# User Experience Cycle 2

The UX 2 plan and results discussed here.

As significant improvisation over the previous user experience design cycle, codebase has been improved as with number of bug findings shown on user interface and also the number of tools shown as integrated is 10 while in previous cycle it was shown only 2. This helps to test whether the previous solution ideas still hold in terms of scalability. In addition, solutions ideas corresponding to code view are evaluated while in previous cycle only analysis view is investigated.

**Research Questions:**

This section shows the list of sub research questions considered for each main research question. In the case of the first primary research question, i.e., with displaying results for a single project from multiple static analysis tools, the following are the sub research questions.

1. From analysis view perspective, does a separate list or single list help the user to identify the common bug?
2. From analysis view perspective, will tags help in scalability of bug results in comparison to separate list or single list?
3. From code view perspective, will single icon suffice the showing of different tools icons?

Next, in the case of the second main research question, i.e., with feedback while bug fixing is on-going, the following are the sub research questions.

1. When submitting the bug for analysis, what feedback does user feel convenient among animation, progress bar or popup?
2. Does a single type of feedback suffice or requires combination?
3. From code view perspective, i.e., once user fixed a bug and submitted for analysis and then off the analysis results screen, then is popup notifications with analysis progress information better to busy status ( spinner )?

Finally, in the case of the third main research question, i.e., with carrying traceability of bug fixing, the following are the sub research questions.

1. In tracing, will the user need to know the changes made to fix a bug affecting the analysis of other tools?
2. Does adjective mapping ease the user to trace the changes made in code in terms of bugs existence?
3. From code view perspective, will the bug tool icons with before/after code help understand the user in easing to fix it?

**User Study Process and Results:**

This section explains the user scenario and questionnaire used during the user study.

**Results**

**Metrics Analyzed:**

**Quantitative measurements:**

* Task Success:

Whether the user able to complete the respective task or not?

**Qualitative measurement:**

1. Usability: Ask the user to provide feedback through cognitive walkthrough process about problems faced while using the prototype and get insights about the solution idea.

2. For each research question, ask users to evaluate the solution idea in terms of perceived usability on a Uni-polar Likert scale of 0-10. Scale: 0 – worst, 10 – best, in comparison to alternative solution idea provided.

**Pre-test:**

By first, the user background is verified with the pre-test questionnaire whether he/she is the right candidate to consider for user study. The ideal choice is the user who has Computer Science studies background and programs software projects. Also, it is found whether the user has used any static analysis tools and if so, the relationship between their favourite tool and usability is found out.

**Questionnaire:**

Q. How often do you do software development (i.e., coding)?

Q. Have you used static analysis tools?

Q. What tools have you used?

Q. Is it IDE integrated tool or any other dedicated tools like FindBugs, PMD etc.

Q. Which is your favourite one?

Q. Why is your favourite? Any correlation to its better user interface feature?

**Results:**

There are 7 users participated in this user study phase. Everyone has Computer Science background with at least a bachelor’s degree, and in addition, 3 users have sound professional experience. Among them, 6 users are pursuing master’s degree in computer science at the time of this user study and 1 user is a research assistant in the research group Software Engineering.

Out of 7 users, 3 users almost do coding on every day, and remaining users on an average do coding three days in a week. Only one user used static analysis tools and in fact a core developer of one of his own static analysis tools and whereas remaining users usually do manual testing or use debugger in order to find any bugs in their code. Nevertheless, all users are familiar or at least heard of static analysis in general. One user used tools namely SpotBugs, FindSectBugs and his own tool titled CogniCrypt which all are integrated into IDE called Eclipse. Among them apart from his own tool. SpotBugs was his favourite one. The reason behind it is that he likes both its functionality with how fast it was and the presentation of results is ok with displaying errors. With regards to the importance of user interface in addition to functionality, users felt the importance should be given for both as they both matter and interface would be beautiful to have for better visualization of existence of bugs. One user said that he uses Visual Studio IDE, which is intuitive and easy to use, thereby expects to be able to have such interface familiarity with static analysis tools. One user expressed that this research work is stepping in the right direction. It is encouraging to hear from a user who is having a lot of programming experience about 10 years. One user mentions that more tools it reports a particular bug then it ensures the finding is actually a bug. This is again interesting to hear as it inlines the purpose of this thesis.

