Bulletin of Electrical Engineering and Informatics
ISSN: 2302-9285
Vol. 6, No. 2, June 2017, pp. 201~218, DOI: 10.11591/eei.v6i2.652

VTrace-A Tool for Visualizing Traceability Links among Software Artefacts for an Evolving System

C. J. Satish*, Anand M.
School of Computer Science and Engineering, VIT University
SJT 116 A07 VIT University, Vellore, Tamilnadu, India
*Corresponding author, e-mail: satish.cj@vit.ac.in

Abstract

Traceability Management plays a key role in tracing the life of a requirement through all the specifications produced during the development phase of a software project. A lack of traceability information not only hinders the understanding of the system but also will prove to be a bottleneck in the future maintenance of the system. Projects that maintain traceability information during the development stages somehow fail to upgrade their artefacts or maintain traceability among the different versions of the artefacts that are produced during the maintenance phase. As a result the software artefacts lose the trustworthiness and engineers mostly work from the source code for impact analysis. The goal of our research is on understanding the impact of visualizing traceability links on change management tasks for an evolving system. As part of our research we have implemented a Traceability Visualization Tool-VTrace that manages software artefacts and also enables the visualization of traceability links. The results of our controlled experiment show that subjects who used the tool were more accurate and faster on change management tasks than subjects that didn't use the tool.

Keywords: *traceability management software maintenance, software engineering, software documentation, traceability visualization*

1. Introduction

The maintenance of software systems is one of the most time consuming and tedious tasks. There are many issues that surround the maintenance of systems and one of the foremost issues is non availability of up to date system artefacts. Comprehension of a systems architecture becomes very complex without up to date documentation [1]. The absence of valuable information makes software maintenance to take up a significant portion of the system cost [2-3]. A lot of importance is given to documentation during the development stages of the project. The baselined documentation gets handed over to the maintenance team for knowledge transfer and for system architecture comprehension during change impact analysis. Such documentation carries with it rich information on the system and plays a significant role in the maintenance of a system. It is proven that usage of system documentation helps in improving the functional correctness of the changes made to the system [4].

The documentation handed over to the maintenance team should be updated for the changes made to the system. As the system evolves there is the need for the documentation to evolve such that it can help with system comprehension for engineers who are new to the system. However maintenance teams fail to update the baseline documentation for all change requests. Documentation over a period of time gets outdated and obsolete. Documentation quality is listed as one of the ten most acute problems in software maintenance [5].

There is a tendency for engineers to create a separate set of documents for every change they are making to the system. After many years of system maintenance, a system tends to have hundreds of documents generated for each version of the system. These documents are highly difficult to trace and comprehending the actual system using these pile of documents is again a very challenging task.

1.1. Significance of Traceability Management

Lifecycle of a requirement can be traced among several software artefacts. This traceability link is important with respect to understanding the transition of a requirement in to design, code and test cases. Such traceability links help maintenance engineers understand the

Received April 10, 2017; Revised May 10, 2017; Accepted May 25, 2017

impact of a change and also aid with estimation of the change requests. The traceability links also give an insight on the artefacts that are missing and the progress of the project. Relationships between software artefacts are usually accomplished by construction of a traceability matrix.

Traceability among software artefacts is very important for understanding the various changes made to the system [7]. Traceability paths can have a positive impact on evolution of system artefacts as we can trace whether a low level code led to artefact changes [8]. There are several activities in software engineering that need information on traceability. These activities are risk analysis, impact analysis, test coverage analysis, verification and validation of systems [9]. Industries have started using more number of traceability environments as traceability is considered to be a critical success factor in software projects [10]. Traceability is considered to be an investment that should be retained [11]. Traceability techniques help us with identification of all the artefacts that should be updated for a change request [12].

Traceability links which gets baselined after implementation of a project play a critical role in maintanence. Maintenance is mostly performed by engineers who were not part of the development life cycle. Industries have a tendency to outsource most of their maintenance tasks to third party vendors as a cost reduction activity. The maintenance engineers solely rely on the training and the documentation provided to them during software handover for system comprehension. The availability of traceability links between software artefacts provides the information on the impact of a change to maintenance engineers.

Most of the research done stresses on the importance of project documentation from the system development perspective. Documentation maintenance for evolving systems is one of the areas that need further investigation. In this paper, we propose a traceability framework which supports maintenance of system documentation content and also enables visualization of traceability links among different versions of the contents using models.

1.2. Traceability Management Issues

Though traceability management brings lot of value addition to a project, establishing traceability links and maintenance of those links is a time consuming and tedious task. Software projects are plagued by many issues like cost over runs, lack of communication among teams, schedule slippages, employee attrition etc. [15-16]. Traceability maintenance needs lot of effort and is error prone [18]. Maintenance of traceability links manually in software projects that has stringent deadlines and budget limitations is a difficult task. As a result established traceability links get outdated and is not maintained by engineers.

Traceability management for a software project that is continuously evolving can be very costly and maintenance of the traceability links involves lot of effort [17]. Certain artefacts are considered to be of higher priority and frequently updated whereas other documents become obsolete. The inconsistency in updating of documents with system evolution leads to broken traceability links [18]. Another problem reported is the enormous dual data entry work that should be done when tools are used for traceability maintenance. The presentation of data by such tools was also disliked by engineers. Many development engineers were not aware of the importance of traceability matrices and its importance. Moreover mainstream development team is not immediately benefitted from traceability and project implementation is not hindered by the absence of a traceability matrix [19].

Traceability matrix is created during the development phase of the project just because some process wanted it.Moreover the engineers do not trust the traceability models because they are not up to date and they do not carry the correct information. The models are reduced to pictures without life [20]. There are many commercial and research tools available that enable traceability among artefacts [21-22]. However these tools do not provide automatic traceability link maintenance. The links have to be maintained manually and the updating must be done frequently due to the iterative nature of software development. Even with semi-automatic traceability the link maintenance needs a lot of manual effort and is also error prone [23].

In this paper we are proposing a tool; VTrace which will help system engineers to maintain software artefacts content online and also maintain the traceability links between different versions of artefacts for an evolving system with minimal effort. We have achieved this by generation of traceability matrix and models for traceability link visualization. In Section 4, we detail the proposed system architecture. In Section 5, we demonstrate our implementation using

a sample scenario. In Section 6, we have documented our experimental results. In Section 7, we have discussed the threats to validity and finally summarized our research in Section 8.

2. Related Work

Patrick Mäder [13] in his paper revealed that traceability had a significant positive impact on subjects who performed maintenance tasks. A controlled experiment with 71 subjects performing real maintenance tasks on two development projects was conducted. The subjects were asked to provide the solution for eight maintenance tasks on the two projects. Subjects were asked to solve half of the tasks using traceability links and other half without using traceability links. The performance of the subjects was measured using the time taken and correctness of the completed task. It was reported that subjects who used traceability performed 24% faster and had 50% more correct solutions. Another observation is that the traceability links provided consistent results for all subjects irrespective of their subject experience and project domain. This study very well establishes the fact that the quality of maintenance tasks improve with the availability of traceability links irrespective of the development experience of the subjects.

Khaled Jaber et al. [12] have conducted a study on the effects of traceability links on the software maintenance activity. The study has proved that traceability links will reduce the effort and increase the accuracy of maintenance tasks. Survey was conducted at an industrial firm to understand the kinds of traceability links that stake holders are interested in.Traceability links were categorized based on stakeholder's roles. A traceability link tracing tool was developed and was given to the experimental group used in the study. The control group was not using the tool.28 subjects from both academia and industry participated on the study consisting of five maintenance tasks. The results show that the subjects who used the traceability tool were 86.06% more accurate than the subjects that didn't use the tool. Another important conclusion that this study has arrived at is that the use of the tool reduced the gap between high and low ability subjects and also between academic and industry subjects. This study establishes the importance of traceability links during the maintenance phase and the fact that a tool for traceability establishment will help even inexperienced users to perform better.

Charrada et al. [23] present an approach for automatically identifying requirements that are outdated for evolving systems. They identify code changes by comparing the new and old versions of the code. Then the keywords about the change are extracted from documentation and elements related to the change. The keywords are then used to trace the requirements for those code changes in the requirements specification documents using an automated traceability tool. This approach is very useful for maintaining up to date requirements specification documentation during the maintenance phase. The team has reported 79% precision by analyzing two source code versions and were able to detect 12 changes made to the code that are requirement related.

Xiaofan et al. [24] have presented an approach for visualizing traceability links between source code and documentation. A treemap view and hierarchical tree view visualization is proposed to avoid visual cluttering when visualizing traceability links for large systems. Parts of the documentation and source code is given as an input to the traceability recovery engine that retrieves the traceability links using composite set of traceability recovery techniques. The traceability links are filtered using a threshold and only those links that are above the threshold are visualized as a treemap or a hierarchical tree. Participants using the tool agreed that the tool help maintain traceability links with ease. This research not only focusses on retrieving links between source and documentation but also on the type of visualization of the links. This approach enables engineers to visualize traceability links for large systems without visual clutter.

Dasgupta, Tathagata, et al. [25] have proposed an approach for effectively retrieving traceability links between source code and artefacts using extended corpora with relevant documentation. The problem reported is that the same names should be used across different artefacts for efficient traceability link retrieval. The program source code, requirement documentation and other artefacts are taken up as the base corpora whereas the API documentation and other external resources for references are considered as expansions to the base corpora. They have tested this approach on three open source java projects and have reported an increase in precision of 31%.

Mohamed Yassine et al. [26] have reported a method for maintenance of traceability for evolving model driven systems. Two model versions are compared for any changes. The changes that need to be made to the traceability links are then identified and classified.The final stage involves the application of the changes to the traceability links. This approach details the automatic updates to the traceability links for any model changes in the maintenance phase. Marcus et al. [27] has reported the importance of visualization techniques for traceability links. They have reported a set of requirements for a traceability visualization tool. The part of the requirements were implemented as a prototype traceability visualization tool. TraceViz, the prototype tool displays the files impacted by a source file as squares. This tool enables the identification of traceability links between a source file and all its artefacts. Different colors were used in the visualization to represent normal links, missing links and warning links.

De Souza et al [28] analysed the importance of traceability links in global offshore development projects. It was reported that management of change propagation for artefacts in globally distributed teams is a major challenge and mostly the change management information if shared through emails. The study has revealed that customers at offshore development projects demanded tools that established different visualizations for the traceability links. The team has investigated the significance of two tools Ariadne and TraceVis for traceability management across globally distributed teams. The study has proved that use of tools for traceability management and visualization is welcomed by practitioners at offshore development projects.Reza Meimandi et al. [29] have analysed all the traceability recovery approaches and tool support available for test to code traceability. The team has analysed several recovery approaches bases on the traceability analysis type, trace type, trace direction, trace scheme, visualization and tool support. There are many papers that address traceability link recovery [30-35] and tools for traceability management [36-40].

Based on our literature survey, a lot research has been done on traceability recovery techniques and these techniques have been demonstrated by the usage of prototypes using subjects from the academia and industry. Traceability management effectiveness has been analyzed for the maintenance phase and also on projects that are globally distributed. The need for traceability links according to user categories and the different modes of visualization of traceability links have also been studied in-depth. Some researchers have also focused on the automatic maintenance of traceability links for evolving systems. A detailed analysis of all the papers reveals the fact that traceability management is considered to be a separate process in the projects life cycle. Research has only focused on establishing traceability links between the available software artefacts created for a project and also the usage of tools to manage the links between these artefacts. As it is well established that traceability management itself is a tedious process and it offers of no immediate benefit to the development team, it is very unrealistic to conclude that such a team will spend more effort on establishing traceability links. Performing various activities like recovering traceability links or trying to understand traceability tools will have a drastic impact on project budget and deadlines. The new algorithms and tools for traceability link recovery has been proposed without keeping in mind the time and effort needed by developers to apply them to real time projects. Though these prototypes and techniques prove beneficial on a controlled group it will indeed be of little interest to engineers as it is not in line with the project's development activities.

Therefore our research was not only focussed on proposing different visualizations for traceability links but also on the process of making traceability management a part of the project development process. Our objective is that traceability links should be automatically available to engineers by the nature of organization of artefacts contents during the development phase and not by an additional process. For instance an engineer will be using our tool for managing all their artefact contents and as a result traceability links will be available to them at no extra effort. We have designed our system such a way that the engineers won't be deviated from the mainstream development or maintenance effort for establishing traceability links. The standards enforced on management of artefacts inside the tool itself will enable the automated traceability link maintenance.

3. Proposed System

3.1. Concept of Systems, Projects and Versions

In the design of our tool we have used the concept of systems, projects and versions. A brief explanation on the context of their usage is given below;

3.1.1. System

Every organization is managed by a collection of systems. A system comes into existence to handle a new business process that has well defined boundaries within an organization. For instance a University is managed by systems like Student Management System, Employee Management System, Transport Management System, etc. Each system manages a set of well-defined functionalities for its users. All these systems interact with each other for accomplishment of various tasks. Usually each system will be maintained by a separate group of maintenance engineers. For every new system created in our tool we assign system ids. The system ids are prefixed with an S. For instance if an organization has two systems S1 and S2 already in place, then the third system which is created is assigned the id S3.

3.1.2. Projects

Every system that manages the organization goes through several revisions due to business demands or internal process refinements. Every time a system change is demanded by the business, a new project is initiated to accomplish the change. The project is handled by the maintenance engineers and a new version of the system with the required changes is released. Every system there by goes through various revisions by means of projects. Every Project is assigned a project id in the tool. If a project is carried out under System S1, then the system id is prefixed with the project id and assigned a project id as S11. For subsequent projects in System S1, the project ids are S12, S13 and so on.

3.1.3. Versions

During the maintenance phase, a system undergoes many revisions. A change request can be initiated to modify an existing functionality or to add a new functionality to the system. Whenever an existing functionality needs modification it means the existing baselined requirement for that functionality should be revised. If the baselined functional requirement is assigned an id S11FR1, then the revised requirement is assigned S12FR1.1 as shown in Figure 1. For subsequent revision of the same requirement in project S13 we assign the requirement id S13FR1.3. If we need to change the system by adding a new functionality then we assign the new requirement id by incrementing the previous requirement id by 1. The new requirement id is S11FR2 and any revisions to S11FR2 will be traced as S12FR2.1, S13FR2.2 etc. This enables us to track the evolution of a requirement within the system across multiple projects. The tracking of requirement revisions is shown in Figure 1.

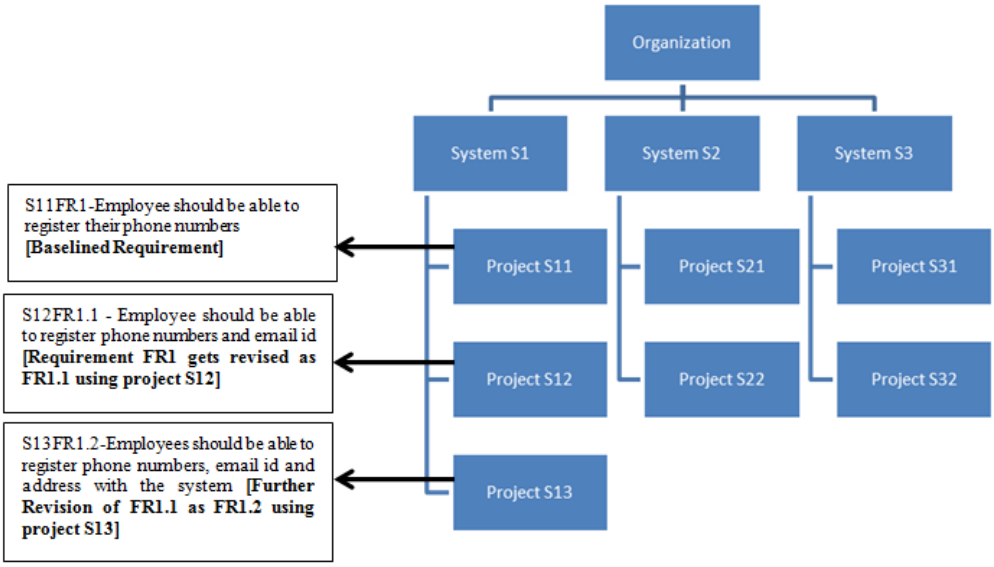

Figure 1. System and project hierarchy

The Sample system ids are S1, S2 and S3. The project ids are prefixed with the system id. For instance if system S1 needs a revision then project with project id S11 is created. Prefixing system ids with project ids for every system allows the unique identification of projects along with the system. The first project created for every system represents the development project with the baselined artefacts. For instance, the project S11 under system S1 represents the development project that lead to the creation of all baselined artefacts. If the baselined functional requirement S11FR1 needs a change, then a new project S12 is created and requirement S12FR1.1 is added to the list of requirements for system S1. Here S12FR1.1 denotes that this requirement is a modified version of the already existing requirement S11FR1 for System S1. If S12FR1.1 gets modified further in a future project S13, then the version of the latest modified requirement is S13FR1.2. For every modified requirement there exist design elements and test cases which are logged in with the respective version number.

4. Vtrace Architecture

The system maintains the requirements, design and test cases in relational tables. The Functional requirements are maintained as part of the requirements database. The requirements are maintained as revisions for each project as mention in section 3.1.3. The system also maintains the mapping of projects to systems and details about the projects in the project database.All design models like UML Class diagrams, sequence diagrams, state chart and data flow models are mapped with the respective requirements using the design database. The test cases and details of the requirements they are being mapped is maintained inside the test case database. Revisions to design and test cases are tracked in for every requirement change. If a design element like class diagram gets revised for requirement S12FR1.1 then the design element is assigned a revision id as Class diagram CD1.1. This enables us to track all design elements for the requirement revision S12FR1.1. The same concept apply to test case modifications.

The system utilizes the databases to retrieve information on the tracelinks between different artefacts. A traceability matrix is generated on demand for the users as a table. The matrix gives the details of the requirements, design and test case elements available for every project. The visualization module generates two models; a tree view model and a version model. A tree view model is generated for every project. This model displays the traceability links between every requirement to its design and test cases as a hierarchical structure. The version model displays the number of versions for a selected requirement as different nodes. The traceability links for each requirement revision and associated design and test cases is shown as a hierarchical structure in the version model. If a requirement has gone through three

revisions S11FR1, S12FR1.1 and S13FR1.2 then the version model displays the three versions of the requirements along with the associated design and test elements for each version. The tool architecture is given in Figure 2.

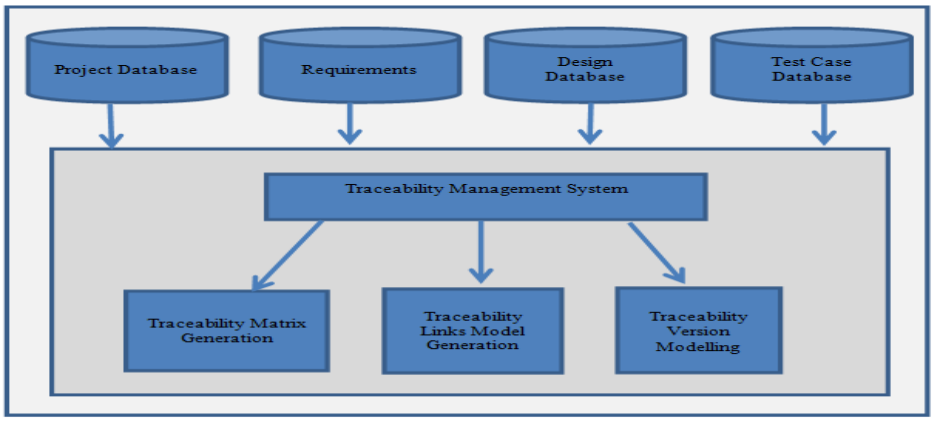

Figure 2. System architecture

5. Implementation

5.1. Sample System

The system that we consider here is a Grievance Redressal system that was developed for software engineering course work. As part of the project all software engineering artefacts namely requirements, design, source code and test cases were created. The requirements, design and test cases for the implemented project were considered to be baselined versions. The Grievance Redressal System was assigned a system id S1 in the tool. As we need to test the version traceability among artefacts that evolved during the maintenance phase we considered ten revisions to our system in terms of change requests. Each change request is assigned a project id in the tool. Change request one for system S1 is assigned a project id S12. For every system the first project is mapped to the baselined content. For system S1, the project S11 refers to the baselined version.

In every project we have considered both requirement revisions and addition of new requirements. We have also created a corresponding design and test case documentation for the revisions. We have used the documents created for Grievance redressal system in our experimental analysis detailed in Section 6. As part of the system implementation demonstration we have limited our system change requests to 2 and the details of the revisions used are given in Figure 3. The grievance redressal system has 2 baselined requirements [S11FR1 and S11FR2]. We have taken three baselined design elements namely class diagram and two sequence diagrams. The project S12 is created to accommodate the revisions to requirement S11FR1 and to add a new requirement S12FR3. The revisions to the design elements impacted by S12FR1.1 is shown as Class diagram GRCD1.1 and GRSD1.1. A new sequence diagram PVSD gets added as a result of project S12.Project S13 is initiated to complete revisions to requirement S11FR2 and S12FR3.The requirement revisions are shown as S13FR2.1 and S13FR3.1.The corresponding design element changes are shown as GRCD1.2, FVSD2.1 and PVSD3.1.

VTrace tool's UI design and validation was done using HTML, CSS and JavaScript. PHP and MySQL were used for server side scripting and construction of all the databases. GoJS libraries were used for construction of the models for visualization. All contents with respect to the artefacts were manually loaded into relational tables using Phpmyadmin interface of MySQL. As part of this implementation we have only completed the traceability matrix generation, traceability links model generation and traceability version modelling modules.

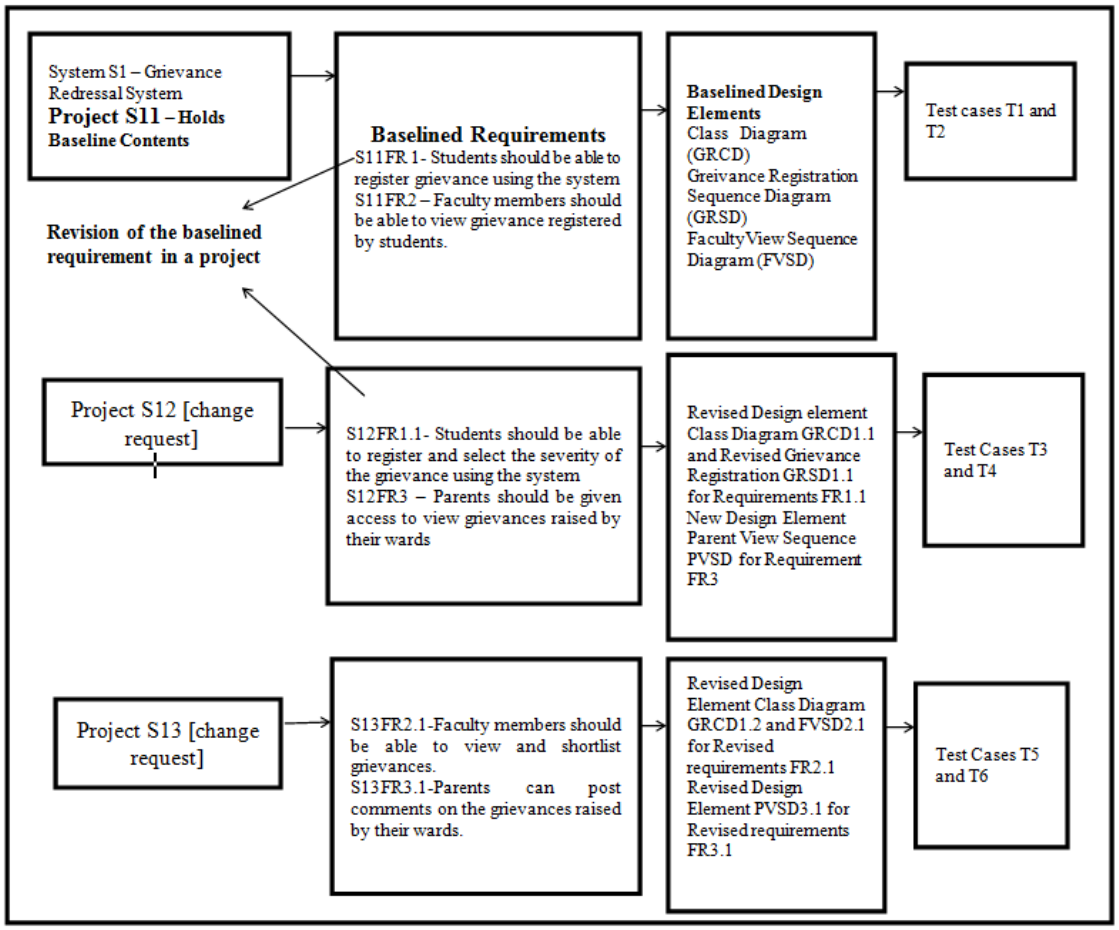

Figure 3. Sample system requirements

5.2. Generation of Traceability Matrix for the Sample System

The contents of the grievance redressal system were manually transferred to the tool. Three projects were created, Project S11 for baselined contents, S12 and S13 for change requests. The requirement revisions were tracked as given in Figure 3. The design and test cases were also mapped to requirements. The traceability matrix is generated for a selected project in a selected system. The generated traceability matrix for project S11 under system S1 is given in Figure 4. The mapping between Design and Requirements is given as Design element id (Requirement Id) in the Table. For instance Grievance Redressal Sequence Diagram GRSD is mapped to Requirement S11FR1 and it is mentioned as GRSD (S11FR1). A similar mapping representation is followed for Test cases and Requirements. Class Diagram GRCD is mapped to the project id and not any specific requirement as it is generated for an entire system. The tool also aids engineers to view the requirement and test case descriptions.

5.3. Generation of Traceability Model for the Sample System

A tree view is generated for a selected project in a selected system. The view represents the traceability links between requirements, design and testcases. The first node of the view represents the project id and the second level nodes denote the requirement ids for that project. The second level nodes further expand to show the availability of design and test cases for a selected requirement. The description of the requirement, design and test case elements is shown to the user as the user hovers the cursor over a node. The information displayed to the user on mouse hover is shown in Figure 5 for Project S13. This model not only gives the traceability links but also information on each node. The user is given the flexibility to expand or shrink a given node there by conserving space and helping the user focus only on a

particular subtree of interest. The models established for all the three projects is given in Figure 5. This graphical representation enhances users perceptions and there by aid quicker system comprehension

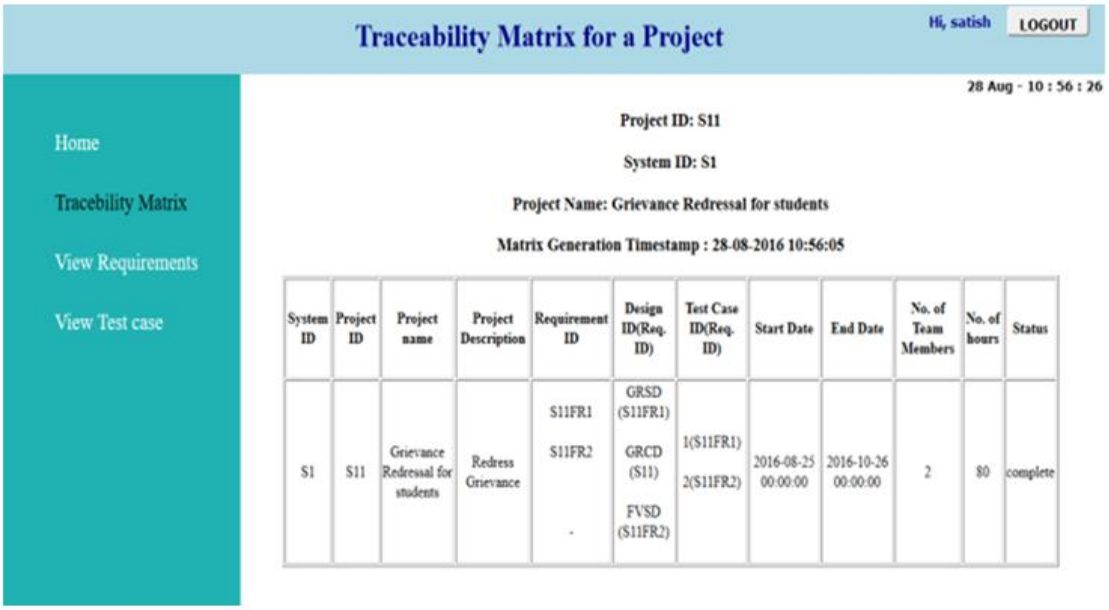

Figure 4. Requirements traceability matrix for project S11

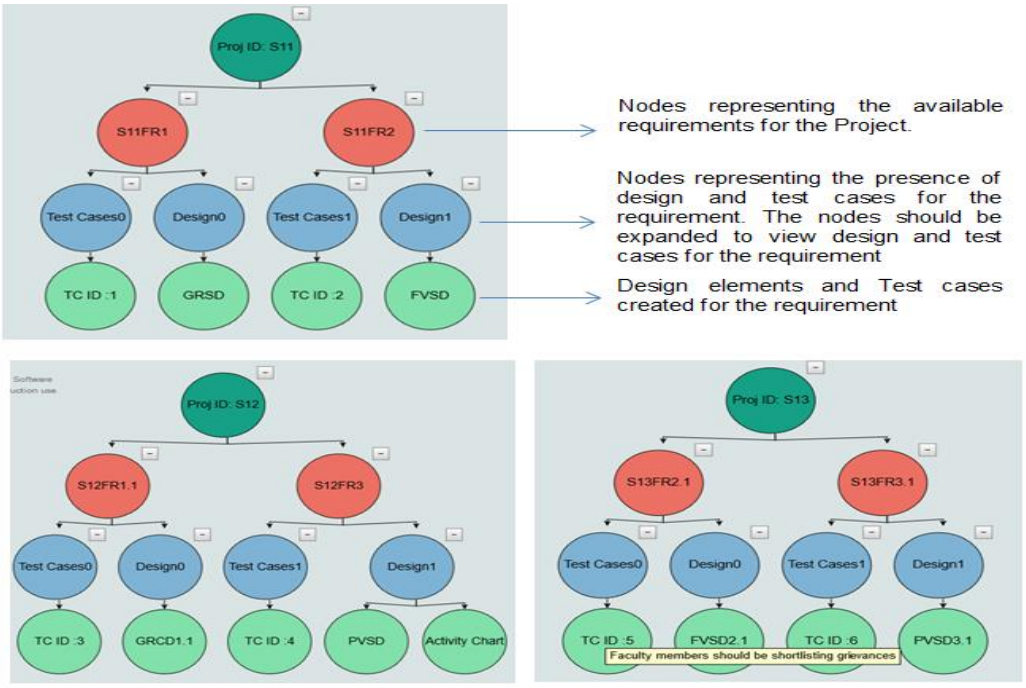

Figure 5. Traceability tree view model for project S11, S12 and S13

5.4. Visualizing Version Traceability

Visualizing how a selected requirement has evolved across multiple projects and what are the associated design and test case elements for each requirement version is the main

concept behind this visualization. The engineer selects a requirement id from a list of requirements and the visualization is generated by the tool for the selected requirement. We have shown the evolution of three requirements across two change requests S12 and S13 in Figure 6. Requirement S11FR1 is the baselined requirement for System S1. The functionality supported by this requirement needs revision and is accomplished using Project S12. Since it's a revision of an existing requirement it is assigned a requirement id as S12FR1.1. The model tracks the revision and also displays the associated test cases and design element for the baselined and revised version of the requirement. We have generated the model for all requirement revisions in our sample system and are shown in Figure 6. The model offers more flexibility in terms of expansion and shrinking of specific versions of interest to engineers.

This kind of a visualization aids the engineers understand the evolution of a system from a requirement perspective and also the design changes and test cases written for that specific change. Moreover the visualization gives us the information on the completeness of documentation for a version. Any missing links indicate the absence of artefact content for that version. A frequently changing requirement can be identified using this model. Information on test cases written and design elements modified for a frequently changing requirement will be of great significance for future change requests on that requirement. The description of the elements is given in a box below the model. This description is shown in the box whenever the engineer selects a node on the model. This model gets generated in a click of a button and it's easy for any user to comprehend the system evolution using this model. As we are generating this model only for selected requirements by the user we are also avoiding information overload and generation of an over complex model for the users.

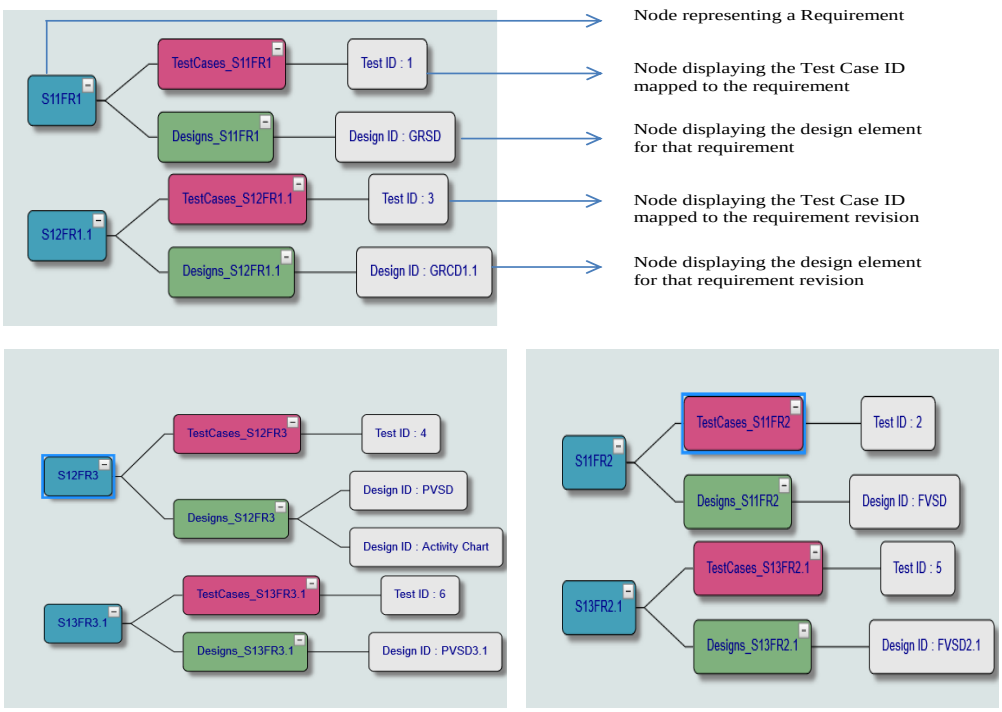

Figure 6. Requirement revisions across projects

6. Experimental Results and Analysis

Research Question 1:

Is the manual effort with respect to system evolution comprehension and change impact analysis reduced by the usage of the tool? Here we analyze the reduction of time taken with the usage of the tool to establish the traceability among different artefact versions for a system and comprehension of the system evolution

Measure:

By manual effort we mean the time taken to identify all the documents and to manually establish the traceability between requirements, design and testcases. This also includes the time taken to understand how the system has evolved with the changes. The measure will be the time taken to complete a given set of tasks on establishing traceability links and change impact analysis with and without the tool.

Research Question 2:

Is the accuracy of the tasks completed using VTrace tool better than the tasks completed without using the tool?

Measure:

By accuracy we mean the number of tasks completed correctly divided by the total number of tasks in each group.

Research Question 3:

How did the tool enhance user experience? Is the tool playing a significant role in increasing the quality of the task completion?

Measure:

By the quality of user experience we mean how far the user's perception gets enhanced by the traceability models and how the tool enables the easy comprehension of traceability and system evolution. We also measure the ease of use of the tool and the flexibility offered by the models for better understandability and usability of the tool.

Experimental Set up

Subjects

The subjects comprised of 60 undergraduate students of computer science. The students have completed their software engineering and software project management courses. The students are well aware of the significance of traceability management in software engineering. The students were all in the final year of their graduation. The students were randomly assigned to two groups. The first group is VT group which uses the VTrace tool to establish and visualize traceability and the next group is the no-VT group which does the task manually. Each group had 30 members assigned to it. All subjects had the same level of expertise in software engineering and therefore we reduced the impact of any other external factor on the experiment.

Activity

Students were given Grievance Redressal System project documentation for completion of their tasks. Sample requirements, design and test case documentation was created for each revision of the system. Six revisions had all the three documents where as two revisions had only requirements and test case documentation and two revisions had only requirements and design documentation.

no-VT group was given the baselined files and the files for all the revisions in separate folders. The name of the folders was given as change request numbers and each document had the change request number to be part of the title. Each revision document had only description on the changes made to the system for that revision. VT group had the contents of the artefacts loaded into the relational databases before the start of the activity. Both groups were asked to complete the given tasks.

Task 1:

The students of both groups were asked to construct a traceability matrix for each project and to identify and list the number of documents missing in each project. The students should record the time taken to complete this task.

Task 2:

The students were asked to study the number of times a requirement got revised across change requests and to identify the respective design elements and test cases for each

requirement revision. They were asked to record the number of documents that are missing in each revision.

The VT group should use the tool for the completion of the two tasks whereas the no VT group should perform the tasks manually. Both the group members were asked to record the time taken to complete the given activity.

Task 3:

Task 3 was on identifying the time taken for solving an emergency fix on the system. Students were asked to analyze the problem behind parent's comments on grievances not getting posted to the system. They have to record the results for the fix on paper and document the time needed to complete the emergency fix.

Task 4:

Students were asked to identify all the artefacts that needs a revision for a given change request. The change request was on introducing a functionality to collect feedback on the resolved grievances. The students were asked to identify all the artefacts that needed revision and record the time taken for completion of analysis of this task.

Experimental Procedure

- All subjects were given a briefing on the concept of systems, projects and versions and significance of traceability management in the maintenance phase.
- All subjects were given a briefing about the sample test documents that was used and the tasks that they should complete as part of the activity.
- VT group subjects were also given a briefing on the tool and the functionalities offered by the tool. They had some hands on training on the generation of traceability matrix and models using the tool. They were briefed on the relational tables used in the tool so that they understand the organization of artefact data for different systems in the tool.
- All subjects were asked to record the time taken to complete the tasks assigned to them in the given questionnaire.
- The VT group students were asked to fill a questionnaire that was focused on the quality and user satisfaction of the tool.

Data Gathering

Data gathering was performed using questionnaire distributed to all subjects. The subjects were asked to record the completion time of their tasks in the given form. VT group also recorded their user experience on the usage of the tool. They gave a rating between 1 and 5 for the product usefulness and usability for the tool.

Hypothesis formulation

The experiment has one independent variable (the use of VTrace Tool) and two treatments (VT and no_VT group).The dependent variables on which we compare treatments are Task Completion Speed [T_{CS}] and Task Accuracy [ΔP].

H1: There is no significance difference between the task completion time [T_{CS}] between the VT group that uses the tool and no VT group that does the task manually.

T_{CS}[VT Group]=T_{CS}[no_VT Group]

H2: There is no significant difference in the precision [ΔP] of tasks completed between the VT and no VT group

ΔP[VT Group]=ΔP[no_VT Group]

Results

Independent 2 sample t tests were conducted to determine the statistical significance between the two groups. The level of significance for the test was set to $\alpha=0.05$.The p values for the t tests are given in Table 1.

Table 1 Computed P Values

Task	Variable	Treatment	Mean	SD	t-Test
All	T_{CS}	No VT Group	66.95	16.76	p<0.0001
		VT Group	22.59	20.82	
Task 1	T_{CS}	No VT Group	62.60	6.14	p<0.0001
		VT Group	4.69	1.11	
Task 2	T_{CS}	No VT Group	56.84	7.88	p<0.0001
		VT Group	4.79	1.00	
Task 3	T_{CS}	No VT Group	58.85	7.25	p<0.0001
		VT Group	30.00	8.78	
Task 4	T_{CS}	No VT Group	91.29	16.41	p<0.0001
		VT Group	53.22	4.85	

Research Question 1:

Based on our independent 2 sample t tests we have found that the calculated t value falls inside the rejection region and hence the Null hypotheses is rejected. The computed p values are given in Table 1. Therefore we conclude that the performance of the VT group with respect to the task completion speed is better than the no-VT group.The students who were using the VTrace tool completed the task faster than then the group that was not using the tool. The students of no_VT group had to spend a lot of time on Task 2 as they need to read every change document and understand if a requirement has been revised or is it a new requirement. They were scanning the documents back and forth and had to spend a lot of time in establishing whether the requirement is a revision or a new requirement. The establishment of traceability links between revised requirements and their design and test case elements was challenging for the no_VT group. 11 students were not able achieve the outcome of Task 2 as they were having difficulty in understanding a requirement revision from a new requirement.

Task 3 was relatively easy for VT group students as they were able to immediately retrieve all the documents associated with the requirement. The test cases retrieved for this requirement helped the students in identifying the solution to the fix at a faster rate when compared to the no_VT group. The no_VT group was able to accomplish the task after retrieval of the associated documents but then they had to scan through all the change request documentation to establish the solution for the fix.

Task 4 was on change management and students had to spend time on impact analysis for the system. As the information was readily available on the evolution of the system and the traceability links well established for all artefacts, the VT group had taken less time for impact analysis on the change. The flow of information from the instigation of the system was well structured for VT group and thereby understanding the system or managing changes was relatively easy.The students of no_VT group spent much of their time locating and organizing the artefacts for understanding the system evolution. A box plot analysis for all the tasks based on the Task completion time for both the groups is given in figure 7

For VT group students the tasks were relatively easy to complete as the contents were already loaded in the relational tables as required and the time they spent was only on logging in, generation of the model and analysis based on information retrieved by the models. The information required was quickly available and hence they were able to complete the two tasks faster than no_VT group. The T test led to a positive conclusion and therefore we conclude that the tool plays an important role in the reduction of manual effort on system evolution comprehension and change management tasks.

Research Question 2:

The accuracy of the tasks completed by two groups was calculated using precision. The results of the groups were manually validated by the facilitator. The number of correct and incorrect results for both the groups across all tasks is represented in Figure 8. Establishing how a requirement got revised across many change requests was a very tedious task for no_VT Group students. It was found that 11 students from the no_VT group did not arrive at the correct results.

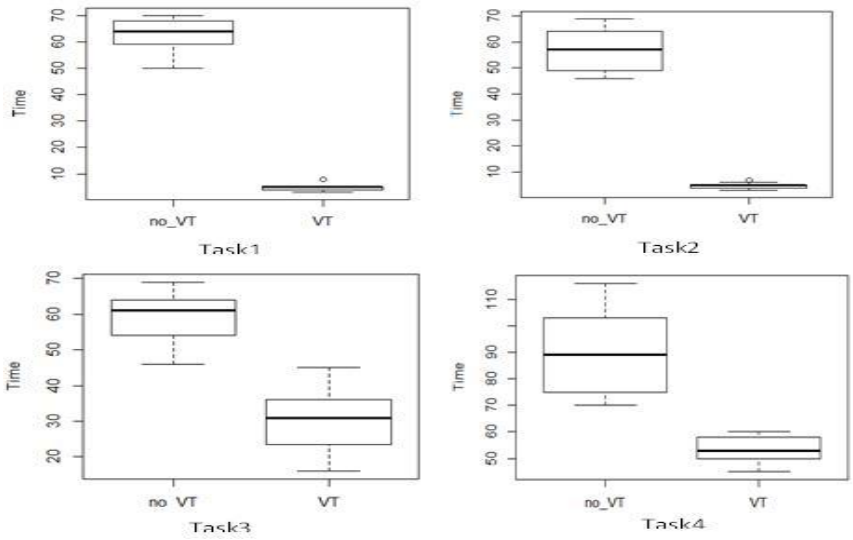

Figure 7. Box plot analysis for tasks based on task completion time

Task 4 was also difficult for no_VT group students as they need to spend a lot of time in locating and organization of content.9 students from no_VT group were not able to arrive at the correct results for Task 4. VT group students were more accurate on the tasks due to the information presented to them in the form of models. Precision ΔP is calculate for each group as shown below

Precision ΔP=Number of tasks completed correctly/Total number of tasks

Total Number of tasks in each group=Number of Tasks * number of subjects in that group

Total Number of tasks in each group=4 * 30=120 Tasks
Total Number of Tasks completed correctly for no_VT group=91
Total Number of Tasks completed correctly for no_VT group=112

ΔP [no_VT]=91/120=75%

ΔP [VT]=112/120=93%

From our experiment we were able to conclude that the precision for VT group is 18% higher than the no_VT group.

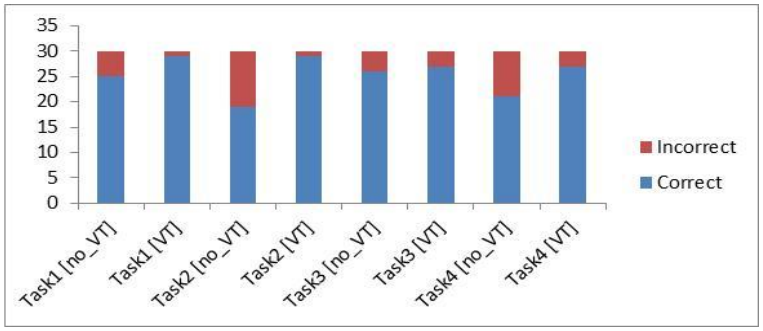

Figure 8. Precision analysis for tasks

Research Question 3

The user experience of the tool was rated based on the usability of the tool and the product usefulness with respect to the completion of the task. The VT group students were used in the rating of the tool. Almost all students who used the tool were satisfied with the usefulness and the usability of the tool. A few of the users felt that the model becomes large for many requirements and comprehension of the traceability model consumes time. There is a need to slide the model back and forth as it is not fitting to the screen. The questions used for collection of user experience feedback on the tool is given in Table 2. The focus was on measuring the ease of learning with respect to using the tool, ease of using the tool, how well the tool helps the user in solving the given task, the flexibility offered to the user in terms of interaction with the tool. Here by flexibility we mean how the user can generate models for different projects with minimal effort. How did the tool solve user's expectations and how motivating it is to use the tool? The users rated the tool from a scale of 1 to 5. In Figure 9 we have given the results of the user experience survey as a graph.

Table 2. User Experience Questionnaire

Question ID	Question
Q1	How easy it is to learn how to use the tool?
Q2	How easy is the tool to use
Q3	How flexible is the interaction with the tool?
Q4	Does the tool meet your expectations?
Q5	Is it motivating and exciting to use the tool
Q6	Was it helpful in solving your tasks without unnecessary effort?

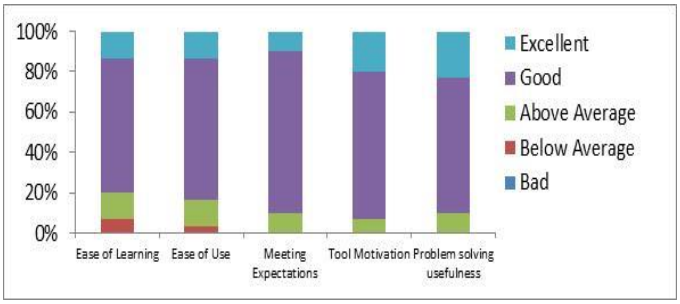

Figure 9. User Experience Graph

The user ratings were mapped to categories like bad, below average, above average, good and excellent as shown. A rating of 1 was mapped to bad, rating 2 to below average, rating 3 to above average, rating 4 to good and rating 5 to excellent. As we can see from the graph that almost 80% of the users have rated the tool above 3 for all the questions. A few of the users have felt the tool was below average when it comes to the learning and ease of use perspective. This should be mainly due to the fact that they have to understand the organization of the content in the relational tables and the mapping of requirement revisions that happen in projects across systems. Initial learning of organization of data and maintenance of revisions using the tool is a time taking task. However once when the traceability links are established and understood, the process becomes easier for further maintenance. Moreover the models get larger for many revisions and the users should be sliding the model back to understand the entire traceability for the project. Thus from our analysis of the questionnaire we can establish the fact that the tool plays a very crucial role in enabling users complete their maintenance tasks with ease and there by enhances their productivity.

7. Threats to Validity

7.1. External Validity

Our objective was to explore how novice engineers who are in to system maintenance understand the system evolution by visualizing traceability links. We have created software

artefacts for each revision and used that in our experiments to simulate a real project. The expertise of the subjects used in the experiments was that of novice engineers who are new to maintenance. We have reduced the impact of any external factors that can play a role on the outcome of our experiment. The usage of the tool reduces the time and effort on system evolution understanding and maintenance of artefact versions. The results that we arrived at using our sample will indeed be applicable to a wider population as the expertise and complexity of the task matches a real world application.

7.2. Internal Validity

Internal Validity deals with establishing a relationship between the use of the tool and the reduction in the time taken to complete the activity assigned to the group. The experience level of all the users of the tool was that of entry level engineers to an organization. We have used the same facilitator for briefing and validation of the results from both the groups. The assignment of team members to groups were also random to keep the experience levels balanced in both the groups.

8. Conclusion and Future Work

In this paper, we have addressed the problem of traceability management among artefact versions created for various change management tasks in the maintenance phase. We have presented our prototype and its usability with results. The current prototype that we have implemented is limited to visualization of traceability links among artefact contents loaded in relational tables. The contents of the artefacts were loaded directly into the tables. The tool should be further enhanced to allow engineers use a GUI interface to enter content. The versions among the contents were also entered manually in to the relational tables. The identification of a requirement revision and maintaining such revisions in our relational tables should be done using a front end application program.

The model has aided the subjects understand system evolution using the version modelling interface. The model also enhances visibility and progress of any maintenance project. It will aid project managers understand the status or the absence of documentation for a release. This enables project status tracking and maintenance of documentation for all versions Moreover we have proposed a design where in the entire project artefact content is maintained inside relational tables. This is not only helpful for traceability establishment but also maintenance of every project artefact online. The tool also offers different views of the traceability relationship in terms of traceability matrix and tree view models. This aids users get a better understanding on the relationship between different versions of the artefacts. The project is still under development and our initial research on the traceability of artefacts versions using the tool has yielded positive results. As the tool provides the needed support for content retrieval, organization and visual analysis of the system evolution, we believe that such a tool will enhance maintenance quality of industry standard applications where impact analysis consumes a major portion of the effort. This research addresses one of the key issues that plague the maintenance of evolving systems and we believe our completed tool will be great interest to software maintenance researchers.

References

[1] Satish CJ, T Raghuveera. *Visualizing Object Oriented Software Using Virtual Worlds*. Proceedings of the 4th WSEAS International Conference on Software Engineering, Parallel & Distributed Systems. World Scientific and Engineering Academy and Society (WSEAS). 2005.
[2] Das Sumita, Wayne G Lutters, Carolyn B Seaman. *Understanding Documentation Value in Software Maintenance.* Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology. ACM, 2007.
[3] Satish CJ, M Anand. Software Documentation Management Issues and Practices: A Survey. *Indian Journal of Science and Technology.* 2016; 9(20).
[4] Fernandez-Saez Ana M et al. On the Use of UML Documentation in Software Maintenance: Results from a Survey in industry. 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE, 2015R. Nicole, The Last Word on Decision Theory, J. *Computer Vision*, submitted for publication. (Pending publication)

[5] Palvia Prashant, Aaron Patula, John Nosek. Problems and Issues in Application Software Maintenance Management. *Journal of Information Technology Management*. 1995; 6(3): 17-28.
[6] Miller William L, Lawrence B Compton, Bruce L Woodmansee. *Assuming Software Maintenance of a Large, Embedded Legacy System from the Original Developer*. Software Maintenance (ICSM), 2013 29th IEEE International Conference on. IEEE, 2013.
[7] I Jacobson, G Booch, J Rumbaugh. The Uni_ed Software Development Process. Addison-Wesley, 1999.Lago, Patricia, Henry Muccini, and Hans Van Vliet. A Scoped Approach to Traceability Management. *Journal of Systems and Software.* 2009; 82(1): 168-182.
[8] Sundaram, Senthil Karthikeyan, et al. Assessing Traceability of Software Engineering Artefacts. *Requirements Engineering*. 2010; 15(3): 313-335.
[9] Dömges Ralf, Klaus Pohl. Adapting Traceability Environments to Project-specific Needs. *Communications of the ACM*. 1998; 41(12): 54-62.
[10] Mäder Patrick, Orlena Gotel, Ilka Philippow. *Enabling Automated Traceability Maintenance Through the Upkeep of Traceability Relations.* European Conference on Model Driven Architecture-Foundations and Applications. Springer Berlin Heidelberg, 2009.
[11] Cleland-Huang, Jane, Carl K Chang, Mark Christensen. Event-based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering*. 2003; 29(9): 796-810.
[12] Jaber, Khaled, Bonita Sharif, Chang Liu. A Study on the Effect of Traceability Links in Software Maintenance. *IEEE Access* 1. 2013: 726-741
[13] Patrick Mäder, Alexander Egyed. Do Developers Benefit from Requirements Traceability When Evolving and Maintaining a Software System?. *Empirical Software Engineering*. April 2015; 20(2): 413-441,
[14] Keil Mark. Pulling the Plug: Software Project Management and the Problem of Project Escalation. *MIS quarterly*. 1995: 421-447.
[15] Wallace Linda, Mark Keil. Software Project Risks and Their Effect on Outcomes. *Communications of the ACM.* 2004; 47(4): 68-73.
[16] Cleland-Huang, Jane, et al. *Software Traceability: Trends and Future Directions*. Proceedings of the on Future of Software Engineering. ACM, 2014.
[17] Perera Indika, Dulani Meedeniya, Madhushi Bandara. A Traceability Management Framework for Artefacts in Self-adaptive Systems. 2015 *IEEE 10th International Conference on Industrial and Information Systems* (ICIIS). IEEE, 2015.
[18] Ramesh B, Stubbs C, Powers T, Edwards, M. *Lessons Learnt from Implementing Requirements Traceability*. IIEEE International Symposium on Requirements Engineering, York, England. 1995. 89-99
[19] Egyed Alexander. *A Scenario-driven Approach to Traceability*. Proceedings of the 23rd international conference on Software engineering. IEEE Computer Society. 2001.
[20] Cleland-Huang Jane, Carl K Chang, Mark Christensen. Event-based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineerin*g. 2003; 29(9): 796-810.
[21] CJ Satish, M Anand, Puyalnithi Thendral. A Review of Tools for Traceability Management in Software Projects. *International Journal for Research in Emerging Science and Technology*. 2016
[22] De Lucia, Andrea, et al. E*nhancing an Artefact Management System with Traceability Recovery Features. Software Maintenance*, 2004. Proceedings. 20th IEEE International Conference on. IEEE, 2004.
[23] Charrada, Eya Ben, Anne Koziolek, Martin Glinz. *Identifying Outdated Requirements Based on Source Code Changes*. 2012 *20th* IEEE International Requirements Engineering Conference (RE). IEEE, 2012.
[24] Chen Xiaofan, John Hosking, John Grundy. *Visualizing Traceability Links between Source Code and Documentation*. 2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, 2012.
[25] Dasgupta, Tathagata, et al. Enhancing Software Traceability by Automatically Expanding Corpora with Relevant Documentation. *ICSM.* 2013.
[26] Haouam, Mohamed Yassine, Djamel Meslati. Towards Automated Traceability Maintenance in Model Driven Engineering. *IAENG International Journal of Computer Science* 43.2 (2016).
[27] Marcus Andrian, Xinrong Xie, Denys Poshyvanyk. When and How to Visualize Traceability Links?. *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*. ACM. 2005.
[28] De Souza, Cleidson RB, Tobias Hildenbrand, David Redmiles. Toward Visualization and Analysis of Traceability Relationships in Distributed and Offshore Software Development Projects. *International Conference on Software Engineering Approaches for Offshore and Outsourced Development*. Springer Berlin Heidelberg. 2007.
[29] Parizi Reza Meimandi, Sai Peck Lee, Mohammad Dabbagh. Achievements and Challenges in State-of-the-art Software Traceability between Test and Code Artefacts. *IEEE Transactions on Reliability* 2014; 63(4): 913-926.

[30] De Lucia A, Oliveto R, Tortora G. *IR-Based Traceability Recovery Processes: An Empirical Comparison of "one-shot" and Incremental Processes*. In: Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), IEEE Computer Society. 2008: 39-48, DOI 10.1109/ASE.2008.14
[31] Mader P, Gotel O. Towards Automated Traceability Maintenance. *Journal of Systems and Software*. 2012; 85(10):2205-2227, DOI 10.1016/j.jss.2011.10.023
[32] Cuddeback D, Dekhtyar A, Hayes J. *Automated Requirements Traceability: The study of Human Analysts*. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE10), 2010; 231-240, DOI 10.1109/RE.2010.35
[33] Walters, Braden, et al. *Capturing Software Traceability Links from Developers' Eye Gazes*. Proceedings of the 22nd International Conference on Program Comprehension. ACM. 2014.
[34] A Marcus, J Maletic, A Sergeyev. Recovery of Traceability Links between Software Documentation and Source Code. *International Journal of Software Engineering and Knowledge Engineering*. 2005; 15(4):811–836.
[35] A Marcus, JI Maletic. Recovering Documentation to-source-code Traceability Links Using Latent Semantic Indexing. In *ICSE*. 2003; 125–137.
[36] Ali Nasir, Yann-Gaël Guéhéneuc, Giuliano Antoniol. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. *IEEE Transactions on Software Engineering* 2013; 39(5): 725-741.
[37] Mäder Patrick, et al. *traceMaintainer-Automated Traceability Maintenance*. 2008. 16th IEEE International Requirements Engineering Conference. IEEE, 2008.
[38] De Lucia, Andrea, et al. *Adams re-trace: A Traceability Recovery Tool*. Ninth European Conference on Software Maintenance and Reengineering. IEEE, 2005.
[39] S Klock, M Gethers, B Dit, D Poshyvanyk. *Traceclipse: An eclipse plug-in for Traceability Link Recovery and Management*. in Proc. 6th Int.Workshop Traceabil. Emerg. Forms Software Eng., Honolulu, HI, USA, 2011: 24-30.
[40] FAC Pinheiro, JA Goguen. An Object-Oriented Tool for Tracing Requirements. *IEEE Software*. Mar. 1996; 13(2): 52-64.