Organizing Software Architecture based Measurement Analysis (OSAMA) Model

Osama M. Abu-Elnasr[1], Mohammed A. Abo-Elsoud[2] , Magdy Z. Rashad[3], and Gamal M. Aly[4]

[1,2,3] Computer Science Department, Mansoura University, College of Computers and Information Sciences, 60 El-Gomhorya st, P.O. 35516, Mansoura,Egypt.

[4] Computer Engineering Department, Ain Shams University Name, Faculty of Engineering, Cairo, Egypt

Abstract
Today's software engineering organizations seek to develop and improve the models that follow and are trying in various ways to reach the highest levels of quality in order to achieve user satisfaction.
Organizing Software Architecture based Measurement Analysis (OSAMA) model provides a way that allows a great control on management, development, review, prediction and estimation issues. OSAMA combines a wide range of software quality assurance (SQA) components and their related static analysis and testing activities all of which are work together in a coherent manner to provide cost-effective approach for achieving high level of quality. It also introduces the mapping between OSAMA and RUP (Rational Unified Process) model as iterative development methodologies. An empirical analysis of NASA CM1 software metrics has been done using statistical regression analysis approach to justify the model. The results show that the behavior of the faults and hence the software quality can be predicted early and accurately based on static measurement analysis.
Keywords: *SQA, Static Analysis, Testing, Fault, OSAMA, RUP, Regression Analysis.*

1. Introduction

Nowadays, the principles of software quality is no longer just a trail of magnificence or fantasy that the organization seeks to achieve, but it has already becomes a matter of life or death. In order to improve software quality, software testing process responsible for detecting software defects and assessing the readiness of the product to be released should be enhanced and integrated early during the development life cycle.

Software testing and static analysis techniques proved that a great cooperation has been established, which is reflected by the fact that their collaboration seem to provide cost-effective approach for achieving high level of quality [1-3]. Testing as a dynamic technique still necessary at the validation level, while static analysis techniques provide complete coverage at the verification level. These static analysis techniques that can be integrated early in the development process as a way of re-evaluating the resulting artifacts through the various stages of the system development, assessing the quality level of these artifacts and producing a decision to move towards the next phase or reassign the artifacts to the development team for repair.

Software measurement analysis, fault prediction and failure estimation techniques become rich area of research related to verification activities that give us a complete control of the behavior of the software and its related quality attributes. By evaluating the attributes of the software, we can know its status, characteristics and behavior, [4-6]. The fault proneness [7-10], defect density [11-13], and failure radiation information [14] provide important guidelines to testing practitioners to prioritize their testing effort and assign verification and validation activities.

There have been great efforts in the attempt to the usage of machine learning techniques in software quality assurance issues, especially software fault prediction and estimation for assessing the correctness and reliability factors of the software quality. These attempts remain work as an individual unit in isolation from the rest of the other elements of SDLC (e.g. Management activities, software testing activities, Static analysis activities, software metrics, fault prediction and failure estimation activities). Research in the area of integrating these elements were rarely articulated to form a unified framework combines all activities of software quality as an integrated system where artificial intelligence techniques play an essential roles to empower the software testing process for better quality, [15-18].

The proposed model extends the work done by Danel Galin [1], Yue Jiang [10] and Radhika D. Amlani [19]. Danel Galin [1], has been introduced SQA system that combines a wide range of SQA components and classified them into six components; Pre-project components, Project life cycle quality components, Infrastructure error preventive and improvement components, Software quality management components, Standardization, certification and SQA assessment components, and Organizing for SQA the human components. Yue Jiang [10], has been addressed the effects of various software metrics along SDLC (System Development Life Cycle) on the accuracy of the fault prediction models. He established that the fault prediction models built from a combination of requirement, design, and code metrics provide better performance than models built from any metrics subset QA framework that integrates the best test and QA practices for handling software quality improvement process. Radhika D. Amlani [19] has been introduced information of the comparative study of different SDLC models.

The paper seeks to concentrate on finding the way to answer the following research questions; it is possible to incorporate the software quality assurance elements such as; testing practices, static analysis techniques, software metrics, fault prediction, and failure estimation activities into a coherent framework to provide a more realistic estimate of the software behavior. It is possible to construct an organization of software measurement analysis that relates the software artifacts and software metrics can be generated from these artifacts along SDLC. It is possible to address the relationship between faults and failures, and their causes, their effects and their prediction and estimation techniques.

