

# Teachure

---

An Eclipse plug-in to help users learn features and shortcuts that they are not currently using

Eclipse has many features which users may not be aware of

Goal is to replace or minimize tedious and inefficient typing

# Motivation

- Existing plugin called “MouseFeed” which notifies the user of the keyboard shortcut whenever they use a menu item button
  - Requires the user to know that the feature exists already
  - Limited to buttons clicked in the IDE
- Examples of our proposal:
  - Typing several import statements rather than cmd-shift-o
  - Adding “//” to the front of several lines to add comments rather than selecting and hitting cmd-/
  - Changing a variable name several times, rather than using the “Refactor->Rename” feature (shift-cmd-r)

# Approach

- Track changes the user makes in the document
- See if the user is typing things they could be using features or hotkeys for
- Display a notification to the user about the feature/hotkey they could be using instead

```
50 public static void parseData(String filename, Set<String> characters,  
51     Map<String, List<String>> books) throws MalformedURLException {  
52     // Why does this method accept the Collections to be filled as  
53     // parameters rather than making them a return value? To allow us to  
54     // "return" two different Collections. If only one or neither Collection  
55     // needs to be returned to the caller, feel free to rewrite this method  
56     // without the parameters. Generally this is better style.  
57     BufferedReader reader = null;  
58     try {  
59         reader = new BufferedReader(new FileReader(filename));  
60  
61         // Construct the collections of characters and books, one  
62         // <character, book> pair at a time.  
63         String inputLine;  
64         while ((inputLine = reader.readLine()) != null) {  
65
```

Console [x] Console JUnit Git Staging FileNotFoundException.class

No consoles to display at this time.

KeySpeed Tip: x

Try using `cmd- /` to add  
comments to several lines.

# Challenges and Risks

- Limited by the capabilities of and our familiarity with the Eclipse API
- Complex parsing of input
  - How do we determine when to fire off a notification?

# Logic Visualization IDE Plugin

Candice Miao  
Leo Gao



## Motivation

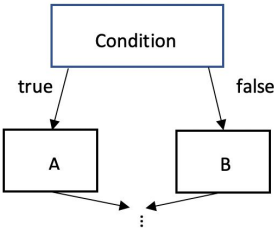
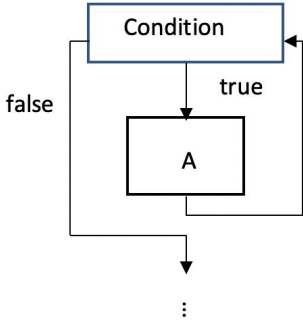
- Make understanding foreign code easier
- Help programmer debug
- Facilitate the learning experience for new learners in Computer Science



# Approach

Building an IDE plugin to visualize the code and to translate graphs into code.

- Easier for people to understand what the functionality of the code (code -> graph)
- Help people focus more on the algorithm and logic rather than syntax during programming (graph -> code)
- Potentially increase efficiency in reading and writing code (both ways)



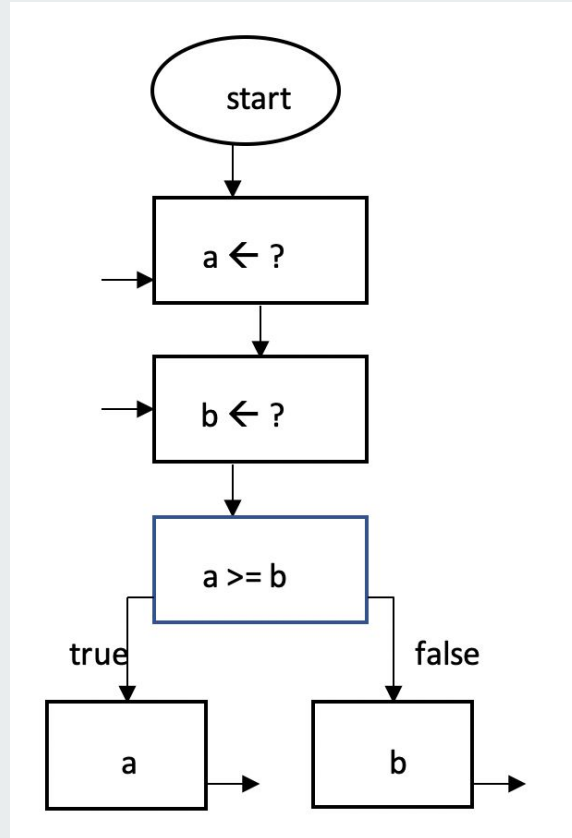
```
X ← a + b
```

```
Sum(int a, int b);
```

