Optimized Software Quality Assurance Model for Testing Scientific Software

G.Uma Maheswari[1] and Dr. V. V. Rama Prasad[2]

[1]M.tech, in Software Engineering

[2]Professor

Department of IT, Sree Vidyanikethan Engineering College,

Tirupati, A.P-517501, India

ABSTRACT

Software projects in R&D organizations differ in their quality assurance process compared to other production and business organizations. Major hard constraints are accuracy and precision. Estimated number of defects the product is likely to contain at release shall be as minimum as possible. Various models for assessing the quality of the software are developed and are in use. The most widely used models are McCall's, ISO 9000, CMM and COCOMO. These models sometimes become laborious during testing. Hence it is necessary to optimize the methodology of software quality assurance. So that it becomes robust, fast and economic. Based on the requirements an effort is made in this project to develop "Optimal Software Quality Assurance Model for Testing Scientific Software" which shall produce reliable and robust software engineering model to meet the requirements of IEEE 12207. Here through several procedures such as clustering of requirements, mapping, and so on, are used in order to find out the Defect Density of software and to predict its reliability and quality. The optimized model can overcome the process limitations of traditionally applied models and to provide an efficient way to assess the quality and other factors pertaining to scientific software systems. The optimized model is validated through comparing previous test results with results obtained from applying this model and the model was found working as per the requirements.

Keywords:

SQA, Optimized Testing Model, Model Testing, Quality Assurance Model for Testing Scientific Software.

1. INTRODUCTION

Software Quality Assurance is an important attribute of software projects. The number of varieties and complexity of software's increases continuously hence quality assurance must be used to make a balance between productivity and quality[1]. It is most important when dependable software-intensive scientific systems are developed, where delivering high quality is a major success factor.

Software projects pertaining to the scientific area differ in their quality assurance and testing process compared to other organizations. Certain important hard constraints as accuracy, precision and soft constraints as cost, effort and schedule are to be taken under consideration.

The estimated number of defects the product is likely to contain at the time of release shall be as minimal as possible. But defects themselves alone are not sufficient to predict the software quality. Software engineering researches still does not have a complete defect prediction for a software project although there are many models that predict software quality. There are many defect prediction models available such as empirical model, Rayleigh model, Constructive quality model and so on. But all of them base on defects alone which is quite insufficient. Also the above mentioned models target only the critical parts of the software product which is considered a drawback as in high-assurance scientific systems all modules should be given same priority.

Models of scientific software development practiced now do not fit the standard software engineering models. Hence, it has become essential to develop a new model which can overcome the drawbacks mentioned above. Accordingly the "Optimized Software Quality Assurance Model for Testing Scientific Software" is developed.

Here a model is proposed to estimate software quality in projects related to the field of science. In order to develop the required model that can overcome the above mentioned problem and fulfill the requirements of the IEEE 12207 standard, study of certain quality assurance concepts, advantages and disadvantages of models that are already proposed, and certain calculations to be conducted using software metrics were done and an optimized model is developed. The optimized model consists of stages in which certain defined processes are executed in order to find out the quality of given software.

After literature survey and studying the data of two specific projects "Global location mapping and analysis" and "Dynamic Pressure Data" procedures present in the developed model are applied and results obtained through the application of this model are used to declare the quality of the software. The model in this paper is developed such that it is reliable and overcomes the drawbacks of traditionally applied models. This model, as it has been checked with some lab data and has provided promising results.

2. OVERVIEW OF OPTIMIZED MODEL

Software quality assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. The methods by which this is accomplished are many and varied, and may include ensuring conformance to one or more standards, such as ISO 9000 and CMMI[2].

Model checking methods are being used widely for software verification. Here we propose a model for checking scientific software while overcoming drawbacks of traditional models such as uneven testing of a given software, i.e., by considering only critical modules and leaving out non critical modules resulting in defects being present in the software even after testing it, also complexity is another problem, a model should not be complex so that its understandability and application will be easy. Other drawbacks are models length and flexibility[3].

By using model testing techniques, it can result in the following benefits:

- Shorter schedules, lower cost, and better quality
- A model of user behavior
- Enhanced communication between developers and testers
- Capability to automatically generate many non-repetitive and useful tests
- Test harness to automatically run generated tests
- Eases the updating of test suites for changed requirements
- Capability to evaluate regression test suites
- Capability to assess software quality

One of the key steps to creating software testing processes that are specific to a given domain is to determine current practices in the domain[4][5]. This step is necessary to determine how current practices are working, in what ways they are not working, and to identify gaps in the testing process. The results of our study shows a correlation between requirements documentation required for scientific software and the actual implementation of the software product. They also identify three sources of requirements volatility[6] – changes in the theory, changes in the scope of the theory, and quality factors[7].

Scientific software, by which we mean application software that has a large computational component, models physical phenomena and provides data for decision support. This can be software that calculates loads on bridges, provides predictions for weather systems, images bone structures for surgical procedures, models subsystems at nuclear generating stations, or processes images from ground-based telescopes. There is no consensus on what the best practices for the development of scientific software are[8][9].

2.1 Optimized SQA model Description

The optimized software quality assurance model is developed for the testing of scientific software. This model is designed to overcome the process limitations of traditionally applied models such as COCOMO and its variations which only target the critical components of a software product.

The design approach of the optimized software quality assurance model was taken after a study of normal software development life cycles. The model was designed similar to a software development model[7], i.e., it comprises of stages of process application as compared to stages of software development. The model consists of requirement analysis in its first stage, mapping methods, Defect containment table, and other processes arranged in different stages according to the model execution and overall combined to form a model which is effective for testing scientific software. The optimized SQA model is designed in such a way that we can apply it not only for testing but also during model development as well.

