An Architecture based Approach for Reliability Estimation of a Source Code Navigation tool

J Emi Retna
Assistant Professor (SG)
School of Computer Science & Technology
Karunya University

Sumy Joseph
II M.Tech SE
School of Computer Science & Technology
Karunya University

Merlin Soosaiya
II M.Tech SE
School of Computer Science & Technology
Karunya University

ABSTRACT
Annotations play a significant role both in software development and software maintenance activities. The semantically rich annotations will be supporting the software developers to a very significant level. The current source code annotations which are provided by modern development environment such as Eclipse are having difficulty in managing the annotations. Thus, the motivation to improve usability, efficiency of development tools and to reduce development time and cost has been emerged. The main objective of this paper is to provide insights in defining semantically rich annotations to source code using Tags for Software Engineering Activities (TagSEA) tool and to improve navigation and management of annotations while estimating the reliability of the tool. Reliability is one of the illusive targets to achieve in the software development for the successful software projects. It is one of the most important parameter or attribute of software to be achieved for the software quality. There are different techniques and models used for estimating the reliability of the software. We are using an architecture-based approach for estimating the reliability.

Keywords
Source code navigation, Reliability estimation, tagging, test resource allocation

1. INTRODUCTION
Software developers use annotations in source code to support the development process. If the annotations are semantically rich annotations, then the support extends to a significant level. Current source code annotations which are provided by modern development environment such as in Eclipse are difficult to manage. Thus, the quest to investigate how to enrich these annotations with additional semantic information is emerged. The motivation that includes improve efficiency of development tools, to improve usability and to reduce development time and cost.

A new approach for software navigation called Tags for Software Engineering Activities (TagSEA) is considered. It combines tagged waypoints in a software context. The reminding and refinding mechanisms from other domains are very much inspired [11]. The two concept used are the waypoints used to assist navigation in sailing and the use of tagging in social bookmarking systems have been delivered the ideas. Tagging examples can be seen on sites such as Flickr etc. Basic hypothesis concerning the use of TagSEA is that tagged waypoints may improve refinding and reminding [12]. In this paper, the use of code navigation tool in software development and maintenance is explored.

1.1 The Existing Tool Support
The various tools that support annotations and navigation from within and outside the source code are available. These tools include: Unstructured Source Code Comments, Task Annotations, Bookmarks, Grouping Concerns [14], Tours and Guides [15] etc. But these existing tools not support both metadata and the critical aspects for managing annotations like reminding and refinding [11].

2. TAGS FOR SOFTWARE ENGINEERING ACTIVITIES
TagSEA is a plug-in for the Eclipse IDE that combines the notion of waypointing with tagging for software development [11].

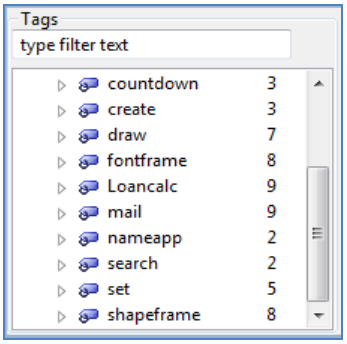

Figure 1 Tag filter and a tree of hierarchical tags

The Figure 1 shows TagSEA tool which includes tag tree and a tag filter. A hierarchy of tags can be viewed which is related with the programs written within Eclipse IDE. Another part of the tool is shown in figure 2 which is known as waypoint view. It contains data like message, location, author and date. There are different highlighted features for this tool such as tags are created in a light weight manner, an easy navigational taxonomy with hierarchical tags and can easily change the tagged code with refactoring facility etc. [11]

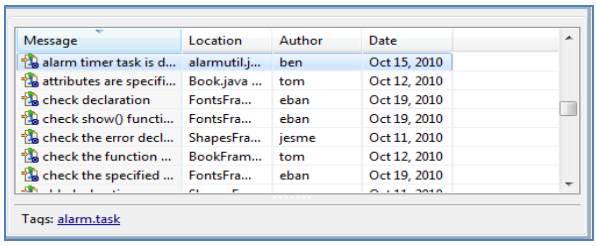

Figure 2 The Waypoint View

We can tag locations of interest directly in to the source code as you type. For example: //@tag fontframe.error -author="ann" -date="enUS:10/19/10" : Check the error

In the above example, the word "@tag" in the comment indicates that the location has been tagged as interesting. The words immediately following this indicator are the tag names or keywords that you can use to return to this location later. The other data included is author metadata, date metadata, a comment or a message [11], which is placed after the colon (:) and has the text: "Check the error". The descriptive keyword for a location is used as tagname in the syntax. Names that have dots (.) in them can be treated as hierarchical in the views. For example, the tag font frame error could represent error in fontframe, which is inside the sample project, and can be visualized as such inside the Tags View. We can add metadata to identify the author of a tagged location and the date of creation.

