A *Journey* through Searching Similar Code

Miryung Kim
Professor and Vice Chair of Graduate Studies at UCLA
Amazon Scholar, Amazon Web Services

UCLA

aws

Outline: A *Journey* through Searching Similar Code

What motivated us?
What were early attempts?
How serious is this problem?
How can we automate?
How can we examine variations at scale?
How to search with a human in the loop?

What ideas have motivated searching similar code?

A Study of Copy and Paste Programming Practices [ISESE 2004]

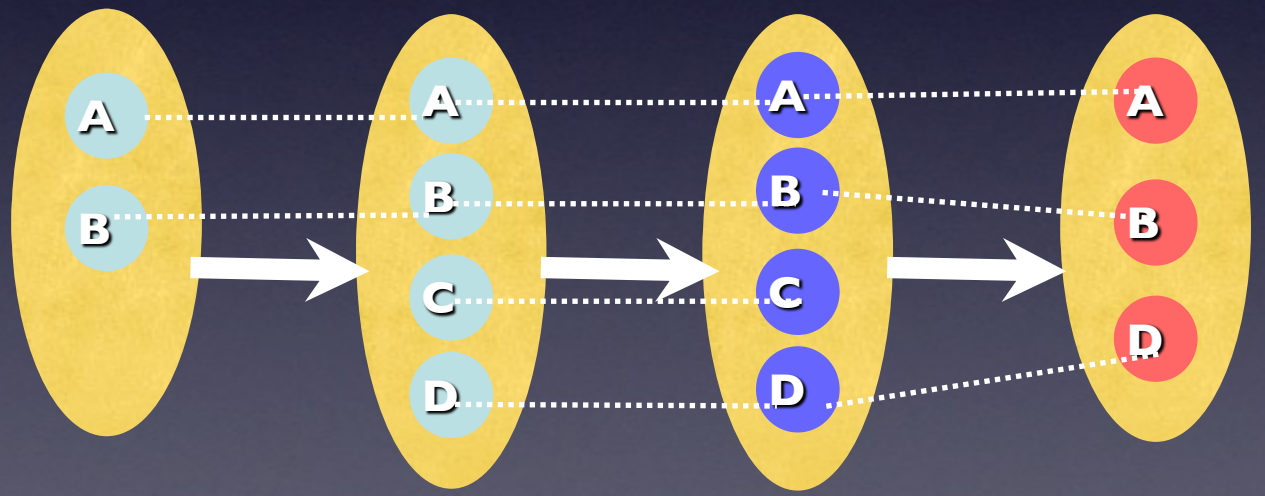
- To understand programmers' copy and paste coding behavior, we ***built an Eclipse plug-in that records edits*** and ***replays the captured edits*** **at IBM**
- Programmers often ***create and manage*** code clones with clear intent

An Empirical Study of Code Clone Genealogies [FSE 2015]

- We developed an approach that ***automatically reconstructs*** the history of code clones ***from a source code repository***
- We studied clone evolution in several Java open source projects.

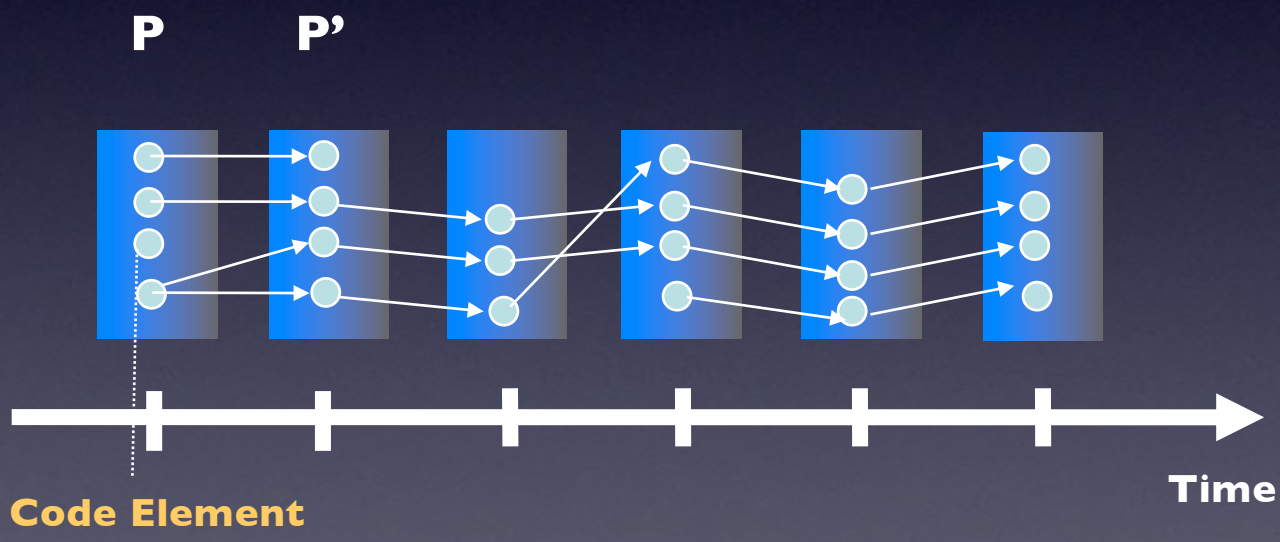
Clone Genealogy

Clone genealogy is a representation that captures clone change patterns over a sequence of program versions

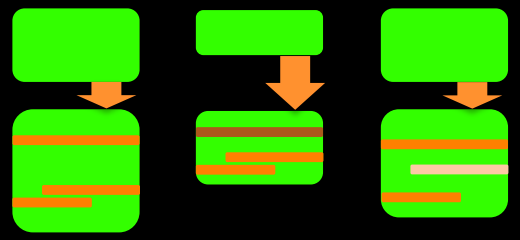


Dagstuhl: Multiversion Program Analysis in 2005

Mining Software Repositories



Systematic Changes (similar updates to similar code)



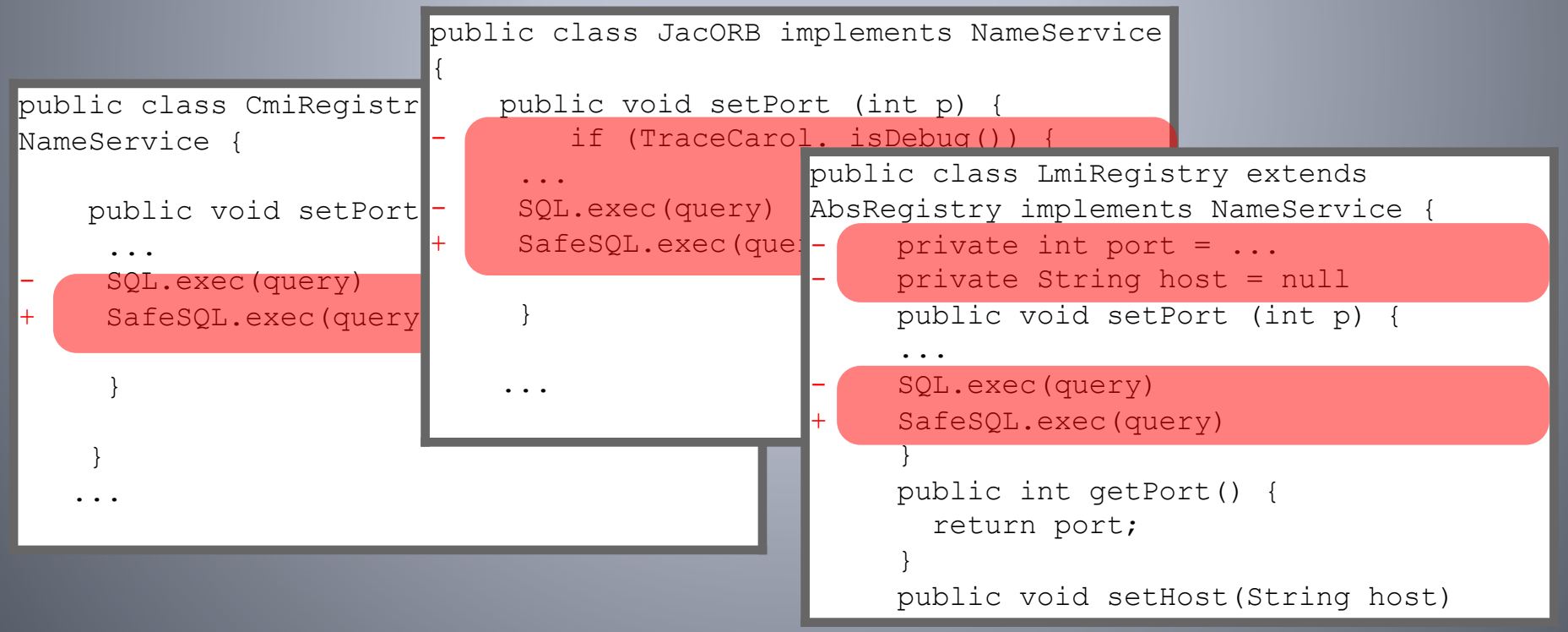
Consistent updates to clones

Managing multiple products, forked projects and versions

API evolution and ripple effects on client applications

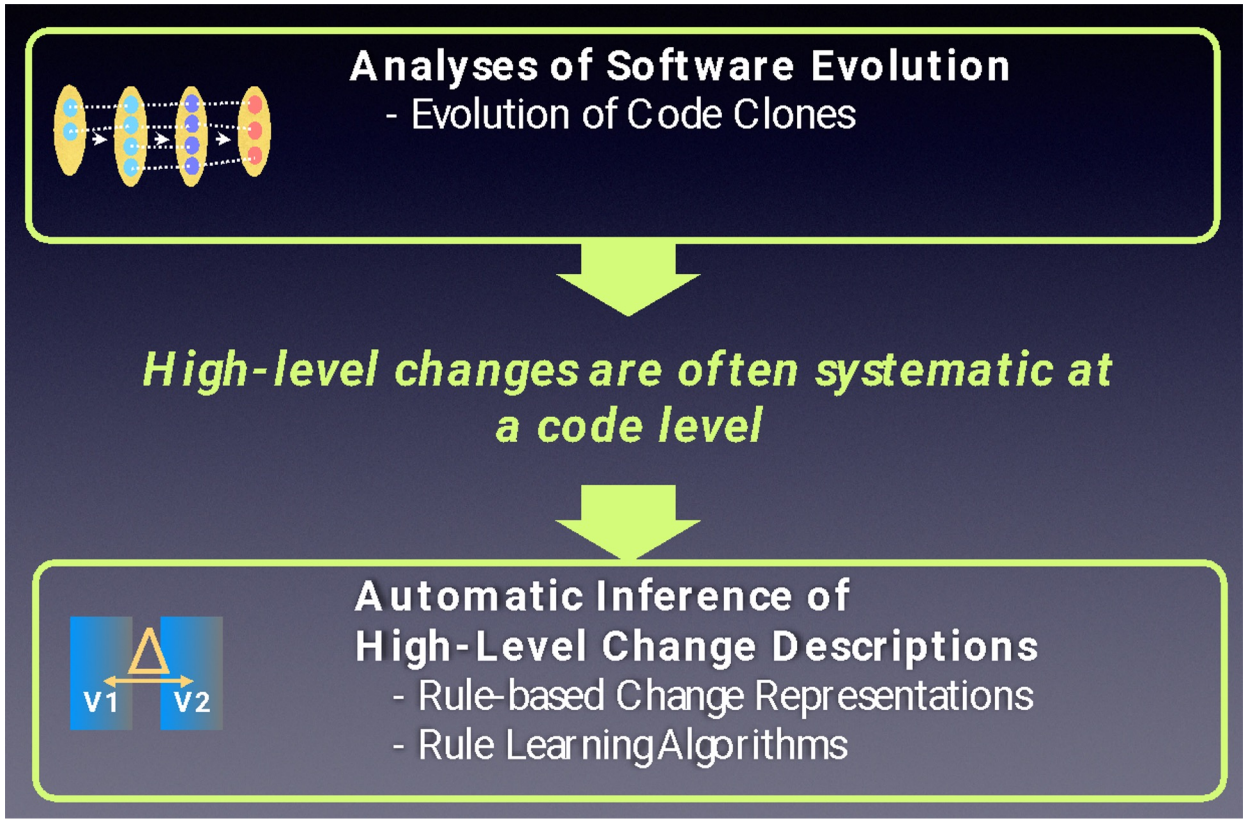
Refactoring

What were early attempts to abstract *systematic changes*?

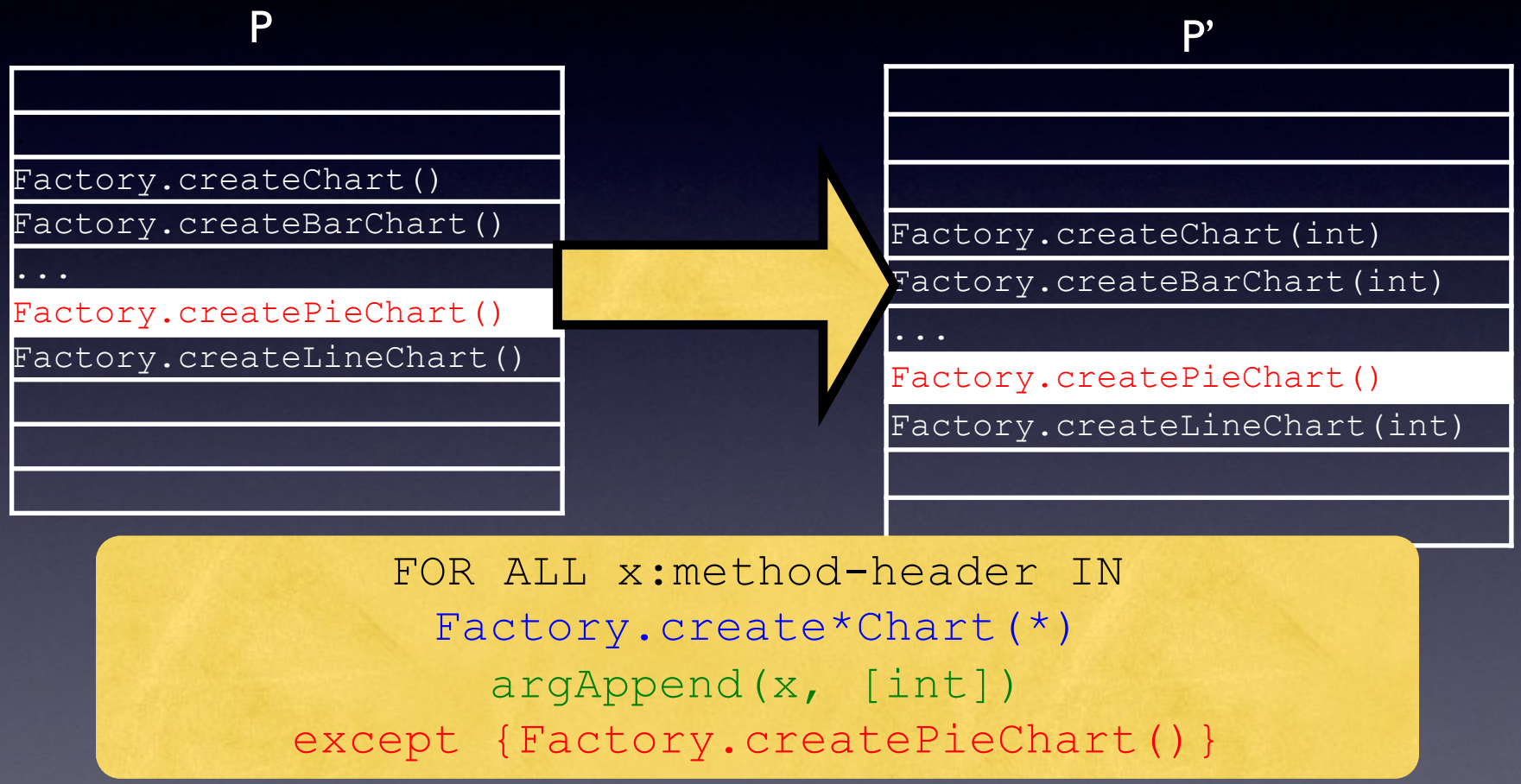


Miryung's PhD @ U of Washington
Automated Change Rule Inference

David Notkin
(1 Jan 1955 – 22 Apr 2013)



API Change Rule Inference [ICSE 2007]



LSDiff Rule Inference [ICSE 2009]

- "Replace all calls to `SQL.exec` with `SafeSQL.exec`"

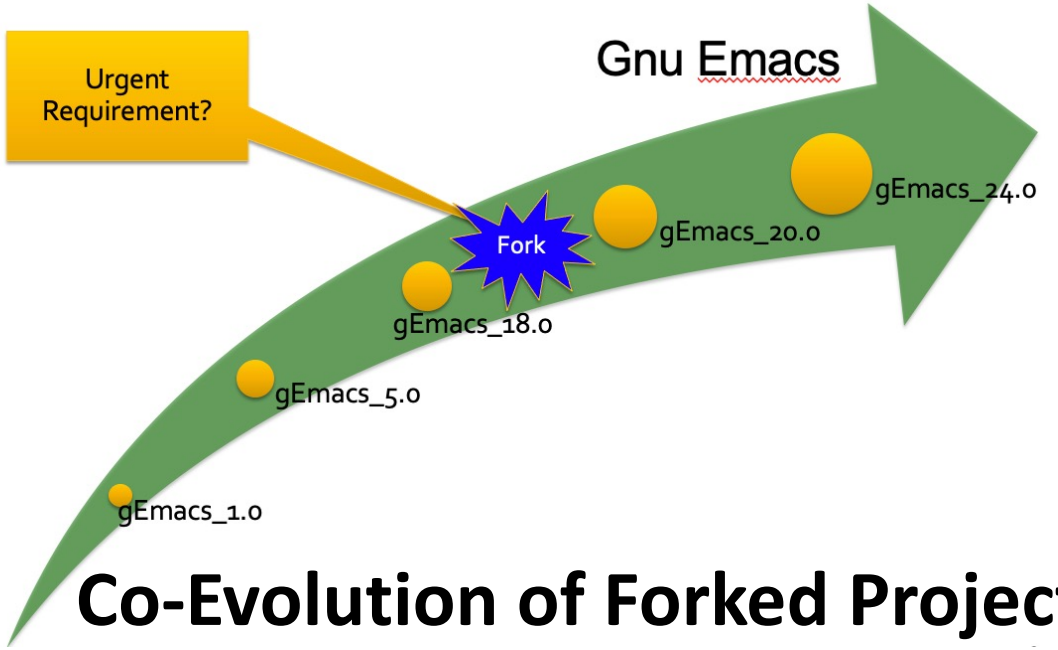
```
deleted_calls(m,"SQL.exec")=>
added_calls(m,"SafeSQL.exec")
```

- "All `setHost` methods in `Service`'s subclasses in the old version deleted calls to `SQL.exec` except the `setHost` method in the `NameSvc` class.

