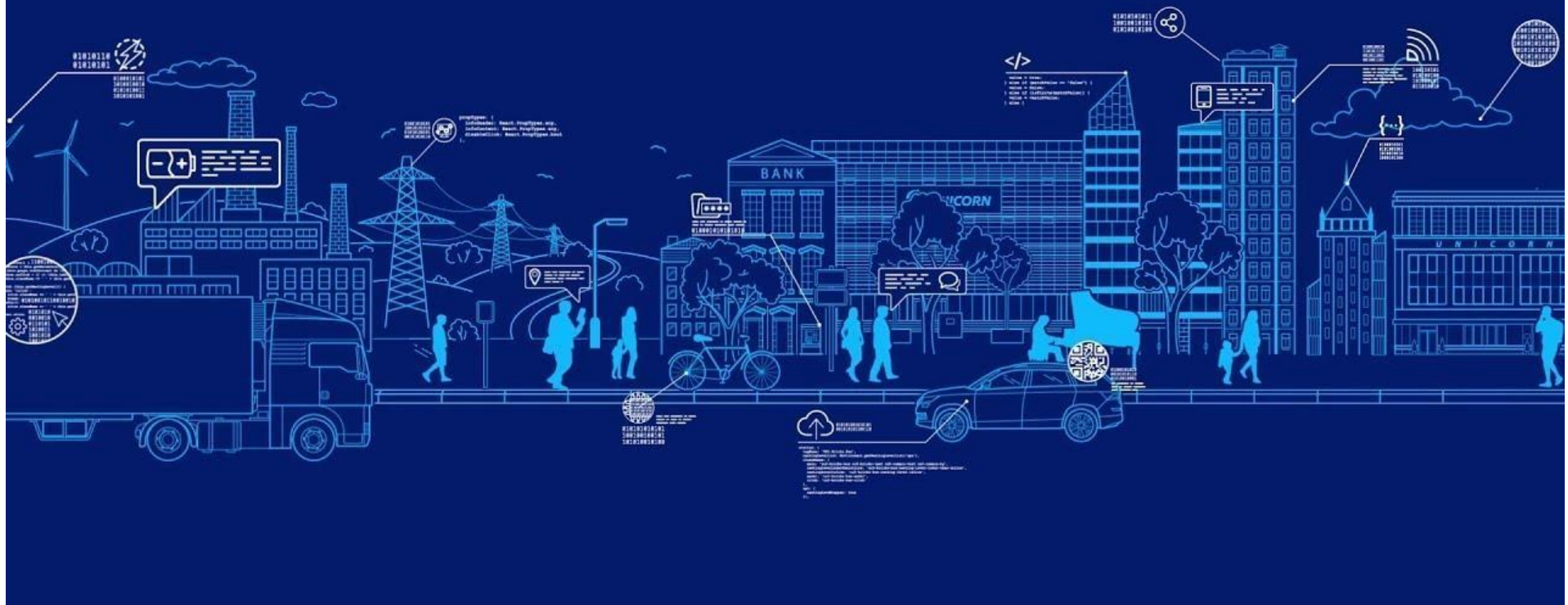


# **Integration of Multiple Static Analysis Tools in a Single Interface**

---

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

# Software Everywhere



❖ <https://unicorn.com/en/software-everywhere>

- “ \$1.1 Trillion in Assets Affected by Software Bugs in 2016 “

- Software Fail Watch Annual Report,

[Tricentis](#)



❖ <https://www.tricentis.com/news/software-fail-watch-says-1-1-trillion-in-assets-affected-by-software-bugs-in-2016/>

# Static Code Analysis

- It helps in prevention of bugs.
- It examines code without execution.
- Detects vulnerabilities :
  - Injections
  - Cross Site Scripting (XSS)
  - Buffer Overflow, and Dead Code etc.



