International Journal of Engineering & Technology, 7 (3.34) (2018) 202-205

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Pattern Oriented Approach for Software Design Reusability for Cost Efficiency

[1]Chethana , [2]Dr.Srinivasan

[1] Lecturer in Computer Science NMKRV PU COLLEGE FOR WOMEN.
[2] Professor RV Engineering College Bangalore, India.

Abstract

The development of software increases their complexity. It has a major influence in business and industry. Therefore it is necessary to always need patterns which satisfy the needs of customers with an affordable cost. The main objective is to enhance the performance of enterprise design pattern for reuse. As a solution for all these requirements a new design called MVM pattern approach is been proposed.

***Keywords**: Software Reuse, Design pattern, MVM, MVC.*

1. Introduction

1.1 Software Engineering Definitions

Software engineering phases or stages incorporate managing, estimation, arranging, demonstrating, dissecting, indicating, planning, executing, testing and maintaining [Fenton97[1]]. Software associations have always been searching for effective ways to create software faster, less expensive and better. The term software crisis was first used in the year 1968 to portray the increasing burden and disappointment that software development and support put on generally glad and beneficial associations [Griss93[2]].Computer-Aided Software Engineering (CASE) uses various tools and formal strategies for automated testing, software development etc.. Following quite a while of software improvement, the software business has understood that there is no "silver bullets"; regardless of arguments of promoters of new innovations that there is. There are a few factors that limit the achievement of developments among these immature procedures, immature strategies and devices, unacceptable training, hierarchical resistance to change, immaturity of innovations and improper use of them, and the intrinsic trouble of creating software, particularly for big and complex software items.

2. Literature Review

In [3] expert designers Neha Budhijahave done an observedlearning of the software reprocessaction with the concept of object-oriented design. The learning concentrated under fundamentally three aspects of reclaim : (1) that is communication between variousblueprint forms, e.g. building an problem representation, searching for and assessing solutions, and reuse forms, i.e. recovering as well as utilizing earlier explanation, (2) intellectual procedures required in reclaim, instancebase recovery otherwise against top-down extending for solutions, in addition to (3) the intellectualportrayals developed all through the reuse action, e.g. activeagainststill illustration.

In Feniosky Pena-Mora [4] introduces an in-advance improvement of a framework for utilizing design rationale and design patterns for creating reusable programming frameworks. The proposed framework will be used as an incorporated design environment for reusable programming configuration, to help the collaborative advancement of programming applications by a gathering of programming authorities from a library of building block cases. These objectives convert into the effort of representing the use of Artificial Intelligence in better administration of software development and maintenance process by giving high speed, less expensive, smarter and on-time choices. 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 [5], the authors described about how the conventionpoint reusable mechanism can be built and how the conventionpointmethod can be designed. It also provides somecodestrategy, principles and finestpracticewornin support of creating reusable conventionpointmechanism and guidelines and best practices for making configurable and easy to use. Tawfig M [6] 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[7] 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. They have also discussed the experience in applying design patterns to the design of object-oriented systems.
William B [8] In their paper authors have briefly summarized about software reclaimexplores most important research assistance and uncertaintribulations in the proposed area, they provided pointers to key publications.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A component-based software designing is proposed in through Software knowledge reuse. Software analysis patterns can be used as a way to provide or facilitate Autonomous mobile robots (AMR).The software analysis patterns for AMR were obtained through a pattern mining process, and reported utilizing a standard index layout. These analysis patterns are classified based on the hybrid layered architecture of robot software: responsive layer, administrator layer, and deliberative layer. The analysis patterns in the reactive layer are featured and presented. The deployment of the analysis patterns are discussed by using an AMR software contextual analysis. The reuse capability of these patterns is assessed by measuring the reusability of components in the analysis pattern.Architectural patterns give structural documentations to any software framework. In authors characterized every segment with proper responsibilities and also provided a few principles, axioms, and relationship between them. Main information and functionality are encompassed in the framework. Model-View Controller structural pattern partitions the software framework into three subsystems, viz., model, view, and controller.

