A Proposed Defect Tracking Model for Classifying the Inserted Defect Reports to Enhance Software Quality Control

Torky Sultan
Information Systems Department
Faculty of computers and Information, Helwan University
Cairo, Egypt

Ayman E.Khedr
Information Systems Department
Faculty of computers and Information, Helwan University
Cairo, Egypt

Mostafa Sayed
Information Systems Department
Council state
Cairo, Egypt

ABSTRACT
Defect tracking systems play an important role in the software development organizations as they can store historical information about defects. There are many research in defect tracking models and systems to enhance their capabilities to be more specifically tracking, and were adopted with new technology. Furthermore, there are different studies in classifying bugs in a step by step method to have clear perception and applicable method in detecting such bugs. Besides, this paper on the other hand, shows a new proposed defect tracking model for the purpose of classifying the inserted defects reports in a step by step method for more enhancement of the software quality.

Keywords
Bugs; defects; bug tracking systems; defect tracking models; software quality.

1. INTRODUCTION
In many software development organizations, bug tracking systems play an important role as they allow different types of users communicating with each other (i.e. developers; testers and customers) to assure that they have the same perception about problems or requesting new features. In addition, bug tracking systems can keep track of more historical information stored of the bugs; and also software requirements requests to learn from the previous bugs through the maintenance or the development process of the software systems. Earlier attempts were made for understanding the way of filtering and classifying inputting data for bugs with a structured way for easy use in tracking defects later [5]; [10]; [19]; [20] and [30]. However, most bug tracking systems are far from perfect [17].

This paper provides new proposed defects tracking model concentrating on the factors for the insertion of defects reports through tracking tools. In addition, it provides an overview of the literature on defects tracking systems and its relation with software quality from different perspectives (section 2). The Paper starts with a theoretical overview of different aspects of defects tracking models and their components. Besides, it illustrates the shortcomings of these models (section 3). The existing attempts to improve the defects tracking systems are highlighted in our synthesizing framework (section 4). The paper ends with a conceptual model for defect tracking system (section 5), followed by section summary of (section 6).

2. Defect Tracking Systems and their Relations with the Software Quality
This section aims at providing a detailed discussion of the background overviews about defects tracking systems. It deals with the importance of the defects tracking systems and their relations with the Software Quality Control; also it explores the different views of the defects tracking models and systems.

Software Quality have a different aspects that influence the software development process such as: quality control, quality improvement, and quality assurance. The Institute of Electrical and Electronics Engineers (IEEE), defined software quality as "the degree to which a system, component or process meets specified requirements and customer (user) needs (expectations) "[13]. Moreover, Quality Control was defined as "A process in which the product is examined and evaluated against the original requirements expressed by the customer. The detected problems are then corrected" [16].

There are many software tools that play an important role in tracking defects of software and which are called "Defects Tracking Systems". Jalbert defined them as "Allow users to report, describe, track, classify and comment on bugs reports and feature requests"[14].

Defects Tracking Systems can be separated systems that can integrate, and contribute in software development process. They can keep, with details of defects reports and information associated with resolving it, in a database storage. Lethbridge, Singer and Forward indicated that developers view the defects tracking systems as important repositories of historical information [20]. Furthermore, software defects data is an important source to the organizations for the software process improvement decisions and that "ignoring defected data, can lead to serious consequences for an organizational business" [11]. In addition "they may be part of an integrated suite of configuration management tools, where the status of the defect may act as a trigger or key for other events within the system" [1].

There is no doubt that software quality which is used in detecting defects, is one of the important factors for evaluating the software process development. Weinberg (1983) documented that an error costing a company 1.6 billion dollars, and was the result of changing a single character in a line of code [30]. Also, Curhan mentioned that "some types of defects have a much higher costs to fix due to the customer

impact and the time needed to fix them, or the wide distribution of the software in which they are embedded " [5].