# Example

Max function in Java:

```
Int max(int a, int b) {  
    If(a >= b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```





## Challenges and Risks

1. Dealing with different languages
  - a. Create different graphing systems for each language
  - b. Focus on one language for this project due to the time restriction
2. How to represent data structures



# GitUp

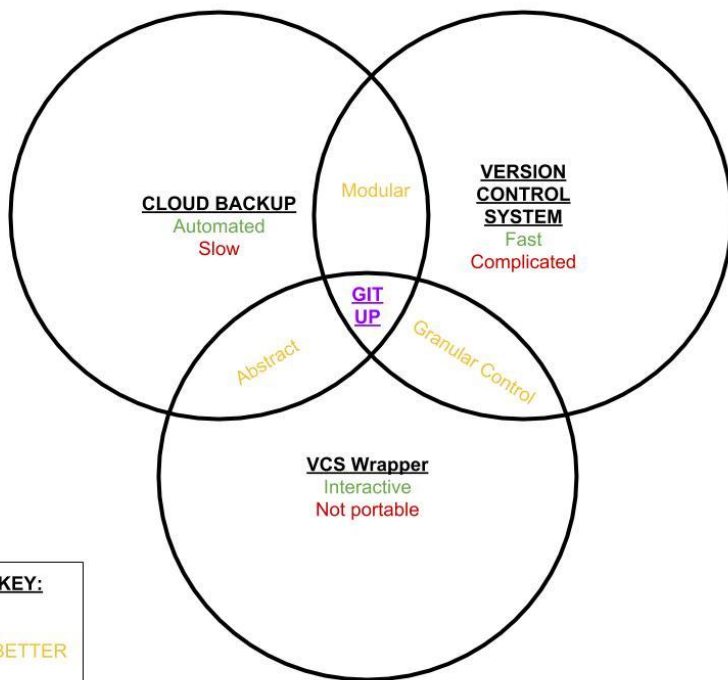
*A simple, portable backup manager with practical version control.*

by

Kaushal Mangipudi and Gerard Gaimari

# Why GitUp?

Cloud based backup systems, version control systems, and VCS wrappers are **non-optimal** and **clunky** for single-user projects



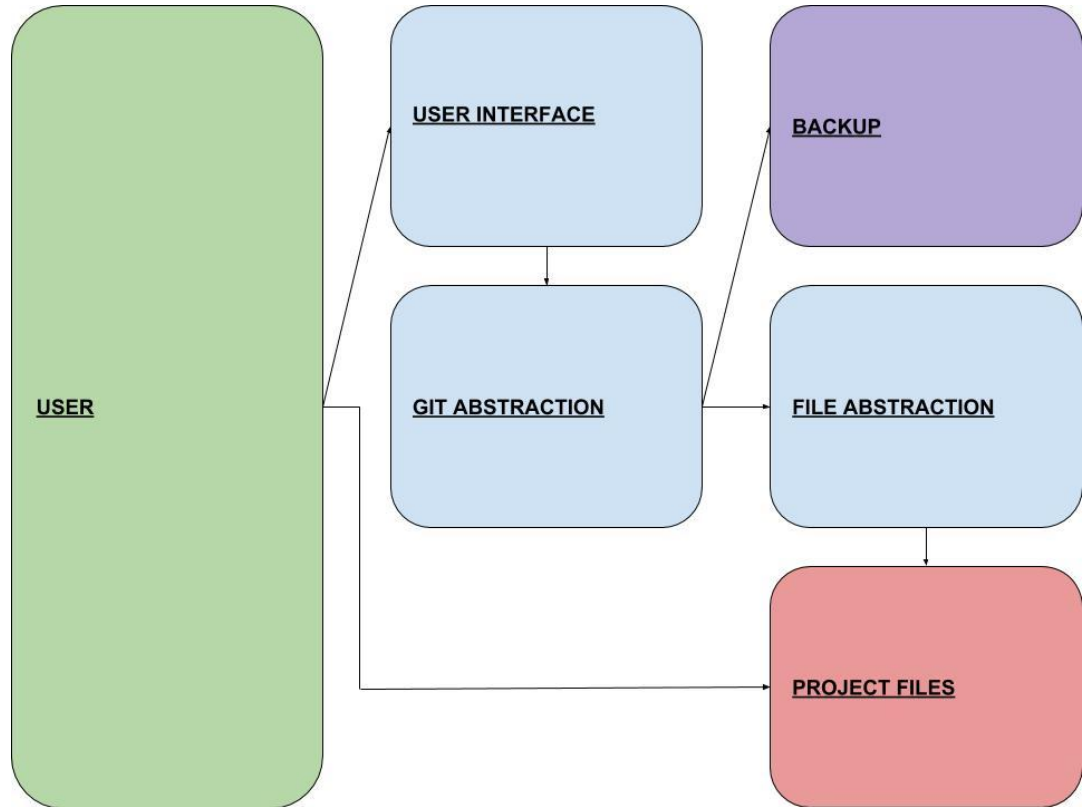
Gitup preserves many of its alternatives' strengths while avoiding their weaknesses

**FEATURES KEY:**  
USEFUL  
HARMFUL  
COULD BE BETTER

# How does GitUp work?

## Sample Use Case:

1. GitUp added to project root
2. GitUp automatically tracks and updates backup
3. Project is moved, GitUp services are unaffected
4. User interacts with GitUp to view and restore past versions of files



# Potential Roadblocks

- How will we keep GitUp focused on providing **essential functionality with maximum simplicity?**
  
