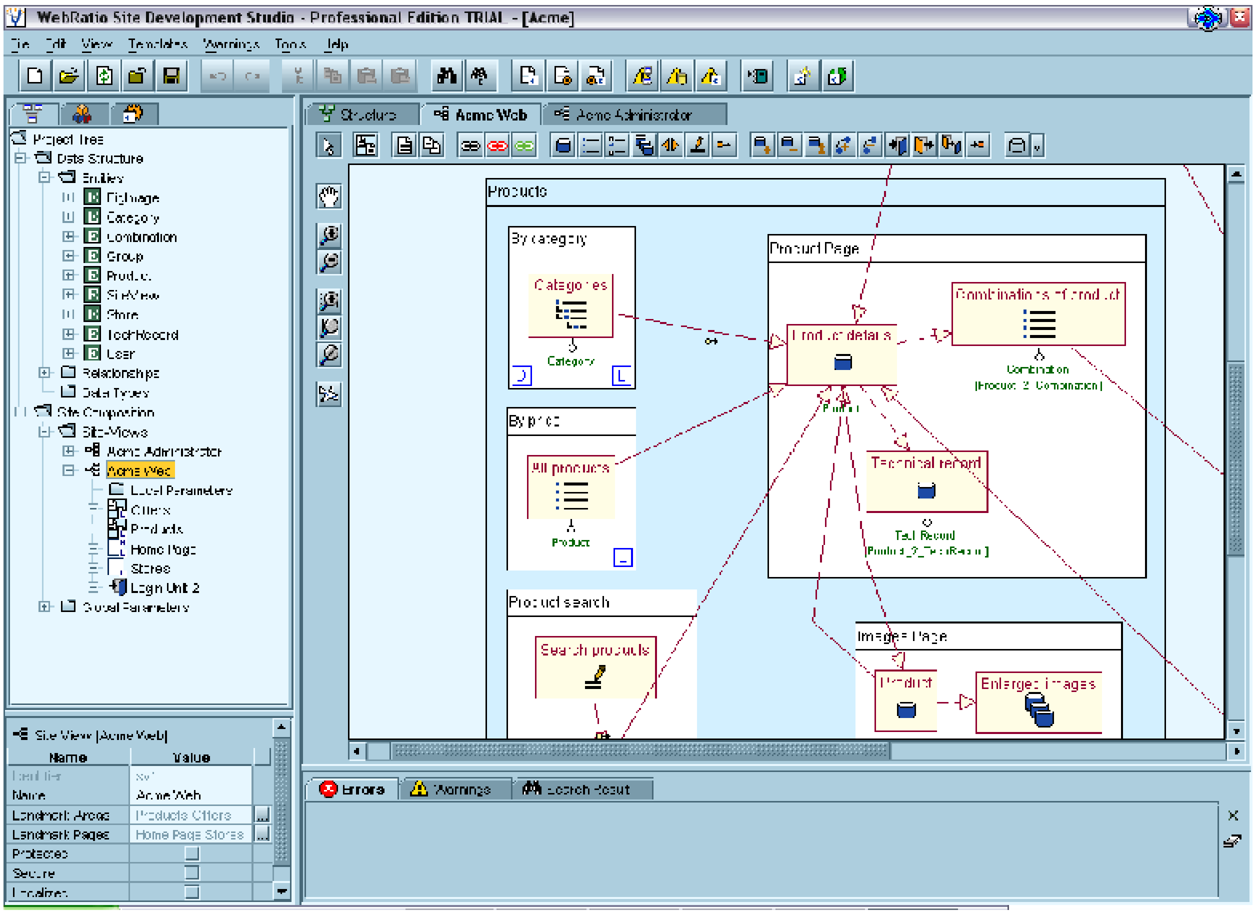


Figure 2.4 WebRatio. Site Composition view

The transitions between pages are provided with the help of links. Links might be defined between units inside a single page, between units placed in different pages, and between pages. A link may serve the following purposes:

1. Moving the focus of presentation from one page to another one.
2. Passing information from a unit to another unit.
3. Producing some side effect (e.g., the execution of an update operation)[WebML].

The information carried along a link is called *navigation context*, or simply *context*. Links that carry context information are called *contextual links*, whereas links that have no associated context information are called *non-contextual links*.

We can relate site views to the basic UML notation where site composition is Use Case diagram, “Site Views” – set of Use Cases for different “Users” – Actors. Site views are trying to represent the dynamics of the web application, where all the dynamics of presentation is based on pages, page-blocks and transitions between (links). I think this approach should be extended, by providing “more dynamic views” on model. Because now site views are more like implementation class diagram in UML stating the dependencies between objects not providing the real transition overview. So the good idea would be to take the best that WebML provides for navigation model and enhance that with some diagrams of UML that are used for modeling interactions and dynamic behavior. I am going to expand this idea further.

2. Mapping view

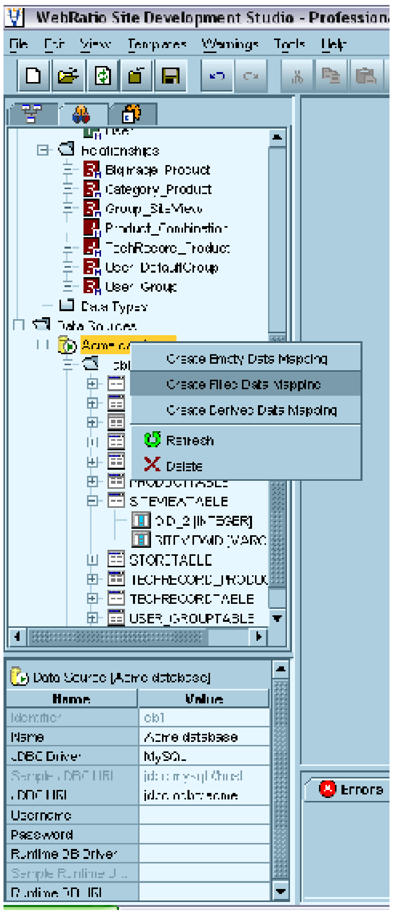
At this project view the user is supposed to create a mapping from ER diagram of data structure view to the real underlying DB. The expected behavior of a user will be to specify Data source, to which the existing model has to be transformed. User has to pick up one of the 5 possible choices which DB he wants to use, give username, password. After that the one thing left is press the button and you will have newly created DB filled with sample data.

This scheme works quite well whilst you don’t have any knowledge about DB architecture, you do not want to get in touch with the process of transformation, or at least choosing the database that is not supported by the tool. If you are the one that want, at least, to set some information you will face some troubles:

1. You cannot specify a new driver for the database that is not included in the tool. So you are restricted to 4 provided DB drivers and one JDBC-ODBC bridge connector.
2. It is not possible to use your own driver even for databases that are provided, so that means that if there will be a new driver for MySQL database you will not be able to use it, because the tools is restricted to “some” concrete MySQL driver, and you even cannot obtain the information easily what driver it actually uses.
3. You are not allowed to specify to what database type specific model type is mapped, neither to see to which they have been mapped.
4. You cannot access the DB schema for created database, as well as the executed SQL statements, that might be very useful in case of analysis and understanding of the architecture of created application.

I can summarize this part as being really very weak concerning the help to the Developer, and not to the person that creates his first “Hello, world” program. Even though WebRatio is not supposed to be the DB CASE tool it still lacks some basic things that were described before.

Another thing that cannot be considered as the good part is the hiding of information about how data mapping is programmatically implemented. As we know, several patterns exist for mapping business layer with the persistent layer, and to have a look at that can help discovering of bottlenecks of the application. I am positively concerned that the approach to connect business and persistence layer should be taken with regards of the application architecture that we are designing.



Data Source [Acme database]

Name	Value
Identifier	db1
Name	Acme database
JDBC Driver	MySQL
Sample JDBC URL	jdbc:mysql://host/
JDBC URL	jdbc:odbc:acme
Username	
Password	
Runtime DB Driver	
Sample Runtime J...	
Runtime DB URL	

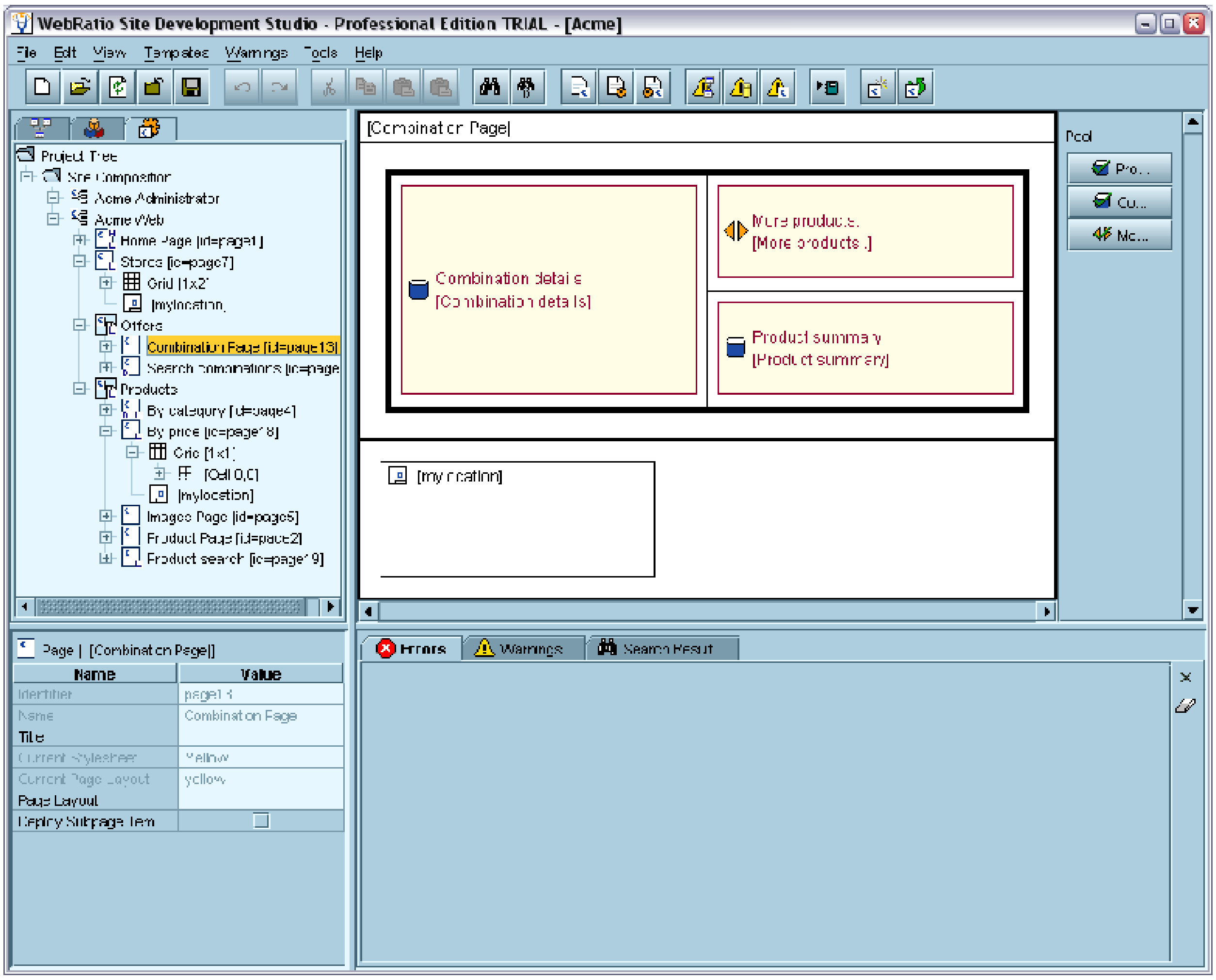
Figure 2.5 WebRatio. Data Mapping view.

3. Presentation view

The generation of the templates constituting the site is the last step to obtain a running application. You can select a style sheet that for the project itself and this value will be default for all paged in the project. Also you can specify style sheet directly for that page. In this view user is entitled to create layout of his pages. At first you have a blank page, for each page or page block that should be represented in the project, for each page you have a set of item units associated with this page on editing stage. You can allocate them on the page on the basis on table layout that is provided here.

The layout management and template management are based on XSL transformation, which map WebML pages into mark-up code.[WebRatio] For direct manipulation with the styles that can be assigned for the item they use Easy-Styler. This component allows you to view existing stylesheets, templates as well as provide new ones. I have to admit that even though this is stated to be true, it is not clear how to do this. The user interface is still does not provide any hint about the correctness or right sequence of actions to achieve desired result.

For each stylesheet and layout combination they assign the view to each item to be displayed in the page. That is quite good, but still you are suggested to use external HTML editor for creating your templates, but while doing this, you have to leave the EasyStyle tags formatting untouched. That supposed to provide compatibility with the style generation framework used within WebRatio.