The remainder of this paper is organized as follows; Section 2 highlights the problems associated with current area of research. Section 3 provides the full description of the proposed framework. Section 4 introduces justification of the proposed model. Section 5 introduces the conclusion and the future work.

2. Problem Description

Nowadays, the efforts that have been done for studying the software testing for improving software quality still suffer from several drawbacks such as:

- The absence of a conceptual framework that provide a clear distinction between software fault and failure, their causes, effects and their prediction and estimation models.
- The absence of a clear distinction between validation and verification activities and their scope in handling software quality.
- The absence of a complete software metrics framework that reflects the behavior of the software being developed or fails to discover the existence of bugs in early stages.
- The need to complete those frameworks that integrate the individual units of testing activities, static analysis techniques and their related quality activities.

It becomes very important to develop a conceptual framework that completes and integrates the previously work in relating the correctness of the software and its reliability. It will also articulate expert beliefs about the dependencies between different metrics and their effects on assessing the construction of a robust fault prediction and failure estimation model.

3. Proposed Model

Organizing Software Architecture based Measurement Analysis (OSAMA) model is built on the idea of reorganizing the activities of the software development process and their related practices in a way that allows a great control on management, development, review, prediction and estimation issues, and provides a clear view of the integration between software quality assurance components and their related testing activities to work as a single unit within the model to develop better software quality.
Section 3.1 introduces the model stages and its related activities. Section 3.2 introduces the basic sectors of the model. Section 3.3 introduces a closer view in the development stage and its related quality assurance components. Finally, Section 3.4 introduces mapping between OSAMA and RUP model.

3.1 OSAMA Stages

OSAMA comprised into three distinct interleaved stages; the Pre-Development, Development and Post-Development stage. These three stages have a number of internal phases.

3.1.1 Software Pre-Development stage assure that the project commitments have been clearly defined considering a set of activities related to the project vision, resources required, the schedule and budget, project risk handling, business requirement and various project management plans through initiation and the planning phases.

3.1.2 Software development stage decomposed into six distinct, often overlapped / interleaved phases; requirement, design (High Level Design and Low Level Design) ,code and unit test, Pre-Testing, Test Execution (Higher Order Test) and Post-testing phases. For each phase in the development stage the possible created artifacts have been introduced to be able to monitor, review, predict and measure the progress of work done through these phases and assuring the quality of the produced release.

3.1.3 Software Post-Development stage concerned with managing the implementation of the produced release in the operational environment through deployment and operation phase.

3.2 OSAMA Sectors

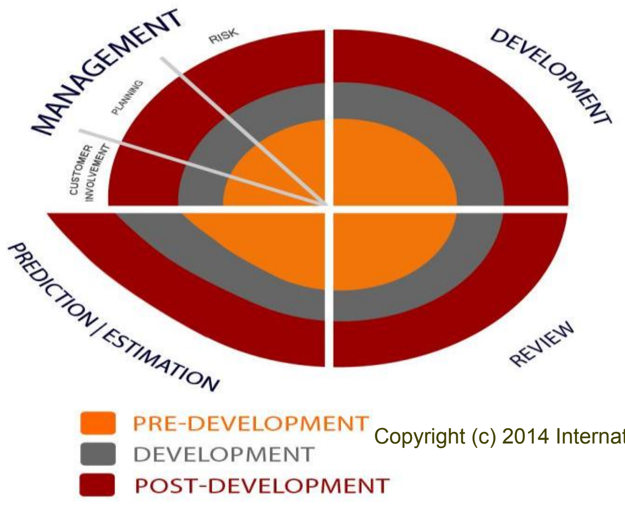
OSAMA is built as spiral form and divided into four sectors; Management, Development, Review and Prediction and / or Estimation sector. Fig.1 shows the basic sectors of OSAMA model.

Fig. 1 Basic sectors of OSAMA Model.

3.2.1 Management sector involves the development of project management plans (PMP) that provide a great control of the development activities and the introduction of managerial support actions that mainly prevent or minimize schedule and budget failures, continually changes in user requirements and identify and assess the risks resulting from the change one of the elements responsible for the software production through the various stages of the system.

3.2.2 Development sector includes a set of processes that concerned with developing the various deliverables of the release / product and managing their delivery to review sector for auditing and reviewing its quality and be ready for redeveloping or modifying its construction until it reaches the desired level of quality.

3.2.3 Review / Measurement Analysis sector includes a range of verification activities such as walkthrough, design review, peer review and inspection. It also locates the associated metrics for each software artifact that reflect the behav ior of the softw are create d



through the development stage.

3.2.4 Prediction/ Estimation sector includes the construction of a range of models designed to empower the software testing process for improving the quality of the product and so by focusing on reducing the effort required to determine the presence or eliminate errors in product or assess the readiness of the product for use.

During the prediction part, fault prediction models that predict either fault-prone modules or fault content of the software modules based on collected metrics has been established. The failure estimation part of this sector attempts to estimate the current level of reliability of the software while in execution and determines the readiness of the system to release. Thus, the number of failures becomes the goal of estimation instead of faults. Fig.2 shows the classification of the fault prediction and estimation models and their implementation techniques.

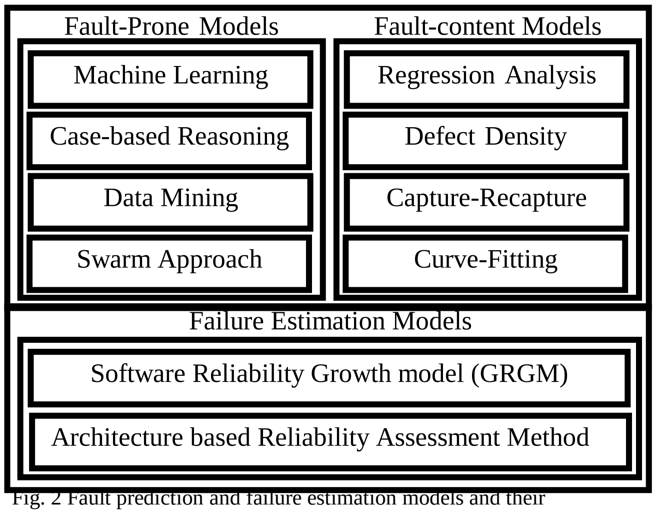


Fig. 2 Fault prediction and failure estimation models and their implementation techniques.

3.3 A closer view in the development stage

The core activities of the testing process have been incorporated as a sequence of practices with other development activities along/ through various phases of the development stage. Fig. 3 relates each development phase with its related software metrics and relates the test activities and QA practices along development stage.