- We're collecting data! Take our survey at <https://tinyurl.com/gitupsurvey>

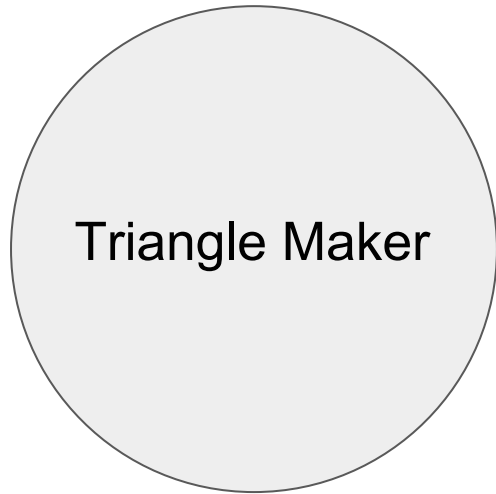
# Are You Tested?

Daniel, Jessica

# Motivation

- Incorporate existing framework
- Navigate through documentation
- Documentation and actual implementation

# Approach

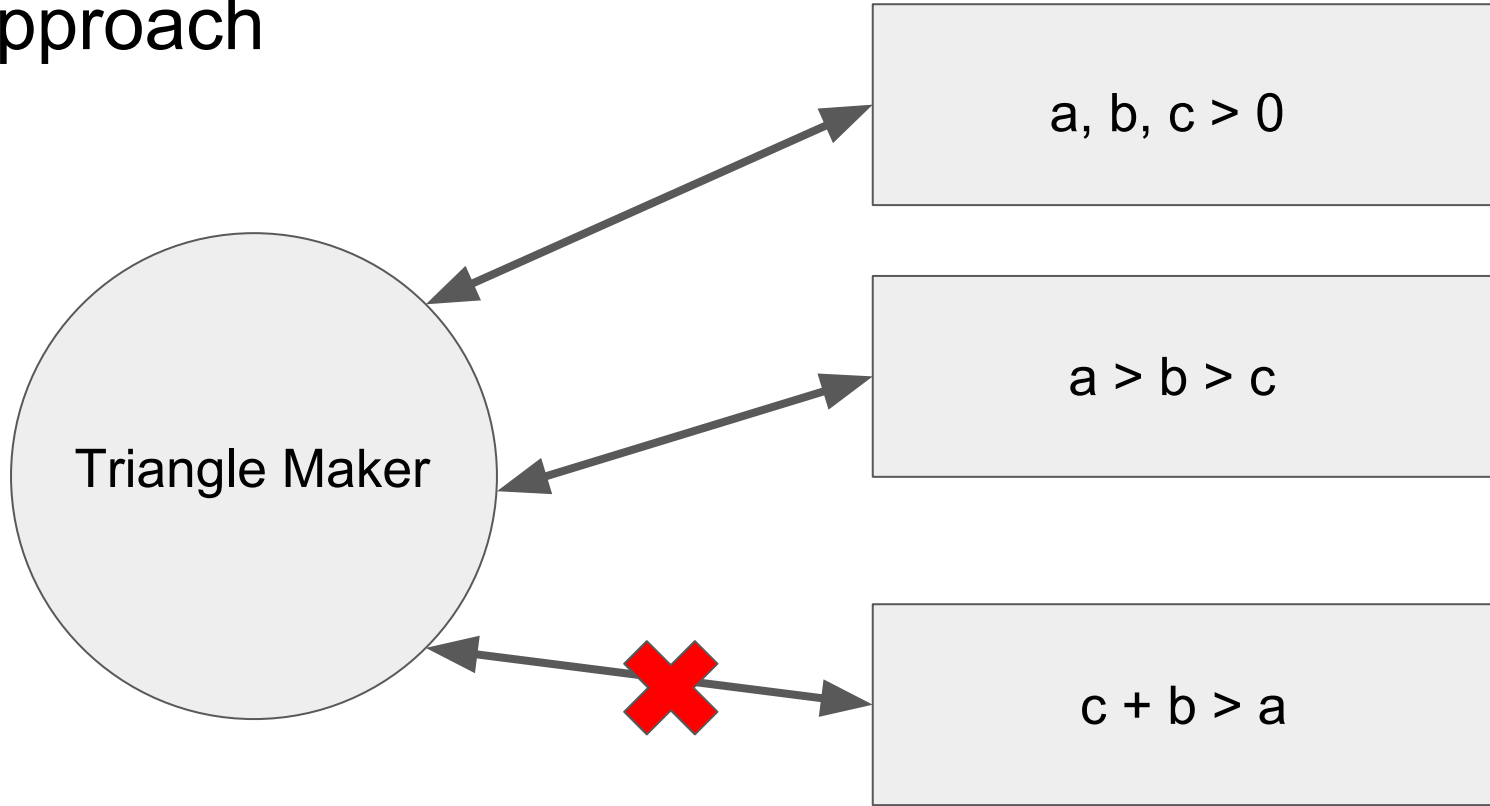


$$a, b, c > 0$$

$$a > b > c$$

$$c + b > a$$

# Approach





# Bug Reporter Edge Extension

By Annie Gesellchen & Ben Celsi

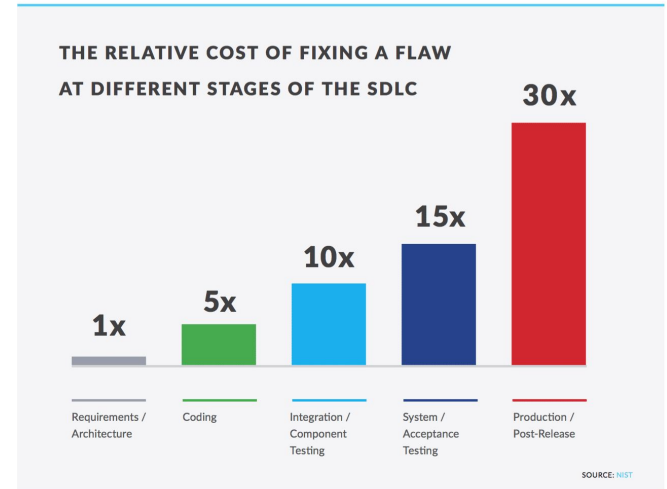




# Problem & Goal

- Users encounter bugs in websites all the time
- These bugs are costly
- It's tedious to try to report these bugs
- Bugs often go unnoticed until developers find them

Goal: To reduce the amount of time that bugs go unnoticed by making bug reporting easy for users





## Solution

Our solution is an Edge browser extension that allows users to easily report bugs they encounter.

- The extension lets users describe the bug and optionally upload a screen capture of it.
- The extension may also automatically store some information about the state of the site at the time of the bug (such as error logs).
- This information is then stored, and sent to companies (either automatically or by request.)