Page | [Combination Page]

Name	Value
Identifier	page13
Name	Combination Page
Title	
Current Stylesheet	Yellow
Current Page Layout	yellow
Page Layout	
Deploy Subpage Item	☐

Figure 2.6 WebRatio. Presentation View

Application execution.

Then after you passed through all that steps for generation of pages you really have to press one button, and all pages and files that is necessary for this web application will be generated in the provided directory. The most stunning part is that if you have done everything correctly in previous step it will run. If not, then you will receive some help about what elements should be changed to omit existing warning or errors.

If you are quite curious to have a look on what actually was generated, you will find

1. A bunch of jsp pages with corresponding xml pages, where as I understand jsp specifies a layout, and xml specifies the data content that should be put to that layout.
2. Set of resource folders and files, providing css files, pictures and other look and feel staff.
3. Within WEB-INF folder you will find struts configuration file and struts tile files [PrgStr] [StrInAct], as well as WebML tile file.
4. Folder “desc” that contains some information about navigation between pages, data units and even more, it contains information what database statements must be executed
5. You will NOT find any java classes generated by the framework except ones for struts forms. This was quite surprising for me, because I was really wondering how data manipulation is done. Obtaining this information is quite a problem. I suppose that all information management is done within the framework taking as an input the content of the desc folder (see previous list item). But again that is a pure assumption.

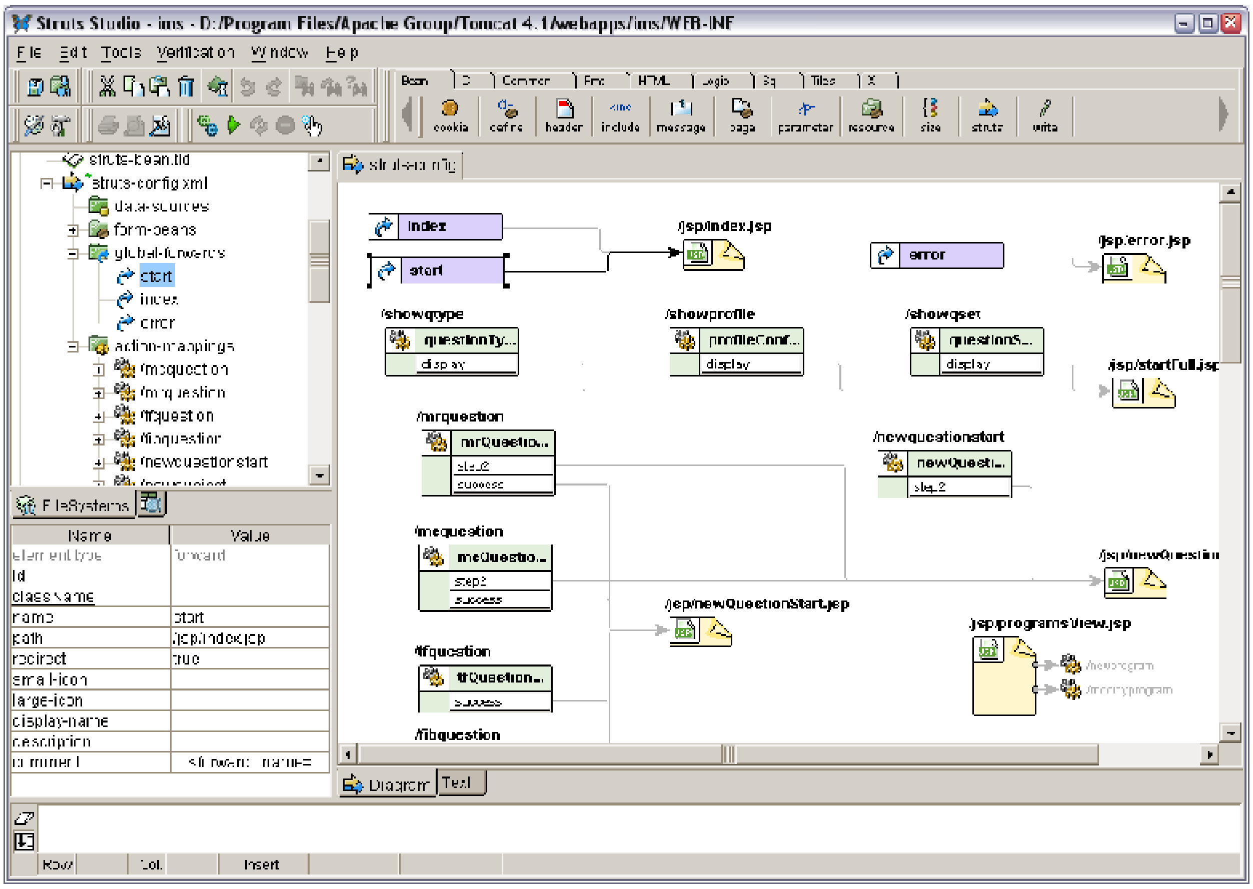
Summary: The only tool for the moment that covers all process of web application development from first stage till last. It is based on the underlying methodology – WebML. The general architecture of web application is hidden from the developer. You can understand that you are using Struts if you know that “do” extension is associated with Struts actions in general. It is not clear how to extent this framework even though it stated that it is possible.

2.1.4 Exadel Struts Studio 4.7

“Struts Studio is an integrated Struts development tool that covers the whole gamut of the Struts application development cycle from creating the application to running it.”[StStudio]

It is really easy to use from creating application from scratch or integrating the existed project into Struts Studio IDE. Design and representation are based on rendering of the struts-config.xml file. In case of creating web application from the beginning with the Struts Studio you design the struts-config.xml file, generate stubs for java classes, code JSP pages and java classes, compile the code, and run the application all without ever leaving Struts Studio. We can position this tool to the presentation layer of our hierarchy, highlighting its deep orientation on the professional developer. A software architect can have such perspectives on the web application

1. **Presentation perspective**. – Set of jsp files whose source code can be viewed by the developer. This view provides wide possibility of using tiles. User can manipulate with already plugged tiles, as well as create new ones. The set of tiles that are already provided is very impressive, and can be used for almost any situation that might occur.
2. **Java Source view**. – This IDE encapsulates quite good Java editor, with such possibilities as “code insight” It is not very impressive, but as this is not the main focus of the application, so we should not expect some outstanding code writing support.



Name	Value
element type	forward
id	
className	
name	start
path	/jsp/index.jsp
redirect	true
small-icon	
large-icon	
display-name	
description	
comment	<forward name=

Figure 2.7 Struts Studio. Dynamic view

3. **Dynamic view**. – That is actually the core feature of the whole application. Dynamic view allows user to create transitions between content related entities. It operates in Struts framework terms, providing actions, forwards, redirects, simple JSP pages, form beans, data sources, global forwards, property handling and all other stuff. This is implemented in a very good, user-friendly manner. For creating a transition or forward you need really closed to nothing number of actions to produce.

   Dynamic view is based on the analysis of struts-config.xml file itself and the syntax analysis of the source pages that you insert into the model. That allows auto creation of the links (transitions) between dependent issues. So if you add a page that has a link to another page or action then this link is automatically shown in the diagram. Jsp-page representation gives you the possibility to specify two ways of representing the link between dependent pages. The first one is the usual – direct connect, and the second one – is the representation as a shortcut, where the dependent page is being displayed next to given page with a short label.

   Dynamic view if being created from scratch allows you to generate stubs for java classes for struts forms, struts actions. On the contrary if being used on the late stages of the project development it helps a lot while analyzing the logic of the application, transitions and dependencies between pages.

4. **Deployment.** Struts Studio has an integrated ANT support. That means that you can specify the ant script file that you want to operate with, and then call ant task from within the IDE itself. The real value of this could not be overestimated. Especially if you use this tool not from the first step of your web application development, then you might have already had a build file for your application that does all the compiling, deployment, and configuration management for you. In this case you have to just attach this file to the Struts Studio project, and continue further rapid development. Even if you are starting you project right from the start in the Struts Studio environment implementing build tasks for ant would be a good idea, for not being dependent on some specific environment variables or other parameters that are in use in IDE.

5. **Execution.** Struts Studio has a lightweight Tomcat web server integrated in it. So you can run your application right from the IDE, just by pressing a button. On that all necessary files will be compiled and updated and you will be able to start your application in usual browser.

We have to emphasize the outstanding customization possibilities of the Struts Studio. For the jsp page itself you can specify any parameter that it might need, as well as all parameters for the whole application, and that is all done in a clear and distinctive way. The only lack of this tool is lacking of the preview tab of the jsp-page, and the possibility of visual rendering of that page. I hope that it will be possible in next releases.

Summarizing all stated above about Struts Studio I could say that this is the best tool for managing presentation layer of the web application and its collaboration with business layer. Struts Studio is possible to use either for newly created projects, or for projects that undergo some changes in design and implementation.

2.1.5 ArcStyler 4

Brief overview. Very generic tool focuses on overall process of application development. There is no special focus on web application development, but it is possible. The main idea behind ArcStyler is transformation of UML model to specific implementation based on the Model Driven Architecture approach. On the basis of the model you are allowed to produce different kinds of implementation depending on the set of *cartridges* that you use. All transformation of the model to specific code is actually hidden within that cartridge. The implementation of the cartridge is unknown, because it's the intellectual property of IO-Software. The output of this transformation is:

- Set of Java source files. In case as a cartridge you have chosen the one for EJB transformation, then you will get all files-stubs implementing home and remote interfaces
- Default test client for testing your applications
- Deployment descriptors for this EJB to be installed into application server
- JBuilder project file that emphasizes the strong trend to cooperation with Borland JBuilder
- ANT build support files. These ant files may be executed as from ArcStyler itself or from the shell directly. From within ArcStyler you have quite a good visual overview of possible ANT task to run, where you can chose appropriate, establish a sequence and execute them.

As a general resource requirements of this tool are beyond any expectations. The minimum requirement is 256 M of RAM. But from my practical experience I can say that that's quite not true. Cause having just exactly 256M for me it was very irritating to wait each time for response for some seconds, and sometimes for some tens of seconds, so more precise requirements concerning RAM should be considered those mentioned as preferable in the tool description, namely 1 Gb of RAM.

2.1.6 AndroMDA + Poseidon for UML

AndroMDA is an open source code generation framework that follows the model driven architecture paradigm. It takes a UML model from a CASE-tool and generates classes and deployable components (J2EE or other), specific for your application architecture.[AndroMDA]

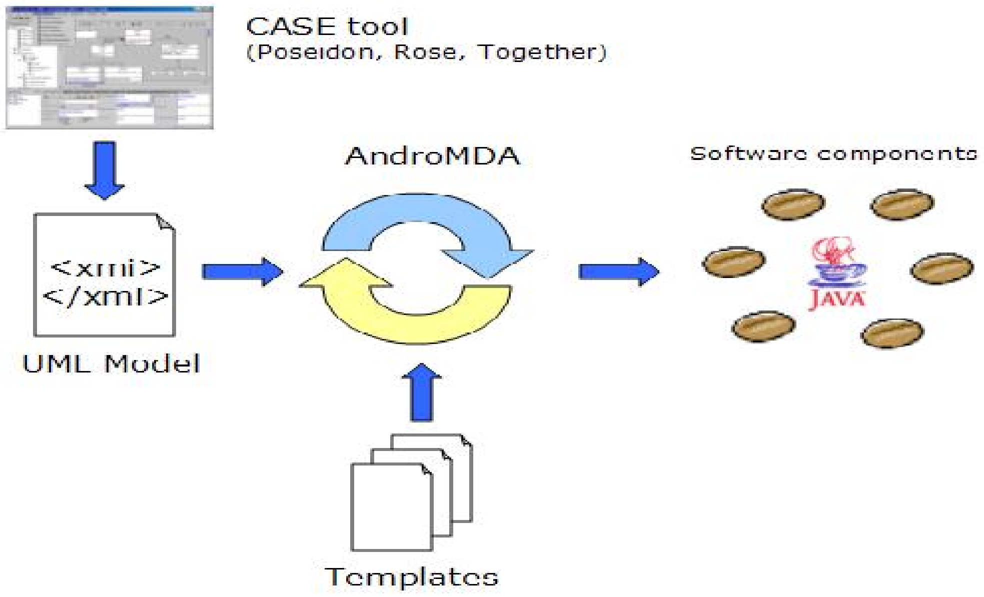


Figure 2.8 AndroMDA framework.

AndroMDA is a code generator framework that takes UML model from a CASE-tool in XMI format and generates custom components. It comes with a set of sample templates that generate classes attributed with XDoclet tags. One build step later, the XDoclet tool generates full-blown components that can readily be deployed in the JBoss application server (and the other servers that XDoclet can feed).