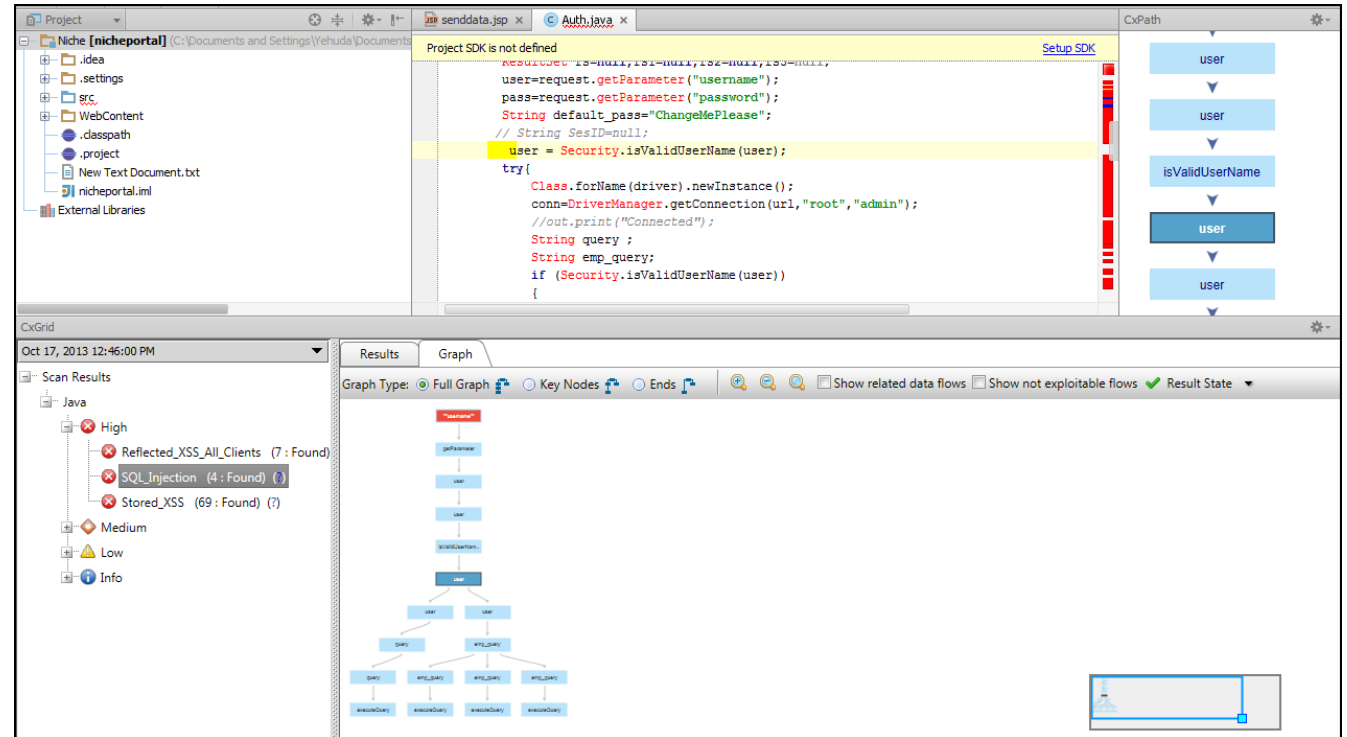
❖ Designing code analyses for Large Software Systems (DECA). url: <https://www.hni.uni-paderborn.de/swt/lehre/deca/>.

# Static Code Analysis



## ■ Tools :

- IDE notifications
- IDE tools
- Dedicated tools
- Linters
- CLI tools



- ❖ Checkmarx – Application Security Testing and Static Code Analysis. url: <https://www.checkmarx.com/>
- ❖ CxViewer - Plugins | JetBrains, url: <https://plugins.jetbrains.com/plugin/7593-cxviewer>
- ❖ FindBugs™ - Find Bugs in Java Programs. url: <http://findbugs.sourceforge.net/>.



# Static Code Analysis

- Johnson et. al.
  - Tool output
  - Result understandability

## Usability Issues

- Christakis et. al.

- ❖ Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. 2013. Why don't software developers use static analysis tools to find bugs?. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, Piscataway, NJ, USA, 672-681.
- ❖ Maria Christakis and Christian Bird. 2016. What developers want and need from program analysis: an empirical study. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE 2016)*. ACM, New York, NY, USA, 332-343. DOI: <https://doi.org/10.1145/2970276.2970347>

# Multiple Tools

- Developers use multiple static analysis tools each having own coverage.
- Research trends:
  - Prioritise the bug warning alerts  
( Flynn et. al. )
  - Merges 3 tools for Java to show warnings  
( Meng et. al. )
- ❖ Lori Flynn, William Snaveley, David Svoboda, Nathan VanHoudnos, Richard Qin, Jennifer Burns, David Zubrow, Robert Stoddard, and Guillermo Marce-Santurio. 2018. Prioritizing alerts from multiple static analysis tools, using classification models. In *Proceedings of the 1st International Workshop on Software Qualities and Their Dependencies* (SQUADE '18). ACM, New York, NY, USA, 13-20. DOI: <https://doi.org/10.1145/3194095.3194100>
- ❖ N. Meng, Q. Wang, Q. Wu and H. Mei, "An Approach to Merge Results of Multiple Static Analysis Tools (Short Paper)," *2008 The Eighth International Conference on Quality Software*, Oxford, 2008, pp. 169-174.doi: 10.1109/QSIC.2008.30

# Multiple Tools

- Tricorder
  - ReviewBot
  - Separate bug coverage by separate tool
  - Evaluation: Summative – Click rates
- Parfait
  - Scalability ( easy , expensive analysis )
  - Precision ( bug track – real, no, potential )

(Sadowski et. al. )

(Cifuentes et. al. )

But **USABILITY** is not addressed...

- ❖ Caitlin Sadowski, Jeffrey van Gogh, Ciera Jaspan, Emma Söderberg, and Collin Winter. 2015. Tricorder: building a program analysis ecosystem. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1 (ICSE '15)*, Vol. 1. IEEE Press, Piscataway, NJ, USA, 598-608.
- ❖ Cristina Cifuentes and Bernhard Scholz. 2008. Parfait: designing a scalable bug checker. In *Proceedings of the 2008 workshop on Static analysis (SAW '08)*. ACM, New York, NY, USA, 4-11. DOI=<http://dx.doi.org/10.1145/1394504.1394505>



# Problem Statement

- How to integrate the results of multiple static analysis tools

in a unified user interface?