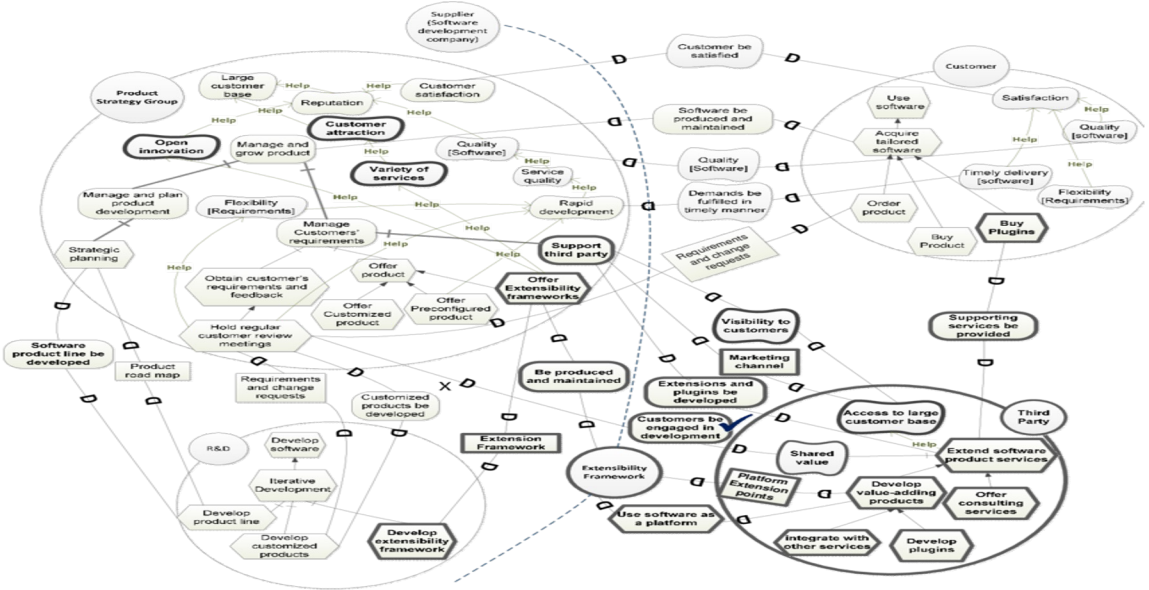
```
past_subtype("Service", t) ∧ past_method(m,
"setHost", t)
⇒ deleted calls(m, "SQL.exec")
except t="NameSvc
```

How serious is this problem of searching similar code?

Baishakhi's PhD @ UT Austin
Cross-system co-evolution



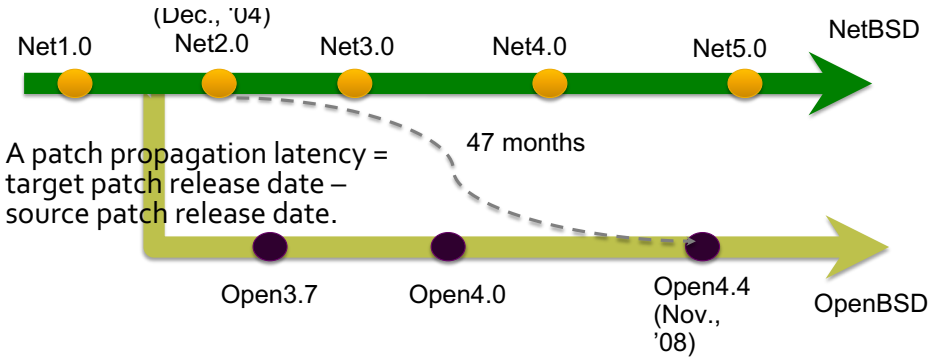
Co-Evolution of Forked Projects

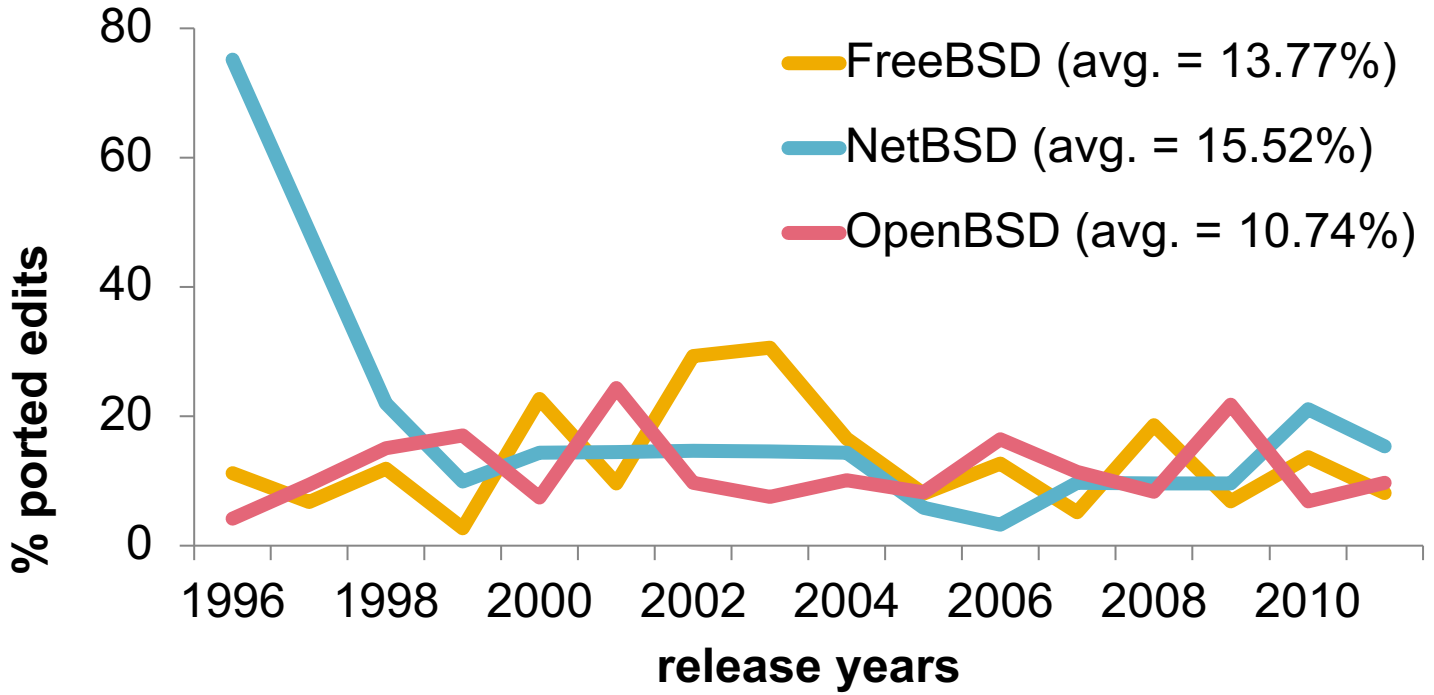


Software Eco-Systems

API Stability and Adoption in Android Ecosystem

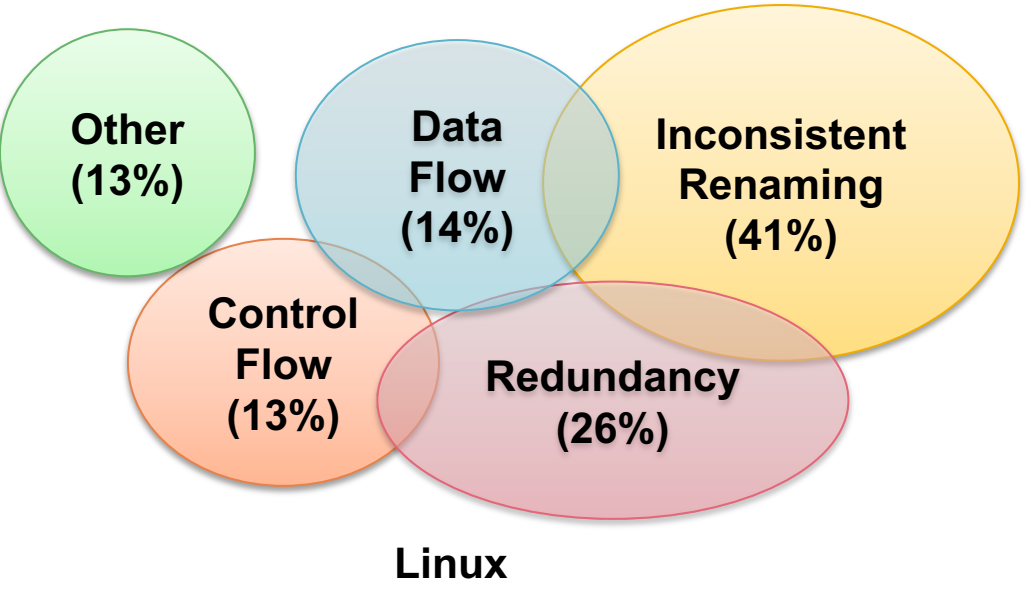
Repertoire: Cross-System Porting Analysis in Forked Projects [FSE 2012]

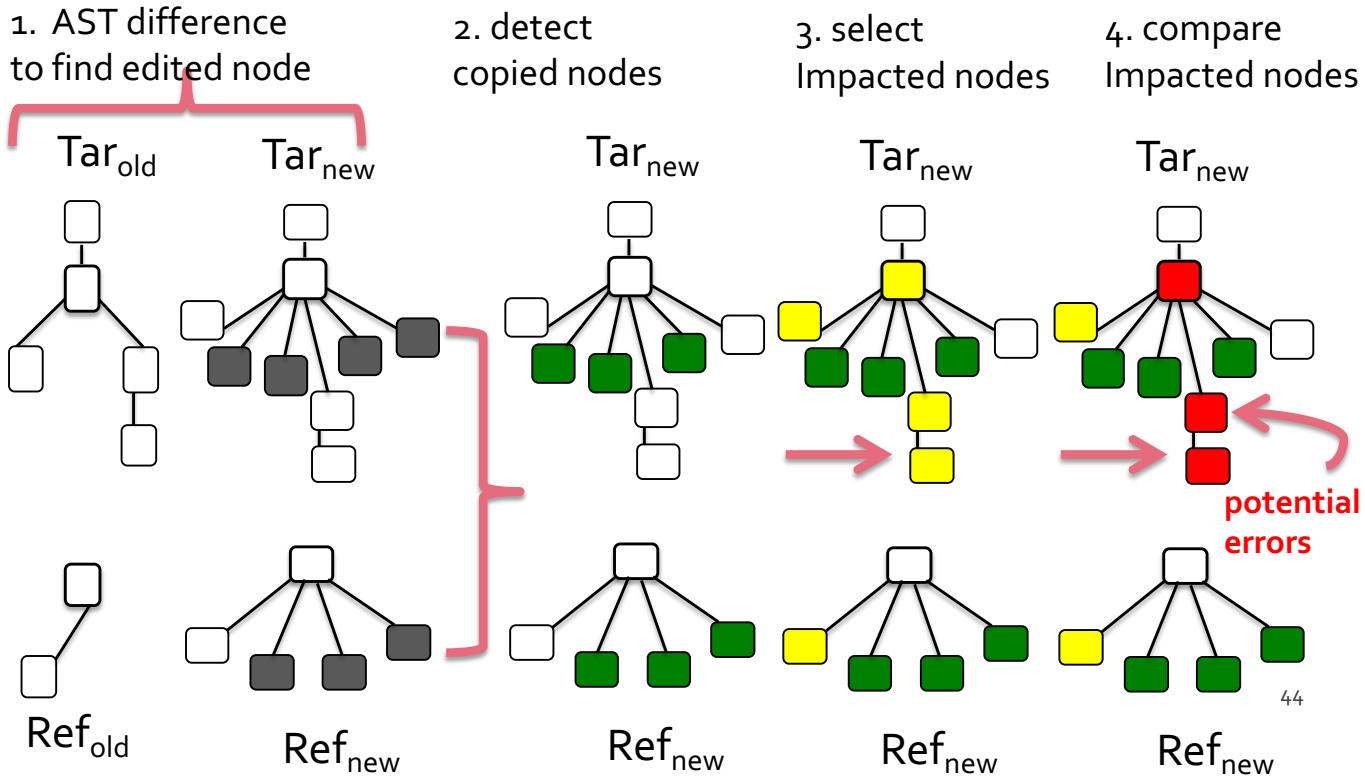




Porting consists of a significant portion of the BSD family evolution and a significant portion of active committers port changes

SPA: Detecting Semantic Inconsistencies in Ported Code in Linux [ASE 2013]



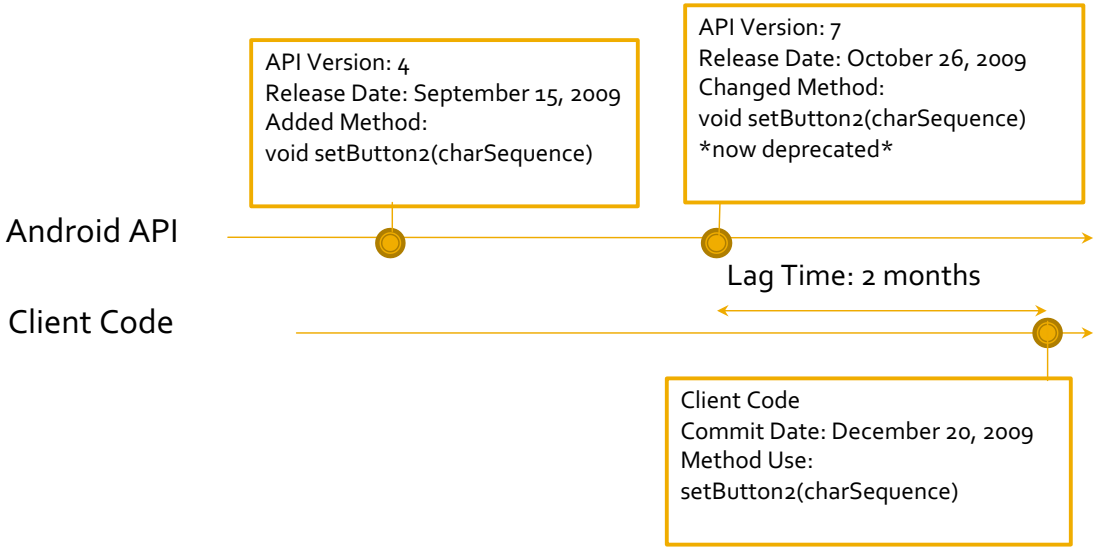


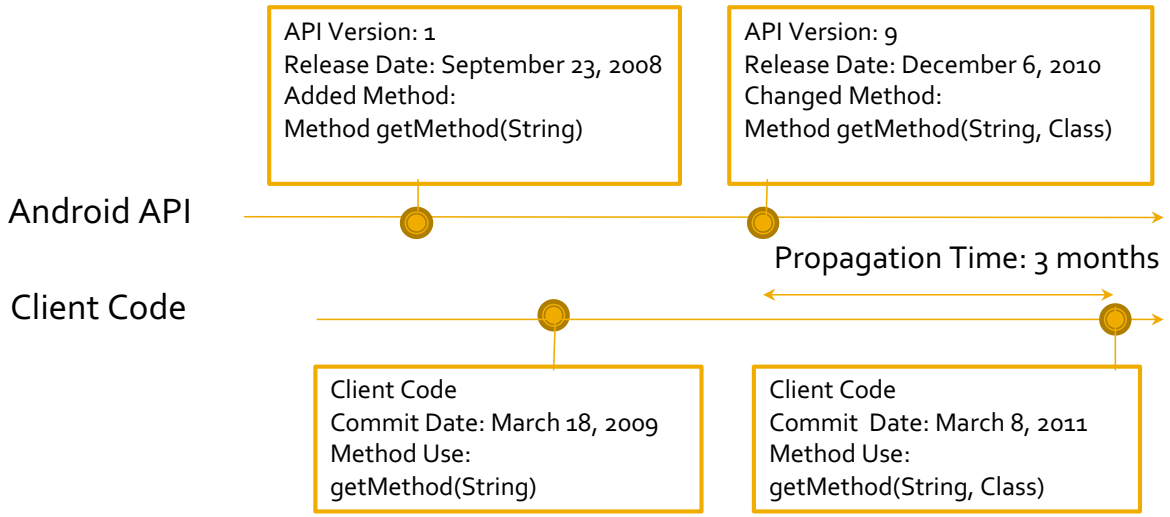
13% to 25% changes are reused in Linux and Microsoft projects

API Stability and Adoption in the Android Ecosystem [ICSME 2013]

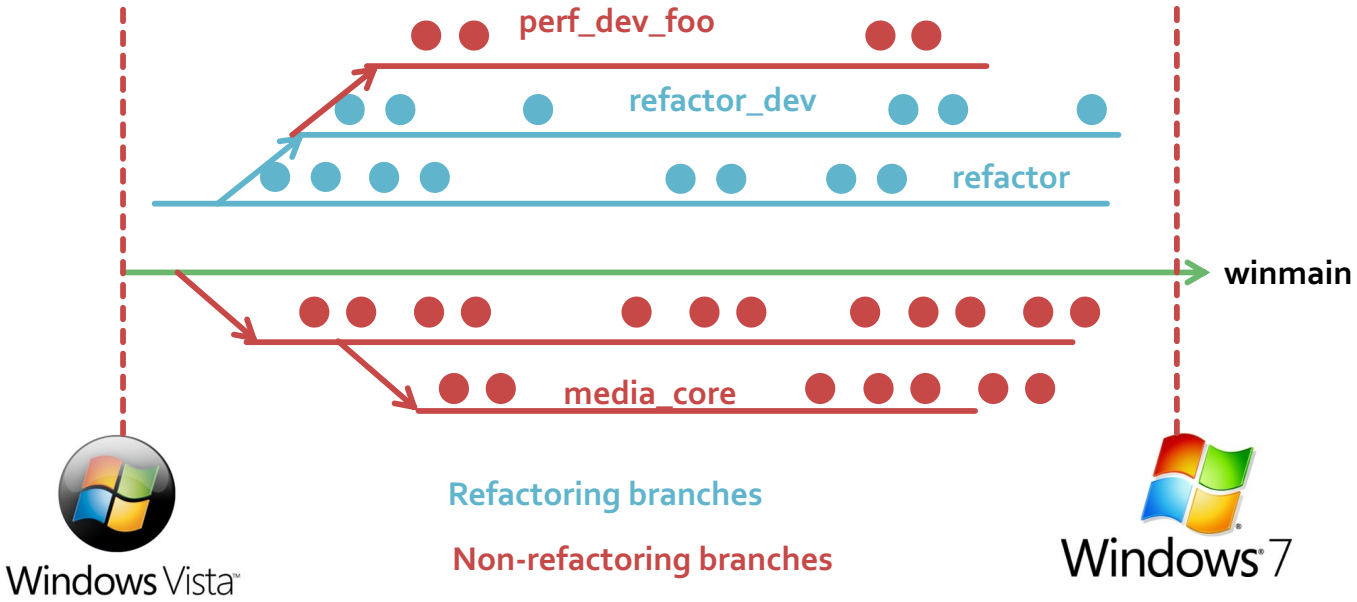
Most Influential Paper Award from ICSME 2013

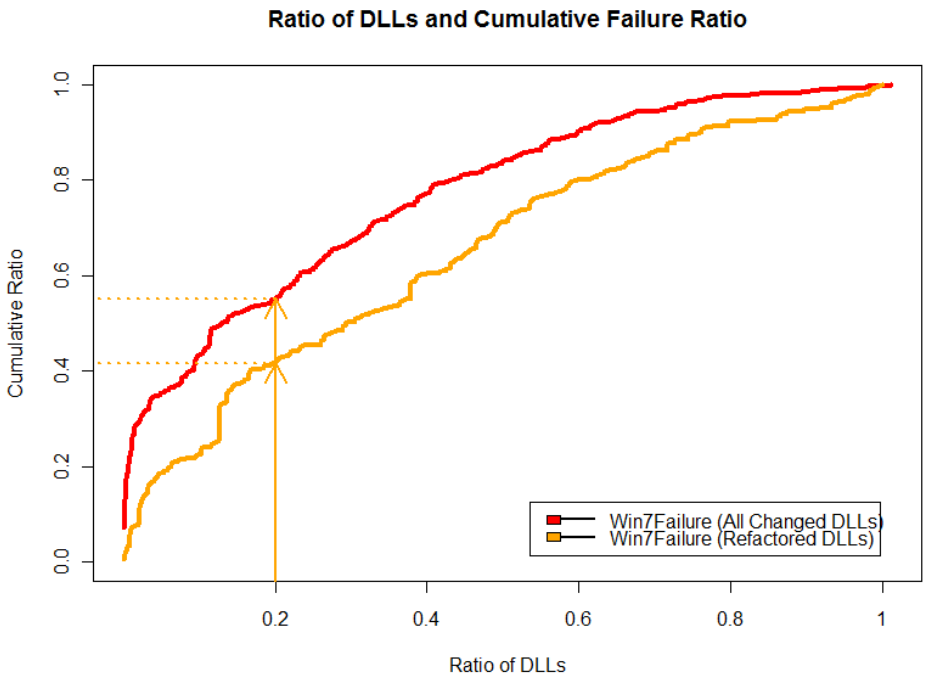
- Android is evolving fast at a rate of 115 API updates per month.
- 28% of API references in client apps are outdated with a median lagging time of 16 months.
- API usage adaptation code is **defect prone** than other code.