XDoclet is an extended Javadoc Doclet engine. It is a generic Java tool that lets you create custom Javadoc @tags and based on those @tags generate source code or other files (such as xml-ish deployment descriptors) using a template engine it provides. XDoclet supports a set of common standard tasks such as web.xml or ejb-jar.xml generation; users and contributors can create other templates and @tags and add support for other technologies, too.[XDoclet]

Appropriate modules implement specific code generation. These modules, called cartridges, are plugged into core framework and produce source code on the basis of template evaluation. There are several cartridges available for specific needs. Among them are

1 andromda-java – generates general Java source code.

2 andromda-ejb – generates Enterprise JavaBeans.

3 andromda-hibernate – generates persistent classes for the Hibernate O/R mapper.

4 andromda-struts – generates web pages, forms and action classes for use with Jakarta Struts.

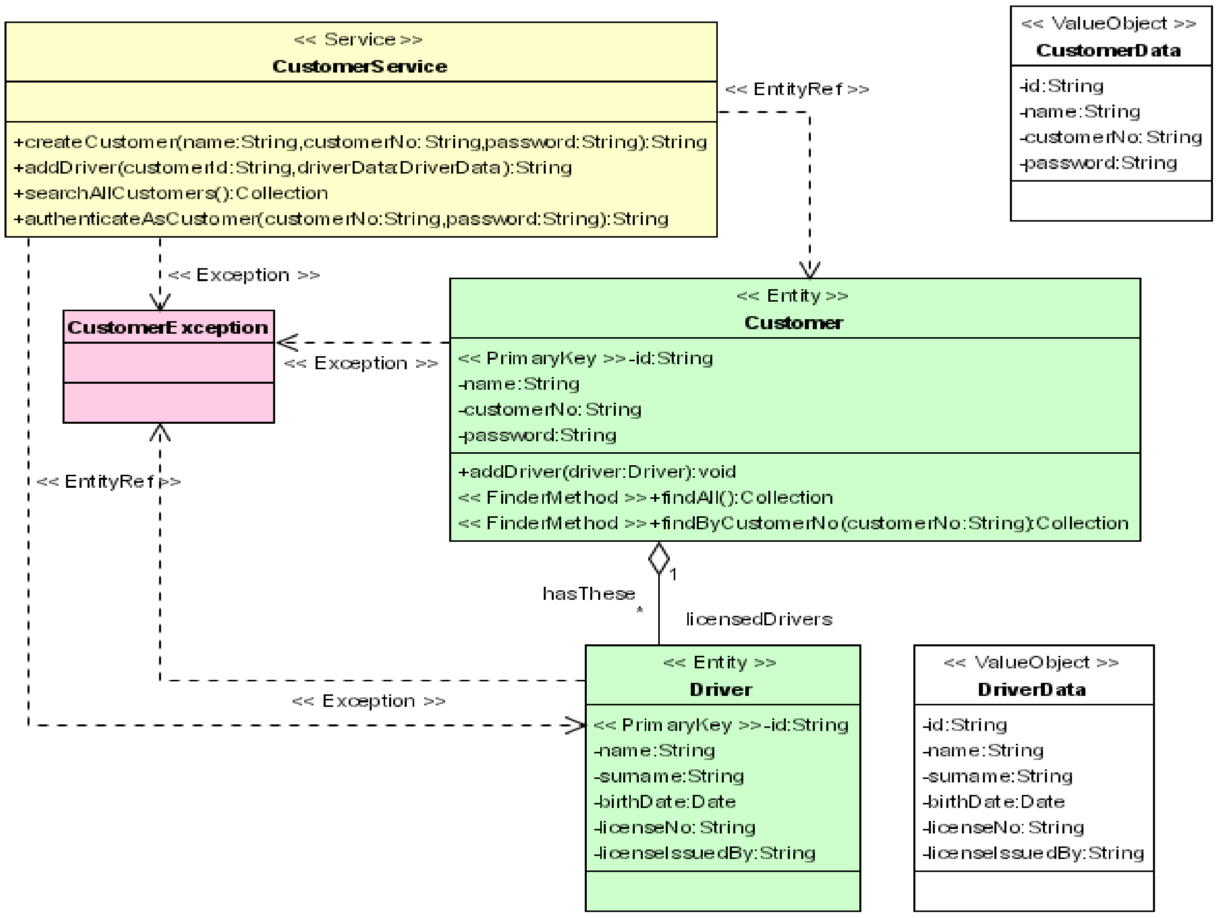


Figure 2.9 AndroMDA sample model.

The current support for Struts includes basic generation of Struts-Form classes, and generation of the stubs for Struts-Action classes. Moreover framework provides generation of struts-config.xml. Generation of struts configuration file is based on the specific diagrams in the model of the application. Currently they use class diagrams and dependency links to provide information about transitions. That seems not very straightforward.

The main idea behind is that user needs to draw a model in the CASE tool like Poseidon, afterwards AndroMDA generates a set of classes, due to the rules specified in plugged cartridges, then you can manually change implementation of those classes where you want to provide your specific business logic, and afterwards all this combination of handwritten and generated classes can be build in distribution, deployed to application server and run with web graphical interface.

Poseidon for UML is a full-fledged UML CASE-Tool. It evolved from the open-source project ArgoUML and has turned it into a world class modeling tool. Today, it has the fastest growing user community and is famous for its superior usability. Its superior usability makes it the easiest tool to learn and work with. With UML standards compliancy, it facilitates interoperability with other sets of tools while maintaining investment security. [gw.com]

Poseidon support quite customizable plug-in mechanism that allows extending its possibilities with new features. The plug-in mechanism and integration interface allows plug-in selection and development, as well as providing a high degree of modularity.

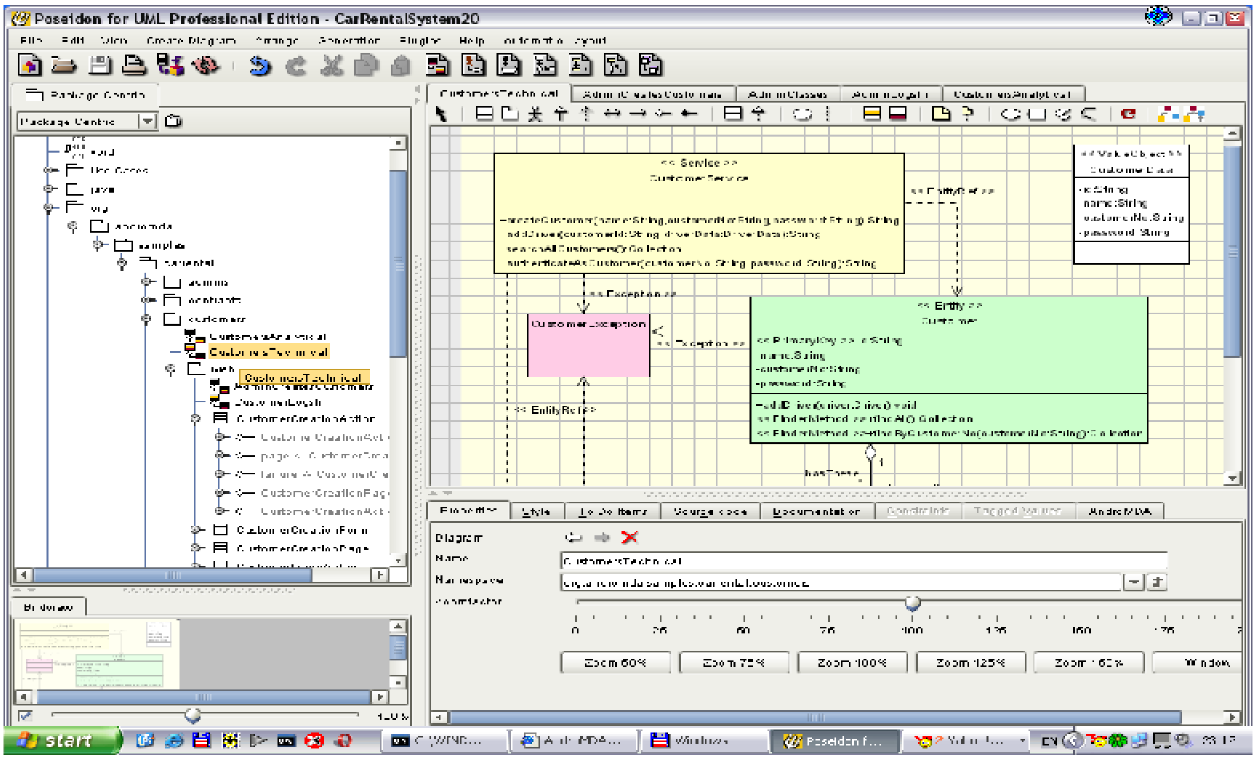


Figure 2.10 Poseidon for UML Car-Rental-System, AndroMDA example

As one of the possible plug-ins it provides a plug-in to the AndroMDA. This is one of the possible good approaches in development of applications in general and web applications in particularly.

Let’s have a look to the process of web application development with help of Poseidon and AndroMDA from the point of having the intention to build something till the last step of deployment to the application server.

This process is equal to the process of general software design, but it has one major difference. All classes that are present in the model should be created regarding the MDA perspective. That means that classes, attributes must have a special stereotype, associated with parameter of such type in the MDA specification. Moreover, some of them should also be designed with appropriate tagged values to foresee future possible AndroMDA transformation.

In the given example we have a stereotypes for whole EJB components development, stereotypes for enabling Struts framework functionality. These examples are present in the Figure 2.10 They design and represent the dynamics of the web application with help of class diagram. The dependencies represent the transitions between pages, actions in the sense of Struts framework.

After we have created our model with regards to the MDA specification and to the restrictions that AndroMDA oblige we can generate source code for the web application. It is possible to generate all ant tasks from within Poseidon IDE. Produced code is very good structured and easy to read and improve.

As a short summary about cooperation of AndroMDA and Poseidon I can say this combination allows you to create web application from scratch, maintain it afterwards, make some changes in the architecture, and update the source.

2.2 Different approaches to web application modeling

All existing approaches for web application modeling can be classified by the target architectural layer. As it was shown in the beginning of this paper I consider Data, Business and Presentational layer as in web application. Data layer is definitely out of scope of concerning interactions modeling that is true for all approaches. Hence I will describe them from the point of Presentation or Business layer focus in web application modeling.

- Presentation layer focused modeling tools.

  Here a distinctive player is "Exadel Struts Studio" they use somehow "from source" approach. They rely on evaluating configuration file for Struts based application as a basis, and on that they create visual presentation of dependencies present in the struts-config.xml, displaying all possible transition specified there. To the presentation user allowed adding additional links that symbolize dependencies between presentational pages or Struts actions. They provide very good support in customizing presentational preferences of a web application. The detailed description is present in previous part of this paper.

- Business layer focused modeling tools plus additional support for presentation layer

  – WebML based – "WebRatio"

  Approach based on another language for modeling applications. This "Web Modeling Language" (WebML) was created specifically for web application modeling by Italian company WebRatio. WebML specifies abstractions for web application development. In the tool they focus on the whole process of web application development. One of the major disadvantages of this approach is its strict borders and ignoring of standardized conventions for modeling, namely UML. Detailed description for this tool is also given in the first part of this paper.

  – Pure UML based – MDA tools

  To this group I can ascribe all tools that advocate MDA driven style of development. They provide some sort of framework for applying rapid transformations of the PIM UML model to PSM UML model, and one-step later transformation of the PSM model to code. All that is done with the idea of cyclic application development in mind. So, they assure resolution of conflict due to overlapping in those transformations in further stages of application development. Each of vendors presented here provides own view of modeling with regards to web application and web front-end of enterprise application.

2.3 MDA tools competitors

Currently on the MDA market of tool providers there is no standardized way of applying model transformations, which is the core thing in MDA paradigm, so each vendor finds its own way of doing that. But still every vendor is trying to force his approach in doing that. All share the same concept of transformation model to code based on some assumption about the model. Among them are

1. “ArcStyler” — created by IO-Software, closed architecture, transformations applied on the basis of set of available cartridges. Very customizable, there are really a lot of target platforms to which it can be deployed to, different application servers and so on. More detailed description can be found in the first part of the paper.
2. “OptimalJ” — created by Compuware Corporation, closed architecture, transformations applied on the basis of patterns found in a model. OptimalJ’s pattern editing functionality enables software architects and senior designers to build and maintain their own patterns; business-focused developers then apply the pattern solution to their specific project and the time it takes to develop an application is significantly reduced, compared to creating one from scratch.[OptimalJ]
3. “AndroMDA” is an open source code generation framework that follows the model driven architecture paradigm. It takes a UML model from a CASE-tool and generates classes and deployable components (J2EE or other), specific for your application architecture

As a basis for this work I have taken AndroMDA framework owing to its open source and possibility to provide additional functionality.

