

Integration of Multiple Static Analysis Tools in a Single Interface

- G. S. Varma
- Supervisors:
 - Prof. Dr. Eric Bodden
 - Dr.-Ing. Ben Hermann

Problem Statement

- How to integrate the results of multiple static analysis tools

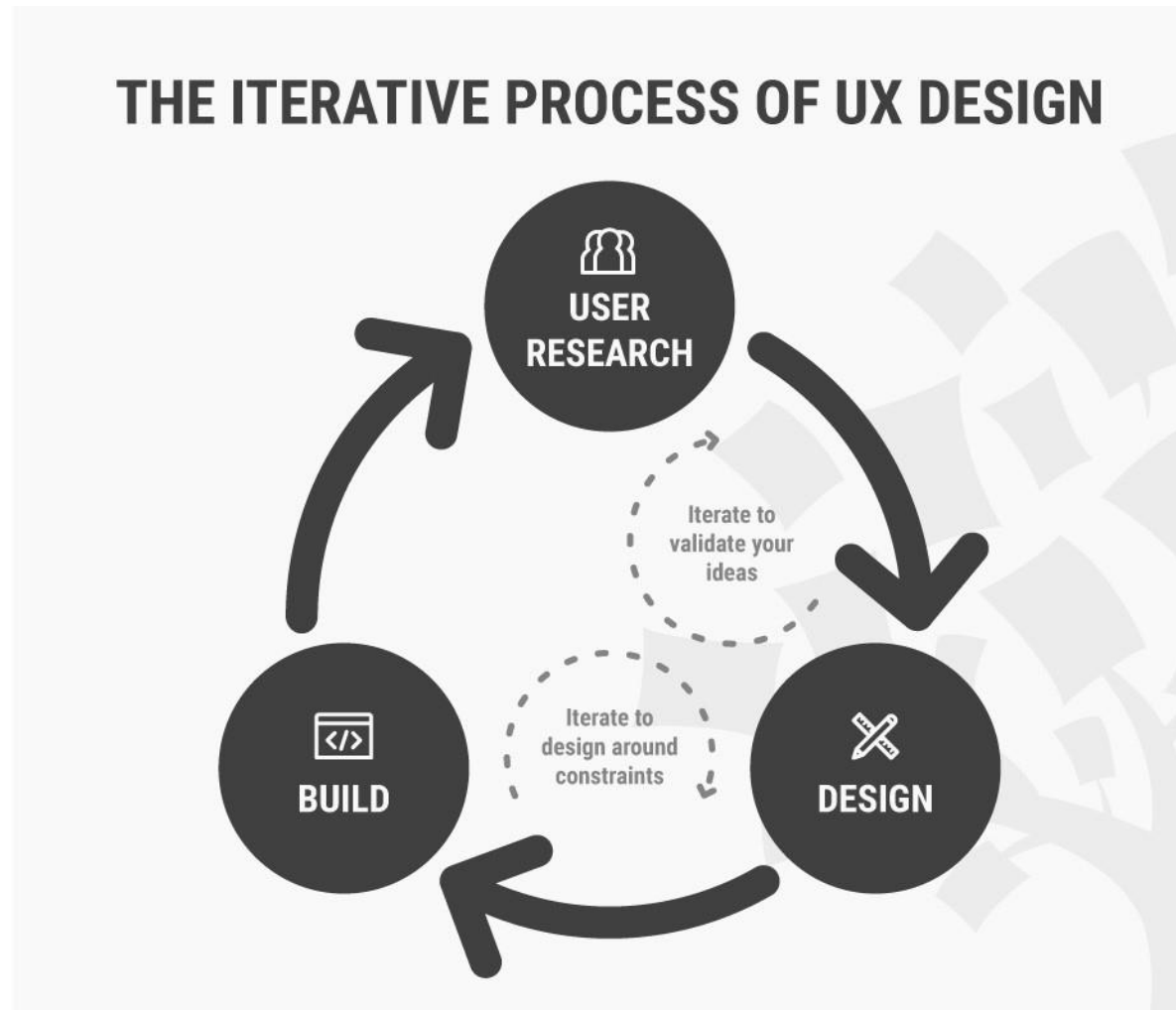
in a unified user interface?

