A Review of Tools for Traceability Management in Software Projects

Satish C J [1], Anand M [2] and Thendral Puyalnithi[3]

[1]Satish C J, School of Computer Science and Engineering VIT University, Vellore India
[1]satish.cj@vit.ac.in
[2]Anand M, School of Computer Science and Engineering VIT University, Vellore India
[2]manand@vit.ac.in
[3]Thendral Puyalnithi, School of Computer Science and Engineering VIT University, Vellore India
[3]thendral.p@vit.ac.in

ABSTRACT

Software Documentation plays an important role in the maintenance of a software project. Many software artifacts are produced during the development phase of the project. These documents keep evolving during the maintenance phase. Before making any changes to the system the maintenance engineer should understand the architecture of the system and the impact of the changes made to the system. Software Documentation aids maintenance engineers understand the evolution of the system better. However for every revision of the Software Requirements Specification there exist revisions of Design, Test Cases and Software Configuration management documents. As these documents are independently maintained a lot of time is spent on understanding how the system evolved by manually mapping the various revisions. For instance understanding how a specific requirement was designed and tested involves mapping the correct design elements and test cases from the corresponding design and test case documentations. In order to reduce the time taken for manual establishment of traceability among documents several tools has been proposed by researchers for automated traceability management. This paper is focused on summarizing the tools available for Traceability management in Software Projects.

Keywords/ Index Term— Traceability Management, Software Engineering, Software Maintenance, Documentation

1. INTRODUCTION

Documentation plays a key role in knowledge transition and reduction of impact analysis time during maintenance phase. Documents like software requirements specification ,architectural design are frequently used in the maintenance phase.S.C.B de Souza gives the importance of various artifacts for the structured analysis and unified process [1].If documentation not maintained it may lead to poor system quality, costly maintenance and even early retirement [2].Documentation is however plagued by a lot of issues .These issues have led to the introduction of new tools and process refinements for documentation maintenance. Some of the common issues reported on documentation are

- Documentation is not up to date [3][4][5]
- Documentation is not accurate and complete [6][7]
- Changes to documents are not traceable[8]
- Lack of standards for documents[9]
- Documentation like UML diagrams are difficult to update[10]

Researchers have proposed various tools and techniques to address these issues. For instance documentation up-to-date ness can be achieved by creating a work culture where in engineers are motivated to work on documentation activities [11].Accurateness and completeness of documents can be achieved through effective reviews before implementation of the system. Many tools have been proposed that aid in automated documentation and also to locate the required information easily [12].One of the key issues that require more attention is traceability among various documents created.

Traceability is established using a traceability matrix and is very important with respect to understanding whether all the features have e been implemented and tested [13]

Traceability when done manually between documents is very time consuming and it also prone to errors. Traceability becomes more tedious in the maintenance phase where there is a need to establish traceability among several versions of different documents. The traceability information helps maintenance engineers understand the evolution of a system from several perspectives. In this paper we are summarizing the tools proposed by researches with respect to establishing traceability among documents.

2. TRACEABILITY MANAGEMENT

Traceability Management deals with the ability towards establishment of predecessors and successors for a given work element. In Software Engineering traceability is established among requirements, design, code and test cases. Requirements are gathered from the business users and documented as user requirements. Each user requirements is then mapped to a software requirement which actually translates the user requirement in terms of software functionality.

Every documented software requirement should be traceable to a high level design component. High level design components include items like class diagrams, Entity Relationship models, User Interface Screens etc... The high level design components are mapped to detailed design components which include detailed algorithms for all identified functions. Every function identified as part of detailed design is implemented in the coding phase.

The implemented system is tested using test cases written for all the software requirements. Therefore there exists a mapping between software requirements and test cases. Traceability management is all about establishing whether there exists a corresponding work element for a given item. For instance if we consider any specific requirement there should exist high level, low level design ,code and test case components for the requirement.

