INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATION & MANAGEMENT

IJRCM

A Monthly Double-Blind Peer Reviewed (Refereed/Juried) Open Access International e-Journal - Included in the International Serial Directories

Indexed & Listed at:

Ulrich's Periodicals Directory ©, ProQuest, U.S.A., EBSCO Publishing, U.S.A., Cabell's Directories of Publishing Opportunities, U.S.A., Google Scholar, Indian Citation Index (ICI), Open J-Gage, India [link of the same is duly available at Inflibnet of University Grants Commission (U.G.C.)], Index Copernicus Publishers Panel, Poland with IC Value of 5.09 (2012) & number of libraries all around the world.

Circulated all over the world & Google has verified that scholars of more than **5656 Cities** in **191 countries/territories** are visiting our journal on regular basis.

Ground Floor, Building No. 1041-C-1, Devi Bhawan Bazar, JAGADHRI – 135 003, Yamunanagar, Haryana, INDIA

http://ijrcm.org.in/

CONTENTS

CALL FOR MANUSCRIPTS

We invite unpublished novel, original, empirical and high quality research work pertaining to the recent developments & practices in the areas of Computer Science & Applications; Commerce; Business; Finance; Marketing; Human Resource Management; General Management; Banking; Economics; Tourism Administration & Management; Education; Law; Library & Information Science; Defence & Strategic Studies; Electronic Science; Corporate Governance; Industrial Relations; and emerging paradigms in allied subjects like Accounting; Accounting Information Systems; Accounting Theory & Practice; Auditing; Behavioral Accounting; Behavioral Economics; Corporate Finance; Cost Accounting; Econometrics; Economic Development; Economic History; Financial Institutions & Markets; Financial Services; Fiscal Policy; Government & Non Profit Accounting; Industrial Organization; International Economics & Trade; International Finance; Macro Economics; Micro Economics; Rural Economics; Co-operation; Demography: Development Planning; Development Studies; Applied Economics; Development Economics; Business Economics; Monetary Policy; Public Policy Economics; Real Estate; Regional Economics; Political Science; Continuing Education; Labour Welfare; Philosophy; Psychology; Sociology; Tax Accounting; Advertising & Promotion Management; Management Information Systems (MIS); Business Law; Public Responsibility & Ethics; Communication; Direct Marketing; E-Commerce; Global Business; Health Care Administration; Labour Relations & Human Resource Management; Marketing Research; Marketing Theory & Applications; Non-Profit Organizations; Office Administration/Management; Operations Research/Statistics; Organizational Behavior & Theory; Organizational Development; Production/Operations; International Relations; Human Rights & Duties; Public Administration; Population Studies; Purchasing/Materials Management; Retailing; Sales/Selling; Services; Small Business Entrepreneurship; Strategic Management Policy; Technology/Innovation; Tourism & Hospitality; Transportation Distribution; Algorithms; Artificial Intelligence; Compilers & Translation; Computer Aided Design (CAD); Computer Aided Manufacturing; Computer Graphics; Computer Organization & Architecture; Database Structures & Systems; Discrete Structures; Internet; Management Information Systems; Modeling & Simulation; Neural Systems/Neural Networks; Numerical Analysis/Scientific Computing; Object Oriented Programming; Operating Systems; Programming Languages; Robotics; Symbolic & Formal Logic; Web Design and emerging paradigms in allied subjects.

Anybody can submit the **soft copy** of unpublished novel; original; empirical and high quality **research work/manuscript** ***anytime*** in ***M.S. Word format*** after preparing the same as per our **GUIDELINES FOR SUBMISSION**; at our email address i.e. infoijrcm@gmail.com or online by clicking the link **online submission** as given on our website (***FOR ONLINE SUBMISSION, CLICK HERE***).

GUIDELINES FOR SUBMISSION OF MANUSCRIPT

1. **COVERING LETTER FOR SUBMISSION:**

DATED: _____

THE EDITOR

IJRCM

Subject: **SUBMISSION OF MANUSCRIPT IN THE AREA OF _____.**

(e.g. Finance/Mkt./HRM/General Mgt./Engineering/Economics/Computer/IT/ Education/Psychology/Law/Math/other, please specify)

DEAR SIR/MADAM

Please find my submission of manuscript titled '_____' for likely publication in one of your journals.

I hereby affirm that the contents of this manuscript are original. Furthermore, it has neither been published anywhere in any language fully or partly, nor it is under review for publication elsewhere.

I affirm that all the co-authors of this manuscript have seen the submitted version of the manuscript and have agreed to inclusion of their names as co-authors.

Also, if my/our manuscript is accepted, I agree to comply with the formalities as given on the website of the journal. The Journal has discretion to publish our contribution in any of its journals.