Microsoft: Quantifying Benefits of Windows Re-architecting [FSE '12, TSE '14]





Refactoring churn is less defect-prone than regular churn.

How can we *automate* systematic changes?

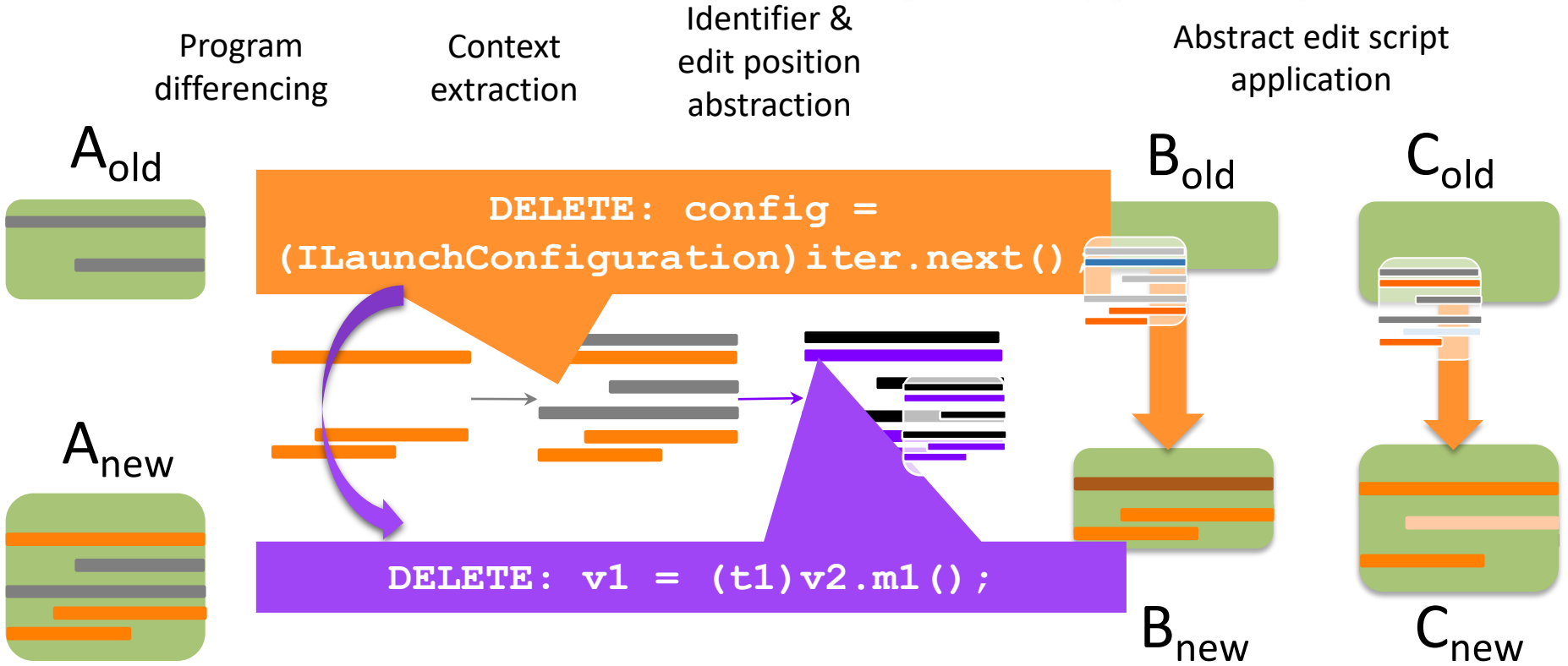
Na Meng's PhD @ UT Austin
Automating Systematic Changes

Systematic Editing:
Generating Program Transformations from an Example

Na Meng Miryung Kim Kathryn S. McKinley
The University of Texas at Austin
mengna152173@gmail.com, miryung@ece.utexas.edu, mckinley@cs.utexas.edu

LASE: Locating and Applying Systematic Edits by Learning from Examples

Na Meng* Miryung Kim* Kathryn S. McKinley*†
The University of Texas at Austin* Microsoft Research†
mengna09@cs.utexas.edu, miryung@ece.utexas.edu, mckinley@microsoft.com



Systematic Editing: Generating Program Transformations from an Example [PLDI 2011]

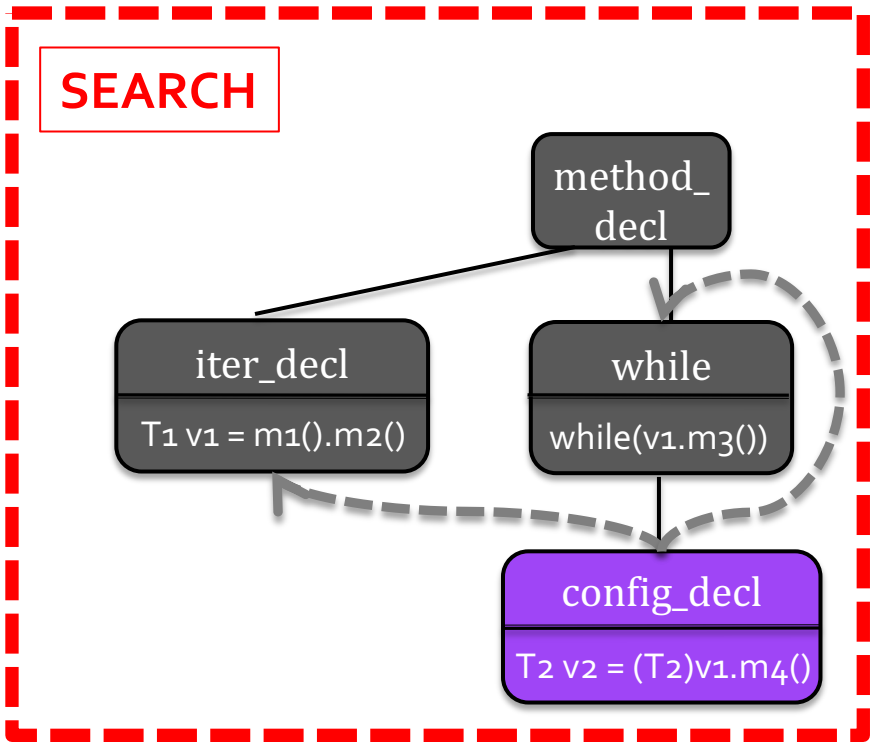
```
1. public ILaunchConfiguration[] getLaunchConfigurations
   (ILaunchConfigurationType type) throws CoreException {
2.    Iterator iter = getAllLaunchConfigurations().iterator();
3.    List configs = new ArrayList();
4.  + ILaunchConfiguration config = null;
5.    while (iter.hasNext()) {
6.     - ILaunchConfiguration config =
                  (ILaunchConfiguration)iter.next();
7.     + config = (ILaunchConfiguration)iter.next();
8.     + if (!config.inValid()) {
9.     +   config.reset();
10.    + }
11.      if (config.getType().equals(type)) {
12.         configs.add(config);
13.      }
14.   }
15.   return (ILaunchConfiguration[])configs.toArray
                  (new ILaunchConfiguration[configs.size()]);
17.}
```

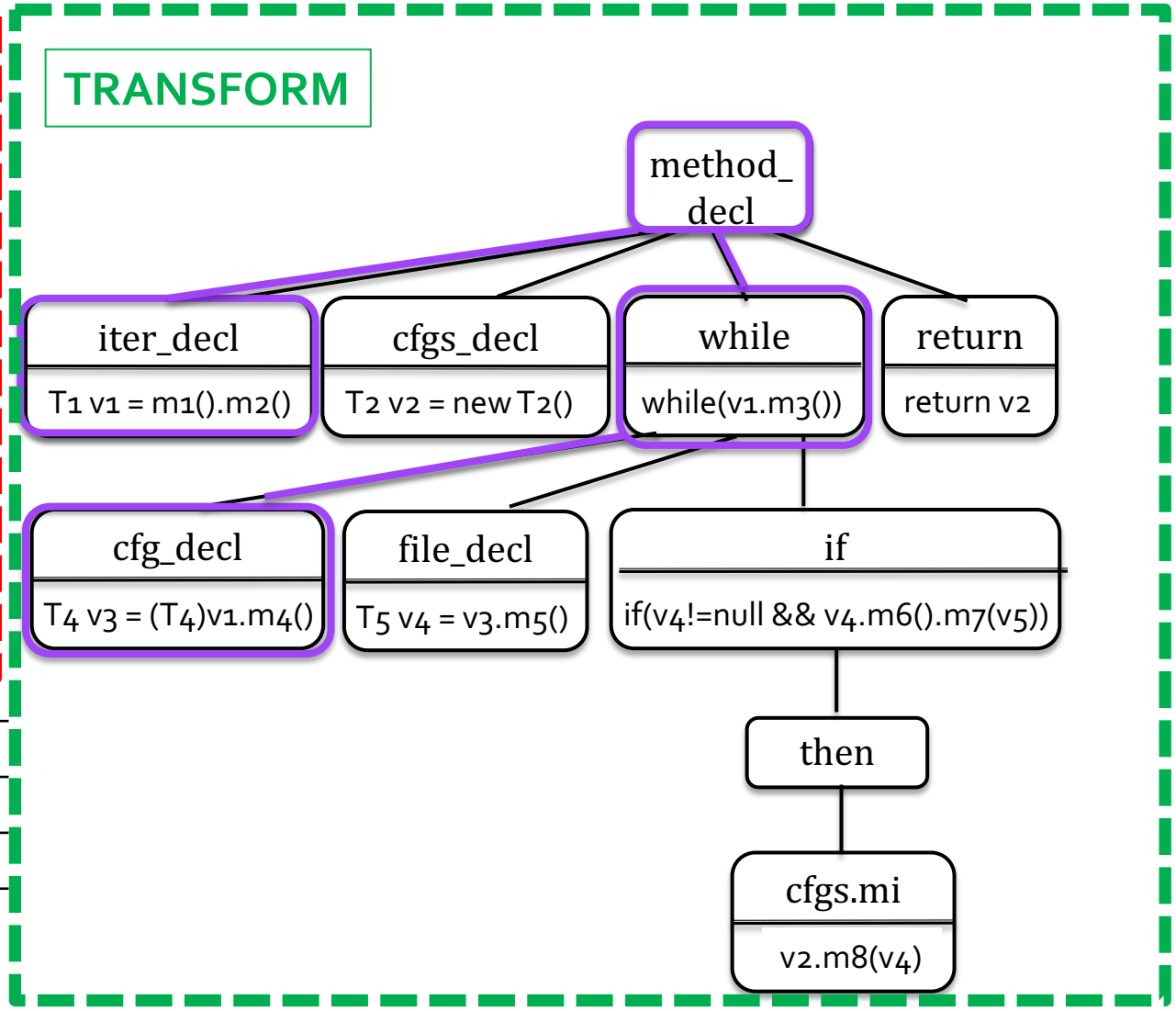
```
1. … … method_declaration(… …){
2.      T1 v1 = m1().m2();
3.      … …
4.      while(v1.m3()){
5.  UPDATE: T2 v2 = (T2)v1.m4();
6.      TO: T2 v2 = null;
7.  INSERT: v2 = (T2)v1.m4();
8.  INSERT: if(!v2.m5()){
9.              INSERT: v2.m6();
10.         }
11.         … …
12.     }
13.     … …
14. }
```

MOVE

Abstract edit script

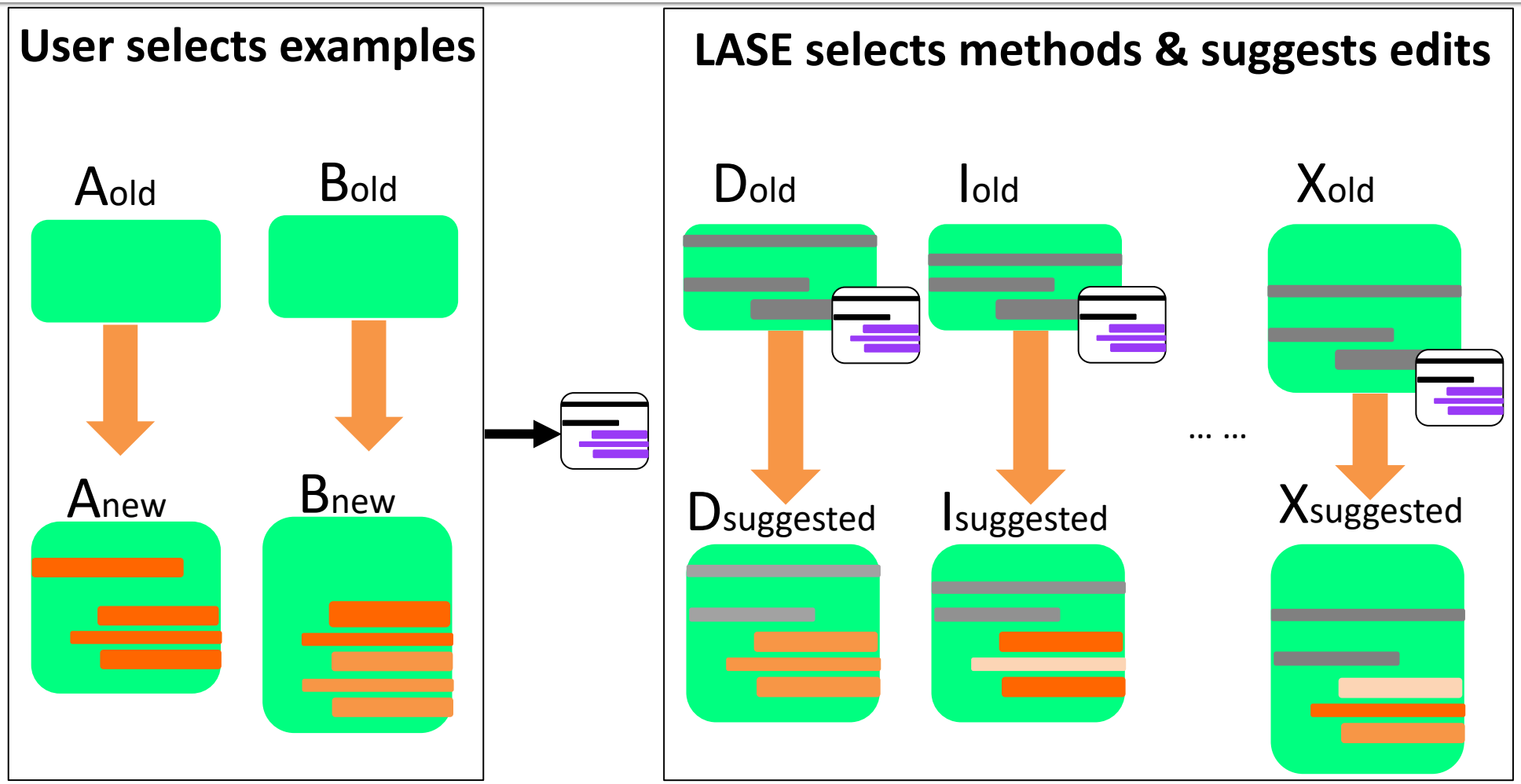
Abstract Edit Script





Variable Map	Method Map	Type Map
v1 <-> v1	m1 <-> m1	T1 <-> T1
v2 <-> v3	m2 <-> m2	T2 <-> T4
	m3 <-> m3	
	m4 <-> m4	

Lase: Locating and Applying Systematic Edits [ICSE 2013]

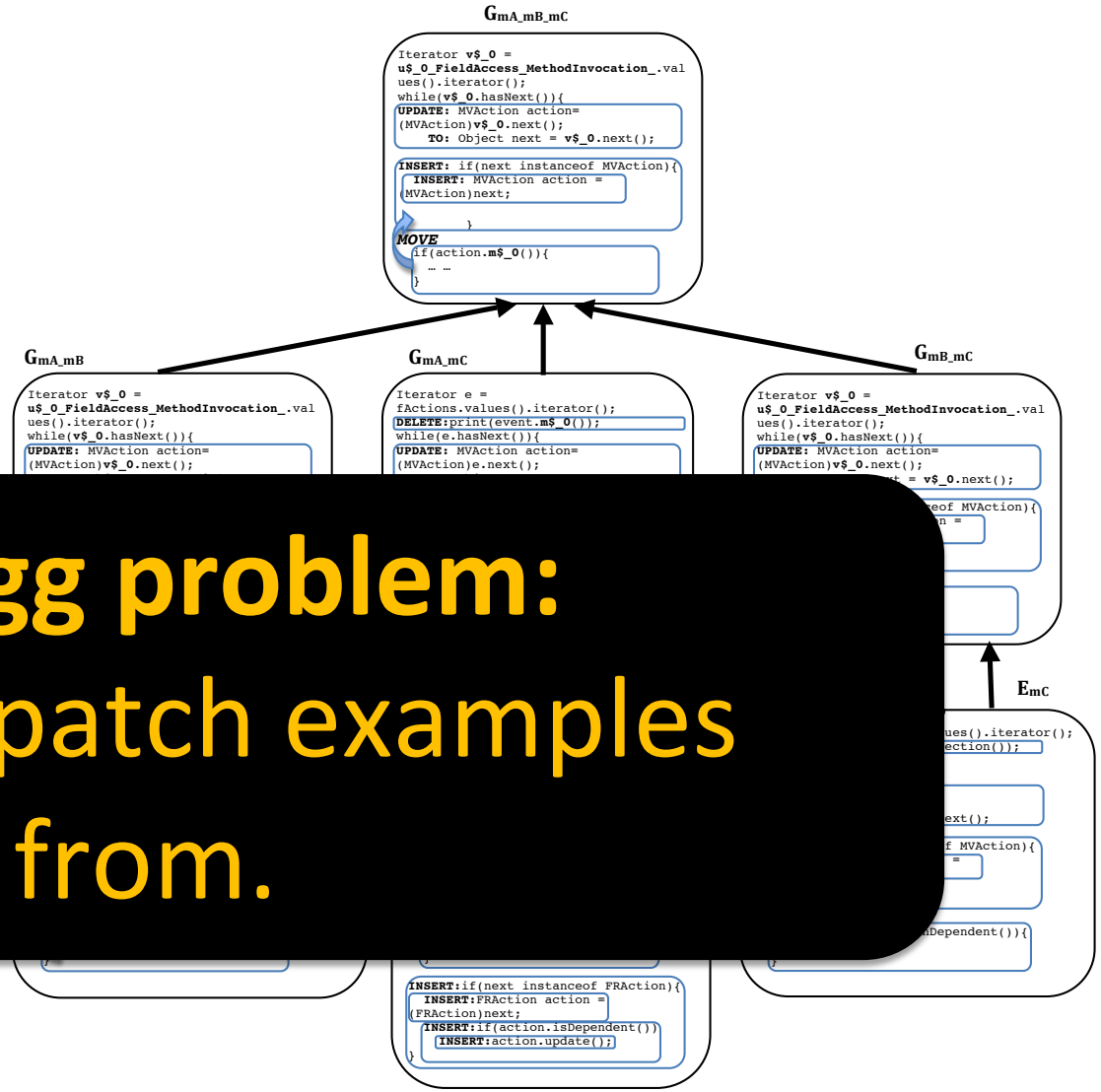


```
Comment[] getLeadingComments(ASTNode node){
-   if (this.leadingComments != null) {
+   if (this.leadingPts >= 0) {
-       int[] range = (int[]) this.leadingComments.get(node);
+       int[] range = null;
+       for (int i = 0; range == null && i <= this.leadingPtr; i++) {
+           if (this.leadingNodes[i] == node) range = this.leadingIndexes[i];
+       }
        if (range != null) {
            int length = range[1] - range[0] + 1;
            Comment[] leadComments = new Comment[length];
            System.arraycopy(this.comments, range[0], leadComments, 0,
```

```
Comment[] getTrailingComments(ASTNode node){
-   if (this.trailingComments != null) {
+   if (this.trailingPts >= 0) {
-       int[] range = (int[]) this.trailingComments.get(node);
+       int[] range = null;
+       for (int i = 0; range == null && i <= this.trailingPtr; i++) {
+
this.traili
+       }
        if
```

```
length);
```

