International Journal of Computer Applications Technology and Research
Volume 6–Issue 8, 391-395, 2017, ISSN:-2319–8656

Model View Mapper Architecture for Software Reusability

Chethana S
Lecturer,Dept.of Computer Science
NMKRV PU College
Bangalore, India

Dr.Srinivasan
Professor
RV Engineering College
Bangalore,India

Abstract: Design Pattern Architecture is a serious issue in the development of any complex software system for Small, Medium and Big Organization. The essential problem is how to incorporate rapidly changing technology and new requirements by composing patterns for creating reusable designs. The main objective of this proposed work is to enhance the performance of enterprise design pattern reuse, to monitor software component problems and to predict the best design pattern for reusability. As a solution for all these problems a new framework called MVM pattern approach is proposed for migrating to a new design approach. It is helpful for all kinds of the organization complex software developments.

Keywords: Software Reuse, Architectures, Framework, Design Patterns, MVC, MVM, MVVM, MVP

1. INTRODUCTION

Software Reuse is characterized as the way toward building or collecting software applications and frameworks from the existing software. By reusing Software pattern there are many advantages includes time can be saved, increase productivity and also reduce the cost of new software development. The proposed work aims on developing a framework for software pattern reuse in enterprise level applications. Frameworks give a standard working structure through which client's primary aim is on creating desired modules than creating lower level points of interest. By utilizing this facility the software designers can invest more time in building up the prerequisite of software, instead of setting up the tools of application development. The framework is a set of the reusable software program that structures the basis for an application. Frameworks help the developers to assemble the application rapidly. At its best code reuse is refined through the sharing of regular classes or collection of methods, frameworks, and techniques.

2. LITERATURE SURVEY

In Neha Budhija [1] expert designers have done an empirical study of the software reuse activity with the concept of object-oriented design. The study concentrated on fundamentally three aspects of reuse : (1) the communication between some design forms (2) the mental procedures required in reuse (3) the mental portrayals developed all through the reuse action. In FENIOSKY PENA-MORA [2] introduces an in-advance improvement of a framework for utilizing design rationale and design patterns for creating reusable programming frameworks. The work describes the use of an explicit software creation procedure to catch and disseminate specific knowledge that augments the depiction of the cases in a library during the development process of software applications by heterogeneous gatherings. B.JALENDER [3], the authors described about how the code level reusable components can be built and how the code level components can be designed. It also provides some coding guidelines, standards and best practices used for creating reusable code level components and guidelines and best practices for making configurable and easy to use. Tawfig M [4] the authors have presented the concept of reuse at design level in more details. Also, the work proposes an approach to improve the reusability of software design by using the concept of directed graph. The outcome of the proposed work is to produce a design to be considered as reusable components which can be adapted in many software systems. Erich Gamma[5] proposed design patterns as a new mechanism for expressing object-oriented design experience and they described that the design patterns can be considered reusable micro-architectures that contribute to an overall system architecture. Authors described how to express and organize design patterns and newly introduced a catalog of design patterns. In Reghu Anguswamy [6] provided a generic list of reuse design principles for component based software development which is based on a preliminary analysis of the literature of software reuse and reuse design over the past few decades. Authors suggested that the proposed list is new since the reuse design principles presented in the past were specific to programming languages, domains, or programming paradigms. William B[7] In their paper authors have briefly summarized about software reuse research, discussed major research contributions and unsolved problems in the proposed area, they provided pointers to key publications. Sajjan G [8] in this paper the authors have done an attempt to answer some unsolvable questions. Authors pointing out that it have been more than three decades since the idea of software reuse was proposed. They have done a research on how far are investigators with software reuse research and practice. In CHARLES W[9] have done a survey on various approaches to software reuse found in the research literature. Some taxonomy

has been used to describe and compare the different approaches and make generalizations about the field of software reuse. The taxonomy used in the work is used to characterize each reuse approach by its reusable artifacts and the way how are organized. SathishKumar[10] ,the authors have given a brief summarization of present research status in the field of software reuse and major research contributions. Some future directions for research in software reuse are also discussed.

Model view mapper pattern

There are many different approaches existing for software reusability such as MVC(Model View Controller),MVM(Model View Presenter) and MVVM(Model-View-ViewModel) and all these approaches have its own advantages and disadvantages. So a new framework is proposed for Software design reusability called Model View Mapper Pattern.

