AGILE PROGRAMMING AND DESIGN PATTERNS IN WEB DEVELOPMENT -A CASE STUDY

V.Dattatreya[1], Dr K.V.Chalapati Rao[1]

[1]Department of Computer Science and Engineering, CVR College of Engineering, Ibrahimpatnam, RR District, Andhra Pradesh, India

valiveti_vdt@rediffmail.com, chalapatiraokv@gmail.com

V.M. Rayudu[2]

[2]Software Engineer, Directorate of Treasuries and Accounts, Government of Andhra Pradesh, Hyderabad.

madhava.rayudu@gmail.com

ABSTRACT

Agile Programming Methodologies prioritize to minimize the risk by developing software in shorter time boxes called iterations. They emphasize real –time communication, and give preference for working Software which satisfies all stake holders. They give less prominence to documentation, as compared to other methods. Design Patterns have received a lot of attention notably from the Object-Oriented world as a technique for design reuse. Design Patterns make it easier to reuse successful designs and architectures by providing solutions to common design problems at an intangible level. A list of Design Patterns are available that can form the basis for creating designs useful for new designers and to deal with reconfigurable design. By combining the application and implementation lessons to the expansion and modifications of this list, one can make those lessons better applicable to the design community. The first section of this paper outlines the Agile Development Methodologies, the second section relates to the Design Patterns, the third section relates to Ruby on Rails, and in the last section we present a case study about Ruby on Rails to a customized accounts package as an integrated approach of Agile Programming and Design Patterns.

KEYWORDS

Agile Development methodologies, Design Patterns, XP, Scrum, Rails, Ruby, Gem, ActiveRecord, MVC, ORM.

1. INTRODUCTION

Most of the conventional software development methodologies like Waterfall Model, Spiral Model, and Iterative Model follow a predefined structured approach. Typically they take long time frames and the success rate to comply with the customer's expectations is less than 50% [1]. Since most of the current software projects fail to be completed within the short time frames, Agile Development Methodologies such as eXtreme Programming [2], Scrum [3], and Lean Development [4] have been proposed in the literature for contemporary situations for faster completion of software projects. Agile programming exploits the best feedback that comes from users interacting with working software, and this promotes early and frequent delivery of well-tested software. These methodologies mainly focus on real-time face-to-face communication.

They put less weight on documentation, compared to conventional software development methodologies.

Design Patterns facilitate reusability at design level for software as compared to Object Oriented approach that enables only code level reusability. For the past 15 years Design Patterns have been receiving attention for researchers and now several Design Patterns exist for varied application areas. Usage of Design Patterns in project development introduces standardization and reduces time frames of development. Integration of Design Patterns and Agile Programming is an ideal approach to achieve significant results to reduce time frames. In the current competitive global scenario, Web based software development requires faster product delivery. Further Web based products normally demand faster reaction times to user demands. From these two points of view, combination of the two important technologies is considered most appropriate for Web based software development. Our paper presents a case study relating to applications in the area of accounts making use of the integrated approach. In the next section we present important Agile Development Methodologies which form the basis for our proposed approach.

2. Agile Development Methodologies

Generally Agile Development Methodologies consider three groups of stakeholders. One group is the software customer, the decision maker with authority to pay for software [5]. Second group is the software users, the persons or organizational units who directly either use the output of the software, or provide the input to the software, or both. Third group is information systems personnel who develop or maintain software. This group may be centralized or distributed within the organization that has users working with systems implemented with software or external to it. Software development creates new software, whereas software maintenance deals with existing software. Software evolution is manifest in some kinds of changes over time in software [6]. While agile methods have been promoted for software development [7], their applicability in software maintenance has been noted [8, p.135].

Out of all the agile methodologies, eXtreme Programming and Scrum are considered the most popular ones, which are briefly described below.

2.1. eXtreme Programming [2]:

eXtreme Programming (XP) has evolved from traditional Software Development models which cause the long development time frames. XP is organized around short iterations. The phases that are involved in the eXtreme Programming are, Exploration phase, Planning, Iteration to Release, Productionizing, Maintenance and Death phase. In the Exploration Phase, the customers write out the user story/story card that they wish to be included in the first release.

A user story is a set of sentences that the end user wants to achieve. Planning Phase sets the priority order for these stories, and an agreement on the contents of the first release. The programmers estimate how much effort each story requires and accordingly give the schedule. Iteration to Release phase includes several iterations of the system before the first release. The scheduled set can be split into a number of iterations in Iteration and Release phase. In the Productionzing phase, new changes may still be made and decision has to be taken on whether they are to be included in the current release. Next is maintenance phase which may require recruiting new people into the team. In the last phase, viz., Death phase, no new stories are to be implemented.

