ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

An Efficient Methodology for Developing and Maintaining Consistent Software Using OOAD Tools

S. Pasupathy[1] , Dr. R. Bhavani [2]

Associate Professor, Dept. of CSE, FEAT, Annamalai University, Tamil Nadu, India[1].

Professor, Dept. of CSE, FEAT, Annamalai University, Tamil Nadu, India[2].

ABSTRACT: In this fast growing world, new technology has been emerged in order to reduce and improve our day-to-day activities. This can be achieved through software. By using the software, the programmer can develop a program to computerize the things. This software may be of various kinds based on the purpose and the kind of programming to be done. Of those kinds, the new emerging and most popularly used programming is Object Oriented Programming. This Object-Oriented Programming performs the task with much less time and with less number of codes. To develop such kind of program, OOAD (Object Oriented Analysis and Design) can be used.

In this paper, we have to use the OOAD technology to propose a methodology for developing a program with better efficiency and quality. Also, by developing a program using our methodology will provide a better maintenance cost.

KEYWORDS: Better Efficiency, Maintenance Cost, Number of Codes, OOAD, Object Oriented Programming, Quality.

I. INTRODUCTION

In this modern world, everything becomes computerized. To computerize the action, we have to develop a program to handle all the activities. To develop a program, it needs a programming language, which is of different categories. Based on the purpose, the programming language can be chosen. Of those categories, one of the most popularly used languages is Object-Oriented Programming language. This OOP language will simplify the task by complete the action with less number of codes. OOAD is one of the technique that related to OOP language.

Simplifying the software development process will contribute significantly towards successful software development. One of the fundamentals to a sound design in software development is the appropriate adoption and employment of Object Oriented technology that will contribute to a much more accurate outcome with better quality attributes, robustness and on-time delivery, all crucial factors for successful software development projects. This translates for the business better planning and utilization of resources, customer satisfaction, competitive edge and timely return on investment.

Design patterns brought a paradigm shift in the way object oriented systems are designed. Instead of relying on the knowledge of problem domain alone, design patterns allow past experience to be utilized to solve the problem. The Object Oriented Design (OOD) approaches advocated identification and specification of individual objects and classes. Design patterns on the other hand promote identification and specification of collaborations of objects and classes.
The object oriented method is a technique for

- ✓ System modeling

ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

- ✓ to manage complexity inherent in analysis, design and implementation.
- ✓ provide a methodology for system development.

The qualities of OO are as follows:

- ✓ Understanding of system is enhanced, as the semantic gap is reduced.
- ✓ Modifications to the model tend to be local as they often result from an individual item, which is represented by a single object.
- ✓ Ideally suited to model real systems, and simulating systems.

The Object-Oriented Modeling attach the behavior and information that is important to objects. It also associates relations between object to describe the static and dynamic organization and structure of real situation. The Object-Oriented Methods advocate integral objects which encapsulate both function and data. The major activities include identification of objects and analyzing their behavior and information.

Object-Oriented databases make the promise of reduced maintenance, code reusability, real world modeling and improved reliability and flexibility. The object-oriented approach does give the ability to reduce some of the major expenses associated with systems, such as maintenance and development of programming code. Some of the benefits of the Object-Oriented Approach are as follows:

- ✓ Reduced Maintenance
- ✓ Real World Modeling
- ✓ Improved Reliability and Flexibility
- ✓ High Code Reusability
- ✓ Easy to Understand

Object Oriented is a method to design and build large programs with a long lifetime. The benefits of using OO are as follows:

- Blueprints of systems before coding
- Development process
- Maintenance and modifications
- Control of dependencies
- Separation into components

Object-Orientation is closer to the way problems appear in life (physical and non-physical). These problems generally can't be formulated in a procedural manner. We think in terms of "objects" or concepts and relations between those concepts. Modeling is simplified with OO because we have objects and relations.

Now-a-days, most of the programmers use OOAD due to the need for tools to deals with the software complexity rises exponentially. OOAD provides these tools to meet the complexity.