2.4 Architectural and modeling aspects of AndroMDA framework

As it was mentioned before AndroMDA is a code generator framework that can generate custom components from UML model taken from a CASE-tool. It allows creation of several types of software artifacts, among them is a set of files generated for web application.

This approach uses class diagrams for representing dynamic behavior of web application. I would say that here we can observe mix of concerns, in class diagrams here we face representation of dependencies between classes, some of which are used for representing control flow from one class to another. By that it is sometimes difficult to distinguish whether presentation and where logic is modeled.

Current release of AndroMDA is going to undertake considerable changes in next, third release. The whole architecture is going to change implementing new concept of meta-model decorators, while current is based on script helpers and dynamic proxy patterns that will be described further. Research and work done in this project were based on stable AndroMDA 2 release. So the proposal and ideas discussed here should be considered as improvements and additions for implementation of dynamic proxy architecture of AndroMDA.

AndroMDA framework facilities might be interconnected with “Poseidon for UML” modeling tool. By this cooperation you can run all AndroMDA transformations from within Poseidon that reduces amount of time necessary from building internal model of AndroMDA, cause the UML model is read not from XMI file, but from Meta Data Repository (MDR) repository directly[MDRWP][MDRHome]

2.5 Problem statement

Representation of dynamic behavior in web application requires specific kinds of diagrams. I point out that point of interest here is representation of dynamics with regards to the Struts framework, Java Server Pages, and web application in general. I will not focus on User Interface modeling and interaction of a user with the system in a sense of supplying data to the system. I will primarily concentrate on interactions within different parts of the system, forced by the user that means that I will describe transitions performed in usual dynamic behavior.

I am interested in specifying the way to display possible transitions from pages to Struts actions and the other way around, transitions from within the pages, and cooperation of dynamically generated pages with static pages.

For representing dynamic behavior of an application UML provide three types of diagrams: sequence, activity and state diagrams. The two that can be applied in situation of web application modeling was: Activity and State Diagrams.

We have to take consider such points for representing dynamic behavior:

1. There should be a possibility to apply Struts framework concepts. That is we have to model struts action behavior, taking care about possible transition of control from Struts action.
2. There should be a possibility to model dependencies between different jsp pages, as well as dependencies within one jsp page.
3. The diagrams have to provide a clear visual way of observing all dependencies and transitions between entities present in web application.
4. We have to ensure that concerns are not mixed in one diagrams, if a diagram is used for logic flow then there should not be object constraints and interdependencies.
5. We have to ensure separation of concerns between business, data and presentation layer modeling.

By that I will try to introduce new visual notation for modeling dynamic behavior of web application as well as providing possibilities for modeling application’s logic.

3 AndroMDA based Web Dynamics Modeling.

The purpose of this work is to extend existing AndroMDA framework so that dynamics of web application can be described and represented with Statechart Diagrams. By that, I will ensure that different concerns are handled individually: Class Diagrams will describe dependencies between classes involved in model, whilst Statechart Diagrams will describe interaction with a user, which activities should web application perform to satisfy user needs.

In this part I am going to describe the whole procedure of AndroMDA transformation process more precisely, pointing out the places where changes will be introduced. I am going to plug some necessary functionality to AndroMDA framework so that it would be possible to subdivide representation of interactions and class dependencies.

The overall process of AndroMDA transformations looks like this:

1. AndroMDA reads the model given either in XMI file in general or directly from MDR repository in case of Poseidon’s plugin.
2. This UML model is transferred into internal model of AndroMDA. How is this done? We search the given UML model for elements with specific stereotypes, which are specified in the AndroMDA, relate those elements in the AndroMDA technological space based on stereotypes, and relationship in the UmlModel.
3. AndroMDA provides the model that it builds from the source to the transformation engine that performs all transformations. The transformations are separated for different domains (target application spaces (TAS)) that means that transformation for EJB is different from transformation for Java, or Struts, These transformations for specific TAS are made with help of cartridges

   Each cartridge has to provide a configuration file that will in specific format describe how to use this cartridge in AndroMDA framework. Basically this file provides information about stereotypes and templates that should be associated with given stereotype. Afterwards while processing a model if a model element with given stereotype is found this model element is processed with help of this cartridge. If some specific implementation was already provided for some elements then we merge the velocity transformed model elements with source code Velocity is a Java-based template engine. It permits anyone to use the simple yet powerful template language to reference objects defined in Java code.[Velocity]

A cartridge consists basically of two parts:

- Java classes, enhancing framework
- Velocity templates files that are used actually to perform transformation.

AndroMDA uses declarative way of applying transformations assuming that for each pattern present in the template files, the model is requested whether such type of pattern can be found in it. If yes then the transformation is applied and the output of that action is some file or files specific for given TAS.

Concerning web application development we have an andromda-struts plugin here, which is used currently for all actions to be done concerning Struts framework. AndroMDA framework as is does not meet a requirement of reading StateChart Diagrams.

For reaching the goal stated for this work I have to extend AndroMDA framework somehow. The easiest way would be to enable AndroMDA core framework read not only class diagram specification, but statechart diagrams also. Then I do not need to handle situation of processing transformations because it is already done in framework. If we observe all process of work of AndroMDA that is done to receive desired results, then I will introduce some changes on the first part while reading model elements and building AndroMDA internal model tree, and change some output templates, providing more flexibility comparing to original templates used for Struts. With this approach we do not affect internal process of AndroMDA processing of model elements, we only feed it with some additional. This will be described in section 3.3

The outcome of this proposal would be a new syntax for visual modeling of dynamic behavior of web application, that syntax will be based on a meta-model of web application that uses Struts framework. This meta-model will be described in 3.4 as well as the exact purpose of its usage.

3.1 Weaknesses of AndroMDA in the area of web application modeling

In this section I will describe AndroMDA way of representing dynamic behavior of web application more precisely, pointing out strengths and weaknesses.

I will show all possibilities for describing dynamics with help of the one scenario present in the AndroMDA example for web application. This scenario represents a use case of creating a car by the system administrator.

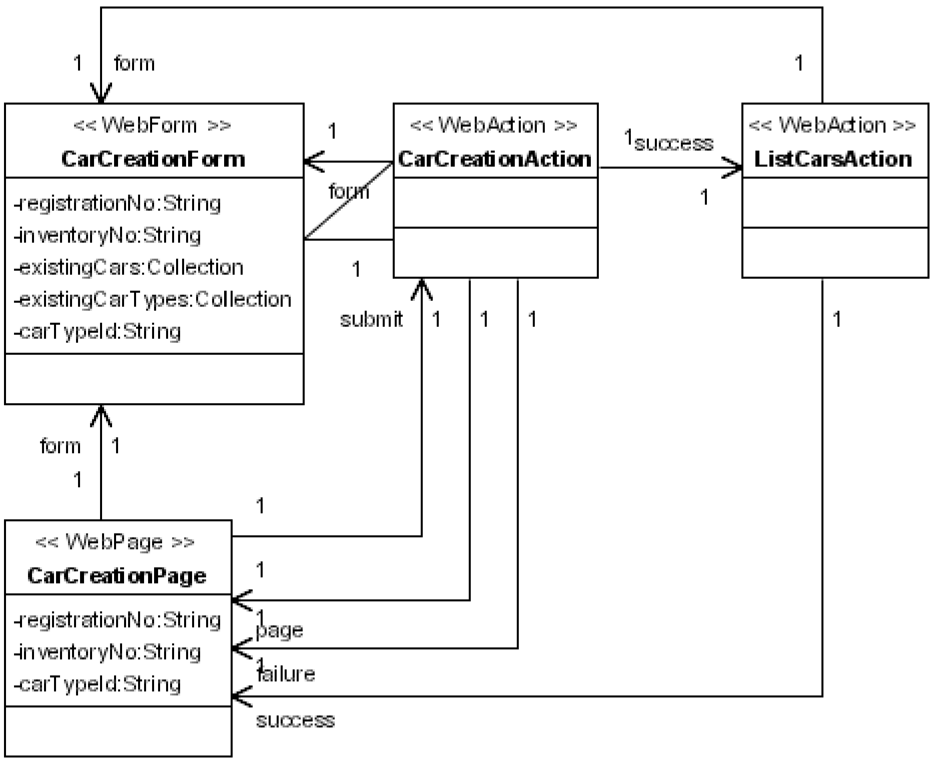


Figure 3.1 Car-Rental-System. AndroMDA example. Admin Creates Car.

We can observe three types of entities: WebForm, WebAction and WebPage. They represent objects of corresponding types of files within Struts framework: StrutsForm, StrutsAction and JspPage. For each class in the model an appropriate file will be generated.

Transitions of control focus between those classes are modeled with dependencies, where association roles serve for specific Struts notions. Thus, association role with name “failure” or “success” represents name of a “forward” from action in the Struts framework. In this representation only these names are fixed and you do not have a possibility to use another names. Role name “page” is handled as an “input” parameter of struts action. That means that an appropriate class with stereotype “WebPage” is used for representing Jsp page, which, in turn, is used for input for struts action. This is also fixed. Association with role name “form” is used for binding together struts action CarCreationAction and struts form CarCreationForm.

This way of representing transition in a web application might work for generating necessary output files and actually it does, but it is unclear what will happened if we want to show other dependencies of these classes. For instance what if our Action class has associations with other classes and why the transition from action class should be always named success or failure. What will happen if there are more then two transitions?

Here by using associations as transitions we observe some substitution of UML concerns. The class diagrams and classes are used for representation of dynamic behavior that was not meant to be. What is also lacking here is some way to model and represent dependencies between Jsp pages. That might become quite challenging to observe.

3.2 Analysis of possible alternatives

I analyzed two types of diagrams that better suit for visual representation of web application:

1. **Activity diagrams**. Here pages are represented as activities that are performed to generate a page. Activities might be nested, so that for producing one page-output we need to undertake several other activities, in the sense of *<include> or <forward>* directives.

   In this situation quite some uncertainty about how to represent struts actions. And visual representation of interaction within jsp pages itself was less informative than in following case. The activity of processing one page to produce output might depend on output from other activity. I tried to keep the layout of the this diagram close to the way of visual representation so this rendering as passing the control to other activity and back was not very clear.

2. **State diagrams.** I will consider a "State" as a processing of some resource by server, i.e. Jsp page, or Struts action or something else. The final result of this processing will be the output (as HTML) to a user. The processes might be interconnected, so that the output is the combination of outputs of several processes, the processes might be organized in a line so that the final output is the output of the last one and so on.

   Transitions might be between different states or within one page, with different data filling, so that the “the state” here is the page that is sent to the client, and “transitions” from one state to another are forced by interactions with a user by means of request-response operations. A “request” consists of some data and metadata that causes the specific transition.

After analyzing all pros and cons of each diagram representation I came to the conclusion that State Diagrams are more informative, in this case, and might be used for clearer modeling of web application. Hence, I do need to provide a possibility of usage of State Diagrams within AndroMDA framework.

There are two slightly different ways of doing that.

1. Read content of State Diagrams from source model (either XMI or MDR-repository) directly to the AndroMDA model that forces us to change or enhance framework somehow.
2. Transform State Diagrams and elements in them to Class diagrams common in AndroMDA. That would require making changes on the level of XMI representation of the model. That will restrict using the AndroMDA framework within Poseidon, because even if we would decide to write a plugin for Poseidon that will do necessary changes in the model within Poseidon, then we would not be allowed to do that, because the MDR repository can be only changed from Poseidon itself, not from a plugin.

In both approaches I will face the problem of identifying the states used for specific processes.

I have chosen the first choice, as it is much easier to implement and that is more important, I can make robust development within Poseidon. By implementing described changes I introduce new Web Dynamics Modeling framework (WDM) based on AndroMDA.

3.3 Detailed specification of required input and output data.

Here I will describe what requirements I will impose on the user model of web application so that it will guarantee applicability of WDM-AndroMDA transformations.

In AndroMDA framework itself such mechanism was already implemented. This correspondence between model elements and transformations that should be applied to them is done via “Stereotype mechanism” So that, if you want that this model element to be transformed to this and this target files, you assign this model element specific stereotype that afterwards is handled by the framework. As we know each cartridge defines set of stereotypes that it can process, so when model element with given stereotype is found, output is produced. As a remark we can mention that if there are more then one template assigned for one stereotype then several output files are produced.

Transformations made with help of described approach are one-way transformations. For making them we have to ensure that the model is “well-formed”, this well-formeness will be guaranteed by the fact that we will assign stereotypes not only for classes and class members as it was in pure AndroMDA, but for states and transitions as well. Specification and correlations between those stereotypes will be described a little bit later in this chapter.

By the output we will get full set of files necessary for working web application, we will ensure that in case of making changes to a model, after rerunning transformation on the modified model we will get freshly configured web application with all business logic inserted to classes preserved.