Traceability is of high significance as it is a process that ensures that every software requirement is indeed translated to a design that is implemented and tested. A traceability matrix is built to understand the traceability of different items in a project. A sample traceability matrix is given below

TABLE-I

Requirement ID	Design ID	Code	Test Case ID
#1	#56	testFunction1()	#T1
	#57	testFunction2()	#T2
			#T3
#2	#58	testFunction3()	#T4
	#59	testFunction4()	#T5
	#60	testFunction5()	#T6

Table-1: Sample Traceability Matrix.

3. ISSUES WITH TRACEABILITY MANAGEMENT

Though Traceability has widely recognized benefits it's often get not maintained successfully. Engineers fail to update the traceability links among impacted artifacts every time a change occurs [14] .The tools offered for Traceability management in software engineering is not satisfactory. Most of the tools that support traceability is lacking automatic or semi-automatic traceability link generation [15]

Integration of Traceability Management tools with artifact management systems and testing them with real users during the software maintenance phase is done rarely [16] Traceability is very expensive, both to reconstruct and to maintain. It is expensive to reconstruct: to recover lost information is very time consuming and difficult, if at all possible [17].

Traceability practices in general are far from mature, benefits are to a large part not conceived in the industry, and we are still standing at the beginning of an emerging discipline. A lot of research—both fundamental and applied—has still to be done [18]The number of artifacts produced as part of a software systems and the complex interrelationships among these artifacts lead to the core of the traceability problem[19]

Traceability Management is not still widely recognized by the industry as there is an overhead involved in maintaining traceability among software artifacts and the tools involved [20].Though tools are available for Traceability Management still lot of manual effort is involved before the actual use of these tools on software documents. For instance, assigning of keywords to every document prior to tracing them [21].

4. TOOLS FOR TRACEABILITY MANAGEMENT

Sherba, Susanne [22] proposes TraceM, a framework for automating the management of traceability relationships.TraceM provides registration, integration, evolution and querying services .Information Integration and Open Hypermedia services are utilized by TraceM for Traceability Management. Open Hypermedia allows storing of relationships separately from the artifacts. Information integration provides the services for creating and maintaining evolving relationships between artifacts. Relationship chaining can be applied to any relationship type known to the system

Ali, Nawazish[23] proposes Trustrace – A traceability recovery approach between requirements and source code using data mining. Trustrace uses a combination of both Information Retrieval Method and Data mining to establish traceability links between requirements and source code.Trustrace consists of three parts namely Histrace,Trumo and DynWing.Histrace mines the software repositories to create links from requirements to source code.Trumo reranks the recovered traceability links based on a web trust model.Dynwing assigns weights to the computed experts in the trust model.Trustrace is found to have more precision and recall when compared to tools that use only Information retrieval(IR) methods

TraceMaintainer [24] [25] is designed such that it can be deployed with any case tool. The changes made to the models in the case tool are recognized by traceMaintainer and the links are updated automatically.TraceMaintainer uses an adapter and rule engine for maintenance of links. The adapter recognizes change events in the case tool and provides elements properties to the rule engine which updates the traceability relations. The drawback of TraceMaintainer is that an adapter needs to be written for the integration of this tool with any specific case tool.

Asuncion, Hazeline proposes an End to End Software Traceability tool [19].This tool maintains the details of the artifacts in a MS SQL database. The trace information among artifacts is also entered by the users using a front end form. The tool follows three tier architecture. It also offers work flow support. This is the only tool that deals with end to end traceability of artifacts and was implemented for an organization.

REquirements TRacing On target(RETRO) tool for tracing requirements.[26].This tool uses Information Retrieval methods (Vector Space Retreival,Latent semantic indexing and Keyword word extraction methods).The IR method is executed using reweighted query vectors. The methods are continuously enhanced with user feedback processing.

Trust Analyzer tool for scenario based analysis [27] was proposed by Alexander. The system is executed using the test cases documented during development. By applying the scenarios the internal activities of the system can be observed and recorded. This tool establishes the traceability between scenarios which can exist in the form of plain English or diagrams and the source code.