OOAD provides tools for:

- ➢ A graphical modeling language – allows to describe systems in terms of classes, objects and their interactions before coding.
- ➢ A Programming language – Classes (data + functions) and data hiding; Interface separation (Class inheritance and member function overload)

An *Object-Oriented System (OOS)* is composed of objects. The behavior of the system is achieved through collaboration between these objects. The state of the system is the combined state of all the objects in it. Collaboration between objects involves them sending messages to each other.

Object-Oriented Analysis (OOA) aims to model the problem domain, the problem we want to solve by developing an object-oriented (OO) system.

ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

The Principles of the Object Oriented Design are as follows:

- ✓ Single Responsibility
- ✓ Stable Abstractions
- ✓ Stable Dependencies
- ✓ Open / Closed
- ✓ Acyclic Dependencies
- ✓ Common Reuse
- ✓ Common Closure
- ✓ Dependency Inversion
- ✓ Interface Segregation

Object-Oriented Analysis and Design (OOAD) is a popular technical approach to analyzing, designing an application, system, or business by applying the object-oriented paradigm and visual modeling throughout the development life cycles to foster better stakeholder communication and product quality. OOAD in modern software engineering is best conducted in an iterative and incremental way. Iteration by iteration, the outputs of OOAD activities, analysis models for OOA and design models for OOD respectively, will be refined and evolve continuously driven by key factors like risks and business values.

By analyzing all the benefits of OOAD, we use this technique to develop our methodology. Our proposed work is to develop a methodology using OOAD in order to develop better quality software in order to reduce the maintenance charge. This can be achieved by reducing the deviation between the design and the building phase.

II. RELATED WORK

In paper [1], Janaki Ram et al stated that the aim of recent research has mainly been restricted to identification of new design patterns from successfully implemented solutions of the past. In this paper, they proposed to utilize the concept of interactions among classes and objects as a key factor during problem analysis phase and as a natural way of designing object oriented systems. We look at different techniques that could be used to enable such an analysis and design process and identify open issues in those directions.

In paper [2, 3], Rajeev et al proposed that the approaches for OO Design proposed in the paper favour automatic techniques over manual ones for reasons described earlier. This means that we need a mechanism to be able to express design patterns in a format amenable to be read and interpreted by programs. Some attempts have been made at defining such pattern description languages.

In paper [4], Saad Alahmari et al discussed that Service-Oriented Architecture (SOA) was intended to improve software interoperability by exposing dynamic applications as services. To evaluate the design of services in service-based systems, quality measurements were essential to decide tradeoffs between SOA quality attributes. In the paper they introduced the structural attribute of service granularity for the analysis of other internal structural software attributes: complexity, cohesion and coupling. Consequently, metrics are proposed for measuring SOA internal attributes using syntax code.

In paper [5], Wayne et al observed that even arriving at good class definitions from the given problem definition is non-trivial. The key to various successful designs is the presence of abstract classes (such as an event handler) which are not modeled as entities in the physical world and hence do not appear in the problem description. In this paper, they stated that anticipating change has been proposed as the method for identifying such abstract classes in a problem domain. Another difficult task is related to assignment of responsibilities to entities identified from the problem description. Different responsibility assignments could lead to completely different designs [6]. Current approaches such as Coad and Yourdon [7], POT etc. follow the simple approach of using entity descriptions in the problem statement to define classes and fix responsibilities.

ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

In paper [8][9][7], Briand et al stated that Object-Oriented Metrics are useless if they are not mapped to software quality parameters. Many number of quality models are proposed to map parameters of the Object Oriented software like Extensibility, Reusability, efforts, manageability and cost. To know more about the internal structure of the product one should know more about the interdependencies of parameters of metrics and Software quality parameters.

In paper [10], Henderson described that Object oriented approach was capable of classifying the problem in terms of objects and provide many paybacks like reliability, reusability, decomposition of problem into easily understood object and aiding of future modifications.

