ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 3, Issue 3, pp: (1-6), Month: September - December 2016, Available at: www.noveltyjournals.com

# REVISITED SOFTWARE ERROR CLASSIFICATION AND TYPES WITH ERROR TAXONOMY

Javed Ahmad Shaheen

Virtual University of Pakistan Lahore, Pakistan

***Abstract:*** **There have been a lot of Errors (defects) classification systems in industry to seek and establish common Error. We know Testing is the one of the important component of any software engineering process which results to find Errors (defects) in software. We must classify Errors and bugs so that the resulted classifications are little bit similar and also contain variations. Therefore there should be specific tool for testing. The tools are exist but none of which is accepted as a basic tool in software engineering projects upon which I emphasis to discover all errors (defects). In this paper I summarize the work on Errors (defects) classifications and its types and I also reexamine Errors (defects) taxonomy so far aims a set of challenges with the direction to a solution. It must be known that Systematic Error management based on bug tracking systems to find and resolve errors in software.**

***Keywords:*** **Errors (Defects), Testing, Errors (defects) Types, Errors (defects) Classification, Error (defects) Taxonomy.**

## I. INTRODUCTION

We want to discover errors in software, we use software testing. As Software testing, identifies Errors, and we must have to know that Error is any variance between actual and expected output. Software testing is that phase in which I are analyzing the software to identifies the errors so that the software become error free. It is impossible to produce Error free software products; however, the main purpose of any software engineering activity is to prevent Errors from being introduced in the first place. A Error is defined as An incorrect step, process, or data definition in a computer program. [Society, 1990, page 32]. There is confusion in the terminology used concerning the terms Errors, mistakes, defects and failures. The difference between the terms is explained by [Society,1990]: The terms Error and fault are same. Both imply a quality problem that is discovered after the software has been released to end users. Error is any flaw in the software system. When I test the Ib site or any Ib application and there is difference between expected results and the actual results, there is Error. Errors are divided in to 3 categories: Wrong, Missing and extra (table 1). There are many examples of Errors that can come in testing the Ibsite or any Ib application or any software: here I explain it with Ib application like:

**Table 1: Categories of Errors**

| **Category** | **Examples** |
|---|---|
| Wrong | User gives wrong/incomplete requirements for developing Ib application. Error in coding, in testing, Data entry errors also Mistakes in error correction or Analyst interprets requirements incorrectly |
| Missing | Incorrect design specifications or missed out some any specifications. |
| Extra | Developer done something extra in Ib application but client doesn’t require. Poor documentation. |

## II. LIFE CYCLE OF ERRORS

There are various Errors like new, open, review, rejected, test verified, not anError etc (fig 4). Error age or phase age is the important concept in testing that means later I find the Error the more it cost to fix it. Error spoilage is the concept which works on same concept that how late I find the Errors or bugs. When Errors are getting fixed during Error life cycle then Retesting and Regression testing is performed. Retesting is testing is performed to check that Error get fixed or not while in regression testing is performed to check that checked tests should not affect the unchecked tests.

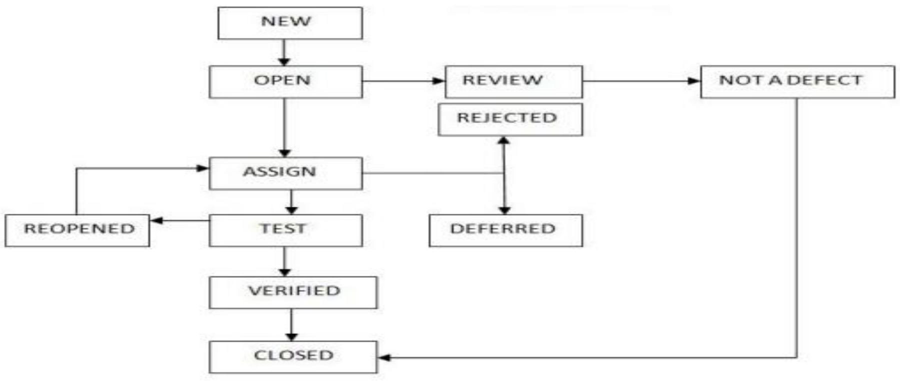

Fig. 4 States of defects

## III. HARSHNESS OF ERRORS

Severity (harshness) shows how bad the bug is and reflects its impact to the product and to the user. This changes from organization to organization or varies from projects to projects (table 2).

