IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 13, Issue 5 (Jul. - Aug. 2013), PP 43-47
www.iosrjournals.org

Relational Analysis of Software Developer's Quality Assures

A. Ravi[1], Dr. Nirmala[2]
[1]*(Research scholar /M.S University,India)*
[2]*(Associate professor Department of computer science/university of madras/India)*

Abstract: *Software engineering approach ensures the quality of software and the delivery of the product on to time to their client. The development process is facing many challenges to optimize the resource and provide the delivery on time. Each phase of the development has its unique functionality. Software development model and the architectural approaches are differ that will creates an impact in the quality of the delivery .however the developers role and their skill sets are vital role in the developmental process and to meet the deadlines of the development tasks. The developer's skills and the developmental tasks inline to the functional domain and its relational process are determine the quality factors such minimum error, functional correctness and user satisfactory. This research attempted observes the functional process of the development sector and determine the influencing factor and its associative relationship. The data from the industry and its descriptive analysis with the determined factors are discussed as part of the paper.*

I. Introduction

The success factor of any software development system highly depends upon the architecture of the software development model, technical reliability used in it, pre-planning and preparatory works done and the effective implementation of the specified works in a time driven manner along with the developer's completion. In the same way Software architecture forms the backbone for building successful software-intensive systems.

Architecture largely permits or precludes a system's quality attributes such as performance or reliability. In general, a critical issue in the design and construction of any complex software system is based on its architecture implementation by the developers. The right architecture is the key to software project success. Conversely, the wrong one is a sure road to failure. A good architecture can help ensure that a system will satisfy key requirements in such areas as performance, reliability, portability, scalability, and interoperability.

This paper attempted to relate the software architecture and its development model along with the developer's role. The developer efficiency and related factors are addressed to develop the model for effective software development process and its metrics.

Software development process is a sequence of activities. These activities are dependent with one another in each level. These activities are related in the next phase. It is a chain of activities involved to determine the resource allocation, utilization and coating process. These relationship activities are used to determine the software process reusability and maintenance. This dependent and independent activities are varies as per the architecture in the software model of development phases. This research process tinted to determine the dependent and regression process values to identify the model and architecture. These create an effect in the cost, development, resource identification and optimization etc.

II. Software Engineering Measures

In software engineering, there are three kinds of entities and attributes to measure [1]:

a. Processes are collection of software-related activities. A process is usually associated with some timescale. The timing can be explicit, as when an activity must be completed by a specific date, or implicit, as when one activity must be completed before another can begin.
b. Products are any artifacts, deliverables or documents that result from a process activity. Products are not restricted to the items that management is committed to deliver to the customer. Any artifact or document produced during the software life cycle can be measured.
c. Resources are entities required by a process activity. The resources that to measure include any input for software production. Thus, personnel (individual or teams), materials (including office supplies), tools (both software and hardware) and methods are candidates for measurement. Within each class of entity difference between internal and external attributes:

Internal attributes of a product, process or resource are those that can be measured purely in terms of the product, process, or resource itself. In other words, an internal attribute can be measured by examining the product, process or resource on its own. External attributes of a product, process or resource are those that can be measured only with respect to how the product process or resource, relates to its environment. Here, the behaviour of the process, product or resource is important, rather than the entity itself.

For instance, consider a set of software modules, without executing the code it can determine several internal attributes: size, complexity, and modularity, etc. However there are other attributes of the code that can be measured only when the code is executed: reliability, usability, maintainability, etc. Managers often want to be able to measure and predict external attribute. However, external attributes are usually more difficult to measure than internal ones, and they can be measured only in the late stages of the development process. Thus, there is a clear need for internal attribute measurements to assess external attributes. One of the goals of software measurement research is to identify the relationships among internal and external attributes. This research work focused to measure the internal attribute of the process.

Software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving software quality. Previous research has indicated that they provide useful information on external quality aspects of software such as its maintainability, reusability and reliability[3].

Software metrics provide a mean of estimating the efforts needed for testing. Software metrics are often categorized into products and process metrics [9].

1.1 Process Metrics:

Process metrics are known as management metrics and used to measure the properties of the process which is used to obtain the software. Process metrics include the cost metrics, efforts metrics, and advancement metrics and reuse metrics. Process metrics help in predicting the size of final system & determining whether a project on running according to the schedule.

1.2 Products Metrics

Product metrics are also known as quality metrics and is used to measure the properties of the software. Product metrics includes product non reliability metrics, functionality metrics, performance metrics, usability metrics, cost metrics, size metrics, complexity metrics and style metrics. Products metrics help in improving the quality of different system component & comparisons between existing systems.

The process measures are observed in three ways in the development environment for the integrated development of the system. The project process and the performance are collected at individual, project and organizational level. The individual who collected the data about his own work knows it is his data, although it may be pooled with data from other individuals to provide an overall project profile. Project Team Data is private to the members of the project team, although it may be pooled with data from other projects to provide an overall organizational profile. Organization Data may be shared throughout the organization. As an example, the work effort distribution data, the number of hours each individual spends working on every development or maintenance activity in a week is private to that individual. The total distribution of hours from all team members is private to the project team, and the distribution across all projects is public to everyone in the organization. View and present the data items that are private to individuals only in the aggregate or as averages over the group.