❖ 3 Research Questions

# Research Question 1

- How to display results of the same codebase from  
different analysis tools?

# What Current Tools do? - RQ 1: .. display results!

## ■ FindBugs

The screenshot shows the FindBugs IDE interface. The left pane displays a project tree with the following structure:

- edu.umd.cs.findbugs.config (3)
- edu.umd.cs.findbugs.filter (1)
- edu.umd.cs.findbugs.util (1)
  - Medium (1)
    - Bad practice (1)
      - Stream not closed on all paths (1)
        - Method may fail to close stream (1)
          - edu.umd.cs.findbugs.util.Util.getXMLType (1)
- edu.umd.cs.findbugs.visitclass (1)
- edu.umd.cs.findbugs.workflow (2)
- java.util (2)

The right pane shows the source code of `Util.java` in `edu.umd.cs.findbugs.util`. The code is as follows:

```
97     assert true;
98   }
99 }
100 static final Pattern tag = Pattern.compile("^\\s*<\\w+");
101 public static String getXMLType(InputStream in) throws IOException {
102     if (!in.markSupported())
103         throw new IllegalArgumentException("Input stream does not support mark");
104     in.mark(5000);
105     BufferedReader r = null;
106     try {
107         r = new BufferedReader(Util.getReader(in), 2000);
108         String s;
109         int count = 0;
110         while (count < 4) {
111             s = r.readLine();
112             if (s == null)
113                 break;
114             Matcher m = tag.matcher(s);
115             if (m.matches())
116                 return m.group(1);
117             count++;
118         }
119     } finally {
120         if (r != null)
121             r.close();
122     }
123 }
```

The bug report at the bottom of the window is titled "Method may fail to close stream" and describes the issue with the `getXMLType` method. The report text is:

edu.umd.cs.findbugs.util.Util.getXMLType(InputStream) may fail to close stream  
At Util.java:[line 108]  
In method edu.umd.cs.findbugs.util.Util.getXMLType(InputStream) [Lines 102 - 123]  
Need to close java.io.Reader

**Method may fail to close stream**  
The method creates an IO stream object, does not assign it to any fields, pass it to other methods that might close it, or return it, and does not appear to close the stream on all paths out of the method. This may result in a file descriptor leak. It is generally a good idea to use a `finally` block to ensure that streams are closed.

<http://findbugs.sourceforge.net/>

UNIVERSITY OF MARYLAND

# What Current Tools do? - RQ 1: .. display results!

## ■ Tricorder

```
package com.google.devtools.staticanalysis;
```

```
public class Test {
```

▼ Lint      Missing a Javadoc comment.  
Java  
1:02 AM, Aug 21

[Please fix](#)

[Not useful](#)

```
public boolean foo() {  
    return getString() == "foo".toString();  
}
```

▼ ErrorProne      String comparison using reference equality instead of value equality  
StringEquality  
1:03 AM, Aug 21  
(see <http://code.google.com/p/error-prone/wiki/StringEquality>)

[Please fix](#)

Suggested fix attached: [show](#)

[Not useful](#)

```
    }  
  
    public String getString() {  
        return new String("foo");  
    }  
}
```

## Research Question 2


- What feedback works to know that the bug fixing is on-going?
- What current tools do?
  - Traditional approach – Nightly Builds

## Research Question 3

- How to carry traceability of bug fixing?


# What Current Tools do? - RQ 3: .. traceability!

## ■ Teamscale




**Added db2 database mapping after reading forum post**  
by [Daniel Lewis](#) in revision [91687a1146419dd23ceaed299185512696643dc1](#) (git)  
Files: 11 changed  
Findings: 0 4 12 1

Jul 17 2014 10:53



**Add getDelegationState() in DelegateTask.**  
by [Anya Hill](#) in revision [812b1e277d844fa48307bcd7c692a6f395c85fbb](#) (git)  
Files: 14 changed  
Findings: 0 3 12 5

Jul 17 2014 10:30

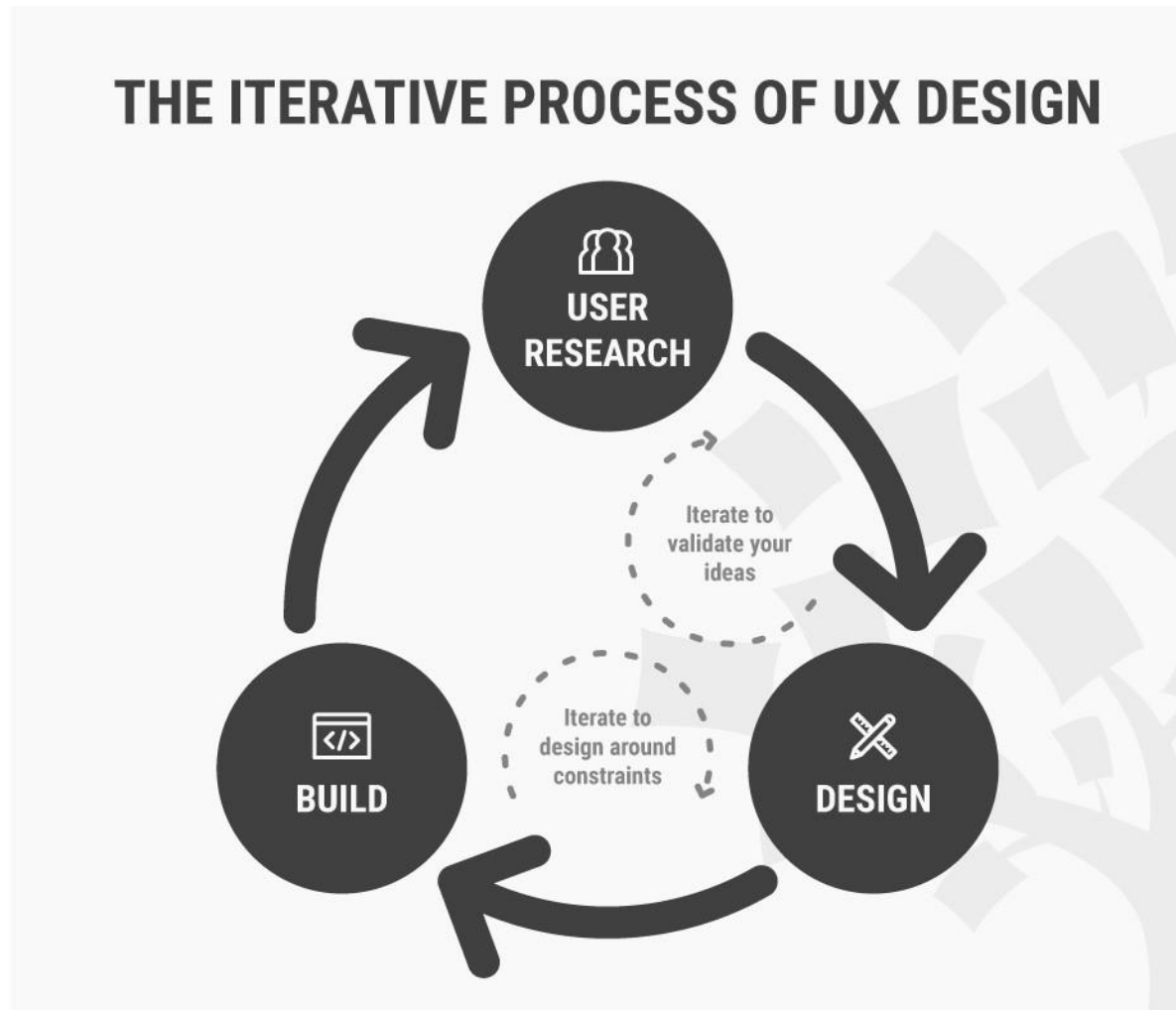


**TASK\_TIMEOUT**  
by [Jacob Nelson](#) in revision [997da57af6f2c08d504473d3e9837788b7592dcb](#) (git)  
Files: 14 changed  
Findings: 0 5 12 3

Jul 17 2014 08:46

❖ Teamscale. url: <https://www.cqse.eu/en/products/teamscale/features/>.

# Our Approaches



- ❖ 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

# Our Approaches – research existing scenarios!

## ■ Complex datasets:

- Dix et. al. - complex grouping and linking of datasets for Spreadsheets application

Design lesson : extensibility of columns



## ■ Issue tracker

- Baysal et. al. :

❖ Information overload

❖ Expressiveness



❖ Alan Dix, Rachel Cowgill, Christina Bashford, Simon McVeigh, and Rupert Ridgewell. 2016. Spreadsheets as User Interfaces. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '16)*, Paolo Buono, Rosa Lanzilotti, and Maristella Matera (Eds.). ACM, New York, NY, USA, 192-195. DOI: <https://doi.org/10.1145/2909132.2909271>

❖ Olga Baysal, Reid Holmes, and Michael W. Godfrey. 2014. No issue left behind: reducing information overload in issue tracking. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*. ACM, New York, NY, USA, 666-677. DOI: <https://doi.org/10.1145/2635868.2635887>

# Example: RQ 1 - .. display results!

## ■ Prototype 1

The screenshot shows the 'D SAT Interface' window. At the top, it says 'Project: Alpha'. Below this is a table of bugs. The table has columns: Name (Bug title), Tool, Type, Fix Location, and Assignee. The first row, 'FI\_EMPTY', is highlighted in blue. Below the table is a section titled 'Bug Description' which contains the text: 'FI: Empty finalizer should be deleted (FI\_EMPTY)' and 'Empty finalize() methods are useless, so they should be deleted.' There are two buttons: 'Fix Now' and 'Know More'. On the right side of the interface is a 'Filters' panel with checkboxes for 'toolLong', 'toolShort', 'My Bugs', and 'All Bugs', and a section for 'Vulnerability Type' with checkboxes for 'SQL Injection' and 'XSS'.