The Model represents a set of packages instead of business logic classes i.e. business model as well as database access

authority along with server side validation i.e. data model. It also defines business rules for data means how the data can be changed and manipulated then updated in to database along with proper validation. The View represents the UI Components like CSS, jQuery, Angular.js, Ajax, html etc. It is also represents as database access authority along with server side as well as client side validation. It is not only responsible

for displaying the data that is received from the mapper and from the database as the result. The Mapper is responsible to process the incoming requests. It receives valid input from users via the View, then process the user's data with the help of Model as well as View and passing the results back to the View. Typically, it acts as the coordinator between the View and the Model. Figure[1] describes the flow of how the communication takes place in the proposed MVM pattern architecture.

Code Classifier & Code Analyzer Algorithm is used in the work to classify and count the number of class name, abstract class name, method name, and abstract method name, interfaces used in the program, iteration counts and the number of lines in the file present in the file. **Tracing Behavioral Dependency algorithm** is used to identify the Dependent Class Name & Depending Class Name in separate table.

Algorithm 3: Hybrid ABC-CM➔Artificial Bee Colony + Naïve Bayes Classifier Model

1: Initialize the population of solutions Bee i, j ,i = 1 ...EL, j = 1 ...P

2: Evaluate the population

3: iteration=1

4: repeat

5: Produce new solutions Val i, j for the employed bees by using (2) and evaluate them

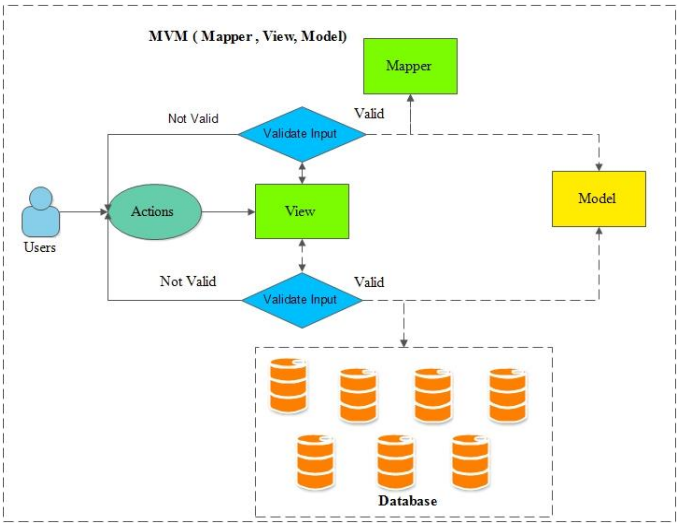
6: Apply the greedy selection process in Bees

7: Calculate the probability values Pop i, j for the solutions Bee i, j by (1)

8: Produce the new solutions Val i, j for the onlookers from the solutions Bee i, j selected depending on Pop i, j and evaluate them

Figure1. Proposed MVM pattern Architecture

9: Apply the greedy selection process



10: Determine the unrestrained solution for the scout, if exists, and replace it with a new randomly produced solution Bee i, j by (3)

11: Memorize the best solution achieved so far.

12. Let **best solution** be a training set of samples, each with their class labels. There are n classes, Cls1, Cls2, . . . ,Clsn.

Figure 2. Sequence Diagram for MVM

Each sample is represented by an n-dimensional vector, X = {x1, x2, . . . ,xn}, depicting n measured values of the n attributes, Attb1, Attb2, . . . , Attbn, respectively.

13. Given a sample X, the classifier will predict that X belongs to the training set taking the highest a posteriori probability, conditioned on X. That is X is predicted to belong to the class Clsi if and only if

$$P(Clsi|X) > P(Clsj|X) \text{ for } 1 \leq j \leq m, j \neq i.$$

Thus we find the class that maximizes P(Clsi |X). The class Clsi for which P(Clsi |X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(Clsi|X) = P(X|Clsi)\, P(Clsi)\, P(X) .$$

14. As P(X) is the same for all classes, only P(X|Clsi)P(Clsi) need be maximized. If the class a priori probabilities, P(Clsi), are not known, then it is commonly assumed that the classes are equally likely, that is, P(Cls1) = P(Cls2) = . . . = P(Clsk), and we would therefore maximize P(X|Clsi). Otherwise we maximize P(X|Clsi)P(Clsi). Note that the class a priori probabilities may be estimated by P(Clsi) = freq(Clsi , T)/|T|.

