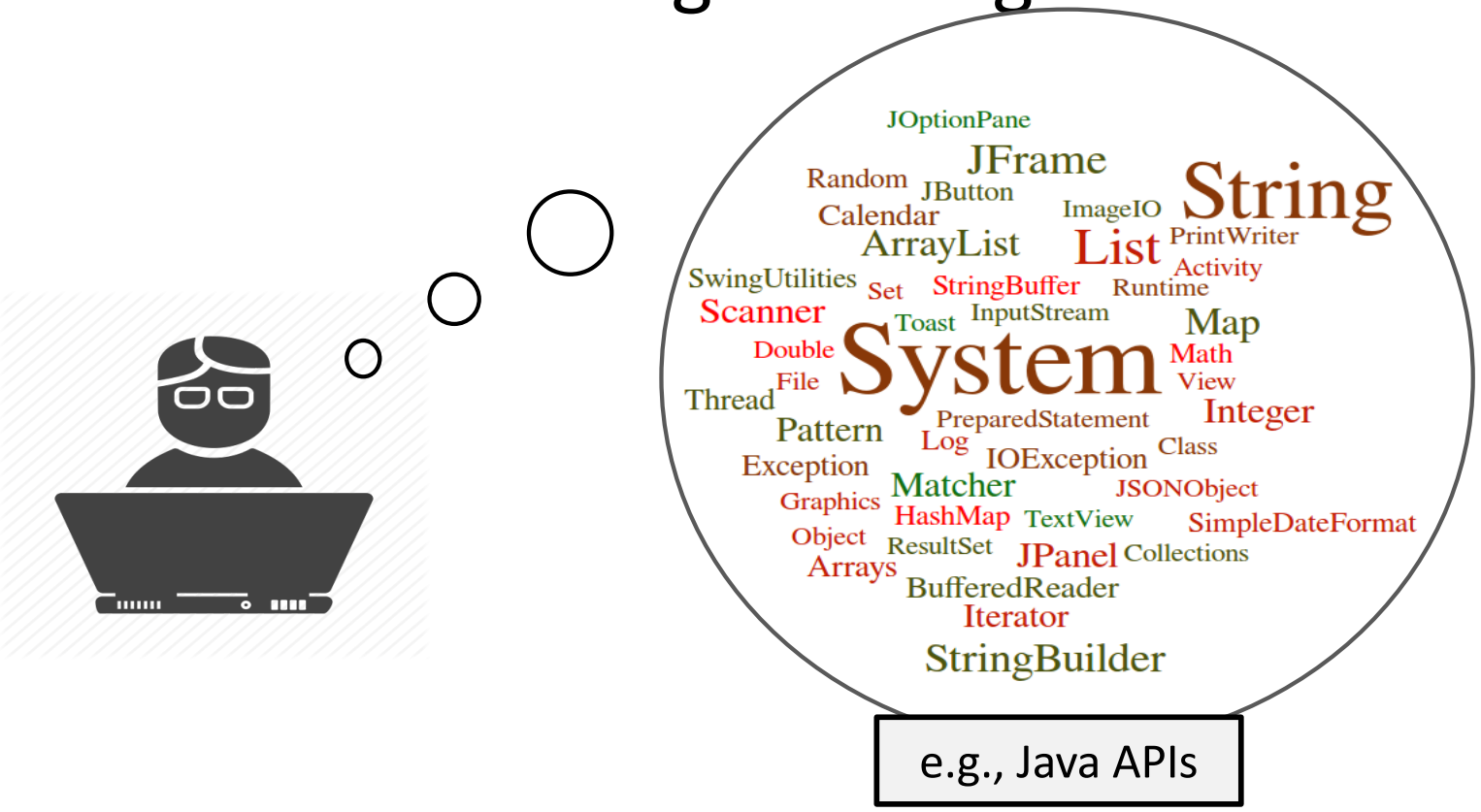
Are Code Examples on an Online Q&A Forum Reliable?

A Study of API Misuse on Stack Overflow

Tianyi Zhang[1], Ganesha Upadhyaya[2], Anastasia Reinhart[3], Hridesh Rajan[2], Miryung Kim[1]

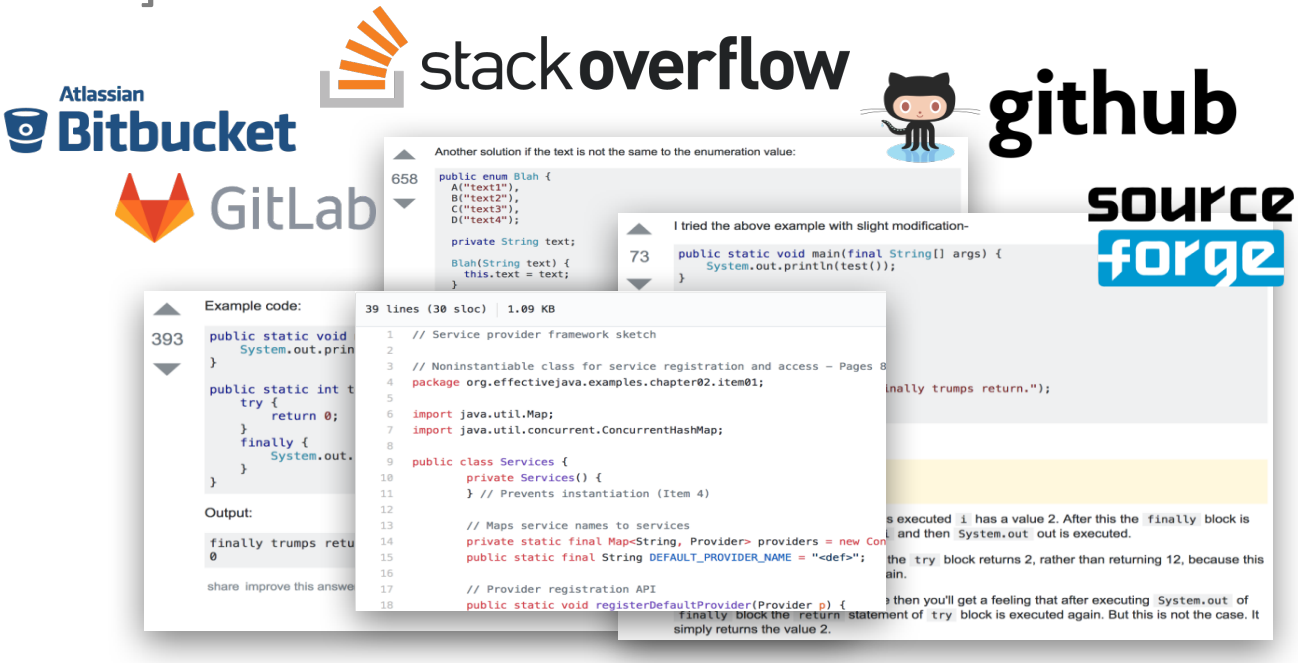
[1]University of California, Los Angeles
[2]Iowa State University
[3]George Fox University