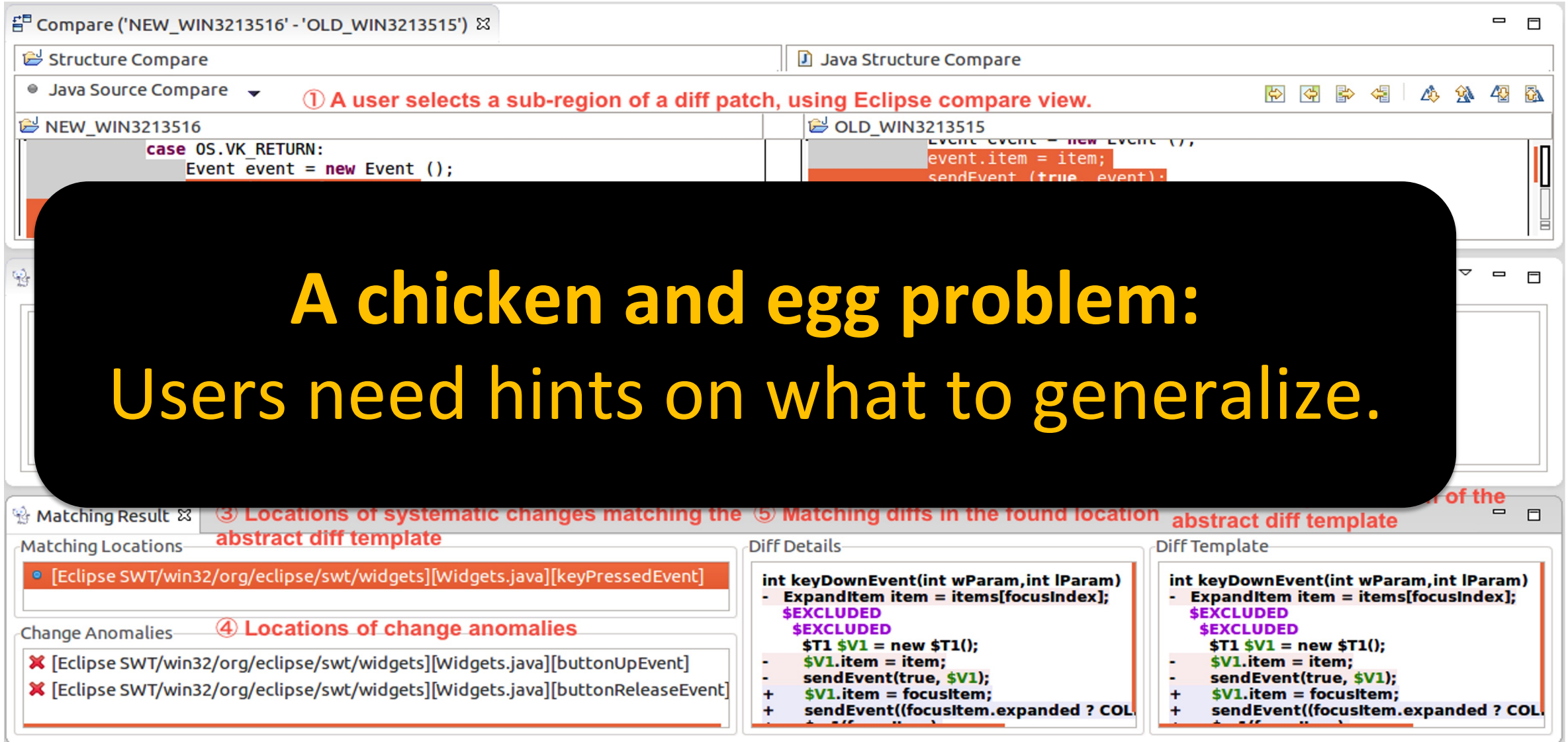
```
ublic int g
    int end
    if (this
    if (this
        int[
        int[
        for (int i = 0; range == null && i <= this.v$_1_; i++) {
            if (this.v$_2_[i] == node) range = this.v$_3_[i];
        }
        if (range[0] == -1 && range[1] == -1) {
            … …
        } else {
            … …
        }
    }
    return end - 1;
```



A chicken and egg problem:

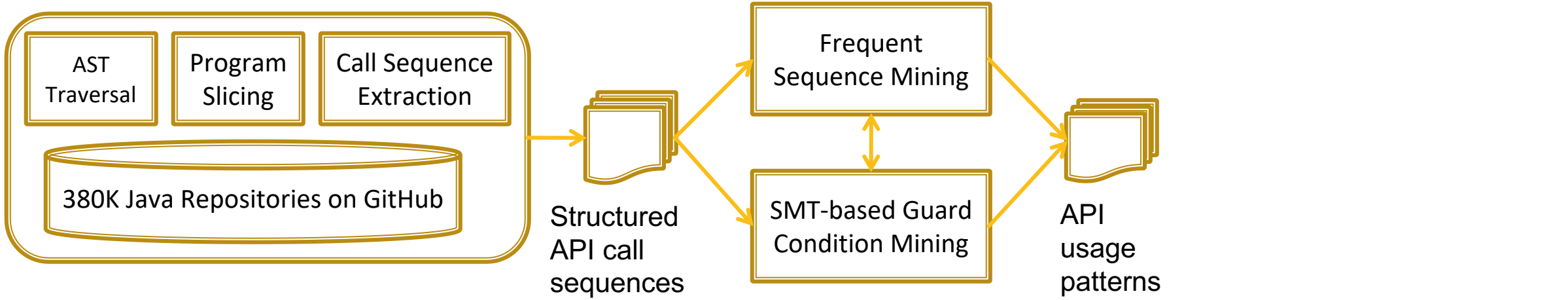
Users needs to find patch examples generalize from.

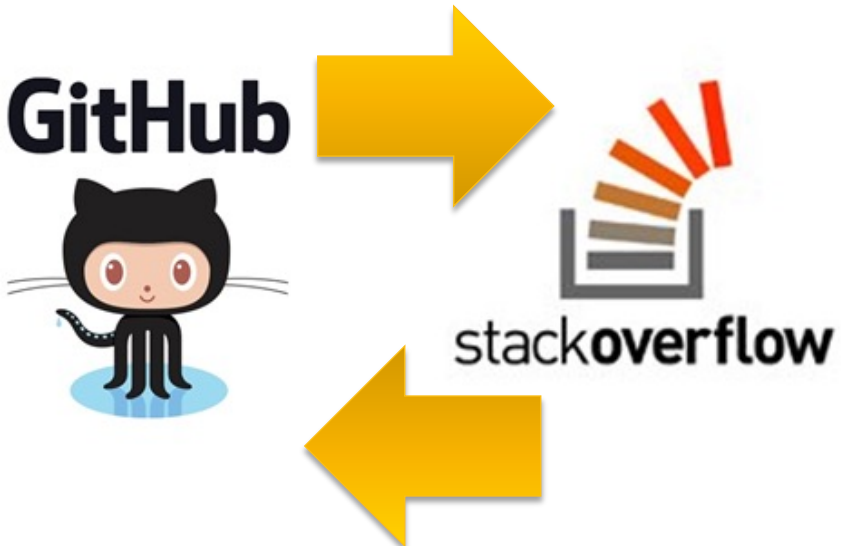
Critics: *Interactive* Code Search and Review [ICSE 2015]

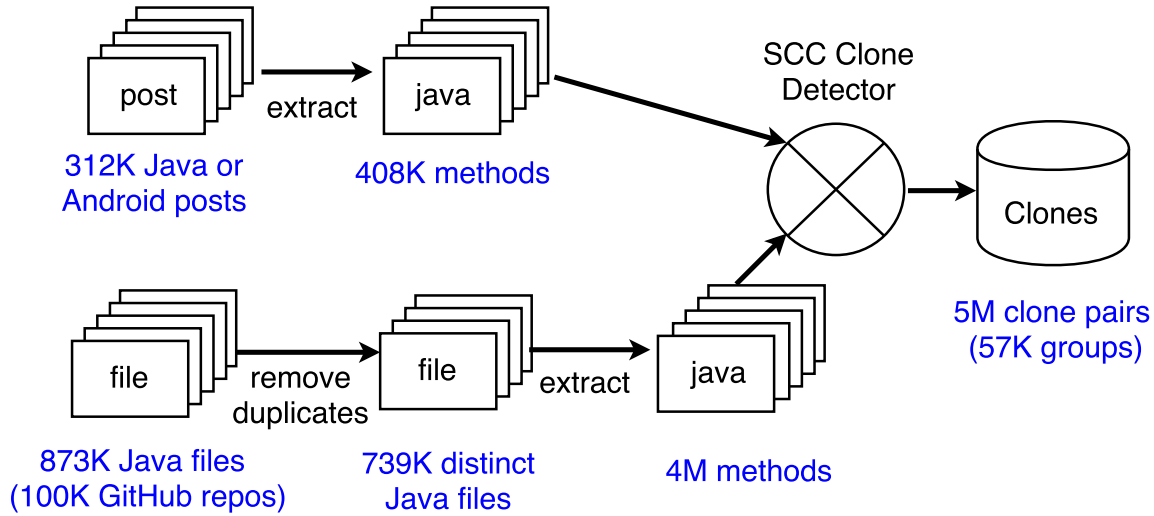


How can we mine and examine variations at scale?

Tianyi Zhang's PhD @ UCLA: Leveraging Commonalities and Variations at Scale



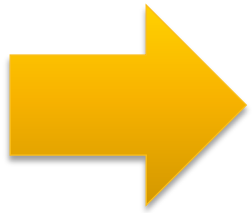


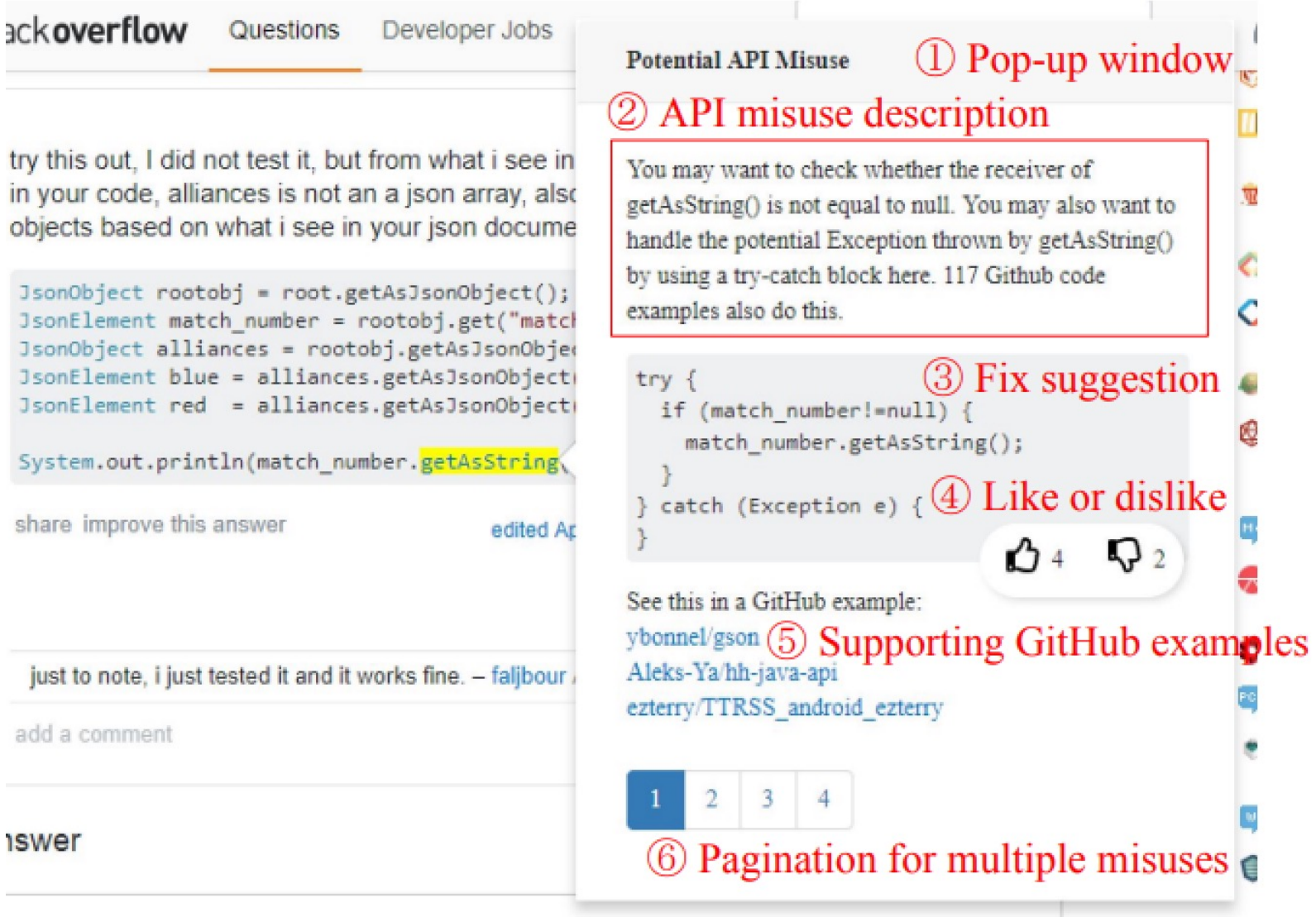


API Usage Mining from GitHub and API Misuse Detection in StackOverflow [ICSE '18]

GitHub

stackoverflow

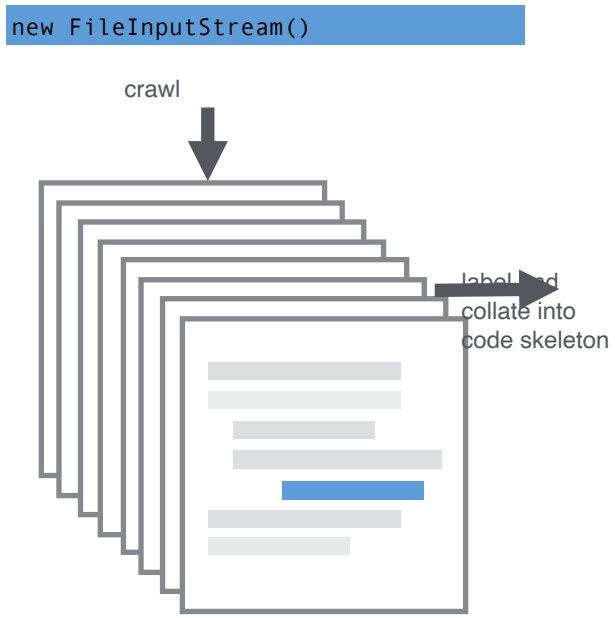




Analyzing and Supporting Adaptation of Online Code Examples [ICSE'19]

Visualizing API Usage Examples at Scale [CHI'18]

Focal API

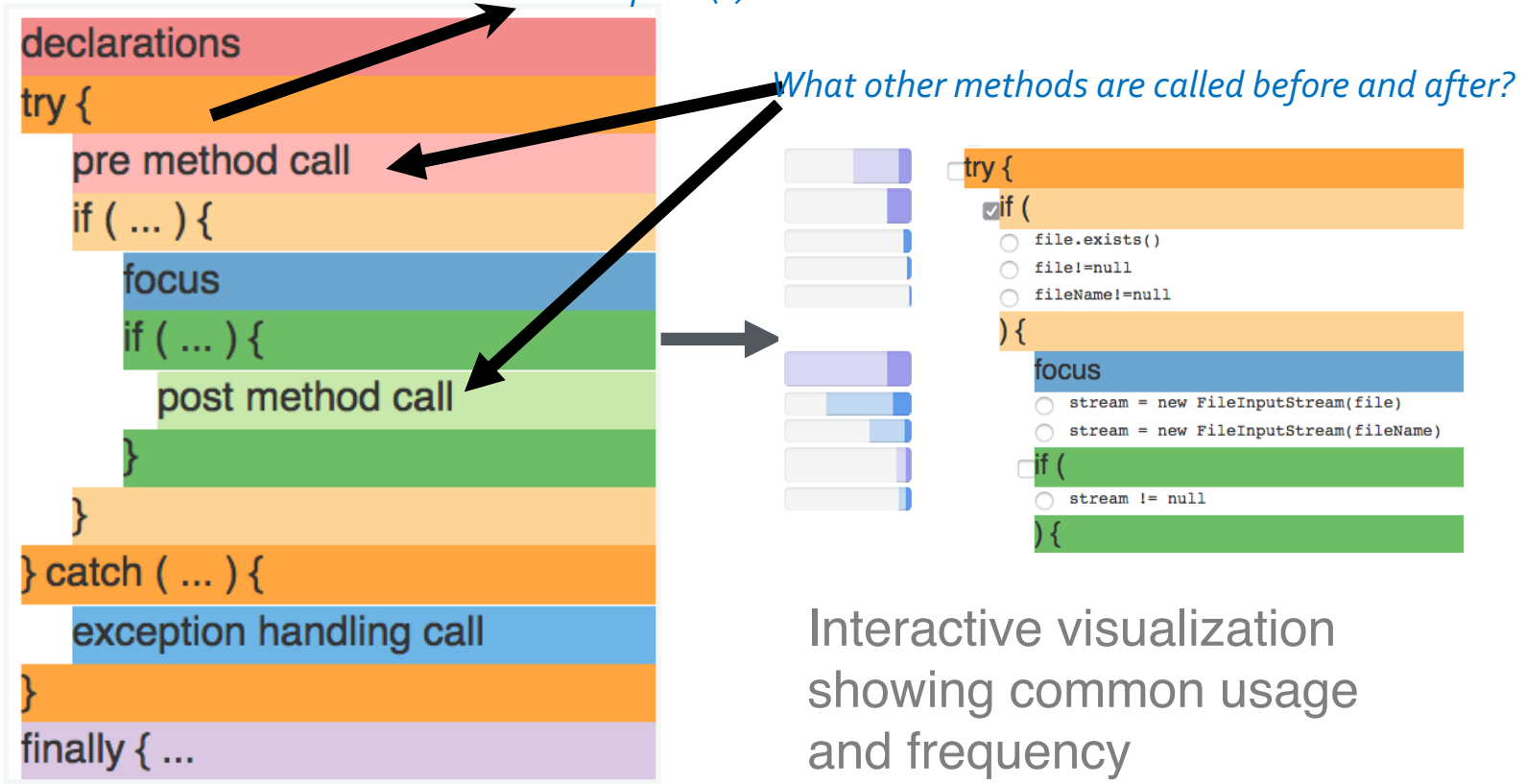


Many code examples using this call

github stackoverflow GitLab Atlassian Bitbucket CODE PROJECT

What exception(s) are thrown?