Mapping process that is used in the model help locate the defect with ease, and not only that but also provide a view which can be used to find out all the modules, units or the interfaces in the product that were affected by either the defect or even by the defect rectification. Certain statistical testing methods and equations have also been introduced here so that the model not only is conceptually developed but also has a mathematical base to back up the results or assessment of quality of software which it provides.

In this model, we use clusters, units and consortiums, which will be explained in the model below. Here not only do we consider internal attributes but also few external attributes have been used for the model development, memory and time are the two main external attributes that are also used for calculations in the quality assessment. Here certain critical values, defect densities and interface or cluster combinations, data flow accuracy and so on also have been utilized.

A tree structure is also used. It shows the project testing completion, as in the project each module is being tested, whatever modules(or clusters as in the model) are finished off are then added to the tree and so when the tree is complete so is the project. In this model, the product that is to be tested is first translated into a model itself, according to the steps defined in the model and then it is mapped using the mapping methods, which too is defined in the model in a step by step process. The defects that arise are tabulated in the defect containment table which is designed and developed as given below in the model.

Usually there is a lot of confusion as to where the defect actually occurred in the module or a unit, in the developed model this problem is targeted with the usage of mapping methodology and the DCT (Defect Containment Table).

3. OPTIMIZED MODEL DESIGN

The optimized software quality model is designed and developed to assess the quality of scientific software. As any other model this comprises of various stages of application.

This model may be applied for both testing as well as during the development of the software product. To clearly explain about the model, it is divided into two parts, exterior and interior. The exterior part of the model shows the various stages comprised in it, where as the interior part shows the various procedures that are to be performed in each stage of the model.

Two partitions of the model:

1. Exterior
2. Interior

3.1 Exterior

In the exterior part of the model, it consists of three stages. The three stages themselves consist of different stages of a product's life cycle comprised in them. Such as the infant stage consisting of analysis and design parts, the production stage consisting of coding, implementation and connections and in testing stage the verification is done. In the exterior part i.e., the outer layer of the project shows three stages of the model as shown in the diagram below.

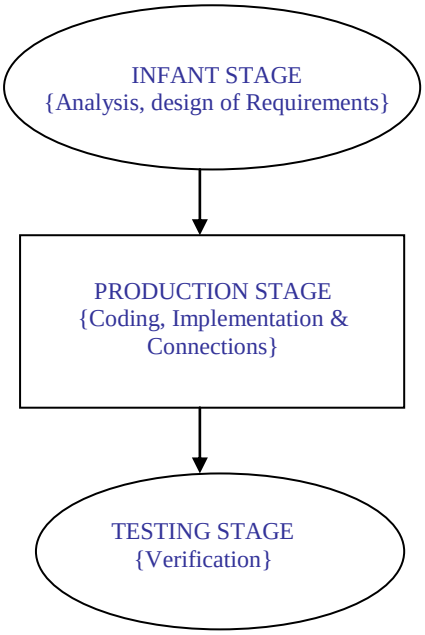

Figure 1: Stages of Optimized model

Each of the above mentioned stages in Fig.1 comprise of sub stages and different procedures that are to be applied on the software being tested. The three stages of the model are mapped from infant stage to the testing stages.

3.1.1 Infant Stage

The infant stage comprises of analysis and design sub stages where requirements are analyzed and the software product is translated into a model. The model translation is done by grouping up similar requirements and then allocating them into their respective clusters. What the clusters are and their use is explained in the production stage .In the infant stages all the requirements are clearly classified and mapped on to the respective design.
R1, R2… = Requirements
C1, C2…. = Clusters

3.1.2 Production Stage

In the production stage the model is first split into parts known as clusters and then added up into a complete consortium. Here the emphasis is on requirements being mapped on to code and then certain internal procedures to be followed.

The above mentioned clusters consists of requirements being mapped to their respective modules, different clusters have different set of requirements mapped to them. A single requirement need not be connected to a single cluster; it can be mapped to different clusters also depending upon its usage. To test the clusters operations etc various Input and Output sets have been created. At the coding stage restricted input and outputs are used to reduce the testing difficulty.

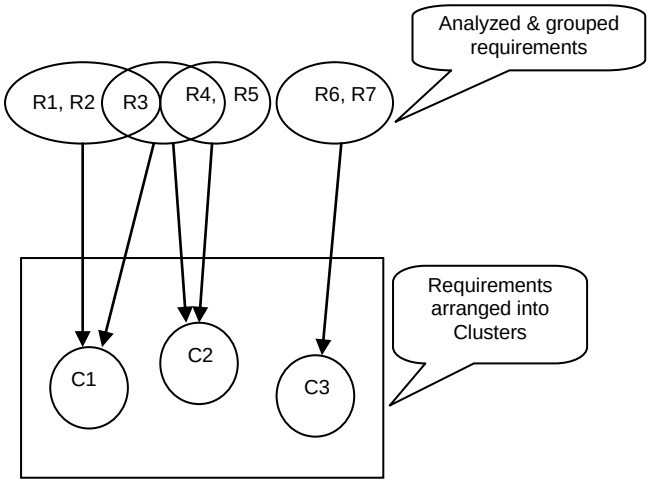

Figure 2: Infant stage analyses of Requirements and allocation into clusters.

This stage also consists of Units, two or more clusters being connected through their interfaces. The use of units in this model is to test the interfaces, data flow accuracy and also the time usage attribute in the system.

The consortium is the product in which all the different clusters are grouped together and connected via interfaces. The entire product which is separately tested as clusters and units is finally grouped together to form a consortium which is the whole product being tested at once.