3. The Different Aspects of the Defects Tracking Models and their Shortcomings

Large number of software companies use Software Tracking Tools to achieve the goals of the Configuration Management. Janák defined configuration management as "the process of controlling and documenting changes to a developing system" [15]. Also, software tracking tools help quality control engineers to accomplish their jobs as good as possible to discover, and to prevent the occurrence of bugs by tracking them.

The Software Tracking Tools are simply built based on defects tracking models. Figure (1) below represents a simple defects tracking model. Edwards and Steinke (2006) simply discussed the defects tracking model, as they divided it into the following two stages: ((repair /resolution)-(verification)) and the following three changes of status: (discovery – resolved – closed) [6].

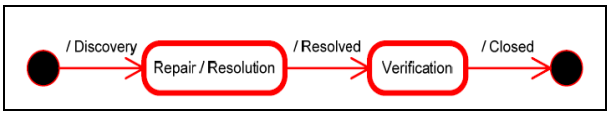

Figure 1: The two stages of the Defect Tracking Model [6]

Microsoft Team Systems used a four-stage defects tracking model for Capability Maturity Model Integration (CMMI); the model expanded and evolved the "open" stage into the following two stages: "proposed" and "active" stage. Although the model enhanced the three- stage defects tracking model, it still works as a framework describing the status and phases of bugs that should followed. The three statuses (deferred – rejected – duplicate) duplicated through two positions, the proposed stage and the active stage [23]. There were no remarks about how to examine and register the bugs.

Edwards et al. (2006), proposed the Full Product Life Cycle Defect (FPLC) Model, which was an extension of IBM/Rational Model with changes to include the test and project management interfaces. The model discussed in details, the five statuses of the defects tracking model which are: Submitted, Open, Postponed, Resolved and Closed. Although the model mentioned perfectly the duplication problem of defects; it still has some remarkable scope for more enhancements.

The research dealt with the status "reject" as not a closed status. It coped with it as a circulating process where it should be a “Closed” status. Also another remarkable note about the postponed defect, Edwards et al. (2006), reported that "Placing any defect in a Postponed status is a tacit admission that it should be repaired, but at a later time." [6] which means that it has the priority to be repaired not to be a closed.

One of the famous defects tracking tools used by quality control engineers is Bugzilla defects tracking system. The work flow of the model showed that it classified the new bugs into the following two categories: the first one comes from a user with a confirmation right, and the second comes from any user but it will not be confirmed till it has enough votes. Also, it concentrated on quality control engineer roles in checking the appropriate solution which being satisfied, verified, closed or didn’t conform with the solution [15].

Although the default IBM Rational Clear Quest Ticket mentioned the workflow path that defect process has taken, and which "Starts when the defect is discovered and ends when the defect is resolved, hopefully repaired, for the most immediate release of the software application" [15]. It still has some shortcomings as the "rejected" status could be at any state. It may be after investigation, the approved state or after the task opened and in all the cases, it should be closed. Also the approved status should be one of the roles of quality control engineer; who should check it as the defect may not exists only in a new project process, but also may exists at the maintenance process.

4. Synthesis Model for the Classification of the Bugs

The last section discussed the different overviews of the defects tracking systems. Their workflow models, the status and paths of the defects through the process of discovering the defect. Also, it mentioned the literature reviews of different research at the same point that dealing with defects in their overall aspects. The proposed model concerned with the following two points: First the different classifications of the bugs; and the second is the tracking history of the defect that was discovered. The two points will be discussed in more details in the next paragraphs.

As mentioned in the last paragraph; this section covers discussing the proposed model and how it makes filtration and classification of the bugs. However, a bug in its default way, is discovered where an action or value is not correct or not being achieved as it decided or going in an unexpected way. We divided the classification and track of the bugs into the following Phases: (Submission – Examination – Registration – Tracking) which interact with each other and will be discussed in more details later.

4.1 The Submission Phase:

The first phase of our defects tracking model will help us in understanding the filtration process of the bugs and the issues in the submission phase.