Using APIs properly is a key challenge in Programming



The Status Quo of Learning APIs

Developers often search online for code examples to learn APIs [Sadowski et al. 2016]



The Limitation of Online Code Examples

- Programmers can only inspect a handful of search results. [Brandt et al. 2009, Starke et al. 2009, Duala-Ekoko and Robillard 2012]
- Individual code examples may suffer from
  - insecure coding practices [Fischer et al. 2017]
  - unchecked obsolete usage [Zhou and Walker 2016]
  - low readability [Treude and Robillard 2017]

"How do I write data to a file using `FileChannel`*?"*

2

You're probably interested in a `FileChannel`. `Channel`s were designed to perform bulk IO operations to and from `Buffer`s.

Ex:

```
FileChannel fileOut = new FileOutputStream(file).getChannel();
fileOut.write(ByteBuffer.wrap("Whatever you want to write".getBytes()));
```

share improve this answer

answered Apr 8 '12 at 19:39

Jeffrey

35.7k ● 7 ● 60 ● 111

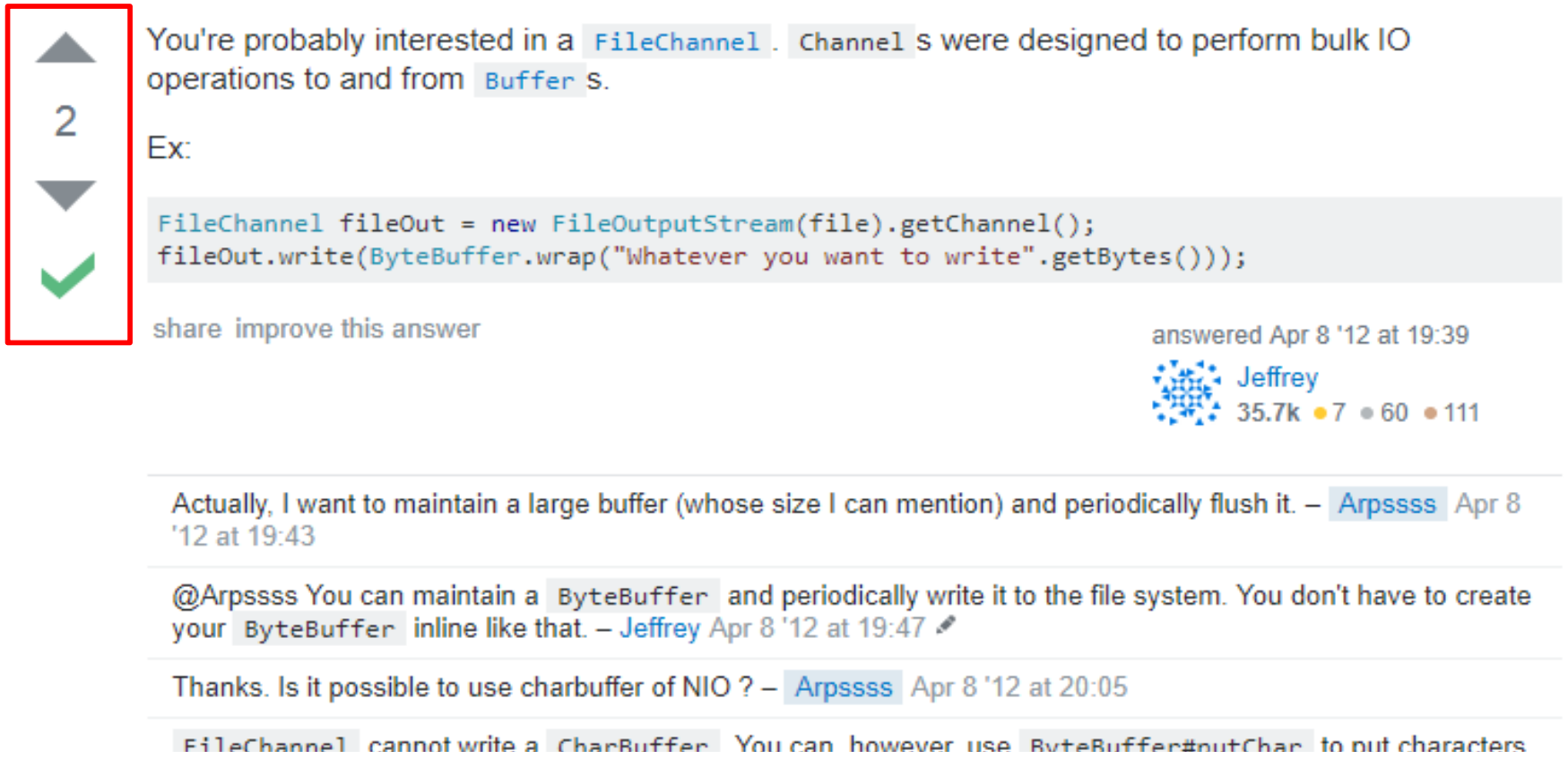
Actually, I want to maintain a large buffer (whose size I can mention) and periodically flush it. – Arpssss Apr 8 '12 at 19:43