What other methods are called before and after?



Interactive visualization showing common usage and frequency

Cross-example counts for `fileInputStream`

Toggle Labels | Fold Code | Show Default | Show Less | Show More | Show All

Counts

Blocks of options

declarations
- File file = new File(String)
- File file = new File(*)
- File file = new File(*,String)

try {
  pre method call
  - file.length()
  - file.getName()
  - file.getAbsolutePath()

  if (
  - file.exists()
  - file!=null
  - fileName!=null

) {
    focus
    - stream = new FileInputStream(file)
    - stream = new FileInputStream(fileName)

    if (
    - stream != null
    - null != stream
    - stream.read(outputByte,0,4096) != -1

) {
      post method call
      - stream.close()
      - Properties.load(stream)
      - new BufferedInputStream(stream)

100 concrete examples from GitHub

Reset | Active Filters:

Link to the GitHub source code

```
@Override
public void readFromFile(String filename) throws IOException {
  in = new FileInputStream(filename);
  prop.load(in);
}
```

Link to the GitHub source code

```
private synchronized InputStream openStream() throws IOException {
  if (file != null) {
    return new FileInputStream(file);
  } else {
    return new ByteArrayInputStream(memory.getBuffer(), 0, memory.getCount());
  }
}
```

Link to the GitHub source code

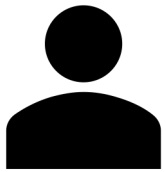
```
public InputStream getResourceContents(String path) {
  File file = new File(_basePath + "/" + path);
  try {
    return new FileInputStream(file);
  } catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
  }
}
```

Link to the GitHub source code

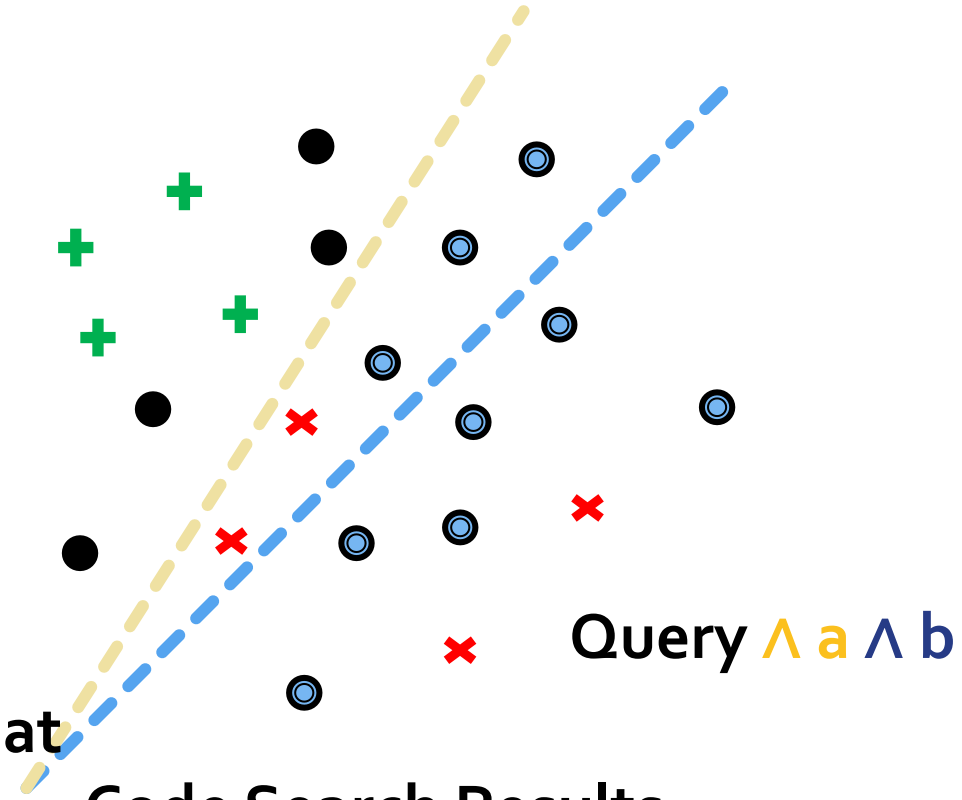
```
public InputStream getInputStream() throws MessagingException {
  try {
    return new BinaryTempFileBodyInputStream(new FileInputStream(mFile));
  } catch (IOException ioe) {
    throw new MessagingException("Unable to open body", ioe);
  }
}
```

How can we construct a search pattern with a human in the loop?

ALICE: Active Inductive Logic Programming for Code Search [ICSE 2019]



Input: More instance labels
ALICE: Keep refining the query
Output: A smaller set of method locations that match the new query



Query Refinement via Active Learning

```
public void getUserName(String id) {
   try {
        ResultSet set = db.executeQuery(
                "select name from users where id=” + id);
        while (set.next()) { … }
   } catch (SQLException e) { …}
}
```

```
public void queryDatabase() {
   try {
        ResultSet result = s.executeQuery("select * from customers”);
        while (result.next()) { … }
   } catch (SQLException e) { …}
}
```

```
public List get() {
    ResultSet set = stmt.executeQuery("select * from t”);
    List l = new List();
    while (set.next()) { … }
    return l;
}
```

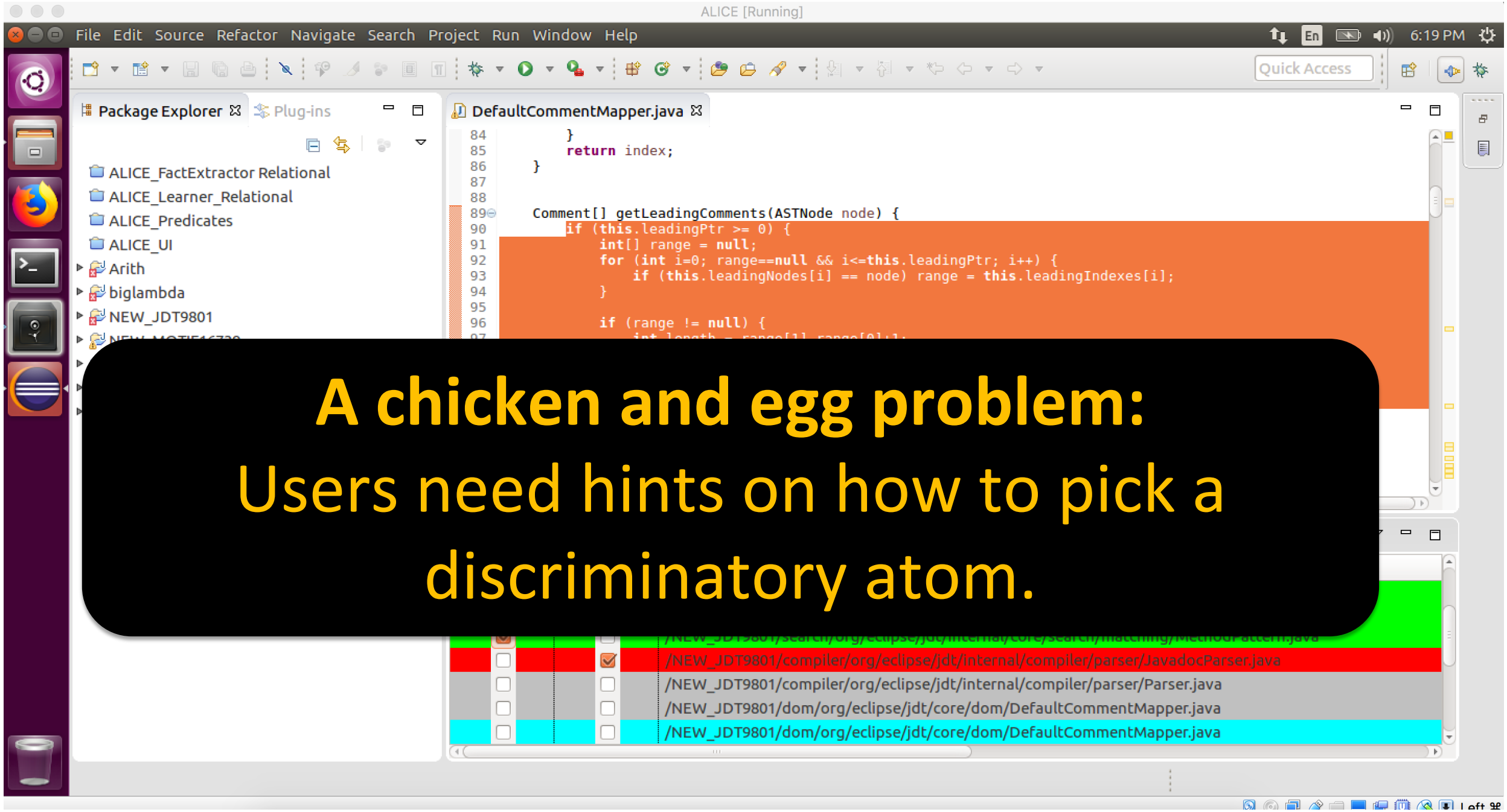
Refined Query

methodDec (i_0, m) ∧
type (i_1, ResultSet) ∧
contains (i_0, i_1) ∧
methodCall(i_2, executeQuery) ∧
contains (i_0, i_2) ∧
looplike (i_3, "*.next()") ∧
contains (i_0, i_3) ∧
exception (i_4, SQLException),
contains (i_0, i_4)

Query Refinement Optimization

$$\texttt{Specialize}(h_{i-1}, P, N) = \underset{h_i}{\text{argmax}} \sum_{p \in P} [p \models h_i]$$

$$\text{such that } h_i \models h_{i-1} \text{ and } \forall n \in N, n \not\models h_i$$


ALICE [Running]
File Edit Source Refactor Navigate Search Project Run Window Help
6:19 PM
Quick Access
Package Explorer
Plug-ins
ALICE_FactExtractor Relational
ALICE_Learner_Relational
ALICE_Predicates
ALICE_UI
Arith
biglambda
NEW_JDT9801
DefaultCommentMapper.java
84 }
85 return index;
86 }
87
88
89 Comment[] getLeadingComments(ASTNode node) {
90 if (this.leadingPtr >= 0) {
91 int[] range = null;
92 for (int i=0; range==null && i<=this.leadingPtr; i++) {
93 if (this.leadingNodes[i] == node) range = this.leadingIndexes[i];
94 }
95
96 if (range != null) {
/NEW_JDT9801/compiler/org/eclipse/jdt/internal/compiler/parser/JavadocParser.java
/NEW_JDT9801/compiler/org/eclipse/jdt/internal/compiler/parser/Parser.java
/NEW_JDT9801/dom/org/eclipse/jdt/core/dom/DefaultCommentMapper.java
/NEW_JDT9801/dom/org/eclipse/jdt/core/dom/DefaultCommentMapper.java
A chicken and egg problem:
Users need hints on how to pick a discriminatory atom.

Developer Tools for Big Data Systems
& Heterogeneous Hardware

Spark hadoop

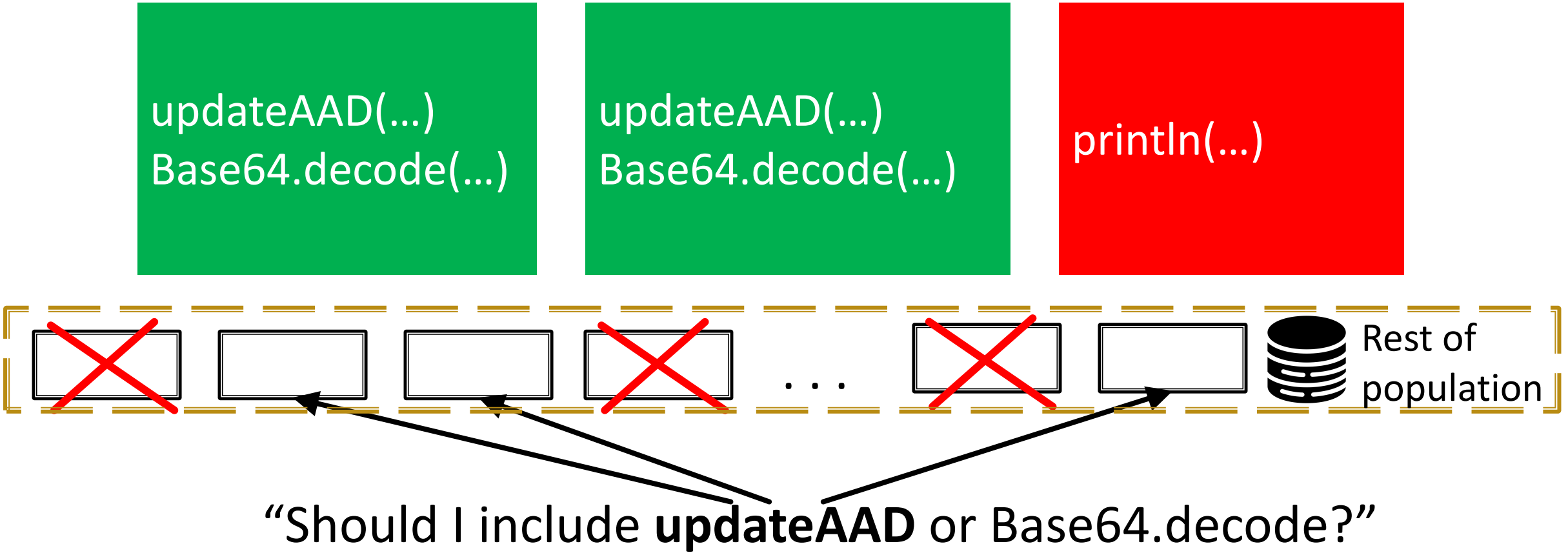
amazon web services | EC2 intel FPGA DEVCLOUD

...

Amazon Scholar

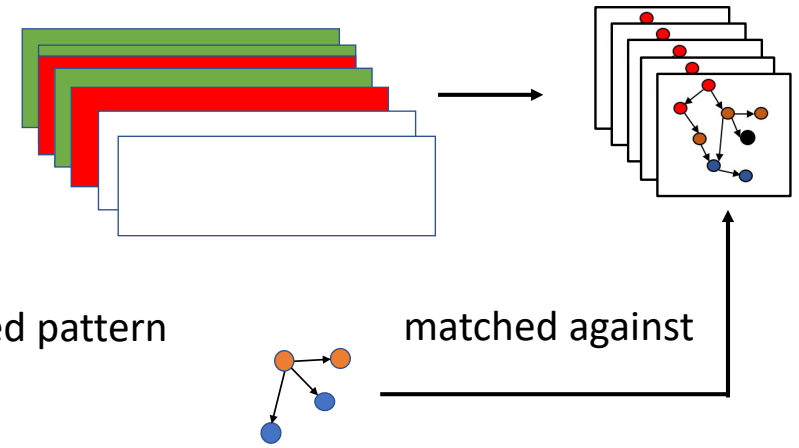
aws

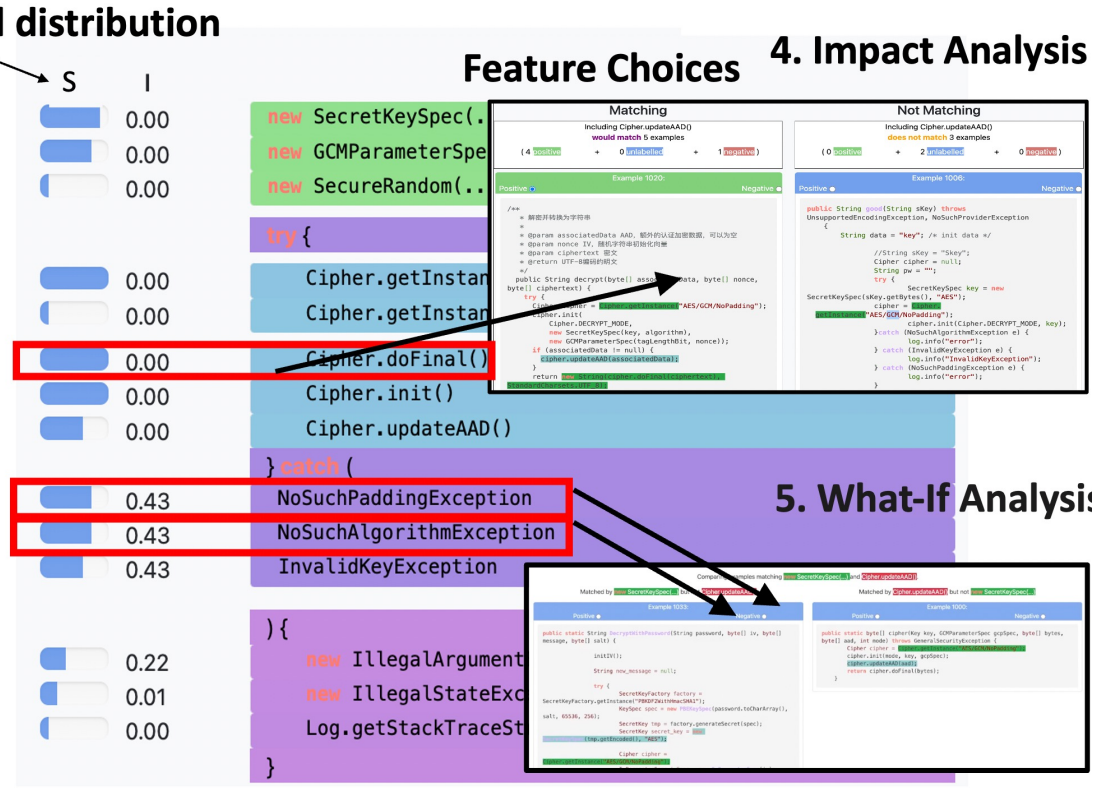
Challenge: Too many possible choices to consider for pattern refinement



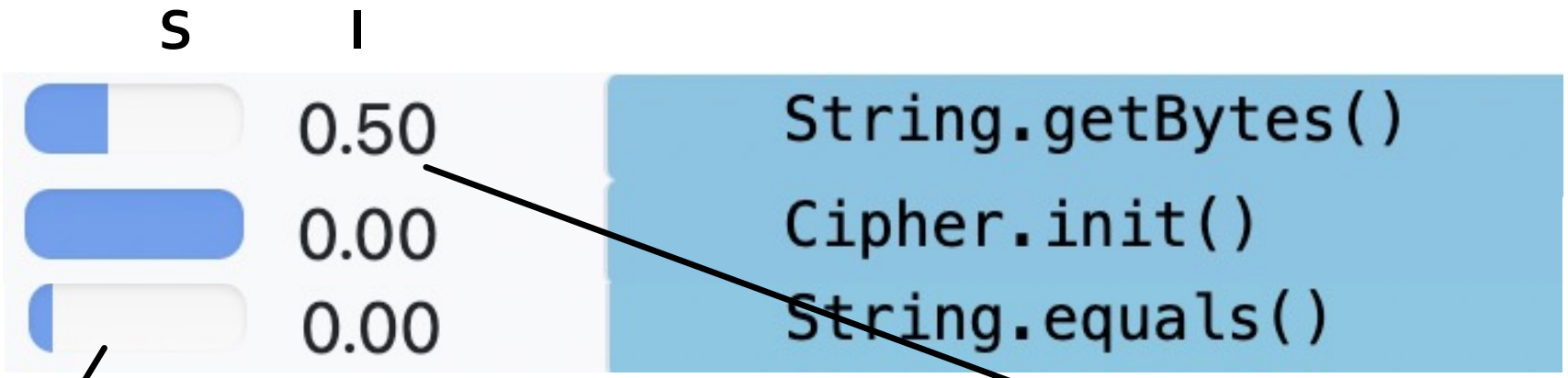
SURF: Scaling Code Pattern Inference with Interactive What-If Analysis [ICSE 2024]