**Table 2: Severity of Errors**

| Severity | Description | Criteria |
|---|---|---|
| 1 | Very high | Inability to install/uninstall the product, product will not start, product hangs or operating system freezes, data corruption, product abnormally terminates etc so they are also called showstopper Errors. |
| 2 | High | Function is not working according to specifications, critical to customer etc that means application can continue with severe Errors. |
| 3 | Medium | Incorrect error message, incorrect data, etc means application continue with unexpected results. |
| 4 | low | Spelling, grammar mistakes etc that is Errors with these severities are suggestions given to client to make application better. |

## IV. PRIORITY OF ERRORS

Priority can be decided on the basis of how frequently the Error occurs i.e. probability of occurrence of Error (table 3).

**Table 3: Priority of Errors**

| Priority | Description | Criteria |
|---|---|---|
| 1 | Very high | Immediate fix, block further testing |
| 2 | High | Must fix before product is released |
| 3 | Medium | Should fix if time permits |
| 4 | Low | Would like fix but can be released as it is |

## V. LITERATURE REVIEW ABOUT ERRORS

In 2009, Chen and Huang performed an e-mail survey with several software projects, and presented the top 10 higher severity problem factors affecting software maintainability, as summarized in table I [2]. The authors has to indicate the following causes of software Errors [2]: a significant percentage of Errors is caused by incorrect specifications and translation of requirements, or incomplete ones [7-8]; half of the problems rooted in requirements are due to ambiguous, poorly written, unclear and incorrect requirements, the other half result of omitted requirements [9]. In 2003, Lutz and Mikulski analyzed the impact and causes of requirements Errors discovered in the testing phase, resulting from non documented changes or Errors in the requirements, and proposed guidelines to distinguish and respond to each situation [10]. Their work emphasizes the importance of requirements management.

**Table 4: Top 10 High Severity Factors:**

| **Sr#** | **development factors** | **Problem Dimension** |
|---|---|---|
| 1 | Inadequacy of source code comments | Programming Quality |
| 2 | Documentation obscure/untrustworthy | Documentation Quality |
| 3 | Changes not adequately documented | Documentation Quality |
| 4 | Lack of traceability | Documentation Quality |
| 5 | Lack of adherence to standards | Programming Quality |
| 6 | Lack of integrity/consistency | Documentation Quality |
| 7 | Continually changing requirements | System Requirements |
| 8 | Frequent turnover within the project team | Personnel Resources |
| 9 | Improper usage of techniques | Programming Quality |
| 10 | Lack of consideration for software quality requirements | System Requirements |

## VI. I PRESENT THE FOLLOWING SUBSECTIONS WORK THAT IS RELATED WITH OR INCLUDES A REQUIREMENTS ERRORS CLASSIFICATION.

### *VI. A Code Errors Classifications, 1992:*

The Error types used are: function, interface, checking, assignment, timing/ serialization, build / package / merge, documentation and algorithm. For each Error it is necessary to indicate if the feature is incorrect or missing [11]. Such classifiers do not seem completely adequate to classify requirements. Errors and documentation is basic to give further information on the Error. The Hewlett-Packard (HP) [12] categorizes the Errors by mode, type and origin, (see figure 1) [6]. From the types of Errors with origin in the requirements /specifications phase, the requirements/ specifications seems to be vague and the interfaces ones are too detailed and more adequate to design specification Errors.

### *VI. B. Quality Based Classifiers, 1976 – 2010:*

This section presents the work of several authors applied quality based requirements Errors. In 1976, Bell and Thayer did a research to verify the impact of software requirements Errors. They resulted that software systems meeting Error requirements will not effectively solve basic needs [13]. They aggregated the Errors in categories, as presented in table 3. In 1989, Ackerman *etal* analyzed the effectiveness of software inspections as a verification process [15]. They presented a sample requirements checklist to use in inspections of requirements documents, containing questions organized by Error categories: completeness, consistency and ambiguity.