3.1.3 Testing Stage

In the testing stage verification is done. In the production stage testing is also done but here it differs as entire product is verified so that defects or any problem that were not found previously are found in this stage. This stage i.e., the verification plays a major role in testing of the software. The procedures done here are explained in the interior part.

3.2 INTERIOR

The interior of the model consists of different procedures to be run within the above mentioned stages of the optimized model. The procedures are arranged in a step by step process. Within those steps mathematical calculations are also performed. The steps provided show that the optimized model developed has the presence of both conceptual and mathematical base in the process of assessing the quality of small or medium scale scientific software.

4. MODEL IMPLEMENTATION

4.1 Model Procedures and its Application

The model and its procedures are developed with adherence to several rules and constraints.

- The model developed should overcome the drawbacks of traditional models.
- The model should adhere to IEEE 12207 standard.
- The optimized model is developed for small and medium scale scientific softwares
- All processes included in the steps are simple and effective.

- The calculations present in the model should provide a solid base in the software quality assessment.
- The model provides even testing throughout the software, i.e., all modules are tested evenly so that proper results are provided.

The model is designed in a way that provides a full view of the software to be tested and also it provides access to all interactions that are present between modules so that not only normal coding defects but also data flow errors, interface errors and data inconsistency errors are also found.

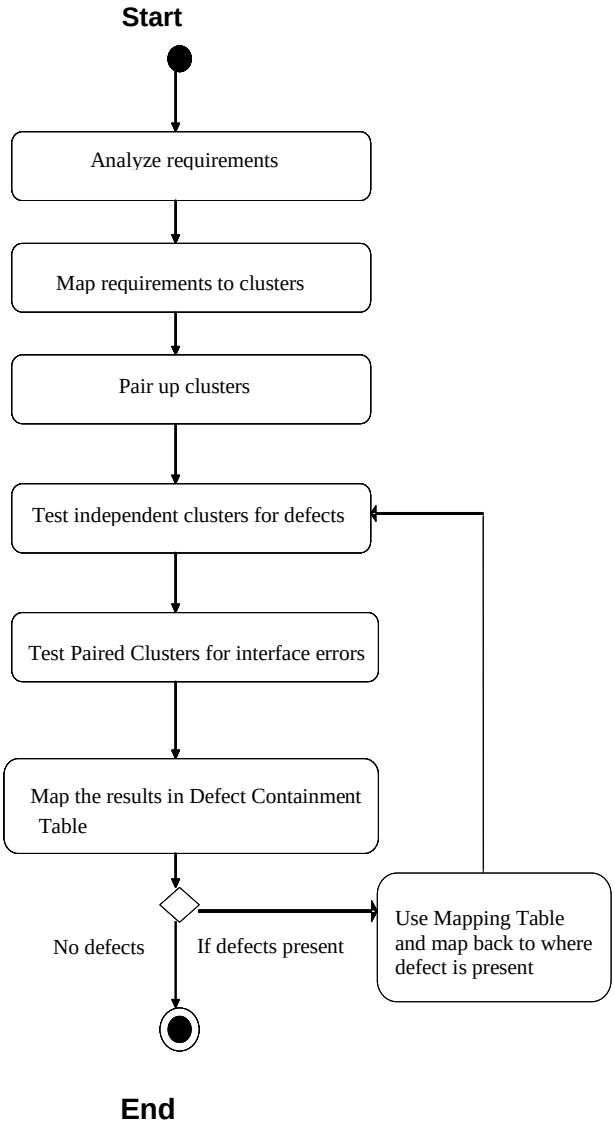

Figure 3: Activity Diagram for the process in Optimized Model

The procedures performed in the model are given below in a step by step procedure:

1. In the infant stage, first the software product is translated into a model and all the requirements are individually sought after and should be clearly analyzed and then grouped up. While the requirements are being analyzed if a requirement is part of two groups then it can be mapped to two clusters as seen in diagram below.
2. In the production stage, all the above analyzed requirements are grouped up and arranged into different into clusters, according to their relations.
3. A cluster is a set of requirements that belong to the same module and in cluster some exceptional requirements may be present which belong to other modules but are interlinked with requirements in this module.
4. An Input and Output sets are created for testing the product. Each cluster is then tested using the input and output sets.

- An input set is a set of alternative inputs that can be given to a module during testing.
  Input set is denoted by I, and there can as many as required

  $\sum I = \{I1, I2, I3 \ldots\ldots.\} \rightarrow (1)$

- An output set is a set of outputs that is expected when input from a specified input set is given to a module.
  Output set is denoted by O, and the numbers of output
  sets are shown as

  $\sum O = \{O1, O2, O3\ldots\ldots\} \rightarrow (2)$

- The number of requirements not implemented properly here are found out by

  $nrp = (\sum I + \sum O) * [PS - FS] \rightarrow (3)$

  In the above equation PS indicates sets that have passed the tests and FS indicates sets where failure occurred.

5. When the clusters are Tested then two or more clusters according to the interface requirements are combined into unit. A unit can be described as module collaboration; links between two or more modules form a unit. Then the unit is tested for data flow accuracy, interface errors, and processing errors.
6. All modules/clusters that are to be integrated to form a unit are tested in sets of two each i.e., combinations of clusters where integration is done are tested.
   If there are 4 clusters then
   C={ (1,2),(1,3),(1,4),(2,3),(2,4),(3,4)}.
   In the above example (3, 4) or (2, 4) need not be tested if there is no interface requirement is between 3 or 4, or between 2or 4.
7. A consortium is the entire product being connected up to form complete working software.
8. Critical values are fixed for each cluster to assess the quality of the software.
   Critical value of a cluster is given as cr
   cr = no. of functions / (no. of input sets + no. of output
   sets)
   Let CR be the average critical value

   CR = avg critical value of all clusters (no. of functions /
   (no. of input sets + no. of Output sets))→ (4)

Let Number of defects found be nd,
Number of modules be nm then,

nd/nm > CR ➔ the defect content is high.
nd/nm < CR ➔ the defect content is low.