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

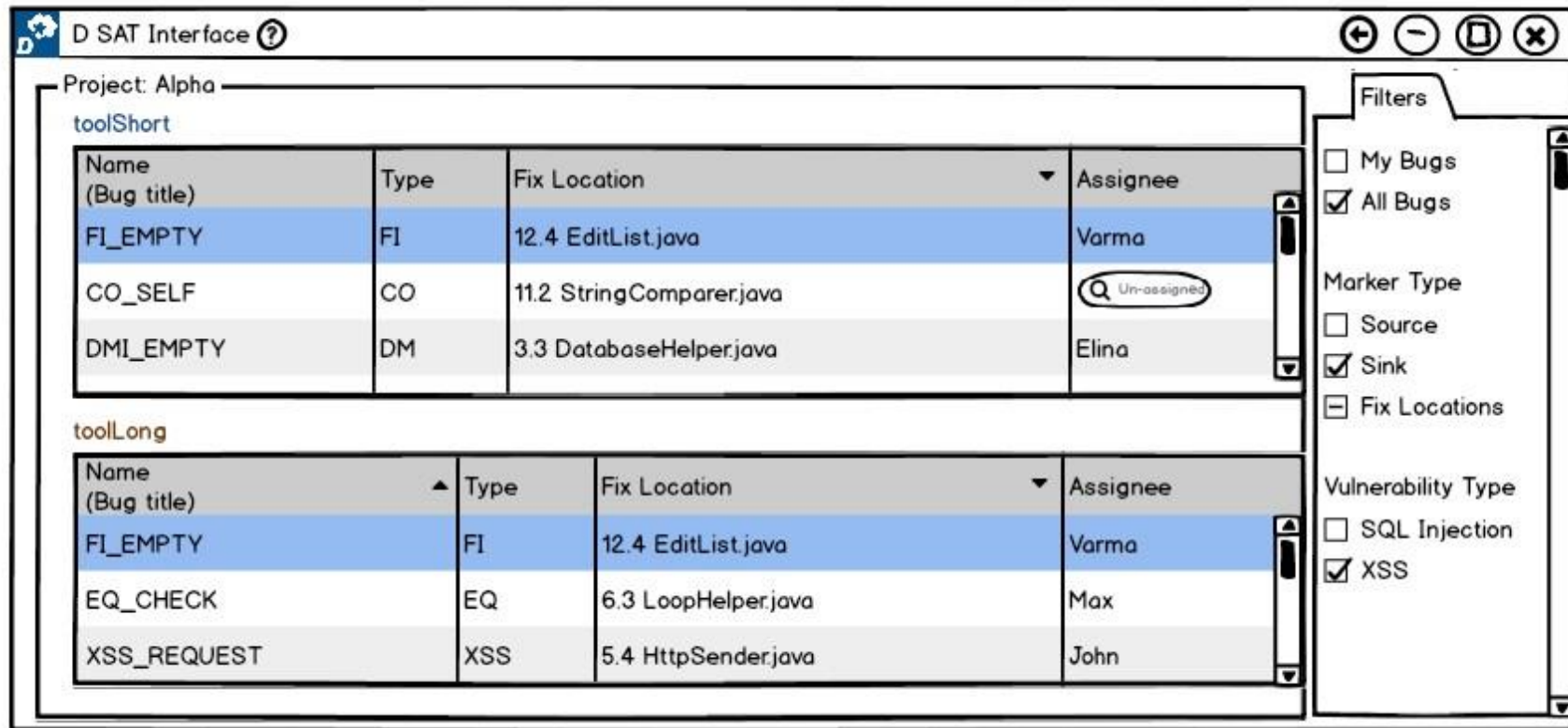
**Vulnerability Type**

☐ SQL Injection

☐ XSS

# Example: RQ 1 - .. display results!

## ■ Prototype 2



D SAT Interface

Project: Alpha

toolShort

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

toolLong

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

Filters

- ☐ My Bugs
- ☒ All Bugs

Marker Type

- ☐ Source
- ☒ Sink
- ☐ Fix Locations

Vulnerability Type

- ☐ SQL Injection
- ☒ XSS

# Evaluation

- Experimental Design
  - Recruit Test Users
  - Order of evaluation altered
  - Usability inspection methods: Cognitive Walkthrough, Heuristic Evaluation
  - Perform Tasks
    - Example: Find a bug which is reported in common by available tools.

# Evaluation – Usability Inspection Methods

## ■ Cognitive Walkthrough

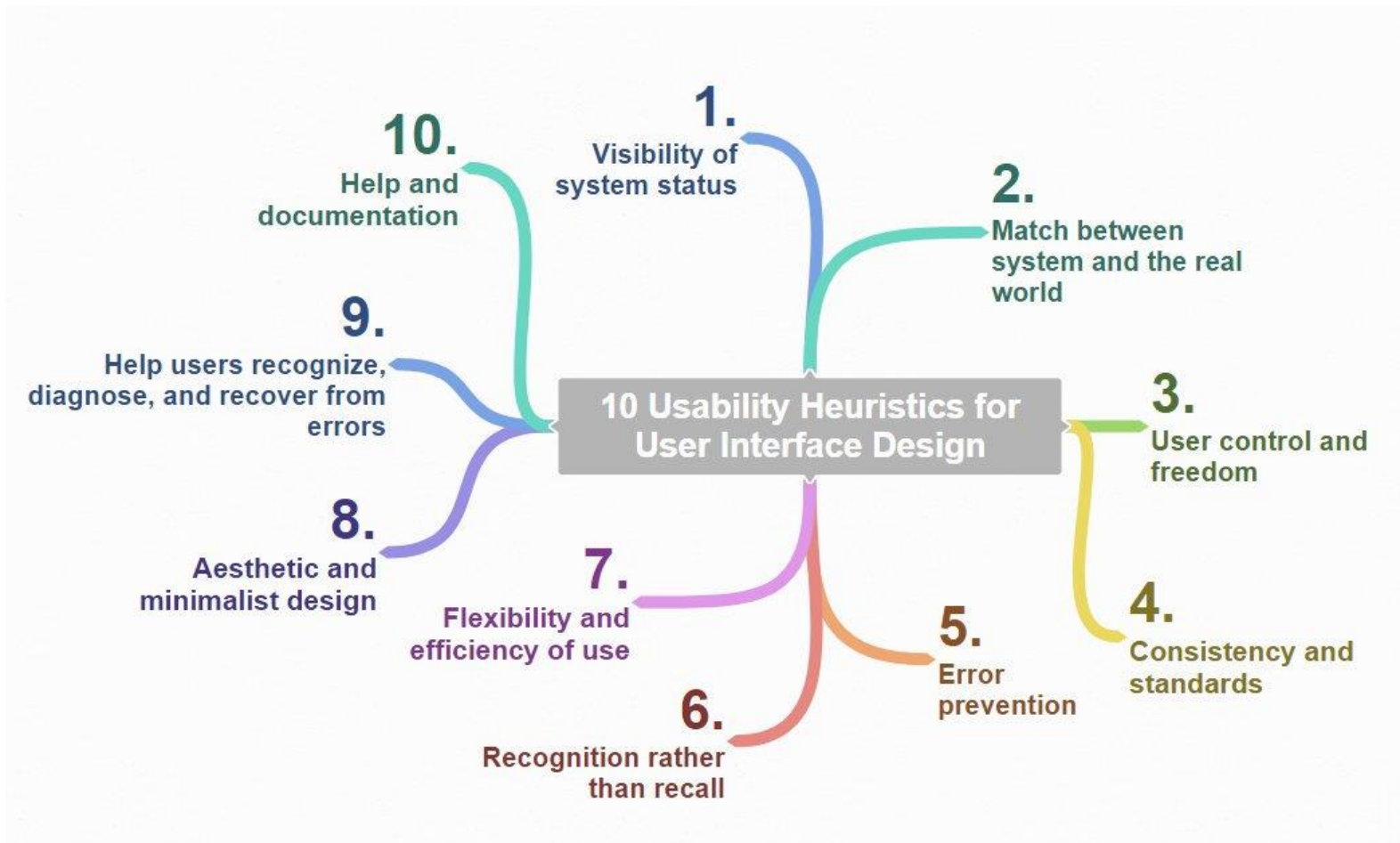
For each step to a predefined task, the following aspects are analysed.

- Will the user try and achieve the right outcome?
- Will the user notice that the correct action is available to them?
- Will the user associate the correct action with the outcome they expect to achieve?
- If the correct action is performed; will the user see that progress is being made towards their intended outcome?

❖ Jakob Nielsen. “Usability inspection methods”. In: Conference companion on Human factors in computing systems. ACM. 1994, pp. 413–414.

# Evaluation – Usability Inspection Methods

## ■ Heuristic Evaluation



❖ 10 Heuristics for User Interface Design: Article by Jakob Nielsen. Available online at <https://www.nngroup.com/articles/ten-usability-heuristics/>, checked on 5/1/2019. Image credits: Miran Janezic

# Evaluation – Usability Inspection Methods

## ■ Heuristic Evaluation

Each problem w.r.t. a heuristic is rated accordingly; 0 – 4

**0** - do not agree this is a usability problem

**1** - cosmetic problem

**2** - minor usability problem

**3** - major usability problem ( important to fix )

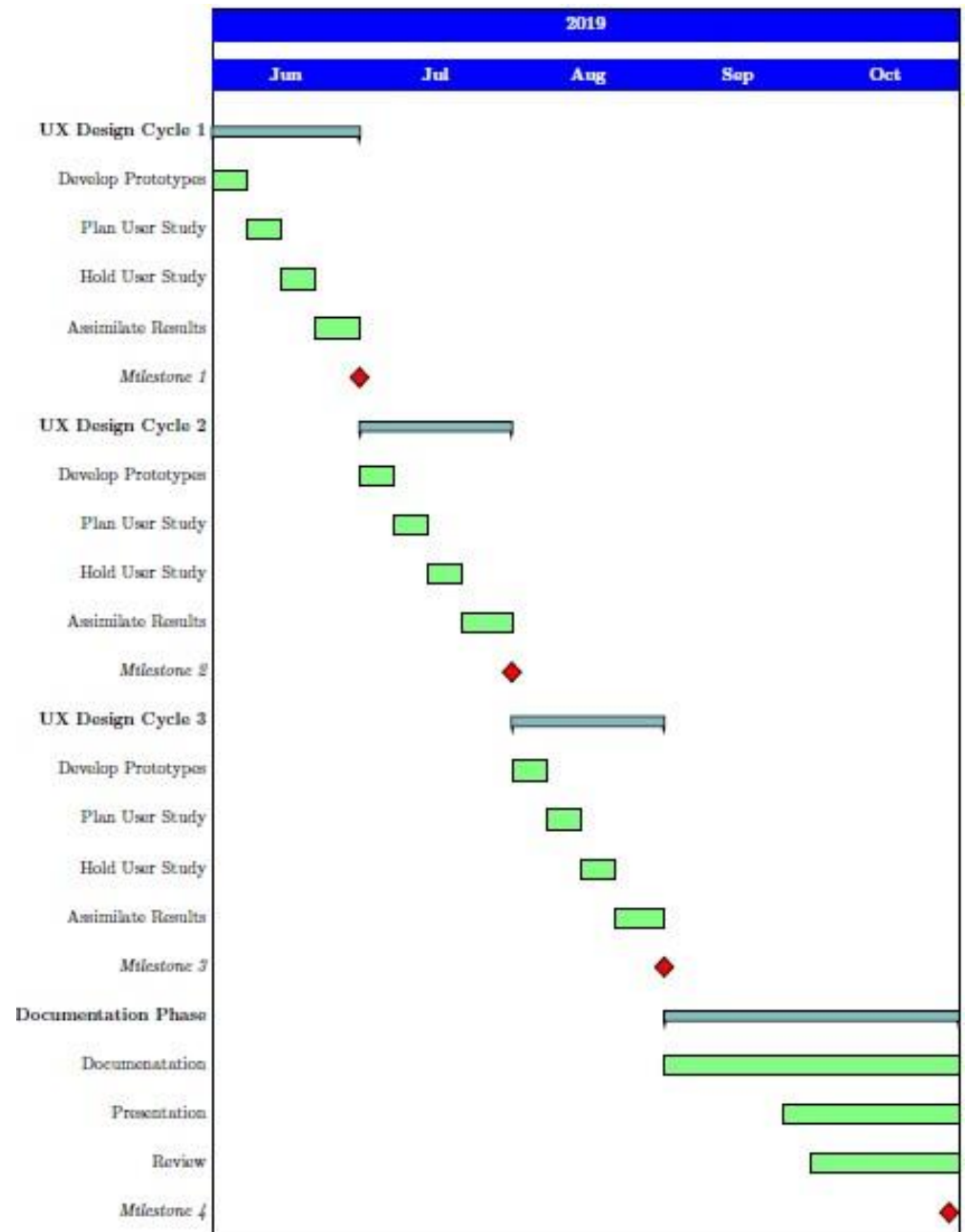
**4** - usability catastrophe ( imperative to fix )

Comparative Study

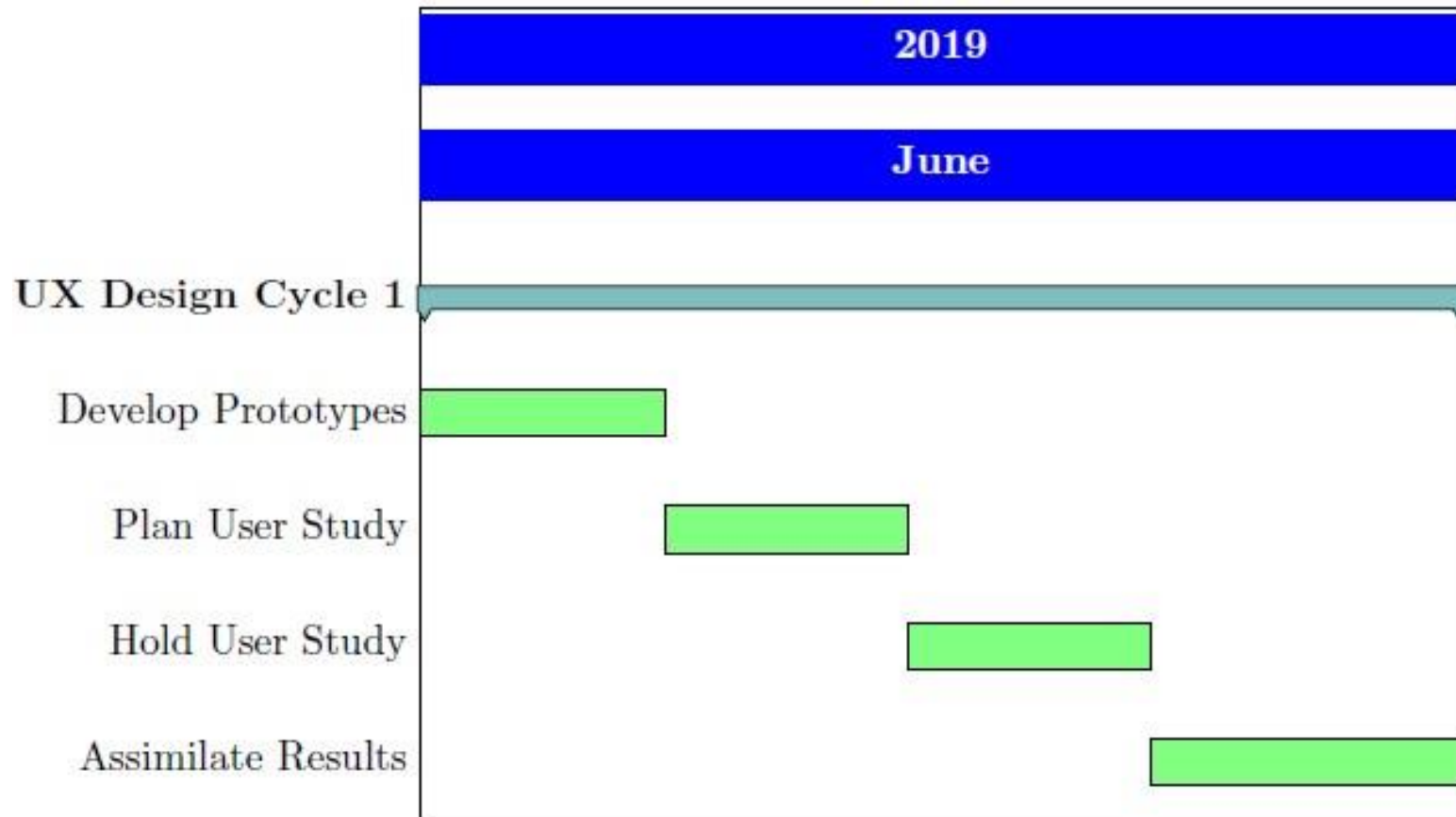


# Time Plan

- Official Time: 5 Months
- Milestones: 4

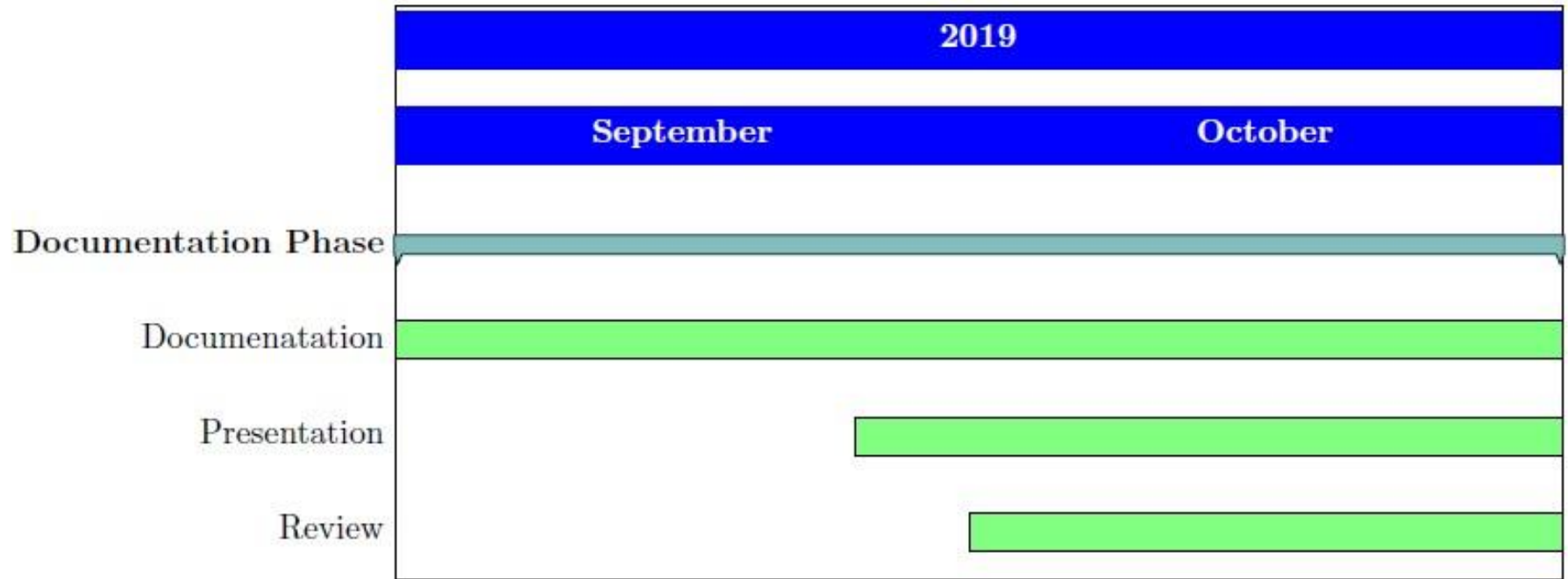


# Milestones 1 2 3



Similarly in July and August...

# Milestone 4



# Summary

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