2.2. Scrum [3]:

Scrum is an agile method that focuses on project management. The main idea of the scrum is that systems development contains environmental and technical variables that are likely to change during the process. Scrum has three phases .1.Pregame 2. Game 3. Postgame.

Pregame: Pregame in turn contains two phases Planning and Architecture /High level design. Planning phase includes the definition of the system being developed. Architecture/High level design includes the high level design of the system including the software architecture based on the current items in the product backlog.

Game: It is the development phase in which the system is developed in sprints. Sprints are iterative cycles where the functionality is enhanced to produce new increments.

Postgame: It is the closure of the release which is entered when an agreement has been made that environmental variables are completed. In the next section we explain the evolution and main parts of Design Patterns.

3. DESIGN PATTERNS

The original concept of Design Patterns was conceived by the architect Christopher Alexander [9]. According to him "A Pattern describes a problem, which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that one can use the solution a million times over". The concept of Design Patterns has been adopted in Object-Oriented Software Development by identifying reusable patterns of design for various existing applications. Since a pattern is a multifaceted concept, several answers can be given to the question "What is a software Design Pattern?". A Design Pattern is a common solution to a recurring problem in design, and it names and specifies the design structure explicitly. A Design Pattern tries to record a solution to a recurrent problem encountered during the design process. By using a design pattern, it is not only possible to document design experience in a very simple and comprehensible format, but also to reuse the same experience several times for different applications. One of the main reasons that computer science researchers began to recognize design patterns was to satisfy the need for good, simple, and reusable solutions. With respect to design methods, design patterns are able to transcend the constraints of a predefined design notation because they help to capture the essential, universal aspects of design problems and their solutions. A pattern can be viewed as a working hypothesis; every pattern describes the current understanding of the best game plan for solving a particular problem. The first pattern conference –Pattern Language of Program Design (PLOP) in 1994 –made an impact in Software Engineering industry and the publication of the first design patterns book "Design Patterns: Elements of Reusable Object-Oriented Software" [10] in 1995 resulted in wide acceptance of Design Patterns approach. A Software Design Pattern has four essential elements, which are given below along with the purpose of each.

1. Pattern **Name** [10] to describe a design problem, its solutions, and consequences.
2. The **Problem [10]** describes when to apply the pattern.
3. The **Solution [10]** enumerates and explains the elements that form the design, along with the relationships and collaborations among them.
4. **Consequences [10]** may address language and implementation issues as well.

We discuss some of the issues related to Design Patterns in the following section:

3.1. Pattern Description

One of the contributions of Gamma *et., al.* is to propose a consistent format and set of contents for the description of a design pattern. This particular description helps some important things required to be known in order to understand and use a pattern. Their form includes [11]:

- **Name** – a standard name for the pattern; pattern names in SMALL CAPS to classify them.
- **Intent** – What problem is the pattern addressing? What is it trying to do?
- **Motivation** – Why do we want to use this pattern?
- **Applicability** – When can this pattern be used? The difference among the patterns is that there are often different patterns for specialized contexts. What special property of the problem would make this pattern applicable or even preferred? What properties might make this pattern inappropriate?
- **Participants** – What are the elementary players in the pattern?
- **Collaborations** – How do the participants act together to solve the problem?
- **Consequences** – What are the advantages and disadvantages in using this pattern?
- **Implementation** – How do we implement this? What pitfalls should be avoided? Are there hints for optimization?
- **Known Uses** – Common examples of usage of this pattern in a real system?
- **Related Patterns** – The list of patterns associated to this one, how they work together and when to prefer one pattern over the other?

4. CASE STUDY ON RUBY ON RAILS

4.1. RUBY

In 1993 Ruby was introduced by Yukihiro Matsumoto (a.k.a Matz) as an alternative to Perl and Python. Ruby is an interpreted language which has many useful features of various languages. Ruby is an Object Oriented language and it supports only single inheritance. It also provides us with the feature called mixins [12]. By the usage of mixins we can import methods from several classes by using modules. Ruby has the scripting features similar to the older scripting languages like Python and Perl. The Object Oriented concept taken from C++ and Java helps to maintain the reliability of programming by providing the security of code. Ruby is freely available in the world. Because of this feature, Ruby is used widely by most of the software developers. Ruby has strict principles of Object Oriented programming languages (OOPS). The essential components of OOPS are Classes, Instances, Objects, Portability, Methods, and Security. **Objects** are the real world entities. These objects are used to apply for the real world problem that makes use of all functionalities available in the classes. Ruby has classes and Instances. Instances of class become objects. All algorithms based on objects are passed into the methods. A method can be defined as a set of instructions and can be called by an object. In Ruby, the entire code consists of the methods of one class or another. Ruby provides portability, using which the programmer can keep his/her code available over the network. The Ruby language is so strong that the programmer has no concern about the credibility and the safety of the code. Ruby provides the Data portability which is the Lock Down property that enables the programmer to decide which part of his code or data may be a risk to the remaining part of the code. We can also execute our Ruby programs under a tight security check by enabling the security check option. We are using a variable $SAFE which helps us to find out the safety level of the code.