@Arpssss You can maintain a `ByteBuffer` and periodically write it to the file system. You don't have to create your `ByteBuffer` inline like that. – Jeffrey Apr 8 '12 at 19:47

Thanks. Is it possible to use charbuffer of NIO ? – Arpssss Apr 8 '12 at 20:05

`FileChannel` cannot write a `CharBuffer`. You can, however, use `ByteBuffer#putChar` to put characters

"How do I write data to a file using `FileChannel`*?"*



2

You're probably interested in a `FileChannel`. `Channel`s were designed to perform bulk IO operations to and from `Buffer`s.

Ex:

```
FileChannel fileOut = new FileOutputStream(file).getChannel();
fileOut.write(ByteBuffer.wrap("Whatever you want to write".getBytes()));
```

share improve this answer

answered Apr 8 '12 at 19:39

Jeffrey

35.7k ● 7 ● 60 ● 111

Actually, I want to maintain a large buffer (whose size I can mention) and periodically flush it. – Arpssss Apr 8 '12 at 19:43

@Arpssss You can maintain a `ByteBuffer` and periodically write it to the file system. You don't have to create your `ByteBuffer` inline like that. – Jeffrey Apr 8 '12 at 19:47

Thanks. Is it possible to use charbuffer of NIO ? – Arpssss Apr 8 '12 at 20:05

`FileChannel` cannot write a `CharBuffer`. You can, however, use `ByteBuffer#putChar` to put characters

"How do I write data to a file using `FileChannel`*?"*

2

You're probably interested in a `FileChannel`. `Channel`s were designed to perform bulk IO operations to and from `Buffer`s.

Ex:

```
FileChannel fileOut = new FileOutputStream(file).getChannel();
fileOut.write(ByteBuffer.wrap("Whatever you want to write".getBytes()));
```

This example forgets to close the `FileChannel` object properly.

35.7k • 7 • 60 • 111

Actually, I want to maintain a large buffer (whose size I can mention) and periodically flush it. – Arpssss Apr 8 '12 at 19:43

@Arpssss You can maintain a `ByteBuffer` and periodically write it to the file system. You don't have to create your `ByteBuffer` inline like that. – Jeffrey Apr 8 '12 at 19:47

Thanks. Is it possible to use charbuffer of NIO ? – Arpssss Apr 8 '12 at 20:05

`FileChannel` cannot write a `CharBuffer`. You can, however, use `ByteBuffer#putChar` to put characters

"How do I write data to a file using `FileChannel`*?"*

Somewhat like this:

```
short[] payload = {1,2,3,4,5,6,7,8,9,0};
ByteBuffer myByteBuffer = ByteBuffer.allocate(20);
myByteBuffer.order(ByteOrder.LITTLE_ENDIAN);

ShortBuffer myShortBuffer = myByteBuffer.asShortBuffer();
myShortBuffer.put(payload);

FileChannel out = new FileOutputStream("sample.bin").getChannel();
out.write(myByteBuffer);
out.close();
```

"How do I write data to a file using `FileChannel`*?"*

Somewhat like this:

```
short[] payload = {1,2,3,4,5,6,7,8,9,0};
ByteBuffer myByteBuffer = ByteBuffer.allocate(20);
myByteBuffer.order(ByteOrder.LITTLE_ENDIAN);

ShortBuffer myShortBuffer = myByteBuffer.asShortBuffer();
myShortBuffer.put(payload);

FileChannel out = new FileOutputStream("sample.bin").getChannel();
out.write(myByteBuffer);
out.close();
```

This example forgets to handle potential exceptions such as IOException and FileNotFoundException.

Research Questions

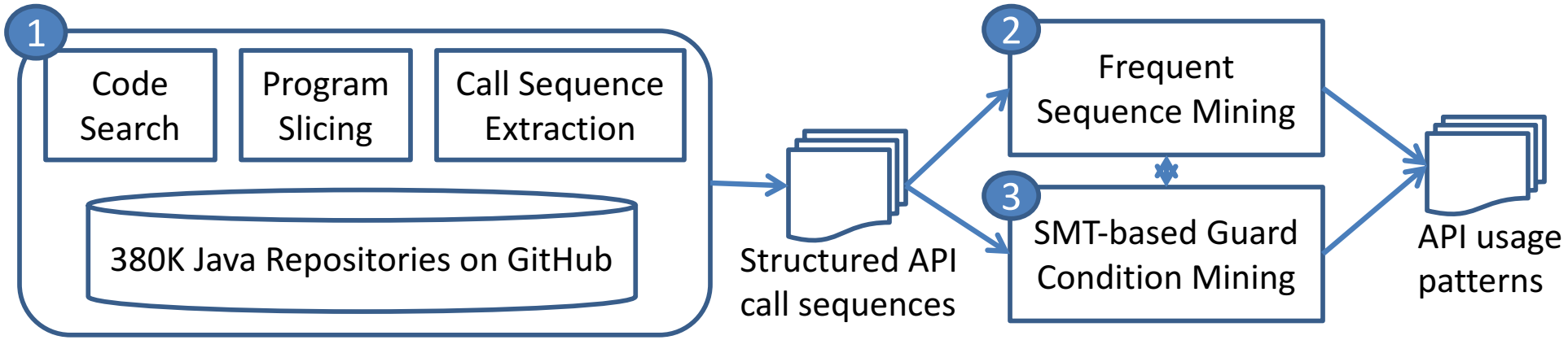
- RQ1. Is API misuse prevalent on Stack Overflow?
- RQ2. Are highly voted posts more reliable?
- RQ3. What are the characteristics of API misuse?