The role of this model in tracking bugs, starts after discovering an incorrect action or value to the system that shouldn’t happen at this point. Describing and stating the problem is a significant component for retrieving a suitable solution. Stimson (1998) mentioned that "A problem well-stated is half-solved" [27], this push us to define of who discovered the bug, and when it is discovered.

There are two different groups of users who can discover the presence bugs. The first Group is: "The Normal User” who deals with the system after released to him to achieve a specific function or goal. The second Group is: "The Authorized User" who participates at any phase of the development process. He may be one of triage team or development engineering team or may be one of technical team leaders. The bug is usually discovered in the following two positions: the first position is before releasing the product, and through the development process. The second position is after releasing the product to users.

Section (3), discussed a number of different defects tracking models; where there were a number of these models which coped with "the submission phase" as the first step of filtration and classification defect reports. The adapted one with our model was "Bugzilla tickets workflow". We will modify it to be more compatible with the proposed model.

Bugzilla Workflow Model, classified the detectors of the bugs into two categories:-

1- Anyone who has enough votes.
2- A user with confirmation rights.

According to the classification of users in Bugzilla Workflow Model, we will classify the users into three categories:-

1- Authorized user with confirmation rights.
2- Trusted User.
3- Normal User.

The first Category: (Authorized User) who is discovering the bugs inside the location of the development process. The authorized user may be one of the quality control engineers i.e. developing engineer or may be anyone who has the ability to discover a bug. The second category: (Trusted User), who can be defined as the user who has the ability, and good experience for dealing with the product or system; also has a recorded history of detecting bugs. The third category: (Normal User) who has the ability but little experience for dealing with the product or system, and has short recorded history of detecting bugs. That is, the Normal User has the ability to inform the presence of a bug but hasn't the priority element without a confirmation of an "Authorized User".

4.2 The Examination Phase:

The examination phase begins after the end of the submission phase. In this phase, the outside or inside user who participates in the development process of the product decides that there is a vision for a defect.

This phase has its own priority as Hooimeijer and Weimer (2007) documented that "Bug report triage and evaluation are the significant part of modern software engineering for many large projects" [11]. It is the first phase of preventing the distortion of recording un-wanted data or duplication announcement of bugs by checking the database for recorded bugs before and through each time of recording a newly discovered problem.

According to the work and efforts made by Mays, Jones, Holloway and Studinski, at IBM 1990 for defects prevention. They analyzed the faults that appeared in order to understand them using casual analysis. In addition, detecting the way of prevents defects from appearing in the future.

They showed the role and importance of the action team whose responsibility was to detect and store the appeared faults in the database and make a checklist to be updated with the new faults. Also the important role of "triage team", which has the ability to discover and deal with the defects mentioned by Black (1999), who assured that the triage team can review, evaluate the defects and assigning them to the development team [3]. For more information in details about triage team see [21].

When the Bug examination process is done, it is followed by rules and strategy of checking tests through quality control engineers. Furthermore, bug examination, is the last phase of deciding whether either the bug was recorded before with a suitable solution, or it will be a new classified bug.

The last statement lead us into the following three states after having the bug's examination recorded history such as:-

1- Bug not found and not registered before.
2- Bug found and resolved before.
3- Bug found with the same condition and need to be in (reopened state).

The first point will be discussed in more details, in the next section as it will be the default path even if it achieved the two conditions: "not found" and "not registered before". The second and the third point, are the core elements in the examination phase as there in no need to move to the next phase of registering an already existing bugs. The second point is that such a bug already exist and has a suitable solution for it; so there is no need to fill the database with such unneeded data, but what will happen if the bug is re-iterated with the same condition of the test case scenario?.

The last question leads to the third point that the bug has recorded actions before closed. This point was achieved by following different test case scenarios, and confirmation that there was a bug with the same conditions registered before. Therefore, a "reopen state" can be released by an authorized user in the examination phase in order to prevent duplication defect reports.