NAME OF CORRESPONDING AUTHOR	:
Designation/Post*	:
Institution/College/University with full address & Pin Code	:
Residential address with Pin Code	:
Mobile Number (s) with country ISD code	:
Is WhatsApp or Viber active on your above noted Mobile Number (Yes/No)	:
Landline Number (s) with country ISD code	:
E-mail Address	:
Alternate E-mail Address	:
Nationality	:

* i.e. Alumnus (Male Alumni), Alumna (Female Alumni), Student, Research Scholar (M. Phil), Research Scholar (Ph. D.), JRF, Research Assistant, Assistant Lecturer, Lecturer, Senior Lecturer, Junior Assistant Professor, Assistant Professor, Senior Assistant Professor, Co-ordinator, Reader, Associate Professor, Professor, Head, Vice-Principal, Dy. Director, Principal, Director, Dean, President, Vice Chancellor, Industry Designation **etc**. ***The qualification of author is not acceptable for the purpose***.

NOTES:

a) The whole manuscript has to be in ***ONE MS WORD FILE*** only, which will start from the covering letter, inside the manuscript. ***pdf. version is liable to be rejected without any consideration***.

b) The sender is required to mention the following in the **SUBJECT COLUMN of the mail**:

New Manuscript for Review in the area of (e.g. Finance/Marketing/HRM/General Mgt./Engineering/Economics/Computer/IT/ Education/Psychology/Law/Math/other, please specify)

c) There is no need to give any text in the body of the mail, except the cases where the author wishes to give any **specific message** w.r.t. to the manuscript.

d) The total size of the file containing the manuscript is expected to be below **1000 KB**.

e) Only the **Abstract will not be considered for review** and the author is required to submit the **complete manuscript** in the first instance.

f) ***The journal gives acknowledgement w.r.t. the receipt of every email within twenty-four hours*** and in case of non-receipt of acknowledgment from the journal, w.r.t. the submission of the manuscript, within two days of its submission, the corresponding author is required to demand for the same by sending a separate mail to the journal.

g) The author (s) name or details should not appear anywhere on the body of the manuscript, except on the covering letter and the cover page of the manuscript, in the manner as mentioned in the guidelines.

2. **MANUSCRIPT TITLE:** The title of the paper should be typed in **bold letters**, **centered** and **fully capitalised**.
3. **AUTHOR NAME (S) & AFFILIATIONS:** Author (s) **name**, **designation**, **affiliation** (s), **address**, **mobile/landline number** (s), and **email/alternate email address** should be given underneath the title.
4. **ACKNOWLEDGMENTS:** Acknowledgements can be given to reviewers, guides, funding institutions, etc., if any.
5. **ABSTRACT:** Abstract should be in **fully Italic printing**, ranging between **150** to **300 words**. The abstract must be informative and elucidating the background, aims, methods, results & conclusion in a **SINGLE PARA**. ***Abbreviations must be mentioned in full***.
6. **KEYWORDS:** Abstract must be followed by a list of keywords, subject to the maximum of **five**. These should be arranged in alphabetic order separated by commas and full stop at the end. All words of the keywords, including the first one should be in small letters, except special words e.g. name of the Countries, abbreviations etc.
7. **JEL CODE:** Provide the appropriate Journal of Economic Literature Classification System code (s). JEL codes are available at www.aea-web.org/econlit/jelCodes.php. However, mentioning of JEL Code is not mandatory.
8. **MANUSCRIPT:** Manuscript must be in ***BRITISH ENGLISH*** prepared on a standard A4 size ***PORTRAIT SETTING PAPER***. *It should be free from any errors i.e.* ***grammatical***, ***spelling*** *or* ***punctuation.*** *It must be thoroughly edited at your end*.
9. **HEADINGS:** All the headings must be bold-faced, aligned left and fully capitalised. Leave a blank line before each heading.
10. **SUB-HEADINGS:** All the sub-headings must be bold-faced, aligned left and fully capitalised.
11. **MAIN TEXT:**

THE MAIN TEXT SHOULD FOLLOW THE FOLLOWING SEQUENCE:

INTRODUCTION

REVIEW OF LITERATURE

NEED/IMPORTANCE OF THE STUDY

STATEMENT OF THE PROBLEM

OBJECTIVES

HYPOTHESIS (ES)

RESEARCH METHODOLOGY

RESULTS & DISCUSSION

FINDINGS

RECOMMENDATIONS/SUGGESTIONS

CONCLUSIONS

LIMITATIONS

SCOPE FOR FURTHER RESEARCH

REFERENCES

APPENDIX/ANNEXURE

The manuscript should preferably be in ***2000 to 5000 WORDS***, But the limits can vary depending on the nature of the manuscript.

12. **FIGURES & TABLES:** These should be simple, crystal **CLEAR**, **centered**, **separately numbered** & self-explained, and the **titles must be above the table/figure**. **Sources of data should be mentioned below the table/figure**. *It should be ensured that the tables/figures are referred to from the main text*.

