Master Thesis in software Engineering and Management

Model Driven Development with Ruby on Rails

Antonios Protopsaltou

Göteborg, Sweden 2004

IT University of Göteborg

CHALMERS | GÖTEBORGS UNIVERSITET

REPORT NO. 2007/58

Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering

Antonios Protopsaltou

Department of Applied Information Technology
IT UNIVERSITY OF GÖTEBORG
GÖTEBORG UNIVERSITY AND CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2004

Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering

ANTONIOS PROTOPSALTOU
© ANTONIOS PROTOPSALTOU, 2007.
Report no 2007:58
ISSN: 1651-4769
Department of Applied Information Technology
IT University of Göteborg
Göteborg University and Chalmers University of Technology
P O Box 8718
SE – 402 75 Göteborg
Sweden
Telephone + 46 (0)31-772 4895

Antonios Protopsaltou
Göteborg, Sweden 2007

Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering

ANTONIOS PROTOPSALTOU

Department of Applied Information Technology

IT University of Göteborg

Göteborg University and Chalmers University of Technology

SUMMARY

This study presents the results from a comparison study performed in order to evaluate Ruby on Rails modeling and code generation features against J2EE environment and AndroMDA. For that purpose quality scenarios were built in order to to perform metrics measurements that in the latter stage were used to evaluate the both platforms. The scenarios' main purposes were to measure development speed, implementation complexity, modeling time, and versioning.

Through this process, it was possible to identify key features that pointed Ruby on Rails as the faster development environment. The results showed that a Rails developer could write less code (for implementing the same functionalities) than a J2EE developer. The time spent for modeling the database schema in textual environment was less than then AndroMDA visual environment. The time needed for creating a diagram of the database schema using Ruby on Rails reverse engineering plugins was less compared to J2EE. Finally, Ruby on Rails database version management allows the developer to switch between different database schemes in less and simpler steps than in J2EE or AndroMDA.

The report is written in English.

Keywords: Ruby on Rails, Model Driven Development, MDA, AndroMDA, Reverse Engineering Plugins

Acknowledgments

I would like to thank my supervisor Thomas Lundqvist for his guidance and encouragement throughout this thesis work. During this research project I benefited from his advices, particularly when exploring new ideas.

Many thanks to my patient and loving girlfriend Fofo, who has been a great supporter.

Finally I would like to thank my parents, Despoina and Stavros, for their love and support.

Table of Contents

Chapter 1 Introduction

Some problems due to ad-hoc implementation of Web-based systems which was the general trend in the 90's (Murugesan and Ginige 2005), could range from project delays to poor quality and uncompleted implementations.

There have been interesting approaches such as Model Driven Development (MDD) and Tool centered approaches that promise to reduce the productivity problems within this domain.

There exist several frameworks in the market for developing web applications, most of them claiming they increase productivity. Software engineers often find it hard to decide which one will be the best choice in selecting the appropriate one, having as criteria requirements satisfaction, development, maintenance and so on.

Most of the times software engineers/chief architects are faced to decide between choices such as MDD vs. Agile MDD, textual vs. graphical modeling, graphical modeling editors vs. Reverse Engineering tools.

This study presents the results from a comparison case study performed in order to evaluate Ruby on Rails framework and software design/ implementation strategies or methodologies in a given context. This case study has been carried in a more exploratory spirit than a strict proposal of a technology evaluation framework. For that purpose, two prototypes with the same requirements were developed, and for each of them a specific development platform was chosen (Ruby on Rails and J2EE). Furthermore, quality scenarios were built in order to perform metrics measurements that in the latter stage were used to evaluate the both platforms. The scenarios' main purposes were to measure development speed, implementation complexity, modeling time and versioning. Some of the scenarios were also applied to a third framework called AndroMDA.

Hence, the main research questions of the study are:

How does Ruby on Rails support Model Driven Development and other development quality attributes in comparison to other platforms like J2EE and AndroMDA? How efficient are the existing reverse engineering tools for Ruby on Rails?

1.1 Importance of the problem

During the 90's an ad hoc implementation process was mainly followed for the development of web applications (Murugesan and Ginige 2005). Some repercussions of this trend are visible on a survey that took place in 2000 on Web-based project development by the Cutter Consortium:

- Delivered systems did not meet business needs 84 percent of the time.
- Schedule delays plagued the projects 79 percent of the time.
- Projects exceeded the budget 63 percent of the time.
- Delivered systems did not have the required functionality 53 percent of the time.
- Deliverables were of poor quality 52 percent of time.(Ginige and Murugesan 2001)

At the World Wide Web conference in 1996 first papers with an interest in web applications problems were presented (Rashid, Zhangi and Farooque 2005). The problematic areas that were presented were the development, deployment, operation, maintenance and quality of web systems. In 1998, Murugesan along with Yogesh Deshpande and Steve Hansen have established a new discipline, Web Engineering:

“*Web engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications.*” (Murugesan et al. 2001)

Web engineering is using software engineering principles. One of the processes being used for the development of web database-based applications is the Model-driven development process. Bran Selic defines Model Driven Development (MDD) as “*a software development approach where models become essential artifacts of the development process, rather than serving an inessential supporting purpose*” (2006).

A study has shown that model driven development process increases productivity by 37% (Middleware Company 2003). However, MDD is not the only buzz word nowadays that claims to increase productivity. Hibbs (2007) argues that developing an application with Ruby on Rails (Ruby on Rails 2007) can be 5 to 10 times faster than a Java framework without making sacrifices in the quality of the application. The main reason behind this is that the framework is written in Ruby (Ruby programming language 2007) an object oriented scripting language and Ruby on Rails guiding principles: “less software and convention over configuration”.