**Questions?**



# RewinDB

---

By: Audrey Tran and Hang Bui

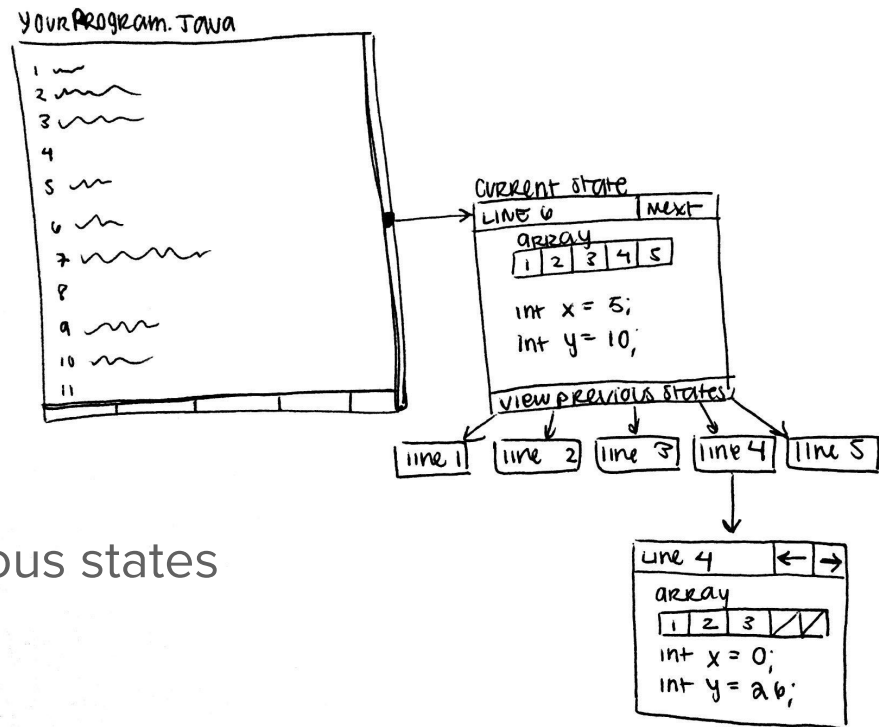
# Motivation

- Time consuming cyclic debugging
- Current solutions:
  - Capture entire stack frame over every step. Heavy!
  - Chronon - DVR for Java
    - Wait until program to finish executing
    - No live debugging



# Approach

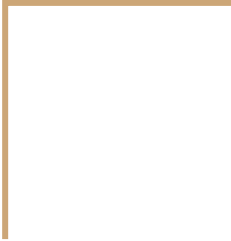
- “Records” instead of undo
  - Catch irreproducible bugs
  - Solves irreversible execution
  - E.g. network calls
- Only store a small number of previous states
  - Faster!
  - Less memory
  - Often times, we only overstep by a few steps
- Still allows for step-by-step live debugging



# Risks and Challenges

- What data to store? How?
- Heavy overhead during recording
- User friendly UI





# corollary

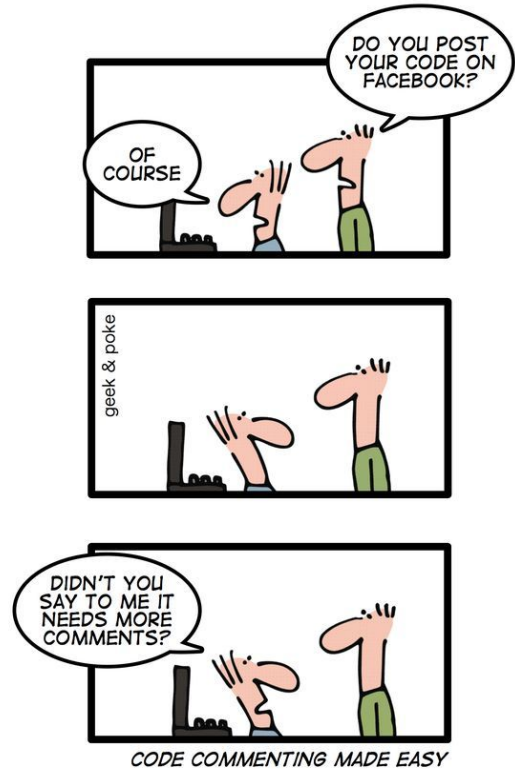
a Java documentation and  
verification tool



# Motivation

Many people have trouble writing good comments

- Lombok Project: good, but not quite right
- Javadocs: needs verification

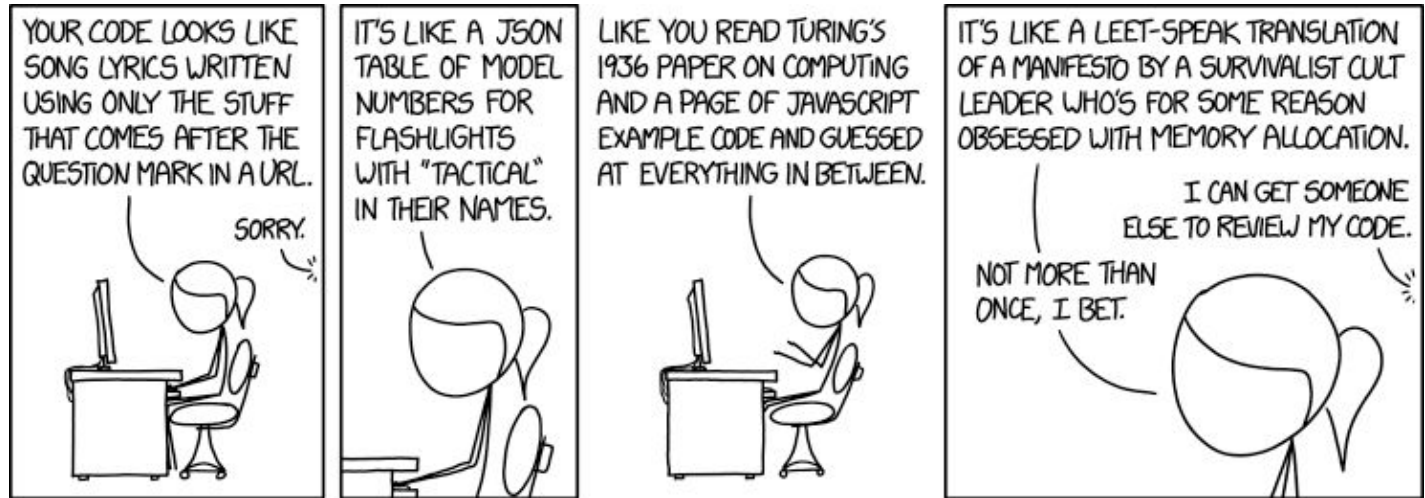


# Corollary

A tool that:

Helps understand the functionality of a piece of code

Verifies that a method keeps its invariants and conditions



# Example