13. **EQUATIONS/FORMULAE:** These should be consecutively numbered in parenthesis, left aligned with equation/formulae number placed at the right. The equation editor provided with standard versions of Microsoft Word may be utilised. If any other equation editor is utilised, author must confirm that these equations may be viewed and edited in versions of Microsoft Office that does not have the editor.

14. **ACRONYMS:** These should not be used in the abstract. The use of acronyms is elsewhere is acceptable. Acronyms should be defined on its first use in each section e.g. Reserve Bank of India (RBI). Acronyms should be redefined on first use in subsequent sections.

15. **REFERENCES:** The list of all references should be alphabetically arranged. ***The author (s) should mention only the actually utilised references in the preparation of manuscript*** and they may follow Harvard Style of Referencing. **Also check to ensure that everything that you are including in the reference section is duly cited in the paper**. The author (s) are supposed to follow the references as per the following:

- All works cited in the text (including sources for tables and figures) should be listed alphabetically.
- Use (**ed.**) for one editor, and (**ed.s)** for multiple editors.
- When listing two or more works by one author, use --- (20xx), such as after Kohl (1997), use --- (2001), etc., in chronologically ascending order.
- Indicate (opening and closing) page numbers for articles in journals and for chapters in books.
- The title of books and journals should be in italic printing. Double quotation marks are used for titles of journal articles, book chapters, dissertations, reports, working papers, unpublished material, etc.
- For titles in a language other than English, provide an English translation in parenthesis.
- ***Headers, footers, endnotes*** *and* ***footnotes*** *should* ***not be used*** *in the document.* However, **you can mention short notes to elucidate some specific point**, which may be placed in number orders before the references.

PLEASE USE THE FOLLOWING FOR STYLE AND PUNCTUATION IN REFERENCES:

BOOKS

- Bowersox, Donald J., Closs, David J., (1996), "Logistical Management." Tata McGraw, Hill, New Delhi.
- Hunker, H.L. and A.J. Wright (1963), "Factors of Industrial Location in Ohio" Ohio State University, Nigeria.

CONTRIBUTIONS TO BOOKS

- Sharma T., Kwatra, G. (2008) Effectiveness of Social Advertising: A Study of Selected Campaigns, Corporate Social Responsibility, Edited by David Crowther & Nicholas Capaldi, Ashgate Research Companion to Corporate Social Responsibility, Chapter 15, pp 287-303.

JOURNAL AND OTHER ARTICLES

- Schemenner, R.W., Huber, J.C. and Cook, R.L. (1987), "Geographic Differences and the Location of New Manufacturing Facilities," Journal of Urban Economics, Vol. 21, No. 1, pp. 83-104.

CONFERENCE PAPERS

- Garg, Sambhav (2011): "Business Ethics" Paper presented at the Annual International Conference for the All India Management Association, New Delhi, India, 19–23

UNPUBLISHED DISSERTATIONS

- Kumar S. (2011): "Customer Value: A Comparative Study of Rural and Urban Customers," Thesis, Kurukshetra University, Kurukshetra.

ONLINE RESOURCES

- Always indicate the date that the source was accessed, as online resources are frequently updated or removed.

WEBSITES

- Garg, Bhavet (2011): Towards a New Gas Policy, Political Weekly, Viewed on January 01, 2012 http://epw.in/user/viewabstract.jsp

RELATIONSHIP BETWEEN OBJECT ORIENTED DESIGN CONSTRUCTS AND DESIGN DEFECTS

PAWAN KUMAR CHAURASIA
ASST. PROFESSOR
DEPARTMENT OF INFORMATION TECHNOLOGY
BABASAHEB BHIMRAO AMBEDKAR UNIVERSITY
LUCKNOW

R A KHAN
PROFESSOR
DEPARTMENT OF INFORMATION TECHNOLOGY
BABASAHEB BHIMRAO AMBEDKAR UNIVERSITY
LUCKNOW

ABSTRACT

Software design starts from the initial development of software. During software development, various types of defects are made. Most of the design defects come from poor design, which affects the quality of the software. The basic principles of object-oriented design are inheritance, cohesion, coupling, abstraction and polymorphism are incorporated during software development. Defects may be occurred at the implementation level of the software, after the delivered of the software or during the maintenance of the software. If the defects are recognize and removed at the initial level of the software, then cost, time and effort are minimized of the software. During software development, OOD constructs are exploited and various types of design defects are taken place. In this paper, author implement the relationship between object oriented design constructs and design defects during software development at the initial phase of the software design.

KEYWORDS

OOD, rigidity, fragility, immobility, complexity.

