A Quantitative Measure for Object Oriented Design Approach for Large-Scale Systems

Ms. Poornima U S
Assistant Professor
Department of Master of Computer Application
Acharya Institute of Management and Sciences
Peenya, Bangalore-58, INDIA
uspaims@gmail.com

Abstract— Object Oriented development methodology is a trend in software development for complex systems. The architecture of the application domain depends on the nature of problem statement in hand. Success depends on the overall design of a software in terms of flexibility and maintainability. Identifying and rectifying the design faults at early stage of development reduces cost and effort on the project by not exceeding the calendar. Quality metrics are helpful for the designer in measuring solution architecture for better product. Many such metrics are proposed in the literature and in practice, available as both commercial and open source tools. Objective of this paper is twofold: Understanding solution domain of Object Oriented system, measuring the design quality using metrics and future enhancements.

Keywords-Solution domain;Metrics;Models;Tools.

I. INTRODUCTION

From last few years, application of computer and information technology has a huge impact on different sectors of business, education and science. Developing such complex software is very challenging for the developers in terms of requirements and technology been used. Several factors like improper problem definition, difficulty in managing the development process may lead to complex software hard to maintain [Booch]. However the selection of development methodology and technology depends on nature of the problem statement. The customer need is either service-oriented or data-centric thereby influencing the system architecture. Two methodologies top-down (Structured) and bottom-up (Object-Oriented) are popular in the literature.

II. STRUCTURED V/S OBJECT-ORIENTED DESIGN METHODOLOGIES

A. Structured Design Methodology

Difference between these methodologies begins right from perceiving the problem statement. Traditional approach encourages the system analyst to gather the data as a set of services. The application domain posses a number of procedures interconnected through procedure-call or parameter passing. This kind of design methodology well suits for problem domain with average number of services. The design factors like coupling and cohesion has an impact on quality of the solution architecture. Cross coupling among functions reduces overall solution clarity. Weak cohesive functions make the software less scalable and maintainable. The concept of global data weakens whole architecture with respect to the integrity of software.

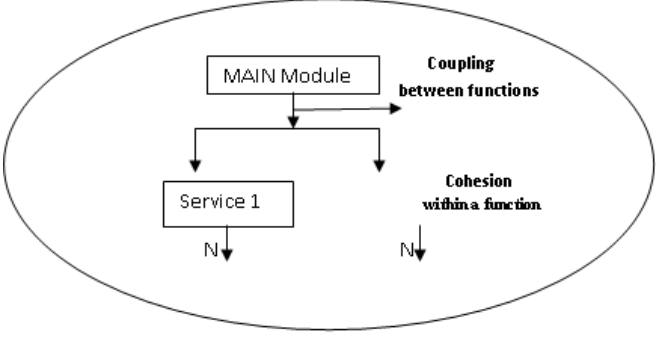


Fig.1. Solution domain of an of Top-Down approach

It is a front-end methodology that allows analyst to represent the system using Structured Chart for control flow and Data Flow diagram for movement of data within and among the modules at different levels.

B. Object Oriented Design Methodology

Object-Oriented development methodology is an emerging trend in software development for large-scale business and scientific applications. The solution is data-centric rather than services. The solution domain is first populated with related data and classified as classes based on similarity and relatedness existing among them. A set of modifiers and selector functions are defined according to customer requirements. Other special functions like constructors and destructors are placed for data initialization and memory management. However classes in isolation will not serve the purpose. The architecture is made complete by relating the classes each other either by class relationships or message passing (coupling).

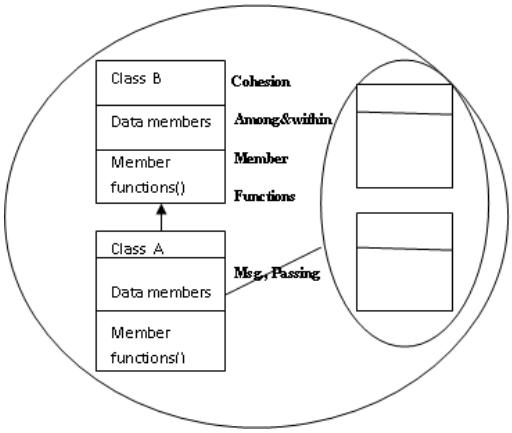


Fig.2. Solution domain of an Object-Oriented approach

III. OBJECT-ORIENTED DESIGN MODELS AND TOOLS

Models are graphical representative of forthcoming system. Process models like Waterfall, Iterative and Rapid Development explores the way a project is developed whereas the product and data models represent the design of the final product and database architecture. Models like Structured chart, Data Flow Diagram, Data Dictionary, Structured English, Decision Tree and Decision Table [Awad] provides comprehensive details on service-oriented systems. However, these traditional CASE tools are not suitable for Object Oriented approach since these tools are strongly associated with Waterfall model and fail to represent Object Model. OO CASE tools are widely used by IT industry for both software modeling and code visualization.UML (Unified Modeling Language) tool provides a set of diagrams considered to be more expressive and flexible. Class and Object diagrams represent static behavior wherein State transition and Interaction diagrams represent dynamic behavior of the system. Plenty of Open source and Commercial design tools are available.