ADAMS Re-Trace: a Traceability Recovery Tool [16] is built within the Advanced Artefact Management System (ADAMS).It uses the Latent Semantic Indexing(LSI) Information Retrieval technique for traceability recovery. This tool also compares the links retrieved by LSI with the links traced by the software engineer. If there are any discrepancies the links are highlighted.

Doors from IBM [28] is a requirements management solution that has the capabilities to link, trace, analyze and manage changes to requirements. Links Explorer in the DOORS helps the engineers understand the how a change in one artifact will affect the entire system. The tree view available in DOORS also allows the tracing of low level requirements to its higher level requirements.Doors offers graphical visualization of traceability information.

Rational Requisite Pro[29] is another tool from IBM that helps management of requirements .The tools allows traceability of

one requirement to another. For instance a feature requirement can be traced to a use case requirement using the tool. The tool can also be integrated with Rational ClearQuest Test manager which manages all the test cases. There by requirements can be associated with test cases and the tool allows report generation on such associations.

CRADLE[30] is a requirements management tool that enables traceability of items. Requirements can be imported from various sources like excel, word files and the traceability among different items can be established using the tool. The traceability and coverage of information can be analyzed using tables, matrices, and graphical hierarchy diagrams.

Innoslate [31] is a requirements management tool that allows import of requirements from other tools. All requirements can be easily traced directly to functional and physical elements of a system model, source documents, test plans, and other requirements. Due to an underlying model-based database, these relationships are automatically generated into Hierarchy Charts, Traceability Spider Diagrams, or 3D Traceability Diagrams. You can edit these requirements live and create new requirements directly in these diagrams

5. CONCLUSIONS

The review of the 10 tools gives us the following insight on tools used for traceability management

Research prototypes are based on Information Retrieval Techniques like Latent Semantic Indexing and Vector Space Modelling .Precision and recall was improved when IR was used in sync with Data mining .Commercial tools like Doors, Requisite pro, Cradle, Innoslate offer model based requirements management and traceability. The tools use relational tables to maintain artefact information and also enable automatic maintenance of traceability links. The impact of changes made to artifacts can also be traced using the tools.

The research prototypes are focused purely on traceability management of artifacts whereas the commercial tools offer traceability as a feature with requirements management.Commerical tools also offer better user interaction capabilities in terms of graphs and models.

The research prototypes can be used for small projects where in the number of users are small and the cost of maintaining a separate tool for traceability is a cost overrun. Though there are several tools for requirements traceability management very few research papers discuss on the need for end to end traceability of software artifacts. By end to end traceability we mean the complete traceability of requirements with design, detailed design, code and test cases. The maintenance of traceability links during the software evolution phase from an end to end perspective is still not addressed by most of the tools. Though commercial tools offer a variety of functionalities more cost effective solutions that offer end to end traceability for a software project should be further investigated.

REFERENCES

[1] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira. A study of the documentation essential to software maintenance. InICDC , pages 68{75, 2005}

[2] Khan, A.S., Kajko-Mattsson, M.. Management of documentation and maintainability in the context of software 8th International Conference on Computing Technology and Information Management 2012.

[3] Garousi, G., V. Garousi, et al. (2013). Evaluating Usage and Quality of Technical Software Documentation: An Empirical Study. EASE '13

[4] T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: The state of the practice. IEEE Software, 20(6): 35-39, 2003.

[5] Das,S.,Lutters,W.,Seaman,C.,(2007).Understanding Documentation Value in Software Maintenance,Proceedings of the 2007 Symposium on Computer human interaction for the management of information technology 2007.

[6] Forward, Andrew. Software Documentation–Building and Maintaining Artefacts of Communication. Diss. University of Ottawa, 2002

[7] Chomal, Vikas S., and Jatinderkumar R. Saini. "Software Template for Evaluating and Scoring Software Project Documentations." International Journal of Computer Applications 116.1 (2015): 15-27.

