EVALUATING THE USABILITY OF COLLABORATION TOOL FOR SOFTWARE INSPECTION PROCESS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Mounisha Kasireddy

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

April 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Evaluating the Usability of Collaboration Tool for Software Inspection Process

By

Mounisha Kasireddy

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia
Chair

Dr. Oksana Myronovych

Dr. Limin Zhang

Approved:

04/05/19
Date

Dr. Gursimran Walia
Department Chair

ABSTRACT

To develop a good software product, the first and foremost step to master is creating complete and accurate software requirements specification document. Requirements documents are created from through research and are documented for inspection. The inspection team work on finding defects at an early stage, to avoid defects passed on to design team, reduce software development time and improve the product quality. Inspection process, when done in conventional method is labor intensive, cost ambitious and takes overwhelming time. But the benefits of inspection outweigh the difficulties. Software Collaboration tool is one such tool to improve inspection process, help produce complete and accurate SRS document. This paper reviews newly developed Software Collaboration tool and conducts a user acceptance study to evaluate tool's quality attributes, functionalities, experience, usefulness and collect suggestions for future development. This paper employs survey research to determine whether the tool characters and functionalities meet user needs.

ACKNOWLEDGEMENTS

I would like to thank my adviser Dr. Gursimran Walia for his thorough guidance, support during this paper and making the whole journey a sweet memory. I have gained immense knowledge and experience working with you. Thank you for being very flexible with the time, generous support, coaching, and teaching me that a simple genuine smile can go a long way. Thanks again for being a great mentor.

I would also like to thank my graduate committee Dr. Oksana Myronovych and Dr. Limin Zhang for being very helpful and encouraging during this process. Thank you for your motivations and sharing defense tips, you have been very kind and generous towards me.

Finally, I would like to thank the North Dakota State University, Computer Science Department Faculty and Staff for your complete support during the past two years.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

1. INTRODUCTION 1

2. BACKGROUND 4

2.1. Inspection Process (Fagan Inspection Process) 4

2.1.1. Fagan Inspection – Operations and Roles 5

2.1.2. Fagan Inspection – Advantages and Disadvantages 7

2.2. Current Inspection Process Support Tools 7

3. SOFTWARE COLLABORATION TOOL 10

3.1. Key Functions 10

3.2. Software Collaboration Interface 11

4. EXPERIMENT METHOD 14

4.1. Research Questions and Hypotheses 14

4.2. Survey Questions 15

4.3. Survey Participants 16

4.4. Software Requirement Artifacts 17

4.5. Experiment Procedure 17

4.6. Data Collection 19

5. DATA ANALYSIS AND RESULTS 20

5.1. Software Collaboration Tool Quality Attributes and Functionalities 20

5.2. Experience using Software Collaboration Tool for Inspection Process 25

5.3. Enhancement Suggestions 28
6. DISCUSSION OF RESULTS 29
6.1. Summary of Results 29
6.2. Improvements to Software Collaboration tool 32
6.3. Threats to Validate 33
6.4. Future Work 33
7. CONCLUSION 34
REFERENCES 35

LIST OF TABLES

Table Page

1. Experiment Teams 17

LIST OF FIGURES

Figure		Page
1.	Michael Fagan's Process- Advances in Software Inspection (1976)	4
2.	Software Collaboration Tool Login Page	11
3.	Software Collaboration Tool Home Screen	12
4.	Software Collaboration Tool Annotations Screen	13
5.	Software Collaboration Tool Consolidate Fault List Screen	13
6.	Experiment Flow Chart	18
7.	Software Collaboration Tool Non-functional Parameters	22
8.	Software Collaboration Tool Functional Parameters	23
9.	Software Collaboration Tool SRS Document Quality	24
10.	Software Collaboration Tool Overall Quality	25
11.	Software Collaboration Tool User's Experience	26
12.	Software Collaboration Tool Usefulness in Software Inspection Process	27
13.	Recommend Software Collaboration Tool to Others	28

1. INTRODUCTION

To develop a good software product, the first and foremost step to master is creating complete and accurate software requirements specification document [1]. The benefits of having gathered quality product requirements are huge. To list a few, saves time to identify defects/faults after design, reduces time spent to correct these defects, remove/reduce tool evolution stage after deployment, hence reducing cost of developing the product and so running an efficient software development process. But, to build a fault free requirements document it is compulsory to review the requirements thoroughly at an early stage. And often, this step requires a group of reviewers spend time, effort to inspect the documents, find the defects and correct them, and/or some software improvement tool to filter the defects out of the requirements document.

Researchers have tried for years to improve the quality and accuracy of software product requirements. Many methods were explored and tested to add more value to requirements documents by eliminating defects. Methods such as practitioner's approach [3], checklists [4,5], testing, quality assurance, quantifiable improvement [7] and prototyping [6]. All methods have produced some improvements to the requirements but none of them have made it to the table due to facing the difficulties of unable to detect the faults that were introduced after the approach and have become harder to resolve at a later stage in the development process [9] or for being not able to justify the cost to results [8, 10].

Among all the methods tried, Fagan's Inspection method has success stories with reducing the defects by about 90% in one case and more than 50% in two other cases tested in a three-case study experiment [1]. Fagan's method has been extensively used in requirements inspection process since its introduction and it's safe to say that this process can detect defects at any step of the software development lifecycle and is justifiable on cost versus benefit measure [1].

Fagan Inspection has huge benefits in requirements review stage, but it is noted that the process is time consuming and labor intensive. So, to further support the software development industry, make the inspection process less labor and cost intensive. Researches are working to streamline the process and to introduce tool support/automation to reduce time and effort. Thus, making the process more valuable and time saving. Some of the tools used to improve inspection process are ICICLE – Intelligent Code Inspection in a C Language Environment [19, 20], Scrutiny [21], Collaborative Software inspection [22] and many more. Each of these tools have their own merits as described in the research paper "A Review of Tool Support for Software Inspection" [14]. More research is being led to build supporting tools for Fagan's inspection process.

This paper performs a brief study on the available tools in support of inspection process and introduces newly developed Software Collaboration tool. Collaboration tool is designed to ease the inspection process by streamlining the preparation stage, inspection meeting and generating consolidated fault list for effortless review. Collaboration tool's functionality and interface is described in the background. An elaborate study is conducted in this research through survey questionnaire to rate Software Collaboration tool's quality attributes, functionality, usefulness, experience and collect suggestions to further improve the tool. The goal of this research is to evaluate user perception of the Software Collaboration tool in terms quality attributes (usability, performance, look & feel, availability, reliability, portability), functionalities (finding SRS documents, SRS document quality, adding faults, removing faults), user experience, and usefulness and future improvement areas.

In the next section, this paper will review the Fagan's inspection process, its benefits and the need to streamline the process. Current tools to support inspection process will be discuss in subsequent sections before introducing Software Collaboration tool, features and interface in

chapter three. Chapter four will review the need for the survey, advantages and few steps taken to reduce the survey weakness. Experiment details are also provided in section 4. Collected data from the survey results are showed in section 5 with results discussion in chapter 6, and, finally conclusions.

2. BACKGROUND

This section reviews the original Fagan Inspection Process, advantages as well as downsides. Further, this section will introduce some of the supportive software inspection tools currently being used in the software development stages.

2.1. Inspection Process (Fagan Inspection Process)

Introduction of Fagan inspection process into software development industry goes back to 1972, when Michael Fagan at IBM used a cyclic continuous improvement process to improve software requirements specification or code. Fagan Inspection process contains six basic operational steps to inspect any software development requirements or code [2]. These operations are as follows Planning, Overview, Preparation, Inspection, Rework and Follow-up. These operations are managed and controlled by a small team of Inspectors, Moderator, Reader, Recorder and Author.

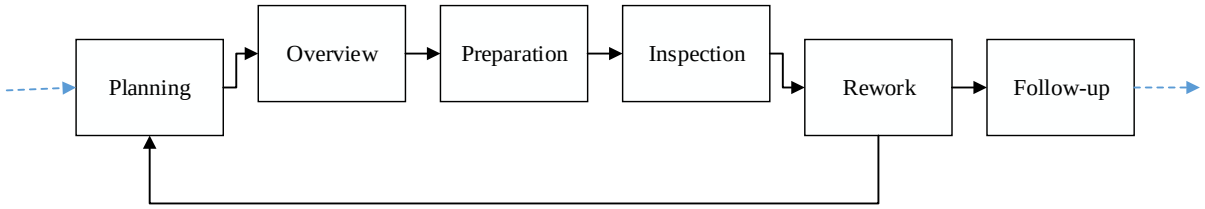


Figure 1. Michael Fagan's Process- Advances in Software Inspection (1976)

Overtime, the process has been tweaked differently by different industries based on their own necessity but the core method is kept intact to serve the review process [4,6]. This method has transformed software development industry due to its simple, efficient and economical method of finding errors in software creation [1]. This method soon picked up the name 'Fagan Inspection' and has found its way to all corners of the industry.

2.1.1. Fagan Inspection – Operations and Roles

Fagan's process comprises of six operational stages and five individual team roles as explained below.

Planning: Planning is the first step of the process, during this stage general checks are made to ensure document or code to be inspected meets the system entry criteria. Then a group of people are selected and given the roles of inspectors, moderator, reader, recorder and author. Planning stage is used for scheduling meetings between the team, setting timelines, select stakeholder's distribution list and to determine if an overview operation is required.