4.3 The Registration Phase:

The registration phase follows the examination phase. It is an important phase for retrieving useful information in the future because there are different factors for classifying defects reports which were registered in this phase. The next paragraphs, will discuss different classification of defects schemes.

Although there are large number of research on classification of defects schemes, they faced a number of problems including "Ambiguous, Overlapping, and Incomplete Categories, Too many categories, and Confusion of error causes, Fault Symptoms, and Actual Faults" [23].

One of the first work for classifying defects, was made by Endres in 1973 of IBM. His work was about gathering data covering 432 defects detected during the Test Phase of Development of DOS Operating System. He classified the defects into six general categories including: Machine Error, User Error, and Documentation Error. Also, he classified each defect by 'type'. But this type of classification scheme was very complex, dividing each defect into groups including: Dynamic Behavior, Communication between Processes, and (Counting and Calculating) among many others [7].

According to Basili, and Perricone's categories, the error classified as one of the following Categories: Requirements incorrect or misinterpreted, Functional Specification incorrect or misinterpreted, a Design Error which spans several modules, a design error or an implementation error in a single module, misunderstanding of the external environment, error in the use of programming language or compiler, clerical error, and error due to previous wrong correction of an error [2].

Another work was made by Sullivan and Chillarege (1992) to analyze the different error classification of software systems. They made their work based on software defects reported at customer sites in two large IBM database management products, DB2 and IMS.

They compared the errors type; defects type and error triggers classifications. They defined error type as "the low level programming mistake that led to the software failure" [28]. Also they defined defect type as" a higher level classification system that distinguishes between design mistakes, coding mistakes, timing mistakes, and administrative mistakes"[28]. The error trigger classification was meant to give insight into the software testing process.

Fredericks and Basili (1998) made analysis to find defects and how organizations dealt with such defects. They focused on achieving three goals that can be defined as significance factors of building a new defect tracking models. These three

goals are: Detecting the Nature of Defects; Detecting the Location of Defects, and when the Defects are Inserted.

In the early 1990's, IBM developed two new Technologies using defects data. The First Technology: "Defect Prevention", which involves development teams contributing to a knowledge database containing: common defects, how they can be prevented, and how to easily detect them. The Second Technology: "Orthogonal Defect Classification", which involves using statistical methods to predict the phase in which a defect originated, rather than relying on the subjective evaluation of an engineer [8].

The Defect Tracking Model had evolved at the end of the nineties; the Defects Classification Scheme shown in Figure (2) was used.

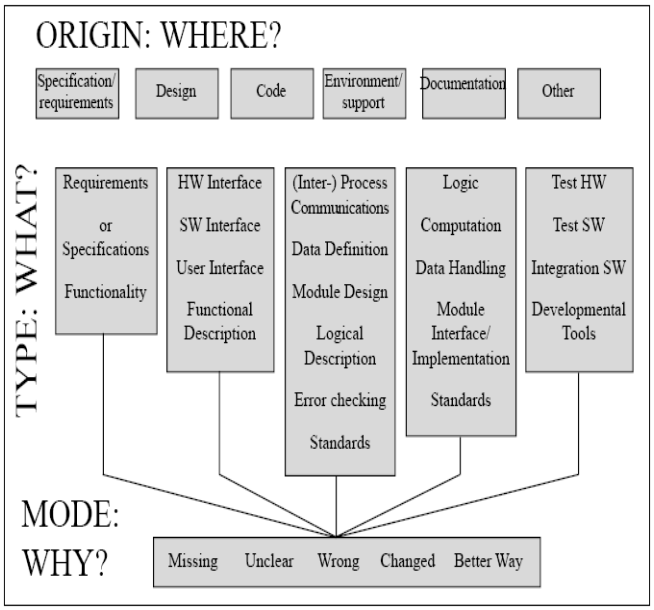

Figure 2: Hewlett-Packard Defect Categorization Scheme [25]

The last Categorization Scheme mentioned three Elements of defects classification. The First Element was the Location where bug is discovered through the development process. The Second Element was the type of defect where may be classified through each phase of the development process has its own kind of defects. The Third Element was the value of each defect which can be measured.