The proposed work in aims to fill the gap by contrasting MVC with other generally used Web development strategies in case of development time, viability, and the capacity to help correspondence among designers and software developers by differentiating a non-MVC Web application with a MVC-based Web application, and featuring the favorable circumstances and hindrances of each approach.

Model View Controller (MVC) is a standard design pattern used as a part of web composition or in web application improvement. It is significant because of its different capacities like extensibility, maintainability, and reusability. The authors in [53] generally isolated an online application into three level architectures. Other than web area MVC Architecture also uses as a part of embedded domain. MVC accomplishes this by isolating the application into three intelligent parts: Model: The model layer is in charge of the business rationale of an application. It will exemplify access to information stores and will give a reusable class library. View: The view layer is normally considered as website composition, or template. It controls the look and feel of information and gives facilities to gather information from the client. Advancements only found in the view are HTML, CSS, and JavaScript. Controller: The controller layer consolidates everything together and combines the styling of the view with the functionalities of the model.

The proposed work distinguishes the qualities of systematic software reuse methodologies and assesses how they add to an effective reuse program using survey data gathered from seventy-one software advancement gatherings. The outcomes demonstrated that these attributes group into five particular reuse methodologies and that every technique has an alternate potential for progress.

In the authors examines the reusability ideas for Component based Systems and explores a few existing measurements for both white-box and black box to quantify reusability specifically or in an indirect way and presents the exceptional prerequisites on software in this domain and Reusability is tied in with building a library of often used segments, along these lines enabling new projects to be assembled rapidly from existing segments. Component Based Systems (CBS) have now turned out to be more generalized approach for application advancement.

Work in software reuse concentrates on reusing artifacts. In this unique situation, finding a reusable artifact is driven by a desired functionality. A change to the basic view is proposed in..Authors argued that it is possible and important to also take a look at reuse from a non-functional (quality) point of view. Grouping thoughts from reuse, from objective situated prerequisites, from aspect oriented programming and quality administration; they have acquired an objective driven procedure to empower the quality-based reusability.

Griffin in a review of studies on new product development, found that various researches supported the suggestion that a formal, organized development process profited extend results, particularly when projects were unpredictable. A formalized reuse process should also guarantee the repeatability of the reuse accomplishment by upholding component standards. A required affirmation process can ensure that all aspects of the repository meets the desired execution principles.

2. 1 Algorithm 1: Code Classifier & Code Analyzer

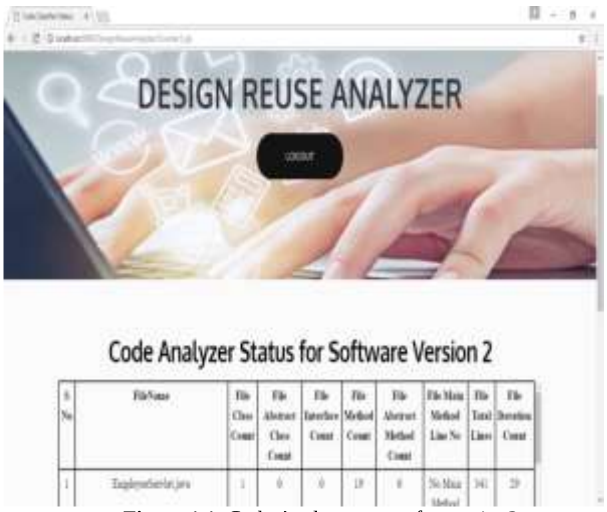
This is a much used tool of automation usedasmetrics for the process of software development among various source treeand is taken asa single "Code Set". The tool uses Singular record that allows the rejection of bits of a source tree.A list ofdefault file extensions explored in the process of filtering the source tree. The computation of the Metrics is done by using the file extension type as well as the same process applies for all of extensions .Simple text report files are created to archive the results. The Analyzer firstly traverses through the Code Set tree and does the parsing of each file and perform the validation checking and parsed file is used for the computation of the. Code Line contains code and comments.

```
begin
for each jpi to n do
begin // class counter
ccnt=1;
while(readline!=null) do
begin
ifreadline contains “class” then
cf.add(ccnt);
ccnt++;
end
end
begin // method counter
mcnt=1;
while(readline!=null) do
begin
ifreadlineendswith “)” then
mf.add(mcnt);
mcnt++;
end
end
begin // abstractclass counter
accnt=1;
while(readline!=null) do
begin
ifreadline contains “class” and readline contains “abstract” then
acf.add(accnt);
accnt++;
end
end
begin // method counter
amcnt=1;
while(readline!=null) do
begin
ifreadlineendswith “)” and readline contains

“abstract”then
amf.add(amcnt);
amcnt++;
end
end
begin // interface counter
icnt=1;
while(readline!=null) do
begin
ifreadline contains “interface”then
interf.add(icnt);
icnt++;
```

```
end
end
begin // main method counter
mmcnt=1;
while(readline!=null) do
begin
ifreadline contains “String args[]) or String [] args)” then
mmf.add(mmcnt);
mmcnt++;
end
end
end
```