Outline

- Problem Statement
- API usage mining from 380K Java Projects on GitHub
- An Empirical Study of API Misuse on Stack Overflow

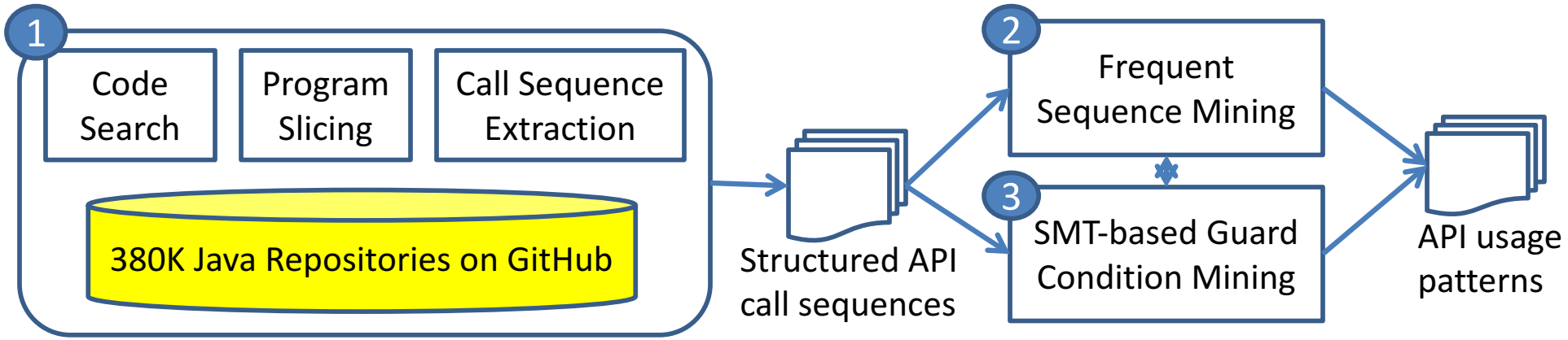
API Usage Mining from GitHub

- We contrast SO snippets with API usage patterns mined from 380K GitHub projects.



Insight 1: Mining a Large Code Corpus

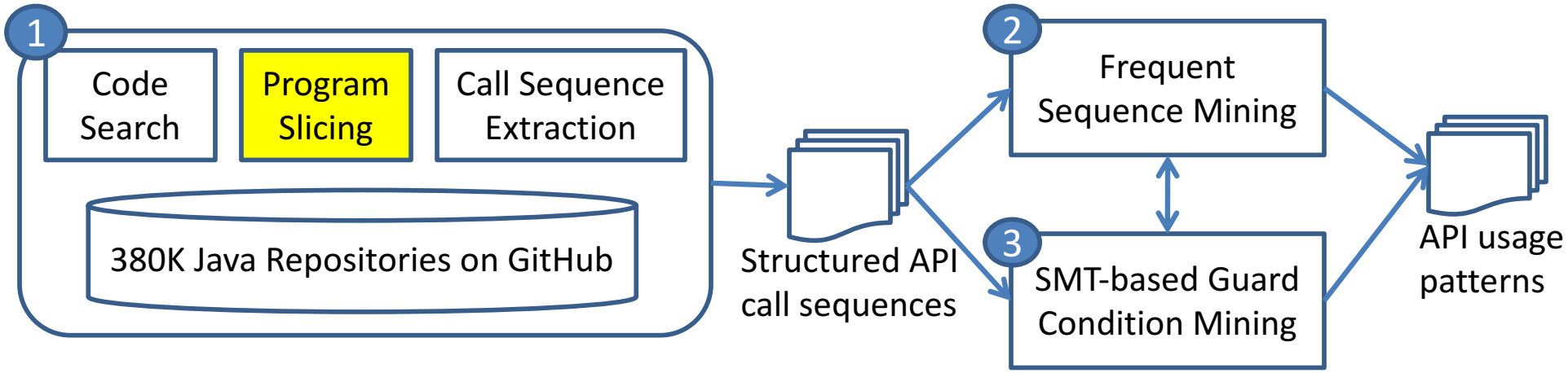
- Our code corpus includes 380K GitHub projects with at least 100 revisions and 2 contributors.



Dyer et al. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. ICSE 2013.

Insight 2: Removing Irrelevant Statements via Program Slicing

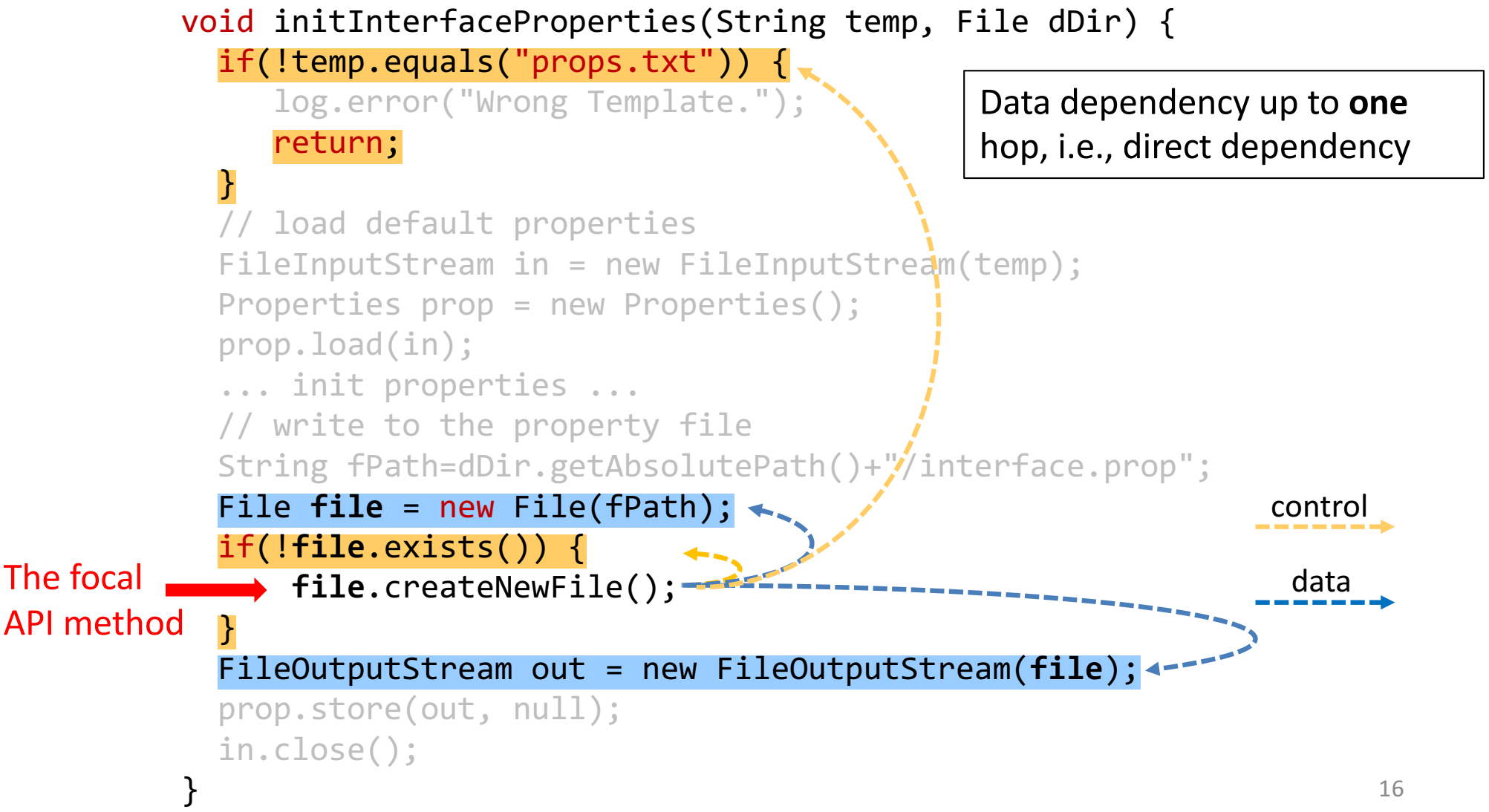
- We perform backward and forward slicing to identify data- and control-dependent statements to an API method of interest.

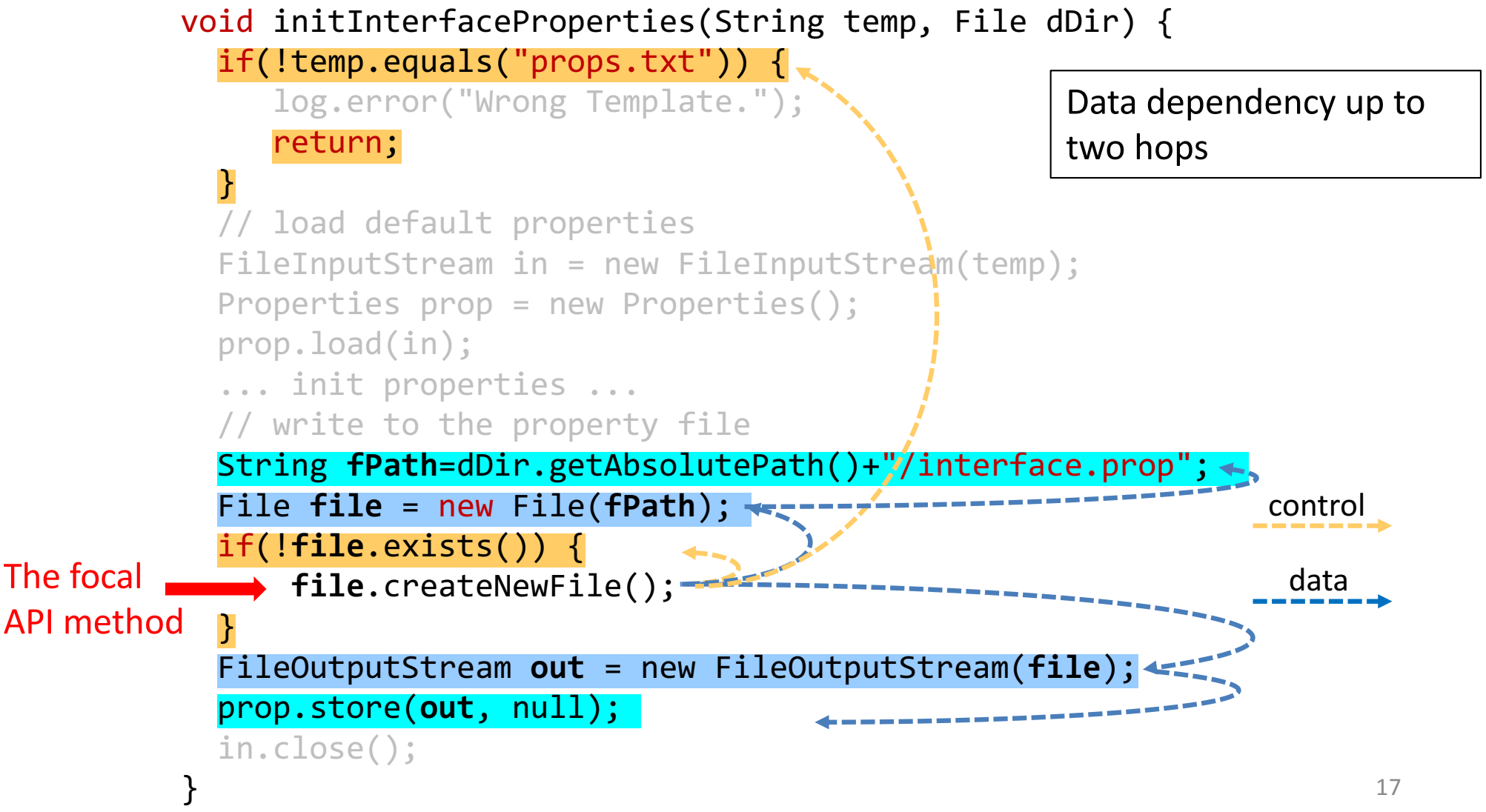


```
void initInterfaceProperties(String temp, File dDir) {
  if(!temp.equals("props.txt")) {
     log.error("Wrong Template.");
     return;
  }
  // load default properties
  FileInputStream in = new FileInputStream(temp);
  Properties prop = new Properties();
  prop.load(in);
  ... init properties ...
  // write to the property file
  String fPath=dDir.getAbsolutePath()+"/interface.prop";
  File file = new File(fPath);
  if(!file.exists()) {
     file.createNewFile();
  }
  FileOutputStream out = new FileOutputStream(file);
  prop.store(out, null);
  in.close();
}
```

GitHub example of `File.createNewFile`

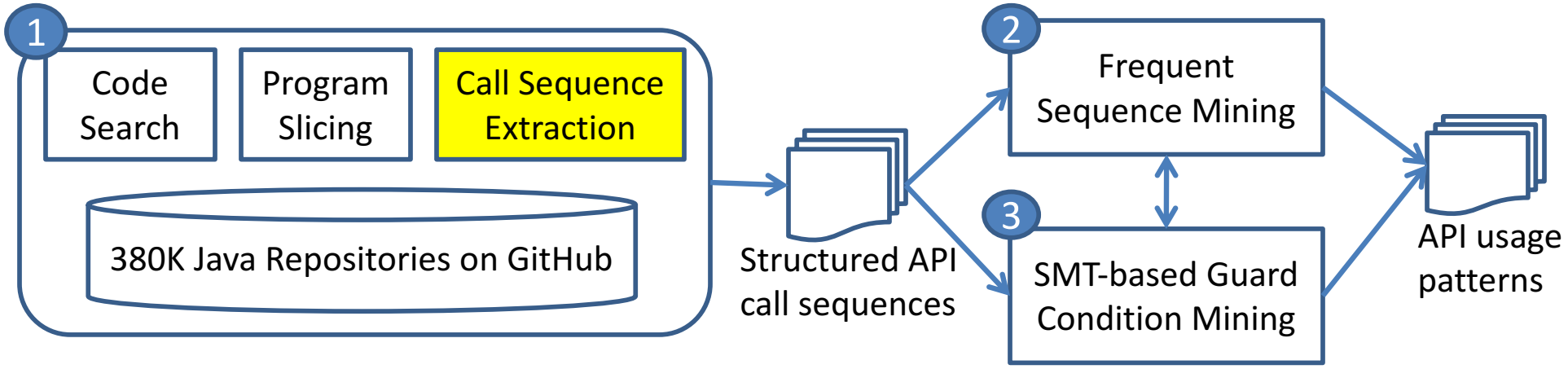
The focal API method


void initInterfaceProperties(String temp, File dDir) {
if(!temp.equals("props.txt")) {
log.error("Wrong Template.");
return;
}
// load default properties
FileInputStream in = new FileInputStream(temp);
Properties prop = new Properties();
prop.load(in);
... init properties ...
// write to the property file
String fPath=dDir.getAbsolutePath()+"/interface.prop";
File file = new File(fPath);
if(!file.exists()) {
file.createNewFile();
}
FileOutputStream out = new FileOutputStream(file);
prop.store(out, null);
in.close();
}
Data dependency up to one hop, i.e., direct dependency
control
data
The focal API method


void initInterfaceProperties(String temp, File dDir) {
if(!temp.equals("props.txt")) {
log.error("Wrong Template.");
return;
}
// load default properties
FileInputStream in = new FileInputStream(temp);
Properties prop = new Properties();
prop.load(in);
... init properties ...
// write to the property file
String fPath=dDir.getAbsolutePath()+"/interface.prop";
File file = new File(fPath);
if(!file.exists()) {
file.createNewFile();
}
FileOutputStream out = new FileOutputStream(file);
prop.store(out, null);
in.close();
}
Data dependency up to two hops
The focal API method
control
data

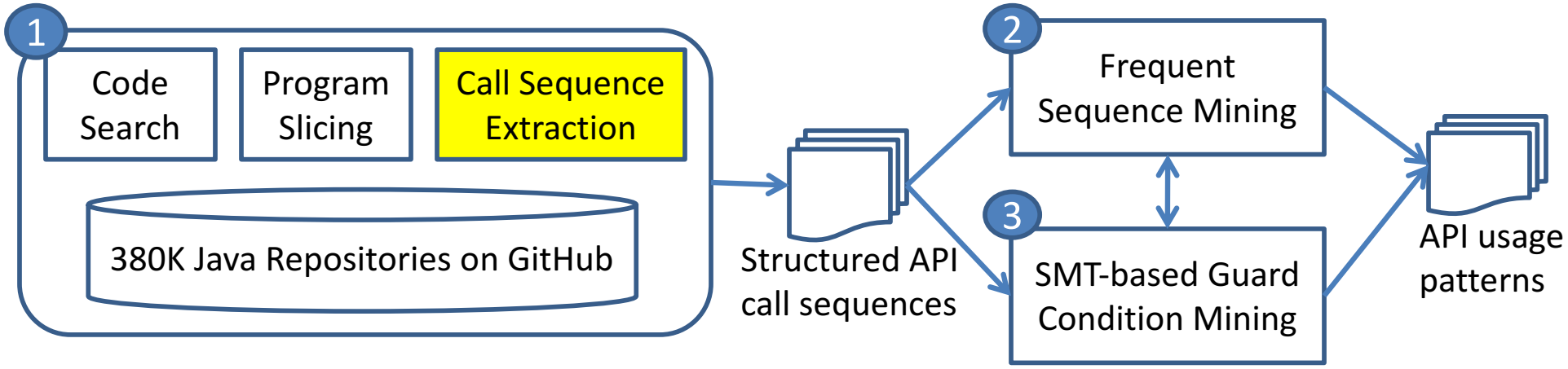
Insight 3: Capture Semantics Info in API Usage

- It is important to capture the temporal ordering, enclosing control structures, and appropriate guard conditions of API calls.