Overview: To use overview stage is determined in the planning stage. So, this stage is an elective step. If selected, this stage is used to update inspection team with project background and purpose. If the team is already aware of this information overview stage can be skipped.

Preparation: This is where the actual document inspection takes place and so is the most important step of the process. Inspectors break out to work independently, review the product and identify potential defects. These defects are well documented by the individual inspectors to reviewed with the whole team in the next step.

Inspection: Preplanned meeting from the planning stage where all the members of the team meet to review defects found during the preparation stage. During this operation, defects are record, categorize, and organize, but not resolve. Defects are noted by a recorder and are passed on to the author.

Rework: Recorded defects are passed to the author to work or take on corrective requirements for the product. If there are many defects or the moderator suggests the need to regroup and review the product again, the team moves back to planning to inspect the product again. This process goes on until the team is happy with the product requirements.

Follow-up: Follow-up is between the author and moderator to discuss next steps or sometimes the whole team. This stage is to verify that all defects are corrected and no new threats are introduced.

Below are the team members/roles who would participate in the six step Fagan Inspection operations.

Inspectors: Everyone in the inspection team is an inspector. Apart from working on reviewing the documents and finding defects, inspectors also act as author, moderator, reader and recorder, as necessary.

Author: Author is the primary inventor of the product requirements document. He is responsible to give all the other inspectors a complete background of the product and a training if needed. Also, the author is responsible to correct the defects found during the inspection process. It also helps if the author is also involved in discussions with the customer business when creating requirements.

Moderator: Moderator role is responsible to make sure the entire inspection process runs smooth and successful. Selecting the team, schedule the meetings and ensuring to capture most of the faults during the process. Since this is an important operation, companies usually provide training and build the portfolio to handle this work.

Reader: Reader is responsible for running the inspection meetings. As the name suggests reader reads the product requirements and the associated defects to open the floor for discussions. It would be beneficial for the organization if the reader is someone who represents the next step in the software development process – design phase. Reading the documents and understanding is helpful to become acquainted with the requirements.

Recorder: Recorder is the one taking notes during the inspection operation. He is responsible to note all the defects found, discussions and record them in an organized and timely manner.

2.1.2. Fagan Inspection – Advantages and Disadvantages

As explained previously, the advantages of the Fagan Inspection process are to produce complete and accurate requirements documents, find and eliminate defeats at an early stage, (Fagan Inspection process can be used at any step of Software development lifecycle to review document, code or tool), reduce/eliminate time spend in fixing defects. Thus, saves time and cost to increase productivity, promote efficiency.

Like with any entity, there is no absolute perfect method. Fagan Inspection methods also shares this suspicion. The process is considered time consuming and labor intense due to having to spend tedious amount of time to maliciously review documents/code, so the cost. However, the benefits achieved by this process outweighs this limitation. In the past decade, researches have worked and have been working to streamline and/or automate this process to address the deficiency.

2.2. Current Inspection Process Support Tools

This section provides a brief review of the available inspection process support tools. Each tool described here uses a different tactic but, the common goal is to improve inspection process. All support tool models use some distinction of the Fagan's process.

Collaboration Software Inspection (CSI) [22]: This tool employs Humphrey [24] or Yourdon [23] model. But one clear difference between Fagan Process and CSI is, the inspectors provide a copy of their defects report from individual preparation to the author before the inspection meeting. The author then reviews the defects on his own before reviewing them in the

inspection team meeting. CSI assigns a serial number to each line in the document, these lines can be annotated when selected to insert a fault and a notepad is available to leave comments. Author will be able to assort all the faults from the tool into one fault list to accept, reject, rate and categorize.

Scrutiny [21]: A four stage inspection method similar to Fagan's six stage operation. Stage one is called initiation to review requirements document and know background of it. Second stage is called preparation which is like Fagan's model. Third stage is the inspection stage to review faults in a meeting setting and final stage is called completion to make corrections to requirements and follow-up just like in Fagan Inspection process. Scrutiny only supports text format and within the tool inspectors can add faults using annotation to record defects, questions or remarks. This tool creates an automated defects list from the inspection meeting review. However, the tool does not assist any supporting documents or checklists and all defects must be found manually.

ICICLE (Intelligent Code Inspection in a C Language Environment) [19, 20]: This tool is an automated inspection support for C Language Code inspection. This tool cannot be used for general inspection process but works great to identify common defects in a C code review. The interface can be split to see two parts of the code at one time and options either to flag a code line or comment specifying the defect or make notes. ICICLE also allow the users to cross reference the notes/defects across document and allows the author to accept or reject the defects during inspection process.

InspeQ (Inspecting software in phases to ensure Quality): is a multi-phased inspection technique developed by Knight and Myers [25, 26] to provide an inspection process that is "rigorous, tailored per request, automated and efficient in using resources". Apart from providing being able to view multiple segments of the document at a given instance like other tools, InspeQ

allow the inspectors to search the document using keywords. Inspectors can record defects, index the line numbers and write notes with in the defects for references. There are several other advantages to the tool such as using checklists to ensure all parts of the document are reviewed while marking each line with a status. These checklists are supported by standard displays and highlights option to focus on specific function code throughout the document while checking for defects. However, InspeQ is built to support individual inspectors and cannot be used to consolidate defects or be presented for group review or during inspection meeting operation.

Collaborative Software Review System (CSRS) [27, 28, 29]: This tool is to support FTArm (Formal Technical Asynchronous review method) [30] also known as Asynchronous Fagan Inspection method. In this method, documents and annotations are stored as a series of nodes in the database. Annotations can be added as an issue, an action or a comment publicly to all inspectors. This visibility allows the inspections to review nodes unaddressed, helps moderator plan the inspection stage based on node status. Additionally, CSRS also sends notifications to the inspectors when a new defect or comment is posted and helps moderator by producing a master defect list in a LaTeX format report.

All the tools mentioned above has merits of its own, consequently good in their own way to support inspection process [14]. There are many notable functions in each tool to help improve the inspection process to make it more organized and efficient. There is no equal scale to vote all the tools on one measure, hence it is impossible to say which tool is the best.

Also along this corridor is a new tool “Software Collaboration tool” for inspection process. In the next section, discussions will be focused on the Software Collaboration Tool, key functions and user interface.

3. SOFTWARE COLLABORATION TOOL

Software Collaboration tool for inspection process is somewhat similar to the tools explained in section 2.2. There are some qualities to the software collaboration tool that are similar and some functionalities previously not covered in other software inspection tools, at the same time a few of the services covered by the other tools are missing from Software Collaboration too. Down below are some of the functionalities available in Software Collaboration tool.

3.1. Key Functions

Software Collaboration tool is developed to streamline Fagan Inspection process. The tool is predominantly used during preparation and inspection meeting operations. When the requirements documents are uploaded, the inspectors can review the documents independently while entering their comments, faults and author/moderator can consolidate the faults to create master list during or after the inspection meeting. During the meeting, inspectors have access to their own documents and faults list in the tool. Inspectors can access their documents and fault list online via tool if the meeting is conducted and attended by individuals from multiple locations.

Software collaboration tool supports documents from different formats like file types, charts, figures, use case diagrams etc., Within the tool SRS document can be opened in multiple tab, so different parts of the document can be simultaneously reviewed on multiple screens. A time log can be maintained in the tool to record time when the fault was entered. Faults list are kept private; each inspector can only view his/her own faults. A serial number is devoted to each fault to keep track of the number of faults entered by inspectors. Users can create faults and delete it if deemed unnecessary or redundant. Faults can be classified into ambiguous information, omission, incorrect facts, inconsistent information, extraneous and miscellaneous types to give moderator/author clear understanding of fault specifics.

List of Key Features:

- Various document formats are accepted in the tool.
- Multiple screens can be setup to review different parts of the document.
- Faults can be classified into ambiguous, omission, incorrect facts, inconsistent information, extraneous and miscellaneous categories.
- Manual time log can be maintained.
- Individual inspector's faults list is kept private from others.

3.2. Software Collaboration Interface

In this section, Software Collaboration tool interface with home screen, annotations screen and consolidate fault list are provided. Below are some screenshots depicting the tool interface. Figure 2 is the login screen for users to login using their ID/password and to the bottom right of the dialog box is a button to sign up for the tool.