According to the work and efforts made by Mays et al., at IBM 1990 on the subject of defects prevention where they concentrated on how to prevent defects from appearing in the future. They analyzed the faults that appeared and to understand them using casual analysis. We divided this work into two points the first was the way they used to classify the faults, and the second was the flowchart that was used by the engineers [8].

Concentrating on the first point; Mays (1990) classified the errors that appeared into four categories as summarized in table1.

According to Rus (2002), there is a defect classifying schema that was developed and used by IBM called "Orthogonal Defect Classification". He defined it as "A measurement concept for software development that uses the defect stream as a source of information on the product and the development process "[26]. He divided it into two classes of defect attributes: the First Class associated with the defect discovery and contained the elements: (activity, trigger, and impact). the Second Class associated with the removal of the defect and contained the elements: (target, defect type, qualifier, source, and age).

Table 1: Defects Classification Scheme [8]

	Category	What it means
1	Oversight	The developer did not completely consider some details of the problem. For example, a case might have been missed or a boundary condition not handled properly.
2	Education	The developer did not understand the process or a portion of the assigned task, Presumably because of lack of training in that area.
3	Communications Failure	Information was not received, or information was communicated incorrectly and thus not understood properly.
4	Transcription Error	A typographical error or other mistake which was made by the developer. The developer fully understood the process to be applied, but made a mistake.

With the last different views of the classification schemes of defects, it appeared that there were a number of factors that describe the defects, and these factors are so important. We will concentrate on the following two Factors that are seen on the degree of importance from quality control perspective.

1-The First Element is "Bug Location":-

To describe "Bug Location", we have to mention what is the meaning of location of the bug. 'Location' is described in reality with how we can arrive to it. The locations of the bugs are determined by, in which stage the bug appeared or discovered; and in which place in the system or the application it appeared. Fry and Weimer (2010) defined fault localization as: "Is the task of determining if a program or code fragment contains a defect, and if so, locating exactly where that defect resides" [9].

As we attempt to enhance the quality control in the software, we have to recognize different phases of the software development such as: Analysis, Design, Testing and Maintenance. At the research context, we will concentrate only on the following two phases: (Maintenance and Testing).

Furthermore, another element of describing the location of bugs is to describe where the bug was discovered through the system, more specifically it is about detecting the location of the bug that appeared in the system such as the input screen, output screen, etc.

Also we have to describe the surrounding environment of the system as an element in classifying the location of the bugs

such the version of the system, the kind of operating system that the system works under.

2-The Second Element is "Bug Type":-

The 'Bug Type' varies from one system to another because the different tools which were used to create such systems, have their own limitations and shortcomings according to the study made by Ko et al., [18]. However, we have to put a dynamic framework for defining the bug type according to the various tools used to build the systems; with respect to the major general bug type. In other words, the bug type contains the general bug types that appear in different systems such as: "interface error" and "wrong result". Moreover, special bug types appear according to the system development tool used such as: "human interpreter problem" and "application terminated" that appear in systems made by the Power Builder Tool.

Sullivan et al., 1992 Classified the Software Defects Type as: function defect, data structure/algorithm defect, assignment/checking defect, interface defect, timing/synchronization defect, and build/package/merge defect [28]. Table (2) Shows the major different proposed defects types that may appear in any system or application.

Table 2: The Proposed Defects Classification Scheme

	Bug Type	Meaning
1	Interface Errors	Errors that visually appear in the user interface which are un-familiar in the general framework.
2	Calculation Errors	Errors that appear in such areas that had some calculations done before. Such errors are in total summation or sub-total results.
3	Loading Errors	Errors that appears in loading data or application that take much more normal time in response
4	Security Errors	Errors that appear in the general imbalance in privileges and rules for each user.
5	Documentation Errors.	Errors that appear based on the various variations between different systems and documentations or between the documentation's versions itself.
6	Enhancement Errors.	Enhancement errors appear when there is a finished task or function and there is a need to enhance in performance or response.
7	Business Logic Bugs.	These bugs that appear when a rule or business logic is Inconsistent with another one.