```
/**
 * Combines the two dictionaries and returns the result.
 * @param dict1 the first dictionary
 * @param dict2 the second dictionary
 * @return combination of the two dictionaries
NEW * @precondition dict1 is not null, dict2 is not null
NEW * @invariant dict1 is immutable, dict2 is immutable
 */
public Set<String> combineDictionaries(Set<String> dict1, Set<String> dict2) {
    for (String word : dict2) {
        dict1.add(word); // Output - InvariantException: dict1 is immutable
    }
    return dict1;
}
```

---

---

# IDE Intelligent Tutorials (IDE-IT)

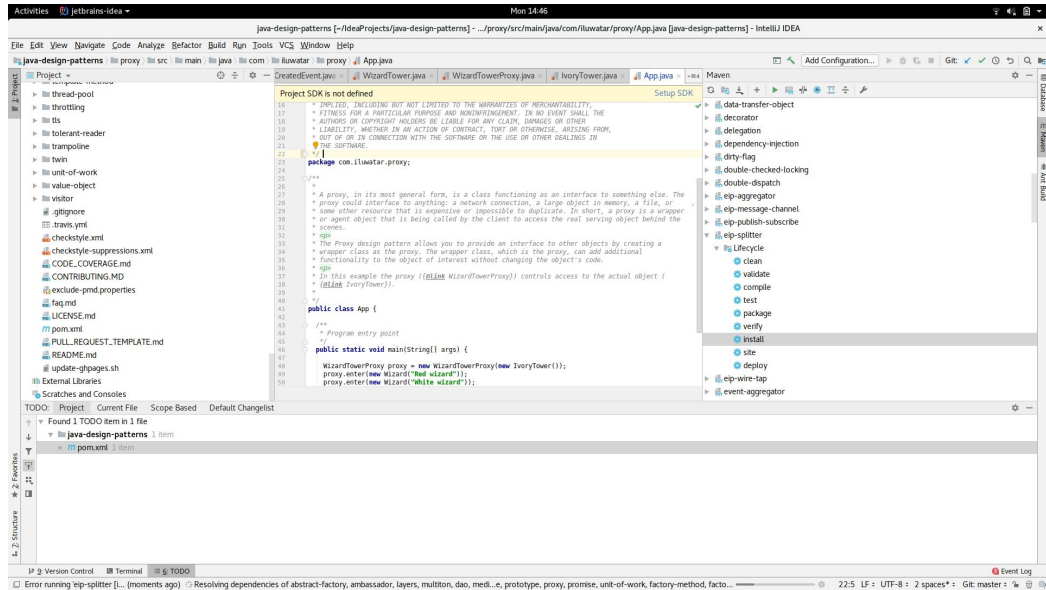
— John Barcellos and David Thien —

---

---

# Motivation

- IDEs have a lot of useful features
- Hard to keep track of them
- Tutorials usually present them all at once



# Solution

- We propose an Intelligent Tutorial system for IDEs that automatically detects when users could make use of a new feature they are not currently using and starts a tutorial
- If a user renames several variables all at once IDE-IT would pop up a tutorial for refactoring
- Tutorials are launched by triggers with heuristics to determine when a user wants a tutorial

# Approach and Challenges

- Add as a plug-in to the IntelliJ IDEA IDE
- IntelliJ is a large standard IDE with a plugin marketplace
- Challenge to our design is being able to accurately detect triggers to start a tutorial
- Not overwhelming the development process with tutorials
- Initially will do a tutorial for the most-used features of the IntelliJ IDE
- Long-term: could provide a library for other people to use
- Creating Intelligent Tutorials could be as easy as making a traditional one



# REALTIME PROTO

Web prototype based on existing code

by  
Annabelle Zha  
Ethan Cui

# Motivation

## Problem

End users often:

- code & design at the same time.
- want to quickly preview possible designs.
- avoid unnecessary code.

## Significance

- focus on design aspect.
- minimize coding workload.
- reduce development time.



# Approach

## A Chrome Extension

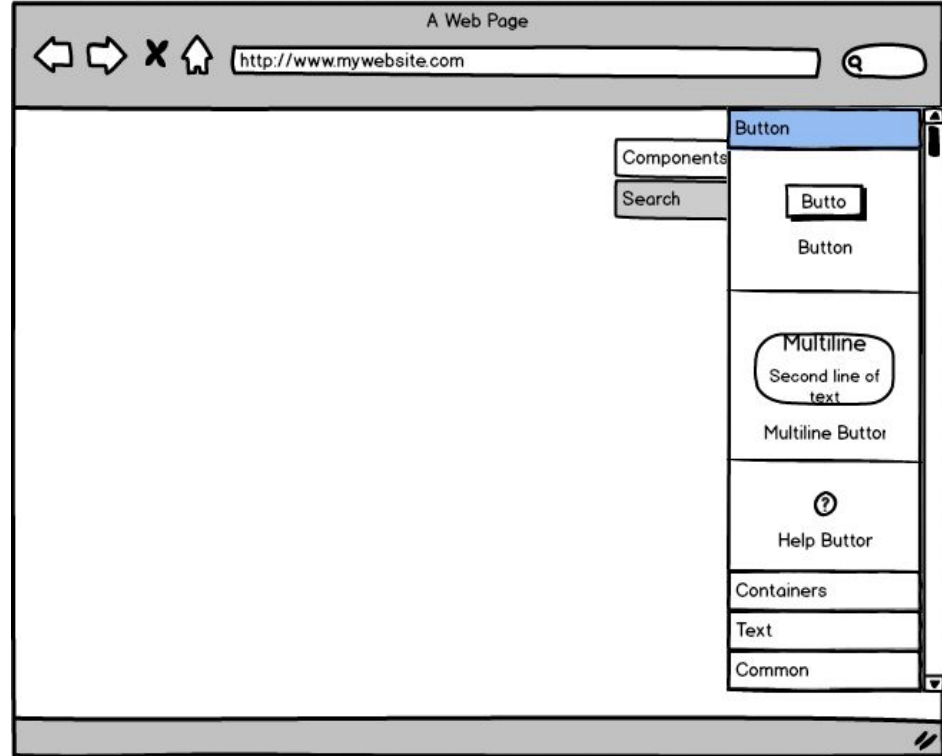
- drag components for previewing.
- style customization.