There exist several frameworks in the market for developing web applications, most of them claiming they increase productivity. Software engineers often find it hard to decide which one will be the best the choice in selecting the appropriate one, having as criteria requirements satisfaction, development speed, complexity, scalability, maintenance and so on. For instance, a software engineer may have to choose between J2EE and Asp.Net. If J2EE is the answer, then the engineer has to decide: Struts or Spring, Hibernate or EJB and so on. Often, this decision is based on developer's previous experience with the framework or the programming language. But sometimes this is not enough, especially when evaluating new technologies having no experience with them. And a software engineer's dilemmas do not stop here:

- Model Driven Architecture (Miller and Mukerji 2004) or Agile Model Driven Development (Agile Modeling 2007)?
- Textual or graphical modeling?
- Unified Modeling Language or Domain Specific Modeling Language (Microsoft Corporation 2007)?
- Graphical modeling editors or Reverse Engineering tools ?

For all the above there exist several debates over the web with criteria similar with the ones referred when selecting development framework. Every choice has its own advantages and drawbacks. A combination of choices could even be harder to evaluate. This process could be aggravated by the incompatibilities between some choices due to their inherent characteristics. (for instance, the choice of Model Driven Architecture eliminates the choice of textual modeling). Some other times a choice may benefit through a combination of choices like choosing graphical instead of textual with a graphical modeling editor that generates code.

1.2 Related work

At the time of the study, International World Wide Web conference was calling for papers relevant to web engineering and:

- Model-driven Web application development
- Case studies and best practices
- Design models and methods (IW3C2 2007)

There exist extensive research for Model Driven Development applied in several application frameworks(Middleware Company, 2003;Bezivin et al. 2004;MacDonald, Russell and Atchison 2005; Hong and Dong 2003). Also, there are some comparative evaluations between Ruby on Rails and J2EE (Rustad 2005) or .Net, most of them on their application framework's architecture. However, there has not been conducted any study about combining Ruby on Rails and Model Driven Development. Also there are not any written experiences or studies for developing software using Ruby on Rails built-in modeling features.

1.3 Methodology

The research methodology used in this thesis work is a case study (Kitchenham and Pickard 1995). The current case study consists of a comparison of RoR against the J2EE framework and AndroMDA (AndroMDA 2007) in terms of development quality aspects, mainly focusing on development speed. It also evaluates Ruby on Rails reverse engineering tools and examines how Ruby on Rails textual modeling and code generation features aid the developer to increase his productivity.

Through this study, it is suggested the usage of model driven development approach for developing of database-backed, small-medium sized, web applications. To give a concrete example, this study involves Ruby on Rails development framework and reverse engineering process. I believe that this approach affects in a positive way the productivity and the maintenance effort of such projects. One of the assumptions here is that there is no need of designing graphical diagrams if using Ruby on Rails and a reverse engineering. The motto for this approach is: “Textual modeling is enough”. Nevertheless, it has to be mentioned that the purpose of the study is not to prove that textual modeling is better than graphical modeling.

In this case study, I used and investigated the model driven development process using textual instead of graphical modeling with Ruby on Rails for the implementation of a web database-based application. I performed a case study approach with one study unit (Vokac and Jensen 2004) – the development of an on-line survey system - and compare aspects relevant to productivity and maintenance (development speed, size, time, etc...) of it against its sister project which is going to be implemented by the first year master students. Finally, Ruby on Rails was compared in terms of modeling and code generation features with model driven development tool, Andromda.

There are six types of data collected in case studies (Tellis 1997): Documents, Archival Records, Direct observation, Participant observation, Interviews and physical artifacts. For the current study, the following data was collected and used:

- Artifacts from the project (i.e. source code and diagrams for the comparison of the different frameworks).
- Software development diary for recording personal experiences.
- Notes from direct observation technique by executing some scenarios with one of the main developers of the sister project
- Notes from participant observation technique where I executed the same scenarios in RoR environment.
- Semi structured interview with the developer of the sister project

For the comparison of Ruby on Rails features with AndroMDA's it was used active participation technique and record time, steps and personal experiences when executing scenarios in both frameworks.

1.4 Structure of the report

The structure of the report is organized as follows: Chapter 2 contains the background theory. Chapter 3 presents the method and the results of the reverse engineering plugins evaluation. Chapter 4 presents the case study methodology and the results of performed scenarios. Chapter 5 analyses the results of the comparison study and Chapter 6 concludes and mentions future research directions.

Chapter 2 Background

This chapter provides the theoretical background and explains the concepts of Model Driven Development, Model Driven Architecture, Reverse Engineering, Textual and graphical modeling. It also provides information about AndroMDA and Ruby on Rails and its reverse engineering plugins.

2.1 Model Driven Development

A model in Model Driven Development is a reduced representation of a system focusing on the properties of interest for a given view point. It is a coherent set of formal elements (Mellor, Clark and Futagami 2003) describing something, e.g. an e-commerce application, built for some purpose that is compliant to a particular form of analysis, such as:

- Communication between developers, managers and customers
- Project planning
- Reduce risks
- Transformation into implementation

According to Selic (2006) a successful engineering model should be:

- Understandable. The model should be expressed in a form to communicate the essential information directly and accurately
- Accurate. The model must correctly specify the properties of interest
- Predictive. The model must be capable to predict the behavior of the system
- Provide abstraction and hide all the unnecessary information

Today, there is a big concern on web applications development methods and processes with a view to quality and integrity (Murugesan et al. 2001) of such systems. Integration problems between design and implementation and track of changes are very common when developing web applications. Two of the most important problems when developing databased-based applications are productivity and maintenance. A typical development process for such applications is shown in Figure 1 (Bingi, Shufen and Zhao 2005)