Once the pretest questionnaire is done, the user is asked to assume that they are working on a software project and want to find bugs in their codebase. There are ten tools linked to their codebase for better coverage of vulnerabilities. Next, walk through three main research questions with respect to its user interface one by one with their sub research questions and evaluate their own solution ideas for both analysis view and code view.

The solution ideas are made into prototypes, i.e., wireframes using Balsamiq software tool. They are presented to user one after other in random order. Cognitive walkthrough is carried out with Think Aloud process during user study.

Now let us walk through each research question.

**Research Question 1:** How to display the results of the same codebase from different analysis tools?

As part of this research question, there are three sub research questions considered. The solution ideas for analysis view such as single list, separate list and tags are made into respective prototypes, and for code view, the ideas are single icon and separate icons. The solution ideas are evaluated with the following pattern with User Scenario. Task and Follow up questionnaire. Whereas success criteria to the task in order to determine whether the user able to perform the task or not on the presented prototype is mentioned at respective solution idea. The follow-up questions are asked once the first 3 solution ideas are presented for analysis view.

**User Scenario:**

The following user scenario is presented to the user, “Assume you as a Software Developer working on a project and about to see the analysis results in a given user interface.”

**Task**: Understand the results and Identify the common bug reported by tools.

**Follow Up:**

1. Do you desire to have ‘union’ or ‘intersection’ results when selecting multiple tools through filter?

2. Among the solution ideas presented, which one do gentle feel convenient (easy to use) with for the given task?

3. Why is it the mentioned solution idea is convenient?

4. How do you rate in terms of perceived usability ranging from O be lower to 10 be high for provided solution designs in comparison?

5. Which UI do you think scales better with increase in bugs?

Now let us begin with first solution idea to analysis view, i.e., Single list.

**Solution Idea: ( Single List )**

As we know, each static analysis tool has a respective name and an icon representing it. So, when we integrate multiple tools, we either must use tool names or icons to represent the respective analysis tool while displaying the bug-finding results that indicates which bug was identified by what tool. In the current scenario, an idea of having a table view with a tool column that shows tool icons for a respective bug finding. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

**Task – Success Criteria:**

User finds the “**Bug name – XSS\_CONFIG”** which is a common bug by **clicking** on **‘Select all’** tools filter.

In addition, user interface is demonstrated by showing multiple screens for each tool and significance of icon. This is to ensure the user gets the solution idea clearly.

**Evaluation:**

**Quantitative Results:**

There are seven user study participants. Among them, 5 users manages to perform the task and remaining 2 users could not do so. This shows the task success rate is 71.43 per cent.

Among the 7 users, 5 users felt this solution idea is convenient in comparison to other solution ideas to be considered as final solution idea in showing the analysis results. In terms of usability on scale of 0 to 10, where 0 being the worst and 10 being the best, 7 users rated as 7,6,7,10,9,9 and 9 which averages to 8.14.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. The reasoning behind the low task success rate is because one user expected to see intersection of results with tool selection and got lost in that thinking as in present prototype it shows union of bug findings by the selected tools in filter and other user understood the task in different way, i.e., finding a same kind of bug identified by the tools. Users felt that selecting of tools shows common bugs, just used to represent data in tables and it is effortless to perceive, more user-friendly, icons are always gentle relief to eye, it scales better, much better to understand and also much convenient. Two users have mentioned about their expectation that selections of tools results in only common bug-findings as results. In addition, three users mentioned the presentation of tool icons and their names. One user suggested to use tool names instead of icons in case where they are not so distinctive. Similarly, one user stated that it could confuse to relate which icon represents what tool. As an improvement, there is a suggestion from different user to have tool names that appear on scroll over tool icon.

**Solution Idea: ( Separate List )**

Similar to the previous one, now the idea is showing separate table view for each tool. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

**Task – Success Criteria:**

User scrolls to see the results **until tool 6** and find the **Bug name – XSS\_CONFIG,** which is a common bug.

The user is able to scroll until tool 6 only as it signifies the real-time scenario where sometimes tool 7, 8. 9 and 10 does not report any bugs. In addition, the user interface is demonstrated by showing the bugs related to different tools separately and the bugs related to tool 1 and tool 3.

**Evaluation:**

**Quantitative Results:**

There are seven user study participants. Among them, only 3 users manage to perform the task, and the remaining users could not do so. This shows the task success rate is 42.85 per cent.