2.1 The Components of TagSEA tool

The TagSEA tool comprises of different components. The main components of TagSEA tool is mentioned below. It includes Tags pane, Waypoint pane and Cloud see. The tags pane gives you a filtered list of all of the tags available in your workspace. You can use the text box to enter words that may match tags that you are interested in [11]. The '.' character has a special meaning in TagSEA: it is used to view tags as a hierarchy. When you view your tags as a hierarchy, a number of refactoring actions are available. You can re-parent tags by dragging them under a new parent. You can delete tags, and you can "generalize" tags. Managing a growing sea of tags is a concern for all social tagging systems [13] and this may be a problem for large software systems as well. Double-clicking on any waypoint will navigate to the location of that waypoint. And list all of the tags that are on the selected waypoints. Clicking on a tag name will open it in the Tags Viewer. Also to find out more detailed properties about a tagged location, you can right click on the waypoint and select 'properties'. Some of these properties can be edited, depending on the type of waypoint you created. Another feature is we can visualize tags using a tag cloud. To access CloudSee, select the CloudSee [11] Icon on the right side of the TagSEA View.

2.2 The Calculation of Tag Reuse

The number of tag occurrences and unique tags are counted. Hierarchical tags (e.g., set.panellayout and search.end) were counted as unique. We used this data to calculate the frequency of tag reuse. For that purpose we have selected ten sample java programs in which the tags will be created with the specified syntax. Across these ten sample programs, we discovered a common pattern: most tags were used only once or twice, while a small number of tags were used quite often. For example, in Banking application program, the tag grouping is used over 10 times. There were instances of the same tag being reused, with one tag describing six waypoints across three files. Actually we counted the number of waypoints and tags at each time slice. We also measure the average number of times a tag is reused: (#tag occurrences)/ (#unique tags) [11].

The analysis of the tag data revealed that all the programs took advantage of the hierarchical naming capability of TagSEA and frequently reused the higher level of tag names. For example, nearly all of the tags in Library management program, many of the tags begin with the prefix "libtodo". The depth of the tag hierarchies indicates that structure and grouping are used for the tags in each of the projects. And in this analysis the tags had a hierarchy depth of at least two and a maximum hierarchy depth of three.

3. 3. THE IMPORTANCE OF SOFTWARE QUALITY

Critical software is an application whose failure or disruption may lead to catastrophic loss in terms of cost, damage to the environment, or even human life. The software systems are increasingly being used in critical context. So assuring the quality for the system is very important. But for the complex systems ensuring the high quality with less verification is complex task. Software quality evaluators use various standard procedures, work instructions, and tools to objectively evaluate the quality of software processes and work products, which provide objective insight based on those evaluations, for achieving the specified requirements. Software quality plays a significant role in software engineering and software quality assurance consists of a means of software engineering processes and methods used to ensure quality.

3.1 Software Quality Concepts

Software quality can be defined as conformance to the stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software. Software quality measures how well software is designed, and how well the software conforms to that design although there are several different definitions. It is often described as the 'fitness for purpose' of a piece of software. Software quality can be viewed in three perspectives as,

- Software requirements are the foundations from which quality is measured and the lack of conformance to requirement is lack of quality.
- Specified standards define a set of development criteria that guide the developers in software development life cycle.
- A set of implicit requirements often goes unmentioned.
- Software which confirms to its explicit requirement but fails to meet implicit requirements, software quality is suspected.

The important software quality factors which lead to the success of the software can be Understandability, Completeness, Conciseness, Portability, Consistency, Maintainability, Testability, Usability, Reliability, and Efficiency.

3.2 The Importance of Software Reliability

Reliability is one of the most important characteristics of software. Thus, reliability can be defined as the quality of the fault free delivery of service within the specified constraints for a particular period of time. Regardless of the criticality of any single software application, it is also more and more frequently observed that software has penetrated deeply into most every aspect of modern life through the technology we use. As software becomes more and more crucial to the operation of the systems on which we depend, the software should behave in the way it is intended, or it should meet the requirements for which is defined and designed.

Ensuring high level of reliability for the software system assures the overall operational success of the system. The system reliability is the product of the individual reliabilities of each component [2]. The efficiency of the verification phase depends upon the correct identification of the most critical components in the software architecture [2]. Some standards and methodologies [5] for critical systems suggest asking the risk levels of the services in which they are involved. The paper deals with the two important concepts in the software engineering. One is about the recent method for the software navigation and the other one is the reliability estimation of the software. The paper describes the way in which the TagSEA [11] navigation can be implemented in the complex software applications and how it helps the software developers to support reminding and refinding. After that it also analyzes how to estimate the reliability of this software.