where:
-jp is java program.
-cf is class found.
-ccnt is class count.
-mf is method found.
-mcnt is method count.
-acf is abstract class found.
-accnt is abstract class count.
-amf is abstract method found.
-amcnt is abstract method count.
-interf is interface found.
-icnt is interface count.
-mmf is main method found.
-mmcnt is main method count.
Code Analyzer:



Code Analyzer Status for Software Version 2

S No	FileName	File Class Count	File Abstract Class Count	File Interface Count	File Method Count	File Abstract Method Count	File Main Method Line No	File Total Lines	File Iteration Count
1	EmployeeServlet.java	1	0	0	19	0	No Main Method	341	29

Figure 1.1: Code Analyzer status for version2

Figure 1.2: Analyzer page with count of classes for Software v2

2.2 Naïve Bayes Classifier Model:

Naive Bayes classifier is a classification algorithm that is derived from the principles of Bayes theory.This algorithm is widely used both in the domains of data mining and machine-learning .In assumption that the n attributes of a random variable are conditionally independent, this model is used to predict the best class in case of a random variable . Bayesian learning is a machine-learning algorithm again derives some of its principle from the Naive Bayes classifier. The discovery of assumption is a very challenging issue in case of learning problems which consists of the most happening probability value in scenarios when there is value vector for arbitrary variable's properties vector exists. Such an assumption is calledMaximum a Posterior (MAP) [10.]