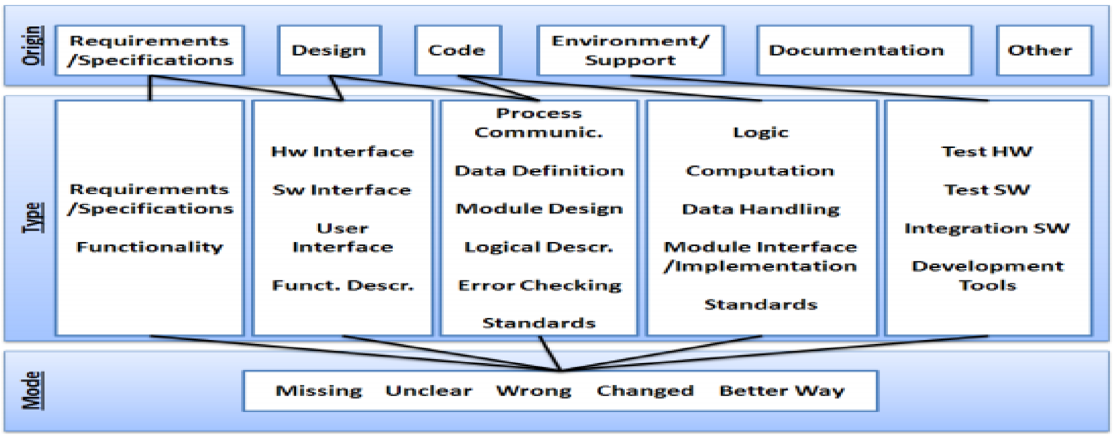

**Figure 1: HP Errors classification scheme [6].**

In 2003, Hayes designed requirements fault taxonomy for NASA's critical/catastrophic high-risk systems. Hayes stated that ODC refers to design and code while their approach emphasized requirements, so they adapted the Nuclear Regulatory Commission (NRC) requirement fault taxonomy from NUREG/CR-6316 (1995). [17] Afterwards, in 2006, Hayes *et al* analyzed a software product related with the previous to build a common cause tree [18]. In both works unachievable was reserved for future.

***VI.C. Functional and Quality Based Classifiers, 1992 – 2009:***

In this section I have to present Error classification taxonomies that are functional and quality based. In our research paper I consider that the functional classifiers represent the function of the requirement in the product. In 1992, Schneider identified two classes of requirements Errors that is missing Information and Wrong Information (table 3). They performed an experiment where two Software Requirements Specification (SRS) documents Ire inspected with a combination of adhoc, checklist and scenario inspection methods. The checklist was organized in categories, resembling a Error classification: omission and commission. The scenarios also included categories: data type consistency, incorrect functionality, ambiguity, and missing functionality. The authors concluded from their results that the scenario inspection method was the most effective for requirements [17]. Later, in 2007, Walia repeated an experiment to show the importance of requirements Errors taxonomy. They involved software engineering students in a SRS document review using an Error checklist. The students repeated the review and after being trained in the error abstraction process and do experiment which shoId that error abstraction leads to more Errors found without losses of efficiency and the abstraction is harder when people are not involved in the elaboration of the SRS and have no contact with developers. Requirements Errors Ire classified as: general, missing functionality, missing performance, missing interface, missing environment, ambiguous information, inconsistent information, incorrect or extra functionality, wrong section, and other faults [12].

## VII. ERROR TAXONOMIES FOR SOFTWARE REQUIREMENT

Error taxonomy accumulates and arranges the domain knowledge, project experience of experts and a valuable instrument for requirements based testing for different issues and reasons. It is systematic backup for the design of tests, support decisions for the allocation of testing resources, improve the review of requirements and offer a suitable basis for measuring the product quality. In this paper, I review a method of requirements based testing with Error taxonomies. I point out how Error taxonomies can be integrated into a standard test process and discuss results and lessons learned with reference to industrial projects from a public health insurance institution where this approach has been successfully applied. Error taxonomy is a system based on hierarchical categories designed. It is useful aid for reproducibly classifying faults and failures [2]. It is concerned with eradicating the prejudice of classifier. It is also creating distinct categories with the purpose of effective measuring and testing, Error management and product quality. Beizer has defined the generic Error taxonomy widely used in software testing [5]. It consists of nine categories:(1) requirements (2) features and functionality (3) structural Errors (4) data (5) implementation and coding (6) integration (7) system, software architecture (8) test definition and execution as Ill as (9) unclassified Errors. The Error taxonomy of Project A is based on the Beizer taxonomy because of its suitability for system testing and the experience with it available in the development organization. It has three levels of abstraction. The high level categories are mapped to product specific categories which are then further refined to concrete low level Error categories with an assigned identifier and severity. The possible values for the severity of the Error categories as Ill as of the failures follow Bugzilla [1], and may be blocker, critical, major, normal, minor, or trivial.When creating Error taxonomies, Error data from completed projects and the feedback from affected roles such as developers or testers should be taken into account.