[8] A. Forward and T.C. Lethbridge, "The Relevance of Software Documentation, Tools and Technologies:A Survey," Proc. ACM Symp. Documentation Eng.(DocEng 2002), ACM Press, pp. 26–33

[9] Khan, A.S., Kajko-Mattsson, M. Management of documentation and maintainability in the context of

software handover 8th International Conference on Computing Technology and Information Management 2012

[10] E. Arisholm, L.C. Briand, S.E. Hove, and Y. Labiche, "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation,"IEEE Trans. Software Eng.,vol. 32, pp. 365-381,2006

[11] Kajko-Mattsson. A survey of documentation practice within corrective maintenance.Empirical Softw. Engg., 10(1):31{55, 2005}

[12] L. Moreno, Summarization of complex software artifacts, in: 36th International Conference on Software Engineering, ICSE, 2014, pp. 654{657}.

[13] Jaber, K., Sharif, B., & Liu, C.. A Study on the Effect of Traceability Links in Software Maintenance,1 2013

[14] Cleland-Huang, Jane, Carl K. Chang, and Mark Christensen. "Event-based traceability for managing evolutionary change." Software Engineering, IEEE Transactions on 29.9 (2003): 796-810.

[15] De Lucia, Andrea, et al. "Enhancing an artefact management system with traceability recovery features." Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on. IEEE, 2004.

[16] Lucia, Andrea D., et al. "Adams re-trace: A traceability recovery tool." Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on. IEEE, 2005.

[17] Lago, Patricia, Henry Muccini, and Hans Van Vliet. "A scoped approach to traceability management." Journal of systems and software 82.1 (2009): 168-182.

[18] Winkler, Stefan, and Jens Pilgrim. "A survey of traceability in requirements engineering and model-driven development." Software and Systems Modeling (SoSyM) 9.4 (2010): 529-565

[19] Asuncion, Hazeline U., Frédéric François, and Richard N. Taylor. "An end-to-end industrial software traceability tool." Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2007

[20] Jaber, Khaled, Bonita Sharif, and Chang Liu. "A study on the effect of traceability links in software maintenance." Access, IEEE 1 (2013): 726-741.

[21] Lucia, Andrea De, et al. "Recovering traceability links in software artifact management systems using information retrieval methods." ACM Transactions on Software Engineering and Methodology (TOSEM) 16.4 (2007): 13.

[22] Sherba, Susanne A., Kenneth M. Anderson, and Maha Faisal. "A framework for mapping traceability relationships." Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering. 2003

[23] Ali, Nawazish, Yann-Gael Gueneuc, and Giuliano Antoniol. "Trustrace: Mining software repositories to improve the accuracy of requirement traceability links." Software Engineering, IEEE Transactions on 39.5 (2013): 725-741

[24] Mäder, Patrick, Orlena Gotel, and Ilka Philippow. "Enabling automated traceability maintenance through the upkeep of traceability relations." Model Driven Architecture-Foundations and Applications. Springer Berlin Heidelberg, 2009.

[25] Mäder, Patrick, et al. "traceMaintainer-Automated Traceability Maintenance." International Requirements Engineering, 2008. RE'08. 16th IEEE. IEEE, 2008.

[26] Sundaram, Senthil Karthikeyan, et al. "Assessing traceability of software engineering artifacts." Requirements engineering 15.3 (2010): 313-335

[27] Egyed, Alexander. "A scenario-driven approach to traceability." Proceedings of the 23rd international conference on Software engineering. IEEE Computer Society, 2001

[28] http://download-na.telelogic.com /download /ugcagenda/visualizingdoorsinformationandtraceability.pdf

[29] http://www-01.ibm.com/support /knowledgecenter/SSRTLW_7.5.5/com.ibm.xtools.reqpro.doc/topics/c_trace.html

[30] https://www.threesl.com/cradle/index.php

[31] https://www.innoslate.com/requirements-management/