15. Given data sets with many attributes, it would be computationally expensive to compute P(X|Clsi). In order to reduce computation in evaluating P(X|Clsi) P(Clsi), the naive assumption of class conditional independence is made. This assumes that the values of the attributes are temporarily independent of one another, given the class label of the trial. Mathematically this means that

$$P(X|Clsi) \approx n \text{ PI } k=1 \; P(xk|Clsi).$$

The probabilities P(x1|Clsi), P(x2|Clsi), . . . , P(xn|Clsi) can easily be predictable from the training set. Recall that here xk refers to the value of attribute Attbk for sample X.

(a) If Attbk is categorical, then P(xk|Clsi) is the number of samples of class Clsi in T having the value xk for attribute Attbk, divided by freq(Clsi , T), the number of sample of class Clsi in T.

(b) If Attbk is continuous-valued, then we typically assume that the training values have a Gaussian distribution with a mean μ and standard deviation σ defined by

$$g(x, \mu, \sigma) = 1 \sqrt{2\pi\sigma} \exp - (x - \mu) 2 \; / 2\sigma 2 \text{ , so that}$$

$$p(xk|Clsi) = g(xk, \mu Clsi , \sigma Clsi).$$

We need to compute μClsi and σClsi , which are the mean and standard deviation of values of attribute Attbk for training samples of class Clsi .

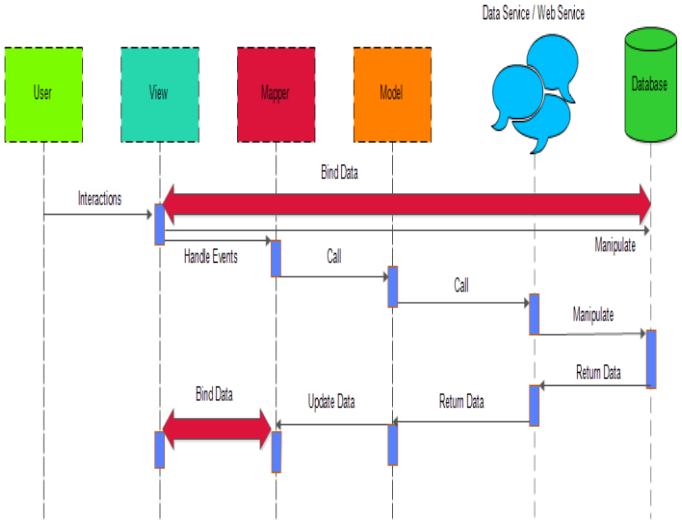


Figure 2.Sequence Diagram of MVM pattern

The sequence diagram in Figure 2shows the difference between the event responses of all patterns.

16. In order to predict the class label of X, P(X|Clsi)P(Clsi) is evaluated for each class Clsi . The classifier predicts that the class label of X is Clsi if and only if it is the class that maximizes P(X|Clsi)P(Clsi).

17: Iteration=Iteration+1

18: until Iteration=MITR

3. RESULTS AND DISCUSSION

3.1. Module 1 Process Flow:

- Upload MVC Software Version 1: MVC Software Version1 have 297 files
- Display Code Analyzer Page for Software version1
- Upload MVC Software Version 2 : MVC Software Version 2 have 299 files.
- Display code Analyzer Page
- Trace events for Software1
- Trace events for Software2

The overall process flow shown in figure 3 to figure 9.Upload both Software1 and Software2.Figure 3 and figure4 shows the Tracing events for both inputs.Figure5 shows the Behavioral dependency table for both software.Figure6 shows the filenames and their reusability percentage by using MVM Pattern. Figure 7 represents the chart which shows the percentage of reusability. Same process continues with proposed MVM pattern and the results are shown in figure 8 and figure9.