Several companies have implemented metrics programs to support the managers in their decisions. However the benefits from the implementation are not as great as expected. Nearly 80% of software metrics programs fail within the first two years [2] This research will further the understanding of how data (in particular metrics) can be used in managing and improving software development processes. This will be done by studying the collection, interpretation, distribution, and the use of data as part of managing software development processes. Decision-making is a very complex process and it is naive to think that it can be based solely on structured data as metrics. However, metrics are useful for decision making of software managers, i.e. metrics are part of the management information system for software managers. Therefore this work is aimed to determine the functional process and the correlation of the development and their functional point usage.

III. Software metrics- Process Metrics

Process Metrics presents the software metrics appropriate to use during the implementation phase of the software development. The primary purpose of development is to flesh out the architecture and the system as a whole. Implementation is the focus during the development process.. The metrics presented in this section are: Defect Metrics and Lines of Code (LOC). During the1998 IFPUG conference, Capers Jones gave a rule of the thumb to get an estimation of the number of defects based on the Function Points of the system [11].

The Lines of Code (LOC) metric specifies the number of lines that the code has [2]. The comments and blank lines are ignored during this measurement. The LOC metric is often presented on thousands of lines of code (KLOC) or source lines of code (SLOC) [3] [4]. LOC is often used during the testing and maintenance phases, not only to specify the size of the software product but also it is used in conjunction with other metrics to analyze other aspects of its quality and cost.

Several LOC tools are enhanced to recognize the number of lines of code that have been modified or deleted from one version to another. Usually, modified lines of code are taken into account to verify software quality, comparing the number of defects found to the modified lines of code[10]. Other LOC tools are also used to recognize the lines of code generated by software tools. Often these lines of code are not taken into account in final count from the quality point of view since they tend to overflow the number. However, those lines of code are taken into account from the developer's performance measurement point of view [6].This paper also addressed the relational factors and the impact , cost effectiveness in the developmental process.

A software development process is a structure imposed on the development of a software product.There is several models for such processes. Each will describe the approaches to a variety of tasks or activities that take place during the process. Here let us discuss about the selection of models, phases of the model, weightage assumptions for each phases and activity specifications of each phases.

3.1 Selection of model:

Software process model is an abstract representation of a software process. Within the several models of software development process let us select a model M which is used to perform the different activities to get a desired product. If choose the model M then the variable representing that model is PxMAy. Here Px is the different phases of the model M and Ay is the different activities of M model in the phase Px. For example if select the waterfall model then that can be represented by the variable PxWAy.

3.2 Phases of Model:

All the software models have different phases and each phase will have specific activities. the Requirement Analysis phase (P_1) of the waterfall model then the activities in that models can be denoted as

a_1 -> Requirement gathering – P_1Wa_1
a_2-> Requirement analysis - P_1Wa_2
a_3-> Requirement specification – P_1Wa_3

Likewise it can determine the variables for all the activities of different phases of the waterfall model.

3.3 Weightage assumptions:

In the field of software development, the cost of the development is the cost incurred during Requirement analysis, development, coding & testing etc. So determine the effort distribution for different phases of the software development. Let us have the effort distribution for different phases as Requirement 10%, design 20%, coding 20%, Implementation & testing 50%.This is called the weightage assumption for different phases of a software model according to the activity carried out by the developers. According to the developers contribution, the effort distribution for waterfall model as follows:

Phases	No.of activities	Weightage
I-Feasibility study	3	10%
II–Requirement Analysis	3	10%
III – Design	4	20%
IV – coding & testing	2	30%
V – Implementation	4	20%
VI – Maintenance	3	10%

3.4Activity specifications:

All the activities of each phase to be defined as a set according to developers involvement and the cost, Px A where x denotes the phases of the model and A denotes the activities of the phases. Each phase of a model carries specific activities to the finished. If a phase consists of set of activities, then the activities of P_1A are determined by the phase as per the model.

$P_1A = \{P_1a_1, P_1a_2, P_1a_3....P_1a_n\}$

Like wise the second phase activities denoted by P2A which contains the activities specified in the second phase of the software development model.

$P_2A = \{P_2a_1, P_2a_2, P_2a_3...P_2a_n\}$

IV. Construction Activity Association Matrix

4.1 Activity Relation:

Taking one activity in phase x compare the impact of that particular activity in the next phase (x+1). During the comparison if the activities, A creates an impact in the next phase that can be called as Activity relations. These types of activities also called on dependent activity. If that particular activity, A did not create any impact in the next phase then it is called as independent activities.

Let us assume that phase value of x is 1. The first phase of the water fall model is feasibility study. The second x+1 phase of the water falls model is requirement analysis.
The feasibility study activities and its corresponding code's represented as follows:
P_1a_1=>clients study
P_1a_2=>process study
P_1a_3=>best process solution.