Among the 7 users, no user felt the solution idea is convenient in comparison to other solution ideas to be considered as final solution idea in showing the analysis results. In terms of usability on scale of 0 to 10, where 0 being the worst and 10 being the best, 7 users rated as 3,6,5,7,6,7 and 4 which averages to 5.43.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. The reasoning behind the low task success rate is because one user expected to see intersection of results with multiple tools selection that made him lost in thinking, one user felt it is not the best UI to do the task so dropped off and two users felt it is time-consuming. In addition, some users felt there is no enough space to properly analyse and especially challenging to identify the common bug. In contrast, there is one user who felt that this solution idea makes more easy to perceive.

**Solution Idea: ( Tags )**

As we have seen so far using icons for representing an analysis tool. Now, having a tag name for each tool and that could be used for bug finding results displayed in a complete list view. The present solution idea is inspired from StackOverflow user interface. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

**Task – Success Criteria:**

User scrolls down **3 screens** to see the results and find the **Bug name – XSS\_CONFIG,** which is a common bug.

In addition, the user interface is demonstrated by showing alternative approaches to see bugs reported by tool1.

**Evaluation:**

**Quantitative Results:**

There are seven user study participants. Among them, 6 users managed to perform the task. This shows the task success rate is 85.71 per cent.

Among the 7 users, 2 users felt the solution idea is convenient in comparison to other solution ideas to be considered as final solution idea in showing the analysis results. In terms of usability on scale of 0 to 10, where 0 being the worst and 10 being the best, 7 users rated as 8,7,6,8,7,9, and 6 which averages to 7.28.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. The only user who did not able to do the task as the user interface is confusing. Some users said that UI is pretty cool and pleasant interface which gives clear idea on how common bug is identified and also the UI is a bit more open. There are a couple of users who did not like the UI as a developer and felt it is not much clear. One user suggested to replace tool name with its respective icon.

Now, the solution ideas in the context of code view are presented. Similar process to solution ideas of analysis view, a user scenario is given to user followed by a task to be performed and finally after presented both the solution ideas prototypes, follow up questionnaire is asked. Whereas success criteria to the task in order to determine whether the user able to perform the task or not on the presented prototype is mentioned at the respective solution idea.

User Scenario:

Assume you are a developer working on project Alpha precisely on software package called ‘scripts\_module’. On your next working day, morning opened your code editor to start your work.

Task:

What is the bug being reported by your analysis tools? And how many tools reported it?

Follow Up:

1. Among the solution ideas presented, which one do you feel convenient (easy to use) with for the given task?

2. Why is it the mentioned solution idea is convenient?

3. How do you rate in terms of perceived usability ranging from O, be lower to 10, be high for provided solution designs in comparison?

Solution Idea – Multiple Icons

Once we have analysis results from the integrated tools, the respective bug findings in code editor are shown. In the present scenario, a tool icon is shown in the gutter at the respective line of code where the bug is identified by that tool. So, every tool shows its respective tool icon. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

**Task – Success Criteria:**

On two clicks with different tool icons know that the bug is “Anti cross-site scripting filter”. The answer for number of tools is 2.

**Evaluation:**

**Quantitative Results:**

There are seven user study participants. All users had managed to perform the given task. This shows the task success rate is 100 per cent.

Among the 7 users, 2 users felt the solution idea is convenient in comparison to other solution ideas to be considered as final solution idea in showing the analysis results. In terms of usability on scale of 0 to 10, where 0 being the worst and 10 being the best, 7 users rated as 5,8,5,8,8,10, and 9 which averages to 7.57.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. User felt that separate icons is better to have in single glance but if more tools, then single standard icon is best in terms of scalability. Others mentioned that it overcrowds quickly if more than 2 gutter icons per row, it does not look pretty good having all icons in one place. One user suggested to have tool icons at project structure besides filename. One exciting aspect is that one user at first did not understand the icons, and later he clicked on icons and then understood the concept without the designer interference.

Solution Idea – Single Icon

Once we have analysis results from the integrated tools, the respective bug findings in code editor are shown. In the present scenario, a single icon is shown in the gutter at the respective line of code where the bug is identified no matter which tool has identified it. For more details about which tool has found it as bug could be given in a text box which is a popup on click of the standard icon. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

**Task – Success Criteria:**

On single click on bug icon know that the bug is ‘Anti cross-site scripting filter’. The answer for number of tools is 2.

**Evaluation:**

**Quantitative Results:**

There are seven user study participants. All users had managed to perform the given task. This shows the task success rate is 100 per cent.

Among the 7 users, 5 users felt the solution idea is convenient in comparison to other solution ideas to be considered as final solution idea in showing the analysis results. In terms of usability on scale of 0 to 10, where 0 being the worst and 10 being the best, 7 users rated as 7,7,6,10,10,9, and 10 which averages to 8.43.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. Users said as a developer the vital aspect is to find bug but not maybe which tool reported it, having single icon is better in terms of readability, more straightforward, more convenient as it doesn’t overcrowd the left-side gutter and looks much better. A couple of suggestions from users are to also add tool name instead of tool icon for general user to understand quickly and other user says in similar context that to have a popup for icons to know the respective tool names. One additional improvisation request is to have some bug severity representation at gutter section.

**[ Research Question 2]**

Q. What feedback works to know that the bug fixing is on-going?

As part of this primary research question, we are going to first evaluate the three feedback mechanisms proposed in context of analysis view, i.e., animated icons, progress bar and pending status popup in our designed MSAT-UI ( Multiple Static Analysis Tools – User Interface ) and later in context of code view with other solution ideas.

The ‘animated icons’ solution idea demonstrates that when a bug finding has been selected, and user attempted to fix it by submitting for re-analysis, a feedback is shown with respective tool icons being animated for that bug which is under analysis. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

Next, ‘progress bar’ solution idea demonstrates that when a bug-finding is been worked on and submitted for analysis, a feedback is given to user about how far the respective analysis tools has finished analyzing by showing progress bars for one or many tools that found the bug. This is displayed when clicked on the bug which is under analysis. In addition, the respective icons are shown as blurred, indicating they are under analysis in contrast to previous solution idea. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

And finally, the third solution idea, i.e., ‘ pending status popup’ demonstrates the textual feedback about what the analysis tools are scanning as an example, tool name, timestamp, respective filename with code under analysis. This is displayed on click on text link named pending in status column in the given table view. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

The User Scenario, Tasks, and Follow up questionnaire is similar to previous research question. However, the tasks are demonstrated by the designer in order to make the feedback mechanism ideas are understood by the user. The reason for not doing so is the limitation with prototype builder, i.e., Balsamiq not having dynamic nature in showing animation effects and thereby designer tries to mock the animation effect with multiple clicks which could be tedious if asked the user to do.

**User Scenario:**

Assume that you worked on a bug and changed some code related to it. Next, submitted bug fix code for analysis.

**Task:**

- Observe what happens after clicking on ‘Fixed’ button.

**Follow Up:**

1. Among the solution ideas presented, which one do you feel convenient (easy to use) with for the given task?

2. Why is it the mentioned solution idea is convenient?

3. How do you rate in terms of perceived usability ranging from O, be lower to 10, be high for provided solution designs in comparison?

**Evaluation:**

**Quantitative Results:**

Since the designer has performed the tasks, there is no such as task success as seen in previous research question. Among the 7 users, no user felt any solution idea is not convenient in comparison to other solution ideas to be considered as final solution idea in showing as a feedback during the bug fixing process. Most users preferred them to have in combinations and especially three users explicitly felt status pending with pop up would be more useful. Nevertheless, each feedback is appreciated in the context of its respective usage scenario. In terms of usability on scale of 0 to 10, where 0 being worst and 10 being the best, 7 users rated ‘animated icons’ solution idea as 7,7,6,8,8,7 and 8 which averages to 7.28, ‘progress bar’ solution idea as 7,8,8,10,9,8 and 6 which averages to 8 and finally ‘pending status pop up’ solution idea as 4,8,8,7,8,9 and 10 which averages to 7.7.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. For animated icons, one user explicitly stated that it is useful to know which bugs are being analysed. For progress bar, one user says as tools are optimized in general and so takes less time and thereby it has no priority as it also kind of fancy in his perspective. For status pending pop up, the user expressed his interest in knowing what is happening, and this feedback helps so. Others also mentioned that it is much convenient, this feedback is solely enough, it is more detailed, having timestamp when it is processed it is good. Overall, the users requested to merge the feedbacks in a sense to have them all in a proposed prototype as each helps in its own way of knowing what bugs are analysed, how far a bug is analysed and more important details of the analysis in third feedback. In addition, one user suggested to merge both code editor view and analysis viewer in the same window.

Let’s now look into the solution ideas for Code view perspective. The proposed solution ideas are, namely ‘status spinner’ and ‘alerts.’

The ‘status spinner’ solution idea demonstrates that when a user has submitted a bug for analysis, and later he if off the analysis screen and probably working on a different task. In this scenario, a feedback at the corner of the screen is shown as with busy indicator like a spinner. This could be well understood, looking at following figure or a complete prototype images added in Appendix. This indicates whether an analysis is running in the background or not.

The ‘alerts’ solution idea demonstrates that when a user has submitted a bug for analysis and later he if off the analysis screen and probably working on a different task. In this scenario, a feedback is shown as a popup whether the bug fix got succeed or not and also two call to action buttons such as ‘try again’ or ‘ok’. The ‘try again’ action button help to work on the same bug again and so this button could link to the program file where previous changes are made with respect to the bug finding. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

The following User Scenario, Task and Follow up questionnaire protocol is examined.

User Scenario:

In addition to the previous scenario, after submitting code for analysis, instead of being on the bug results window, you moved on to the next task in code editor.

Task:

Observe the visuals with alert box and status bar with spinner as demo.

Follow up:

1. Among the solution ideas presented, which one do you feel convenient (easy to use) with for the given task?

2. Why is it the mentioned solution idea is convenient?

3. How do you rate in terms of perceived usability ranging from O, be lower to 10, be high for provided solution designs in comparison?

4. Would you prefer having multiple feedbacks?

**Evaluation:**

**Quantitative Results:**

Since the designer has performed the tasks, there is no such as task success as seen in previous research question. Among the 7 users, 4 users felt it ‘alerts’ is convenient in comparison to other solution ideas to , as final solution idea in showing as a feedback during the bug fixing process. In terms of usability on scale of 0 to 10, where 0 being worst and 10 being the best, 7 users rated ‘status spinner’ solution idea as 6,8,6,10,6,9 and 8 which averages to 7.57 and for ‘alert’ solution idea as 2,8,8,8,9,9 and 9 which averages to 7.57.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. In context of status spinner, users felt that it is simple, if you want to know status then click on analysis, In context of alerts, couple of users felt it is annoying, not useful while working on different priority work and also takes them out of context their mind was in. Nevertheless, there are positive opinions from other users saying that it draws attention which is positive, it shows details, and it does not require much transitions to know the status. There is one suggestion from his expertise background in coding says that if alerts needs to be implemented, it is better to have in context of bug fix failure but not in success as a reason the tool does not need to say it fixed as developer know it is fixed before submitting to analysis. This shows that this user is usually confident about his bug fixes. Some users preferred to have both as they serve different purposes and indeed, they are both helpful feedbacks.

**[ Research Question 3]**

Let’s switch to the final main question;  
  
Q. How to carry traceability of bug fixing?

The following sub research questions are evaluated in this context.

1. In tracing, will the user need to know the changes made to fix a bug affecting the analysis of other tools?
2. Does adjective mapping ease the user to trace the changes made in code in terms of bugs existence?
3. From code view perspective, will the bug tool icons with before/after code help understand the user in easing to fix it?

The first two questions correspond to analysis view, and the third corresponds to code view. The solution ideas tested are namely ‘numbers’ and ‘adjectives’ for analysis view and ‘before/after’ for code view. Firstly, let us look at the analysis view solution ideas and see how users evaluated them.

Now the user is provided with the following User Scenario, Task and Follow up questionnaire once both prototypes are presented. Task Success Criteria is mentioned along with respective solution idea below.

User Scenario:

You have been working on some part of code to fix a bug in last few working days. Now you would like to see how the changes ( commits ) you made are affecting the analysis results of other tools.

Task:

Watch the results of all tools and decide on the best commit among last 3 to revert as to start from that point.

Follow up:

1. Among the two solution ideas presented, which one do you feel convenient (easy to use) with for the given task?

2. Why is it the mentioned solution idea is convenient?

3. How do you rate in terms of perceived usability ranging from O, be lower to 10, be high for provided solution designs in comparison?

Solution Idea – Numbers:

This idea demonstrates as when respective commits are made to fix a particular bug. The commit ids are shown along with some metrics, i.e., numbers of new bugs introduced or got fixed by its respective change as determined by a tool. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

User Success Criteria:

Selecting the commit id – fse254

**Evaluation:**

**Quantitative Results:**

There are 7 users participated in this user study. Out of which, 5 users are able to perform the task. This shows the task success rate is 71.42 per cent.

Among the 7 users, 4 users felt it is convenient in comparison to other solution ideas to be considered as final solution idea. In terms of usability on scale of 0 to 10, where 0 being worst and 10 being the best, 7 users rated it as 7,6,7,10,8,8 and 5, which averages to 7.28.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. Users felt it is incredibly complicated when he is unsure of what a number is, and he tends to click on it as if it is a link, challenging to get to conclusion for the action needs to done in the context of the given task, it depends on the difficulty of the bugs represented in the numbers. In addition, a couple of users suggested to have latest commits on the top as in terms of user consistency model, which could be considered as improvisation for prototype design in general.

Solution Idea – Adjectives:

This idea demonstrates as when respective commits are made to fix a particular bug. The commit ids are shown along with some adjective words, i.e., worst, bad, good, better and best. Each word demonstrates how easy or hard to get the bugs introduced are fixed. As in previous solution idea, there could be a scenario where a change in specific commit could introduce new bugs or fix some bugs as reported by some tool. Now, in order to make it easy to evaluate or understand the numbers, adjectives could be used as a replacement. These adjectives are determined as rough time estimate difference to fix new bugs, and bugs got fixed. In practice, a rough time estimate is shown in tools like Codacy. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

User Success Criteria:

Selecting the commit id – fgd547

**Evaluation:**

**Quantitative Results:**

There are 7 users participated in this user study. Out of which, 6 users are able to perform the task. This shows the task success rate is 85.71 per cent.

Among the 7 users, 3 users felt it is convenient in comparison to other solution ideas to be considered as final solution idea. In terms of usability on scale of 0 to 10, where 0 being worst and 10 being the best, 7 users rated it as 4,6,5,8,6,10 and 7, which averages to 6.57.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. Users felt that it depends on the expertise of the user, rough estimation of time is helpful and agreed to the scenario of having time estimate for this open research question. One user stated it is confusing, but however, all others accepted the idea. There are couple of suggestions such as combining this idea with previous one like numbers are shown at first and on click, shows what are those bugs whether good or bad in terms of fixing it in general and for beginners this idea is helpful, but numbers are helpful for experts as time estimate could not be same among all colleagues in a software development team. One additional suggestion is to show what bugs are reported and where.

Now let us look into the solution idea in the context of code view, which is named as ‘Before/After’. It shows two columns namely Before and After where Before column has the code status before the commit happened while the bug is identified by a particular tool, wherein After column, it shows the changes made to code and if the tool states the bug is fixed then it shows positive code change and if the tool says it needs fix then it determines the code where the bug is identified and at this point, Before column shows the code before the change is made that lead to presence of bug. This could be well understood, looking at following figure or a complete prototype images added in Appendix.

The following User Scenario, Task and its Success Criteria and finally, a Follow-Up questionnaire is used to evaluate the design idea.

**Scenario:**

As usual, you are working on a code editor and notice a bug reported. You remember you have worked on this part of code before to fix some other bug.

**Task:**

Find out why you changed the code before and plan to come up with better solutions that satisfies the tools reporting same part of code. In this case, what was the code line you changed earlier to fix the bug reported earlier?

**Success Criteria:**

It is found out that code line “ **value = value.replaceAll(“”,””);** “ is uncommented.

**Follow up:**

Q. Do you feel convenient by this UI with your workflow?

Q. Why do you think is the shown design is convenient or not so?

**Evaluation:**

**Quantitative Results:**

There are 7 users participated in this user study. Out of which, 6 users are able to perform the task. This shows the task success rate is 85.71 per cent. Accordingly, users accepted the convenience with their usual workflow.

**Qualitative Results:**

Let us now investigate the reasoning behind the user’s choice of solution idea and respective ratings. One user who was not able to do the task was he could not understand the user interface in first glance as it is novel. This could be considered as negligible and thereby proves the potential of the idea once design is clearly understood.

Users felt that this interface is definitely more helpful than other views shown in analysis view solution ideas, helpful, novel, outstanding that user able to understand what changes are made, which all are positive. There are however few remarks and suggestions to make it more useful from user perspective such as having some background colour for changed line, colours for change in code, a single screen then no need of trace button but however, the same user agreed with proposed design in context of multiple tools .Other design idea for this scenario shared by the participant is Git like merge conflict design concept. Overall, this design idea is welcomed by the users. One general suggestion is to have the flexibility of going to a code file on selection of a bug finding.

**Limitations:**

There are mainly a couple of limitations noticed during this user study process. They are:

1. Mock screens jump to next screen without of scope click.
2. Animation effects are not possible with available prototype builder, i.e., Balsamiq.