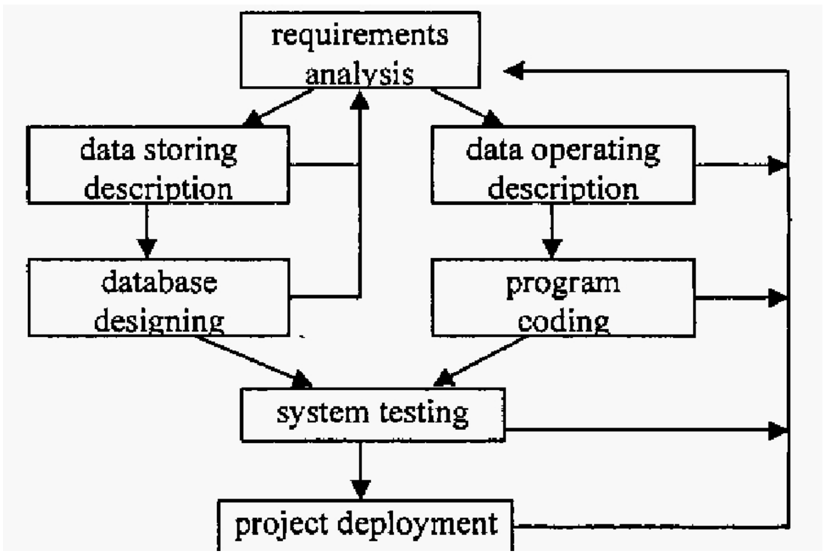

Figure 1: Flow of traditional databased development

As we can see in Figure 1 , the development process is an iterative waterfall process where a large number of documents during phases 1 (requirement analysis) and 2 (data storing description and data operating documents description) are being produced. When the system is changing the changes need to be documented. The developers need to modify the documents in order to be consistent with the source code. When the changes are often the productivity is getting low since updating artifacts by hand requires time. In addition, most of the developers consider as their main task to produce source code and not writing/updating documents which costs them time and slows down the development process. Thus, documents are rarely synchronized with the source code during later stages of projects. The result of this is maintainer needs to read a lot of code to understand the functionality and behavior of the system. Also , the developers are often forced to implement suboptimal solutions with duplicated code, violate key architectural principles, and complicate system and quality assurance because of lack of consistent artifacts of the system and the source code (Douglas 2006).

During the life cycle of a web application the software design might change lot of times because web applications in general constantly evolve (Murugesan and Ginige 2005). Use of a model driven development approach will help not only to make these changes in a short period of time but also to track the changes and have a consistency between application implementation (e.g. source code) and analysis information (design diagrams).

2.2 Reverse Engineering

Model Driven Development is following a forward engineering process, where software engineers from high-level abstractions and logical designs move to the physical implementation of the system. While forward engineering is the traditional development process, reverse engineering goes the opposite direction;it follows a bottom-up direction by generating representations of the system in another form or in a higher level of abstraction (Buss and Henshaw 1991). Understanding the source code of an application involves the reverse engineering process. Cros and Chikofsky define reverse engineering as “the process of analyzing a subject system to identify the system’s components and their interrelationships and create representations of the system in another form or at a higher level of abstraction" (1990). In nowadays there are several open source and commercial tools that support reverse engineering . The major purpose of such tools is to aid software engineers in the process of understanding and analyzing the system during maintenance activities (Rugaber 1994).

2.3 Graphical and Textual model design

Model Driven Development has been applied in many different forms. One of the most simple MDD forms is one-time automatic generation of source code from a simple model. In most of the cases, for the creation of the models of a software system, a visual/graphical modeling language is used (for example the Unified Modeling Language (UML) (Object Management Group). However, there are some people saying that graphical models are not always that effective as textual. Steffen Prochnow models statechart diagrams using a textual modeling language Esterel (Prochnow, Traulsen and Hanxleden 2006) and compares it against graphical modeling language in:

- Comprehensibility. Graphical modeling can describe better complex systems in comparison with the one dimensional flow of text. On the other had often the results of real applications are very large and unmanageable graphics making harder for the reader to use and read them. Textual modeling can represent precise details very well while graphical modeling is good for higher level of abstraction.
- Editing Speed. Entering textual programming code to specify an application is very efficient and faster than a graphical modeling language. Because of the linear text flow, inserting and deleting symbols and words is very simple. An example of the steps that needed in order to add a new state at a statechart diagram is presented at KIEL (Prochnow, Traulsen). While in textual environment there are only two steps with two lines of text, there are 19 steps at the graphical environment.
- Revision Management. When large repositories of textual code are developed, evolution is well traceable. At anytime of the project’s lifecycle a person can obtain revealing information about the increments of the programming work by comparing the versions of the

file (e.g. using WinMerge (Winmerge Development Team, 2007). According to Prochnow this can not be done in graphical modeling.

Diomidis Spinellis (2003) says that there is no rule specifying that models should have a graphical form. He proposes the use of textual languages for the creation of design models and automatic generation of graphical diagrams instead of drawing the diagrams in a visual environment. Drawing diagrams requires to place and manipulate shapes on the canvas which is time consuming and for some people, boring. Also, software engineers often focus on delivering nice pictures wasting effort on the tidy arrangement of graph nodes instead of delivering an effective design. On the negative side Spinellis mentions that learning the declarative notation might be harder than experimenting with a visual tool and his automatic generation tool of graph diagrams has some visualization problems. Another example of textual modeling is a tool for Agent UML. Michael Winikoff (Winkoff 2005) presents a textual notation for Agent UML protocols. Using this textual notation the tool can generate sequence diagrams.

2.4 Model Driven Architecture

Model Driven Architecture, MDA is the Object Management Group (OMG) methodology to model driven development process. MDA comes to solve the problematic programmer's shortcut (Figure 2) (Warmer, Kleppe and Bast 2003).