9. Each cluster is now attached with an input and output set.
10. Usually during project development itself, restricted input should be practiced at coding area so that testing can be more simplified as well as it results in easier way of finding out where the problem is if a defect arises.
11. A software product can have any number of input and output sets as required by the modules.
12. Next a Mapping table is created and results of the tests are entered here, where Y being true or passed and N being False or failure. Here the values Y and N in the table give a precise idea about where and how a defect has entered and is present in a system and it also provides idea on which modules does this defect have an effect on. So that correction can be made with ease. Given below is the defect containment table.

Table 1: Mapping table presentation example

	REQ	CLUSTER 1	CLUSTER 2	CLUSTER 3
	R1	Y	Y	
	R2	Y		
	R3			Y
	R4		Y	
	R5	Y		Y
	R6		Y	
INPUT SETS	I/P 1	Y	Y	
	I/P2	Y	Y	Y
OUTPUT SETS	O/P 1	Y	Y	
	O/P 2	Y	N	Y
	DPV	Verified	Re-verify	Verified

In the above diagram the DPV indicates defect presence verification, where ever N is there then it indicates defect presence. The mapping table indicates that in output set 2 the outcome is N i.e., wrong out come and since it was for cluster 2 then cluster 2 can be checked for errors.

According to the diagram fig.4, the procedure is run. In the testing phase all modules are tested in form of request response methodology. Also the modules are to be categorized into two groups MAC and MIC, MAC indicates majority class and MIC indicates minority class.

Also we classify each module or cluster as DP defect present and non defect present NDP, and also critical weights are added to them and so with this we can say that if two modules of DP with high critical weight equal to or more dangerous than 5 modules of DP but with low critical weight. Usually we consider MAC > MIC. If MAC consists of DP and MIC consists of NDP then according to MAC>MIC the software is completely defect prone, but if it is vice versa then software product quality is at an acceptable level.

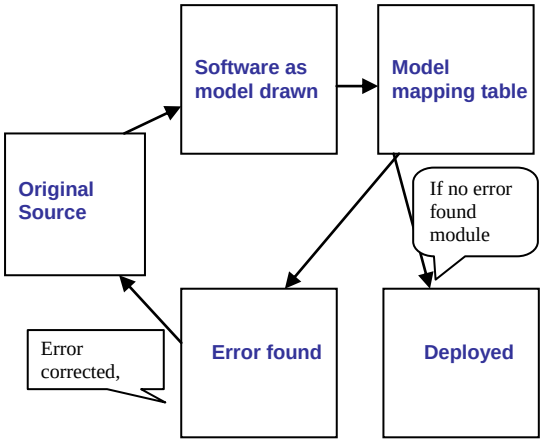

Figure 4: Procedure for execution of model application

13. Finally after finishing all the above steps, mathematical calculations are done as given below
By calculating through the formula of products operational interface efficiencies we can be able to estimate the overall product working capability and through that find out the product quality.

Product operational Interface efficiency is given by
POIE = (C-Ec (nrp))*Dp+(Mu-Mr)+(Tu-Tr)➔(5)
Where C is the number of cluster sets to be tested,
Ec is the number of errors found in the clusters connections, which normally can be defined as when connected the number of Input and Output faults occurred
Dp which is the data flow between the connections.
Mu is the memory that is used by the system
Mr is the memory that is actually required by the system
Tr is the time required and
Tu is the time utilized by the system
After the above calculation, we can also perform integrated calculation given
Below, this calculation can be denoted as IC,

$$IC = \int_0^c Px^t \, dx \rightarrow (6)$$

In the above equation P is the total number of clusters connected to form a consortium, t is the number of interfaces present, and c is the total number of cluster sets. For example for the above equation let if P = 5, t = 12, c = 4 then the out come will be IC = 240.

Similarly in equ (5) if we calculate according to the test results and obtain a value of 350 then for both equ (5) and equ (6) we have to round off the resultant value to its nearest complete value i.e., for example we have obtained IC = 240 so we have to round it off to 300 or calculate the percentage of 240 for 300, and similarly if value of POIE = 350 then we have to round it off or calculate 350 is how much percentage of 400.

By rounding off the value of the above equations if we get the value of the above both POIE and IC greater than or equal to 82% then the software product can be considered to be of good quality. The software product quality assessment can be given as or considered as acceptable.

Product testing and its completion can be known by following a tree structure, here nodes represent modules and after proper

testing only each node is added to the tree and if you arrive to the bottom of the tree with all nodes present in it then your project quality assessment can be deemed completed. In the below diagram, A, B, C, D are different modules and in those modules:

- A, B, C and D are inter connected modules.
- B1, B2 are sub modules of B.
- C1 is a sub module of C.
- C11 is also a sub-module of C1.
- B11 and B12 are sub modules of B1.

If each module is properly tested only then it will appear on the tree and if the tree is complete then the testing is also complete.

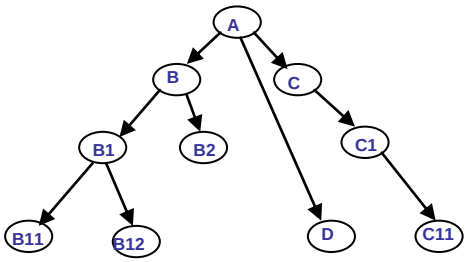

Figure. 5: Tree Structure for analysis of the tested product using optimized model

5. MODEL TESTING

Testing of the model is done by applying the model on two scientific software projects. Here data has been gathered on two projects, they are "Global Location Mapping and Analysis" and "Dynamic Pressure Data". The optimized model has been applied on the two projects for assessing the quality of the softwares and results were obtained. The applied procedures, methods and calculations done are explained below.