❖ 3 Research Questions

Research Questions

- RQ 1: How to display results of the same codebase from different analysis tools?
- RQ 2: What feedback works to know that bug fixing is on-going?
- RQ 3: How to carry traceability of bug fixing?

Research Methodology



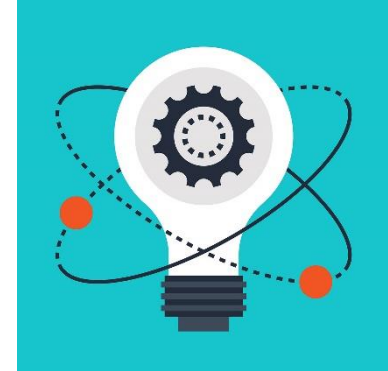
- ❖ How to Change Your Career from Graphic Design to UX Design. url: <https://www.interaction-design.org/literature/article/how-to-change-your-career-from-graphic-design-to-ux-design>.

Our Approaches

- Software Engineering disciplines:
 - Complex datasets
 - Compiler reporting
 - Continuous integration
 - Refactoring tools
 - Issue tracker
 - Stack Overflow
 - Gamification
 - Usability Engineering

Evaluation

- Experimental Design
 - Recruit Test Users
 - Order of evaluation altered
 - Perform Tasks (Metric 1 – Task Success)
 - Likert Scale (Metric 2 – Perceived Usability)
 - Voting (Metric 3 – Preferred Design)
 - Usability inspection methods: Cognitive Walkthrough



❖ Rensis Likert. "A technique for the measurement of attitudes." In: Archives of psychology (1932).

Analysis View

MSAT Interface

Project: Alpha

S. No.	Name (Bug title)	Tool	Type	Fix Location	Assignee
1	XSS_CONFIG		XSS	12.4 XSSFILTER.java	Varma
2	EQ_CHECK		EQ	6.3 LoopHelper.java	Max
3	CO_SELF		CO	11.2 StringComparer.java	Un-assigned
4	XSS_REQUEST		XSS	5.4 HttpSender.java	John
5	DMI_EMPTY		DM	3.3 DatabaseHelper.java	Elina
6	BC_EQUALS		BC	2.4HttpReceiver.java	Tom
7	BIT_CHECK		BIT	3.3 NetworkConnect.java	John
8	CN_CLONE		CN	6.7 CloneMessage.java	Max
9	DE_EXCEPTION		DE	2.2 StringPlacer.java	Elina
10	DMI_RANDOM		DMI	3.7 DatabaseConnect.java	Elina
11	EQ_EQUALS		EQ	1.3 StringCheck.java	John
12	IJU_TEST		IJU	9.3 DatabaseTest.java	John
13	IL_LOOP		IL	7.2 FormValidate.java	Tom
14	CI_FINAL		CI	1.6 MessageSender.java	Max
15	SQL_CONSTANT		SQL	3.5 DatabaseInsert.java	Elina

Bug Description

XSS: Anti cross-site scripting filter (XSS_CONFIG)

Wrap the HTTP request object in a specialized HttpServletRequestWrapper that will perform filtering.

Fix Now

Know More

Filters

☒ Select All ☐ Deselect All

☒ tool1

☒ tool2

☒ tool3

☒ tool4

☒ tool5

☒ tool6

☒ tool7

☒ tool8

☒ tool9

☒ tool10

Bugs

☐ My Bugs

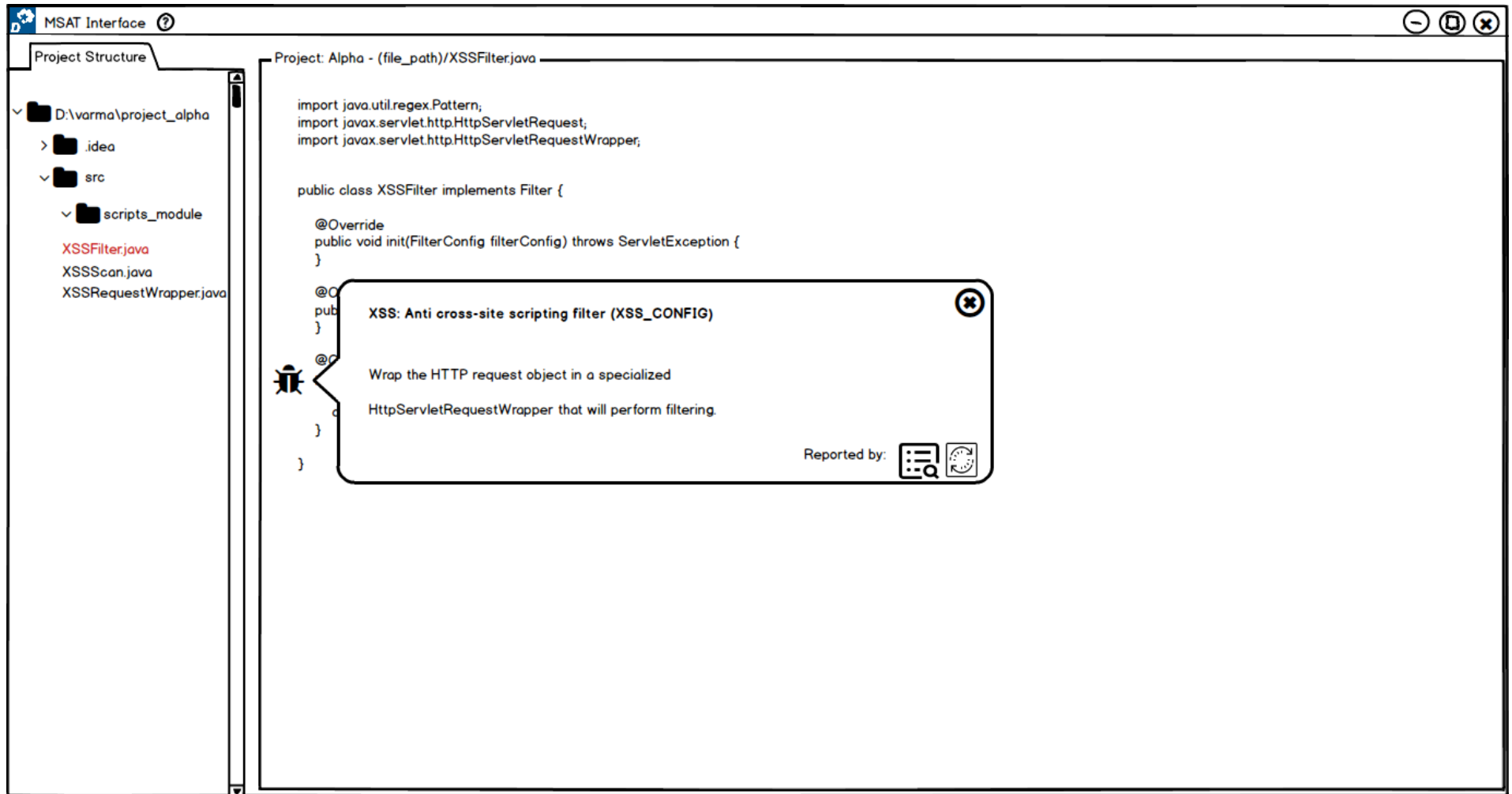
☒ All Bugs

Vulnerability Type

☐ SQL Injection

☐ XSS

Code View



UX Design Cycle 1

UX 1

- Users: 5
- Sub research questions: 9
- Each session: ~ 90 minutes





Added db2 database mapping after reading forum post

Jul 17 2014 10:53

by [Daniel Lewis](#) in revision [91687a1146419dd23ceaed299185512696643dc1](#) (git)

Files: 11 changed

Findings: 0 4 12 1



Add getDelegationState() in DelegateTask.

Jul 17 2014 10:30

by [Anya Hill](#) in revision [812b1e277d844fa48307bcd7c692a6f395c85fbb](#) (git)

Files: 14 changed

Findings: 0 3 12 5



TASK_TIMEOUT

Jul 17 2014 08:46

by [Jacob Nelson](#) in revision [997da57af6f2c08d504473d3e9837788b7592dcb](#) (git)

Files: 14 changed

Findings: 0 5 12 3

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

The screenshot shows the MSAT Interface with a project named 'Alpha'. It displays a table of bugs and a detailed view of the 'FI_EMPTY' bug.

Name (Bug title)	Tool	Type	Fix Location	Assignee
FI_EMPTY		FI	12.4 EditList.java	Varma
EQ_CHECK		EQ	6.3 LoopHelper.java	Max
CO_SELF		CO	11.2 StringComparer.java	Un-assigned
XSS_REQUEST		XSS	5.4 HttpSender.java	John
DMI_EMPTY		DM	3.3 DatabaseHelper.java	Elina

Bug Description

FI: Empty finalizer should be deleted (FI_EMPTY)

Empty **finalize()** methods are useless, so they should be deleted.

[Fix Now](#) [Know More](#)

Filters

Tool

- ☒ toolLong
- ☒ toolShort

Bugs

- ☐ My Bugs
- ☒ All Bugs

Vulnerability Type

- ☐ SQL Injection
- ☐ XSS

■ Single List

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

The screenshot shows the MSAT Interface with a window titled "MSAT Interface". It displays two separate lists of bugs for different tools. The top list is for "toolShort" and the bottom list is for "toolLong". Both lists have columns for Name (Bug title), Type, Fix Location, and Assignee. The "toolShort" list shows bugs FI_EMPTY, CO_SELF, and DMI_EMPTY. The "toolLong" list shows bugs FI_EMPTY, EQ_CHECK, and XSS_REQUEST. The "toolLong" list also has a "click here to select" link next to the FI_EMPTY bug. A search bar with "Un-assigned" is visible in the "toolLong" list. On the right side, there is a "Filters" panel with checkboxes for "Tool" (toolLong, toolShort), "Bugs" (My Bugs, All Bugs), and "Vulnerability Type" (SQL Injection, XSS).

Name (Bug title)	Type	Fix Location	Assignee
FI_EMPTY	FI	12.4 EditList.java click here to select	Varma
CO_SELF	CO	11.2 StringComparer.java	Un-assigned
DMI_EMPTY	DM	3.3 DatabaseHelper.java	Elina

Name (Bug title)	Type	Fix Location	Assignee
FI_EMPTY	FI	12.4 EditList.java click here to select	Varma
EQ_CHECK	EQ	6.3 LoopHelper.java	Max
XSS_REQUEST	XSS	5.4 HttpSender.java	John

■ Separate List

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

■ Results:

	Single List	Separate List
Task Success	100 %	100 %
Usability	8	7.6
Votes	2	3

- Separate List – “more effective when using more tools”

MSAT Interface ?

Project: Alpha - Selected Bug Description

FI: Empty finalizer should be deleted (FI_EMPTY)

Empty **finalize()** methods are useless, so they should be deleted.

A computer uses several types of hardware resources, and those are physically limited by nature. Therefore, we have to use those resources wisely. In Java, unlike in older programming languages like C++, memory is automatically managed by the garbage collector; however, there are other resources, usually "IO resources", that we need to manage by ourselves.

Filters

Tool

☒ toolLong
 ☒ toolShort

Bugs

☐ My Bugs
 ☒ All Bugs

Vulnerability Type

☐ SQL Injection
 ☐ XSS

S. No.	Commit ID	Time Stamp	toolLong	toolShort	Revert
1	fcd121	17-07-2019 09:56	0 1	0 2	
2	efd008	17-07-2019 14:09	0 1	1 1	
3	vcs113	19-07-2019 10:23	2 0	2 1	

Back

UX 1 – Lessons

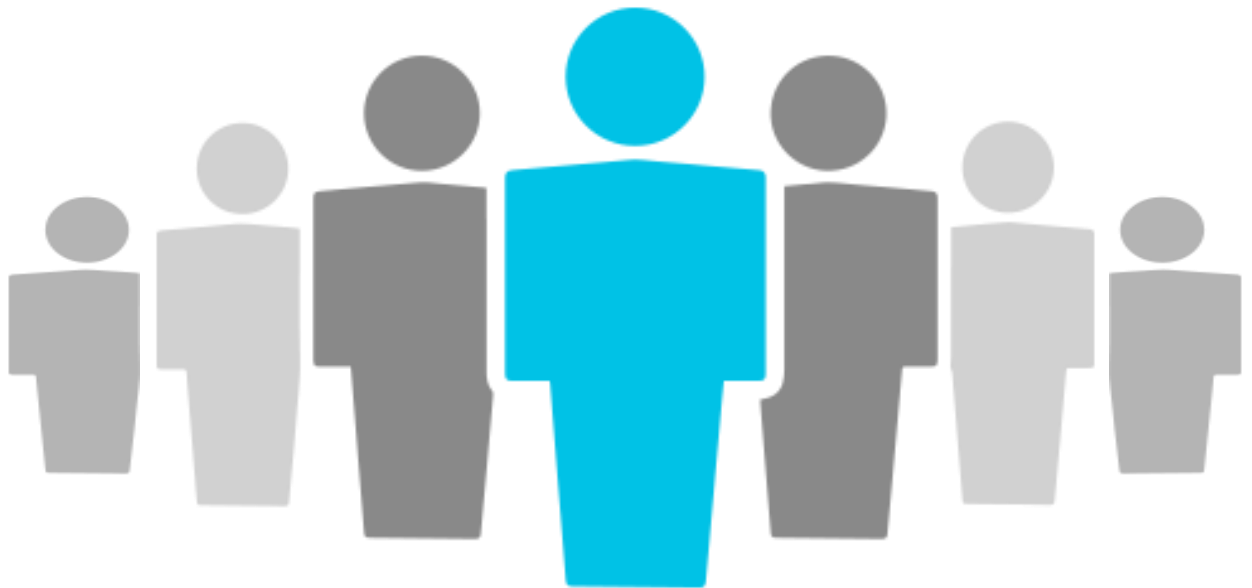
- Analysis View
- Improvisations for next UX cycle:
 - Increase code base
 - Volume of bugs (+ scroll)
 - Integrate more tools
 - Code view perspective
 - + new sub RQ's



UX Design Cycle 2

UX 2

- Users: 7
- Sub research questions: 9
- Each session: ~ 105 minutes



Issues - userstudyJS - us2019

https://sonarcloud.io/project/issues?id=userstudy2019_userstudyJS&resolved=false

sonarcloud

Explore ? Search for projects and files... Log in

us2019 / userstudyJS master

August 4, 2019, 6:19 PM Version not provided

Overview Issues Measures Code Activity

Filters

Type

- Bug 300
- Vulnerability 0
- Code Smell 1.1k
- Security Hotspot 0

Severity

- Blocker 0
- Critical 5
- Major 900
- Minor 450
- Info 0

Resolution

Status

Security Category

index.js

2 duplicated blocks of code must be removed. See Rule

Code Smell Major Open Not assigned 30min effort

3 months ago

pitfall

Add a new line at the end of this file. See Rule

Code Smell Minor Open Not assigned 1min effort

3 months ago

convention

Remove this useless assignment to local variable "c". See Rule

Code Smell Major Open Not assigned 15min effort

3 months ago L11

cert, cwe, unused

Remove the declaration of the unused 'c' variable. See Rule

Code Smell Minor Open Not assigned 5min effort

3 months ago L11

unused

Remove this useless assignment to local variable "d". See Rule

Code Smell Major Open Not assigned 15min effort

3 months ago L12

cert, cwe, unused

30min effort

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

The screenshot displays the MSAT Interface, a software tool for managing bugs. The main window is titled "Project: Alpha" and contains a table of bugs. The table has columns for S. No., Name (Bug title), Tool, Type, Fix Location, and Assignee. Below the table, there is a "Bug Description" section for the selected bug (XSS_CONFIG). To the right of the table, there is a "Filters" panel with checkboxes for "Select All", "Deselect All", and "tool1" through "tool10". Below the filters, there are sections for "Bugs" (My Bugs, All Bugs) and "Vulnerability Type" (SQL Injection, XSS).

S. No.	Name (Bug title)	Tool	Type	Fix Location	Assignee
1	XSS_CONFIG		XSS	12.4 XSSFILTER.java	Varma
2	EQ_CHECK		EQ	6.3 LoopHelper.java	Max
3	CO_SELF		CO	11.2 StringComparer.java	Un-assigned
4	XSS_REQUEST		XSS	5.4 HttpSender.java	John
5	DMI_EMPTY		DM	3.3 DatabaseHelper.java	Elina
6	BC_EQUALS		BC	2.4HttpReceiver.java	Tom
7	BIT_CHECK		BIT	3.3 NetworkConnect.java	John
8	CN_CLONE		CN	6.7 CloneMessage.java	Max
9	DE_EXCEPTION		DE	2.2 StringPlacer.java	Elina
10	DMI_RANDOM		DMI	3.7 DatabaseConnect.java	Elina
11	EQ_EQUALS		EQ	1.3 StringCheck.java	John
12	IJU_TEST		IJU	9.3 DatabaseTest.java	John
13	IL_LOOP		IL	7.2 FormValidate.java	Tom
14	CI_FINAL		CI	1.6 MessageSender.java	Max
15	SQL_CONSTANT		SQL	3.5 DatabaseInsert.java	Elina

Bug Description

XSS: Anti cross-site scripting filter (XSS_CONFIG)

Wrap the HTTP request object in a specialized HttpServletRequestWrapper that will perform filtering.

Fix Now **Know More**

Filters

☒ Select All ☐ Deselect All

☒ tool1 ☒ tool2 ☒ tool3 ☒ tool4 ☒ tool5 ☒ tool6 ☒ tool7 ☒ tool8 ☒ tool9 ☒ tool10

Bugs

☐ My Bugs ☒ All Bugs

Vulnerability Type

☐ SQL Injection ☐ XSS

■ Single List

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

The screenshot shows the MSAT Interface with a project named 'Alpha'. It displays two separate lists of bugs, one for 'tool1' and one for 'tool2'. Each list has columns for S. No., Name (Bug title), Type, Fix Location, and Assignee. The 'tool1' list has 7 items, and the 'tool2' list has 4 items. A 'Bug Description' section at the bottom provides details for the 'XSS: Anti cross-site scripting filter (XSS_CONFIG)' bug, including a description and buttons for 'Fix Now' and 'Know More'. On the right, there are filters for selecting all/deselecting all bugs, a list of tools (tool1 to tool10) with icons, and checkboxes for 'My Bugs' and 'All Bugs' under the 'Bugs' section, and checkboxes for 'SQL Injection' and 'XSS' under the 'Vulnerability Type' section.

Project: Alpha

tool1

S. No.	Name (Bug title)	Type	Fix Location	Assignee
1	XSS_CONFIG	XSS	12.4 XSSFilter.java	Varma
2	EQ_CHECK	EQ	6.3 LoopHelper.java	Max
3	CO_SELF	CO	11.2 StringComparer.java	Q Unassigned
4	XSS_REQUEST	XSS	5.4 HttpSender.java	John
5	DMI_EMPTY	DM	3.3 DatabaseHelper.java	Elina
6	BC_EQUALS	BC	2.4 HttpReceiver.java	Tom
7	BIT_CHECK	BIT	3.3 NetworkConnect.java	John

tool2

S. No.	Name (Bug title)	Type	Fix Location	Assignee
1	XSS_CONFIG	XSS	12.4 XSSFilter.java	Varma
2	CO_SELF	CO	11.2 StringComparer.java	Q Unassigned
3	XSS_REQUEST	XSS	5.4 HttpSender.java	John
4	B_NAMET2	BT2	5.5 BT2.java	Bob

Bug Description

XSS: Anti cross-site scripting filter (XSS_CONFIG)

Wrap the HTTP request object in a specialized HttpServletRequestWrapper that will perform filtering.

Fix Now

Know More

Filters

☒ Select All ☐ Deselect All

☒ tool1 ☒ tool2 ☒ tool3 ☒ tool4 ☒ tool5 ☒ tool6 ☒ tool7 ☒ tool8 ☒ tool9 ☒ tool10

Bugs

☐ My Bugs ☒ All Bugs

Vulnerability Type

☐ SQL Injection ☐ XSS

■ Separate List

[RQ 1.1] Does a separate list or single list help the user to identify the common bug?

■ Results:

	Single List	Separate List
Task Success	71.43 %	42.85 %
Usability	8.14	5.43
Votes	5	0



- Single List – “effortless to perceive, more user friendly”

MSAT Interface

Project Structure

D:\varma\project_alpha

.idea

src

scripts_module

XSSFilter.java

XSSScan.java

XSSRequestWrapper.java

Project: Alpha - (file_path)/XSSRequestWrapper.java

```

import java.util.regex.Pattern;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

public class XSSRequestWrapper extends HttpServletRequestWrapper {

    public XSSRequestWrapper(HttpServletRequest servletRequest) {
        super(servletRequest);
    }

    @Override
    public String stripXSS(String value) {
        if (value != null) {
            // NOTE: It's highly recommended to use the ESAPI
            // library and uncomment the following line to
            // avoid encoded attacks.

            value = ESAPI.encoder().canonicalize(value);

            // Avoid null characters {color}

            value = value.replaceAll("", "");
        }
    }
}

```

Before	After
<pre> private String stripXSS(String value) { if (value != null) { // NOTE: It's highly recommended to use the ESAPI // library and uncomment the following line to // avoid encoded attacks. value = ESAPI.encoder().canonicalize(value); // Avoid null characters {color} value = value.replaceAll("", ""); } } </pre>	<pre> private String stripXSS(String value) { if (value != null) { // NOTE: It's highly recommended to use the ESAPI // library and uncomment the following line to // avoid encoded attacks. // value = ESAPI.encoder().canonicalize(value); // Avoid null characters {color} value = value.replaceAll("", ""); } } </pre>

<< trace >>

XSS: Anti cross-site scripting filter (XSS_CONFIG)

Status: needs fix

Reported by:

```

// avoid encoded attacks.
// value = ESAPI.encoder().canonicalize(value);

// Avoid null characters
value = value.replaceAll("", "");

```

UX 2 – Lessons

- Analysis View
- Code View
- UX 1 Scalability



- Improvisations for next UX cycle:
 - UX 2 Scalability
 - + new sub RQ's

UX Design Cycle 3

UX 3

- Users: 5
- Sub research questions: 13
- Each session: ~ 120 minutes



UX 3	Analysis View	Code View
RQ 1 (display)		<ul style="list-style-type: none"> • Next • List view • Bug icons • Table view • Vertical view • Horizontal view • Similar boxes • Similar list
RQ 2 (feedback)	<ul style="list-style-type: none"> • Animated Icons • Progress Bar • Popup 	<ul style="list-style-type: none"> • Toasts (alerts) • Spinner (status)
RQ 3 (trace)		<ul style="list-style-type: none"> • Before/After • Table view

[RQ 2.1]

The screenshot displays the SonarCloud web interface for the project 'userstudyJS' (us2019). The browser address bar shows the URL: https://sonarcloud.io/project/issues?id=userstudy2019_userstudyJS&resolved=false. The page header includes the SonarCloud logo, navigation links (Explore, Log in), and a search bar. The main navigation bar shows tabs for Overview, Issues (selected), Measures, Code, and Activity. The right side of the header indicates the date 'August 4, 2019, 6:19 PM' and 'Version not provided'.

On the left, the 'Filters' sidebar is expanded, showing the following counts:

- Type
 - Bug: 300
 - Vulnerability: 0
 - Code Smell: 1.1k
 - Security Hotspot: 0
- Severity
 - Blocker: 0
 - Critical: 5
 - Major: 900
 - Minor: 450
 - Info: 0
- Resolution
- Status
- Security Category

The main content area displays a list of issues for the file 'index.js'. The issues are:

- 2 duplicated blocks of code must be removed.** (Code Smell, Major, Open, Not assigned, 30min effort, 3 months ago, pitfall)
- Add a new line at the end of this file.** (Code Smell, Minor, Open, Not assigned, 1min effort, 3 months ago, convention)
- Remove this useless assignment to local variable "c".** (Code Smell, Major, Open, Not assigned, 15min effort, 3 months ago, L11, cert, cwe, unused)
- Remove the declaration of the unused 'c' variable.** (Code Smell, Minor, Open, Not assigned, 5min effort, 3 months ago, L11, unused)
- Remove this useless assignment to local variable "d".** (Code Smell, Major, Open, Not assigned, 15min effort, 3 months ago, L12, cert, cwe, unused)

Navigation controls at the top of the issue list include arrows to select issues and to navigate, a refresh icon, and summary statistics: '1 / 1,355 issues' and '38d effort'.

[RQ 2.1]

How usable are each feedback functionality compared to the scenario of using unified UI to native UIs?

Result	MSAT-UI*	Native UIs
Animated Icons	8.4	0
Progress Bar	7.6	0.8
Pending Status Popup	7.8	0
Alerts	9.2	0
Status	8.8	4



Almost all users agreed the ideas being novel and hardly present with native UIs.

*Multiple Static Analysis Tools – User Interface

MSAT Interface

Project Structure

D:\varma\project_alpha

idea

src

scripts_module

XSSFilter.java

XSSScan.java

XSSRequestWrapper.java

Project: Alpha - (file_path)/XSSRequestWrapper.java

```

import java.util.regex.Pattern;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

public class XSSRequestWrapper extends HttpServletRequestWrapper {

    public XSSRequestWrapper(HttpServletRequest servletRequest) {
        super(servletRequest);
    }

    @Override
    public String getParameter(String name) {
        if (name == null) return null;

        in
        S
        fo
        }
        re
        }

        @Override
        public String getHeader(String name) {
            re
            }
        }
    }

```

S. No.	Bug Name	Before	After	Status
2	XSS: Anti cross-site scripting filter	L44: value = value.replace("", "");	L44: // avoid encoded attacks. L45: value = ESAPI.encoder().canonicalize(value); L46: // Avoid null characters L47: value = value.replaceAll("", "");	fixed
3	SA: Self comparison of value with itself	L44: value = value.replaceBy("", "");	L44: value = value.replace("", ""); L34: // value = ESAPI.encoder().canonicalize(value);	fixed

Fixed

Findings

	Analysis View	Code View
RQ 1 (display)	<ul style="list-style-type: none">• Single list	<ul style="list-style-type: none">• List view• Single icon• Table view• Similar list
RQ 2 (feedback)	<ul style="list-style-type: none">• Animated Icons• Progress Bar• Popup	<ul style="list-style-type: none">• Toasts (alerts)
RQ 3 (trace)	<ul style="list-style-type: none">• Adjectives	<ul style="list-style-type: none">• Table view

Limitations

- Number of participants: > **5**
 - UX 1 – 5
 - UX 2 – 7
 - UX 3 – 5
- Priming, Recency bias: Latin Square partition
- Closed Study, Design Tool

balsamiq®

- ❖ Why You Only Need to Test with 5 Users. url: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users>
- ❖ Balsamiq – Rapid, effective and fun wireframing software. url: <https://balsamiq.com/>

Future Work

- Q. Would having **tabs** help scale the tools visibility with bugs results?
- Q. Do **graphs** help in understanding the bugs reported?
- RQ 4: How is **teamwork** facilitated in bug fixing in context of multiple tools?

... many more!

Summary

- Importance of Static Analysis tools
- Usage of Multiple Static Analysis tools
- Need for a single user interface for multiple tools
- This thesis work followed UX Design Cycle to achieve usable prototypes focussing on primary research questions such as,
 - How to display results of the same codebase from different analysis tools?
 - What feedback works to know that bug fixing is on-going?
 - How to carry traceability of bug fixing?