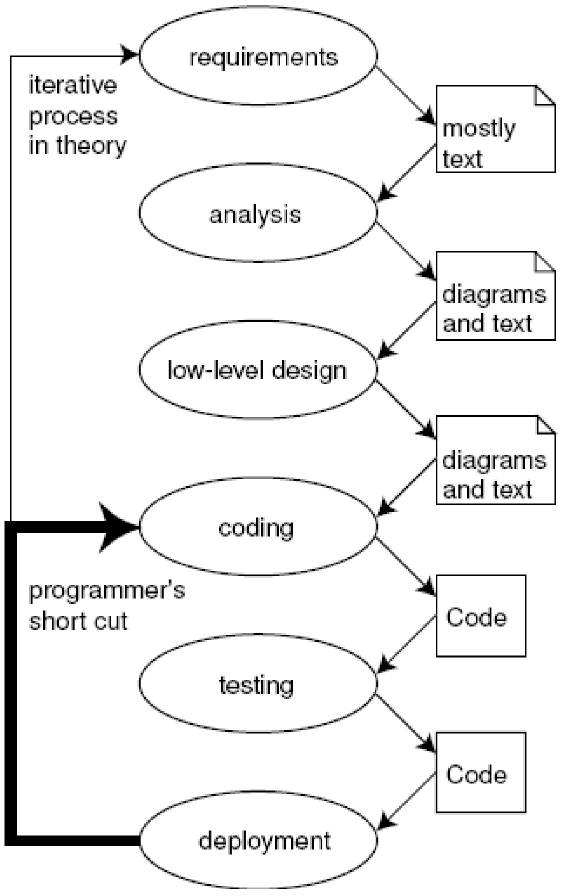

Figure 2: Traditional Software Development life cycle

As we can see in Figure 2, a developer is not following all the steps of the iterative process development life cycle. He prefers to jump to coding phase instead of updating design documents. Following an MDA process this is not possible. In MDA modeling is the central part of the development life cycle that will lead to the deployment of good applications.

The development life cycle in MDA has the same phases as the traditional development life cycle. One of the major difference between the two development approaches is the artifacts' nature that are created between the development phases and their usage. These artifacts are essential models that can be interpreted by computers. In Figure 3 we can see the core models of MDA:

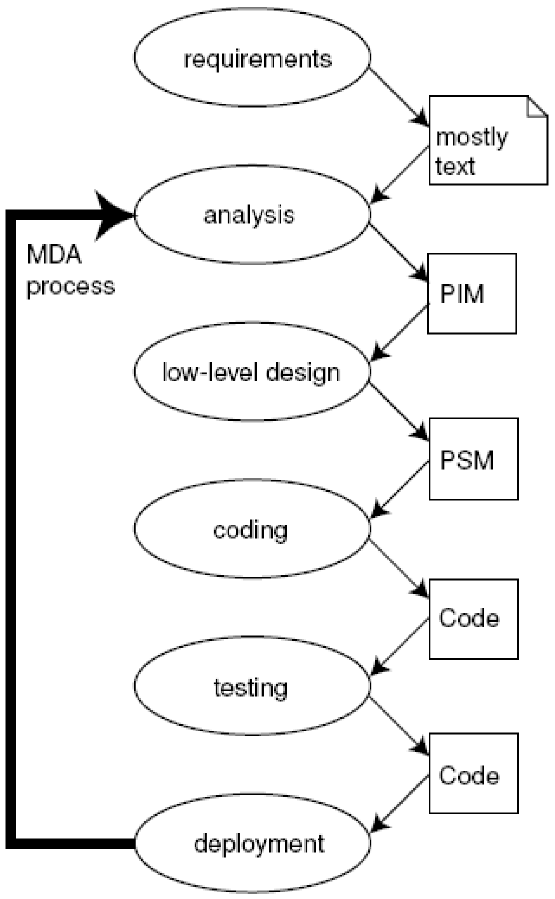

Figure 3: MDA software development life cycle

- Platform Independent Model (PIM). PIM is a description of the desired system. It specifies the structure and the functions of it excluding all the technical details.
- Platform Specific Model (PSM). PSM is the result of applying specific transformation rules to PIM.
- Code, the final step where source code is automatically generated. Figure 4(Warmer, Kleppe and Bast 2003) shows the major steps in MDA development process.

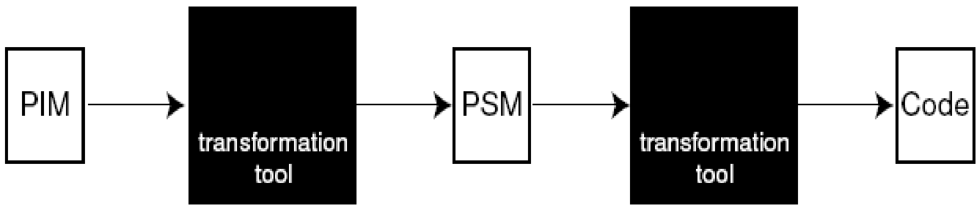

Figure 4: Steps in the MDA development process

The transformation tool (Figure 4) is responsible the translation process from model to model. There exist several MDA tools in the market like OptimalJ (Compuware OptimalJ 2007) and ArcStyler (Interactive Objects 2007).

2.5 Ruby on Rails

Ruby on Rails is a full stack development framework for developing database-backed web applications (Dave and Heinemeier 2006). Its architecture is based on the Model View Controller architecture.

Rails has an inbuilt layer for object relationship mapping named Active Record (Rails Framework Documentation). Active Record follows the standard Object Relational Mapping (ORM) model. It maps tables to classes, rows to objects, and columns to object attributes. However, Active Record is not only an ORM package for Rails. It is also the model part for Rails giving the power to the developer to define the relations among the tables using a simple-natural language such as belongs_to, has_many. An example of such is:

```
Class Department < ActiveRecord :: Base
      has_many :employees
end
```

The above code describes the one-to-many relationship between the Departments and Employees tables. Another interesting feature of ActiveRecord is the find methods that supplies to the developer:

```
e =  Employees.find_by_name(〝some_name〞)
deps = Departments.find_all()
```

The first line of code returns an Employee object with name equals to "some_name" and the second line a list of department objects. As we can see there is no need for writing Sql queries. Even when we want to save an object to the database we type:

```
e.name = 〝new_name〞
e.save
```

At the first line we change the name of an employee and at the second line we save it at the database. What we can notice from the above code (e.name = "new_name") is that there is no need to create any getter and setter methods for the objects. And also the update method (save the changes to an object to the database) for the employee object is automatically created (e.save). Rails creates all the CRUD (create, read, update, delete) methods for every object in the database.

A valuable feature in rails are the migrations. Migrations are small programs/scripts that allow the developer to incrementally create the database. These scripts are currently supported by MySql, Oracle, Sqlite, Sql Server, PostgreSql and Sybase (Rails Framework Documentation). Thus, creating the database schema using migrations, the developer can easily change to another supported database. In addition, using Rails migrations the developer has then option to switch between several versions of the database very fast and easy. Moreover, migrations are following the DRY principle of Rails (Don't Repeat

Yourself) where the column definition is specified only once (and not for example in getter methods, setter methods, configuration files).

Reverse Engineering Plugins

After conducting research I have found four reverse engineering plugins for RoR framework:

- Visualize models (Franzen 2007)
- Rails application visualizer, RAV (Sawicki and Hagelberg 2007)
- RailRoad (Smaldone 2007)
- UmlDumber (Škultéty 2007)

Visualize models, RAV and RailRoad generate pictures and UmlDumber generates an xml file to be imported to a UML tool. Visualize models and UmlDumber are using the schema.rb file- generated by RoR framework representing the actual database schema-as input data to reverse engineer. RailRoad and RAV are reading the model classes in order to identify the database tables and the relationships among them. RailRoad plugin though goes one step further. It can generate class diagrams showing inheritance associations.

2.6 AndroMDA

AndroMDA is an open source generation framework that follows the MDA paradigm. It can transform models created by UML tools (e.g MagicDraw, into deployable components of J2EE or .Net platform.

Besides the UML tools, AndroMDA requires Maven (Apache Software Foundation) for building and deploying its applications. As an application server it uses Jboss and for the tutorial example it uses MySql as database server. The environment setup is different for creating a .Net application where Microsoft Visual Studio 2005 and Microsoft SQL Server are prerequisites.

J2EE is a platform independent environment written in Java for developing web applications.

Chapter 3 Reverse Engineering Plugins

This chapter describes the methodology that performed for the evaluation of Ruby on Rails reverse engineering plugins. It also presents the results of the evaluation since the plugins are going to be used for the comparison case study (chapter 4)

3.1 Evaluation Methodology

The evaluation of the reverse engineering plugins was based on the online prototype system developed with RoR. The evaluation criteria that are used for the assessment of the four plugins are:

- Installation effort: how easy is the installation of the plugins
- Support: documentation and on-line help
- Usability
- Speed for generating diagrams
- Filters : generate diagrams with specific part of interest
- Layout : how "readable" are the generated diagrams (layout algorithm, notation)

3.2 Reverse Engineering Plugins Evaluation Results

I have executed several cases to evaluate the plugins. These are the results of the plugins evaluation based on the defined criteria:

- Ease of installation. The installation of these plugins for someone that is familiar with SVN is not difficult. Visualize models and RAV plugins can be installed by downloading the source code from the repositories and copy it to the vendor folder of the rails project. RailRoad has the easiest installation of all. It can be installed by using Gem (RubyGems) running in a command window the command: gem install railroad.
- Support. There is no detailed documentation for the plugins, probably because their capabilities are very limited. There exists supporting forums for the plugins that generate images, since all of them are hosted at hosted at RubyForge. UmlDumber in not at RubyForge server and users can post comments/questions at its blog site.
- Usability. The plugins are very usable. By running a command in a command window the developer can generate a .png picture or an .xml file.
- Speed. It took less than 5 seconds to generate a picture diagram or an xml file.
- Filters. Only RailRoad plugin gives the developer the option to specify which information should be included at the diagram like properties

type or inheritance associations and add an information label on the diagram(date, database schema version).

- Layout. The three plugins that generate pictures are using the dot layout algorithm and Graphviz software (Graphviz) which makes them more readable. UmlDumber is not using any layout algorithm. However, the developer has the power to edit the model using a UML tool, e.g StarUML and rearrange the boxes, group them, add/remove information and at the end save it as a uml model file or export it as an image. None of the plugins that generate image files is using the UML notation for creating diagrams. RailRoad plugin uses a notation which is closer to Business Object Notation, BON (Walden and Data, 1998).

A disadvantage of these plugins is that they may produce diagrams with poor readability when the number of classes is big. Another disadvantage is that they do not generate other type of diagrams (i.e. activity or state chart diagrams). Nevertheless, RoR plugins seem to provide a fast solution for reverse engineering, database schema verification and communication. Is not possible to state which is the best plugin to use in this case, since all of them have quite similar capabilities. The choice of the plugin depends on the case that is going to be used and the developer's personal taste.

Chapter 4 Case Study

This chapter contains information for the comparison case study. It describes the performed methodology and presents the results of the study.

4.1 Case study Methodology

The case study is based on an on-line survey system similar to a commercial system available at www.surveymonkey.com. The first year master students in the software engineering and management master program were developing this on-line system using java web services. I developed part of this system in the Ruby on Rails framework while paying special attention to modeling, reverse engineering and development speed aspects of the process.

Environments Comparison

The first year master students chose to develop the application in a J2EE environment using web services and enterprise java beans. MySql was chosen as the development database and Sun server as the application server. As a java IDE they were using NetBeans and for database management MySql Query Browser tool. The team was not following a model driven development approach and was using extreme programming as software development method.

In Ruby on Rails it was used default web application server WebRick and Notepad++ as an editing source code IDE. The development database was MySql.

For working with AndroMDA I chose to use MagicDraw as a UML editor and Eclipse as java IDE. As an application server I used Jboss and for the database MySQL server.

Evaluation Scenarios

In order to examine the time and steps needed for developing same functionality in J2EE and Ruby on Rails I have created some scenarios. The first scenario was to create a picture diagram of the database schema. The second scenario was to add a new table to the database, update the source code and deploy the new component in order to test the changes. The primary key of the new table (departments) should also be a foreign key to an existing table (surveys) having a one to many relationship (one department has many surveys). The third scenario was to change the name of an existing column (“name” to “survey_name”) of a table (surveys), update the code, manually update all the configuration files (without using NetBeans wizards) and deploy the component to test the changes.

While performing the scenarios at the J2EE environment and recording the steps, I conducted a semi-structure interview with the developer.

Interview Questions:

- Number of versions of the database schema
- Do you have a graphical view of the database schema

In AndroMDA environment I did not execute the above scenarios. Instead, I read about the framework, installed it and followed one of the tutorials for creating part of an employees' time tracker application in J2EE. When the prototype was deployed I added a new column at a table to the database schema using MagicDraw to record steps needed for changes to take affect.

Background of the participants

The developer that participated for executing the scenarios had 6 months experience working with enterprise java beans and web services. I have three years working experience as a web developer, last one developing only J2EE applications (using eclipse IDE, struts framework and JSP). While I was conducting this case study I had no experience in Ruby language, RoR framework, Model Driven Architecture and MDA.

4.2 Case study results

In the beginning of our session with the J2EE developer I asked him how many versions of the database they had and he answered me that there were only two. When I asked if he had graphical view of the database schemes in order to see the differences between the two versions the answer was “*no, but I can create them by using MySql Workbench tool”*. So I asked him to execute the first scenario and create a visual design of the database schema.

First Scenario

The developer spent approximately 3 minutes to open the tool, generate the visual schema of the database and rearrange the database tables. After that it took 3 minutes to export the model as a png image. During the conversion time (model to image), the tool took all the cpu usage and the memory used by the tool went from 40 to 300 Mb.

I executed the same scenario with Ruby on Rails using the reverse engineering plugins. The time for generating a diagram using single command in a command window was less than 5 seconds

Second Scenario

The J2EE developer in approximately two minutes did the changes to the database using MySql Query Browser tool for the second scenario. The next thing he did was to open NetBeans IDE and delete a package generated by NetBeans(including java files and configuration files) and by using another wizard he generated this package. The reason for doing that was to update the configuration files for the changes at the database and generate source code relevant with enterprise java beans. What was left was to create manually all the CRUD methods for the new table (departments) and change all the CRUD methods for the existing table (surveys) since he added a new column (the foreign key with reference to the departments table). I asked him to change only one method (create new survey) and test it, just to check that what he did was running. Everything was working fine.

For the second scenario I used RoR's migrations to add the new table at the database. In a command window I run the command: ruby script/generate model department. The model generator generated two files: 002_create_departments.rb and department.rb. I have added the following lines at the 002_create_departments.rb file :

```
class CreateDepartments < ActiveRecord::Migration
  def self.up
    create_table :departments do |t|
-->   t.column :name, :string
-->   t.column :description, :string
-->   t.column :manager, :string
      end
-->   add_column :surveys, :department_id, :integer
end

  def self.down
    drop_table :departments
-->   remove_column :surveys, :department_id
  end
end
```

After that I run the command: ruby db:migrate. That command made the changes to the database. As mentioned before RoR uses its own domain specific language to define the relationships among the database tables. In order to define the new relationship I had to specify it at the already existing survey.rb file and the new department.rb file:

```
class Survey < ActiveRecord::Base
      has_many :survey_pages
      has_and_belongs_to_many :users
-->   belongs_to :department
end
class Department < ActiveRecord::Base
-->   has_many :surveys
end
```

The time I spent for making these changes was approximately two minutes. And that was all. I did not have to write any other code for the CRUD methods.

Third Scenario

The developer did the changes to the database using the same tool as in the second scenario. When he opened NetBeans I asked him to update the configuration files and the java beans manually. He spent about 30 minutes trying to make the changes but he could not make it run. He told me that he tried that once at the past but these xml configuration files "*gave me a headache*". So we decided to go for the auto generation solution and do the same steps as he did at the first scenario (delete and automatically generate configuration and enterprise bean files, change the CRUD methods manually).

For the third scenario I run the command: ruby script/generate migration rename_survey_column_name. This generator generated a 004_rename_survey_column_name.rb file. I added two lines in this file:

```
class RenameSurveyColumn < ActiveRecord::Migration
  def self.up
-->   rename_column :surveys, :name, :survey_name
  end

  def self.down
-->   rename_column :surveys, :survey_name, :name
  end
end
```

The second line was not necessary. But by doing so, I could undo the changes and switch to the previous version of the database. There was no need to add or modify source code for the CRUD methods and it took less than a minute to make this change.

J2EE Environment Performance

During the execution of the scenarios, the J2EE developer said twice "*when you do programming you have to be patient*". The reason for saying that was that plenty of times he had to wait for the IDE"s response. In addition there were pop out windows messages like, "*the memory load of the system is extremely high*" or "*your system is low on virtual memory*" making his working environment very slow. This issue gave him a disappointed look. In windows task manager the memory usage during the changes was 500 mb and the "commit charge" almost 900 mb.

AndroMDA

After spending one and a half day for setting up the environment I started the implementation of the Timetracker application. AndroMDA set up the environment of the application from scratch and generated code for Spring, Struts and Hibernate frameworks. When I finished modeling AndroMDA generated all the source code. Only two lines of source code were needed to be added manually to display the contents of a database table to web browser.

Using a command in a command window AndroMDA created the database based on the model. However, when I made a change to the database model AndroMDA could not update the database. First I had to drop the table and then run AndoMDA's command to create the database.

Next table shows the steps needed to add a new column at AndroMDA and Ruby on Rails

AndroMDA	**Ruby on Rails**
add the address attribute at the UserVO box with the stereotype value object	Add the following line : `add_column :users,:address, :string`
add the address attribute at the User box with the stereotype entity	run migration script
drop table users from the database	Reverse engineering plugin to generate database schema diagram
run mvn install script	
run mvn create database schema script	

Chapter 5 Analysis

Chapter 5 presents the analysis of the comparison case study results.

5.1 RoR VS Traditional J2EE environment

Ruby on Rails and its textual modeling features provides the developer with the option to make the database changes via a text editor with the use of domain specific language. By doing so, the editing speed is enhanced. By doing so, affects the editing speed in a positive way. For instance, It is much faster to rename a column by adding a line in a text file than open a database editing IDE, load the database schema,open the table and edit the column. Nevertheless making the changes using Active Record and migrations, allows the developer to switch between different versions of the database very easily and fast. Based on my working experience, in the traditional J2EE environment this can be done with the use of a version control system, where the software engineer needs to:

- create a database schema script
- put it under version control
- update to specific revision of the "create database schema" script
- drop the database schema
- create the new database schema

Another result of modeling with RoR is that following its modeling restrictions (e.g primary key name must be id) you can get all the simple CRUD methods automatically generated. This is done with one line of code:

```
scaffolding :name_of_the_table
```

In the traditional J2EE environment the developer needs to create all these methods (a sample of the sister project is 120 lines of code for the CRUD methods of one table.

Working with RoR did not require editing any configuration file when executing the scenarios. In the traditional j2ee environment besides the updates to database it requires to be updated the configuration files between the database and their representation inside the application. For all developers this is a tedious work. NetBeans provides wizards for automatic generation of such files. In RoR these configuration files do not exist. Another interesting result is that with RoR reverse engineering plugins a developer can generate the database schema diagram in less than 5 seconds while in the j2ee environment took 6 minutes. But Ruby on Rails textual model of the database has another advantage. When it is hard to find the differences between two database versions by looking the generated picture diagrams, a developer can "diff" them and find the differences really fast.

In my opinion every developer likes fast tools with fast response. The development environment of Rails is such one. A text editor, a command window and a lightweight application server is all you need to develop a web application. During my professional carrier, I use to have plenty of times the "disappointed look" that the developer of the sister project had. This situation didn't occur with Ruby on Rails.. Moreover, during the implementation of web applications a developer wants feedback to check his changes. In J2EE environment this requires to compile the code and deploy the component. In Ruby on Rails this was not necessary. Every change I made to the source code was easily tested in a web browser by just refreshing the page.

5.2 RoR VS AndroMDA

Ruby on Rails is not an MDA tool like AndroMDA. It does not use a graphical environment for modeling and it does not follow the MDA paradigm (but still Ruby on Rails can run in every platform). In spite of this it follows a model driven development approach, by giving the option to the developer to model that database using a domain specific language and create the database schema. Since I did not perform the same scenarios as I did with RoR and J2EE I can not compare the development speed effort between the two environments.

With a focus on the database modeling I can say that RoR textual modeling language was very easy to learn and use. There is no answer on which modeling environment is better. It is a matter of :

- Developer's taste. keyboard and text or mouse, boxes, arrows and dialogs
- Switching database version method: using migrations – or drop db schema and generate a new one
- Diffing between two models: textual modeling language or xml

When it comes to database modeling speed though, textual modeling is faster. It was much faster to add a single line of code in Rails than filling in dialog boxes and recreating the database schema.

In terms of development environment performance, Ruby on Rails is the winner. WebRick memory usages in windows task manager is 25MB of memory while Jboss and MagicDraw is 250MB.

5.3 Experiences working with AndroMDA

Although AndroMDA's documentation was almost up-to-date there was nowhere specified that there are compatibility issues with the latest version of maven. That makes it hard to install AndroMDA's environment.

Modeling for AndroMDA is not the common UML modeling activity that software engineers are used to. AndroMDA requires a specific UML model structure in order to be interpreted. Also specific stereotypes are mandatory to be used to define which cartridge is going to be used for code generation. Every cartridge requires specific properties and modeling rules. From the above we understand that UML knowledge is not the only important issue for AndroMDA. I found the modeling details and rules quite complex (this could be due to the fact that I spent only one week examining AndroMDA).

An advantage of using AndroMDA is that the application's architecture is based on design patterns (e.g Data Access Object, DAO pattern). Also most of the code is automatically generated with the use of cartridges. The developers need to write only domain specific code. Another advantage is that the deployment of the application was done by writing a running a single command in a command window. There was no need to create an ant file to generate ear or war component. AndroMDA did that for me. An interesting feature of AndroMDA was the use of activity diagrams to design the process model of the application. With the use of Bpm4Struts cartridge AndroMDA generates the struts code.

On the negative side it was a surprise that AndroMDA could not update the database schema. However, in my opinion, the major disadvantage working with AndroMDA was the level of details and the specific rules of the UML models. The developer needs to have modeling abilities and I am not sure that all developers are good modelers. While MDA tools become more and more popular a new debate is on the road: graphical versus textual programming.

Chapter 6 Conclusions and future work

In this thesis I have investigated the Ruby on Rails framework, its modeling features and its reverse engineering plugins. The comparison with J2EE showed that the Rails developer gains in productivity by writing less code and having a lighter development environment. Moreover, the deployment features in Ruby on Rails gives the developer direct feedback for his changes. In addition a Ruby on Rails developer is free from the tedious work of writing configuration files. Also, the Ruby on Rails migrations showed that you can save time when creating the database schema and switching between database versions. The reverse engineering plugins can generate the database model very fast and are great for database verification although they are not following a UML notation.

AndroMDA is a great open source MDA tool. Models are always synchronized with the application's source code and the developer needs to write only domain specific code. A software engineer can easily extend it to satisfy his domain needs by writing his own cartridges.

However AndroMDA didn’t provide enough merits to be considered as a faster solution for developing web applications. I prefer to write code for the executable details of the system and reverse engineer my database schema when I want a graphical diagram of the database model.

Further investigation needs to be done to evaluate RoR scalability and the deployed application's performance. Moreover RoR reverse engineering plugins should be extended with extra capabilities and the use of UML notation. Finally, another interesting idea would an be evaluation, using the evaluation framework for software technology proposed by Brown and Wallnau (1996), of AndroMDA against a commercial tool (e.g optimalJ) and a traditional development environment.

Chapter 7 REFERENCES

Agile Modeling, 2007 : Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development, http://www.agilemodeling.com/essays/amdd.htm

AndroMDA, 2007 : http://www.andromda.org/

Apache Software Foundation, 2007: Maven, http://maven.apache.org/

Bezivin Jean, Hammoudi Slimane, Lopes Denivaldo, Jouault Frederic, 2004: Applying MDA Approach for Web Service Platform, Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conf

Bing Li, Shufen Liu, Zhao Yu, **2005** : Applying MDA in Traditional Database-based Application Development, The 9th international Conference on Computer Supported Cooperative Work in Design Proceedings

Brown Alan, Wallnau Kurt, 1996: A framework for evaluating software technology, IEEE Software

Buss Eric, Henshaw John, 1991: A Software Reverse Engineering Experience, Proceedings of the 1991 conference of the Centre for Advanced Studies on Collaborative research, Pages: 55 – 73

Chikofsky E. , Cross J., 1990: Reverse engineering and design recovery: a taxonomy,IEEE Software

Code Generation Network, **2007**: An Interview with Matthias Bohlen of UML2EJB and AndroMDA, http://www.codegeneration.net/tiki-read_article.php?articleId=8

Compuware OptimalJ, 2007 : Model Driven Java Development Tool http://www.compuware.com/products/optimalj/

Dave Thomas, Heinemeier David, 2006: Agile web development with Rails, Second Edition

Douglas C. Schmidt, **2006** : Model-Driven Engineering, IEEE Computer

Franzen Nils, 2007 : Visualize models, http://visualizemodels.rubyforge.org

Ginige Athula, Murugesan San , **2001** : The Essence of Web Engineering. Managing the Diversity and Complexity of Web Application Development, IEEE Multimedia

Graphviz, 2007: Graph Visualization Software, http://www.graphviz.org/

Hansson David Heinemeier, 2007 : Ruby on Rails, http://www.rubyonrails.org

Hibbs Curt, 2007 : ONLamp : Rolling with Ruby on Rails, http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html

Hong Wang , Dong Zhang, 2003 : MDA-based Development of E-Learning System,Proceedings of the 27th Annual International Computer Software and Applications Conference

Interactive Objects , 2007 : ArchStyler http://www.arcstyler.com/

ISWorld, **2007** : Design research in Information Systems, http://isworld.org/Researchdesign/drisISworld.htm"

IW3C2, 2007 : Sixteenth International World Wide Web Conference", http://www2007.org/cfp-WE.php"

Joaquin Miller, Jishnu Mukerji, **2003** : MDA Guide Version 1.0.1, OMG

Kitchenham Barbara, Pickard Lesley, **1995** : Case studies for Method and tool evaluation, IEEE Software

MacDonald Anthony, Russell Danny, Atchison Brenton, 2005: Model-driven Development within a Legacy System: An industry experience report, Proceedings of the 2005 Australian Software Engineering Conference

Microsoft Corporation, 2007: Domain-Specific Language Tools, http://msdn2.microsoft.com/en-us/library/bb126235(VS.80).aspx

Middleware Company, 2003 : Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach

Murugesan San, Deshpande Yogesh, Hansen Steve, Ginige Athula" 2001 : Web Engineering: A New Discipline for Development of Web-Based Systems, IEEE MultiMedia

Murugesan San, Ginige Athula, **2005** : Web Engineering: Introduction and Perspectives, Idea Group Inc.2005

Object Management Group, **2007** : Unified Modeling Language,

http://www.uml.org

Prochnow Steffen , Traulsen Claus, **2005** : Textual and Graphical Representations of Statecharts

Prochnow Steffen , Traulsen Claus , Reinhard von Hanxleden, 2006 : Synthesizing Safe State Machines from Esterel, Proceedings of the 2006 LCTES Conference

Rails Framework Documentation, 2007 : Active Record, http://api.rubyonrails.org/classes/ActiveRecord/Base.html

Rails Framework Documentation, 2007 : Migration Record,http://api.rubyonrails.com/classes/ActiveRecord/Migration.html

Rashid Ahmad, Zhang Li, Farooque Azam, **2005** : Web Engineering: A new Emerging Discipline, International Conference on Emerging Technologies

RubyForge, **2007** : Rails Application Visualizer, http://rubyforge.org/projects/rav"

RubyGems Manuals, 2007: http://www.rubygems.org/

Ruby programming language, 2007 : Rubyhttp://www.ruby-lang.org/en/

Ruby on Rails, 2007 : http://www.rubyonrails.org/

Rugaber S., 1994 : White Paper on Reverse Engineering. Georgia Institute of Technology

Rustad Aaron, 2005: Ruby on Rails and J2EE: Is there room for both? Two Web application frameworks compared, http://www.ibm.com/developerworks/web/library/wa-rubyonrails/

Sawicki Christoffer, Hagelberg Phil, 2007 : Rails application visualizer, http://rubyforge.org/projects/rav

Skultety Miroslav, 2007 : UmlDumber, http://blog.zmok.net/articles/2006/11/13/visualize-your-rails-schema

Smalddone Jaview, 2007 : RailRoad, http://railroad.rubyforge.org/

Selic Bran, **2006** : Model-Driven Development: Its Essence and Opportunities, Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing

Spinellis Diomidis, 2003 : On the Declarative Specification of Models, IEEE Software

Stephen J. Mellor, Anthony N. Clark,Takao Futagami, 2003 : Model Driven Development, IEEE Software

Takeda, H., Veerkamp, P., Tomiyama, T., Yoshikawam, H, **1990** : Modeling Design Processes, AI MagazineWinter: 37-48

Tellis Winston, **1997** : Introduction to Case Study, The Qualitative Report, Volume 3, Number 2, http://www.nova.edu/ssss/QR/QR3-2/tellis1.html

Tikhomirov Artem, **2006** : Textual Modeling Framework, http://www.eclipsecon.org/2006/Sub.do?id=438

Vokac Marek , Jensen Oluf, **2004**: Using a Reference Application with Design Patterns to Produce Industrial Software, Springer-Verlag Berlin Heidelberg

Walden Kim, Data Enea, 1998 : Business Object Notation (BON)

Warmer Jos, Kleppe Anneke , Bast Wim, 2003: MDA Explained, the Model Driven Architecture: Practice and Promise, Chapter 1

Winikoff Michael, **2005** : Towards Making Agent UML Practical, Proceedings of the Fifth International Conference on Quality Software (QSIC'05)
Winmerge Development Team, **2007** : Winmerge 2.6,
http://www.winmerge.org