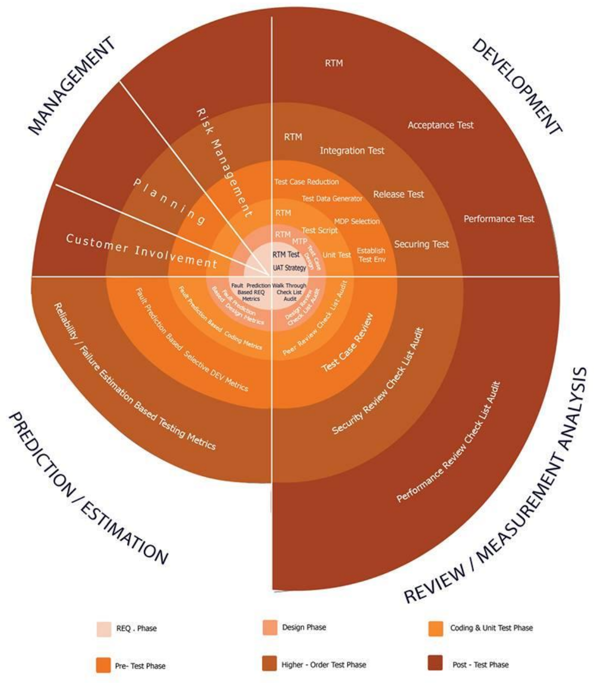

MANAGEMENT
DEVELOPMENT
PREDICTION / ESTIMATION
REVIEW / MEASUREMENT ANALYSIS
Risk Management
Planning
Customer Involvement
RTM
Acceptance Test
RTM
Integration Test
Test Case Reduction
Test Data Generator
Release Test
RTM
MDP Selection
Performance Test
Test Script
RTM
MTP
Test Case Design
Unit Test
Establish Test Env
Securing Test
RTM Test
UAT Strategy
Fault Prediction Based REQ Metrics
Walk Through Check List Audit
Fault Prediction Based Design Metrics
Design Review Check List Audit
Fault Prediction Based Coding Metrics
Peer Review Check List Audit
Fault Prediction Based Selective DEV Metrics
Test Case Review
Reliability / Failure Estimation Based Testing Metrics
Security Review Check List Audit
Performance Review Check List Audit
REQ . Phase
Design Phase
Coding & Unit Test Phase
Pre- Test Phase
Higher - Order Test Phase
Post - Test Phase

Fig. 3 Basic activities of the development stage

3.4 OSAMA via RUP Model

OSAMA built upon RUP (Rational Unified Process) methodology [19] that provides an iterative way for companies to envision create software programs. Table.1 illustrates the mapping between OSAMA and RUP model.

Table. 1 Mapping between OSAMA and RUP model

Features	**RUP**	**OSAMA**
Specification of all requirements at	Yes	Not all, and Frequently
Customer Involvement	High, after each iteration	High, after each iteration
Risk Involvement	Low	Low , Estimated
Phase overlapping	No	Yes
Framework Type	Iterative	Iterative and Incremental
Testing	During construction	Integrated
Documentation	Limited	Yes, but not much
Time Frame	Long	Moderate
Availability of working Software	At the end of the life cycle	At the end of every iteration
Project Scale	Large	Low to Medium
Primary Objective	High Assurance	High Assurance , Rapid Development
Release Cycle	Big band	In Phases
Stages	Inception, Elaboration, Construction, Transition	Pre-Development, Development, Post-Development

4. A Model Justification

This section justifies the idea of using the measurement analysis techniques as a basis of empowering testing process. These preliminary results were based on the regression analysis method to address the effects of various metrics derived from possible artifacts produced before executing the higher order testing. It also establishes the relationships between various software metrics as independent parameters and the probability of the module containing faults as an independent parameter.

4.1 Data Collection

The experiments were done on a data set CM1 from the NASA Metrics Data Program (MDP) repository [20], which is comprised of 38 features of C code for a NASA spacecraft instrument system. Table.2 illustrates the description of NASA CM1 MDP.

Table. 2 Details of the CM1 NASA MDP

Project	Description	Attributes	Cases	%Defective cases
CM1	NASA spacecraft instrument	38	344	10

4.2 Preliminary Results

Table. 3 illustrates the results of applying statistical Regression Analysis approach for establishing the predictive equations for the defective profile based on the contribution of the independent metrics (Loc Comments, Halstead Effort, Design Density, Num Unique Operators, Loc Code And Comment).

Table. 3 Regression Analysis for establishing the relationship between the the independent metrics and the defective profile.

Predictors	Metrics	Std. Error (S. R)	Beta	R^2	Const.
First Predictor	Loc Comments	0.001	0.308	0.095	1.066
Second Predictor	Loc Comments	0.001	0.455	0.120	1.060
	Halstead Effort	0.000	-0.216		
Third Predictor	Loc Comments	0.001	0.469	0.137	0.944
	Halstead Effort	0.000	-0.219		
	Design Density	0.061	0.133		
Fourth Predictor	Loc Comments	0.001	0.396	0.152	0.840
	Halstead Effort	0.000	-0.305		
	Design Density	0.061	0.133		
	Num Unique Operators	0.003	0.190		
Fifth Predictor	Loc Comments	0.001	0.393	0.164	0.835
	Halstead Effort	0.000	-0.238		
	Design Density	0.061	0.123		
	Num Unique Operators	0.003	0.244		
	Loc Code And Comment	0.002	-0.156		

As shown from table. 3 above, the first predictor metric (Loc Comments) contributes by 9.5 % in the interpretation of the total variance in the dependent variable, and this indicates that its effect size is medium. The predictive equation Eq. (1) can be formulated to predict the defective class by knowing the profile of Loc Comments as follows:

$$\textit{Profile of Defects} = 1.066 + (0.308 \times \textit{Loc Comments}) \quad (1)$$