TABLE.1. OBJECT-ORIENTED DESIGN TOOLS

TOOL NAME	***OPEN SOURCE***	***COMMERCIAL***	***PROGRAMMING LANGUAGE USED***
ArgoUML	√		Java
Magic Draw UML		√	Java
StarUML	√		Delphi
Rational Rose		√	Java
Eclipse UML2	√		Java
Violet	√		Java
Argo	√		C++

IV. METRICS AND TOOLS

Metrics are quantitative measure of a product before and after implementation. Software metrics are used to access the quality of a process from requirement analysis through design to implementation and the final product. Godman defines software metrics as, "The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products." [1]. Hence metrics are powerful tools to track the progress of ongoing projects for an organization. Errors in product design can be found out at early stage of development reducing cost and effort for software implementation. A well-designed and documented metrics are an asset of the organization helps to improve the quality of process, products and

services. Hence each organization has free hand to define their own metrics depending on type of products they develop/manufacture.

Two major categories of metrics are in practice: **Direct and Indirect metrics**. Process metrics checks the software development planning and scheduling so that process should not exceed the calendar. Product metrics are either direct or indirect measures of developed software. The direct measure checks LOC, Execution time, Memory usage, cost and effort on development, and number of defects. The indirect measures are on the software environment includes security, reliability, scalability, portability and maintainability.

A. Design Quality Metrics

In recent years, developing large-scale software in distributed and multi-user environment is becoming quite a necessity since users and requirements are growing drastically. The software is expected to be more user-friendly and reliable in user's perspective where as for developers a flexible and sound architecture so that system can be easily scalable and maintainable. Testing software at the end of the development only results in finding whether it is in line with user requirement or not. However checking the software quality at design level will make it more successful since the faulty design is found and corrected before implementation. Many design quality metrics have been proposed in the literature for both Traditional and Object-Oriented approach [2][3].

Traditional Metrics: Traditional metrics in practice are based on system architecture of Procedure-Oriented systems. Since software system is collection of procedures, metrics for size and complexity of modules were proposed. These metrics are applied in Object-Oriented methodology for class methods.

Source of Lines of Code (SLOC)/Lines Of Code (LOC)

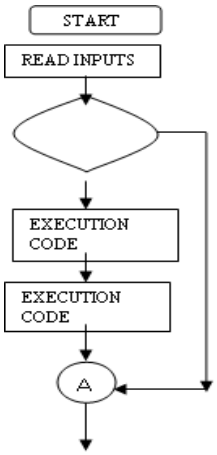
It is used to estimate total time and effort required to develop software. The SLOC metric measures number of physical lines of code excluding blanks and comments where as LOC counts all. It plays an important role since functionality can be written in different logics. A module with less line of code is more appreciated since cost and effort is saved. Thresholds of SLOC vary depending on logic and programming language used.

Comment Percentage:

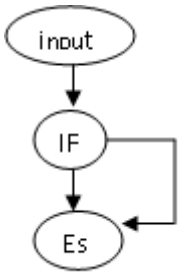
Understandability of software can be increased by providing comments either on-line of code or stand-alone. Number of comment lines divided by number of lines of code less than total blank lines gives comment percentage. It is developer's responsibility to place comments at relevant places to ease the maintenance process.

Cyclomatric Complexity:

Cyclomatic complexity is introduced by Thomas McCabe in 1976 to determine the risk and test scenario of developed software. It is a code based metric used to find number of linearly independent execution paths in a module, hence determines control flow complexity of an application using graph theory. Using this metric developer can test data structure boundary conditions and execution of each logical path at least once [Pressman]. Test cases are generated for these paths and software is tested. The software complexity is sum of complexity of all of its modules.



Flowchart



Graph

Predicate node+1

1+1=2

No.of Edges+No.of nodes+2

3-3+2=2

No. of regionss+1

1+1=2

Number of paths

B. Object- Oriented Design Quality Metrics

Object-Oriented development methodology is a recent methodology for multi-user, distributed and data-centric system. Class is a basic building block and vehicle for decomposition with operational attribute; methods and data attribute; data members. Success of the product is depending on design quality of the software. Beside traditional metrics, many design quality metrics are proposed in the literature among which CK and MOOD metrics are popular in the literature [4][5].