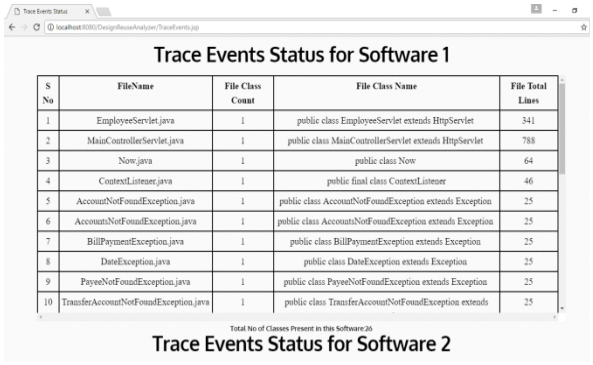


Figure 3.Trace events status for Software1

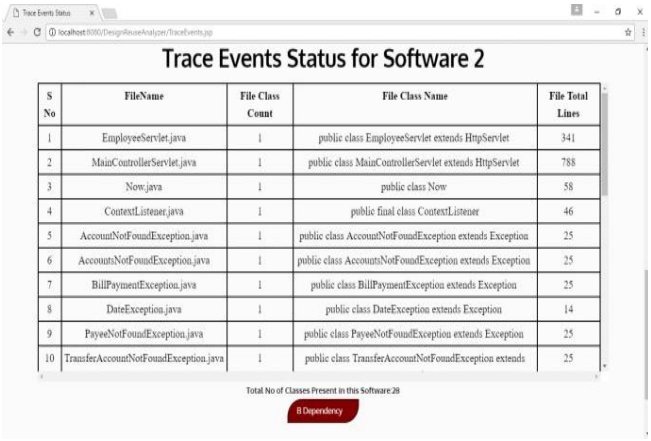


Figure 4. Trace events for Software2

- Trace the Behavioral Dependency for both software V1 and V2

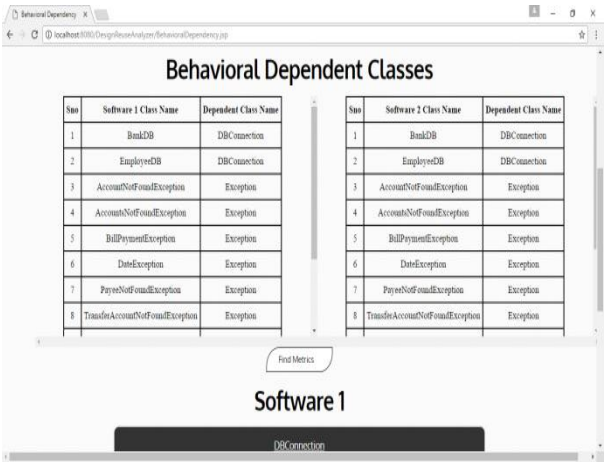


Figure 5. Behavioral Dependent Class

- Finding out the design reuse level.

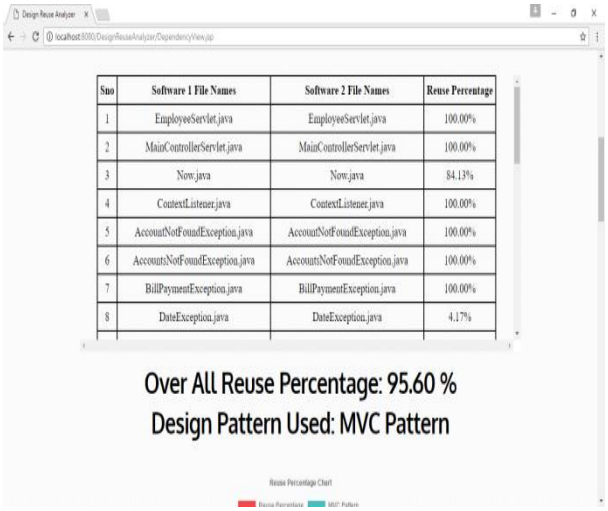


Figure 6. Overall Percentage using MVC Pattern

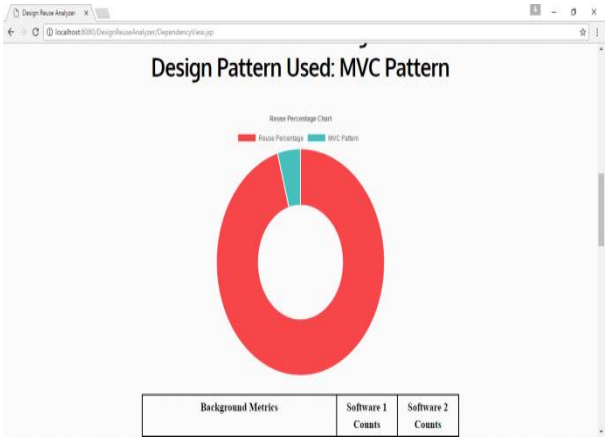


Figure 7. Reusability using MVC pattern

3.2. Module 2 Process Flow – Type 1:

- Upload MVC Software Version 1:
- MVC Software Version 1 are having 297 files.
- Display code analyzer for software V1: Total number of classes present in V1
- Upload MVM software
- Code analyzer algorithm works for MVM pattern for softwareV1:Total number of classes present in SoftwareV1 are 26
- Trace Events will extract both MVC Software Version 1 & MVM Software class names.
- Behavioral Dependency algorithm gets executed and it will extract all the MVC Software Version 1 &MVM Software Dependent Class Name & Depending Class Name in separate table.
- Finally displays the design reusability percentage.