3.4 WDM Meta-model description. Visual syntax.

For making transformations from initial model to set of implementation classes we definitely have to make some assumptions about, what the model look like. What types of entities must be present, what might be present, and cannot be at the model. Here I am stating that I will need a model of data types and relation between those that later will be used as stereotypes for objects in user model, hence I need a meta-model for web application.

In this chapter I will explain the basics for technological space or domain that I will use in WDM-AndroMDA framework.

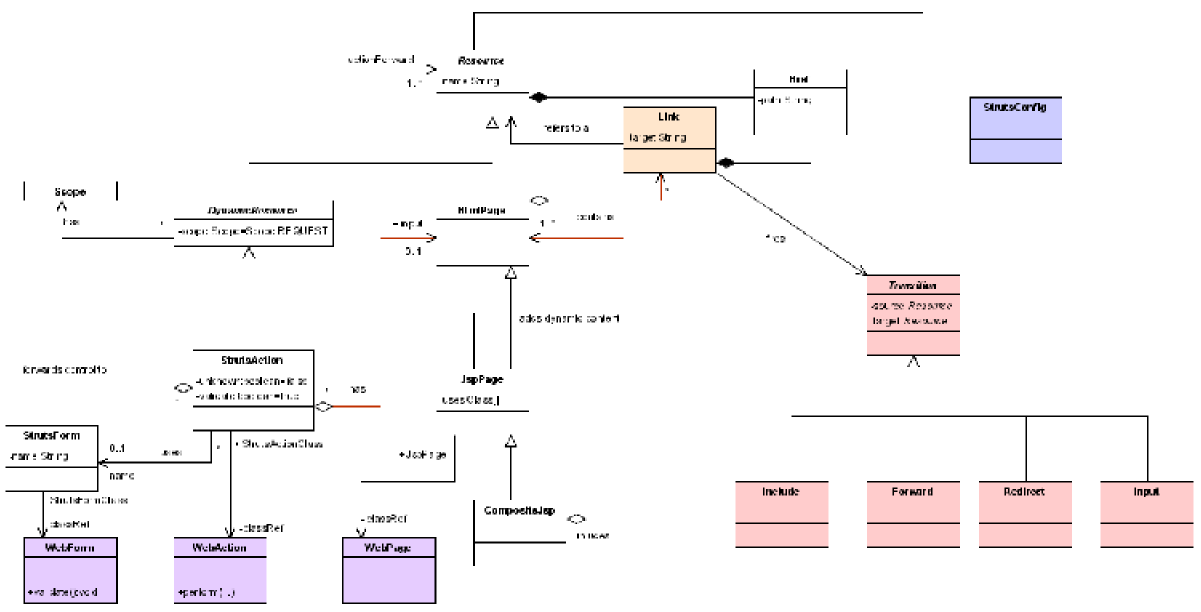


Figure 3.2 WDM. Meta-model

This diagram represents all classes that are relevant for WDM framework, it combines elements from four different target fields: Resources (white), Transitions (red) and Struts framework related implementation classes (violet) and one class that serve as Configuration class (blue).

For dynamics modeling the main group here is Resources. “Resource” is actually anything that might be accessed in web application from user point of view. Currently I am not interested on any type of media resources, so from our point of view resources might be either static or dynamic. So I am not interested in image files, media files and so on they all can be considered as static resources. User will use resources only for creating State Diagrams. This statement implies that resource will be used only for modeling interactions; they are not focused on filling web application entities with content. They serve as a facade for real implementation. The classes used for real implementation are those in violet: Struts Framework related. In user model of web application instances of these violet classes might be used to provide and add some domain specific content to generated classes.

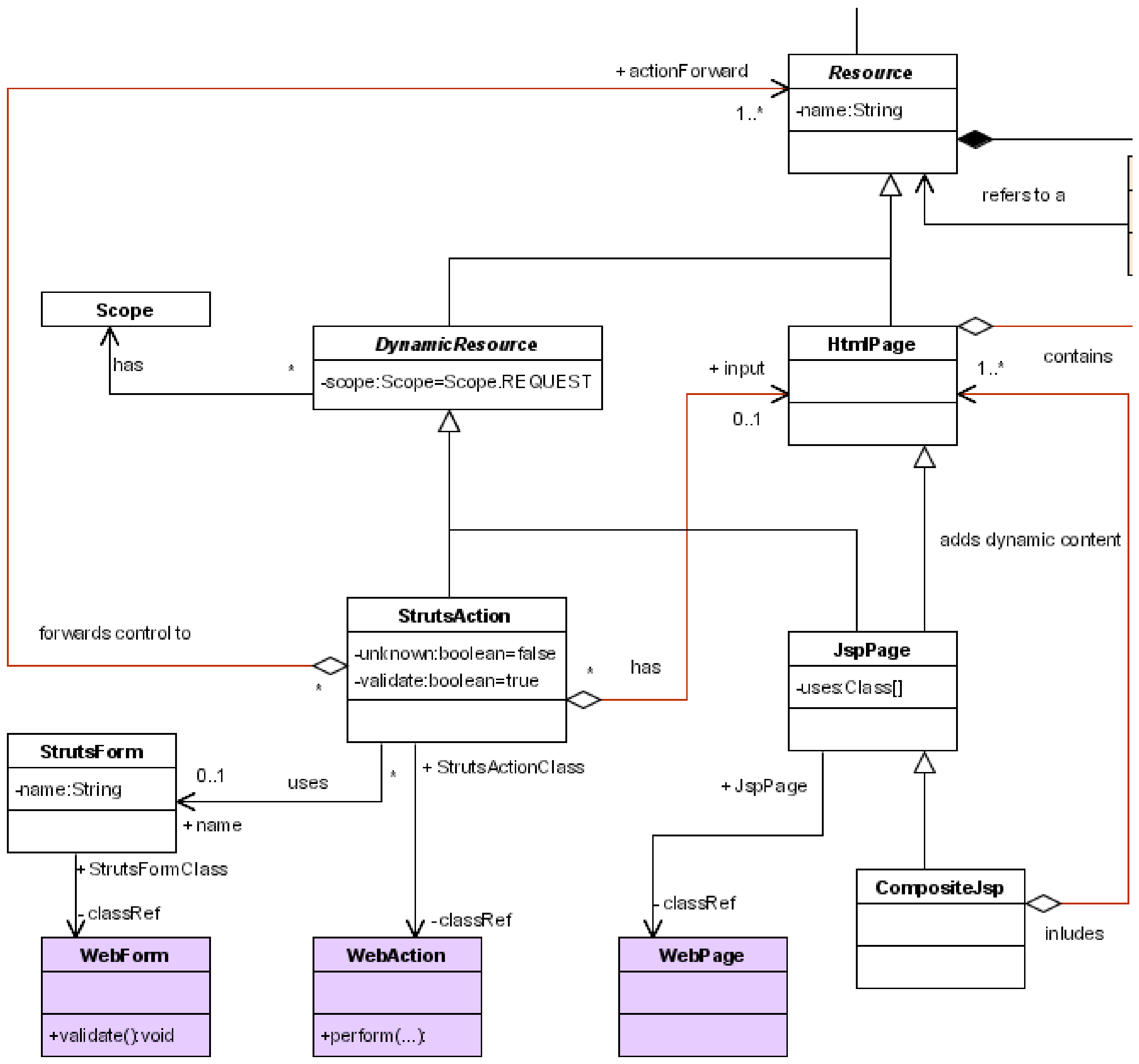


Figure 3.3 WDM. Resource and Struts related domain

A resource is characterized by its location and name, location is represented as a separate class "*Href*" that exactly specifies the unique address of the resource. The purpose of making Href as a separate class is because it might be also used as part of another very important class: "*Link*".

As I have said before we distinguish static and dynamic resources, "*Dynamic resource*" is characterized by its scope (session, page, application or request), while for static resources I consider, for given example, only static web pages. This is the reason why "*HtmlPage*" is direct descendant from *Resource* not from some static resource that would be the issue of we would consider some other types of static resources.

Concerning dynamic resources, we have here either "*StrutsAction*" or "*JpsPage*". First one describes representation of Struts action; attributes symbolize Struts action configuration parameters that will be used for generating struts-config.xml – Struts configuration file. *JspPage* class represents jsp page that can be used either as a page for representation of information or as well as input resource for Struts action. Here we consider Jsp Page to be a simple page that has no dependencies to another pages. JspPage is specialized to "*CompositJsp*" the difference between them that CompositeJsp contains another pages included into it. That might be done using <jsp:include ..> directive in source code of generated file.

StrutsAction class once again represents Action from Struts framework, has a dependency on StrutsForm, which if present in the cooperation with StrutsAction means that a control flow may pass to the StrutsForm class if appropriate attribute is specified in the StrutsAction. StrutsAction may depend on some page or descendants as an input, and has one or more Resources as forwards to which control is passed to render the request further.

Classes used as resources will be used for modeling states in the state diagrams, to bind those states with classes that will serve for realization- struts framework related we need somehow have a reference within a state to a realization class. That will be done with the help of tagged values [OMG-UML]. Ideally we would need just a small feature like attribute assigned to state that can keep reference to the class of realization, but that is not possible, so like workaround I will used special tagged value named “classRef” that will be used as this reference. The value of this tagged value, should be a name of a class associated with this state.

The same idea of tagged values will be used to specify additional, not required parameters of Struts actions, like “unknown” and “validate”. Those parameters are of type boolean so if such tagged values will be present in the state, then they must have appropriate values either “true” or “false”.

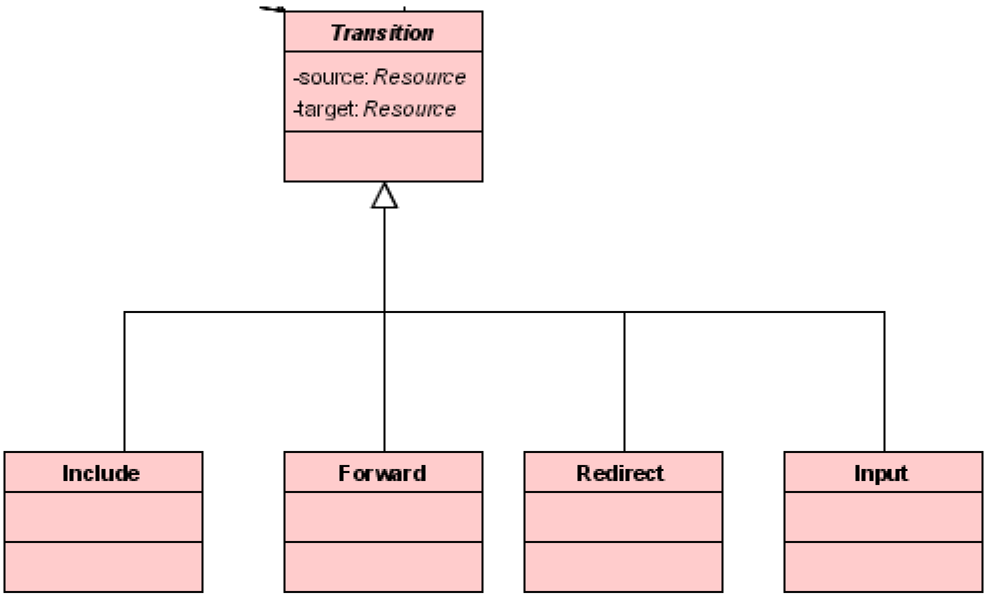


Figure 3.4 WDM. Transition domain

Class “*Transition*” represents transition from one resource to another, so actually it is the transition from one state to another. Transition might be “*Forward*”, “*Redirect*”, “*Include*” or “*Input*” each of them serves specific purpose. Forward and Redirect are used for outputs from StrutsAction, Input is used for ingoing transition to StrutsAction. Include is the one used to represent transitions and interdependencies within one CompositeJsp, that corresponds to <jsp:include > directive. Class “Link” may be considered as the one forcing transition so it might be used for representing possible transition.

Class StrutsConfig will be used as a stereotype for generating configuration file for Struts.

3.5 Bridge between Meta model and Web application model

In this part I will specify which classes will serve for what goal in the web application modeling. As it was stated before I am going to use stereotypes assigned to specific model elements so that they will meet our requirements and assumptions about web application abstractions used for modeling. That means that classes shown in meta-model will be used as stereotypes for model elements in real web application model.

Four groups of entities will be mapped each to specific group of model elements in user created model of web application. So I assume and require that

1. "Resources" stereotypes should be used for stereotypes of states.

   Remark: I only mean those classes might be used as stereotypes that are not abstract in meta-model, namely: StrutsAction, JspPage, HtmlPage, CompositeJsp and we add here a StrutsForm class. Furthermore there are some more restrictions:

   1.1 Stereotypes "*JspPage*", "*StrutsForm*" and "*HtmlPage*" should be used for Simple States in State diagram in terms of UML.