In paper [11], Alok et al described that the Complexity metrics have been intensively studied in predicting fault-prone software modules. However, little work is done in studying how to effectively use the complexity metrics and the prediction models under realistic conditions. In this paper, they presented a study showing how to utilize the prediction models generated from existing projects to improve the fault detection on other projects. The binary logistic regression method is used in studying publicly available data of five commercial products. This study shows (1) models generated using more datasets can improve the prediction accuracy but not the recall rate; (2) lowering the cut-off value can improve the recall rate, but the number of false positives will be increased, which will result in higher maintenance effort. They further suggested that in order to improve model prediction efficiency, the selection of source datasets and the determination of cut-off values should be based on specific properties of a project. So far, there are no general rules that have been found and reported to follow.

In paper [12], Radica et al stated that most of the current project management software's are utilizing resources on developing areas in software projects. This is considerably essential in view of the meaningful impact towards time and cost-effective development. One of the major areas is the fault proneness prediction, which is used to find out the impact areas by using several approaches, techniques and applications. Software fault proneness application is an application based on computer aided approach to predict the probability that the software contains faults. The majority of software faults are present in small number of modules, therefore accurate prediction of fault-prone modules helps to improve software quality by focusing testing efforts on a subset of modules. This will discuss the detail design of software fault proneness application using the object oriented approach. Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. The primary goal is to develop and refine techniques for early prediction of fault-prone modules.

In this paper, the proposed method has to be developed by consolidating the related papers and also improve with more special features.

III. METHODOLOGY

3.1 Proposed Work

The aim of the proposed work is to develop a methodology to build and maintain the project in an efficient manner.

The summary of our proposed methodology is described below: In order to maintain the project efficiently, our proposed methodology is developed with 3 phases such as:

- Design Phase
- Build Phase
- Maintenance Phase

In 'Design Phase', the member in the design team creates a design to develop the project. This design is based on the development related activities. In order to carried out such activities, we can use the new emerging technology called OOAD(Object-Oriented Analysis and Design). This OOAD can be used to simplify the design phase. The design can be developed by using the tools and concepts available in OOAD. The concepts such as

ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

classes, methods, inheritance, attributes, overloading, overriding and so on may be included by the designer. All these concepts can be finalized in this design phase.
Based on the design phase, the project can be developed. All the requirements can be overdrawn in this design phase using the OOAD tools and concepts.

The next phase is the 'Build Phase', in which the design can be built using OOAD code. The developers in the build phase can develop the project using the design provided in the design phase. They can write the code in object-oriented programming language to develop the project from design. In this building process, there may be the chance for deviation in the design. This deviation can be analyzed by matching the count of concepts (classes, methods, attributes, and so on) between the design phase and the build phase. If the count differed, then the design cannot be fully build in the second phase. For example, if the number of count in the design phase be 20 and in the build phase be either 18 or 22, then it is clearly identified that there is a deviation between the two phase and also it is conclude that the design phase cannot be carried out efficiently.

The quality of the software can be measured as follows: If there is no deviation occurs, the software is identified as better quality. Otherwise, if there is a deviation, then the quality is measured by setting the upper limit for deviation such as 3%. That is, if the deviation is within the upper limit, it can be accepted and the build phase can be continued. Otherwise, the software can again be developed from the design phase.

Based on this methodology of calculating the deviation, the software can be maintained easily with moderate cost. When the deviation overflows the upper limit, then the maintenance becomes difficult. Thus by implementing our proposed methodology, the software can be easily designed, built and maintained easily and efficiently.
This proposed methodology also consists of an algorithm to develop a consistent programming for better maintenance. The algorithm is given below with proper explanation. Also, the experimental explanation is provided by carrying out experiments using the proposed algorithm.