Insight 3: Capture Semantics Info in API Usage

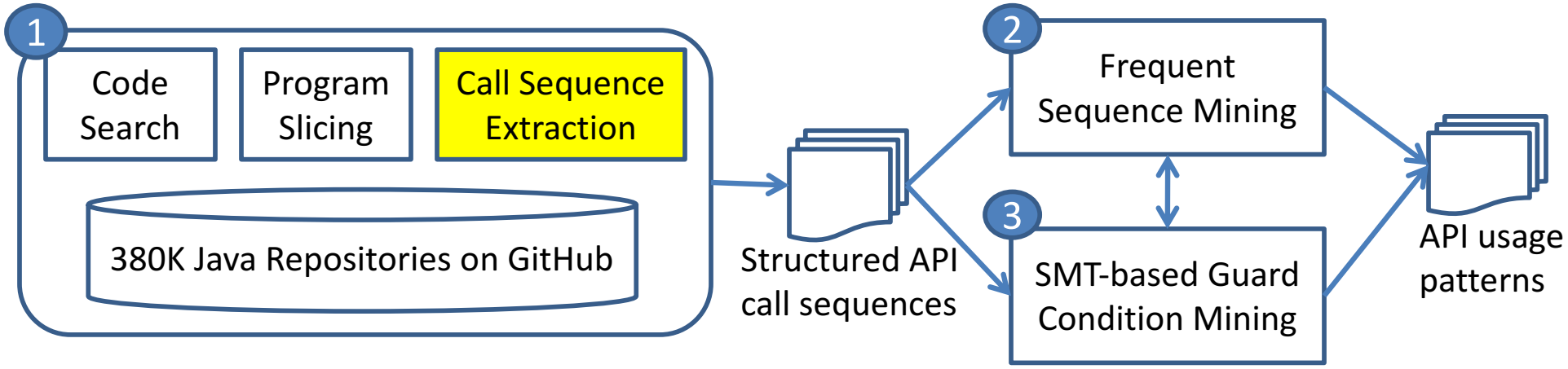
- It is important to capture the temporal ordering, enclosing control structures, and appropriate guard conditions of API calls.




```
new File (String); try {; new FileInputStream(File)@arg0.exists(); } catch (IOException) {; }
```

Insight 3: Capture Semantics Info in API Usage

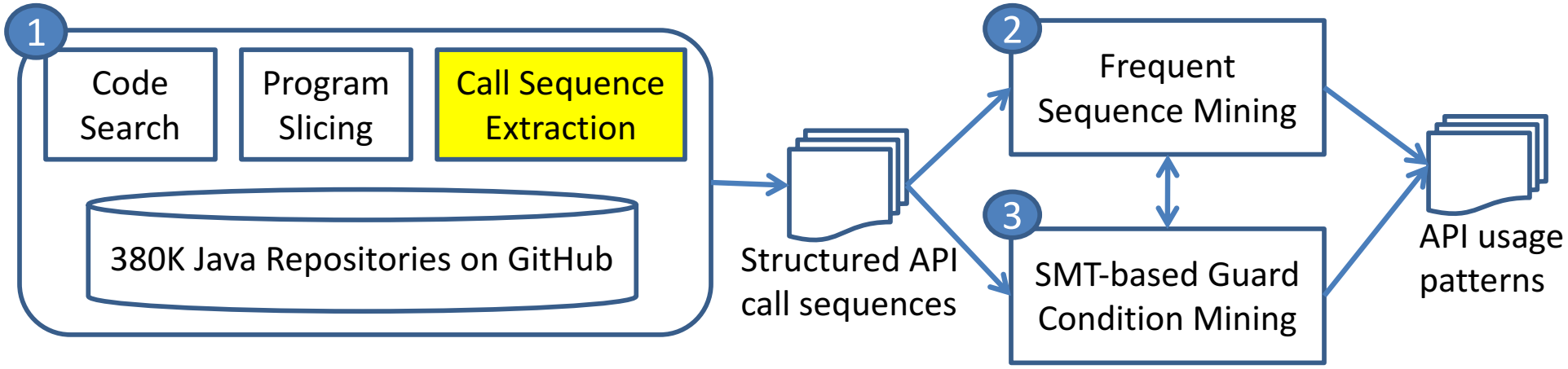
- It is important to capture the temporal ordering, enclosing control structures, and appropriate guard conditions of API calls.




```
new File (String); try {; new FileInputStream(File)@arg0.exists(); } catch (IOException) {; }
```