## Previous Approaches

Previous Approaches	Our Advantage
bootstrap component	code free
web paint extension	higher fidelity
web prototype builder	real-time prototyping

## Limitation:

purely visual, no code generation.



# Challenges & Risks

## Key Challenges

- how to use web page as a canvas without disabling its functionalities.  
eg. hyperlink, scrollability.
- how to allow users to customize the components.  
eg. size, color, font, shadow.

## Risk Mitigation

- earlier stage: prioritize features
- later stage: add components on web page screenshot.



# Q Review

for Slack

# What:

A Slack integrated tool that allows users to complete code reviews within Slack and publish changes to the assigned PR.

# Why:

Reviews command a lot of developer time in the software development workflow

Q Review allows developers to **review on the fly**

Reviews are poorly split amongst devs and communication can be slow

Q Review allows **group messages** to form around a review

Reviews and PRs often live on their own platform, away from the main discussion

Q Review means the reviews **live amongst the discussion**

# Implementation:

Q Review needs to plug into the common PR tools

Github, Jira, etc.

When the reviewer(s) are alerted, they can then interact with the review through text commands

**/qreview comment line 142** “Should you check for an out of bounds here?”

**/qreview request edits**

Finally, approve PRs without skipping a beat!

**/qreview ship it!**



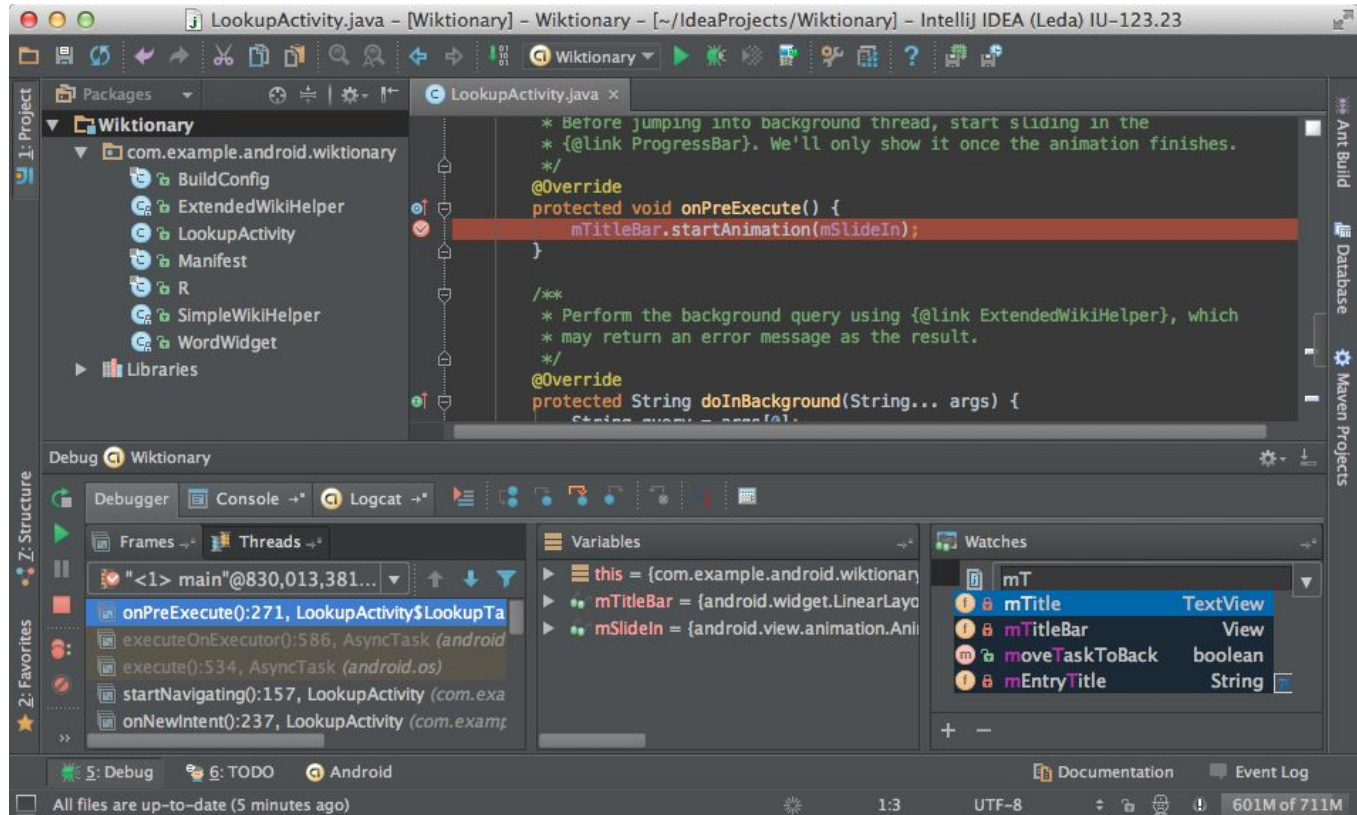
# FootPrint

The Data Tracker





# What is wrong in your life?



# FootPrint

Allows developers to access and explore what happens during debugging, including dynamic changes in data structures, variables, and functions called.

# Limitation & Potential Problems

- **Footprint is text-based**

- Not a tool for understanding the layout or design of the data structures
- Difficult to display large data structures, especially the graphs

- **Feature Creep**

- Lots of useful information to track, adding features may delay the schedule



# Java SmartMerger

By: Kamden Chew & Robert Kolmos



# White Space

**A:**