INTRODUCTION

As the use of software industry grows in today's scenario, software quality is one of the important factors in software development. The defects must be recognized and corrected as early as possible, rather than delivery of the software. From various surveys, it is found that 90% of the total cost of a typical software project is consumed after delivery of the software [1]. High-quality software is typical to develop and maintain in large software projects, since they are costly to maintain because of bad design practices are the major cause of design defects, which make it difficult to add, debug and evolve all the features. Many studies reported that software maintenance is very costly, if modification made on a system after the delivery of the software. When adding new functionalities, bugs correction code modification, which improves the quality of the software as well as the cost optimization [2]. If these defects are identified at initial stage of the software, then the post maintenance cost and time consuming is minimized. Software quality is assessed and improved during reviews and testing. The primary objective is to detect defects and correct as early in the software process, before they are passed to another software engineering process and released to the customer. There has been a lot of researcher focus on bad designs also known as defects, anti-patterns [3], bad smells [2] or anomalies. These bad practices sometimes pay a huge loss generally, they are avoided by the development team at design level and removed at the code. Several approaches were proposed for maintenance and improve the quality of the software.

This paper is organized as follows: I section introduced about the paper. In section II, defines the defects and its types. In section III, literature work done by the researcher for OOD and design defects. In section IV, OOD design defects are identified and section V, defines the OOD constructs. In section VI, implement the relationship between OOD constructs and design defects. Last section is the conclusion of the paper.

DEFECTS

The responsibility of software developer is to deliver the quality products to the customer within schedule and cost. Software products meet-out all the requirement of the customer. These functions are the most important to the users to run the software without any defects. There are many factors of software quality, the first quality concerns with the defects. Defects may occur in all the process of life cycle of software development. The word *defect* sometimes refers to error, fault and failure. Defect can be referred either as fault (cause) or a failure (effect). Design defects are also called anomalies that refer to the design situation that affect the development of the models. Design defects are occurred early in the software process that increased the cost of later stages of software process. Some researchers have proposed processes and techniques to detect design defects. Beck [8], defines 22 sets of common defects. Design defects are common and repeating design defects that comes from design defects or code defects, which are defined as follows:

1. Design Defects

Design defects are also known as anomalies that directly affect the development models. Design defects and design patterns are same used in various organizations. Design defects used as wrong solution to repeat problems in case of object oriented design. Ghannem et. al. [4], focus on the detection of defects, which are appeared at the model level, especially in class diagram. Defect comes from various sources of objects either it comes from class, methods or attributes.

2. Code Defects

It is occurred at the code level. Fowler describes code smells as "certain structures in code that suggest the possibility of refactoring". It is occurred in the form of duplicate code, large class, long method, long parameter list and data class. It is also linked with the inner working of the class and design defects reflect the relationship between inner class and intra class.

LITERATURE REVIEW

Object oriented concept is the most common concept in today environment [6]. Object oriented system deals as the primary object in computation process. Object oriented principles, direct the designer what to hold up and what to prevent. A good object oriented design needs operations and functions that must be used in the development cycle. Defects are poor design selection, that decrease the quality of the object oriented design. It is similar to design patterns, which are used and studied in the industry and academic. Design patterns are proposed as "good" solutions N. Moha found that design defects are based not only on metrics but also on semantical and structural properties [7].Brown et. al. [5]. In 2011, G. Suryanarayan works on principle-based classification of structural based on structural design smells [8]. There is a lot of study on characterization and detection of object oriented design (OOD) defects [9]. Various empirical studies have been conducted to observe the impact of various OOD defects on code. These studies prove that OOD defects have a negative impact on maintainability [10], understandability [11] and fault proneness [12]. There is not any research work on relation between design constructs and design defects. Spoiled pattern can be used according two viewpoints; architectural and knowledge viewpoint [15]. A single flaw in design will manifest in next phase of software life cycle and it will become harder to diagnose the vulnerability as the software life cycle proceeds [16].

IDENTIFYING OBJECT ORIENTED DESIGN DEFECTS

Design defects are certain structures in the design that indicate the violation of fundamental design principles. Design defects have many root, some of them come from the limitation of the programming and other come from the work pressure or inexperience of the programmers. In paper [8], defects are catalog into four categories (Abstraction, Encapsulation, Modularization and hierarchy). There are various types of mistakes execute by the developers at the time of software design. The objective of these principles is to identify and highlight the most important defects keep in mind when design and developing defect free software. There are four basics symptoms that are rotting our designs. They are not relevant, but they are related to each other [13][14]. These design defects are defined below.

- **Rigidity (Design is hard to change):** It is the tendency for software to be hard to change in any ways. Every change causes a fall down of subsequent changes in dependent modules. Effect of changes is reflected in descendent module [13][14].
- **Fragility (Design is easy to break):** It is related to rigidity. It is the trend of software to be changed in various places. Sometimes breakages occurred in areas that have no relationship from various breakage points. Every time the developers are fear for the software failure from unexpected manner. Fragility is the opposite of robustness. Example: suppose we touch a drinking glass and that glass breaks, it is fragile. Even if we put our fingers only on a small part of the glass, the whole glass collapsed. A system that is fragile behaves similar. After a simple change has been made to the system, by adding new functionality or fixing a problem, the whole system breaks. Often, breakages occur in many places that have no conceptual relationship to the changed part [14].
- **Immobility (Design is hard to reuse):** It is the quality of the software; to be reuse from other projects or same project. Generally, it occurs that one engineer will discover that he needs a module similar to other module written by the other engineer. But he feels that there is too much change required to separate the desirable parts that he depends upon. It is hard to reuse, so the software is rewritten [13].
- **Viscosity (Hard to do the right thing):** It comes in two forms; design and environment. Developers usually find more than one mode of changes. Some of them continue with the design, some of them are not. When the design continued with the methods are harder to deploy. In case of environment development environment is slow and incompetent. Suppose a program compile time is too long, and developers invited to make changes to minimize the compile time, those changes which are optimal from design. If it is minimized from code by making changes in few files, then it is tempted to make changes, irrespective of design is preserved [13].
- **Needless Complexity (Overdesign):** It contains those elements, which are not presently useful. It happens when developers predict to change to the requirements and put facilities in the software to deal with those changes in future [13]. Good software design is lightweight, flexible, easy to read and understand and above all easy to change in future. Building infrastructure where not needed leads to more time spent in the development process and therefore higher costs. Additionally, the time to maintain the complicated parts of the system is increased, because it is hard to understand. Another term for Needless Complexity is Building for the future [14].
- **Needless repetition (Copy/Paste):** Sometimes copy and paste is useful, but they can be disastrous in code editing operations. Generally, it is happen that software systems are built in hundreds and thousands of repeated code elements [14]. When the same code appears again and again, in slightly different forms, the developers are missing an abstraction. Identify all the repetition and eliminating it with substitute or abstraction method, it is a long process towards make the system easier to understand and maintain [14].
- **Opacity (Disorganized expression):** These are the most common bad design principles. Some of them are applicable at design phase and some of them at coding level or both. There are various researcher addressed either one or two design principles of bad design to minimize the design defects. To minimize design defects at design level; identify subset and source of the above bad design defects [13].

From the above design principles author identified four design principles to address in the work. R. C Martin [13], said that rigidity, fragility, immobility and viscosity are the four symptoms are the cause of poor design. Generally software are developed either start from commence stage or prototype software is used by making some changes and reused it. When prototype software is used, some changes are required on various stages of software development. So, it is difficult to identify the effect of changes on software. In general, cases design defects are introduced at design time, upgrade or during maintenance of the software. During this time, most of the defects are occurred from rigidity, fragility, immobility and needless complexity of the design.

OBJECT ORIENTED DESIGN CONSTRUCTS

It facilitates to the designer to reusability, flexibility, maintainability, consistency etc. Object oriented plays an important role for providing various types of design factors to implement OOD principle concept. There are various object oriented design constructs, which are used:

- **INHERITANCE**

It is the mechanism that allows new class to be created by extending and refining its capabilities. The existing classes are called the base/parent/super classes and the new classes are called the derived/child/subclasses. Therefore, the new classes inherit the attributes and methods of the derived class provided that the base class allows. Also, the subclass may add its own attributes and methods and modify methods of any super-class methods. There are various types of inheritance, which are generally used:

- **Single Inheritance**: A sub class is derived from a single base class. In figure 1 Student class used as base class and the attributes and methods are inherited in Student_Scholarship defined in derived class.
- **Multiple Inheritance**: A sub class derived from more than one base class. In figure 2, Wheel and Rubber are two base class. Tyre works as a derived class, which inherit feature from two different base class, is called as multiple inheritance.
- **Multilevel Inheritance**: A sub class is derived from a base class and which is extended to derived from another base class and so on. In figure 3, animal class behaves as a super class and mammal sub class derived from super class. Mammal sub class behaves as a super class for human class.
- **Hierarchical inheritance**: It is a form of tree structure in which a base class has number of derived class, each of which may have consequent number of subclasses, with multiple levels. In figure 4, Vehicle super class has a number of sub classes and each of which has subsequent classes with number of levels to form a tree structure.

FIGURE 1: SINGLE INHERITANCE

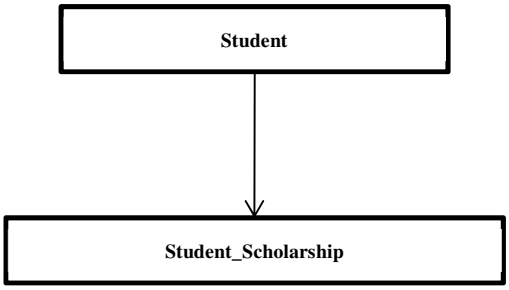


FIGURE 2: MULTIPLE INHERITANCE

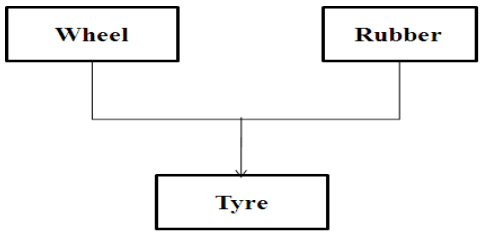


FIGURE 3: MULTI-LEVEL INHERITANCE

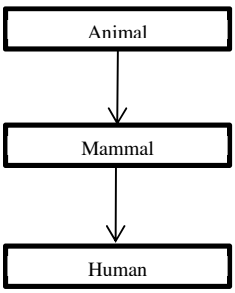
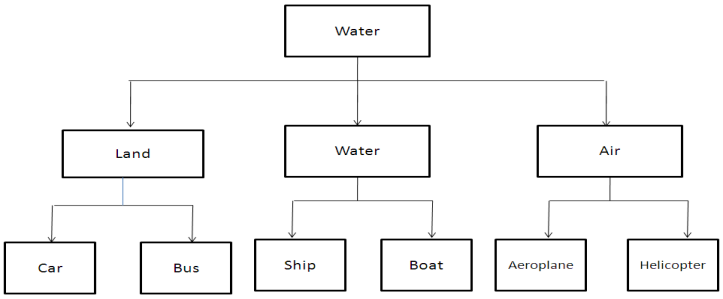


FIGURE 4: HIERARCHICAL INHERITANCE



- **COHESION AND COUPLING**

It refers to the internal stability of the design. It focused on data that is encapsulated within an object and how methods interact with data to provide limited behavior. A class is cohesive only when its methods are highly used. Highly cohesive class are typical to split. It helps to identify the poor design of the class. Cohesion is a measure of the functional strength of a module whereas the coupling between two modules is a measure of the degree of interdependence or interaction between the two modules. Suppose a module having high cohesion and low coupling is said to be functionally independent of other modules. A cohesive module performs a single task or function.

- **POLYMORPHISM**

It comes from Greek word; it has the ability to take multiple forms. In view of object-oriented concept, it uses operations in different ways, depending upon the instance they are operating upon. It also allows objects with different internal structures to form a common interface. It is generally effective during implementation of inheritance. Example: Two classes circle and square have a method genArea(), In both classes methods are same for internal implementation, but the calculation procedure is different of both classes.

- **ENCAPSULATION**

It is the process of binding both attributes and methods together within a class. Through this process, the internal details of a class are hidden for end user. The class has methods that provide services to user interface by which the services provided by the class may be used.

From the above object oriented design principles, researcher identified the three most relevant object oriented design concepts used for minimization of object oriented design defects. Inheritance cohesion and coupling are the basic OOD principles, which are used in, object oriented design.

RELATIONSHIP BETWEEN OOD CONSTRUCTS AND DESIGN DEFECTS

Design defects can be minimized by minimizing the no of defects (flow of defect in diagram). Research is focused on how design defects can be minimized using the object oriented constructs while using the identified design defects. In [20], how OO design mechanism such as inheritance, polymorphism, information hiding and coupling can influence quality characteristics like reliability or maintainability. The relationship between design constructs and design defects are shown in figure 5, which are described as follows:

OOD CONSTRUCT AND 'RIGIDITY'

OOD constructs are required to change in class/module and class/module consist of methods and attributes. During development, if there are any changes required, it may come from attributes and methods, which are known as defect attributes and defect methods. As the software, design suggests that software can be modified at any level without changes its behavior.

- If a design supports inheritance then all the methods and attributes are defined in parent class automatically available in its subclasses, as well as these sub classes generate hierarchy of subclasses. If there is any changes required in methods and attributes, these changes are required in descendent classes also. More influence of attributes and methods in depth make them more design defects. Therefore, with the increase of inheritance; 'rigidity' also increases [18].
- In case of cohesion, high cohesion suggested to improve the quality of an object-oriented design. High cohesion forms the sequence of modules where the output of one module behaves as input to other modules. Therefore, the input depends on the output of the other modules through attributes and methods. Hence, cohesion increases 'rigidity' of module decreases.
- In case of coupling, the degree of interdependence between modules increases and the parameters are exchanges between modules reflect the degree of modules. The defects of the design are reflected during interchange of the data between modules. Classes and subclasses have several limitations. Tight coupling introduces rigidity and makes further enhancements harder to implement which is undesirable in a software system of any size. If changes are made in the parent, it can affect the functionality of the children [17]. Therefore, degree of coupling increase, change the 'rigidity' of the design.

OOD CONSTRUCT AND 'FRAGILITY'

Fragility is the tendency of the software to break in many places with single change of modules. If there is, any changes occurred in one module and there is not any relationship with other modules, which change the behavior of the module. Hence, fragility bad design changes the behavior of the Object Oriented Design (OOD). Suppose a car system in changing the radio system affects the windows.

- In inheritance, methods and attributes are inherited in descendent classes and it is automatically accessible to the child class from the parent class due to its transitive nature. This accessibility is increased level by level and if any changes required in methods and attributes which affect the unknown class. Therefore, inheritance increases fragility also increases with the depth of the level [18].
- In case of cohesion, cohesion among defect attributes and methods cannot affect the connectionless classes. If there is any changes in one class, that can be changed in attributes and methods of other class which are not cohesively related is known as fragile defects in design. Hence, with increase of cohesion fragility is decreases.
- Coupling is directly related to sharing among software modules. More coupling increases the module more fragile. Fragile system is difficult to modify and cannot deal with a changing environment.

OOD CONSTRUCT AND 'IMMOBILITY'

It is the reverse of mobility. In an immobile software project, there are various parts of the system, which cannot be moved from one part to another part, because there are too many dependencies. Immobile systems are very bad for reuse purposes. Everything is coupled tightly together and no small updates can be applied to the system. There are only big modules with lots of dependencies.

- In case of inheritance, level of inheritance increases it is difficult to change in one method and attributes in base class. If there is any change in base class, then derived class and its descendent class are also effective. It is difficult to use the module from parent class to child class, which increased the immobility design [18].
- Strong class cohesion improved the quality of an OOD, when cohesion improves design defects are improved. In case of immobility, it is difficult to reuse the function. Therefore, cohesion increase immobility also decreased.

- It is found from various researchers that coupling enforces complexities in the code as well as in the design of the software. When coupling increase the complexity of the code and design also increase. When design complexity increases it is difficult to reuse the design and the number of interdependence modules are reduced. Hence, the immobility of the software decreases.

FIGURE 5: RELATIONSHIP BETWEEN DESIGN DEFECTS AND CONSTRUCTS

Design Constructs / Design Defects	INHERITANCE	COHESION	COUPLING
Rigidity	↑	↓	↑
Fragility	↑	↓	↑
Immobility	↑	↓	↑
Needless Complexity	↑	↓	↑

OOD CONSTRUCT AND 'NEEDLESS COMPLEXITY'

Unnecessary design increased the complexity of the software. Good software design is easy to reuse, easy to move, modify and usage of the design. In this section, analyzes the object oriented design constructs with respect to the principle of bad design defects of 'Needless Complexity'.

- In case of inheritance, unwanted level of inheritance increased the complexity of the design. Some future plan design increased the complexity of the design. So, when level of inheritance increased, complexity of design also increased [18].
- From various studies, it is found that high cohesion in classes reduces the complexity of the design. Hence, cohesion assist to building object oriented design simple and easy to use.
- Coupling is directly related to sharing data through software entities. More sharing of entities increase the complexity of the design. Hence, coupling between classes increase the complexity of the design.

CONCLUSION

As the software industry grows, size and complexity of the software also grows with time. Today most of the software is based on object-oriented concept. At the time of software development, various types of diagrams are implemented by using Class, Methods and Attributes. If these items are not properly deployed then the defects are occurred in the software. These defects may be occurred at the time of development, after deployment or maintenance of the software. During software design, OOD constructs are deployed during development of the software like (Inheritance, Cohesion, Coupling etc.). With the help of object oriented concept and various researcher views in figure 5 identified the relationship between design constructs and design defects and found that when inheritance increases all design defect are increased. But when cohesion increases, all the design defects are decreased. While in case of coupling, coupling increases all the design defects are also increased. Therefore, when inheritance and coupling are increased, design defects are increased. Therefore, a methodology is required to minimize the design defects of the software.