- Matching on Program Dependence Graphs
- Simultaneous overlay
- Global distribution
- Impact Analysis, and
- What-if Analysis





Design 1: Hint on Global Distribution



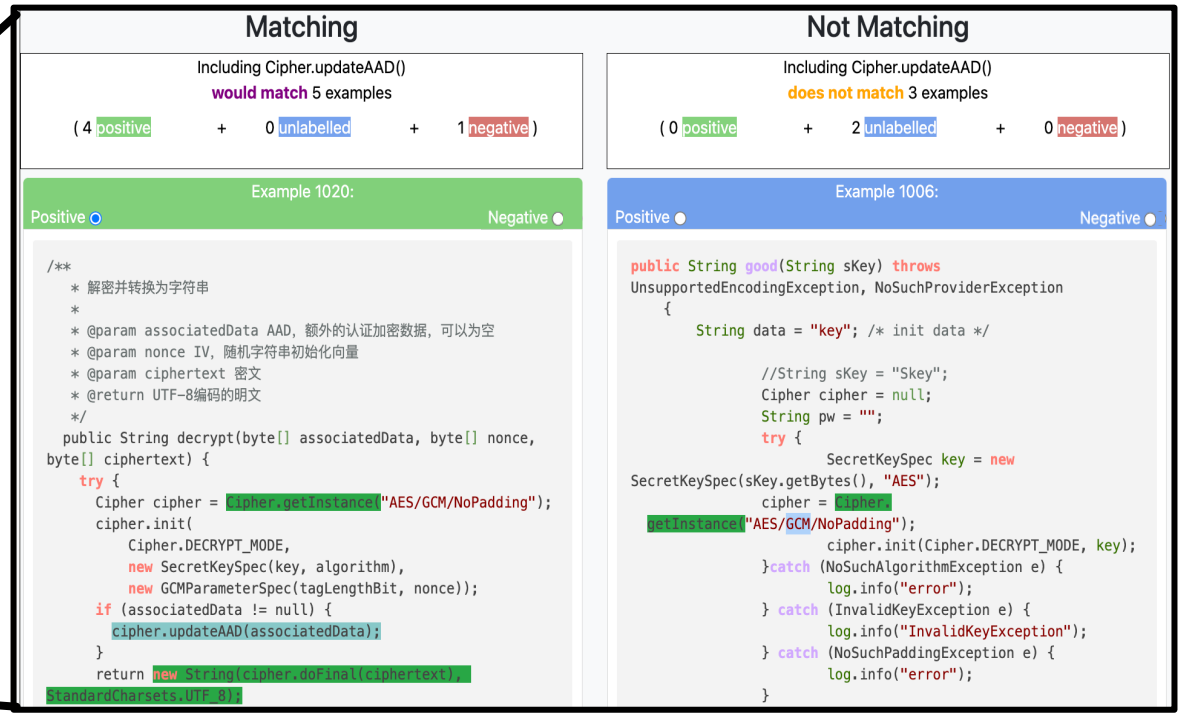
Support:
Number of times a statement appears, regardless of label

Information gain:
Improvement in entropy for separating positive from negative instances

$$\text{Entropy}(G_P) - \left(\frac{|G_m|}{|G_P|} \times \text{Entropy}(G_m) + \frac{|G_e|}{|G_P|} \times \text{Entropy}(G_e) \right)$$

Design 2: Impact Analysis

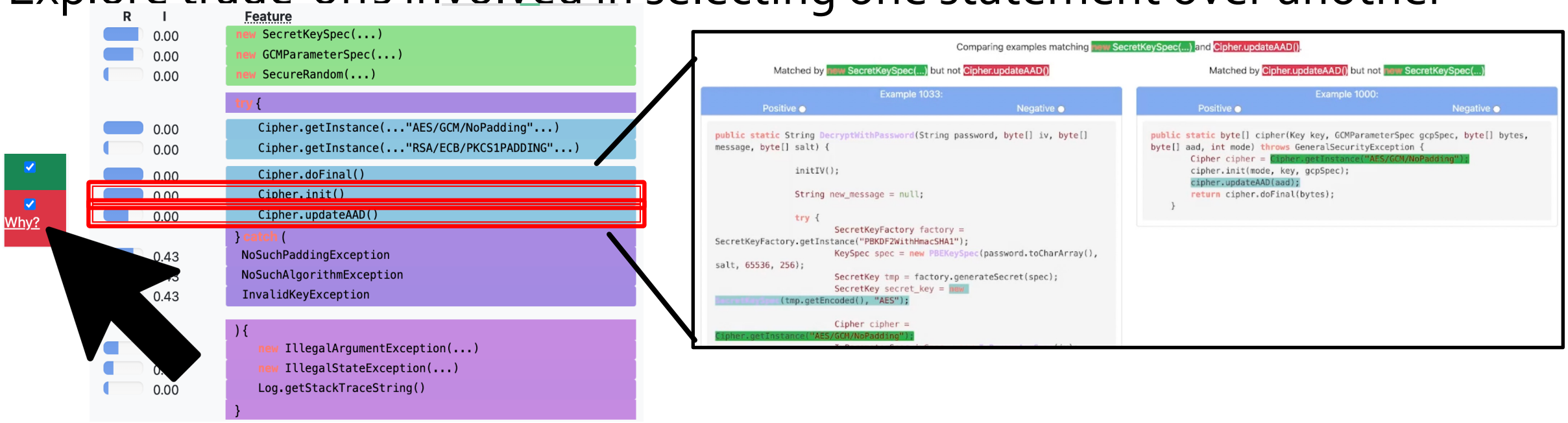
- Focuses attention to which instances will be included/excluded for each statement choice



"If I include this specific code line, how many will I match?"

Design 3: What-If Analysis

Explore trade-offs involved in selecting one statement over another



"Which code statement is better to include?"

41

Recap: A *Journey* through Searching Similar Code

- What motivated us? Systematic changes
- What were early attempts? Rule-based change abstraction
- How serious is this problem? Pretty serious
- How can we automate? Generalized patch synthesis

Several chicken and egg problems: Users need example patches, hints on what to generalize, and hints on how to pick a discriminatory atom.

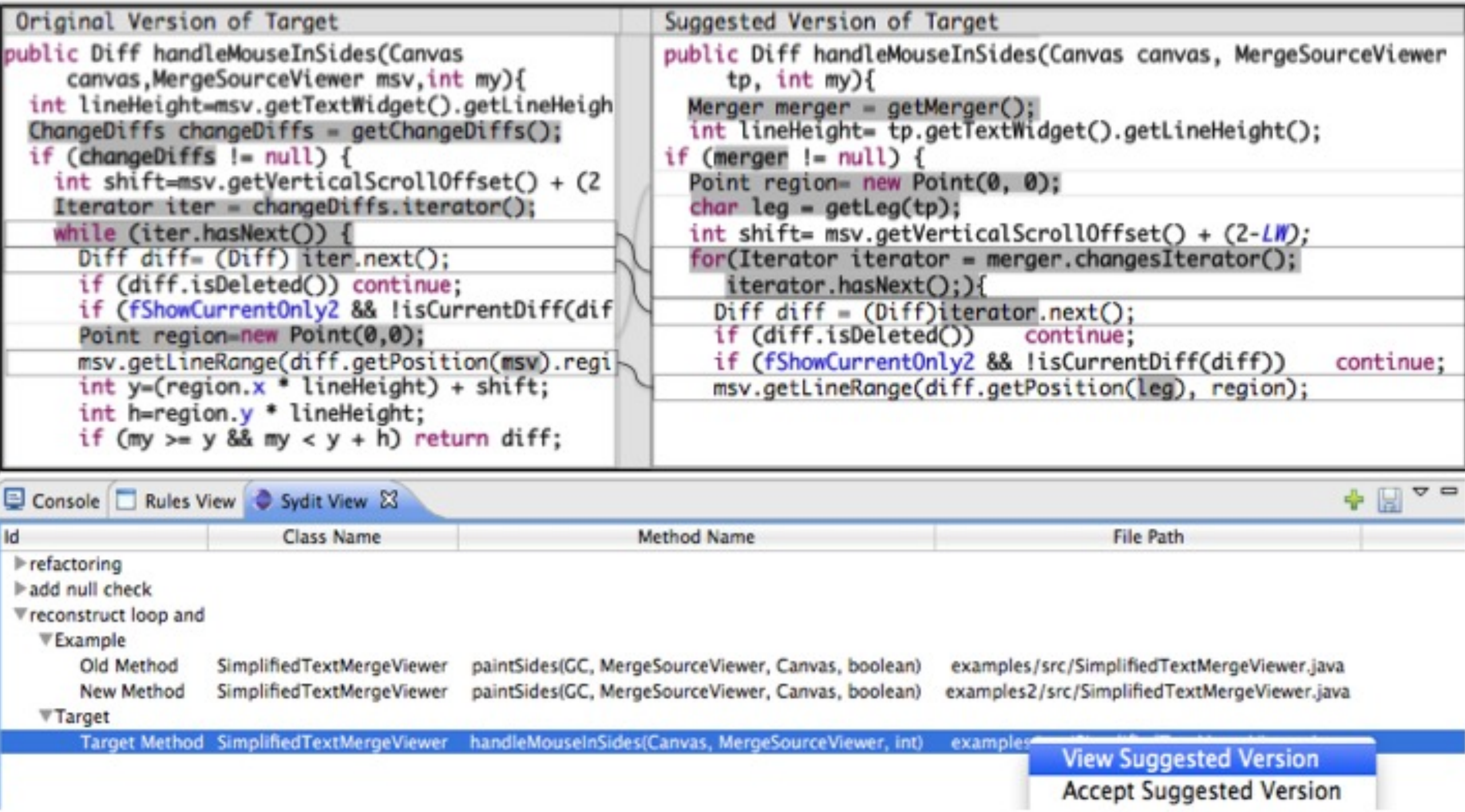
- How can we examine variations at scale? Simultaneous overlay
- How to construct a search pattern with a human in the loop? Hints on global distribution and interactive what-if analysis

Discussion: Would've, could've, should've, etc.

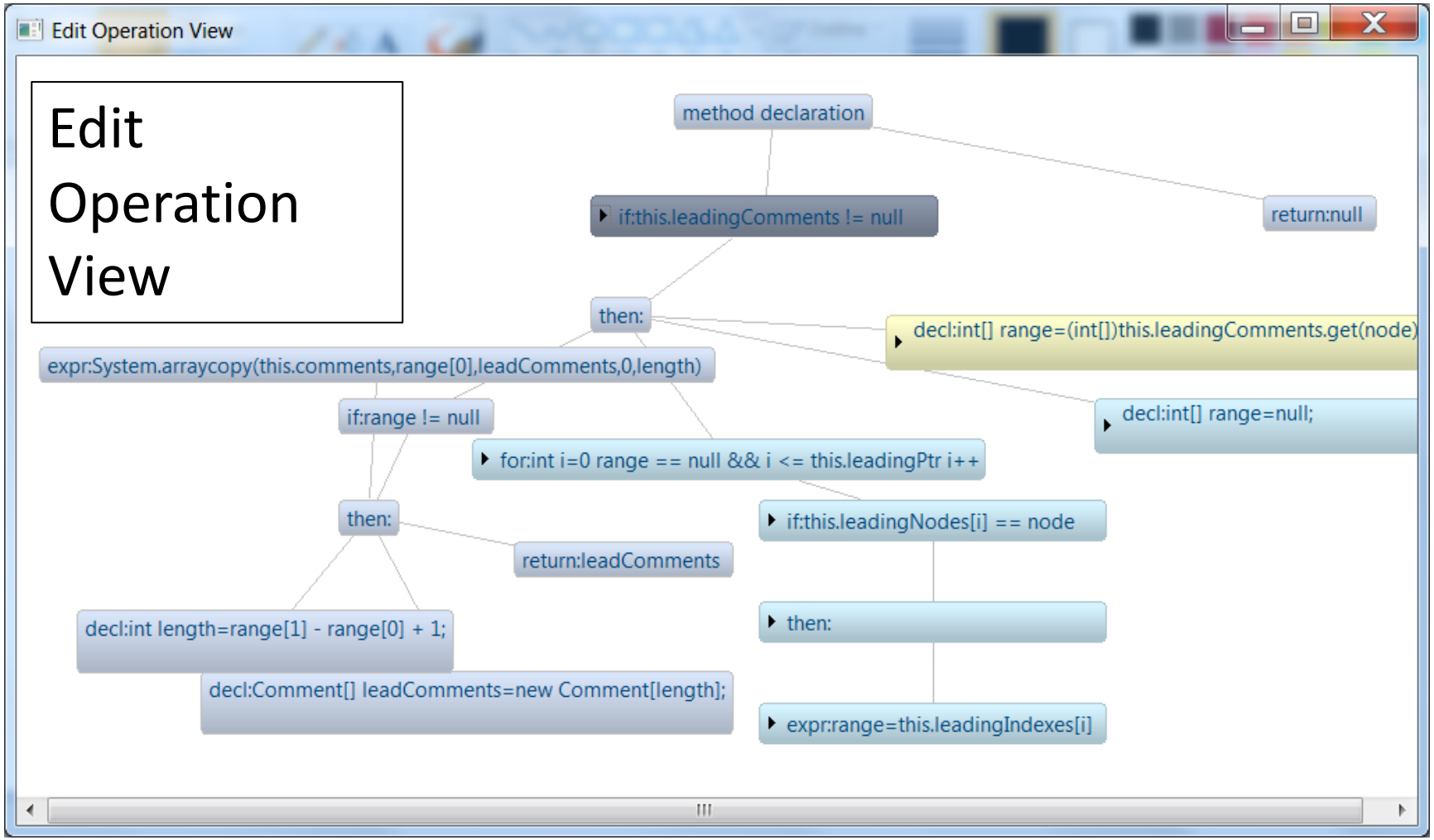
- DSL
- Code embeddings
- LLM
- Information retrieval
- Search with multi-modality: text, video, etc.

Thank you!

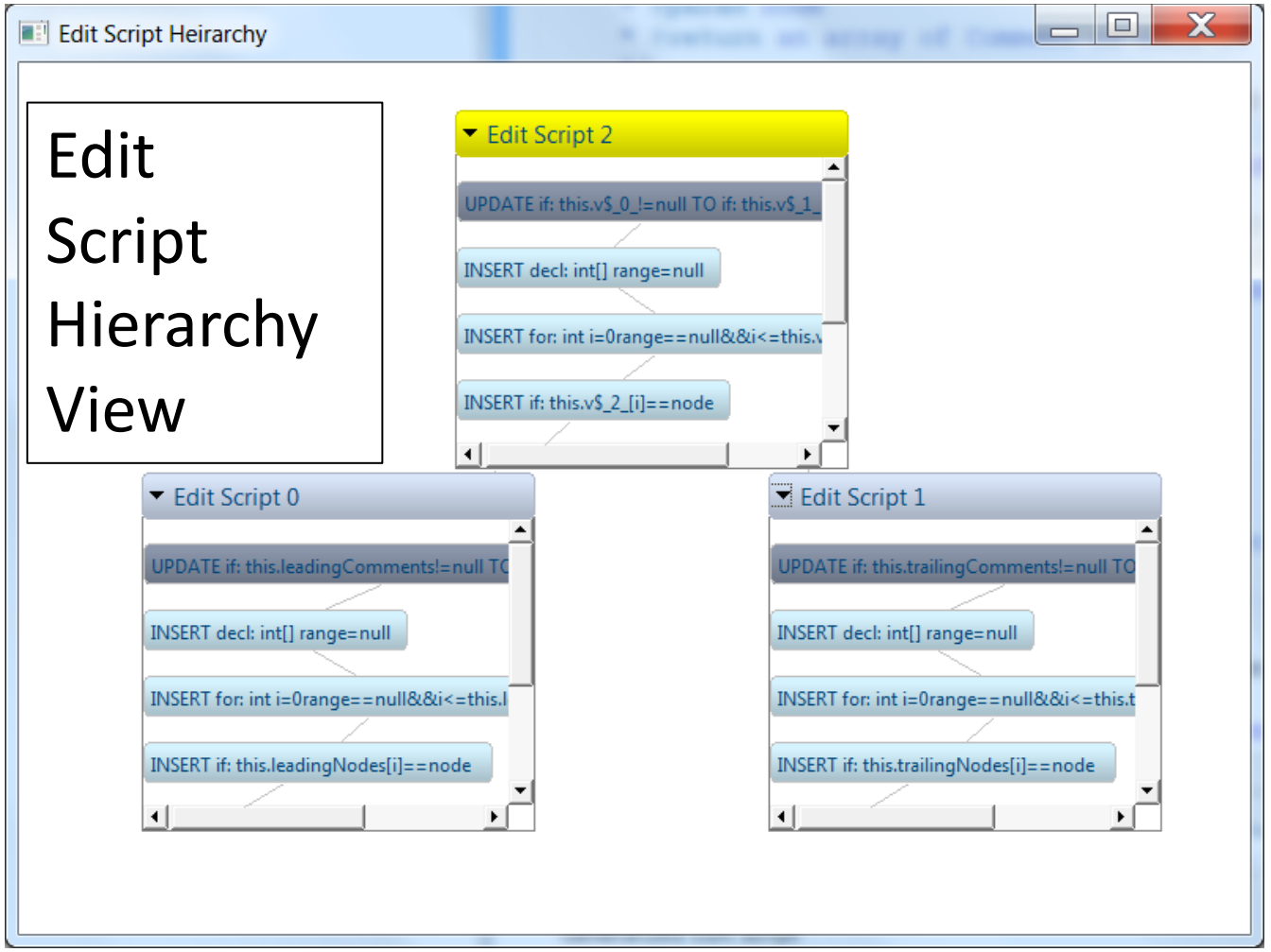
Thanks to Baishakhi Ray, Na Meng, Tianyi Zhang, HongJin Kang, Myoungkyu Song, David Notkin, Elena Glassman, Dan Grossman, John Jacobellis, Björn Hartmann, Cristina Lopes, Tyler McDonnell, Nachiappan Nagappan, Mihir Mathur, Kathryn McKinley, Suzette Person, Joseph Pinedo, Hridesh Rajan, Anastasia Reinhardt, Neha Rungta, Aishwarya Sivaraman, Ganesha Upadhyaya, Guy Van den Broeck, Christopher Wiley, Gary Wilson Jr., Di Yang, Thomas Zimmermann


Original Version of Target
public Diff handleMouseInSides(Canvas
canvas,MergeSourceViewer msv,int my){
int lineHeight=msv.getTextWidget().getLineHeigh
ChangeDiffs changeDiffs = getChangeDiffs();
if (changeDiffs != null) {
int shift=msv.getVerticalScrollOffset() + (2
Iterator iter = changeDiffs.iterator();
while (iter.hasNext()) {
Diff diff= (Diff) iter.next();
if (diff.isDeleted()) continue;
if (fShowCurrentOnly2 && !isCurrentDiff(dif
Point region=new Point(0,0);
msv.getLineRange(diff.getPosition(msv).regi
int y=(region.x * lineHeight) + shift;
int h=region.y * lineHeight;
if (my >= y && my < y + h) return diff;
Suggested Version of Target
public Diff handleMouseInSides(Canvas canvas, MergeSourceViewer
tp, int my){
Merger merger = getMerger();
int lineHeight= tp.getTextWidget().getLineHeight();
if (merger != null) {
Point region= new Point(0, 0);
char leg = getLeg(tp);
int shift= msv.getVerticalScrollOffset() + (2-LW);
for(Iterator iterator = merger.changesIterator();
iterator.hasNext();){
Diff diff = (Diff)iterator.next();
if (diff.isDeleted()) continue;
if (fShowCurrentOnly2 && !isCurrentDiff(diff)) continue;
msv.getLineRange(diff.getPosition(leg), region);
Console
Rules View
Sydit View
Id
Class Name
Method Name
File Path
refactoring
add null check
reconstruct loop and
Example
Old Method
SimplifiedTextMergeViewer
paintSides(GC, MergeSourceViewer, Canvas, boolean)
examples/src/SimplifiedTextMergeViewer.java
New Method
SimplifiedTextMergeViewer
paintSides(GC, MergeSourceViewer, Canvas, boolean)
examples2/src/SimplifiedTextMergeViewer.java
Target
Target Method
SimplifiedTextMergeViewer
handleMouseInSides(Canvas, MergeSourceViewer, int)
View Suggested Version
Accept Suggested Version