. *CK (Chidamdert&Kemerer) Metrics*

Object Model of the application domain has a class as basic building block and principles like Abstraction, Encapsulation, Inheritance and Polymorphism to make it complete.

Class: A basic unit of the solution framework with data and methods as attributes. The behavior of the system is represented by methods coupling the classes through message passing.

1. *Weighted Methods per Class (WMC)* measures the time and effort required for developing and maintaining a class operational attribute; methods. A class Complexity is a cumulative sum of complexity of all its methods. The objective is to keep it low to uphold design quality.
   WMC(C) $=\sum ci(Mi)$ $i=1….n$
   Where C is a class and M is a class method.
2. *Coupling Between Object classes (CBO)* measures the degree of interdependency between the classes. An object of a class can use the service or object of another class. The objective is to reduce cross coupling to increase the clarity of the solution

3. *Response For a Class (RFC)* measures response set of a class. When an object of a class sends a message, the methods executed inside and outside of a class are counted. The amount of effort in debugging, testing and maintenance is depending on response count.

$$|RS| = \{ M \} U \text{ all } i \{ Ri \}$$

   where { Ri } = set of methods called by method *i* and { M } = set of all methods in the class.

Inheritance: Size and complexity of the system is reduced by reusable components. Inheritance supports generalization and specialization concepts making the solution rich in terms of design.

4. *Depth of Inheritance Tree (DIT) is* a metric for measuring *vertical* growth of a class. Inheritance supports reusability; however the complexity is directly proportional to the distance between leaf and parent class. Deeper tree structure is prone to higher complexity as it is difficult to access end class behavior.
5. *Number Of Children (NOC)* measures the *horizontal* growth of a class. The immediate subclasses in a hierarchy show the greater reusability. System functional quality is highly dependable on abstractness of the parent class. Much effort is required in testing if tree grows in both directions.

Abstraction and Encapsulation: Identifying the properties for a problem has an impact on quality of the project. Cohesive methods makes the architecture more sound, flexible and maintainable.

6. *Lack of Cohesion in Methods (LCOM)* measures the quality of a class in a solution domain. Cohesion refers the degree of interconnectivity between attributes of a class. A class is cohesive if it cannot be further divided in to subclasses. It measures the method behavior and its relevance where it is defined. Pair of methods using data object proves the cohesiveness where as the methods not participating in data access makes it less cohesive. Consider C is a class and M1, M2,...Mn are its methods using set of class instances. I1={a,b,c,d}, I2={a,b,c} and I3={x,y,z} are set of instances used by the methods M1,M2 and M3 respectively. If intersection of object set is non-empty then the methods using them is cohesive and their relevance in the class is proved. i.e. I1 I2={{a,b,c} means M1 and M2 are cohesive. But intersection of I1 ,I3 and I2 , I3 is empty set. High count in LCOM shows less cohesiveness and class need to be divided to subclasses.

MOOD Metrics

CK metrics is applied for classes and methods individually. The MOOD metrics, defined by Fernando Brito e Abreu, measures overall quality of a software. Six original MOOD metrics are in practice; MHF, AHF, MIF, AIF, PF, CF.

1. *Method and Attribute hiding Factor (MHF, AHF)*

   The main focus of Object-Oriented concept is hiding (protecting) the class attributes from unauthorized access. Number of visible methods and attributes are counted and MHF, AHF is measured with respect to other classes in the same system.

   MHF=1-MethodsVisible
   AHF=1-AttributesVisible

   MethodsVisible = $\sum_i^n C(MV) / (C_i\text{-}1)$ / Number of methods

   MV is a count of number of all other classes with visible methods.

   AttributesVisible = $\sum_i^n C(AV) / (C_i\text{-}1)$ / Number of attributes

AV is a count of number of all other classes with visible attributes.
The value of MHF and AHF is high when class contains more of its attributes in provide section. However reusability is reduced when MHF and AHF are high.

2. *Method and Attribute Inheritance Factor (MIF, AIF)*
Inheritance principle of Object-Oriented Technique makes the software both scalable and reusable. MIF and AIF factors are high when derived classes use more properties of their parent class/s, hence increasing the dependency and maintenance risk.

MIF = Inherited methods/total methods available in classes
AIF = Inherited attributes/total attributes available in classes

The value depends on nature of the requirements set given by the customers, project analyzer and the designer to keep it average.

3. *Polymorphism Factor(PF)*
System architecture of Object-Oriented approach made enriched by using the concept of polymorphism. It is an ability given to the derived classes to override the definition (functionality) of parent class. Degree of overriding is known at the run time of a software depending on objects been passed as arguments. Hence the Overriding factor is ratio of static and dynamic binding, i.e, actual overrides/maximum number of possible overrides.

PF = overrides / $\sum_i^n$ C (new methods * descendants)

4. *Coupling Factor(CF)*
It is a measure of dependency between classes. Class A is coupled with class B only when A uses services or object of B thereby dependent on B, but not the vice versa.