4.2. Rails

Rails is a framework using Ruby language. It is designed to make programming Web applications easier, a feature every developer needs to get started. Ruby on Rails is a framework that gives support to develop, deploy, and maintain Web 2.0 applications. Rails applications gives use the Model-View- Controller (MVC) architecture. Java developers are used to frameworks such as Struts, which are based on MVC. When we are working with rails application there is a place for each code. Rails applications are written in Ruby which is an object-oriented scripting language. Any Developer can express ideas naturally and cleanly in Ruby code. Rails is based on three philosophies: code is short and readable, DRY and convention over configuration. Using DRY (Don't Repeat Yourself) philosophy every piece of knowledge in a system should be expressed in just one place. Convention Over Configuration – means that Rails makes assumptions about what we want to do and how we are going to do it, rather than requiring us to specify every little thing through endless configuration files. Rails supports ActiveRecord in the object-relational mapping (ORM) layer. Applying ActiveRecord, it is possible to manage table relationships and in the process create, read, update, and delete operations (commonly referred to in the industry as CRUD methods) are covered [13]. Action Pack acts as core of Rails applications and it contains three Ruby modules: Action Dispatch, Action Controller, and Action View. Action Dispatch routes requests to controllers and Action Controller converts requests into responses. The Action Controller makes use of Action View for the purpose of formatting the responses. Rails decodes information in the request URL and uses a scheme called *Action Dispatch* to determine what should be done with that request. At the end Rails determines the name of the *controller* that handles the particular request, along with a list of any other request parameters. In this process either of these parameters or the HTTP method itself is used to identify the *action* that has to be invoked in the target controller. Rails supports features such as Ajax and Restful interfaces that help the programmers in code integration.

4.3. Agile Approach in Rails

Agile Manifesto as a set of four preferences as given below:

a) Individuals and interactions over processes and tools: Rails is all about individuals and interactions. Rails contains simple tool sets, no complex configurations, and no elaborate processes. There are small groups of developers, their favorite editors, and chunks of Ruby code which leads to transparency and what the developers do is reflected immediately in what the customer sees. It is basically interactive process.

b) Working software over comprehensive documentation: Rails does not support documentation and specifications. Rails delivers working software early in the development cycle.

c) Customer collaboration over contract negotiation: Rails project can quickly respond to changes required by the customers, who are thereby convinced that the team can deliver what is required, not just what has been requested.

4.4. Rails MVC Architecture

Rails is a Model-View-Controller framework. Rails receives requests from a browser, decodes the request and calls the corresponding method in the appropriate controller. The controller then evokes a specific view to display the results to the user in Web Environment. Generally if we want to create an application in Rails we need to code for a controller and view and make a route between these two. To create a rails application use *rails generate* command, it will create some files such as HTML files, and JavaScript files and so on. Any modification to the existing HTML file can be done by opening the views directory and then modifying/adding the code to that HTML file with .html.erb extension means extended ruby file. Fig 1 [14] explains how rails process the user requests based on router.

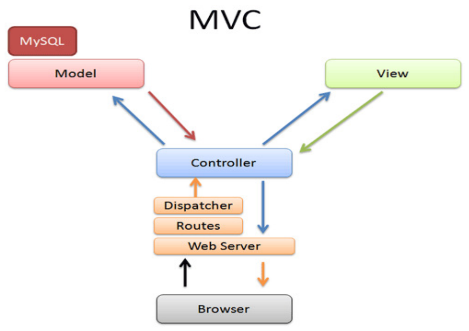


Fig 1

4.5. Rails Support Object –Relational Mapping

Designing software to connect an object-oriented business system with a relational database is a tedious task. Object-orientation and the relational paradigm differ quite a bit. An application that maps between the two paradigms needs to be designed with respect to performance, maintainability and cost to name just a few requirements. There are numerous patterns of object/relational access layers, but looking at the body of pattern literature we will find that some patterns are still to be mined, while there is no generative "one stop" pattern language for the problem domain [15]. **Object-relational mapping** (**ORM**) is a programming technique for converting data between incompatible type systems in object-oriented programming languages. ORM libraries map database tables to classes. In Rails if a class is defined as Order then the database has a table called orders. Rows in orders table are mapped to objects of the class—a specific order is represented as an object of class Order. The individual columns are set making use of the object attributes. ORM layer represents tables as classes, rows as objects, and columns as attributes of those objects. Class methods are used to perform table-level operations, and methods can perform operations on the individual rows of a table.