## VIII. CONCLUSIONS AND FUTURE WORK

If I elaborate the software testing, I can find that software performed as what it is supposed to do and software not performing what it is not supposed to do. No errors identification without software testing and I cannot find the Errors as soon as possible and not to get them fixed which means it is not all about to correct the code but to search the bugs or find the errors in code. Therefore it is necessary for testing process that the testers must write accurately test cases so that it can find out hidden Errors from the programs. An Error taxonomy is created as, it supports the specific analysis interests of the organization which uses it, namely in the implementation of Error causal analysis [11]. In our work based on a literature review I assembled a classification for Error types in requirements specifications, following the recommendations in [6]. Such classification is important to support the analysis of root causes of Errors and their resolution, to create checklists that improve requirements reviews and to prevent risks resulting from requirements Errors. I evaluated our classification scheme through two experiments where students had to classify Errors identified in a SRS document. I concluded that after refining the classification list, different people may classify the same Error in a different way so by choosing a classification for requirements Errors organizations it is need to be aware of the problems of using them. People may interpret the classifiers differently and performing retrospective analysis of Errors which is simply based on the type of Errors, it might be misleading. Experiments similar to the ones discussed in this paper may be adopted to get the degree of consensus betIen their personnel. As future research work I propose to enhance the classifier and tester and perform modified experiments “on the job” using individuals from industry; (2) by using a SRS document from a project they are involved in; (3) having each individual conduct a complete SRS review to detect and subsequently classify Errors. I expect that the classification difficulties will be attenuated in this setting, leading to more accurate and unanimous classifications. I will also use the Errors classification to create a checklist to be used in the requirements inspections, and will conduct experiments with a control group not using the checklist to assess their impact on the review efficacy, efficiency and convergence.

## REFERENCES

[1] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," ed: IEEE Press, 1990.

[2] J.-C. Chen and S.-J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability," Journal of Systems and Software, vol. 82, pp. 981-992 June 2009.

[3] M. Hamill and G.-P. Katerina, "Common Trends in Software Fault andFailure Data," IEEE Trans. Softw. Eng., vol. 35, pp. 484-496, 2009.

[4] D. N. Card, "Learning from Our Mistakes with Error Causal Analysis," IEEE Softw., vol. 15, pp. 56-63, 1998.

[5] K. Henningsson and C. Wohlin, "Assuring Fault Classification Agreement - An Empirical Evaluation," presented at the International Symposium on Empirical Software Engineering, Redondo Beach, California, 2004.

[6] B. Freimut, et al., "An Industrial Case Study of Implementing and Validating Error Classification for Process Improvement and Quality Management," presented at the Proceedings of the 11th IEEE International Software Metrics Symposium, 2005.

[7] L. Apfelbaum and J. Doyle, "Model based testing," presented at the 10th International Software Quality Iek Conference, San Francisco, 1997.

[8] *Software Risk Abatement*, AFCS/AFLC Pamphlet 800-45, U.S. Air Force, September 30, 1988.

[9] Arthur, L. J., “Quantum Improvements in Software System Quality,” *CACM*, vol. 40, no. 6,June 1997, pp. 47–52.

[10] J.H. Hayes, et al., "A metrics-based software maintenance effort model," *Proc. Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04)*, IEEE Computer Society, 2004, pp. 254.

[11] .A. De Lucia, et al., "Assessing effort estimation models for corrective maintenance through empirical studies," *Information and Software Technology*, vol. 47, no. 1, 2005, pp. 3-15.

[12] N. Gorla, et al., "Debugging effort estimation using software Metrics," *IEEE Transactions on Software Engineering*, vol. 16, no. 2, 1990, pp. 223-231.

[13] .M. Xie and B. Yang, "A Study of the effect of imperfect debugging on software development cost," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, 2003, pp. 471-473.

[14] .M. Jørgensen and D.I.K. Sjøberg, "Impact of experience on maintenance skills," *Journal of Software Maintenance*, vol. 14, no. 2, 2002, pp. 123-146.

[15] B.S. Rao and N.L. Sarda, "Effort drivers in maintenance outsourcing - an experiment using taguchi's methodology," *Proc.Seventh European Conference on Software Maintenance and Reengineering*, IEEE Computer Society, 2003,

[16] Software testing best practices,Ram Chillarege,IBM Research-Technical report RC 21457

[17] Software Testing Techniques- Shivkumar Hasmukhrai Trivedi [B.com, M.Sc I.T (Information Technology), P.G.D.B.M –Pursuing] Senior System Administrator, S.E.C.C [Socio Economic and Cast Census] Central Govt. Project – Bhavnagar [Gujarat – India], Volume 2, Issue 10, October 2012 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering