Received: February 22, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

The Impact of Visualizing Traceability Links for Evolving Requirements in Software Maintenance– A Controlled Experiment

Satish Chinnaiyan Jayaraman[1]* Mahendran Anand[1]

[1]*School Of Computer Science and Engineering, VIT University, Vellore, Tamilnadu, India*
* Corresponding author's Email: satish.cj@vit.ac.in

Abstract: Traceability Management plays a key role in tracing the artefacts associated with every requirement during the life cycle of a software project. However, the trace links between artefacts are not updated as the system evolves during the maintenance phase. This often leads to untrustworthy documentation and information gets scattered across a pile of untraceable documents that were created for various change management tasks. The goal of our research is on identifying an approach to trace requirements evolution across change requests and establish trace links between artifacts for such evolving requirements in the software maintenance phase. We have implemented a prototype Traceability Visualization Tool – VTrace for tracing requirements evolution. This tool also supports the visualization of trace links for evolving requirements. The effectiveness of the tool on change management tasks was tested using a controlled experiment. The results of our controlled experiment show that subjects who used the tool were 21% more accurate on change management tasks than subjects that didn't use the tool. This study provides us with the evidence that tracing the evolution of software artefacts is highly significant for better system maintenance by novice engineers.

Keywords: Software maintenance, Software engineering, Traceability management, Trace link 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 system's architecture becomes very complex without up to date documentation [1,2]. 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. 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 [3].

The documentation handed over to the maintenance team should be updated for the changes made to the system. As the system evolves there is a need for the documentation to evolve. However maintenance teams fail to update the baseline documentation for all change requests [4]. Documentation over a period of time gets outdated and obsolete. Documentation inconsistency affects to ability of software maintainers to conduct 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. Almost 120 different design documents that were updated during software revisions were handed over to the US army by the original equipment manufacturer during software maintenance handover for operational flight program [6].

1.1 Challenges in traceability management

It is difficult to understand the impact of a requirement change to a large system. Development of tools and techniques that support requirement evolution is a very important issue [7].One of the important challenges in Traceability management is to establish trace link evolution as artefacts change. This research has gathered less than one third of the effort attributed towards traceability creation problems. Trace information is difficult for engineers to use as little research effort has been put towards presentation of trace links [8].

When an artefact evolves the trace link should be updated to connect to the new versions of the artefacts. However when an artefact is updated the connected artefacts are not updated. This is identified to be the important reason for incorrect trace links by majority of the interviewees. It is a challenge in globalized working environments [9].

Poor requirement traceability is considered to be one of the important requirement engineering challenges. Even with the modern tools it is highly difficult to trace requirements to design and architecture [10]. Peng Liang et al identify management of requirements in an evolving system to be a research challenge. Linking requirements rational knowledge to requirement artefacts created at different levels of the requirement engineering phase is the further refinement of the challenge [11].

Huge number of links between requirements and affected artefacts is the reason behind the scalability issues in requirement change management [12].

Visualization is widely being used by researchers and practitioners to comprehend requirement engineering activities. Zahra et al conducted a requirements engineering visualization literature review and found that requirements evolution is having the least visualization support [13]. Requirement management tools support visualization of requirements however the important limitation is that these visualizations do not show the relationship between requirements [14].

Researchers have focussed on understanding the effectiveness of traceability links in maintenance [15,16] but they have not considered the impact of traceability links for evolving requirements. The visualization of such evolving requirements and their relationships were also not considered by researchers who focussed on visualizing trace links [14,17,18].

It is clearly evident from the recent literature that tracing requirements evolution is still an open challenge. Visualizing the evolution of requirements and the relationships between the several versions of an evolving requirement in a global software development project is still an unaddressed problem. In this paper we are focused on proposing a traceability framework for addressing challenges in managing artefacts for evolving requirements. The knowledge on how a system evolved through its requirements coupled with the visualization of the evolution can aid novice maintenance engineers perform change impact analysis better. We are proposing a novel approach towards maintaining and visualization of artefact traceability for evolving requirements As part of our research we have developed a Traceability Management Tool – VTrace for tracing evolving requirements across change requests. This tool also supports the visualization of the requirement evolution.

2. Proposed tool architecture

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.

2.1 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 and 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.The tool can handle a maximum of 10 systems for an organization.

2.2 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 a project id S11 will be assigned. For subsequent projects in System S1, the project ids are S12, S13 and so on.

2.2 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 Fig.1. For subsequent revisions 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 Fig.1.

The Sample system ids are S1, S2 and S3.The project ids are prefixed with the system id. For 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 exists corresponding modified design elements.

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. 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 details of test cases and the requirements to which they are mapped to, is maintained inside the test case database. Revisions to design elements are tracked for every requirement change by assigning version number of the revised requirement to the design elements.

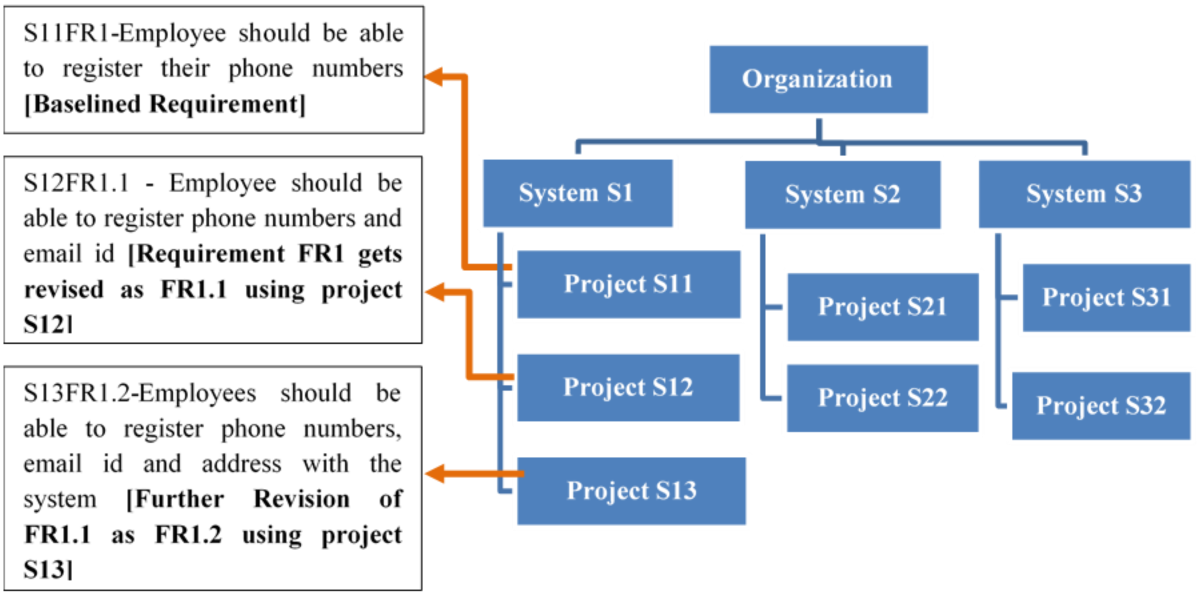

S11FR1-Employee should be able to register their phone numbers [Baselined Requirement]
S12FR1.1 - Employee should be able to register phone numbers and email id [Requirement FR1 gets revised as FR1.1 using project S12]
S13FR1.2-Employees should be able to register phone numbers, email id and address with the system [Further Revision of FR1.1 as FR1.2 using project S13]
Organization
System S1
System S2
System S3
Project S11
Project S12
Project S13
Project S21
Project S22
Project S31
Project S32

Figure.1 System and project hierarchy

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 system utilizes the databases to retrieve information on the trace links between different artefacts.

The requirement's evolution model displays the traceability links for an evolving requirement to it artefacts. The model displays the number of requirement 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 model. If a requirement has gone through three revisions S11FR1, S12FR1.1 and S13FR1.2 then the model displays the three versions of requirements along with the associated design and test elements for each version.

3. Implementation

3.1 Sample system

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. The requirement revisions considered in our sample system is given in Fig.2.

3.2 Visualizing requirement's evolution

This model enables the visualizing how a selected requirement has evolved across multiple projects. The model also displays traceability links between the revised requirements and the associated design and test case elements created for each requirement version. The model is shown is Fig.3.

The engineer selects a requirement id from a list of requirements as shown in Fig.3 (a). The visualization is generated by the tool for the selected requirement as shown in Fig.3 (b). The detailed description of the nodes is given in Fig.3 (c).

The model tracks the revision and also displays the associated test cases and design element for the baselined and revised versions of a selected requirement. The model offers more flexibility in terms of expansion and shrinking of specific versions of interest to engineers. This kind of a model and visualization aids the maintenance engineers to understand the evolution of a system from a requirement perspective. Moreover the visualization gives us the information on the completeness of documentation for a requirement revision. 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 nodes 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 a selected requirement by the user we are also avoiding information overload and generation of an over complex model for the users.

4. Experimental results and analysis

4.1 Research questions

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 complete change management tasks.

Measure:

By manual effort we mean the time taken to identify all the documents and to manually establish the traceability between requirements, design and test cases for system evolution comprehension during change management. The measure will be the time taken to complete a given set change management tasks 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.

4.2 Experimental set up

4.2.1 Subjects

The subjects comprised of 120 final year undergraduate students of computer science. All students have completed their software engineering and software project management courses.

The students were randomly assigned to two groups. The first group is VT group which uses the VTrace and the next group is the no_VT group which does the task manually. Each group had 60 members assigned to it. All subjects had the same level of expertise in software engineering.

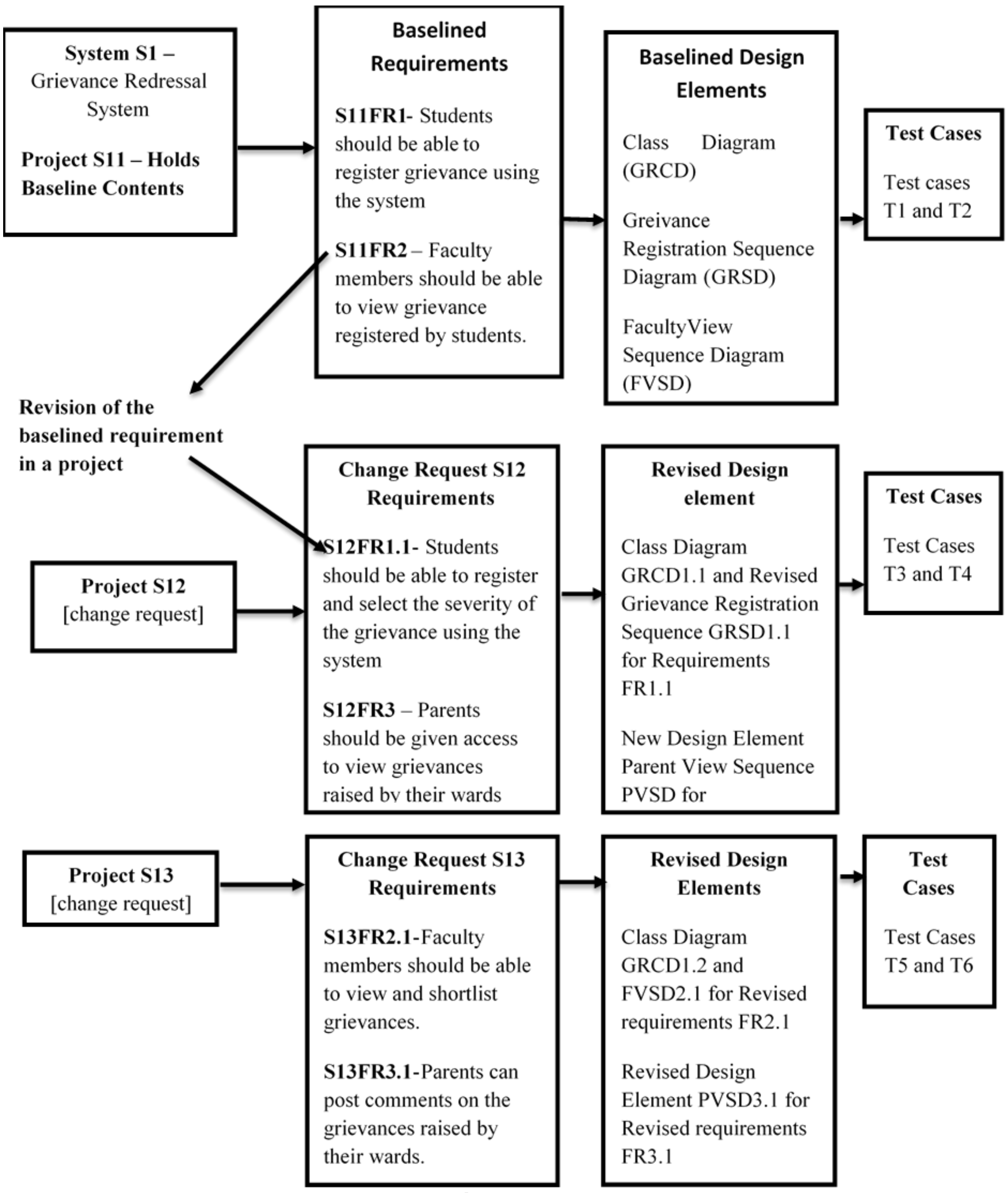
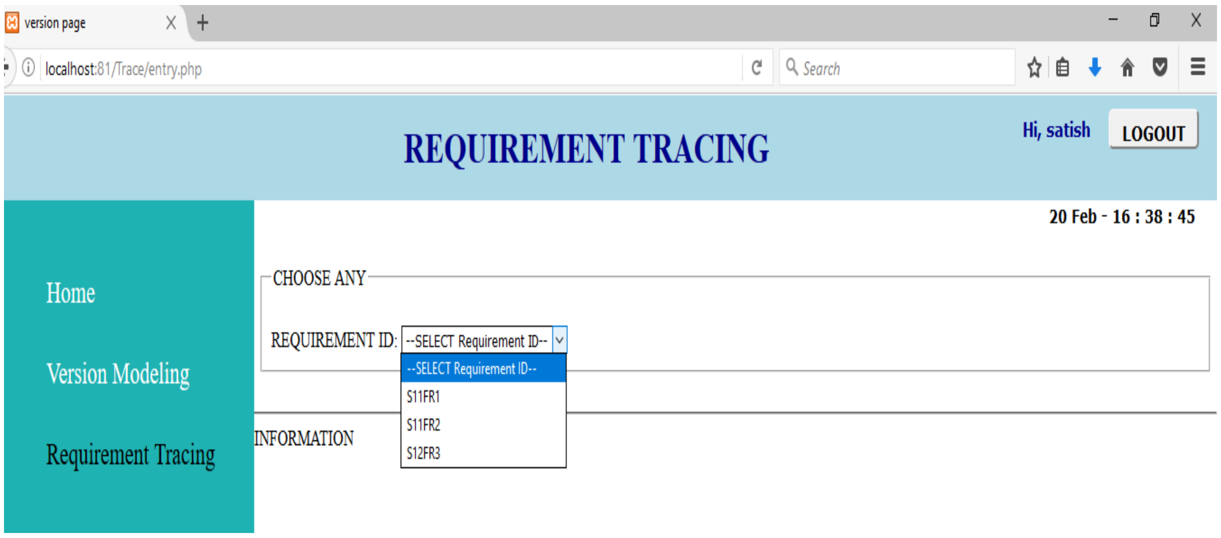
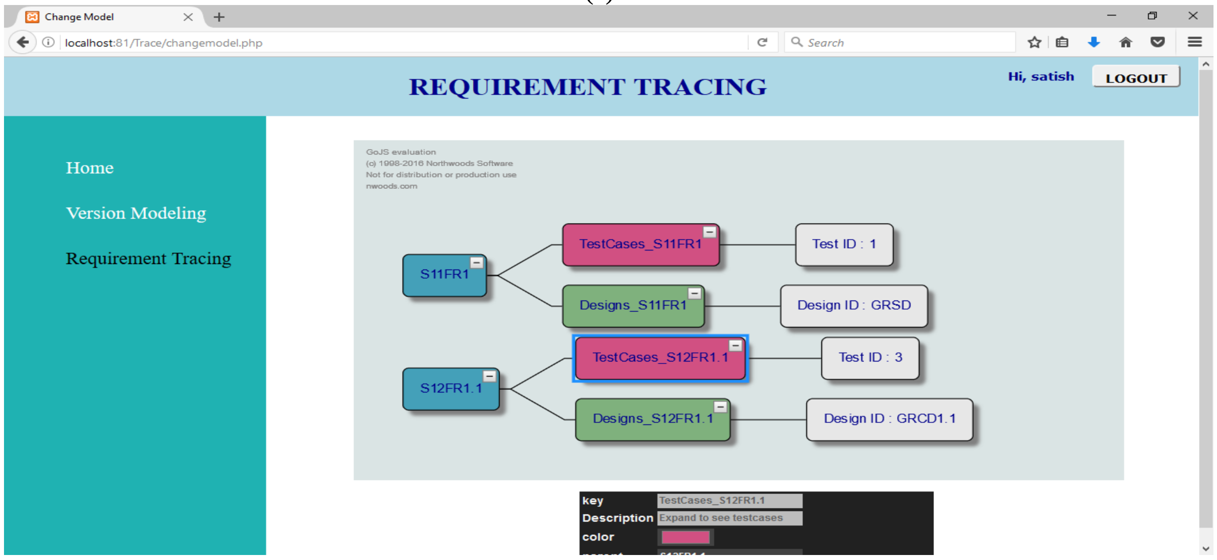

System S1 – Grievance Redressal System
Project S11 – Holds Baseline Contents
Baselined Requirements
S11FR1- Students should be able to register grievance using the system
S11FR2 – Faculty members should be able to view grievance registered by students.
Baselined Design Elements
Class Diagram (GRCD)
Greivance Registration Sequence Diagram (GRSD)
FacultyView Sequence Diagram (FVSD)
Test Cases
Test cases T1 and T2
Revision of the baselined requirement in a project
Project S12 [change request]
Change Request S12 Requirements
S12FR1.1- Students should be able to register and select the severity of the grievance using the system
S12FR3 – Parents should be given access to view grievances raised by their wards
Revised Design element
Class Diagram GRCD1.1 and Revised Grievance Registration Sequence GRSD1.1 for Requirements FR1.1
New Design Element Parent View Sequence PVSD for
Test Cases
Test Cases T3 and T4
Project S13 [change request]
Change Request S13 Requirements
S13FR2.1-Faculty members should be able to view and shortlist grievances.
S13FR3.1-Parents can post comments on the grievances raised by their wards.
Revised Design Elements
Class Diagram GRCD1.2 and FVSD2.1 for Revised requirements FR2.1
Revised Design Element PVSD3.1 for Revised requirements FR3.1
Test Cases
Test Cases T5 and T6

Figure.2 Sample System Requirements



(a)



(b)

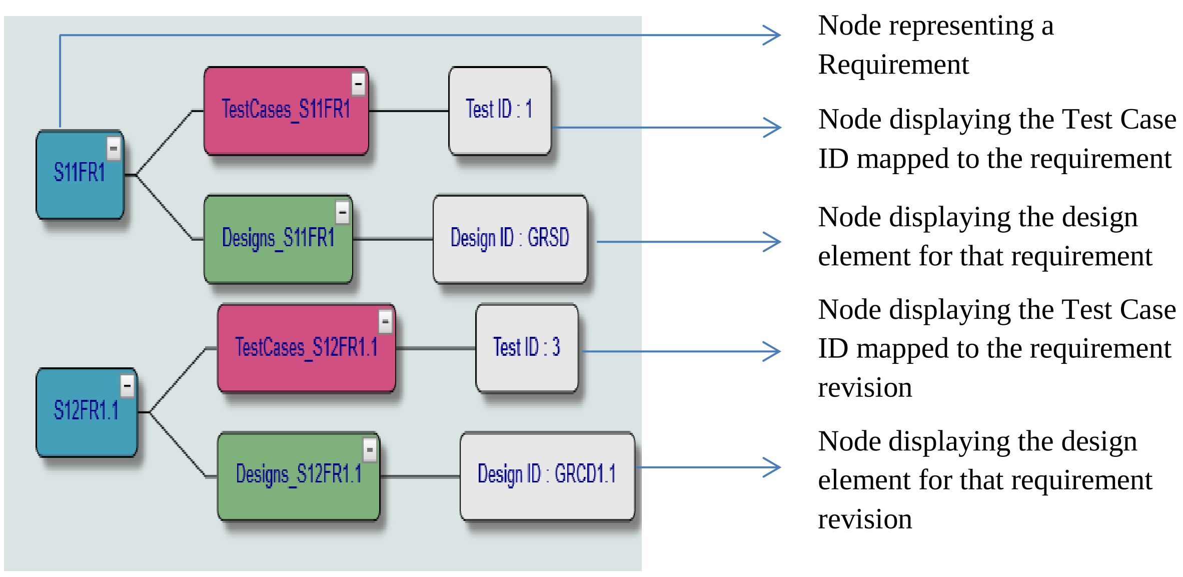

S11FR1
TestCases_S11FR1
Test ID : 1
Designs_S11FR1
Design ID : GRSD
S12FR1.1
TestCases_S12FR1.1
Test ID : 3
Designs_S12FR1.1
Design ID : GRCD1.1
Node representing a Requirement
Node displaying the Test Case ID mapped to the requirement
Node displaying the design element for that requirement
Node displaying the Test Case ID mapped to the requirement revision
Node displaying the design element for that requirement revision

Figure.3 System model: (a) requirement selection, (b) model generation, and (c) model description

4.3 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 whereas 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 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 2:

Task 2 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 3:

Students were asked to identify all changes that should be done to existing artefacts 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 changes and record the time taken for completion of analysis of this task.

4.4 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 requirements evolution 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.

4.5 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.

4.6 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 speed [T_{CS}] for the VT group that uses the tool and no_VT group that does the task manually.

$$T_{CS}[\text{VT Group}] = T_{CS}\,[\text{no\_VT Group}] \qquad (1)$$

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

$$\Delta P\,[\text{VT Group}] = \Delta P\,[\text{no\_VT Group}] \qquad (2)$$

Results

Independent 2 sample t tests were conducted to determine the statistical significance between the two groups. The level of significance for the test

Table 1. Computed P Values

Task	Variable	Treatment	Mean	SD	t-Test
All	TCS	No VT Group	70.2	18.71	p<0.0001
		VT Group	29.81	20.69	
Task 1	TCS	No VT Group	59.38	5.85	p<0.0001
		VT Group	4.75	1.17	
Task 2	TCS	No VT Group	58.15	7.60	p<0.0001
		VT Group	31.6	8.95	
Task 3	TCS	No VT Group	93.28	12.41	p<0.0001
		VT Group	53.1	4.86	

was set to α = 0.05.The calculated p values for the t tests are given in Table 1.

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 .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 Task1 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. 20 students were not able achieve the outcome of Task 1 as they were having difficulty in understanding a requirement revision from a new requirement.

Task 2 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 3 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. For VT group students the tasks were relatively easy to complete as the contents were already loaded in the relational tables. VT group students spent their time only on analysis of the change rather than locating and organizing documents. 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 Fig.4. Establishing how a requirement got revised across many change requests was a very tedious task for no_VT Group students. It was found that 20 students from the no_VT group did not arrive at the correct results. Task 3 was also difficult for no_VT group students as they need to spend a lot of time in locating and organization of content. VT group students were more accurate on the tasks due to the information presented to them in the form of models. Precision ΔP is calculated for each group as shown below

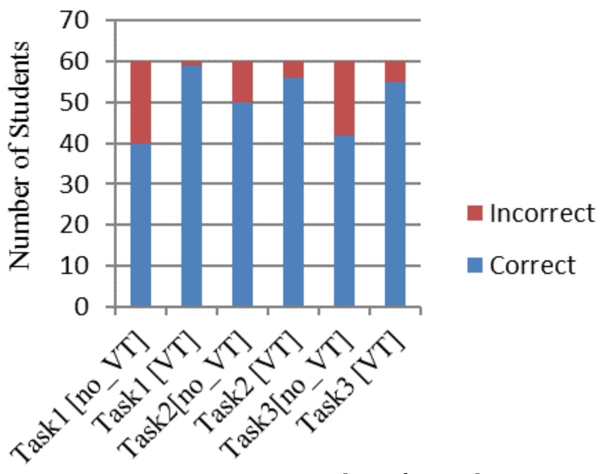


Figure.4 Precision analysis for Tasks

Precision ΔP = Number of tasks completed correctly/Total number of tasks (3)

Total Number of tasks in each group = Number of Tasks * number of subjects in that group (4)

Total Number of tasks in each group = 3 * 60 = 180 Tasks.

Total number of Tasks completed incorrectly for no_VT group = 48 Tasks (Task 1[20], Task 2[10], Task 3[18]).

Total Number of Tasks completed correctly for no_VT group = 132 Tasks (Task 1[40].Task 2[50], Task 3[42]).

Total number of Tasks completed incorrectly for VT group = 10 Tasks (Task1 [1], Task 2[4], Task 3[5]).

Total Number of Tasks completed correctly for VT group = 170 Tasks (Task 1[59].Task 2[56], Task 3[55]).

$$\Delta P\ [\text{no\_VT}] = 132/180 = 73\% \quad (5)$$

$$\Delta P\ [\text{VT}] = 170/180 = 94\% \quad (6)$$

From our experiment we were able to conclude that the precision for VT group is 21% higher than the no_VT group.

The results prove that tracing artifacts for evolving requirements in the maintenance phase indeed helps novice engineers to perform maintenance tasks more accurately. The time taken for the completion of the tasks is also reduced due to the effective organization and traceability of contents in the tool. This tool can also address the problems faced by globally distributed teams with respect to tracking changes to its artifacts. Many of the earlier research was focused on effectiveness of traceability on maintenance or the effectiveness of trace link visualization, where as we have demonstrated an approach that not only traces but also enables visualization of trace links for an evolving requirement.

5 Conclusion

In this paper, we have addressed the problem of traceability management among artefact versions created for evolving requirements. 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 .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 project is still under development and our initial research on the traceability of artefact 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 surround the maintenance of evolving systems and we believe our completed tool will be great interest to software maintenance researchers.

References

[1] C.J. Satish and T. Raghuveera, "Visualizing object oriented software using virtual worlds", In*: Proc. of the 4th WSEAS International Conf.*

on Software Engineering, Parallel & Distributed Systems. World Scientific and Engineering Academy and Society (WSEAS), Salzburg, Austria, Article No 3, 2005.

[2] D. Sumita, G. Lutters, and B. Seaman, "Understanding documentation value in software maintenance", In: *Proc. of the 2007 Symposium on Computer human interaction for the management of information technology, ACM*, Cambridge, Massachusetts, Article No 2, 2007.

[3] A. Erik, L. C. Briand, S.E. Hove, and Y. Labiche, "The impact of UML documentation on software maintenance: An experimental evaluation", *IEEE Transactions on Software Engineering,* Vol.32, No.6, pp. 365-381, 2006.

[4] C.J. Satish and M. Anand, "Software Documentation Management Issues and Practices: A Survey", *Indian Journal of Science and Technology,* Vol.9, No.20, pp.1-7, 2016.

[5] A.S. Khan and M.K. Mattsson, "Management of documentation and maintainability in the context of software handover", In: *Proc. of the Computing Technology and Information Management (ICCM), 2012 8th International Conference on.* Vol. 1, pp. 238-243, 2012.

[6] W.L. Miller, L.B. Compton, and B.L. Woodmansee, "Assuming Software Maintenance of a Large, Embedded Legacy System from the Original Developer", In: *Proc. of the Software Maintenance (ICSM), 2013 29th IEEE International Conference on, IEEE,* pp. 552-555, 2013.

[7] R.P. Gohil, S. Bhattacharya, and R. Chauhan, "Requirements Change Impact Analysis Using Event Based Traceability", *Lecture Notes on Software Engineering Vol.*4, No.3, pp. 162-168, 2016.

[8] J. Cleland-Huang, O.C.Z. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, “Software traceability: trends and future directions", In: *Proceedings of the on Future of Software Engineering, ACM*, pp. 55-69, 2014.

[9] R. Wohlrab, J.P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, “Collaborative traceability management: Challenges and opportunities”, In: *Proc. of the Requirements Engineering Conference (RE), 2016 IEEE 24th International IEEE*, pp. 216-225, 2016.

[10] T. Shah and V.S. Patel, "A review of requirement engineering issues and challenges in various software development methods", *International Journal of Computer Applications* Vol.99, No.15, pp. 36-45, 2014.

[11] H. Ahmed, A. Hussain, and F. Baharom, "Current Challenges of Requirement Change Management", *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* Vol.8, No.10, pp. 173-176, 2016.

[12] Z.S.H. Abad, M. Noaeen, G. Ruhe, "Requirements Engineering Visualization: A Systematic Literature Review", In*: Proc. of the Requirements Engineering Conference (RE), 2016 IEEE 24th International. IEEE,* pp.6-15, 2016.

[13] A. Rodrigues, M. Lencastre and A. de A. Gilberto Filho, "Multi-VisioTrace: Traceability Visualization Tool”, In: *Proc. of the Quality of Information and Communications Technology (QUATIC), 2016 10th International Conference on the. IEEE,* pp. 61-66, 2016.

[14] P. Liang, P. Avgeriou, and K. He, "Rationale management challenges in requirements engineering", In: *Proc. of the Managing Requirements Knowledge (MARK), 2010 Third International Workshop on. IEEE,* pp.16-21, 2010.

[15] P. Mäder and A. Egyed, “Do developers benefit from requirements traceability when evolving and maintaining a software system?”, *Empirical Software Engineering,* Vol.20 No.2, pp.413-441, 2015.

[16] K. Jaber, B. Sharif and C. Liu, "A study on the effect of traceability links in software maintenance", *IEEE Access* Vol.1, pp. 726-741, 2013.

[17] A. Marcus, X. Xie, and D. Poshyvanyk, "When and how to visualize traceability links?", In: *Proc. of the 3rd international workshop on Traceability in emerging forms of software engineering. ACM*, pp. 56-61, 2005.

[18] X. Chen, J. Hosking, and J. Grundy, "Visualizing traceability links between source code and documentation", In: *Proc. of the 2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, pp. 119-126, 2012.