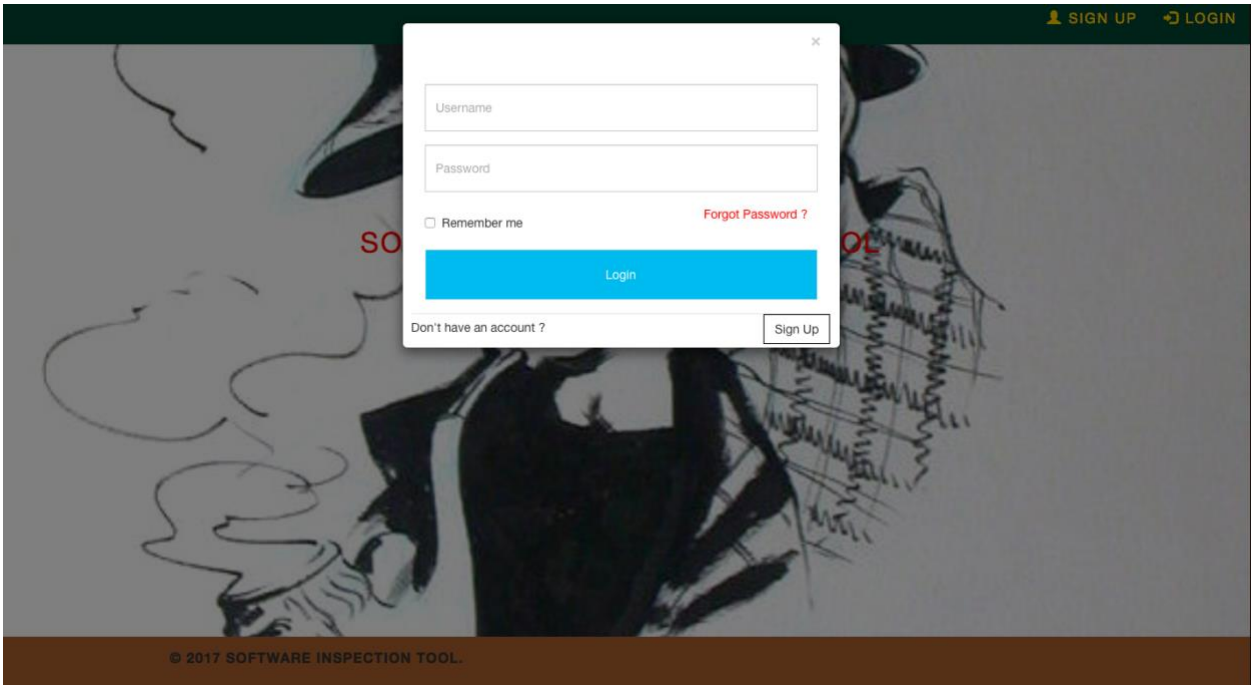


Figure 2. Software Collaboration Tool Login Page

Figure 3 is the tool's home screen with teams Software Requirements Specification documents. Tool admins have visibility to all the team's documents but inspectors only have visibility, read and write access to their own documents. Tool admin/moderator can grant access to users based on the team they are part of.

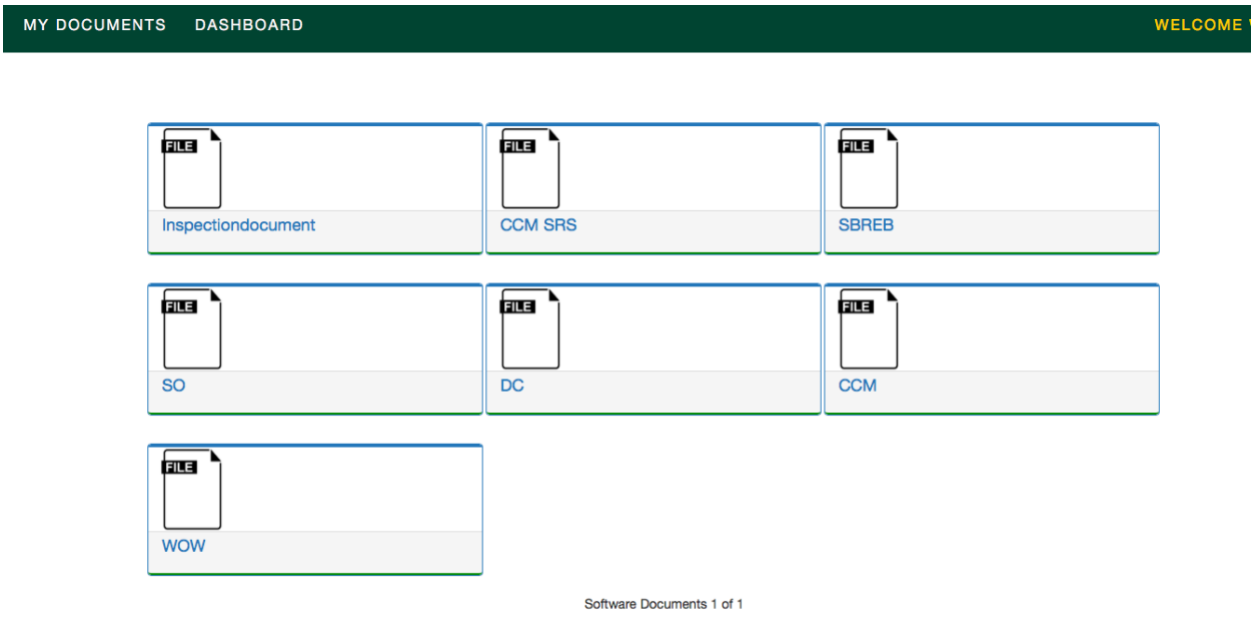


Figure 3. Software Collaboration Tool Home Screen

Figure 4 is the Annotations page where inspectors can enter the defects or comments. Inspectors also have options to edit existing faults or delete redundant annotations. Inspectors can also enter time manually.

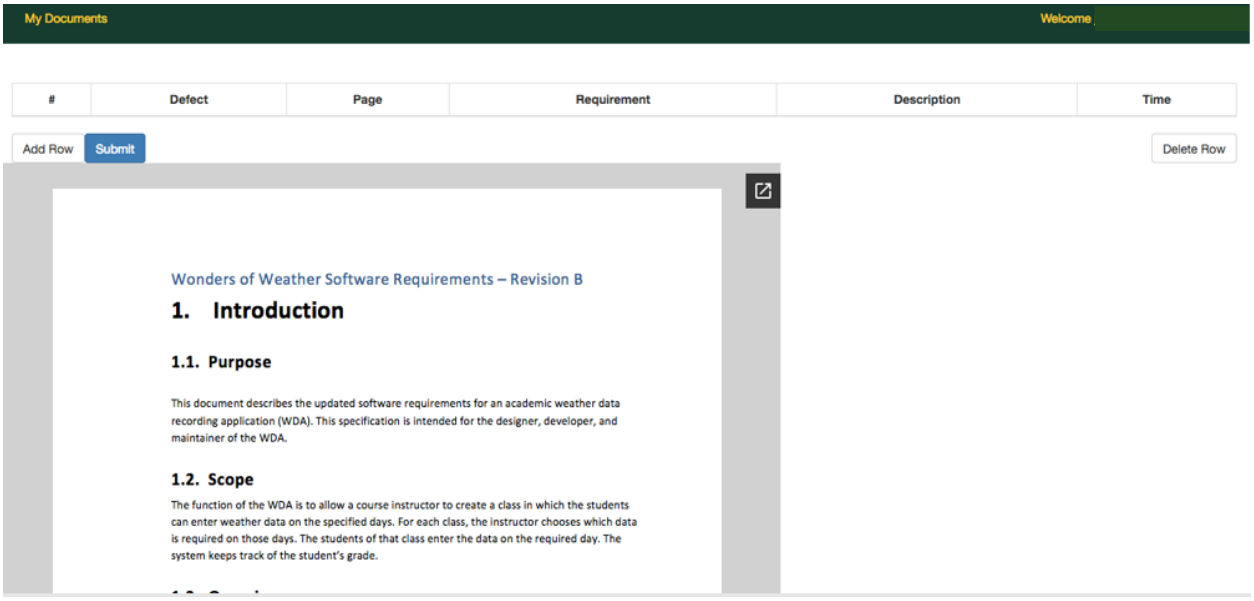
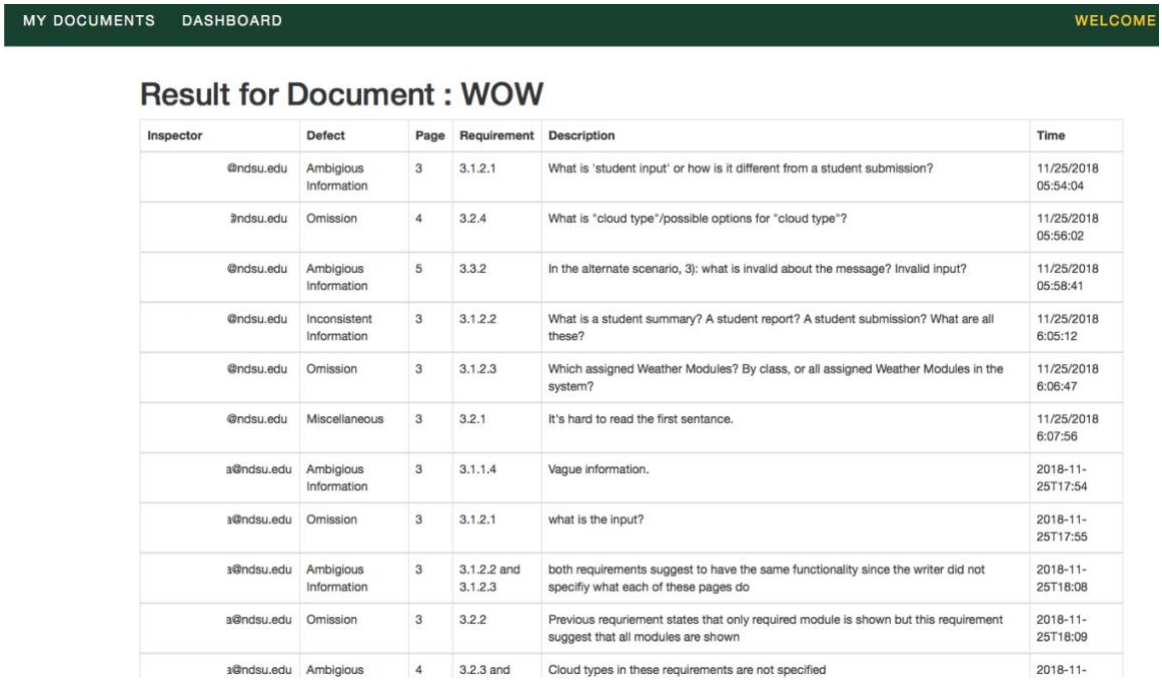


Figure 4. Software Collaboration Tool Annotations Screen

Figure 5 is the final list of all defects visible to the moderator/author to view. Table provides inspector ID, defect type, page number, requirements number and description.



Inspector	Defect	Page	Requirement	Description	Time
@ndsu.edu	Ambigious Information	3	3.1.2.1	What is 'student input' or how is it different from a student submission?	11/25/2018 05:54:04
ɜndsu.edu	Omission	4	3.2.4	What is "cloud type"/possible options for "cloud type"?	11/25/2018 05:56:02
@ndsu.edu	Ambigious Information	5	3.3.2	In the alternate scenario, 3): what is invalid about the message? Invalid input?	11/25/2018 05:58:41
@ndsu.edu	Inconsistent Information	3	3.1.2.2	What is a student summary? A student report? A student submission? What are all these?	11/25/2018 6:05:12
@ndsu.edu	Omission	3	3.1.2.3	Which assigned Weather Modules? By class, or all assigned Weather Modules in the system?	11/25/2018 6:06:47
@ndsu.edu	Miscellaneous	3	3.2.1	It's hard to read the first sentance.	11/25/2018 6:07:56
a@ndsu.edu	Ambigious Information	3	3.1.1.4	Vague information.	2018-11-25T17:54
ɜ@ndsu.edu	Omission	3	3.1.2.1	what is the input?	2018-11-25T17:55
ɜ@ndsu.edu	Ambigious Information	3	3.1.2.2 and 3.1.2.3	both requirements suggest to have the same functionality since the writer did not specifiy what each of these pages do	2018-11-25T18:08
a@ndsu.edu	Omission	3	3.2.2	Previous requriement states that only required module is shown but this requirement suggest that all modules are shown	2018-11-25T18:09
ɜ@ndsu.edu	Ambigious	4	3.2.3 and	Cloud types in these requirements are not specified	2018-11-

Figure 5. Software Collaboration Tool Consolidate Fault List Screen

4. EXPERIMENT METHOD

Surveys have been identified as simple but efficient data collection instrument to understand distinguishing characteristics using a group of people. Surveys are used to rate specific aspects of a research population, results from the survey are collected from people. Therefore, they are subjective and survey findings can be generalized to a larger population [32]. Therefore, Survey is used in this research to evaluate Software Collaboration Tool key performance parameters as identified by the researcher. However, as mentioned by Bell 1996 paper [31], Surveys have weaknesses and this SC Tool research has taken few steps to minimize this survey weaknesses. Three major weaknesses listed on Bell 1996 paper [31] are population biases – lack of response from participants or nature and accuracy of the results, intentional misreporting, and participants may have difficulty assessing their own behavior or have poor recall of the circumstances surrounding their behavior. To overcome these weakness participants were encouraged to complete the survey within one week of project completion to maintain fresh, accurate memory of the tool behavior and thereby reducing most of the above stated survey weakness. Intentional misreporting of behaviors by participants are outside this research's control and are consider as outliers from the survey results.

4.1. Research Questions and Hypotheses

As mentioned previously, this researchers' intentions are to study Software Collaboration Tool quality attributes, functionalities, user experience, and usefulness. Motivation for this subjective study is to address three prime research questions. The primary emphasis is to evaluate the Software Collaboration tool quality attributes and functional features. The secondary focus is to study Software Collaboration tool user experience, usefulness. And finally, to understand the possible improvement areas that the tool developers can work on for future development.

Research Question 1: How did participants perceive software collaboration tool in terms of quality attributes (usability, performance, look & feel, availability, reliability, portability) and functionalities (finding SRS documents, SRS document quality, adding faults, removing faults)?

Research Question 2: How did participants perceive collaboration tool's usefulness and overall experience?

Research Questions 3: What are the future areas to improve the Software Collaboration Tool?

4.2. Survey Questions

Survey questions were designed to capture user experience, satisfaction and fulfilment of tools quality attributes, functionalities to determine Software Collaboration tool effectiveness to help customers with software inspection process. Questions were asked to understand time spent reviewing Software Collaboration tool by each individual and their previous experience with any other inspection tools. Participants were also asked to provide suggestions to improve Software Collaboration tool. Participants were asked to respond to the questions immediately after using the tool to retain accuracy of their experience and reduce misreporting due to poor recall of tool behavior. Below are all ten survey questions used during this experiment:

1. Prior to using the "Software Collaboration" tool for review, did you use any other Software Inspection tool?
2. How many hours did you spend reviewing and learning about "Software Collaboration" tool?
3. How satisfied are you about "Software Collaboration" tool on the below parameters: (Usability, Look and Feel, Availability, Reliability, Performance, Portability)?

4. How easy or difficult are the functionalities in "Software Collaboration" tool? (Logging in, finding documents, adding/deleting faults, changing password)
5. How satisfied are you with the quality of SRS document in "Software Collaboration" tool? (document handling)
6. How do you rate your experience on "Software collaboration" tool?
7. How much do you agree with this statement: "Software Collaboration tool is very useful in the Software Inspection process"?
8. How satisfied are you about "Software Collaboration" tool quality?
9. How likely would you recommend "Software Collaboration" to others?
10. Do you have any suggestions for improving "Software Collaboration" tool?

Most questions were framed to capture responses on a 1 to 5 scale to keep the statistical analysis outcome in a standard format. For the last question, text form was provided for the participants to enter their suggestions.

4.3. Survey Participants

All 65 students of 'Principles of Software Engineering' class at North Dakota State University were invited to take the survey due to their equivalent knowledge of software development process. Students have attended three to four months of schooling in the concepts of Planning, Requirements gathering, Software Requirements Specification document, Software Design, Implementation, Software Testing and Debugging, Deployment, and Maintenance which covers overall Software Development Lifecycle stages. All students were in the age group between 18 and 24 and were given a one hour extensive training on how to use Software Collaboration tool. Students were randomly grouped into five teams of 11 to 14 participants. Each team was then

tasked to develop Software Requirements Specification document for different projects assigned by the professor.

4.4. Software Requirement Artifacts

All five teams were asked to create Software Requirements Specification document for their respective systems show in table below (number of participants per team, requirements document description and document size is specified in the below table). Participants have used the same software requirements document created by their team to look for Ambiguous information, Omission, Incorrect facts, Inconsistent information, Extraneous, Miscellaneous faults. Team inspectors have then entered their faults into the Software Collaboration Tool. Team moderator have then compiled all the faults and remove any duplicate or false-positive faults to create master list for their identified system.

Table 1

Experiment Teams

Team Number	Team Name	No. of Participants	Document Name	System Description	Document Size
1	Team 1	14	WOW	Weather Data Recording Application	12
2	Team 2	13	SO	Science Olympiad Scoring System	25
3	Team 3	14	SBREB	Sugar Beet Research and Education Board	7
4	Team 4	13	DC	Dissertation Calculator	29
5	Team 5	11	CCM	Capstone Course Management	30

4.5. Experiment Procedure

Participants were given information about the software development lifecycle and the steps involved to create outstanding software. It was clear that to produce an efficient and well served software the first and most important step of the process was to create the requirements specifications. Participants were educated in both traditional/paper inspection process and

computer based inspection process. They are aware of the background and benefits of computer based inspection process.

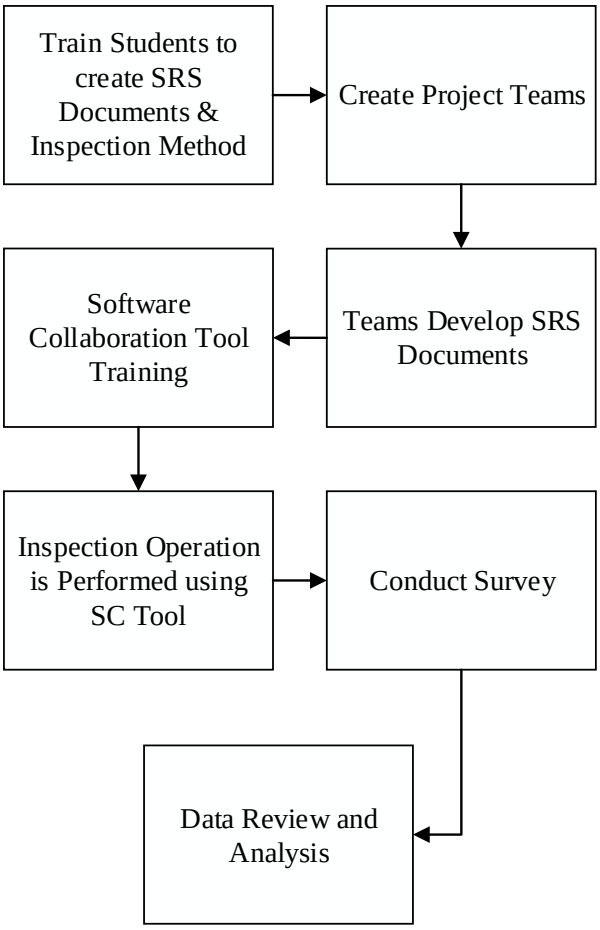


Figure 6. Experiment Flow Chart

All 65 students were given 90 minutes of Software Collaboration tool training after completing the SRS documents. During the training, participants were taught to login to the system, change password and customize per their convenience. Participants were explained about Software Collaboration tool interface and how to navigate through the system. Extensive training was provided on how to upload SRS Document in the tool and access it individually. Participants were allowed to practice entering faults, select fault type, requirement number, page number, description and manual entry time. Later in the training, participants were taught to retrieve

previously entered faults and identify fault type (read the data which is entered). Finally, users were taught to create master list. All participant's questions were answered at end of the session and were asked to contact the trainer or assistant with any questions while working in the tool.

After the training, Software Collaboration tool administrators have created 65 individual user accounts for each participant by using their college ID and unique password for each group. Participants were asked to change their passwords up on their first login. Admins have uploaded 5 team's SRS documents to the tool. Teams were given access, visibility to their own SRS document and other team's documents were not visible to users. Teams have then reviewed their SRS documents, entered faults, other respective information in the tool and generated master list.

4.6. Data Collection

After project conclusion participants were asked to complete a short 10 questions survey listed in 4.2 section to rate tool quality attributes, functionalities, experience, usefulness and their satisfaction. The survey was built on Qualtrics tool [33], which is a tool to conduct surveys and record user experience. Participants were given 14 days to complete the survey.

The questionnaire was designed to convert qualitative information regarding the tool and report on quantitative Likert 5-point scale system. This data are later be converted to present the results. Qualitative tool feedback suggestions were also collected using the survey. All the results will be directly reported to the tool developers for tool enhancement.

5. DATA ANALYSIS AND RESULTS

This chapter provides a detailed analysis of the data collected from the research survey and suggestions proposed by the participants. The analysis performed on the research questions listed in section 4.1 are checked to see how the participants have perceived software collaboration tool and provides the evidence for future direction.

Subjective results from the survey research which are marked on Likert 5-point scale are converted into percentages to represent measures on common proportions. Percentages are calculated by taking the number of selections made per 5-point scale rating on a parameter to the total number of selections made against the same parameter. This percentage conversation allows to compare multiple parameters, ratings to establish clear and measurable goals for this research.

Survey questions from the research are broadly categorized into the segments as stacked below:

- Questions to rate software collaboration tool quality attributes and functional features.
- Questions to rate participant experience and usefulness of the tool.
- Question to collect future enhancement suggestions from the participants

Results of all the questions from survey research are shared in the next sections by the above categories.

5.1. Software Collaboration Tool Quality Attributes and Functionalities

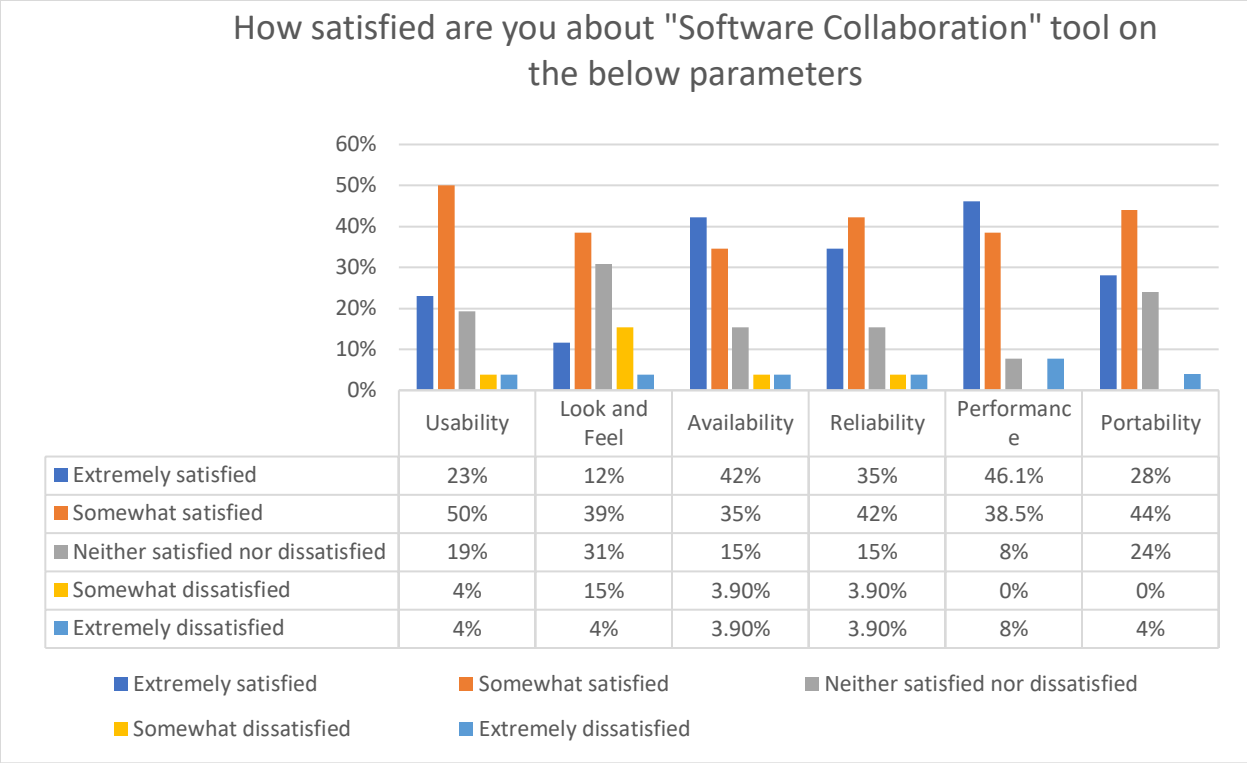
Following queries in the questionnaire are meant to capture Software Collaboration tool usability, performance, look & feel, availability, reliability, portability and other key functionalities like finding SRS documents, SRS document quality, adding faults, and removing faults.

1. How satisfied are you about "Software Collaboration" tool on the below parameters: (Usability, Look and Feel, Availability, Reliability, Performance, Portability)?

As shown in figure 7, the vertical axis depicts the rating percentage and the horizontal axis represents software collaboration tool parameters. From the chart results, majority of the survey population have either expressed extreme to somewhat satisfaction on all the six parameters. Breaking down the results from left to right, the first parameter tool usability has received 50% somewhat satisfied rating and 23.1% are extremely satisfied. 19.2% were undecided on tool usability and 7.8% expressed dissatisfaction.

Tool 'Look and Feel' parameter has received the lowest rating for satisfaction among all six parameters with only 11.6% population showing extreme satisfaction and 38.5% said somewhat satisfied a combined total of 50.1%. Whereas 30.8% users said they were undecided making it the third highest undecided choice for the users, sharing the stage with 'Finding Documents' and 'Changing Password' tool functionality. And, giving it the last position in the overall tool parameters.

The next two parameters in the list – availability and reliability go hand in hand with the overall satisfaction and dissatisfaction ratings. As both have a 76.9% approval rating, 15.3% undecided and 7.8% disapproval. Overall 'Performance' parameter of the survey has received an aggregate total satisfaction of 84.6%. Whereas 7.7% population undecided and extremely unsatisfied, hence topping the charts in tool experience parameters. Portability parameter has received a 28% extremely and 44% somewhat satisfied rating, 24% population was undecided with only 4% extremely dissatisfied.



	Usability	Look and Feel	Availability	Reliability	Performance	Portability
Extremely satisfied	23%	12%	42%	35%	46.1%	28%
Somewhat satisfied	50%	39%	35%	42%	38.5%	44%
Neither satisfied nor dissatisfied	19%	31%	15%	15%	8%	24%
Somewhat dissatisfied	4%	15%	3.90%	3.90%	0%	0%
Extremely dissatisfied	4%	4%	3.90%	3.90%	8%	4%

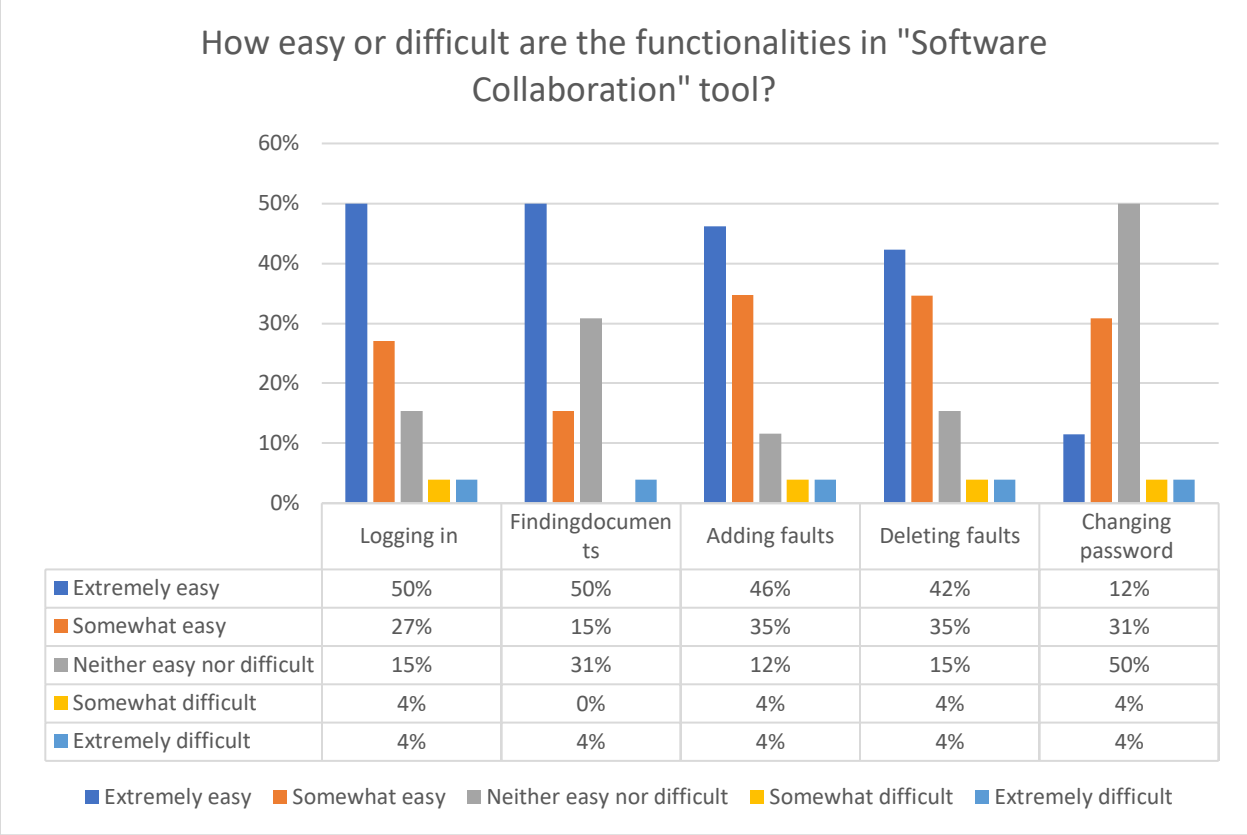
Figure 7. Software Collaboration Tool Non-functional Parameters

2. How easy or difficult are the functionalities in "Software Collaboration" tool? (Logging in, finding documents, adding/deleting faults, changing password)

Participants were asked to rate various tool functionalities to determine the effectiveness of the software collaboration tool. From the survey results, it is quite evident that only 7.8% or less users have expressed any level difficulties with each of five tool functions and that puts the overall tool functionalities in satisfaction zone or users have no opinion.

Going by the individual functionality (left to right in the figure). For 'logging in' functionality, 50% user have said that it was extreme easy for them to 'log in' to the tool and 27% users said it was somewhat easy. The remaining 15.4% participants were undecided and 7.8% had difficulties. Half the participants have alleged it was very easy and 15.4% users have said it was somewhat easy for them to find documents in the system. Almost one third of the users had no

recollection and only 3.9% said it was extremely difficult to find documents. When asked about ‘Adding faults functionality, 80.9% participants have reported it was easy, 11.6% remained undecided and 7.8% said it was difficult to add faults. Deleting faults have a similar rating as adding faults functionality with 76.9% agreeing it was easy, 15.4 undecided and 7.8% said deleting faults was difficult.



	Logging in	Findingdocuments	Adding faults	Deleting faults	Changing password
Extremely easy	50%	50%	46%	42%	12%
Somewhat easy	27%	15%	35%	35%	31%
Neither easy nor difficult	15%	31%	12%	15%	50%
Somewhat difficult	4%	0%	4%	4%	4%
Extremely difficult	4%	4%	4%	4%	4%

Figure 8. Software Collaboration Tool Functional Parameters

Changing tool password functionality has received some discontentment with half the population rating the function neither easy nor difficult. (leading us to assume that users have not changed their initial password while reviewing SRS document)

3. How satisfied are you with the quality of SRS document in "Software Collaboration" tool?

For this survey question, 16% of the participants have said they were extremely satisfied with the SRS document quality and majority of users (64%) have said they were somewhat satisfied. Other 16% had no opinion and 4% has shown dissatisfaction with the document quality.

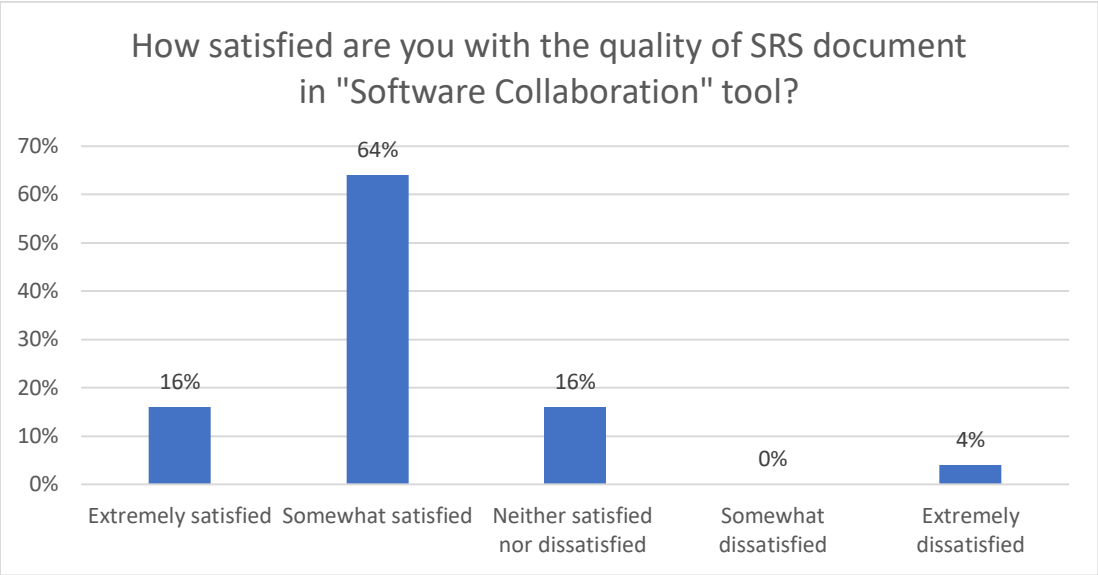


Figure 9. Software Collaboration Tool SRS Document Quality

4. How satisfied are you about "Software Collaboration" tool quality?

Tool general quality survey results are immensely tilted to the satisfied side of the Likert scale with 16.7% population extremely satisfied and 58.3% expressed mostly satisfied ratings. 20.8% of the users have remained in the middle undecided area and only 4.2% have reported some displeasure and no user has expressed complete disapproval with the tool quality.

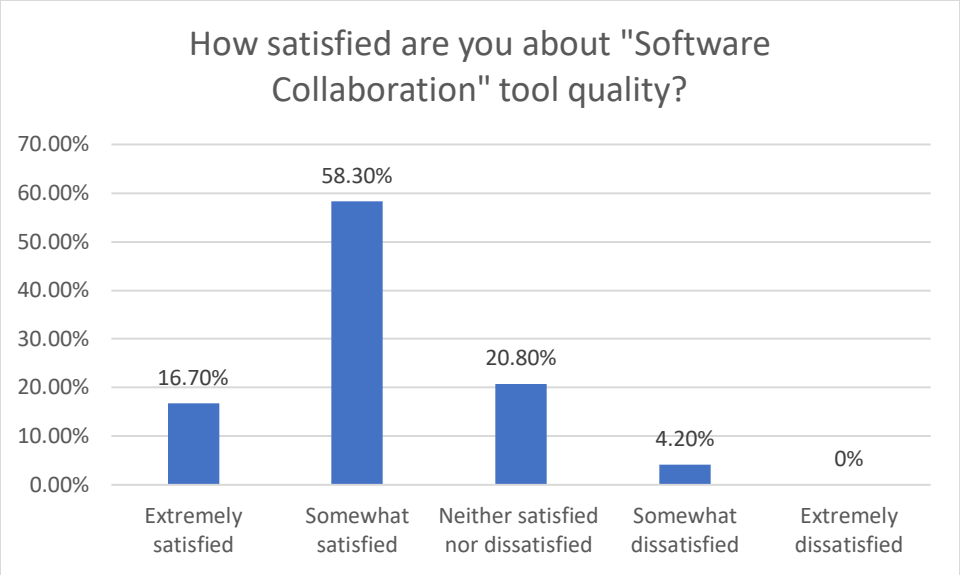


Figure 10. Software Collaboration Tool Overall Quality

5.2. Experience using Software Collaboration Tool for Inspection Process

Overall user experience with software collaboration tool for inspection process was recorded using below five survey questions.

1. Prior to using the Software Collaboration tool for review, did you use any other software inspection tool?

This question allowed researchers to understand participant' level of expertise with the software inspection tools. The survey results show that 96% people were using software collaboration tool for the first time.

2. Survey participants are asked "how many hours did you spend reviewing and learning about Software Collaboration tool?" two thirds of the users have spent between 1 to 2 hours to review the tool and learn about the tool functionality. One quarter of the users have spent less than one hour to learn about the tool. Around 8% of the users have spent more than two hours to learn the tool functionality.

3. How do you rate your experience on "Software collaboration" tool?

This question is to rate the general user experience during their time using the tool. Survey results indicate that 25% of users are extremely satisfied with the tool and about 58.3% users are somewhat satisfied. The rest of the users (16.7%) either have no opinion or were dissatisfied with the tool.