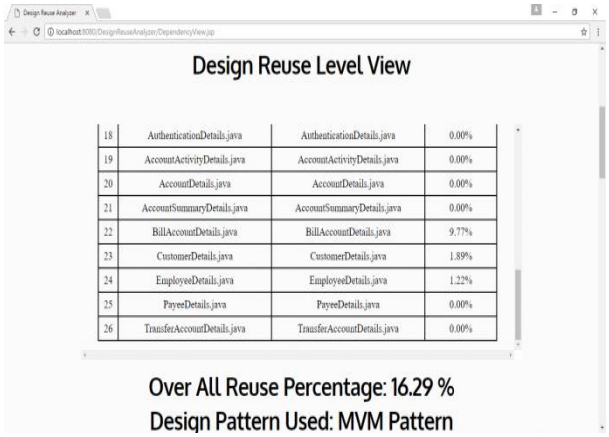


Figure 8. Overall reuse percentage using MVM pattern

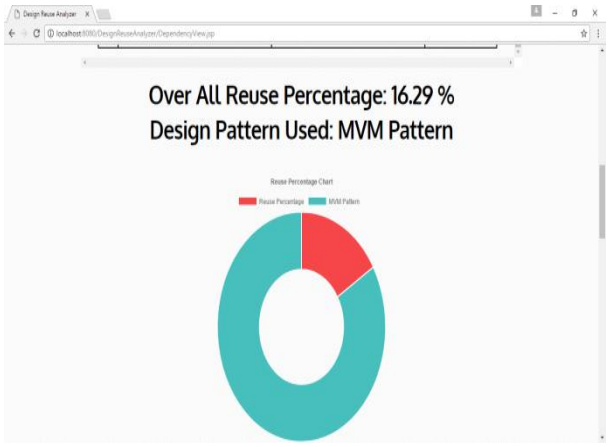


Figure 9. Reusability percentage in MVM pattern

In Background Hybrid ABCCM algorithm will works to track the Pattern Reuse Percentage for MVC Software Version 1 & MVM Software. Then extract some additional background metrics. Same process continues with different software modules.

4. CONCLUSION

In order for programmers to be able to reuse those design whose existence is not known to them, a design approach which help them in locating a pattern for reusage and converting them into components is proposed .The outcome of this research is to develop a framework for software pattern reusability at the code level .Methods by which the framework may be used to develop reusability will be proposed in the future research work.

5. REFERENCES

[1] “Review of Software Reusability” NehaBudhija and Satinder Pal Ahujain “International Conference On Computer Science And Information Technology (Iccsit'2011)” Pattaya Dec. 2011

[2] “Design Rationale And Design Patterns In Reusable Software Design” Feniosky Pena-Mora And Sanjeev Vadhavkar

[3] “Designing Code Level Reusable Software Components” ,B.Jalender 1 , DrA.Govardhan 2 , DrP.Premchand ,International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.

[4] Tawfig M. Abdelaziz, Yasmeen.N.Zada and Mohamed A. Hagal ,” A STRUCTURAL APPROACH TO IMPROVE SOFTWARE DESIGN REUSABILITY”

[5] “Design Patterns: Abstraction and Reuse of Object-Oriented Design” Erich Gamma , Richard Helm , Ralph Johnson, John Vlissides , Conference Proceedings, Springer-Verlag Lecture Notes in Computer Science.

[6] Reuse Design Principles ,ReghuAnguswamy and William B Frakes, International Workshop on Designing Reusable Components and Measuring Reusability Picture held in conjunction with the 13th International Conference on Software Reuse.

[7]”Software Reuse Research: Status and Future” William B. Frakes and KyoKang ,IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 7, JULY 2005

[8] Software Reuse: Research and Practice “Sajjan G. Shiva and LubnaAbouShala ,International Conference on Information Technology (ITNG'07).

[9] “Software Reuse” CHARLES W. KRUEGER ACM Computing Surveys, Vol. 24, No. 2, June 1992.

[10] “A Framework for Software Reuse and Research Challenges” Sathish Kumar Soora, International Journal of Advanced Research in Computer Science and Software Engineering.