4. SOURCE CODE TAGGING AND RELIABILITY ESTIMATION

Software Reliability can be estimated by using the various matrices in the software engineering. The important matrices [4] which are used mostly includes,

- Exponential distribution models
- Weibull distribution model
- Thompson and Chelson's model
- Jelinski Moranda model
- LittleWood Model
- MTTF (Mean Time To Failure)
- MTTR (Mean Time To Repair)
- POFOD (Probability of failure on demand)
- Rate of fault occurrence
- Reliability=MTBF/(1+MTBF)

TagSEA; a complex application used for the source code navigation is considered for evaluating the software quality concepts. The paper proposes an efficient method for reliability estimation of system in an architecture-based approach [1]. There were different existing methods used for the reliability estimation of an application. The most common method was the black box based approach of software quality evaluation. In this particular approach, the reliability is evaluated by considering the entire software as a single entity and it does not consider the internal structure of the application. Here the reliability estimation is considered with an architecture-based approach. The approach is called an architecture-based one because; it captures all the behavior of the software and represents it. This can be done in such a way that, the tool can be represented with its components using a DTMC model [4]. The individual components of this tool is identified and represented independently in DTMC. The DTMC model can be used for the application architecture representation, because the state of each component changes in a random manner. A component can be a logically independent unit which will perform a particular function. Once the reliability of each component is estimated, then the overall system reliability can be calculated. System reliability is the product of the individual reliabilities of the component raised to the power of the number of visits to each units of application [1]. When estimating the reliability of the individual components, it is possible to identify the critical components of the application to an extent. For determining the reliability of the tool, we will count the expected visit count to each component. Expected visit count can be used for describing the usage of each component in the TagSEA Tool. The DTMC can be represented as a state transition diagram, in which each state represents as the TagSEA tool components. Different preconditions will be given and if all the components are satisfying the preconditions then we can ensure that the application is a reliable one.

There are two main models used, one for the reliability estimation and the other for testing time allocation. In order to represent the software architecture, we use an architecture based reliability model; Discrete Time Markov Chain (DTMC) type state-based model, which is mentioned above. A DTMC is characterized by its states and transition probabilities among the states. To represent the application as a DTMC [12], control flow graph is used. Assuming an application has 'n' components and with the initial index'1' and the final component by n, DTMC states represent the components and the transition from state 'i' to state j represents the transfer of control from component 'i' to component 'j'.

Software reliability of TagSEA can be estimated with a reliability allocation model. The approach can be used for evaluating the reliability of TagSEA components by running the application iteratively. From which, the successful course of execution path can be determined. Finally all the components which have done with the successful execution can be considered as reliable components. Based upon these reliability criteria the criticality of each component is evaluated. During the execution of the application, the transition probability, DTMC model [1, 2] can be obtained. SRGM (Software Reliability Growth Model) also constructed for the failure data. SRGM is used for representing the relationship between the reliability and the testing required. So for this purpose an SRGM model can be used which will describes how reliability grows as software is improved. The output of the reliability allocation model is given to the optimization model. Thus, required testing resources are allocated to each component. So the predefined reliability is compared with the actual reliability which is calculated by the formula (1-limn Nf/N), [14] where Nf is the number of observed failures and N is the number of executions of the input cases. The Figure 3 shows a sample graph with reliability estimates for the different components.

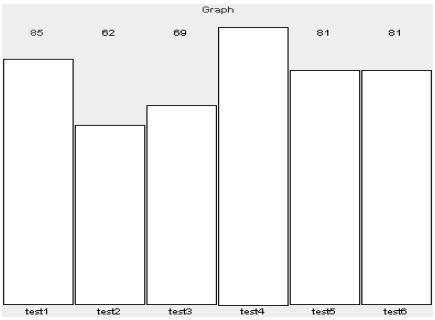

Figure 3 Reliability estimation with different components of an application

The main advantage is that we can analyze reliability based on software architecture which is an in depth approach. The fault tolerant mechanism is also evaluated and different levels of solutions are provided. But one of the drawback includes the model is not applicable for concurrent system evaluation.

5. CONCLUSION

Tags have been used in software engineering activities like development, maintenance and also for documenting bugs in bug tracking systems. Thus to define semantically rich annotations to source code, a new approach for source code navigation called Tags for Software Engineering Activities (TagSEA) is analyzed and reviewed. The reliability estimation of the tool is explored well with an architecture based approach in which we will be considering the fault tolerance, operational environment and the behavior of the tool. Hence we can conclude that using this architecture based approach for reliability estimation, the reliability of TagSEA can be easily estimated so that it can be used to enhance the software developers to improve navigation and management of annotations and also to reduce the development time and cost.

6. ACKNOWLEDGEMENT

We would like to thank to K.S. Trivedi and all the other authors who all were participated in proposing reliability and testing resource allocation model and also we would like to thank the researchers at the University of Victoria and at IBM Cambridge who all have been developed the TagSEA tool and for providing the results of different series of case studies.

7. REFERENCES

[1] Roberto Pietrantuono, Member, IEEE, Stefano Russo, Member, IEEE, and Kishor S. Trivedi, Fellow, IEEE "Software Reliability and Testing Time Allocation: An Architecture-Based Approach", IEEE transactions on software engineering, vol. 36, no. 3, may/June 2010

[2] S.Ramani, S.Gokhale and K. Trivedi,"SREPT: Software Reliability Estimation and Prediction Tool", Performance Evaluation, Special issue on Tools for Performance Evaluation, vol.39, no. 1, 2000 , pp. 37-60.

[3] S.S.Gokhale, W.E Wong, J. R.Horgan and Kishor S. Trivedi, "An Analytical Approach to Architecture-Based Software Performance and Reliability Prediction", Performance Evaluation, vol.58, no. 4,pp. 391-412, 2004

[4] A. Mettas,"Reliability Allocation and Optimization for Complex Systems," Proc. Ann. Reliability and Maintainability Symp. pp. 216-221, 2000

[5] K.Goseva-Popstojanova, and K. S Trivedi, "Architecture-based approach to reliability assessment of software systems", Performance Evaluation, vol.45, nos2/3,pp.179-204, 2001

[6] K. Goseva-Popstojanova, A.P Mathur, K. S. Trivedi,"Comparison of Architecture Based Software Reliability Models"

[7] M. R. Lyu,, S. Rangarajan, and A. P. A. van Moorsel,"Optimal Allocation of Test Resources for Software Reliability Growth Modelling in Software Development", IEEE Trans. Reliability, vol.51, no.2,pp. 183-192, June 2002

[8] S. Yacoub, B. Cukic, and H. H. Ammar, "A Scenario-Based Reliability Analysis Approach for Component-Based Software", IEEE Trans. Reliability, vol.53, no.4, pp.465-480,Dec. 2004

[9] Rani and R.B. Misra,"Economic Allocation of Target Reliability in Modular Software Systems," Proc. Ann. Reliability and Maintainability Symp., pp. 428-432, 2005

[10] Vibhu Saujanya Sharma a,*, Kishor S. Trivedi Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, UP 208016, Indiab Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA "Quantifying software performance, reliability and security: An architecture-based approach"

[11] Margaret-Anne Storey, Jody Ryall, Janice Singer, Del Myers, Li-Te Cheng, and Michael Muller, "How Software Developers Use Tagging to Support Reminding and Refinding,"IEEE Transactions on software engineering, vol.35, No.4, pp. 470- 483, Jul/Aug.2009

[12] M.-A. Storey, L.-T. Cheng, J. Singer, M. Muller, D. Myers, and J.Ryall, "How Programmers Can Turn Comments into Waypoints for Code Navigation," Proc. Int'l Conf. Software Maintenance, pp. 265-274, 2007

[13] M.-A. Storey, L.-T. Cheng, I. Bull, and P. Rigby, "Shared Waypoints and Social Tagging to Support Collaboration in Software Development," Proc. Conf. Computer Supported Cooperative Work, pp. 195-198, 2006

[14] M.P. Robillard and F. Weigand-Warr, "ConcernMapper: Simple View-Based Separation of Scattered Concerns," Proc. Workshop Eclipse Technology Exchange, pp. 65-69, Oct. 2005

[15] L.-T. Cheng, M. Desmond, and M.-A. Storey, "Presentations by Programmers for Program," Proc. Int'l Conf. Software Eng., pp. 788-792, 2007

J. Emi Retna, Assistant Professor (SG), Head of Computer Technology Centre, Karunya University, where she is currently working toward the PhD degree in the area of Software Engineering at the School of Computer Science and Technology.

Sumy Joseph is doing her M.Tech in Software Engineering from Karunya University, Coimbatore, Tamil Nadu.

Merlin Soosaiya is pursuing her M.Tech degree in Software Engineering from Karunya University, Coimbatore, Tamil Nadu.