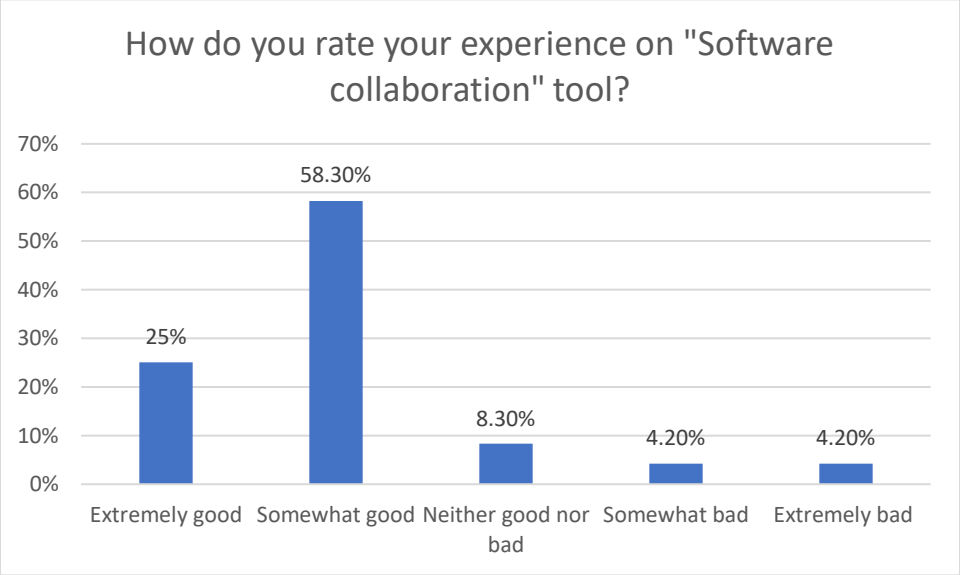


Figure 11. Software Collaboration Tool User's Experience

4. How much do you agree with this statement: "Software Collaboration tool is very useful in the Software Inspection process"?

Participants have strongly (79.2%) agreed that the Software Collaboration tool was helpful in their Software Inspection process. Among the rest, about 12.5% users neither agree or disagree to this question, leaving 8.4% of the population disagreed with the statement.

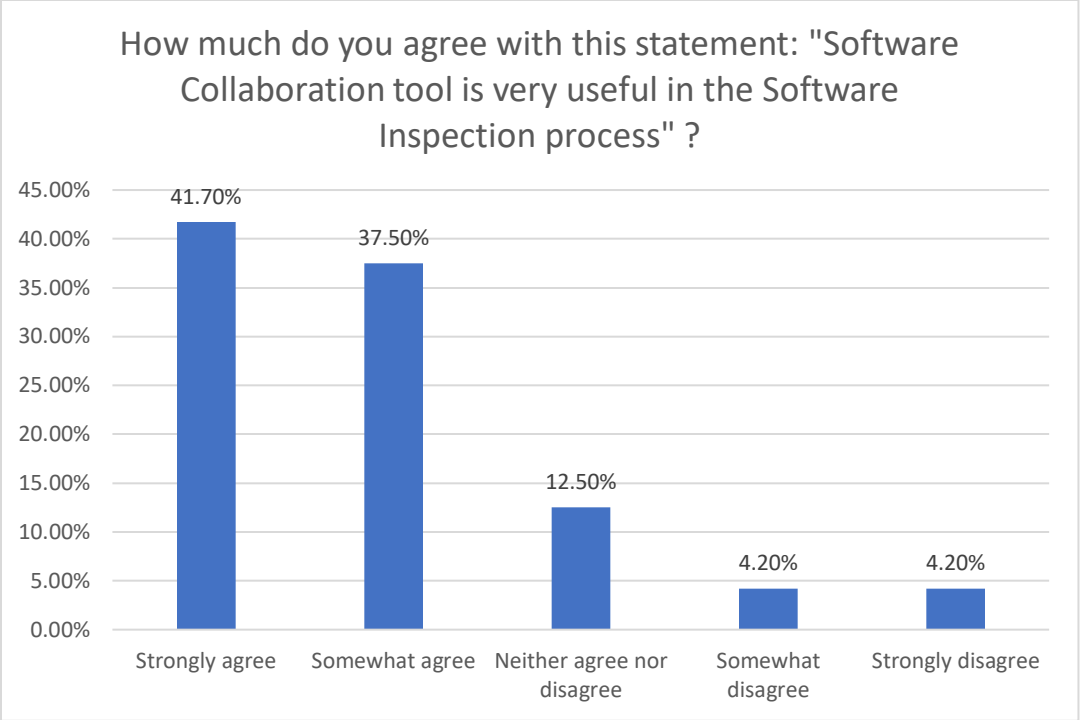


Figure 12. Software Collaboration Tool Usefulness in Software Inspection Process

5. How likely would you recommend "Software Collaboration" to others?

Question was considered to check if the survey participant would entrust the tool with a colleague or associate who is likely to work in the inspection process. 8% of the users have expressed strong likeliness to endorse the tool to others and a majority of 52% have slightly agreed to recommend. Whereas 32% participants were undecided and 8% participants have said they would not recommend the tool.

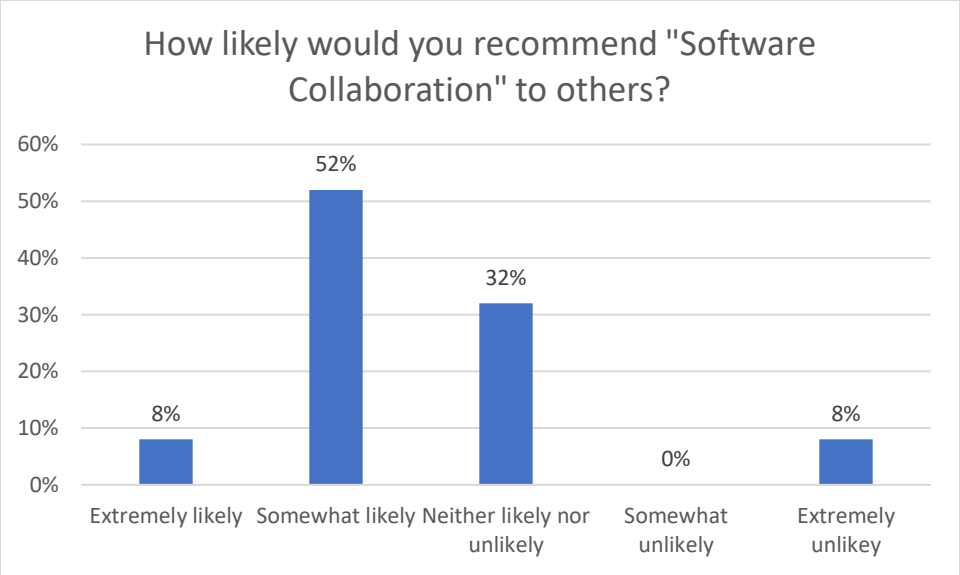


Figure 13. Recommend Software Collaboration Tool to Others

5.3. Enhancement Suggestions

Last survey question, “do you have any suggestions for improving "Software Collaboration" tool?” is to gather firsthand experience and suggestions from the participants regarding the tool after having using the tool for a live project with actual inspection process. The outcome to this research question is, participants have genuinely valued the tool’s support during their software inspection process and have also suggested a few enhancements requests to improve the tool appearance and performance. Participants have also shared tool behavior when working on generating defects list. These suggestions and tool behavior is further discussed in the summary of results and improvements to Software collaboration improvements section.

6. DISCUSSION OF RESULTS

This section summarized the survey results from chapter 5. Each of the three research questions are answered using the subjective survey results. Section also includes improvements to software collaboration tool, threats to validate and future work.

6.1. Summary of Results

In this section, the study discusses the survey results and presents its summary in response to the initially stated research questions.

Research Question 1: How did participants perceive software collaboration tool in terms of quality attributes (usability, performance, look & feel, availability, reliability, portability) and functionalities (finding SRS documents, SRS document quality, adding faults, removing faults)?

Survey results show that participants were satisfied with the collaboration tool and have expressed that using the tool has helped them in the inspection process. More than a landslide worth of Results indicate that major tool parameters and functionalities are satisfying for the customers which are specific for user loyalty. Some of the results from the survey are discussed below to show the tool's eminent magnitude and all the downsides per participants' survey results are also listed below.

- Almost 85% users have liked the tool's performance and were happy to use the tool for inspection process. These participants have said that the tool's capability to perform inspection operations were effective and satisfying. A clear indication that the tool is proficient in performing inspection process.
- 77% participants have expressed their satisfaction with the availability, reliability of the tool. Implying that the tool is consistently operable without any cause for concern and it is dependable/trustworthy in the software inspection process.

- Around 72% users have said the tool was portable and rated high on usability. Telling the tool could be used at different environments/browsers and tool is easy to use.
- 50% users were satisfied with the look and feel of the tool, 30% were undecided and 20% were dissatisfied. Users have also suggested in the feedback field to improve interface. Explaining some need to improve look and feel of the tool.
- Close to 80% of the users have expressed that functionalities listed in the survey question 4 were easy to use. Consequently, explaining that it is easy to find documents, enter faults or delete faults making it an easy tool to navigate and operate for inspection process.
- 50% users were undecided when asked if changing the tool password was easy. Indicating, that they have either not attempted changing password or have attempted changing but were undecided with this functionality.
- On tool functionality question, of all the participants only 8% have expressed the tool functionalities were difficult. Which is a good indicator to show that the software collaboration tool functionalities are helpful.
- 80% users were satisfied with SRS document quality, signifying that the Software Collaboration tool has accepted multiple text formats, pictures, charts and table information.
- Participants have rate the tool high (75%) on overall quality. Quality of the tool is important to drive tool popularity and loyalty in the software development industry. Having approval from three quarters of the total population is a certificate of fineness.

Research Question 2: How did participants perceive collaboration tool's usefulness and overall experience?

- 83% have rated good experience implying it was easy or smooth running the tool without difficulties.
- 60% people are willing to recommend the tool to others and 32% undecided – shows participants loyalty towards tool.
- Four out of every five users have said the tool is very helpful in the inspection process.