4.4 The Tracking Phase:

This phase is concerned with the traceability of the defects that were recorded in the system before. Traceability means to have a historic record of what had been done for each defect from initial state to the closed state with respect to the different states of development process between the last two states.

There were a number of scenarios expected from the proposed model to achieve. The first Scenario is the traditional scenario which is searching a new defect which is then classified and will be on the first status which is "Initial Phase".

At this phase, the defect start and fixing dates are stored together with the name of the person from the development team who will fix the defect. At this stage, the development engineer who is responsible for fixing such defect starts to work under the status: "under development" with a new date recorded of the beginning of the development process. Having finished the development process of the defect, its status changes to "development completed" with respect to recording the date of finishing this process.

With small calculations of dates that recorded between "Under Development" and "Development Completed", we can measure how much time it took the development team to fix this defect. Hooimeijer et al., documented that Bugs fixing is a time-consuming process, with half of all the fixed defects in Mozilla requiring over 29 days from start to finish date [11].

After the status "Development Completed" is finished, the Testing Phase becomes more accurately a Regression Test Phase is going to achieve.

The role of quality control engineers appears here by making the regression test with test cases; and scenarios to realize that the defects are really fixed at the development process phase. When finishing all test cases and scenarios for the defects, the quality control engineers release the status "Test Complete" which means that the defects were fixed and the product has become ready for release only for discovered defects.

The status "Test Complete" differs from the status "Closed" in that it reflects the end of the regression test phase, but the later reflects that the "user acceptance test" is finished.

The last scenario showed different status which defect moves through it, concerning the time element that recoded before and recording every change on defect status, as mentioned by Tammana and Faught (1998) "A defect tracking system that lets the user query the defect database is useful not only to generate summary reports, but also to track the status and the progress of a project that's underway" [29]. Therefore, through the power of DBMS (data base management system), we can achieve a good tracking of defects through this last scenario.

The Following Table, Abbreviated the different status of tracking scenarios that the defects take through different phases of the product development phases, its meaning and whose responsibility to check.

Table 3: Proposed Defect status Scheme

	Status	Meaning	Responsibility
1	Initial	Declaration status for Fixing the Defect.	Quality Team
2	Under Development	Declaration status for starting Development on Defect.	Development Team
3	Development Complete	Declaration status for finishing development on defect	Development team leader
4	Under Test	Declaration status for starting test	Quality Team
5	Test	Declaration status	Quality Team

	Complete	for finishing test	Leader
6	User Acceptance Test Complete	Declaration status for user acceptance that the defect fixed	Users and Quality team leader
7	Closed	Declaration status for finishing all process needed for fixing defect	Project Manager
8	Need more Details	Declaration status for misunderstanding the requirement or function needed	Development Team
9	Postponed	Declaration status for postponing task for a time or next versions	Project Manager Development Team Leader
10	Refused	Declaration status for Refusing the Task for unacceptable requirement or function needed.	Development Team leader
11	Reopen	Declaration status for reopening a defect with a same condition of a recorded defect before	Quality team leader

5. Conceptual Framework Design for the Proposed Defects Tracking Model

Based on the previous discussions in sections 1, 2&3 and that derived from the development of the research synthesis model for different ways of defect classifications that were presented in the preceding section, the ultimate conceptual model is given in Figure (3). The present research adopts the following model for classifying defects that appear in maintenance and testing phases of the software development process. The present research adopts the following model for classifying bugs which appear through different development phases especially in the maintenance and testing phases. As mentioned by Boehm and Basili, the maintenance phase consumes over 70% of the total life cycle cost of the software development projects [4]. Also, the proposed model which tends to evaluate the tracking system, gives real historical information about bugs recorded. The model developed was based on the previous work of [6], [15] and on our general synthesized model for classifying and tracking defects (section 3 & 4). It is quite suitable for a case study. This model will guide us through our exploration for classification of defects to enhance quality control for the software development and maintenance processes.