REFERENCES

1. W. J. Brown, R. C. Malveau, W. H. Brown, H. W. McCormick III, and T. J. Mowbray, "Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis", 1st ed. John Wiley and Sons, March 1998.
2. M. Fowler, "Refactoring – Improving the Design of Existing Code", 1st ed. Addison-Wesley, June 1999.
3. N. Fenton and S. L. P. Fleeger, "Software Metrics: A Rigorous and Practical Approach", 2nd ed. London, UK: International Thomson Computer Press, 1997.
4. A. Ghannem, G. El-Boussaidi, "Detecting Model Refactoring Opportunities Using Heuristic Search" In Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research (IBM Corp. Riverton, NJ, USA), p. 175-187.
5. J. W. Brown, C.M. Raphael Malveau, W. Hays, "Anti Patterns: Refactoring Software", Architectures, and Projects in Crisis, WILEY, 1998.
6. L. Zhao, "A New Approach for Software Testability Analysis", In: Proceeding of the 28th International Conference on Software Engineering, Shanghai, pp. 985–988 (2006).
7. N. Moha, "Detection and Correction of Design Defects in Object-Oriented Design", OOPSLA 22nd ACM SIGPLAN conference on Object Oriented Programming systems and applications companion, 2007, pp. 949-950.
8. S.G. Ganesh, T. Sharma, G. Suryanaryan, "Towards a Principle-based Classification of structural design smells", Journal of Object Technology, Vol. 12, no. 2, pp. 1–29, 2013. doi:10.5381/jot.2013.12.2.a1
9. G. Rasool, Z. Arshad, "A review of code smell mining techniques" Journal of Software: Evolution and Process 27, 11, pp. 867–895, 2015.
10. A. Yamashita, L. Moonen, "Exploring the impact of inter-smell relations on software maintainability: An empirical study" In 35th International Conference on Software Engineering, ICSE '13. 682–691, 2013.
11. M. Abbes, F. Khomh, Y. Guéhéneuc, G. Antoniol, "An empirical study of the impact of two Antipatterns, Blob and Spaghetti Code, on program comprehension" In Proceedings of the European Conference on Software Maintenance and Reengineering. 181–190.
12. F. Khomh, M.D. Penta, Y.G. Guéhéneuc, G. Antoniol, "An exploratory study of the impact of Antipatterns on class change- and fault-proneness", Empirical Software Engineering 17, 3 (2012), 243–275. DOI:
13. R. C Martin, "Design Principles and Design Patterns", Prentice Hall, 2006.
14. R. C Martin, M. Martin "Agile Principles and Practices in C#", Prentice Hall, 2006.
15. C. Bouhours, H. Leblanc, C. Percebois, "Bad smells in design and design Patterns", Journal of Object Technology, Vol. 8. No.3, 2009.
16. A. Agrawal, S Chandra, R. A. Khan, "An efficient of Object Oriented design Vulnerability" International Conference on Availability, Reliability and Security, pp. 618-623, 2009.
17. J. McGovern, S.W. Ambler, M.E Stevens, J Linn, V Sharan and E K Jo, "A Practical Guide to Enterprise Architecture" Prentice Hall, 2003.
18. A. Shaout, J.C Gracia, "Fuzzy Rule Base System for Software Classification", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 5, No 3, 2013.
19. H. Younessi, "Managing Software Defects in an Object-Oriented Environment", Defense Software Engineering, 3(1), 2003, pp. 13-16.
20. E Abreu, F. Brito, and Walcelio Melo. "Evaluating the impact of object-oriented design on software quality." Software Metrics Symposium, 1996., Proceedings of the 3rd International. IEEE, 1996.

REQUEST FOR FEEDBACK

Dear Readers

At the very outset, International Journal of Research in Computer Application & Management (IJRCM) acknowledges & appreciates your efforts in showing interest in our present issue under your kind perusal.

I would like to request you to supply your critical comments and suggestions about the material published in this issue, as well as on the journal as a whole, on our e-mail **infoijrcm@gmail.com** for further improvements in the interest of research.

If you have any queries, please feel free to contact us on our e-mail **infoijrcm@gmail.com**.

I am sure that your feedback and deliberations would make future issues better – a result of our joint effort.

Looking forward to an appropriate consideration.

With sincere regards

Thanking you profoundly

Academically yours

Sd/-
Co-ordinator

DISCLAIMER

The information and opinions presented in the Journal reflect the views of the authors and not of the Journal or its Editorial Board or the Publishers/Editors. Publication does not constitute endorsement by the journal. Neither the Journal nor its publishers/Editors/Editorial Board nor anyone else involved in creating, producing or delivering the journal or the materials contained therein, assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information provided in the journal, nor shall they be liable for any direct, indirect, incidental, special, consequential or punitive damages arising out of the use of information/material contained in the journal. The journal, neither its publishers/Editors/ Editorial Board, nor any other party involved in the preparation of material contained in the journal represents or warrants that the information contained herein is in every respect accurate or complete, and they are not responsible for any errors or omissions or for the results obtained from the use of such material. Readers are encouraged to confirm the information contained herein with other sources. The responsibility of the contents and the opinions expressed in this journal are exclusively of the author (s) concerned.

ABOUT THE JOURNAL

In this age of Commerce, Economics, Computer, I.T. & Management and cut throat competition, a group of intellectuals felt the need to have some platform, where young and budding managers and academicians could express their views and discuss the problems among their peers. This journal was conceived with this noble intention in view. This journal has been introduced to give an opportunity for expressing refined and innovative ideas in this field. It is our humble endeavour to provide a springboard to the upcoming specialists and give a chance to know about the latest in the sphere of research and knowledge. We have taken a small step and we hope that with the active co-operation of like-minded scholars, we shall be able to serve the society with our humble efforts.

Our Other Journals