3.2 Algorithm

Start
Get the customer requirements
Develop the design using OOAD tools and concepts, d
Build the design,b
Count the concepts in d and b, dcount and bcount
Compare dcount and bcount
If dcount && bcount < 3% then
Compile the program
Software moves to Maintenance Phase
else
Software moves to Design Phase, d
End If
End

3.3 Algorithm Explanation
The algorithm can be processed as follows: The initial step is to develop a design based on the customer requirements by using OOAD tools and concepts. The next step is to build the design that can be developed in the above step. This building can be carried out by writing the code in Object-Oriented language. Upon completing the build phase, the quality of the software can be evaluated. The evaluation can be carried out by counting the number of concepts in design phase and the build phase. Then the comparison can be carried out by matching the count. If the resultant is less than 3% deviation, then the software can be moved to maintenance phase after compilation. Otherwise, the software can be redesigned in the design phase. Thus the software can be developed with better quality with moderate maintenance charge.

ISSN(Online): 2319 - 8753
ISSN (Print) :2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

IV. EXPERIMENTAL SETUP

The proposed methodology is very efficient to maintain the software with moderate cost. To evaluate the performance of our methodology, various experimental setups can be carried out and the result is analyzed. The experiment is undertaken in a software development service center with a group of members of different phase. The members in the design phase develop a design to develop the project using OOAD tools and concepts. Then the design is developed by the members in the build phase. Upon building the software, the next step is to verify whether the software is deviated from the design. If so, then calculate the percentage of deviation to analysis the quality of software. If the software satisfies our proposed algorithm, then the developed software has better quality and then the software is allowed to move for maintenance.

Thus our proposed methodology provides better result to develop good quality software to maintain it with moderate cost.

V. CONCLUSION

The aim of the paper to develop best quality software in order to maintain the software with moderate cost is successfully implemented in our proposed algorithm. By using our proposed work, better software can be developed with less deviation between the design phase and the build phase. This can be achieved through the most popularly emerging technology, OOAD. OOAD helps us to develop the methodology efficiently by providing the tools and concepts. Thus proposed methodology provides an efficient result to the software developer.

REFERENCES

1. D. Janaki Ram, Arun Kumar, "Interaction Driven OOAD", Indian Institute of Technonology.
2. Object Management Group. UML 2.0 OCL Specification. http://www.omg.org/docs/ptc/03-10-14.pdf.
3. Rajeev R. Raje and Sivakumar Chinnasamy. eLePUS - a language for specification of software design patterns. In SAC '01: Proceedings of the 2001 ACM symposium on Applied computing, pages 600–604, 2001.
4. Saad Alahmari, Ed Zaluska, David C De Roure, School of Electronics and Computer Science University Southampton, UK, "A Metrics Framework for Evaluating SOA Service Granularity", International Conference on Service Computing, pp. 512-519, Jul 2011.
5. Wayne Haythorn. What is object-oriented design ? Journal of Object-Oriented Programming, 7(1), March-April 1994. pp. 67-78.
6. Leonor Barroca, Jon Hall, and Patrick Hall. Software Architectures. Springer-Verlag London Limited, 2000, pages 87-99.
7. Peter Coad and Edward Yourdon. Object Oriented Analysis. Prentice Hall PTR, 1990.
8. L.C.Briand, J.Wuest, J.Daly and Porter V., "Exploring the Relationships Between Design Measures and Software Quality In Object Oriented Systems", Journal of Systems and Software, 51, 2000.
9. L.C. Briand, W.L. Melo and J.Wust, " Assessing the Applicability of Fault Proneness Models Across Object Oriented Software Projects", IEEE transactions on Software Engineering. Vol. 28, No. 7, 2002.
10. B.Henderson-sellers, "Object-Oriented Metrics: Measures of Complexity" Prentice Hall, 1996.
11. Liguo Yu and Alok Mishra, "Experience in Predicting Fault-Prone Software Modules using Complexity Metrics", Quality Technology , & Quantitative Management, Vol. 9, No.4, pp. 421-433, 2012.
12. Ritika Sharma, NehaBudhija, Bhupinder Singh, "Study of Predicting Fault Prone Software Modules", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 2, Feb 2012.