5.1 Sample testing project.1: Global Location Mapping and Analysis

5.1.1 Project Description

Global Location Mapping and Analysis project is a virtual map of the earth, various locations of the world are mapped onto a graphical map. For the locations on the graphical map data is gathered and stored in the database. Each location presented on the map through code is attached with a link to the data pertaining to the selected location.

The map is designed to provide users with specific data required by the user on the location chosen by him/her. The data consists of latitudes, longitudes, terrain specifications, routes to and from the chosen location to another selected location and the location's local data such as its main areas and its importance.

This software can also be used as a search engine in finding locations on the graphical map based on given latitudes and longitudes degrees. Search of the location can be single or multiple i.e, an exact location can be found according to the latitudes and longitudes or areas between two sets of latitudes and longitudes can be acquired using this software.
The graphical map is a world map with links attached to all places displayed on the map. By clicking on each link, data of the link of the location is displayed. This project basically consists of

- Client side Scripts – HTML as User Interface.
- Server side Scripts – JSP
- Database – Oracle 9i

5.1.2 Testing of GLMA Project

5.1.2.1 Requirements Analysis

All the requirements found according to the documentation of the project are analyzed according to the optimized model and are tabulated below:

Table 2: Requirements Declaration

Requirement Declaration	Definition
R1	Graphical Map presentation
R2	Graphical Map Link Processing
R3	Single Explicit Input provided by user
R4	Input through selection.
R5	Multiple Explicit Input provided by user
R6	Accurate Data stored in Database
R7	Output data presented by Database

All the above declared requirements are analyzed according to the model's analysis perspective and also through the diagrams presented in the documentation such as usecase and sequence diagrams etc. As per the usage of the product or project the requirements are analyzed and are classified into clusters. Given below are steps that are taken for testing this software according to the optimized model.

1. Requirements analysis is done according to the usage and specifications present in the project.
2. Requirements are declared according to the project usage i.e., each task that need be performed in the project can be considered as a requirements of the project itself.
3. After analysis all requirements are clearly defined and are arranged into clusters so that actual testing can be started.
4. Number of Clusters present = 3
   C1 = {R1, R2, R7}
   C2 = {R3, R4, R5, R7}
   C3 = {R2, R6, R7}
5. Input value sets and output value sets for the project are defined as given below

Input sets ∑ I = {I1, I2, I3 …….}
I1= {11.74 N, 92.65 E}→ Single Set
I2= {User Selection on Map}→ Assuming selection of Diu
I3= { (17.04 N, 80.09 E), (30.42 N, 76.54 E), (20.12 N, 7.00 E)}→ Multiple Sets

Output sets ∑ O = {O1, O2, O3 …….}
O1= {Output of Data pertaining to latitude 11.74 and longitude 92.65}
O2= {Data pertaining to location of Diu}
O3= {Data Pertaining to the multiple inputs provided by user → AP, Daman & Diu, Dadra & Nagar Haveli}

6. The number of defects that can arise due to requirements not implemented properly here are found out by
   nrp = $(\sum I + \sum O) * [PS – FS] = (3 + 3) * [4 – 2] =$ 12
   PS = 4 → (I1, I3, O1, O3)
   FS = 2 → (I2, O2)
7. Cluster combination pairs are defined for testing interfaces and data flow accuracy,
   C={(C1, C2), (C1, C3), (C2, C3)}
8. Critical values are fixed for each cluster to assess the quality of the software.
   Critical value of a cluster is given as cr
   cr1 = 3/ (1+1) = 3/2
   cr2 = 5/ (3+3) = 5/6
   cr3 = 3/ (1+1) = 3/2
   CR be the average critical value
   CR = ((3/2)+(5/6)+(3/2))/3 = 1.27 ~ 1
9. Mapping table is given below to show the Defect Presence and its Verification process.

Table 3: Defect Containment Table

	REQ	CLUSTER 1	CLUSTER 2	CLUSTER 3
	R1	Y		
	R2	Y		Y
	R3		Y	
	R4		Y	
	R5		Y	
	R6			Y
	R7	Y	Y	Y
	I/P1	Y	Y	Y
INPUT SETS	I/P2	N	N	N
	I/P3	Y	Y	Y
	O/P1	Y	Y	Y
OUTPUT SETS	O/P2	N	N	N
	O/P3	Y	Y	Y
	DPV	RE-VERIFY	RE-VERIFY	RE-VERIFY

10. POIE = (C-Ec (nrp)) * Dp + (Mu-Mr) + (Tu-Tr)
    = (3 – 6(12)) * (3) + 0 + 0 = 207 ~ 82.8%(207
    rounded off by 250 integer usage)
    Perform integrated calculation given
    Below, this calculation can be denoted as IC,

$$IC = \int_0^c Px^t \, dx$$

$$= \int_0^3 3x^9 \, dx = 81 \sim 81\% \text{ (Rounded off by 100)}$$

And through comparison of both POIE and IC equations and their values being approximately equal to each other, the projects quality assessment can be declared as being 82% accurate, still verification need to be done as errors are present in the project. The last stage tree diagram is to show that how each module is inter-related and is tested.