Research Questions 3: What are the future areas to improve the Software Collaboration Tool?

Last question of the survey is for the participants to suggest improvements to the software collaboration tool. This question was intentionally left open ended to collect participant's opinion and suggestions for the tool. Students have provided valuable feedback for the Software Collaboration tool evaluation. Some of the major suggestions are discussed here and a copy of all suggestions will be passed on to the Software Collaboration tool developers.

From the list of suggestions received for the survey question the first was to improve the tool interface, this was also observed in the tool "functionality survey question 4" with low rating. So, this would be of high importance for the developers to work on in the tool evolution stage. One of the participants have requested to be able to create an automatic link from the fault to the line in the SRS document. This functionality is available in Collaboration Software Inspection CSI tool. It is also evident from one user comment that he had fields covered up by entry form indicting of either an abnormality or a glitch. This wasn't observed by the other users, making us to reflect that it was one off occurrence and we could consider this an outlier. Suggestions to improve the drop-down menu for the fault type, navigation and to correct numbering system. Users have asked developers to automate date/time (instead of entering date/time manually). One user has stated an

ambiguous comment “Home page design looked a little funny if I recall correctly but it worked fine for what we needed it to do”. Participants felt difficulty to export consolidated fault list and work on it. Instead they requested to edit consolidated fault list on tool itself.

6.2. Improvements to Software Collaboration tool

These are the list of improvements/changes that should be include in the evolution plan of Software Collaboration tool:

- Firstly, software collaboration tool user interface should be enhanced by using better formatting, dropdown menu for the fault types and more importantly navigation (menu options).
- The number system is off in software collaboration tools annotation page (figure 4). This minor mistake should be fixed.
- Moderator cannot verify the time spent by each inspector in the tool. This functionality should be included in the tool to estimate state of inspector’s preparation and ensure sufficient time spent for the review.
- Inspectors are putting an extra effort to enter date/time manually in software collaboration tools annotation page (figure 4). This should be prevented by automating date/time entry.
- Software Collaboration tool cannot highlight specific features of the document to make it a stand out. This feature is important for inspector during their preparation period. Thus, faults should be linked to SRS document lines to track specific fault location.
- Author/moderator should be able to modify the consolidated faults list on the tool to remove duplicate and false-positive faults before generating master list.

6.3. Threats to Validate

Although this research has taken steps to minimize threats presented by survey study, the experiment has faced one unplanned threat that was outside researcher's control. The challenge was lack of response from intended participants. Out of 65 intended class group only half population has completed the survey even after encouraging the class to participate by explaining the need for this research. This can be negated by future researchers by making user take the survey in class. Taking the survey would take less than 10 mins and Qualtrics [33], the software used for the survey can be accessed on smart phone or all internet browsers. Research results securely indicate that the lack of participation threat has not affected the outcome of the research as the results were unbiased and almost all participants had same experience.

6.4. Future Work

Our survey research has shown that tools performance is well accepted by the participants with minor improvements as suggested in the previous subsection. The suggestions provided by the participants and the survey results will be shared with the tool developers to include them in the tool evolution plan. When the enhancements are included in the tool, this tool will be ready to be used in the software development industry for requirements inspection process.

7. CONCLUSION

In general, the survey participants are satisfied with the Software Collaboration tool and have agreed that the tool is very useful for software inspection process. Based on the survey results on hand, participants have expressed profound satisfaction for Software Collaboration Tool usability, appearance, performance, availability, reliability and portability. Also, the participants have thoroughly rated the logging in, finding documents, adding/deleting fault functionalities as easy to use but, were indistinct with ease to change password function. Like any new software user acceptance research, there were also a few users (less than 8%) who have expressed displeasure with the tool appearance and functionalities (listed in the data analysis and results). These suggestions were reviewed, documented and will be passed on to Software Collaboration tool developers to include them in the tool evolution plan.

REFERENCES

[1] Fagan, M.E. (1976). Design and code inspections to reduce errors in program development. IBM System Journal, Vol. 15, No.3, pp.182-211.

[2] Fagan, M.E. (1986). Advances in software inspections. Software Engineering IEEE Transactions on, 12(7), 744-751.

[3] Pressman, R. S. (1982). Software engineering: A practitioner's approach. New York: McGraw-Hill.

[4] Parnas, D., & Lawford, M. (2003). The role of inspection in software quality assurance. IEEE Transactions on Software Engineering IIEEE Trans. Software Eng., 29(8), 674- 676.

[5] Laitenberger, O. (2002). A Survey of software inspection technologies. Handbook of Software Engineering and Knowledge Engineering. Volume II: Emerging Technologies In 2 Volumes, 517-555. Retrieved April 11, 2016, from http://programmingresearch.com/content/misc/a-survey-of-sw-inspection-technologies-Laitenberger.pdf

[6] Subramanian, G. H., Jiang, J. J., & Klein, G. (2007). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. Journal of Systems and Software, 80(4), 616-627.

[7] Tian, J. (2005). Software quality engineering: Testing, quality assurance, and quantifiable improvement. Hoboken, NJ: Wiley.

[8] Freimut, B., Briand, L., & Vollei, F. (2005). Determining inspection cost-effectiveness by combining project data and expert opinion. IEEE Transactions on Software Engineering IIEEE Trans. Software Eng., 31(12), 1074-1092.

[9] Briand, L., Freimut, B., & Vollei, F. (n.d.). (2000). Assessing the cost-effectiveness of inspections by combining project data and expert opinion. Proceedings 11th International Symposium on Software Reliability Engineering. ISSRE.

[10] Perry, W. E. (2006). Effective methods for software testing: Includes complete guidelines, checklists, and templates. New York: Wiley.

[11] Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. The Journal of Systems and Software 80 (2007) 571–583.

[12] Bassil, S., & Keller, R.K. (2001). Software visualization tools: Survey and analysis. Proceedings 9th International Workshop on Program Comprehension. IWPC.

[13] Glasow, P.A. (25988). (2005, April). Fundamentals of survey research methodology. Mitre, Washington C3 Center McLean, Virginia.

[14] Macdonald, F., Miller, J., Brooks, A., Roper, M., & Wood, M. (1995, January). A Review of tool support for software inspection. Empirical Foundations of Computer Science (EFoCS) University of Strathclyde RR-95-181 [EFoCS-6-95].

[15] National Aeronautics. (1993, August). Software formal inspections guidebook. Space Administration Washington, DC 20546.

[16] Gupta, M. (2014, February). Investigating the use of model-based method for improving the quality of natural language requirements: a controlled empirical study. North Dakota State University Paper.

[17] Lu, Z. (2016, April). A software tool to facilitate automate creation of virtual inspection teams and inspection performance evaluation. North Dakota State University Paper.

[18] Kollanus, S., & Koskinen, J. (2009). Survey of software inspection research. The Open Software Engineering Journal, Volume 3, 15-34.

[19] Sembugamoorthy, V., & Brothers, L. (1990, October). ICICLE: Intelligent code inspection in a C language environment. The 14th Annual Computer Software and Applications Conference, pp.146-154.

[20] Brothers, L., Sembugamoorthy, V., & Muller, M. (1990, October). ICICLE: Groupware for code inspections. In Proceedings of the 1990 ACM Conference on Computer Supported Cooperative Work, pp.169-181.

[21] Gintell, J.W., Arnold, J., Houde, M., Kruszelnicki, J., McKenney, R., & Memmi, G. (1993, September). Scrutiny: A collaborative inspection and review system. In Proceedings of the Fourth European Software Engineering Conference, Garwisch-Partenkirchen, Germany.

[22] Mashayekhi, V., Drake, J.M., Tsai, W.T., & Reidl, J. (1993, September). Distributed, collaborative software inspection. IEEE Software, Vol. 10, No. 5, pp.66-75.

[23] Yourdon, E. (1989). Structured walkthroughs. Prentice Hall.

[24] Humphrey, W.S. (1989). Managing the software process. Addison-Wesley.

[25] Knight, J.C., & Meyers, E.A. (1991, July). Phased inspections and their implementation. Software Engineering Notes, Vol. 16, No. 3, pp.29-35.

[26] Knight, J.C., & Meyers, E.A. (1993, November). An improved inspection technique. Communications of the ACM, Vol. 11, No. 11, pp.51-61.

[27] Johnson, P.M., & Tjahjono, D. CSRS users guide. Technical Report ICS-TR-93-16, University of Hawaii.

[28] Johnson, P.M., & Tjahjono, D. (1993, September). Improving software quality through computer supported collaborative review. In Proceedings of the Third European Conference on Computer Supported Cooperative Work, Milan, Italy.

[29] Johnson, P.M., Tjahjono, D., Wan, D., & Brewer, R. (1993). Experience with CSRS: An instrumented software review environment. In Proceedings of the Pacific Northwest Software Quality Conference, Portland, OR.

[30] Johnson, P.M. (1994, May). An instrumented approach to improving software quality through formal technical review. In Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy.

[31] Bell, S. (1996). Learning with information systems: Learning cycles in information systems development. New York: Routledge.

[32] Pinsonneault, A., & Kraemer, K. L. (1993). Survey research methodology in management information systems: An assessment. Journal of Management Information Systems, 10, 75-105.

[33] The survey for this research was generated using Qualtrics Software. Copyright year 2019. Qualtrics and all other Qualtrics product or service names are registered trademarks of Qualtics, Provo, UT, USA. https://www.qualtrics.com