Our model will focus, in details, on the phases of preventing defects and classifying them. The conceptual framework for classifying and tracking defects as shown in Figure (3), shows that the tracking system gives real historical information for maintaining the bugs through the development life cycle. Moreover, it concentrates on the bug examination, location and type factors for the insertion process of defect reports. Due to the highly exploratory nature of this study, all isolated conceptual variables/factors only represent the initial ideas about the discussion of defects tracking phases and classification method to deal with in the future.

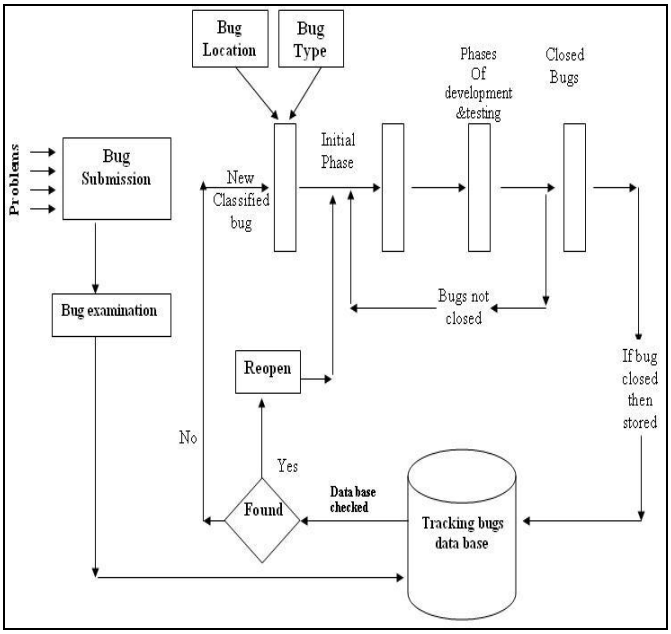

Figure 3: Proposed Defect Tracking Model

6. Summary

This Paper described terms and findings from significant earlier research, thereby forming a conceptual context and foundation for the exploratory observational study that is central to this research. The first part was a discussion of different perspectives of what translated a subtle relation between, on the one hand, defect tracking systems and its relation with the software quality and, on the other hand, different classifications of defects with phases of their tracking and shortcomings of these models.

The last section described the various models of the defect classifications coupled with the illustration of these models shortcomings. It was argued that the previously discussed models concentrated on the tracking phases of defects without caring of the classification factors of the defects. Furthermore, there were different directions of preventing defects and classification of errors and bugs in various languages and contexts, in chronological order.

7. REFERENCES

[1] (Avram, 2007) Gabriela Avram, Anne Sheehan and Daniel K. Sullivan, Defect Tracking Systems in Global Software Development – a work practice study, Limerick, Ireland, 2007.

[2] (Basili, 1984) Victor R. Basili, and Barry T. Perricone, Software Errors and Complexity: An Empirical Investigation, Communications of the ACM, P.42-52, 1984.

[3] (Black, 1999) Rex Black, Managing the Testing Process, Wiley, 1999.

[4] (Boehm, 2001) Barry Boehm and Vector R. Basili, Software Defect Reduction Top 10 List, p.135–137, 2001.

[5] (Curhan, 2005) Lisa A. Curhan, Software Defect Tracking during New Product Development of Computer System, Massachusetts institute of technology, 2005.

[6] (Edwards, 2006) Jim Nindel-Edwards and Gerhard Steinke, A Full Life Cycle Defect Process Model That Supports Defect Tracking, Software Product Cycles, And Test Iterations, Communications of the IIMA ,Volume 6 Issue 1, 2006.

[7] (Endres, 1975) Albert Endres, an Analysis of Errors and Their Causes in System Programs, Proceedings: International Conference on Reliable Software, P327-336, 1975.