Fig 2 explains the concept of ORM, according to Martin Fowler from Patterns of Enterprise Application Architecture (P of EAA) [16]

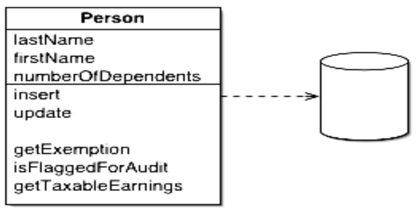


Fig 2

5. Integrating Agile Development Methodologies and Design Patterns in Web Development:

Now we examine how to integrate the methodologies Agile Development methodologies and Design Patterns to Accounts Package called IMPAct [17] in Directorate of Treasuries and Accounts, Government of Andhra Pradesh. We are applying some of the agile principles such as *short development life cycles* that pay attention to shorter time duration for completing the project. It is different from the traditional Software Development Methodologies that involve higher cost and time of development than the Agile model. Secondly we follow *Testing each iteration,* as each iteration includes new features to project, we implement automated tests to make sure they work right. This way we squash more bugs and have greater confidence the next time we change or add features. We initially start with basic features, later on add some requirements with minimal documentation.

ActiveRecord: ActiveRecord is the simplest of database design patterns originally published by Martin Fowler in his book Patterns of Enterprise Application Architecture. David Heinemeier Hansson (the creator of the Rails framework) took the concepts laid out by Fowler and implemented them as a Ruby library called Active Record. ActiveRecord has been released with the Rails framework to the public and is also available as a part of the core bundle with its own Ruby Gem. ActiveRecord differs from other ORM Libraries because it makes a number of configuration assumptions, without requiring any outside configuration files or mapping details. With Rails, the model section is generally Active Record classes and other data-descriptive or data communication code. The view section is mainly for the user interface which involves elaborate HTML code in Rails applications. Fig 3 shows the Architecture of ActiveRecord.

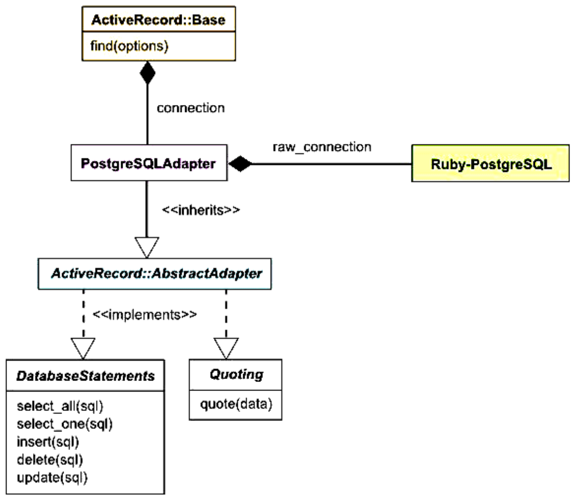
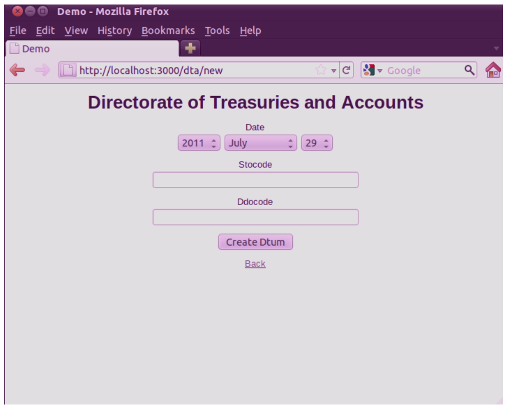


Fig 3

We implement the existing system into ORM using ActiveRecord. The knowledge of how to interact with the database is directly embeded into the class performing action, by making use of ActiveRecord. It is the ORM layer supplied with Rails. We will build on the mapping data rows and columns using Active Record to manage table relationships and in the implementation of CRUD Methods. Instances of ActiveRecord classes correspond to rows in database table. The Instances have attributes corresponding to the columns in the table. ActiveRecord determines the columns of the corresponding class dynamically. The ActiveRecord object attributes in general correspond to the data in the row of the corresponding table. Fig 4 shows one of the sample screens of the project. Active Record is database-neutral, so it does not care which database software we use, and it supports nearly every database. Since it is a high-level abstraction, the code we write remains the same no matter which database we are using. For the present project we use PostgreSQL.


Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Demo
http://localhost:3000/dta/new
Google
Directorate of Treasuries and Accounts
Date
2011
July
29
Stocode
Ddocode
Create Dtum
Back

6. CONCLUSIONS

Design Patterns help in faster and more effective deign for software, with reduction in effort achieved through making use of existing/standard patterns in design. Agile programming is a software development methodology better suited to the current demands of shorter time frames and easily adaptable software. A combination of the two approaches, adopting Scrum [3] as the Agile methodology is attempted in the work presented in this paper. Our proposed approach is being applied to meet the application requirement where the Web pages are rapidly changing based on the onsite customer demands. In this paper we have given the basic database Design Pattern i.e. ActiveRecord. ActiveRecord follows the benefit of simplified configuration and default assumptions, automated mapping between tables and classes and between columns and attributes, database abstraction through adapters, direct manipulation of data as well as schema objects. It can be integrated into other frameworks like Merb. Compared to traditional techniques of exchange between an object-oriented language and a relational database, Object Relational Mapper (ORM) using ActiveRecord often reduces the amount of code that needs to be written. ActiveRecord attempts to provide a coherent wrapper as a solution for the inconvenience involved in direct implementation of object-relational mapping. Object-Relational Mapping is primarily used to minimize the amount of code needed to build a real-world model. The software used is based on the Ruby on Rails and PostgreSQL.

References

[1] The Standish Group, "The Chaos Report," West Yarmouth, MA: 1995.
[2] Beck .k, "Extreme Programming Explained –Embrace Change," 2/e, Addison Wesley, Boston, MA, 2005
[3] Beedle .M, Schwaber. S, "Agile Software Development with SCRUM," Prentice Hall, Upper Saddle River, NJ, 2001
[4] Poppendieck. M, Poppendieck. T, "Lean Software Development: An Agile Toolkit," Addison-Wesley, Boston, MA, 2003

[5] Ned Chapin, "Agile Methods' Contributions in Software Evolution," in Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04)

[6] N. Chapin, et al., "Types of software evolution and software maintenance," Journal of Software Maintenance and Evolution, John Wiley & Sons, Chichester UK, January 2001, pp. 3–30.

[7] Agile Alliance, Manifesto for Agile Software Development, 2001. http://www.agilemanifesto.org/

[8] K. Beck, Extreme Programming Explained, Addison-Wesley, Boston MA, 2000.

[9] Alexander C., et al *Pattern Language: Towns, Building and Construction.* New York: Oxford University Press, 1977.

[10] Erich Gamma. , et al. "Design Patterns Elements of Reusable Object-Oriented Software," Singapore, Pearson Education, 2003

[11] Andr´e DeHon., et.al. "Design Patterns for Reconfigurable Computing". *IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM 2004), April 20–23, 2004.

[12] http://www.rubyist.net/~slagell/ruby/

[13] Sam Ruby., et al. "Agile Web Development with Rails", 4th Edition, The Pragmatic Bookshelf.

[14] http://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/

[15] Wolfgang Keller, "Object/Relational Access Layers A Roadmap, Missing Links and More Patterns" in Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing, 1998.

[16] http://martinfowler.com/eaaCatalogindex.html

[17] http://treasury.ap.gov.in

Authors

Mr. V.Dattatreya is currently working with CVR College of Engineering in the Department of Computer Science and Engineering as an Asst Professor. He is pursuing his part time PhD from Jawaharlal Nehru Technological University. He received his M.Tech in Computer S cience and Engineering from I.E.T.E. His area of interest includes Databases, Software Engineering, Software Architecture and Design patterns, Web Technologies.

Dr K.V. Chalapati Rao is a Professor of Computer Science & Engg., and Dean, Academics at CVR College of Engineering. Prior to joining the CVR, he served Osmania University as a Professor & Head, Department of CSE and Dean of Eng ineering. After obtaining his PhD, Dr. Rao joined Electronics Corporation of India Limited and worked in various capacities for 16 years, before joining the Osmania University. He guided number of PhD scholars in areas of Real time systems, Operating Systems, Software Engineering, Distributed Systems, Knowledge and Data Engineering.

Mr. V.M.Rayudu is working as Software Engineer in AP Techno logy Services Ltd (A Government of Andhra Pradesh Enterprise), now on Treasuries & Accounts Department projects. His job role includes Rapid Deployment of Web Applications using Agile techniques, Vulnerability Assessment of Applications, Networks and Systems Software and Intrusion Detection.