   1.2 Stereotypes "*StrutsAction*" and "*CompositeJsp*" must be the only two that should be used for Composite States

   1.3 Stereotype "StrutsForm" can be used for nested state in composite state that has stereotype "StrutsAction", in this case this means that this action uses this StrutsForm

2. "Transition" and "Link" stereotypes are used to represent transitions between states in StateDiagram

   Remark

   2.1 Stereotype "Forward" and "Redirect" can be used for transitions from the state marked with stereotype "StrutsAction".

   2.2 Stereotype "Input" can be used for transition to the state that has stereotype "Struts Action"

   2.3 Stereotype "Include" can be used to specify transition to inner states of a composite state that is marked with stereotype "CompositeJsp". This transition specifies the insertion of output of inner state to the output that produces Composite state.

3. Struts related stereotypes can be assigned to the classes in the Class Diagrams. For some state that can have a reference to the Struts related this reference must be specified in the tagged value "classRef" and set to the name of appropriate class from Class Diagram.

   3.1 States that has stereotype "StrutsAction" or "StrutsForm" MUST have such a reference. We need to have a reference in this case, cause we will need to include a fully qualified name of the references class in the configuration file for struts based web application. As a remark concerning "StrutsForm" elements we can add: if a same Struts form is intended to be used in several Struts actions, then at least one of them must have such reference.

   3.2 States that has stereotype "JspPage" or "CompositeJsp" might have such a reference, in case they do, data provided in associated class will be used for code generation for this jsp files.

   3.3 States with stereotype "HtmlPage" need not do have such reference cause the code generated does not depend on associated class.

4. Stereotype "StrutsConfig" must be assigned to one class in the model of web application this file will be used to create configuration file for Struts, namely struts-config.xml. Once again there should be only once class in the whole model with such stereotype assigned to it.

3.6 Sample usage WDM meta-model stereotypes are assigned to model elements

In this section I will provide examples of usage of those stereotypes, how they might be used in State Diagrams. I will describe different types of transition that might occur between different resources. A user will finally operate precisely with these transitions, combining them to some diagram specifying some process of interactions in web application.

Basically there are 4 different cases, I will divide them by the target of transition principle. So we have: Transition to HtmlPage, Transition to JspPage, and Transition to CompositeJsp, Transition to and from Struts Action.

1. Transition to simple html page:

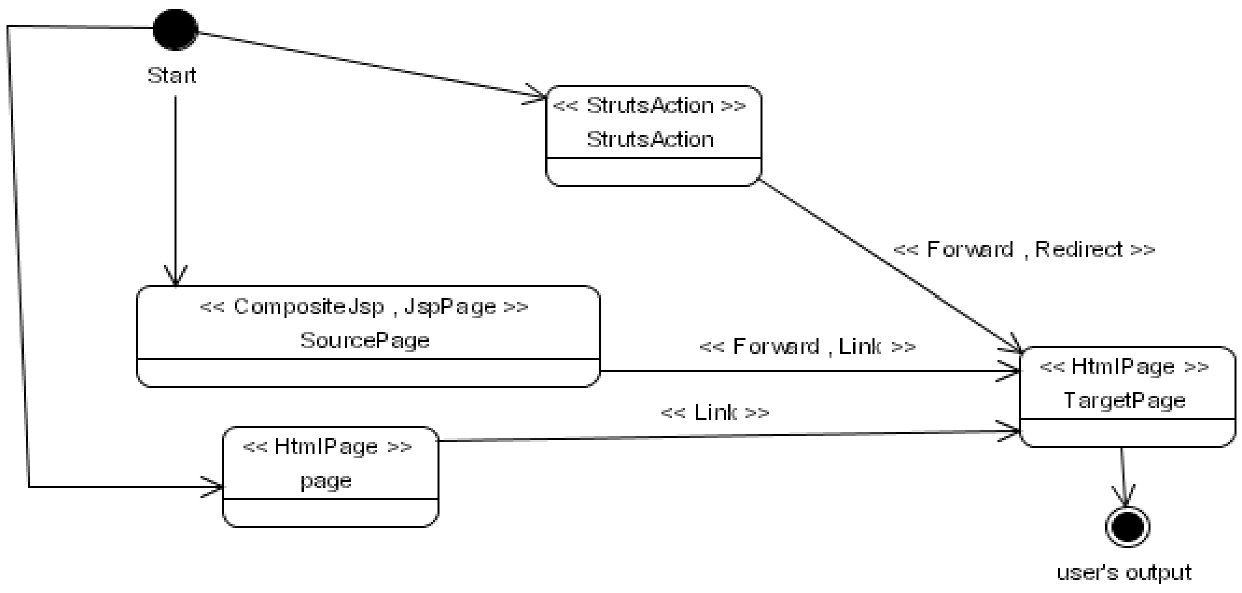


Figure 3.5 Transition to static resource.

We can reach our Target page either from another page, using the link to make transition from Source to destination. Or we can access target page as a result of Forward or Redirect processing, so that some process (StrutsAction) forwarded a request to this page.

Source page should have one of the following stereotypes: CompositeJsp, JspPage, HtmlPage. The reason for using Forward stereotype for a transition from Jsp page is that jsp sometimes can be used with some logic assign to it, so it can decide to forward a request somewhere else. But that must be omitted in correct MVC based application, where Jsps are used only for presentation purposes.

2. Transition to Simple Jsp.

There is no much difference from the previous picture, except that for creating the output to a user, we create some content on the fly.

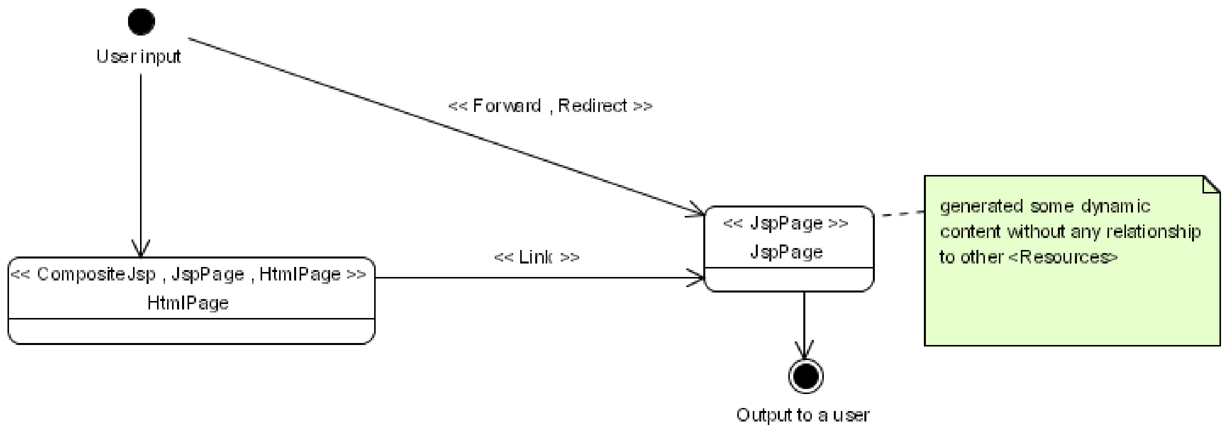

User input
<< Forward , Redirect >>
<< JspPage >>
JspPage
generated some dynamic content without any relationship to other <Resources>
<< Link >>
<< CompositeJsp , JspPage , HtmlPage >>
HtmlPage
Output to a user

Figure 3.6 Transition to jsp page.

3. Transition to Composite Jsp.

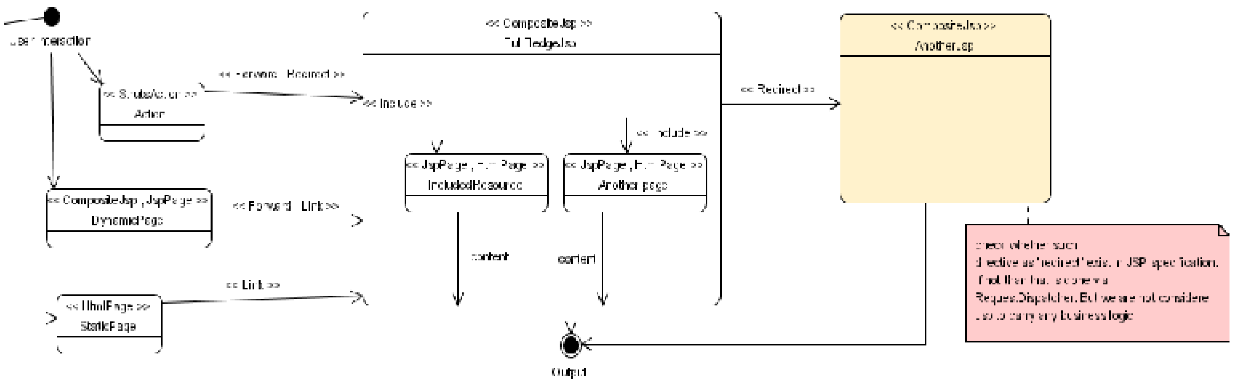


Figure 3.7 Transition to CompositeJsp

The way to reach the resource is still the same, but for producing an output to user in this case we depend on the other Jsp pages or static pages. That is symbolized by the processing of several inner jsp states within one composite state. The output might be a combination of self produced content by the composite jsp with output of included resources.

A "Redirect" transition should be checked. The idea was if jsp directive allows such kind of references then they should be shown

4. Transition to Struts action

Transition to Struts action might be as a result of usual link; in that case referencee might be used as an "input" in the Struts framework. Or might be the result of processing of previous Struts action that used such kind of forward to pass control over user's request. The main property of Struts action is that it doesn't produce direct output, as output we receive one of possible "Forwards" or "Redirects" that should be processed further to obtain the output to be sent to a user. The selection of "Forward" is done depending on the outcome of the request processing in business layer. So, while using diagrams representing transitions to Struts actions the user will need to provide all "Forwards" that might be the outcome of the Struts action.

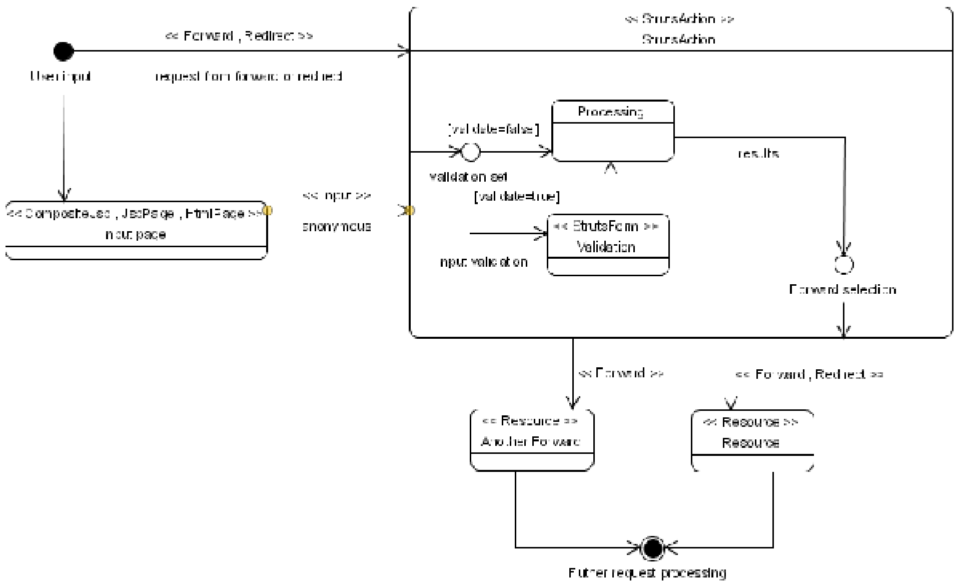


Figure 3.8 Transition to StrutsAction

Those possible transitions form full set of available to a user to design his transitions in web application. This set is full and I was not able to find out any other possible transition in web application that might not be covered with given types of diagrams

3.7 Comparison of once and many times generated target elements

Now I want to dwell on the output that we will receive after applying transformations based on the WDM-AndroMDA. How those files will be organized, how they can be changed and modified.

As for any web application that uses Struts framework we need to obtain following files:

- Configuration file, one for whole application
- Java source files representing classes used for Struts actions
- Java source files representing classes used for Struts forms
- Source files for presentation that is set of jsp pages and html pages if any.

AndroMDA framework transformation of model might be customized in whether the transformations should be undertaken for given model element or not. That is done in configuration file for specific cartridge. When you provide information about stereotypes that this cartridge is uses, at the point where you set values for specific stereotype you can set “overwrite” attribute either to true or false. What it does?

If this value is set to false, then while processing a model and finding a element with corresponding template, framework checks whether an output file was generated previously for this model element or not. If nor then it performs transformations and creates an output file. If a file already exists then no output that would overwrite possible manual changes is produced.