```
void foo() {  
    System.out.println("foo");  
}  
void bar() {  
    System.out.println("bar");  
}
```

**B:**

```
void foo() {  
    System.out.println("foo");  
}  
  
void bar() {  
    System.out.println("bar");  
}
```

**Tool Suggests:**

```
void foo() {  
    System.out.println("foo");  
}  
  
void bar() {  
    System.out.println("bar");  
}
```

# Refactoring Variable/Method Names

<p><b>A:</b></p> <pre>void foo() {     System.out.println("foo"); }  void bar() {     foo();     System.out.println("bar"); }</pre>	<p><b>B:</b></p> <pre>void baz() {     System.out.println("foo"); }  void bar() {     baz();     System.out.println("bar"); }</pre>	<p>User accepts foo-&gt;baz header change.</p> <p><b>Solution:</b></p> <pre>void baz() {     System.out.println("foo"); }  void bar() {     baz();     System.out.println("bar"); }</pre>
---	---	---

# Unused Imports

**A:**

```
import java.util.ArrayList;
void foo() {
    new ArrayList<String>();
    System.out.println("foo");
}

void bar() {
    System.out.println("bar");
}
```

**B:**

```
void foo() {
    System.out.println("foo");
}

void bar() {
    System.out.println("bar");
}
```

User accepts deletion of the first line of method foo().

**Solution:**

```
void foo() {
    System.out.println("foo");
}

void bar() {
    System.out.println("bar");
}
```

# TyrantVC

By: Carson Wilk and Volodymyr Loyko

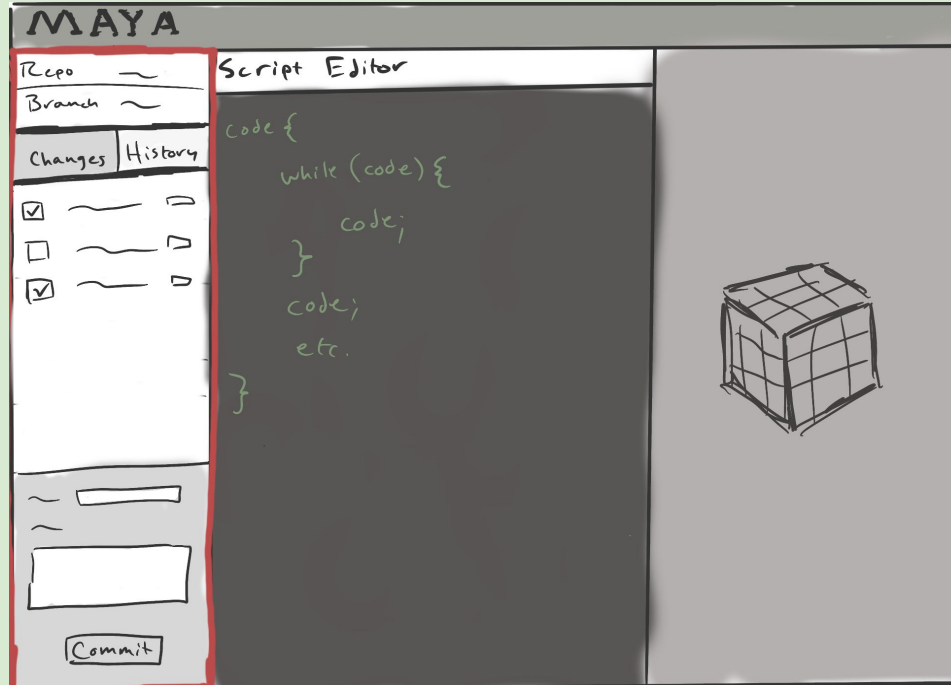


# Problem

- Maya lacks a version control system



# Approach



# Discoverability of Existing IDE Tools

...

Alyssa Ricketts and Rachel Zigman

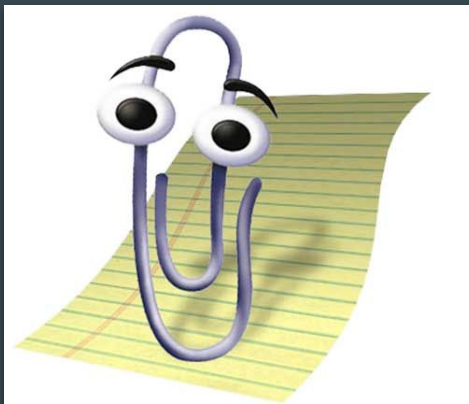
# Motivation

- Only a small subset of IDE tools typically get used
- Developers are often unaware of tools that are offered
- Difficult to find configuration menus
- Tools should be easily customizable



# Approach

- User interface identifies developers current use
- Produces suggestions and immediate links



# Challenges

- Interpreting what the user is trying to do and identifying the best tool
- Rating scheme to gain immediate feedback on new tool

Package Explorer

- cse332.datastructures.worklists
- cse332.exceptions
- cse332.interfaces.misc
- cse332.interfaces.trie
- cse332.interfaces.worklists
- cse332.misc
- cse332.sorts
- cse332.types
- datastructures.dictionaries
- datastructures.worklists
- experiment.BSTvsAVL
- experiment.ChainingHashTable
- experiment.GeneralPurposeDictionary
- experiment.HashFunctions
- p2.clients
- p2.sorts
- p2.wordcorrector
- p2.wordsuggestor
- p2.writeup
- tests.gitlab.ckpt1
  - CircularArrayComparatorTests.java
  - Ckpt1 Tests.java
  - MoveToFrontListTests.java
  - NGramToNextChoicesMapTests.java
- tests.gitlab.ckpt2
  - AVLTreeTests.java
  - CircularArrayHashCodeTests.java
  - Ckpt2Tests.java
  - HashTableTests.java
  - HeapSortTests.java
  - QuickSortTests.java
  - TopKSortTests.java
- JRE System Library [Java SE 8 [1.8.0\_111]