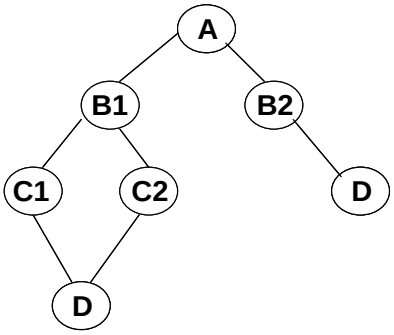

Figure 6: Tree Representation of all modules and their connections and testing stages.

5.2. Sample testing project.2: Dynamic Pressure Data

5.2.1 Project Description

The software product Dynamic Pressure Data is a data acquisition and plotting application that will be used for real-time plotting of pressure/thrust parameters during static tests. This application provides numerical display, real-time graph and storage of acquired thrust/pressure raw data during test. Printable processed data file for the displayed parameter is also be generated. The system helps in knowing the performance of the article under test in real time during test.

The following decomposition description records the division of the software system into design entities. It describes the way the system has been structured and the purpose and function of each entity.

During testing of typical pressure data which is continuously varying related to a parameter, Motor Pressure/Thrust are acquired using dedicated data acquisition systems. Raw data processing will be done off-line after the test to prepare the performance report. This software shall have the capability to acquire and plot thrust/pressure data of two redundant channels in real time during the test. The software shall also generate printable processed data file for the parameters displayed during real-time.

The Dynamic pressure data software shall be a window-based, self-contained and independent software product which shall be used for the real time display of thrust /pressure parameters during testing.

5.2.2. Testing of Dynamic Pressure Data

5.2.2.1 Requirements Analysis

All the requirements found according to the documentation of the project are analyzed according to the optimized model and are tabulated below:

All the above declared requirements are analyzed according to the model's analysis perspective and also through the diagrams presented in the documentation. As per the usage of the product or project the requirements are analyzed and are classified into clusters.

Table 4: Requirements Declaration for DPD project.

Requirement Declaration	Definition
R1	Measure thrust /pressure during static test
R2	convert thrust/pressure analog signal to digital
R3	Updating the details in calibration file
R4	Updating of the input details file
R5	Availability of lower and upper bound data file to user
R6	Reading input requirements Test no, Rocket motor name, channel name, test duration, plot parameters & predicted data files from input details file
R7	Get calibration details from channel calibration data file.
R8	Initialization of the NI PCI-6034E ADC card for acquiring the data
R9	Plotting of predicted data, mission bounds plots
R10	Data acquisitions for 8 channels and store the data in binary file
R11	Plotting of one thrust /pressure redundant parameter in Real-time during test superimposed on the predicted data and mission bounds plots
R12	Displaying the one thrust/pressure channel data and time in digital display
R13	Generate text file of thrust/pressure parameter data plotted in real time
R14	Configuring the acquisition mode (manual/Level trigger), trigger level, and configuring the plot color properties
R15	provide menus for top level system Feature Selection
R16	Separate Display views should be provided for each major functions and features of software requirements
R17	provide Input file selection feature
R18	Provide error, alert message, operation completed status
R19	Command Buttons
R20	Read the Input details file, nominal, upper and lower bounds Files, channel Calibration file
R21	Configure acquisition Properties, Real time Plot properties
R22	View the real time plot details selected and Thrust/Pressure channel calibration details
R23	Start Real Time Graph, Stop Real Time Graph, Exit, Real Time Display, Initialization of DAQ card Data Acquisition.
R24	Real time Plotting and digital display
R25	Real time data logging of thrust/pressure channels and converting to engineering units.
R26	Real time plot and digital display of thrust/pressure parameters.
R27	Real time plot and digital display refresh shall be at least for every 125ms.
R28	Maximum duration of real time display duration is 200 seconds.
R29	The Real Time Plot of the Thrust/Pressure parameter must start when the motor thrust / pressure starts rising.
R31	Trigger level for starting the real time display shall be 10% of nominal voltage.
R32	The Time display also must be provided, taking start of the plot as the '0 s' and incrementing every one second.
R33	Raw and processed data kept in ram during the test and storage to files shall be done at the end of the test.

Given below are steps that are taken for testing this software according to the optimized model.

1. Requirements analysis is done according to the usage and specifications present in the project.
2. Requirements are declared according to the project usage.
3. After analysis all requirements are clearly defined and are arranged into clusters so that actual testing can be started.

4. Number of Clusters present = 5
   C1 = {R4, R6, R17, R20}
   C2 = {R3, R7, R20, R22}
   C3 = {R1, R2, R9, R10, R11, R12, R13, R14}
   C4 = {R15, R16, R18, R24, R27, R28, R30, R31, R32}
   C5 = {R5, R8, R19, R21, R23, R25, R26, R29, R33}

5. Input value sets and output value sets for the project are defined as given in the actual tests that were run on the software. The input sets and output sets for this project can only be assumed as live project is not available to perform the actual testing.

   Let us assume the input and output sets given below.
   Input sets $\sum$ I = {I1, I2, I3 …….}
   Output sets $\sum$ O = {O1, O2, O3 …….}

6. The number of defects that can arise due to requirements not implemented properly here are found out by finding out requirements implementation problems

   nrp = $(\sum I + \sum O) * [PS – FS] = (21) * [21-0] = 441$
   Cluster combination pairs are defined for testing interfaces and data flow accuracy,

   C= {(C1, C2), (C1, C3), (C1, C4), (C1, C5), (C2, C4), (C2, C5), (C3, C4), (C3, C5), (C4, C5), (C2, C3)}
7. Critical values are fixed for each cluster to assess the quality of the software.