If a value of overwrite attribute is set to “true” then no matter was the output file present or not a new will be generated and the old one will be overwritten. How can we use that in our framework? That is really quite useful feature of AndroMDA framework that allows us to create model, then apply some changes to the model, generate transformations once again and obtain new working implementation of the system, with all business logic saved from previous transformations and manual changes.

Concerning our model I want a configuration file to be generated every time we do transformations. Source files for Struts actions and Jsp pages should be generated only once, cause afterwards we need to provide additional business logic to them, and Struts form source files can also be generated each time depending on current state of the model.

3.8 WDM vs. AndroMDA. Enhancements

This chapter will focus of what changes I have introduced to existing AndroMDA framework and what additional flexibility it provides for us.

- First of all, a new representation possibility was added that allows a new more intuitive modeling of web application.
- Representation of transitions between all parts of web application is now possible, I introduced transitions between web pages, struts actions and forms, moreover I provided more clear view on processes performed by the server to process a user’s request
- I provided more flexibility in class dependencies for Struts actions and Struts forms. These classes now can have inheritance relationship with other classes and the generated

files will have corresponding information. On the contrary, on original AndroMDA framework struts actions and forms were inherited directly and from Action or ActionForm classes of Struts framework

- I separated presentation logic from business logic, allowing user to concentrate on one issue at a time.
- I provided more flexibility in customizing Struts configuration file parameters, such as I provided a possibility to adjust such setting valid for Struts action as “validate” and “unknown”. Struts action and form in configuration file and within web application no longer have to have their names set to the names of the classes that realize them. Those values are now different, name in web application is taken as a State name, and referenced class is being assigned to this form or action in the configuration file.
- I introduced some enhancements in generated code for Struts actions. So, in original AndroMDA framework Struts action always had to have only two forwards from action, one is named *success,* another *failure.* This was not customizable. Now a Struts action can have any number of forwards, and their names are used for generating source code, that might be adopted afterwards.

3.9 Comparison to BPM4Struts

Currently the AndroMDA development team works on the next thirds release of AndroMDA, there will some considerable changes in the architecture. They are going to move from Proxy pattern to MetaModelDecorator Pattern [amdaWiki]. One of the other enhancement might be a proposal for another cartridge for dealing with web applications, namely bpm4struts.

Among stated features of new cartridge are:

- Modeling using activity graphs to express the application flow
- Modeling using class diagrams to express MVC implementation details
- Auto-generation of abstract action classes and associated skeleton implementation classes
- Auto-generation of abstract form classes and associated skeleton implementation classes
- Auto-generation of Javadoc from the documentation added by the user in the UML model
- Auto-generation of suitable method and file names from the UML model
- Auto-generation of JSP files that do not require manual updating to make them work, the available input fields will also be generated and depend on the request parameter types
- Auto-generation of struts-config.xml (done by XDoclet)
- Auto-generation of javascript validation (done by XDoclet)

We can observe quite some similarities and differences in two approaches:

Similarities:

1. Both authors (author of this work ☺ and author of bmp4struts: Wouter Zoons) agree on separating two concerns presentation workflow modeling from business logic modeling.
2. More detailed generation of output target files implementing MVC pattern
3. Almost the same concept is used for representation of workflow within web application, as you remember the possibility to use activity diagrams was discussed earlier in this paper.

Differences

1. – Author of Bmp4Struts mostly focuses on business modeling. He presents good idea about separating the customizable and not customizable behavior of Struts action classes. By means of delegation pattern he specifies a workflow of action handling in abstract action class, and forces a user to overwrite some methods in inherited class. By these actions it is possible to provide necessary decision selection for workflow of action

   – This work basically targets representation of dynamic behavior of web application in the scope of interactions with a user. Prove of this concept was done during the whole paper.
2. Bmp4strtus uses and tries to implement MetaModel-Decorator pattern while WDM currently uses proxies pattern. That is caused by the fact that the only stable release available at that time was AndroMDA 2. The architecture of WDM later can be changed to satisfy needs of new architecture requirements of AndroMDA.
3. Bmp4Struts generates configuration file for Struts1.1 on the contrary to 1.0 configuration file of WDM.

Summarizing all written above about bmp4struts vs. WDM I can assume that these 2 approaches might be combined to implement full set of requirements that might occur for modeling web application. The correspondence between authors of these two approaches assumes that this cooperation might happen:

>>“Your ideas are very welcome, I think it is best we bundle our efforts on the same cartridge,
>>rather than work in parallel on different ones.”

3.10 Integration with Poseidon

Here I will provide some justifications for integration of AndroMDA into Poseidon for UML.

Poseidon support quite customizable plug-in mechanism that allows extending possibilities of Poseidon with new features. The plug-in mechanism and integration interface allows plug-in selection and development, as well as providing a high degree of modularity.

AndroMDA framework might be integrated into Poseidon for UML modeling tool with help of special plug-in written by the lead developer of AndroMDA framework. What does it provide? As we know, AndroMDA expects a well-formed model as an input that is stored in XMI file. While reading this model it builds its internal model filling MDR repository with data stored in XMI file and later applies transformations to the created model. By using AndroMDA within Poseidon we can omit step of reading model from XMI file and creating internal MDR repository model. The reason is that Poseidon itself uses MDR repository for storing a model of designed application. So, AndroMDA reads data directly from Poseidon’s repository; that saves at least 30% of time required for performing transformations from scratch.

What do we need to make all that work, is just provide paths to the appropriate jars during start up of Poseidon. Those jars are related to AndroMDA and Java, we need to have tools.jar in the classpath of started application to be able to run some AndroMDA ant-task from Poseidon.

AndroMDA plugin extends Poseidon with set of Stereotypes that are used in AndroMDA framework. This is done in AndroMDA configuration file that is being read on the start of Poseidon. To extend that to satisfy our needs, (we need to add our own designed stereotypes) we can either edit AndroMDA configuration file, or use Profiling mechanism of Poseidon.

For enabling usage of our stereotypes while modeling of a web application I store created meta-model of Web Dynamics Modeling as a profile. Later, when we are starting new project, we load stored profile into Poseidon, by that adding all stereotypes to the Poseidon environment. Now we can assign WDM stereotypes for model elements as any other stereotypes already present in the system.

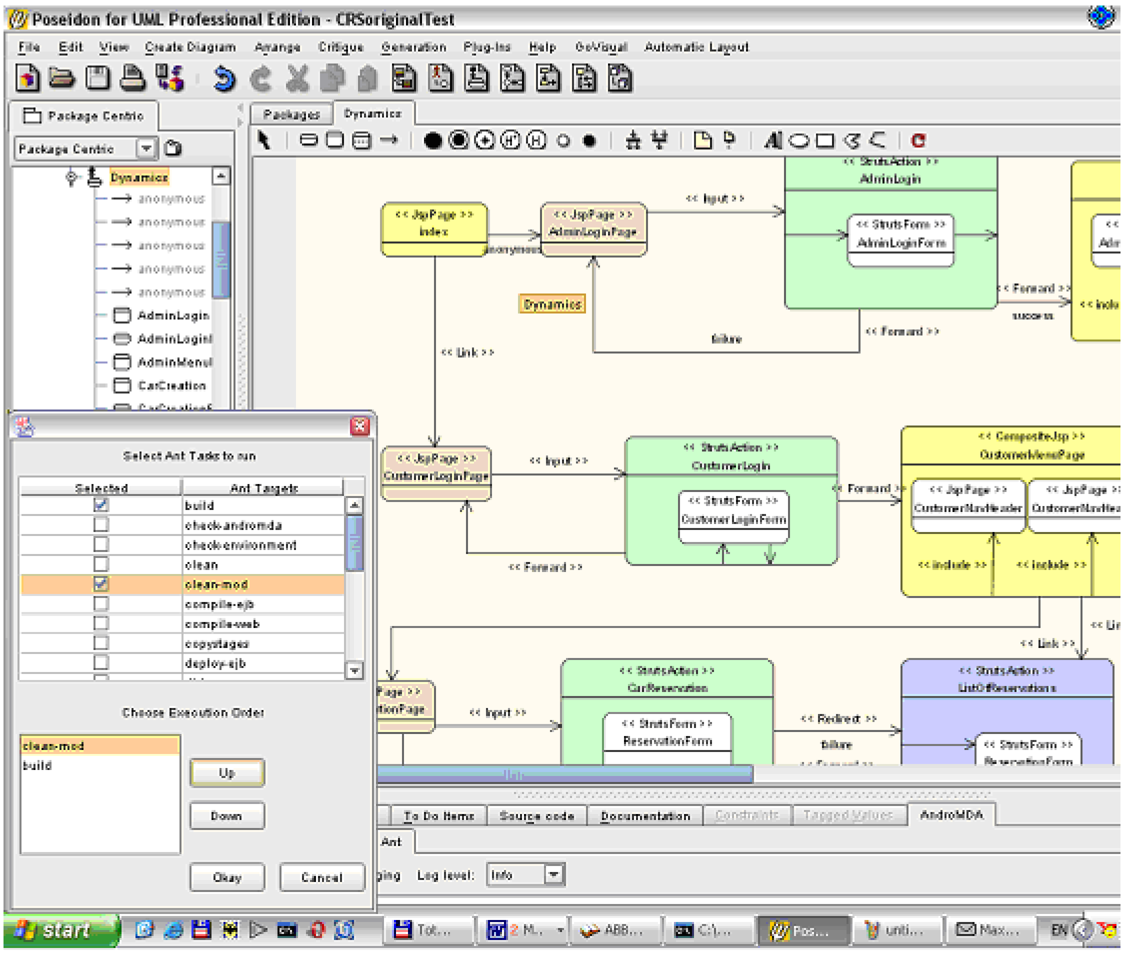

Poseidon for UML Professional Edition - CRSoriginalTest
File Edit View Create Diagram Arrange Critique Generation Plug-Ins Help GoVisual Automatic Layout
Package Centric
Packages Dynamics
Select Ant Tasks to run
Selected Ant Targets
build
check-andromda
check-environment
clean
clean-mod
compile-ejb
compile-web
copystages
deploy-ejb
Choose Execution Order
clean-mod
build
Up
Down
Okay
Cancel
<< JspPage >> index
<< JspPage >> AdminLoginPage
<< StrutsAction >> AdminLogin
<< StrutsForm >> AdminLoginForm
<< JspPage >> CustomerLoginPage
<< StrutsAction >> CustomerLogin
<< StrutsForm >> CustomerLoginForm
<< CompositeJsp >> CustomerMenuPage
<< StrutsAction >> CarReservation
<< StrutsForm >> ReservationForm
<< StrutsAction >> ListOfReservations
<< Input >>
<< Link >>
<< Forward >>
<< Redirect >>
<< include >>
Dynamics
To Do Items Source code Documentation Constraints Tagged Values AndroMDA
Ant
Log level: Info

Figure 3.9 Integration of “Poseidon for Uml” and AndroMDA.

For building our application one selects ANT build file from, and then he can select what tasks to run, arrange them in the necessary order and undergo transformations. As an output we will receive the same set of files that we would in case of using AndroMDA from shell, but much faster.

3.11 Sample project for web application development.

For proving that given concept can work I have changed existing sample web application from AndroMDA framework CarRentalSystem. The changes were done to reflect new designing and modeling capabilities. I have changed configuration of AndroMDA framework so that new files, generated by means of WDM extension, are stored in another directory then originally generated files.

By comparing a content of two different directories I afterwards can make a decision whether the new extension can be used instead of old andromda-struts cartridge.

A new state diagram was added to existing Poseidon project, representing CarRentalSystem – “Dynamics”. I added new stereotype to existing class representing configuration of web application. A full state diagram might be found in second enclosure.

The result of comparison is adequate. The sets of generated files are identical representing files for Jsps, StrutsActions, StrutsForms and configuration file. Content from WDM code generation part is slightly different from that of pure AndroMDA that was expected owing to enhancement introduced in WDM. First of all that apply to configuration file and Struts Action source files. The differences for configuration file can be observed in 2 following listings

Listing 3.1Comparison of AndroMDA and WDM.