```

1 package datastructures.dictionaries;
2
3 import cse332.datastructures.trees.BinarySearchTree;
4
5 /**
6  * Interface for an AVL tree: a self-balancing Binary Search Tree
7  * where the difference between heights of left and right subtrees
8  * cannot be more than one for all nodes.
9  */
10
11 // @SuppressWarnings("hiding")
12 public class AVLTree<K extends Comparable<K>, V> extends BinarySearchTree<K, V>
13     private int size;
14     private int testVal;
15
16     public AVLTree() {
17         super();
18         this.size = 0;
19     }
20
21     /**
22      * Inner class to represent a node in the tree.
23      */
24     public class AVLNode extends BSTNode {
25         public int height; // Height field
26     }

```

Task List

Find

All Activate...

### Tool Suggestions:

- Enable compiler warning for hidden fields
- Enable spell checker
- Enable auto-complete
- Enable content assist

Problems @ Javadoc Declaration Console Search Git Staging

```

<terminated> Ckpt2Tests [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/
AVLTree
    testTreeWith5Items      ✗ Failed Test(s)
    testHugeTree            ✓ Passed
    checkStructure          ✗ Failed
    HashTable                ✓ Passed Test(s)
    testHugeHashTable       ✓ Passed
    equality                 ✓ Passed Test(s)
    ineq1                    ✓ Passed
    ineq2                    ✓ Passed
    ineq3                    ✓ Passed

```

Outline Visualizer Debug Saros

- datastructures.dictionaries
  - AVLTree<K extends Comparable<K>, V>
    - size : int
    - testVal : int
    - AVLTree()
    - AVLNode
      - height : int
      - AVLNode(K, V)
      - insert(K, V) : V
      - insertHelper(AVLNode, K, V) : AVLNode
      - updateHeight(AVLNode) : void
      - getBalance(AVLNode) : int

# Flint

A customizable style linter for Java



Elliott de Bruin, Ofek Inbar

# Motivation

- Developing and maintaining a clean and readable codebase is difficult for teams with many contributors
- Style guides can help keep the codebase readable (and uniform)
- Linters can help enforce style rules, either as command line tools or ide plugins/extensions
- CheckStyle is a good example of a Java linter:
  - Pros:
    - Comes with many pre-written style rules
    - Supports customizing which rules to check for a project
    - Supports writing custom rules
  - Cons:
    - Can only be run as either an Ant task or as a CLI (inferior to the red squiggly lines most IDEs display for errors on every keystroke or file save)
    - Configuration files are written in XML, a Java programmer shouldn't need to learn XML to use a Java linter



# Approach

- Build a linter that supports customizing which style rules to apply and also supports writing custom style rules
  - Offer the linter as an IDE plugin/extension that can run on every file-save/keystroke and very quickly informs the user of code locations where the rules are not met
  - Rule configuration/custom rule creation will be defined completely in Java code
- Possible implementation of a configuration system:

```
package demo;

import AbstractConfiguration;
import CheckRunner;

@Configuration
public class CustomConfiguration extends AbstractConfiguration {
    @Override
    runChecks(CheckRunner runner) {
        TabsNotSpacesRule.runCheck(runner);
        NoTrailingWhitespaceRule.runCheck(runner);
        LineLimitRule.runCheck(runner, 100);
    }
}
```

# PhotoGram



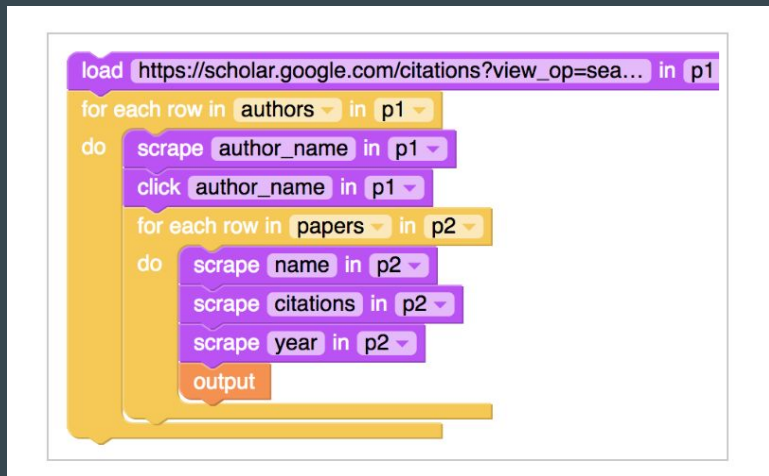
A Programming Language For Photographers

# User Group

- Photographers
- Photography Lovers

# Motivation

- Problem: Too much repetitive work for photographers that can be programmed
- Solution: Design A programming Language with Block Based GUI for Batch Image Processing.



```
load https://scholar.google.com/citations?view_op=sea... in p1
for each row in authors in p1
do
  scrape author_name in p1
  click author_name in p1
  for each row in papers in p2
  do
    scrape name in p2
    scrape citations in p2
    scrape year in p2
  output
```

One example of a block-based GUI is “Helena”(<http://helena-lang.org/>)

# Risks and Challenges

- Making it trivial enough for non-programmer users
  - Image Processing algorithms is to the development but is not a topic that most developers are familiar with
  - Make it smart to be able to develop each photo differently
- 
- Challenges depends on the scale of this project.
  - May need focusing on integration with existing algorithms.