[8] (Fredericks, 1998) Michael Fredericks and Victor Basili , A State-of-the-Art-Report for Using Defect Tracking and Analysis to Improve Software Quality , 1998.

[9] (Fry, 2010) Zachary P. Fry and Westley Weimer, A Human Study of Fault Localization Accuracy, 2010.

[10] (Grady, 1996) Robert B. Grady, Software Failure Analysis for High-Return Process Improvement Decisions, Hewlett-Packard Journal, 47(4), 1996.

[11] (Hooimeijer, 2007) Pieter Hooimeijer and Westley Weimer, Modeling Bug Report Quality, In Automated Software Engineering, pages 34–43, 2007.

[12] (IBM, 2005) IBM® Rational® ClearQuest® Deployment Kit, Usage Model Guidelines, Version 1.1, Jan., p. 4, 2005.

[13] (IEEE, 1990) IEEE, IEEE Std. 610.12: Standard Glossary of Software Engineering Terminology. The Institute of Electrical and Electronics Engineers, New York, NY, USA, 1990.

[14] (Jalbert, 2008) Nicholas Jalbert and Westley Weimer, Automated Duplicate Detection for Bug Tracking Systems, IEEE, 2008.

[15] (Janák, 2009) Jiří Janák, Issue Tracking Systems, Brno, spring 2009

[16] (Juran, 1988) Joseph M. Juran, Juran's Quality Control Handbook, McGraw-Hill, 1988.

[17] (Just, 2008) Sascha Just, Rahul Premraj and Thomas Zimmermann, Towards the Next Generation of Bug Tracking Systems, 2008.

[18] (KO, 2003) Andrew J. KO and Brad A. Myers, Development and Evaluation of a Model of Programming Errors, 2003.

[19] (Lenin, 2010) R.B. Lenin, R. B. Govindan and S. Ramaswamy, Predicting Bugs in Distributed Large Scale Software Systems Development, SCS M&S Magazine, 2010

[20] (Lethbridge, 2003) Timothy C. Lethbridge , Janice Singer and Andrew Forward , How Software Engineers Use Documentation: The State of the Practice, IEEE Software Volume 20 (Issue 6), 2003.

[21] (Mays, 1990) R.G. Mays, C. L. Jones, G. J. Holloway and D. P. Studinski, Experiences with Defect Prevention, IBM Systems Journal, 29 (1) (January):4-32,1990.

[22] (Microsoft, 2012) Process Guidance documentation provided in the documentation for Microsoft Visual Studio,2012,http://msdn.microsoft.com/enus/library/ee332488.aspx.

[23] (Ostrand, 1984) Thomas S. Ostrand, and Elaine Weyuker, Collecting and Categorizing Software Error Data in an Industrial Environment, the Journal of Systems and Software, 4:289-300, 1984.

[24] (Ostrand, 2004) Thomas J. Ostrand and Elaine J. Weyuker, A Tool for Mining Defect-Tracking Systems to Predict Fault-Prone Files , 1st International Workshop on Mining Software Repositories, united kingdom, 2004 .

[25] (Pfleeger, 1998) Pfleeger, and Shari Lawrence, software engineering theory and practice upper saddel river, NJ: prentice hall, 1998.

[26] (Rus, 2002) Ioana Rus, Combining Process Simulation and Orthogonal Defect Classification for Improving Software Dependability, Chillarege Press, 2002.

[27] (Stimson, 1998) Carl Stimson, the Quality Improvement Story, 1998.

[28] (Sullivan, 1992) Mark Sullivan and Ram Chillarege , A comparison of Software Defects in Database Management Systems and Operating Systems ,IEEE,1992 .

[29] (Tammana, 1998) Prathibha Tammana and Danny Faught, Software Defect Isolation, 1998.

[30] (Weinberg, 1983) Gerald. M. Weinberg, Kill That Code, Info systems, 1983, P.49.