   Critical value of a cluster is given as cr
   cr = no. of functions / (no. of input sets + no. of output sets)
   cr1 = 4/ (4+4) = 1/2
   cr2 = 8/ (4+4) = 1
   cr3 = 8/ (2+2) = 2
   cr4 = 9/ (2+2) = 2.25
   cr5 = 9/ (3+3) = 1.5
   CR be the average critical value

   Number of function, input sets and output sets taken in the
   above equations are assumptions taken from the previous
   tests performed on this project.

   CR = avg critical value of all clusters (no. of functions / (no.
   of input sets + no. of Output sets))
   CR = (0.5+1+2+(2.25)+(1.5))/3 = 1.45 ~ 1

8. Mapping table is given below to show the Defect Presence
   and its Verification process.

9. POIE = (C-Ec (nrp)) + Dp + (Mu-Mr) + (Tu-Tr)
   = (5 – (21*21)) + (5) + 0 + 0 = 441 ~ 98%(441 rounded off by 450 Integer usage)

where Dp (multiply if equation result is below 50 and add if result might increase above 50)which is the data flow between the connections obtained by testing the cluster sets connections, if data flow accuracy is maintained then put down the number of clusters working accurately and tasks done by them.

Table 5: Defect Containment Table

REQ	CLUSTER1	CLUSTER 2	CLUSTER3	CLUSTER4	CLUSTER5
R1			Y		
R2			Y		
R3		Y			
R4	Y				
R5					Y
R6	Y				
R7		Y			
R8					Y
R9			Y		
R10			Y		
R11			Y		
R12			Y		
R13			Y		
R14			Y		
R15				Y	
R16				Y	
R17	Y				
R18				Y	
R19					Y
R20	Y	Y			
R21					Y
R22		Y			
R23					Y
R24				Y	
R25					Y
R26					Y
R27				Y	
R28				Y	
R29					Y
R30				Y	
R31				Y	
R32				Y	
R33					Y
IP SET	Y	Y	Y	Y	Y
OP SET	Y	Y	Y	Y	Y
DPV	VERIFIED	VERIFIED	VERIIFIED	VERIFIED	VERIFIED

10. Perform integrated calculation given
Below, this calculation can be denoted as IC,

$$IC = \int_0^c Px^t\,dx = \int_0^{10} 5x^9\,dx = 450 \sim 100\%$$

And through comparison of both POIE and IC equations and their values being approximately equal to each other, the projects quality assessment can be declared as being 99% accurate, still verification needs to be done as errors are present in the project. The last stage tree diagram here shows not the actual modules but modules made up through requirements segregation, each node represents a part which has been tested completely and how each one is inter-related.

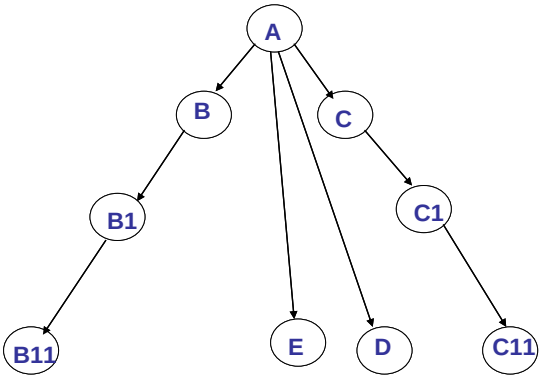

Figure 7: Tree structure of dynamic pressure data testing presentation.

6. EXPERIMENTAL RESULTS

The results of applying the optimized model on the projects "Global Location Mapping and Analysis" and "Dynamic Pressure Data" are provided below in the form of graphs.

6.1 Result analysis

Below Graph shows the results of testing of both the softwares before and after the application of the optimized model.

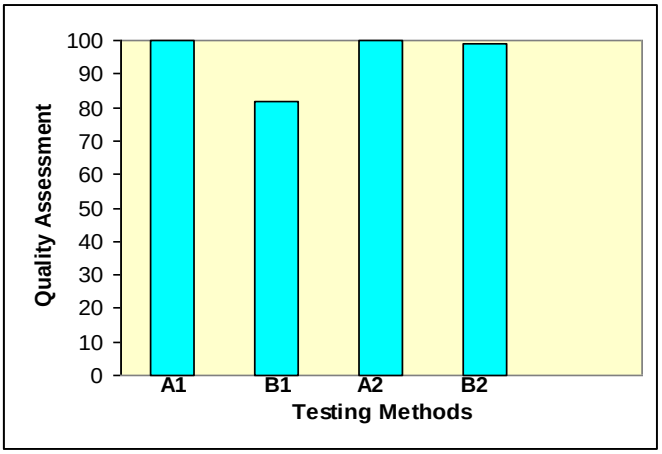

Figure 8: Graph Representing Test Results OF GLMA and DPD projects

In the above shown graphs, both for GLMA and DPD projects

- A1, A2 represents test results of projects before the application of optimized model.
- B1, B2 represents test results after the application of the optimized model.
- A1 & B1 represent GLMA project test results.
- A2 & B2 represent DPD project test results.

For project-1, several defects were found through the application of the optimized model, the calculated results can be viewed in chapter 6. By using the mapping technique that is through the use of Defect Containment Table the defects can be easily traced back to their exact location. The quality of the software project in terms of calculations resulted in 82%. In the below given graph A shows through previous test results, the quality as near to 100% where as B which is optimized model testing shows result of quality being 82%.

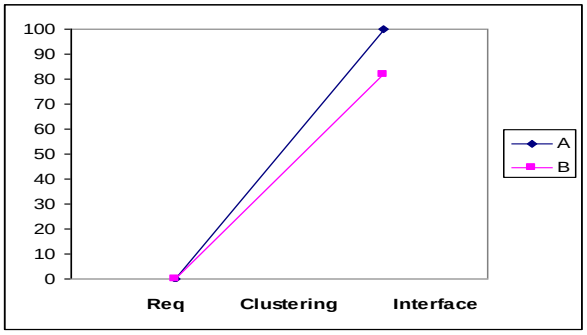

Figure 9: Graph displaying Test results of GLMA project.