CF=Actual couplings/Maximum possible couplings

More CF makes the architecture rigid and difficult to maintain in future.

C. Design Quality metric Tools

A number of commercial and open source metric tools are available to assess the quality of solution design and code generated. A few tools are available to check the quality of the UML diagram for the solution framework and many tests quality of code generated or assess the metrics proposed in the literature [6].

TABLE 2. TOOLS AND METRICS EVALUATION

TOOLS	Metrics					Formatted O/P
	WMC	DIT	NOC	CBO	LCOM	
JDepend		√	√			√
Classycle		√				√
Ckjm	√	√	√	√	√	
CCCC		√	√	√		
RSM	√	√	√			√
ES2	√	√	√	√		√

V. LITERATURE SURVEY

To improve the quality of large-scale software product different models, methodologies, metrics and tools been proposed in the literature.

Jamilah Din , Sufian Idris(2009) discussed the importance of Design Process Model for an Object Oriented systems. The paper presents OO Design methods and a Process Model for novice designers.

Dr. Waralak V. Siricharoen(2007) presents knowledge representation and Object Model. It also focuses on tools, procedures and methods for incorporation of Ontology with Object Model.

Dr. Rakesh Kumar and Gurvider Kaur (2011) have done a comparative study on the complexity of Object Oriented Design metrics proposed by Shyam R. Chidamber , Chris F. Kemerer and Li.

Ahmed M. Salem, Abrar A. Qureshi(2011) have discussed the inconsistency in proposed metrics in the literature with examples on complexity and cohesion metric of a project.

VI. CONCLUSION AND FUTURE SCOPE

Success of complex software with large problem domain and rich set of services highly depends on the methodology used and quality of the solution architecture. Object Oriented Methodology is suitable for data-centric and distributed systems for more flexibility and maintainability. Several metrics are proposed and are in practice for measuring the quality of a system much before implementation. Though new metrics are proposed as a variant of Chidamber and Kemerer metrics, there is scope in future to improve existing metrics based on the nature of the problem statement . There is a scope for Object Oriented CASE Tools to make it more designer friendly to suit the Object Model. Automatic code generation from UML tool supports rapid project development reducing time, effort and cost of the project.

ACKNOWLEDGMENT

Sincere thanks to Mr. Nandakumar V Purohit, a Technical Team Leader for continuous guidance on the Tools and Techniques for software development and Ms. Bhagyalakshmi (Phd) , Mr. Mallikarjuna Shastri P M (Phd.) for the suggestions on improving the content of the paper.

REFERENCES

[1] Linda Westfall, The Westfall Team, "12 Steps to Useful Software Metrics", The Westfall Team, 2005
[2] Cem Kaner, Senior Member, IEEE, and Walter P. Bond, "Software Engineering Metrics: What Do They Measur and Hoe Do We Know", 10th INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS 2004.
[3] Dr. Linda H. Rosenberg, "Applying and Interpreting Object Oriented Metrics", Track 7 - Measures/Metrics.
[4] Shyam R. Chidamber and Chris F. Kemerer "A Metrics Suite for Object Oriented Design", IEEE TRANSACTION ON SOFTWARE ENGINEERING, VOL 20,No 6, JUNE 1994.
[5] Seyyed Mohsen Jamali, "Object Oriented Metrics –A Survey Approach", January 2006.
[6] Rüdiger Lincke, Jonas Lundberg and Welf Löwe, "Comparing Software Metric Tools",2008 ACM 978-1-59593-904-3/08/07.
[7] Jamilah Din , Sufian Idris, "Object-Oriented Design Process Model", *IJCSNS International Journal of Computer Science and Network Security*, VOL.9 No.10, pg.no.75-79, October 2009 .
[8] Dr. Waralak V. Siricharoen , "Ontologies and Object Models in Object Oriented Software Engineering", *IAENG International Journal of Computer Science*, 33:1, IJCS_33_1_4.
[9] Dr. Rakesh Kumar and Gurvider Kaur , "Comparing Complexity in Accordance with Object Oriented metrics", *International Journal of Computer Applications*(0975-8887),Volume 15-No.8, February 2011 .
[10] Ahmed M. Salem, Abrar A. Qureshi, "Analysis of Inconsistencies in Object Oriented Metrics", *Journal of Software Engineering and Applications*, February 2011, 4, 123-128.

AUTHORS PROFILE

Ms. Poornima U S has done B.E. MTech in Computer Science and Engineering, presently working as Assistant Professor in the Department of Master of Computer Application, Acharya Institute of Management and Sciences, Peenya, Bangalore. She has got 13 years of teaching experience and published research papers in International Journals. She is guiding post graduate students on various live projects .