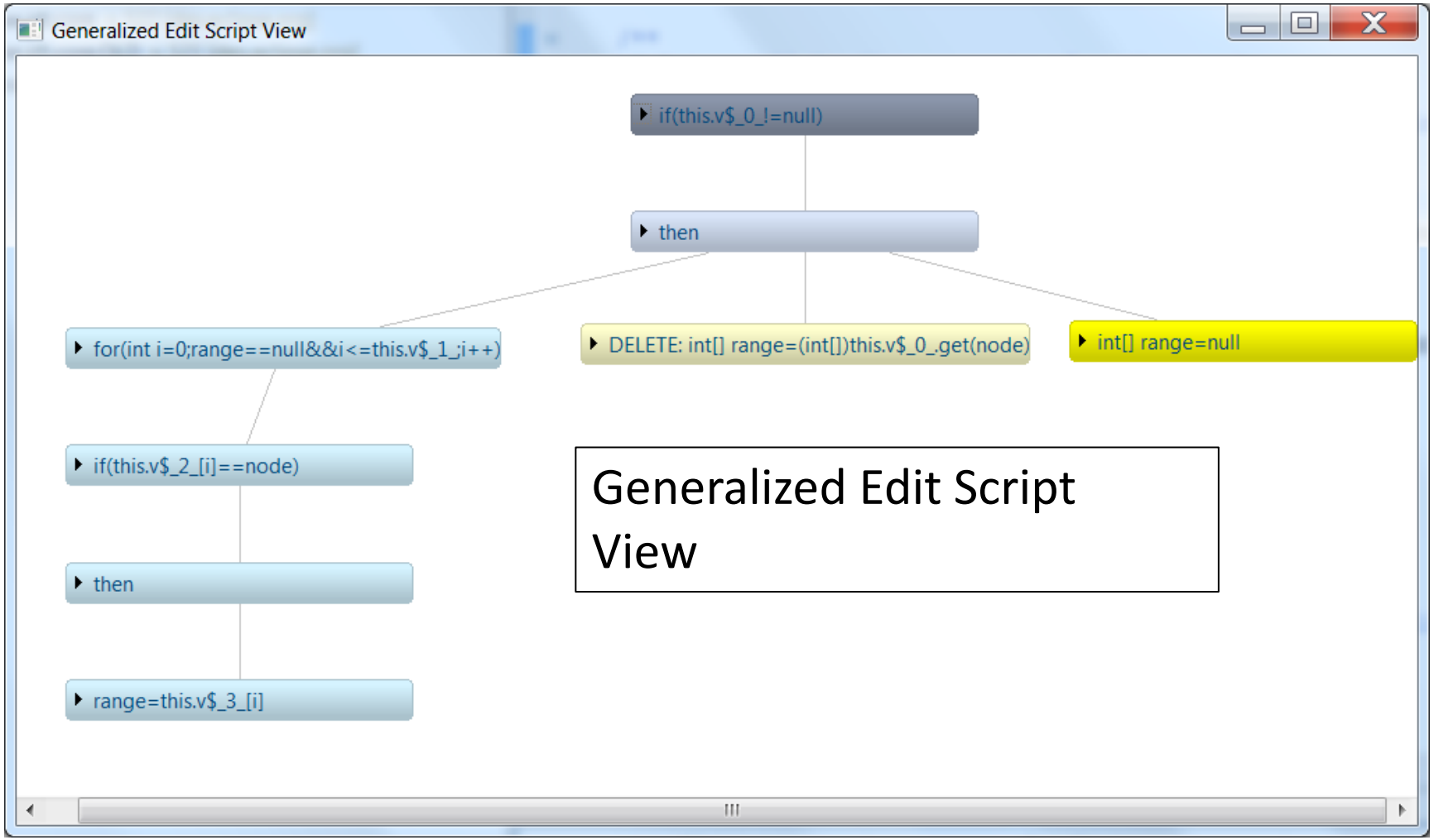
For each change example, a user can view AST edit operations



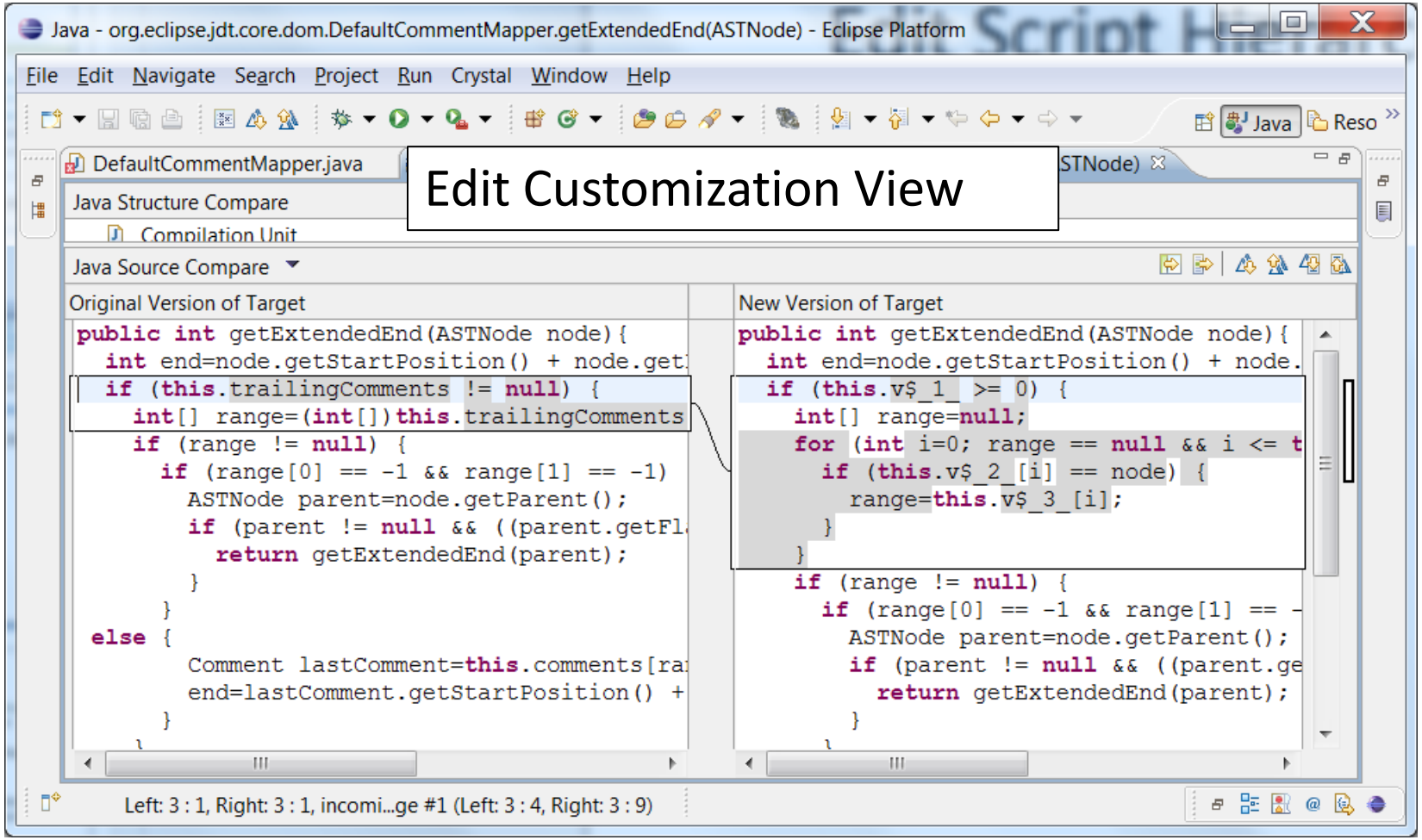
A user can view a hierarchy of edit scripts and select one of them



A User can inspect a generalized edit script



A user can correct suggested edits before applying the suggestion

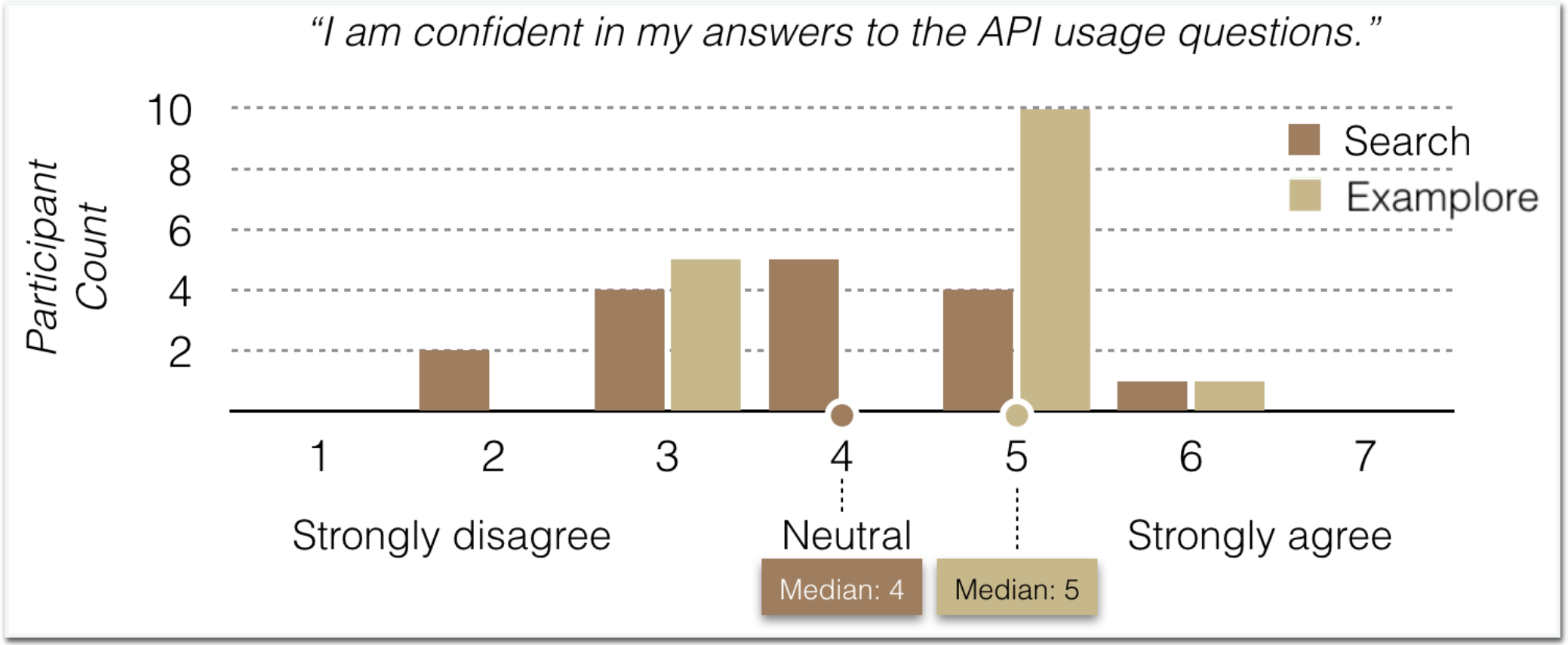


Index	Bug(patches)	m_i	Edit Location					Operations		
			Σ	✓	P%	R%	A%	E	C	$E_{A\%}$
2	82429(2)	16	13	12	92	75	81	9	9	100
4	139329(3)	6	2	2	100	33	74	6	3	50
7	103863(5)	7	7	7	100	100	100	34	34	100
8	129314(3)	3	4	4	100	100	100	2	2	100
16	95409(3)	7	9	9	100	100	78	4	4	100
24	98198(2)	9	15	15	100	100	95	3	3	100

On average, Lase finds edit locations with 99% precision, 89% recall, and applies edits with 91% accuracy.

For three bugs, Lase suggests in total 9 edits that developers missed and later confirmed.

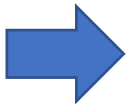
Lab Study Results



Represent Code as Logic Facts

Fact Predicate
if (ID, CONDITION)
loop (ID, CONDITION)
parent (ID, ID)
next (ID, ID)
methodCall (ID, NAME)
type (ID, NAME)
exception (ID, NAME)
methodDec (ID, NAME)

```
public void queryDB() {
   try {
        Connection con = DriverManager.getConnection(
              "jdbc:mysql://localhost:3306/db","root","root");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select * from emp");
        while (rs.next())  {
           System.out.println(rs.getInt(1));
        }
        con.close();
   } catch (SQLException e) {
       System.out.println(e);
   }
}
```



Extracted Logic Facts

methodDec (0, queryDB),
type (1, Connection),
parent (0, 1),
methodCall(2, getConnection),
parent (0, 2),
next (2, 1),
…
loop (7, "rs.next()"),
methodCall (8, getInt),
parent (7, 8),
…
exception (10, SQLException),
parent (0, 10),
…

Formulate a Search Query

- A user selects a code example and annotate important features.

```
public void queryDB() {
   try {
        Connection con = DriverManager.getConnection(
              "jdbc:mysql://localhost:3306/db","root","root");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select * from emp");
        while (rs.next()) {
           System.out.println(rs.getInt(1));
        }
        con.close();
   } catch (SQLException e){
       System.out.println(e);
   }
}
```

A code example with user annotations

$$
\begin{aligned}
&\text{methodDec}(i_0, m) \wedge \\
&\text{type}(i_1, \text{ResultSet}) \wedge \\
&\text{contains}(i_0, i_1) \wedge \\
&\text{methodCall}(i_2, \text{executeQuery}) \wedge \\
&\text{contains}(i_0, i_2) \wedge \\
&\text{looplike}(i_3, \text{"*.next()"}) \wedge \\
&\text{contains}(i_0, i_3)
\end{aligned}
$$

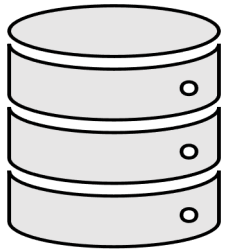
search query

Logic-based Code Search

Search Query

methodDec (i_0, m) ∧
type (i_1, ResultSet) ∧
contains (i_0, i_1) ∧
methodCall(i_2, executeQuery) ∧
contains (i_0, i_2) ∧
looplike (i_3, "*.next()") ∧
contains (i_0, i_3)

Fact Base



+

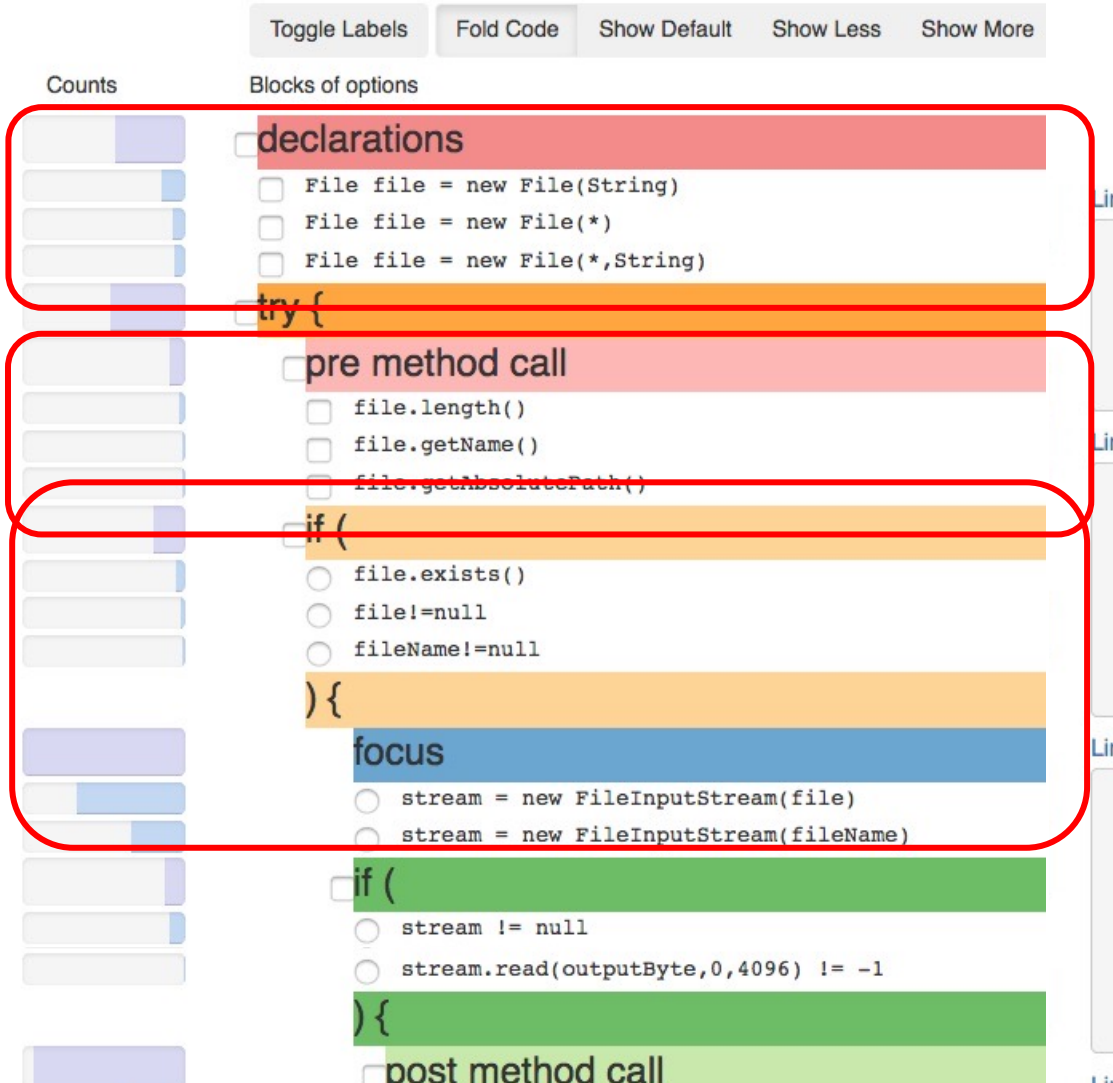
Fact Rules

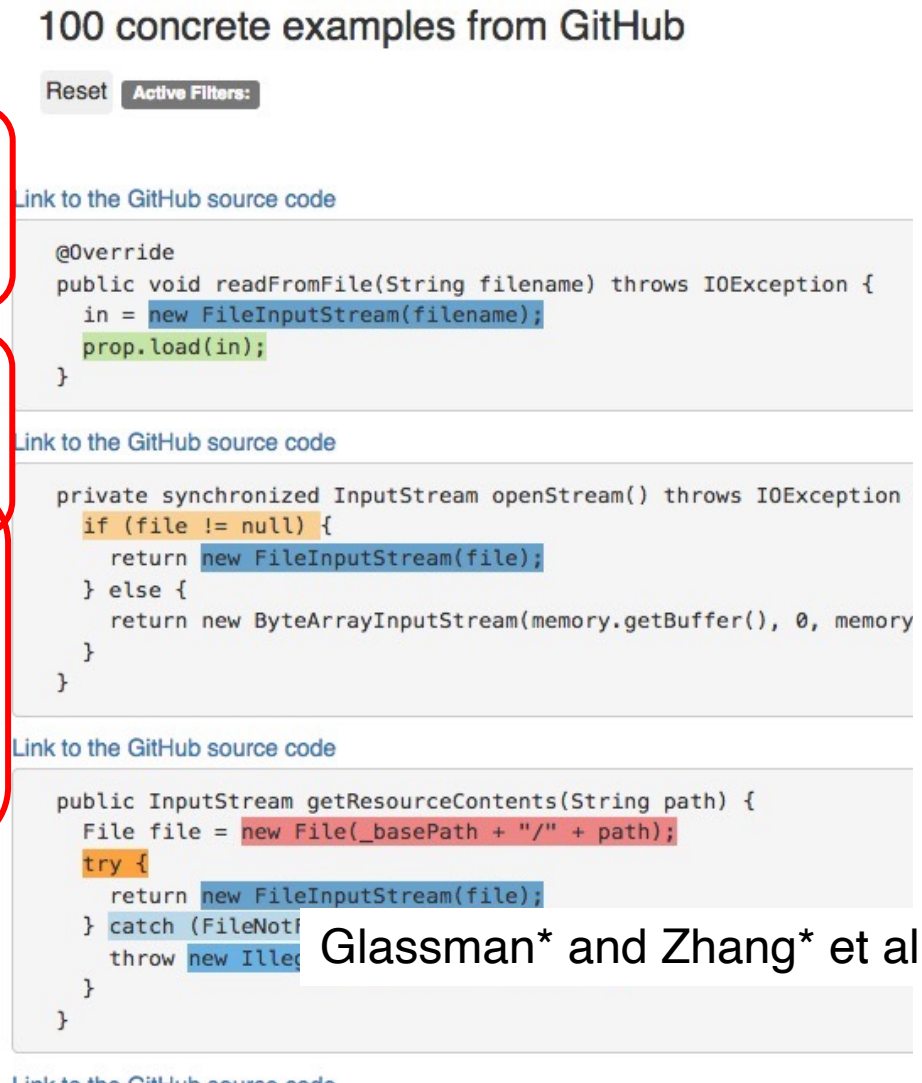
Matched Code

```
public void getUserName(String id) {
   try {
        ResultSet set = db.executeQuery(
                "select name from users where id=” + id);
        while (set.next()) { … }
   } catch (SQLException e) { …}
}

public void queryDatabase() {
   try {
        ResultSet result = s.executeQuery("select * from customers”);
        while (result.next()) { … }
   } catch (SQLException e) { …}
}

public List get() {
    ResultSet set = stmt.executeQuery("select * from t”);
    List l = new List();
    while (set.next()) { … }
    return l;
}
```