Original AndroMDA framework	WDM-AndroMDA framework
<?xml version="1.0" encoding="ISO-8859-1"?>	<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software	<!DOCTYPE struts-config PUBLIC "-//Apache Software
Foundation//DTD Struts Configuration 1.0//EN"	Foundation//DTD Struts Configuration 1.0//EN"
"http://jakarta.apache.org/struts/dtds/struts-	"http://jakarta.apache.org/struts/dtds/struts-
config_1_0.dtd">	config_1_0.dtd">
<struts-config>	<struts-config>
<!-- ======= Form Bean Definitions -->	<!-- ========== Form Bean Definitions -->
<form-beans>	<form-beans>
<form-bean name="AdminLoginForm"	<form-bean name="AdminLoginForm"
type="org.andromda.samples.carrental.admins.web.AdminL	type="org.andromda.samples.carrental.admins.web.AdminL
oginForm"/>	oginForm"/>
<form-bean name="ReservationForm"	<form-bean name="ReservationForm"
type="org.andromda.samples.carrental.contracts.web.Rese	type="org.andromda.samples.carrental.contracts.web.Rese
rvationForm"/>	rvationForm"/>
<form-bean name="CustomerCreationForm"	<form-bean name="CustomerCreationForm"
type="org.andromda.samples.carrental.customers.web.Cust	type="org.andromda.samples.carrental.customers.web.Cust
omerCreationForm"/>	omerCreationForm"/>
<form-bean name="CustomerLoginForm"	<form-bean name="CustomerLoginForm"
type="org.andromda.samples.carrental.customers.web.Cust	type="org.andromda.samples.carrental.customers.web.Cust
omerLoginForm"/>	omerLoginForm"/>
<form-bean name="CarCreationForm"	<form-bean name="CarCreationForm"
type="org.andromda.samples.carrental.inventory.web.CarC	type="org.andromda.samples.carrental.inventory.web.CarC
reationForm"/>	reationForm"/>
<form-bean name="CarTypeCreationForm"	<form-bean name="CarTypeCreationForm"
type="org.andromda.samples.carrental.inventory.web.CarT	type="org.andromda.samples.carrental.inventory.web.CarT
ypeCreationForm"/>	ypeCreationForm"/>
</form-beans>	</form-beans>
<!-- ========== Action Mapping Definitions -->	<!-- ========== Action Mapping Definitions -->
<action-mappings>	<action-mappings>
<action path="/AdminLoginAction"	<action path="/AdminLogin"
type="org.andromda.samples.carrental.admins.web.AdminL	type="org.andromda.samples.carrental.admins.web.AdminL
oginAction" name="AdminLoginForm" scope="request"	oginAction" name="AdminLoginForm"
input="/AdminLoginPage.jsp" unknown="false"	input="/AdminLoginPage.jsp" scope="request"
validate="true">	unknown="false" validate="true">
<forward name="failure"	<forward name="success"
path="/AdminLoginPage.jsp" redirect="false"/>	path="/AdminMenuPage" redirect="false"/>
<forward name="success"	<forward name="failure"
path="/AdminMenuPage.jsp" redirect="false"/>	path="/AdminLoginPage.jsp" redirect="false"/>
</action>	</action>
<action path="/CarReservationAction"	<action path="/CarReservation"
type="org.andromda.samples.carrental.contracts.web.CarR	type="org.andromda.samples.carrental.contracts.web.CarR
eservationAction" name="ReservationForm"	eservationAction" name="ReservationForm"
scope="request" input="/ReservationPage.jsp"	input="/ReservationPage.jsp" scope="request"
unknown="false" validate="true">	unknown="false" validate="true">
<forward name="failure"	<forward name="success"
path="/ListOfReservationsAction.do" redirect="false"/>	path="/ListOfReservations.do" redirect="false"/>
<forward name="success"	<forward name="failure"
path="/ListOfReservationsAction.do" redirect="false"/>	path="/ListOfReservations.do" redirect="true"/>
</action>	</action>
<action path="/ListOfReservationsAction"	<action path="/ListOfReservations"
type="org.andromda.samples.carrental.contracts.web.ListOf	type="org.andromda.samples.carrental.contracts.web.ListOf
ReservationsAction" name="ReservationForm"	ReservationsAction" name="ReservationForm"
scope="request" input="/ReservationPage.jsp"	scope="request" unknown="false" validate="true">
unknown="false" validate="true">	<forward name="success"
<forward name="success"	path="/ReservationPage.jsp" redirect="false"/>
path="/ReservationPage.jsp" redirect="false"/>	</action>
</action>	<action path="/CustomerCreation"
<action path="/CustomerCreationAction"	type="org.andromda.samples.carrental.customers.web.Cust
type="org.andromda.samples.carrental.customers.web.Cust	omerCreationAction" name="CustomerCreationForm"
omerCreationAction" name="CustomerCreationForm"	input="/CustomerCreationPage.jsp" scope="request"
scope="request" input="/CustomerCreationPage.jsp"	unknown="false" validate="true">
unknown="false" validate="true">	<forward name="failure"
<forward name="failure"	path="/CustomerCreationPage.jsp" redirect="true"/>
path="/CustomerCreationPage.jsp" redirect="false"/>	<forward name="success"
<forward name="success"	path="/ListCustomers.do" redirect="false"/>
path="/ListCustomersAction.do" redirect="false"/>	</action>
</action>	<action path="/ListCustomers"
<action path="/ListCustomersAction"	type="org.andromda.samples.carrental.customers.web.List
type="org.andromda.samples.carrental.customers.web.List	CustomersAction" name="CustomerCreationForm"
CustomersAction" name="CustomerCreationForm"	scope="request" unknown="false" validate="true">

```
scope="request" unknown="false" validate="true">
            <forward name="success"
path="/CustomerCreationPage.jsp" redirect="false"/>
        </action>
        <action path="/CustomerLoginAction"
type="org.andromda.samples.carrental.customers.web.Cust
omerLoginAction" name="CustomerLoginForm"
scope="request" input="/CustomerLoginPage.jsp"
unknown="false" validate="true">
            <forward name="failure"
path="/CustomerLoginPage.jsp" redirect="false"/>
            <forward name="success"
path="/CustomerMenuPage.jsp" redirect="false"/>
        </action>
        <action path="/CarCreationAction"
type="org.andromda.samples.carrental.inventory.web.CarC
reationAction" name="CarCreationForm" scope="request"
input="/CarCreationPage.jsp" unknown="false"
validate="true">
            <forward name="failure"
path="/CarCreationPage.jsp" redirect="false"/>
            <forward name="success"
path="/ListCarsAction.do" redirect="false"/>
        </action>
        <action path="/ListCarsAction"
type="org.andromda.samples.carrental.inventory.web.ListC
arsAction" name="CarCreationForm" scope="request"
unknown="false" validate="true">
            <forward name="success"
path="/CarCreationPage.jsp" redirect="false"/>
        </action>
        <action path="/CarTypeCreationAction"
type="org.andromda.samples.carrental.inventory.web.CarT
ypeCreationAction" name="CarTypeCreationForm"
scope="request" input="/CarTypeCreationPage.jsp"
unknown="false" validate="true">
            <forward name="failure"
path="/CarTypeCreationPage.jsp" redirect="false"/>
            <forward name="success"
path="/ListCarTypesAction.do" redirect="false"/>
        </action>
        <action path="/ListCarTypesAction"
type="org.andromda.samples.carrental.inventory.web.ListC
arTypesAction" name="CarTypeCreationForm"
scope="request" unknown="false" validate="true">
            <forward name="success"
path="/CarTypeCreationPage.jsp" redirect="false"/>
        </action>
    </action-mappings>
</struts-config>
```

```
            <forward name="success"
path="/CustomerCreationPage.jsp" redirect="true"/>
        </action>
        <action path="/CustomerLogin"
type="org.andromda.samples.carrental.customers.web.Cust
omerLoginAction" name="CustomerLoginForm"
input="/CustomerLoginPage.jsp" scope="request"
unknown="false" validate="true">
            <forward name=""
path="/CustomerLoginPage.jsp" redirect="false"/>
            <forward name="" path="/CustomerMenuPage"
redirect="false"/>
        </action>
        <action path="/CarCreation"
type="org.andromda.samples.carrental.inventory.web.CarC
reationAction" name="CarCreationForm"
input="/CarCreationPage.jsp" scope="request"
unknown="false" validate="true">
            <forward name="failure"
path="/CarCreationPage.jsp" redirect="false"/>
            <forward name="succcess" path="/ListCars.do"
redirect="false"/>
        </action>
        <action path="/ListCars"
type="org.andromda.samples.carrental.inventory.web.ListC
arsAction" name="CarCreationForm" scope="request"
unknown="false" validate="true">
            <forward name="success"
path="/CarCreationPage.jsp" redirect="true"/>
        </action>
        <action path="/CarTypeCreation"
type="org.andromda.samples.carrental.inventory.web.CarT
ypeCreationAction" name="CarTypeCreationForm"
input="/CarTypeCreationPage.jsp" scope="request"
unknown="false" validate="true">
            <forward name="success"
path="/ListCarTypes.do" redirect="false"/>
            <forward name="failure"
path="/CarTypeCreationPage.jsp" redirect="false"/>
        </action>
        <action path="/ListCarTypes"
type="org.andromda.samples.carrental.inventory.web.ListC
arTypesAction" name="CarTypeCreationForm"
scope="request" unknown="false" validate="true">
            <forward name="success"
path="/CarTypeCreationPage.jsp" redirect="true"/>
        </action>
    </action-mappings>
</struts-config>
```

4 Directions for further work

This project might be further continued in three quite related directions. Parts of them are bound to a development of AndroMDA framework itself, and in some aspects changes that might be done are discussed in response to the new language for transformations specified by OMG.

4.1 AndroMDA 3. Decorator Pattern, Bpm4Struts

AndroMDA on its next release will no longer support proxies architecture, but will move to the Meta-model Delegation pattern. So the whole architecture of core AndroMDA and all cartridges used in it are being changed to satisfy new requirements. The state of the process is not even pre-alfa release.

Besides, as it was already mentioned, in new AndroMDA 3, the former part that was related to web application modeling will be substituted with a new cartridge bpm4struts. This cartridge is now also changing architecture to satisfy overall architecture of AndroMDA.

One of possible improvements and directions of further research might be close cooperation with bpm4struts developers so that in final plugin combine positive ideas from both present approaches. This seams to be not so difficult to implement, because WDM has not changed AndroMDA core framework very much, so that makes us hope that that changes will not require many efforts.

4.2 QVT based model elements transformations for precise source code generation

QVT stands for Query/View/Transformation is the proposal of OMG for language that will specify transformations between different models or within one model [QVT]. Currently the state of standardization process passed second revised submission, so we may hope that one of the proposals will be approved in near future.

How that can be used here? As followed from the target of QVT we will be able to create transformations not like now from model to code, but from one generic model to more specific, and so on. The first example that we can imagine might be transformations from platform independent to platform specific model. That in tern will give us more flexibility in designing applications in general and web application in particularly. By that finally we will be able to generate code more eligible for user needs and expectations.

4.3 Granularity of transformations.

Taking under consideration the previous aspect, afterward we can concentrate on more distinct transformations, so that we can generate code for operation and overwrite only that operation in the previously generated file. For instance we might want to change method signature in operation, add or remove some parameters.

It might be also possible to provide a more precise way of handling and modeling business logic in application. So, we can more precisely model workflows, and transitions and later transform that model to real application.

Conclusion:

The goal of the project was to extend the existing AndroMDA framework by providing a statechart-based notation to model dynamic behavior aspects of web application based on the Struts framework. The main achievements of this work are:

- Separation of concerns in modeling web application
- New notation for modeling transitions in web application and interactions with a user.
- More customizable code generation facilities.

An extension at the Meta-Model level for Web Dynamics Modeling is explicitly described in the paper and examples of usage are given, as well as some generated code for sample web application.

Sample project present on the AndroMDA framework is changed in compliance with new requirements imposed by the WDM specification. The project is presented in “Poseidon for UML” that allows us to observe increase of performance in applying WDM-AndroMDA transformations.

Bibliography:

[FStrt]	http://www.omg.org/faststart/program.htm
[CaseSt]	MDA Productivity case study. Middleware company. “Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach”
[Concole]	http://www.jamesholmes.com/struts/console/
[StBlder]	http://sourceforge.net/projects/rivernorth/
[WebML]	http://www.webratio.com/upload/webml_guide.pdf
[WebRatio]	Site Development Studio user guide
[PrgStr]	“Programming Jakarta Struts”, Chuck Cavaness, O’Reilly
[StrInAct]	“Struts in Action. Building web applications with the leading Java framework.”, Ted Husted, Cedric Dumoulin, George Franciscus, David Winterfeldt. Manning 2003
[StStudio]	http://www.exadel.com/downloads/struts/doc/StrutsStudioGuide.pdf
[AndroMDA]	http://www.andromda.org
[XDoclet]	http://xdoclet.sourceforge.net/
[gw.com]	http://www.gentleware.com
[OptimalJ]	http://www.compuware.com/products/optimalj/
[MDRWP]	http://mdr.netbeans.org/MDR-whitepaper.pdf
[MDRHome]	http://mdr.netbeans.org/
[Velocity]	http://jakarta.apache.org/velocity/index.html
[OMG-UML]	OMG. Unified Modeling Language Specification. Version 1.5. March 2003
[amdaWiki]	http://team.andromda.org/tiki/tiki-index.php?page=MetamodelDecorators
[QVT]	http://www.omg.org/cgi-bin/apps/doc?ad/02-04-10.pdf