The second predictor metrics (Loc Comments and Halstead Effort) contribute by 12 % in the interpretation of the total variance in the dependent variable, and this indicates that their effect size is medium. The predictive equation Eq. (2) can be formulated to predict the defective class by knowing the profile of Loc Comments and Halstead Effort as follows:

$$\textit{Profile of Defects} = 1.060 + (0.455 \times \textit{Loc Comments} - 0.216 \times \textit{Halstead Effort}) \quad (2)$$

The third predictor metrics (Loc Comments, Halstead Effort and Design Density) contribute by 13.7 % in the interpretation of the total variance in the dependent variable, and this indicates that its effect size is medium. The predictive equation Eq. (3) can be formulated to predict the defective class by knowing the profile of Loc Comments, Halstead Effort and Design Density as follows:

$$\textit{Profile of Defects} = 0.944 + (0.469 \times \textit{Loc Comments} - 0.219 \times \textit{Halstead Effort} + 0.133 \times \textit{Design Density}) \quad (3)$$

The fourth predictor metrics (Loc Comments, Halstead Effort, Design Density and Num Unique Operators) contribute by 15.2 % in the interpretation of the total variance in the dependent variable, and this indicates that its effect size is high. The predictive equation Eq. (4) can be formulated to predict the defective class by knowing the profile of Loc Comments, Halstead Effort, Design Density and Num Unique Operators as follows:

$$\textit{Profile of Defects} = 0.840 + (0.396\ x\ \textit{Loc Comments} - 0.305\ x\ \textit{Halstead Effort} + 0.133\ x\ \textit{Design Density} + 0.190\ x\ \textit{Num Unique Operators}) \quad (4)$$

The fifth predictor metrics (Loc Comments, Halstead Effort, Design Density, Num Unique Operators, and Loc Code And Comment) contribute by 16.4 % in the interpretation of the total variance in the dependent variable, and this indicates that its effect size is high. The predictive equation Eq. (5) can be formulated to predict the defective class by knowing the profile of Loc Comments, Halstead Effort, Design Density and Num Unique Operators as follows:

$$\textit{Profile of Defects} = 0.835 + (0.393\ x\ \textit{Loc Comments} - 0.238\ x\ \textit{Halstead Effort} + 0.123\ x\ \textit{Design Density} + 0.244\ x\ \textit{Num Unique Operators} - 0.156\ x\ \textit{Loc Code And Comment}) \quad (5)$$

Table. 4 illustrates the results of applying statistical Regression Analysis approach for addressing the effect of the independent metrics (Loc Comments, Halstead Effort, Design Density, Num Unique Operators, Loc Code And Comment) on the defective profile. It finds that the ability to predict the profile of the defective class by knowing the profile of the fifth variables is statistically significant at the level (0.001).

Table 4 The results of ANOVA for significance of regression coefficients

Predictors	Model	Sum of Square	Mean Square	F	Sig.
First Predictor	Regression	3.501	3.501	35.882	0.001
	Residual	33.371	0.098		
	Total	36.872			
Second Predictor	Regression	4.421	2.210	23.226	0.001
	Residual	32.451	0.095		
	Total	36.872			
Third Predictor	Regression	5.064	1.688	18.043	0.001
	Residual	31.808	0.094		
	Total	36.872			
Fourth Predictor	Regression	5.617	1.404	15.231	0.001
	Residual	31.255	0.092		
	Total	36.872			
Fifth Predictor	Regression	6.062	1.212	13.302	0.001
	Residual	30.810	0.091		
	Total	36.872			

4. Conclusions

This paper investigates the steps towards constructing a model that complements the existing models of software testing and quality assurance. It provides an integration model that relates the various activities of the testing process, static analysis activities, fault and failure prediction and estimation that fulfill the quality management issues related to development process. RUP model introduced, checked against OSAMA model, in such way that the mapping between the basic cycles of their structures and their activities has been established.
A series of tests were performed to experimentally justify our model. During the course of our experiments we endeavored to identify the significance of regression coefficients on the defective profile. The results showed that the using of measurement analysis as a verification technique provides a better chance to empower the software testing process and hence the software quality.

We aim to touch in our future studies the implementation of prediction and estimation sector of OSAMA using various intelligent approaches and verify their quality on real projects to provide a more realistic estimate of software fault prediction and its reliability.

5. References

[1] Daniel Galin, "Software Quality Assurance from theory to implementation", Press: Pearson, Addison-Wesley, 2004.
[2] P. Ranjeet Kumar, R. Ramesh, T.Venkat Narayana Rao, Shireesha Dara, "Software Quality Prediction: A Review and Current Trends", International Journal of Engineering And Computer Science (IJECS), Vol. 2, No. 4, 2013, pp. 1147-1155.
[3] Denis Kozlov, Jussi Koskinen and Markku Sakkinen," Fault-Proneness of Open Source Software: Exploring its Relations to Internal Software Quality and Maintenance Process", the Open Software Engineering Journal, 2013, 7, pp. 1-23.
[4] Hilda B. Klasky, "A study of Software Mertics", M.S. thesis, Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, New Jersey, 2003.
[5] Ayman Madi, Oussama Kassem Zein and Seifedine Kadry, "On the Improvement of Cyclomatic Complexity Metric", International Journal of Software Engineering and Its Applications, Vol. 7, No. 2, 2013, pp. 67-82.
[6] Saddam H. Ahmed, Taysir Hassan A. Soliman , and Adel A. Sewisy, " A Hybrid Metrics Suite for Evaluating Object-Oriented Design", International Journal of Software Engineering, Vol. 6, No. 1, 2013, pp. 65-82.
[7] Kaur, A.; Sandhu, P.S.; Bra, A.S., "Early Software Fault Prediction Using Real Time Defect Data", in 2nd International Conference on Machine Vision (ICMV), 2009, pp. 242 – 245.
[8] Kaur et al., " A Hybrid Metrics Suite for Evaluating Object-Oriented Design", International Journal of Software and Web Sciences (IJSWS), Vol. 3, No. 1, 2012, pp. 54-57.
[9] Rachna, R.; Navneet S.; Parneet, K.; Gurdev, S., "Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems", International Journal of Electronics & Communication Technology (IJECT), Vol. 3, No. 4, 2011, pp. 47-50.
[10] Yue Jiang, "Incremental Development and Cost-based Evaluation of Software Fault Prediction Models", Ph.D. thesis, Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, West Virginia, 2009.
[11] Nirvikar Katiyar, Raghuraj Singh, "Prediction of Number of Faults And Time To Remove Errors", International Journal of Computational Engineering Research, Vol. 3, No. 4, 2013, pp. 57-65.

[12] S.Kim, T.Zimmermann, E.J.W.Jr., and A.Zeller, "Predicting faults from cached history", in 29th International Conference on Software Engineering (ICSE), 2007, pp. 489-498.

[13] T.J.Ostrand, E.J.Weyuker, and R.M.Bell, "Predicting the location and number of faults in large software system", IEEE Transactions on Software Engineering, Vol. 31, No. 4, 2005, pp. 340-355.

[14] Anshu Basal, and Sudhir Pundir, " A Review on approaches and models proposed for software reliability Testing", International Journal of Computer & Communication Technology (IJCCT), Vol. 4, No. 2, 2013, pp. 7-9.

[15] Heena Kapila, Satwinder singh, "Analysis of CK Metrics to predict Software Fault-Proneness using Bayesian Inference", International Journal of Computer Applications (IJCA), Vol. 74, No. 2, 2013, pp. 1-4.

[16] Chayanika Sharma, Sangeeta Sabharwal, Ritu Sibal, " A Survey on Software Testing Techniques using Genetic Algorithm ", International Journal of Computer Science Issues (IJCSI), Vol. 10, No. 1, 2013, pp. 381-393.

[17] Hassan Najadat and Izzat Alsmadi, "Enhanced Rule Based Detection for Software Fault Prone Modules", International Journal of Software Engineering and its Applications, Vol. 6, No. 1, 2012, pp. 75-85.

[18] Rita G. Al gargoor, and , Nada N. Saleem, "Software Reliability Prediction Using Artificial Techniques", International Journal of Computer Science Issues (IJCSI), Vol. 10, No. 2, 2013, pp. 274-281.

[19] Radhika D. Amlani, "Comparison of different SDLC Models", International Journal of Computer Applications & Information Technology, Vol. 2, No. 1, 2013, pp. 1-8.

[20] http://nasa-softwaredefectdatasets.wikispaces.com/, Last Accessed March 2014.