Algorithm 2: Hybrid ABC-CM➔Artificial Bee Colony + Naïve Bayes Classifier Model

So
1: Initialize the population of solutions Bee i, j, i = 1 ...EL, j = 1 ...P
2: Calculate the populations
3: iteration=1
4: **repeat**
5: Create novel solutions Val i, j for employed bees by using (2) and compute them
6: Employ the greedy selection method in Bees
7: Calculate the most probably values Pop i, j for the solution Bee i, j by (1)
8: Generate the novel solutions Val i, j for the bystanders from the solutions Bee i, j selected liable on Pop i, j and evaluate them
9: Apply the greedy selection method
10: Fix the uncontrolled solution for the scout, if exists, and substitute it with a novelarbitrarilyformed solution Bee i, j by (3)
11: Learn the **best solution** attained so far
12. Let **best solution** be a training set of samples, each with their class tags. There are n classes, Cls1, Cls2, . . . ,Clsn. Each sample is symbolised by an n-dimensional vector, X = {x1, x2, . . . ,xn}, depicting n measured values of the n attributes, Attb1, Attb2, . . . , Attbn, respectively.
13. Assumed a model X, the classifier will forecast that X goes to the training set taking the extreme a posteriori probability, conditioned on X. That is X is predicted to belong to the class Clsi if and only if
$P(Clsi |X) > P(Clsj |X)$ for $1 \leq j \leq m, j != i$.
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 identical for every classes, only P(X|Clsi)P(Clsi) need be maximized. If the class a priori probabilities, P(Clsi), are not identified, then it is usually 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). Notice that the class a priori probabilities may be valued by P(Clsi) = freq(Clsi , T)/|T|.
15. Known data sets with numerous attributes, it would be computationally costly to calculate P(X|Clsi). To decreasecalculation in evaluating P(X|Clsi) P(Clsi), the simple assumption of class restrictedfreedom is made. This assumes that the value of the attributes are temporarily free of one another, given the class value of the trial. Arithmetically this mean that
$P(X|Clsi) \approx n \, PI \, k=1 \, P(xk|Clsi)$.
The probabilities P(x1|Clsi), P(x2|Clsi), . . . , P(xn|Clsi) can effortlessly be predictable from the training set. Recall that here xk denotes 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 quantity of sample of class Clsi in T.
(b) If Attbk is continuous-valued, then we naturallyaccept that the training values have a Gaussian distribution with a mean μ and standard deviation σ defined by
$g(x, \mu, \sigma) = 1 \sqrt{} 2\pi\sigma \exp - (x - \mu) 2 \; / 2\sigma 2$,
so that
$p(xk|Clsi) = g(xk, \mu Clsi , \sigma Clsi)$.
We required to compute σClsi and μClsi, which are the standard deviationand mean of values of attribute Attbk for training samples of class Clsi .
16. To forecast the class value of X, P(X|Clsi)P(Clsi) is evaluated for every class Clsi. The classifier predicts that the class value 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
```

Where,

-Bee is each solution.
-EL is Employee bee or onlooker bee or scouts
-P is parameters up to n
-Val is new solution found in optimization
-Pop is probability of bee solutions
-MTR is Maximum Iteration Limit

3. Conclusion

Software reclaim is the method of developing new software by using the present software components. The major advantages of reusability are it increases the software productivity, save time, reduce maintenance cost and require less number of manpower and so on. 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 reusing and converting them into components is proposed .A new design reuse pattern has been developed and have done comparison on all the existing design patterns and proved that proposed MVM pattern is better compared to MVC architecture since MVM pattern can be used for small, medium and large scale industries. The outcome of this research is a design 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.

References

[1] Fenton, N., Pfleeger, S.L.: Software metrics: A Rigorous and Practical Approach. International Thomson Computer Press, 2nd edition, 1997.

[2] Griss93, M.L.: Software Reuse: From Library to Factory. IBM Systems Journal, November-December 1993, 32(4), pp. 548-566.

[3] W. Frakes and P. Gandel, "Representing Reusable Software, Information and Software Technology", vol. 32, no. 10, (1990), pp. 654-664.

[4] W. B. Frakes and B. A. Nejmeh, "An Information System for Software Reuse. In: Software,Reuse: Emerging Technology", IEEECS Press,(1990), pp. 142-151.

[5] D. Embley, et. al., "OO System Analysis: Is it or isn't It", IEEE Software, vol. 12, no. 7, (1995), pp.19-33.

[6] D. A. M. Lessandro, et al., "The Generic Reusable Component: An Approach to Reuse Hierarchical OO Designs. In: Proc Advance in Software Reuse", Los A lam itos,California: IEEE Computer Soeiety Press, (1993), pp. 39-46.

[7] William B. Frakes and Christopher J. Fox, "Quality Improvements Using A Software Reuse Failure Modes Model", IEEE Transactions on Software Engineering, vol.22, no.4, Apr. 1996.

[8] Basili, V., Briand, L. &Melo, W. (1996): "How Reuse Influences Productivity in Object-Oriented Systems". Communications of the ACM, Vol. 39, No. 10, pp. 105-116.

[9] Boehm, B. (1999), Managing Software Productivity and Reuse, *IEEE Computer* **16**(9), 111–113.

[10] Maqbool O., and Babri H.A, "Bayesian Learning for Software Architecture Recovery", International Conference on Electrical Enginnering(ICEE 07), 2007 ,1-6.