For project -2, the calculated results concur with the previous tests results and so the quality of the software was assessed and the optimized model was effective in application to the project. The quality of the software project in terms of calculations resulted in ~100%. Both test results, i.e., previous and current model applied test results show approximately same results for DPD project.

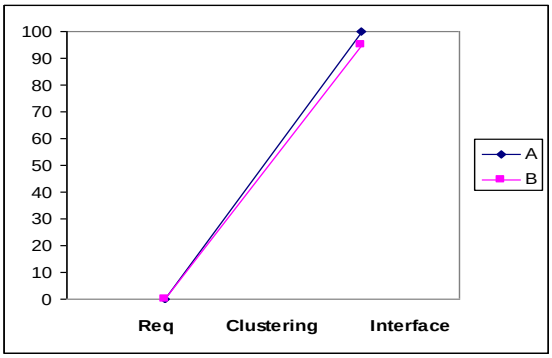

Figure 10: Graph Display of test results of DPD project.

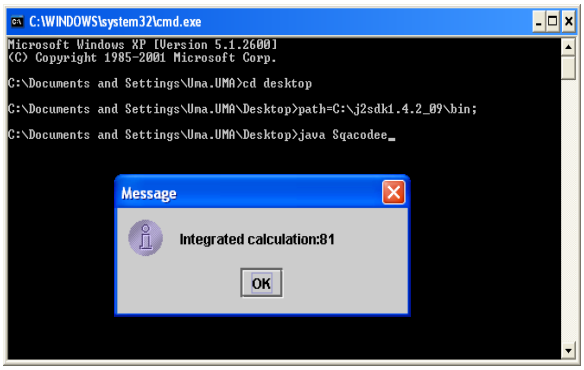

Fig.11: Screen shot of output for GLMA project

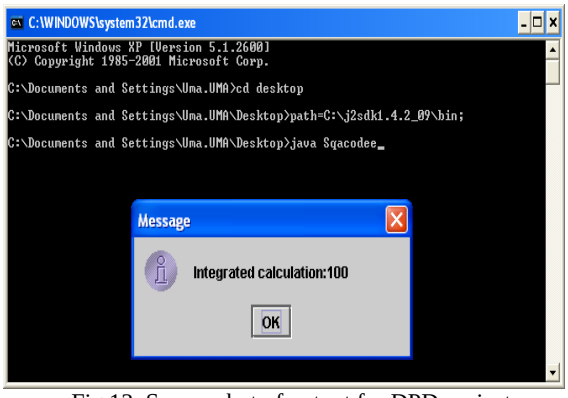

Fig.12: Screen shot of output for DPD project

7. CONCLUSION & FUTURE WORK

This model is developed for small and medium scale scientific software projects. Application of this optimized model has given positive results on assessing quality of softwares. As data acquired by applying this model on scientific project shows that this can increase the quality as well reduce testing time, cost and schedule of the project.

By comparing this model's approach with other models approaches an added advantage is acquired. Unlike other models that give importance to critical modules only, the provided model gives equal importance to all modules also considers more relevant influencing factors. And thus provides more reliable results and shows that this model and equations are feasible.

The applicability of the model and its methods has been shown by applying this model for the assessment of quality for two projects. The usefulness of the resultant model has been seen by the resultant data acquired from the two model applied projects. Only on a small number of projects this model has been applied, since more thorough work has not been, threat to validity of this model can occur. But future work in this area with application on vast area of projects, an extension of the current procedures and mathematical methods of this model can lead to a more accurate, precise and effective model.

8. REFERENCES

[1] Feldman.S, Software quality assurance is more than testing and IEEE standard 12207. Retrieved 5/3/2009, 2009, from http://elsmar.com/Forums/showthread.php?t=11148, 2005.

[2] (Webopedia, 2009) Webopedia. (2009). Retrieved 5/5/2009, 2009, from http://www.webopedia.com/TERM/S/Software_Quality_Assurance.html

[3] NASA, Software definitions. Retrieved 5/5/2009, 2009, fromhttp://ezproxy.library.nyu.edu:13976/office/codeq/software/umbrella_defs.htm

[4] Hayes, L. (2003). Column info : Test organization strategies: Centralized, distributed, or hybrid?2008(11/25/2008)

[5] Podlogar, Y. (2002). Taking advantage of a centralized QA organization. Retrieved May 28, 2009, 2009, from http://www.stickyminds.com/s.asp?F=S3516_ART_2

[6] Topping.S. (2009). Organizing localization in large companies, achieving balance between centralized and decentralized models. Multilingual Computing and Technology, 10(6)

[7] (Ogale, 2005): Ogale, C. (2005). Testers vs programmers. Retrieved 5/26, 2009, from http://www.indicthreads.com/1330/testers-vs-programmers-2/

[8] Kitchenham, BA et al (1995) Towards a framework for software measurement validation IEEE Trans. Software Engineering 21(12) pp. 929-944

[9] Gilb, T (1987) Principles of Software Engineering Management Reading, Massachusetts: Addison-Wesley

[10] Evans, M and Marciniak, J (1987) Software Quality Assurance and Management New York: Wiley

[11] Shen, V (1987) Quality Time IEEE Software September 1987, p. 84

[12] Basili, V and Rombach (1987) "Implementing Quantitative SQA: A practical model", IEEE Software September 1987 pp. 6-9