and 32 other matched locations

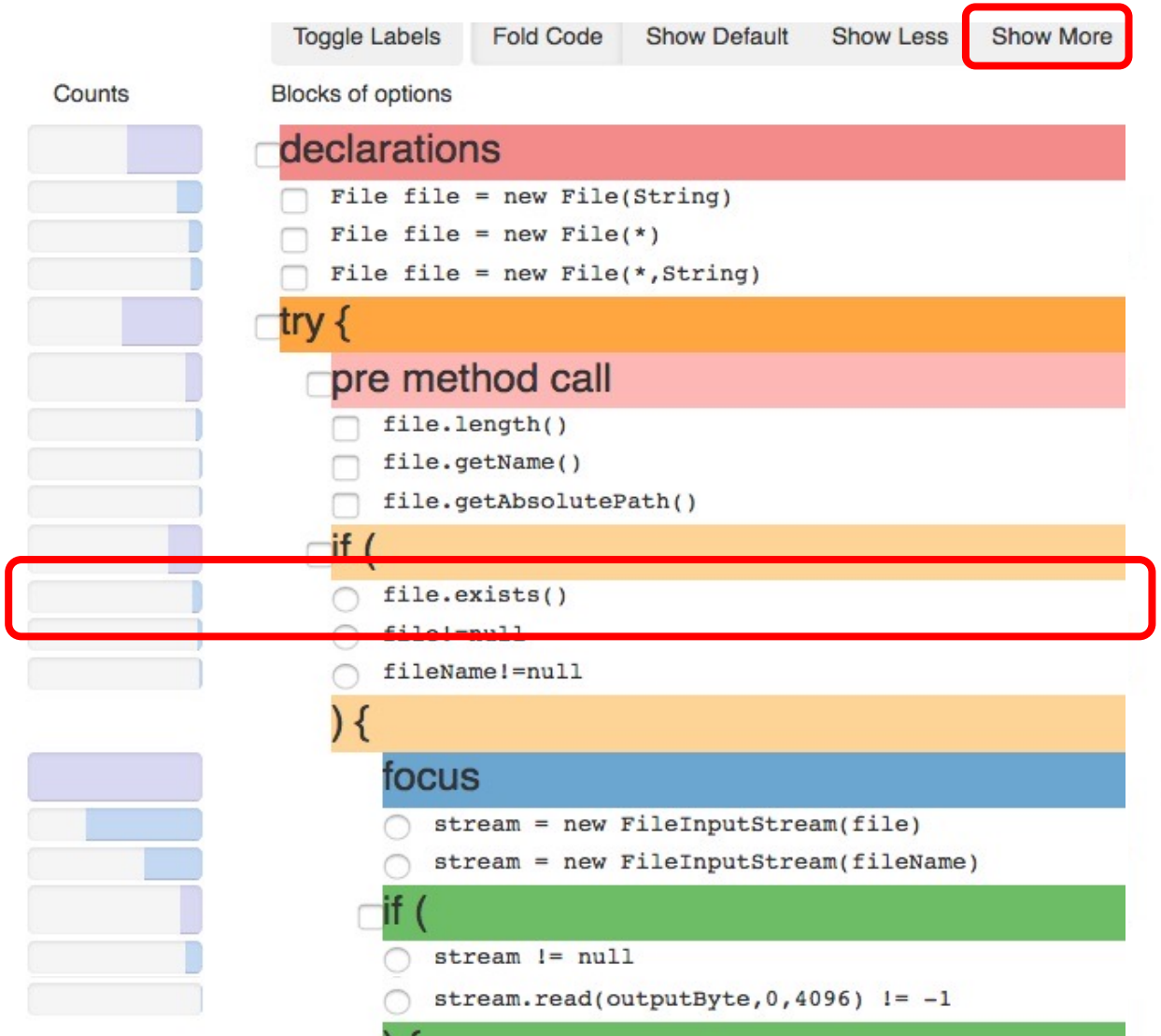
Align and aggregate structured call sequences into a single view





Glassman* and Zhang* et al. CHI 2018

Explore less frequent but critical API usage features



100 concrete examples from GitHub

Reset Active Filters:

Link to the GitHub source code

```
@Override
public void readFromFile(String filename) throws IOException {
  in = new FileInputStream(filename);
  prop.load(in);
}
```

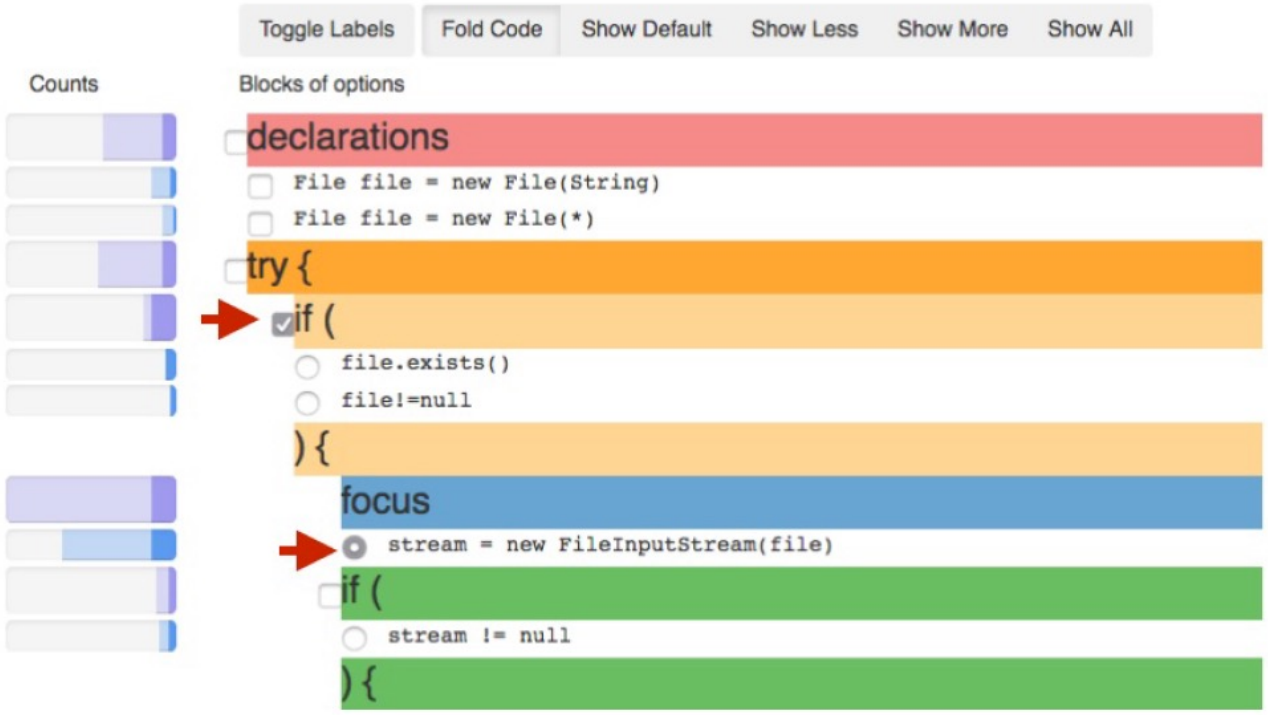
Link to the GitHub source code

```
private synchronized InputStream openStream() throws IOException
  if (file != null) {
    return new FileInputStream(file);
  } else {
    return new ByteArrayInputStream(memory.getBuffer(), 0, memory
  }
}
```

Link to the GitHub source code

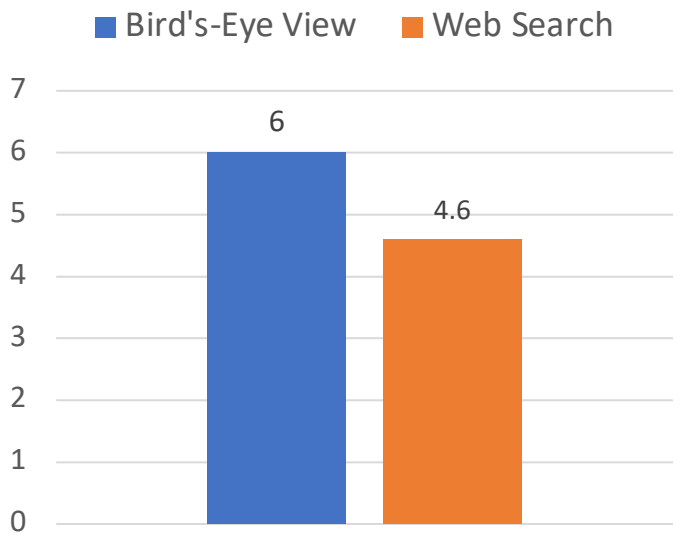
```
public InputStream getResourceContents(String path) {
  File file = new File(_basePath + "/" + path);
  try {
    return new FileInputStream(file);
  } catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
  }
```

Interactively building your own patterns



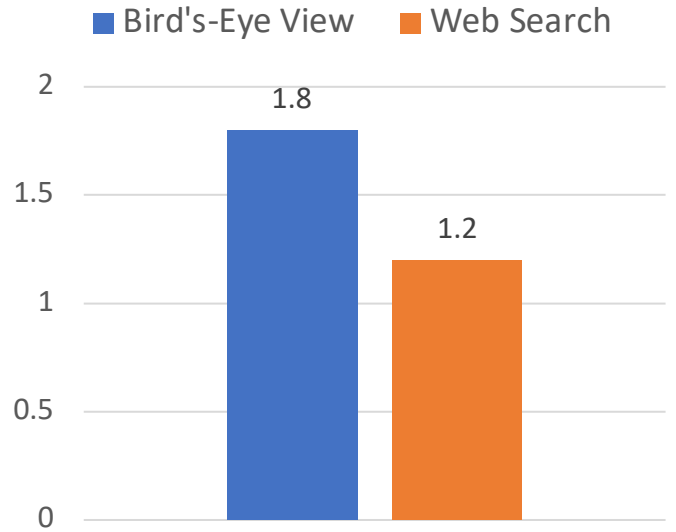
A within-subject user study
RQ1. Does the bird's-eye view help build robust API knowledge?

Number of questions that are answered correctly



(paired t-test: t=3.02, df=15, p-value=0.0086)

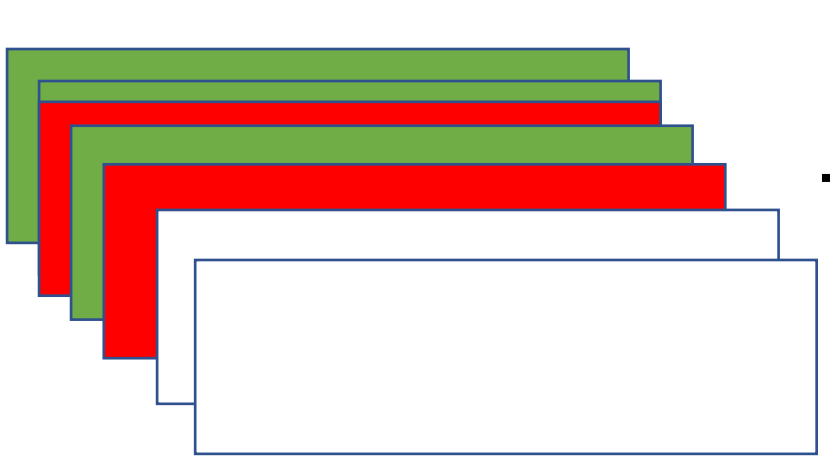
Number of correct answers per question

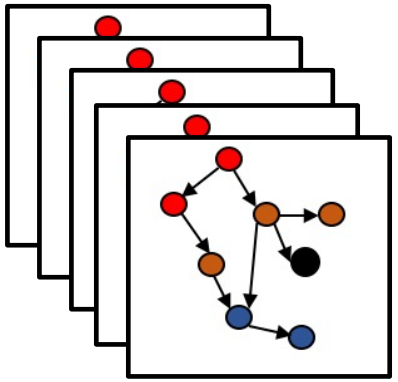


(paired t-test: t=3.84, df=15, p-value=0.0016)

Key Result: Users with the bird's-eye view answered API usage questions more ***correctly*** and ***comprehensively***.

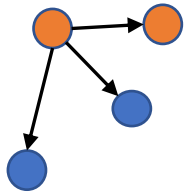
Design 1: Infer Common PDG Subgraph





Population of graphs

The inferred pattern



matched against

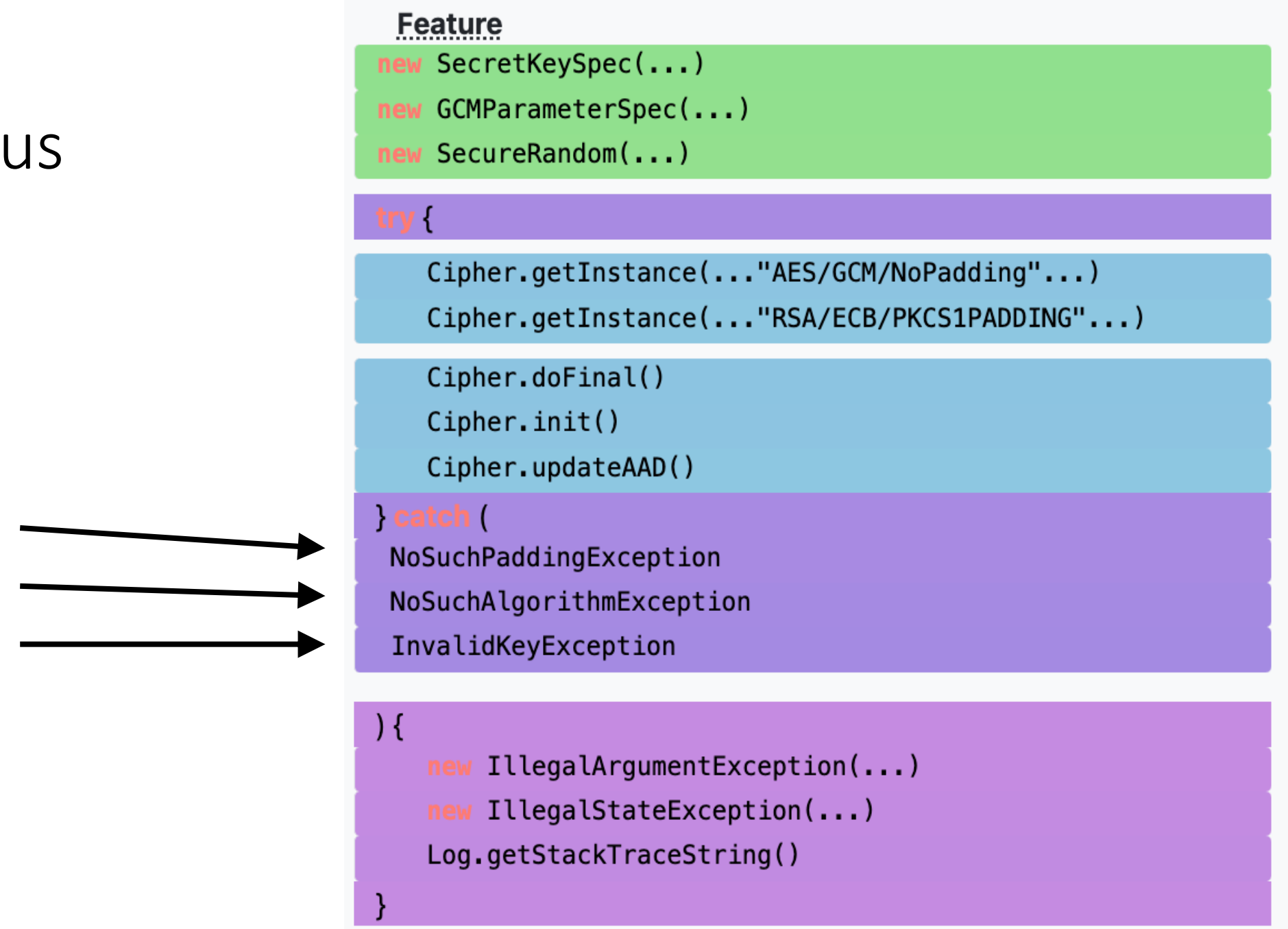
Design 2: Simultaneous Overlay

"What constructors are called?"

"What exceptions are caught?"

Design 2: Simultaneous Overlay

Statement Choice 1
Statement Choice 2
Statement Choice 3



Challenge 1: Instance-level feedback provides too little information

Cipher.getInstance(AES)
Cipher.init(…)
System.println(…)

Cipher.getInstance(AES)
Cipher.init(…)
System.println(…)

Cipher.getInstance(DES)

Cipher.getInstance(..) ?

Cipher.getInstance(AES) ?

Cipher.getInstance(AES)
System.println(…) ?

Results: Users can better understand the API usage distribution when given guidance

30%

more correct answers

20%

less time required

1.8X

more likely to construct the target pattern

Improvements are statistically significant ($p < 0.001$) following a mixed-effects linear model accounting for ordering, tool, and task.