The activities of the requirements phase and its codes represented as
P_2a_1=>requirements gathering
P_2a_2=>requirements analysis
P_2a_3=>software requirements

Level I (Feasibility analysis set(P_1A) vs Requirement analysis set (P_2A))

4.2 Independent activity association matrix

The F**easibility analysis set (P_1A)** considered as column and **Requirement analysis set (P_2A)** considered as row then a two dimensional association matrix framed with the following condition.

Condition 1:

The activities of F**easibility analysis set (P_1A)** are created an impact with activities of **Requirement analysis set (P_2A)** then the value is set as '1'. These activities are considered as related activities/dependant activities.

Condition 2:

The activities of F**easibility analysis set (P_1A)** is not create on impact on the activities of **Requirement analysis set (P_2A)** then the value is set as '0'. These activities are considered as isolated activities/independent activities

Condition 3:

If the activities of F**easibility analysis set (P_1A)** partially created the impact on **Requirement analysis set (P_2A)** then those activities to be called as partial dependant activities. If its impact is dominant on the process then the value to be considered as approximate equivalent to the dependant activity else it is treated as an isolated activities.

In certain cases these partial activities are treated as X don't care condition.

Feasibility study(P1wA)/required analysis (P2wA)	Required gathering (P_2wa_1)	Required analysis (P_2wa_2)	s/w requirements (P_2wa_3)
Client study(P_1wa_1)	1	1	0
Process study(P_1wa_2)	0	X	1
Find best process(P_1wa_3)	0	1	1

V. Relationship weight calculation

5.1 Construction of relationship between

Phases Px => Py

Activities of (p1)+activities(p2) * weight % of p1 + weight % of p2
--- --------------------- Total of all phase * total % of weight

```
3+3 *10+10=>  6 * 20
----- ---------- ---  -----
 19    100      19  100
```

=> 0.3158 * 0.2 = 0.0632
This is the total relation value of p1 & p2
Possible relation value = 3*3=9
Per activity relation between p1 & p2=
total relation value = 0.0632
Possible relation value 9
= 0.0632/9 = 0.0070

5.2 Occurrence relationship of PxA=>PyA where A is an existence

As per the association map table & water fall model, 5 relationships occur between p1 & p2

Per activity relation value * 5

= 0.0070 * 5

= 0.035

The feasibility study phase possibility to affect in the requirement analysis phase as follows

$$= \frac{0.035}{0.01579 + 0.01579} * \frac{20}{100}$$

=1.1083 * 0.2

Originally affected =0.2217

Dependency level of p1 + p2 = 0.2217

Independency level of p1 + p2= 1-0.2217

= 0.7783

According to the increase of the intendancy of the module the error level may reduced. But the functional relationship alone can't be finalizing the quality the developer's skill on the module also to be consider.

VI. Conclusion

This paper addresses the software development architecture model and the software engineering overview and its importance of the development sector. The metrics and quality impact factors are addressed.

All the development process is manipulated according to the developers skill set and their involvement. The developer's activity and the relational model is constructed. The independent calculation made for the developers activity. The implementation impact and the related issues are further research work to enrich the quality software product by the researcher.

References

[1]. Chidamber, R., Kemerer, F.: A Metrics Suite for Object- Oriented Design. IEEE Trans.software Eng., Vol. 20, No. 6 (1994).

[2]. Dekkers, C.S. (1999). The Secrets of Highly Successful Measurement Programs. Cutter IT Journal, vol 12 no. 4, pp. 29-35

[3]. Fenton, N., S.L. Pfleeger (1997) Software Metrics: A Rigorous and Practical Approach, PWS Publishing Co. Jagdish Bansiya, and Carl G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," IEEE Trans. Software Eng., vol.28, no.1, pp.4-15, 2002.

[4]. Khaled El Emam,Saida Benlarbi, Nishith Goel, and Shesh N. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics," IEEE Trans. Software Eng., vol.27, no.7, pp. 630-650, 2001.

[5]. Lionel C. Briand, and Victor R. Basili, "Property-Based Software Engineering Measurement," IEEE Trans. Software Eng., vol.22,no.1, 1996. ISSN: 0975-5462

[6]. M.C. Paulk, B. Curtis, M.B. Chrissis, et al, Capability Maturity Model for Software, Software Engineering Institute,CMU/SEI-91-TR-24, ADA240603, August 1991.

[7]. Rachel Harrison, Steve J. Counsell, and Reben V. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," IEEE Trans. Software Eng., vol. 24, no.6, pp.491-496, 1998.

[8]. Rajender Singh, Grover P.S., "A new program weighted complexity metrics" proc.International conference on Software Engg. (CONSEG'97), January Chennai (Madras) India, pp 33-39

[9]. V.R. Basili and B.R. Perricone, "Software Errors and Complexity," Comm. ACM, vol. 27, pp. 42-52, 1984.

[10]. William Frakes, and C. Terry, "Software Reuse: Metrics and Models," ACM Computing Surveys, vol.28, no.2, pp.415-435, 1996.