Insight 3: Capture Semantics Info in API Usage

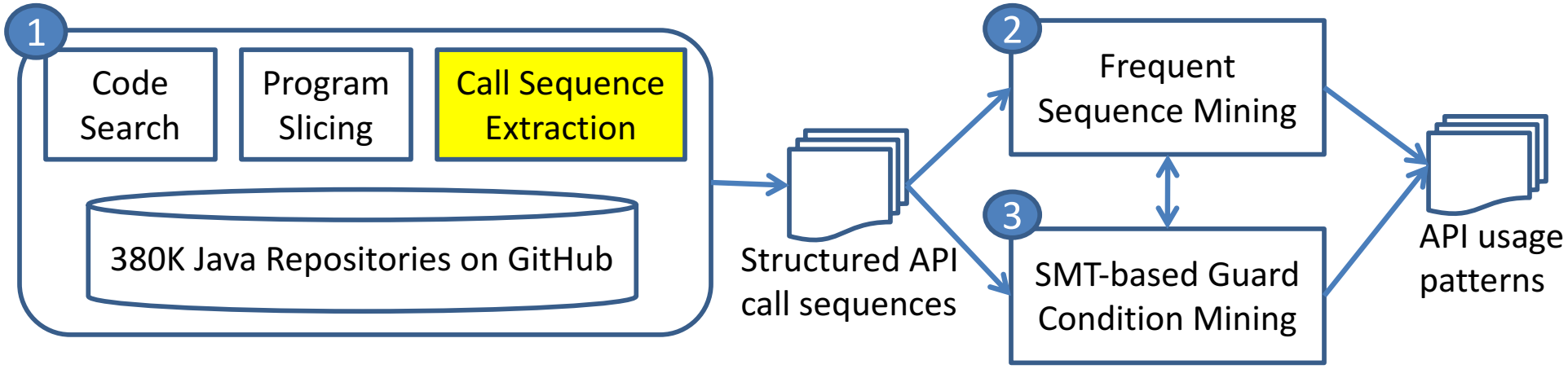
- It is important to capture the temporal ordering, enclosing control structures, and appropriate guard conditions of API calls.




```
new File (String); try {; new FileInputStream(File)@arg0.exists(); } catch (IOException) {; }
```

Insight 3: Capture Semantics Info in API Usage

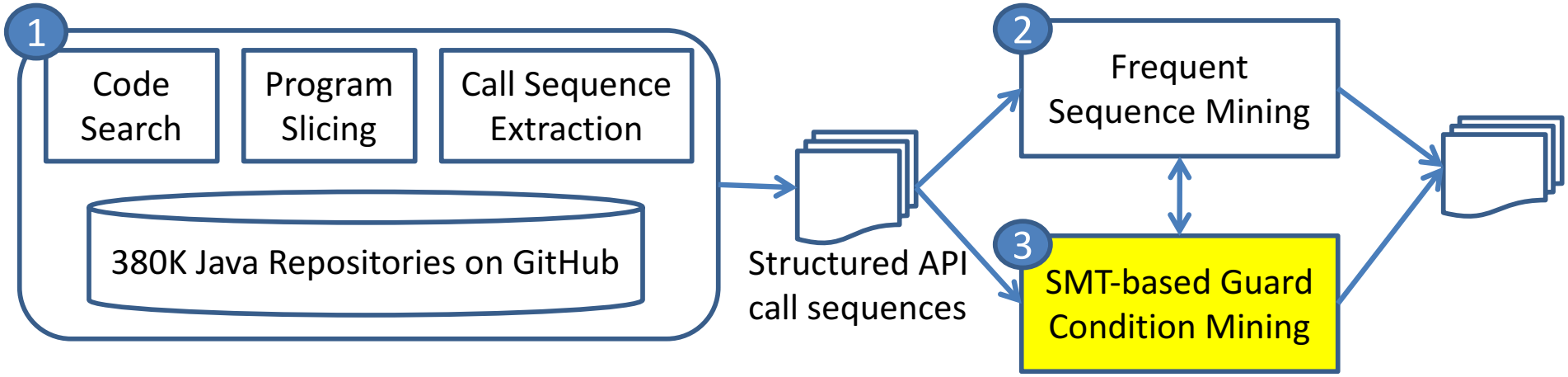
- It is important to capture the temporal ordering, enclosing control structures, and appropriate guard conditions of API calls.




```
new File (String); try {; new FileInputStream(File)@arg0.exists(); } catch (IOException) {; }
```

Insight 4: Variations in Guard Conditions

- Guard conditions are canonicalized and grouped based on logical equivalence.

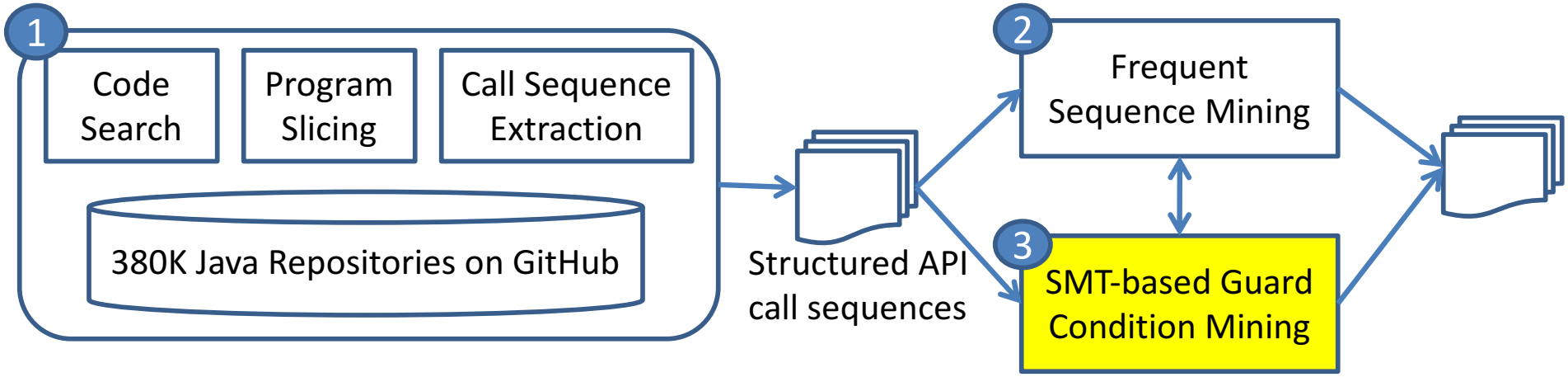


Two equivalent guard conditions for `String.substring`:

`arg0>=0 && arg0<=rcv.length()` ⇔ `arg0>-1 && arg0<rcv.length()+1`

Insight 4: Variations in Guard Conditions

- Guard conditions are canonicalized and grouped based on logical equivalence.

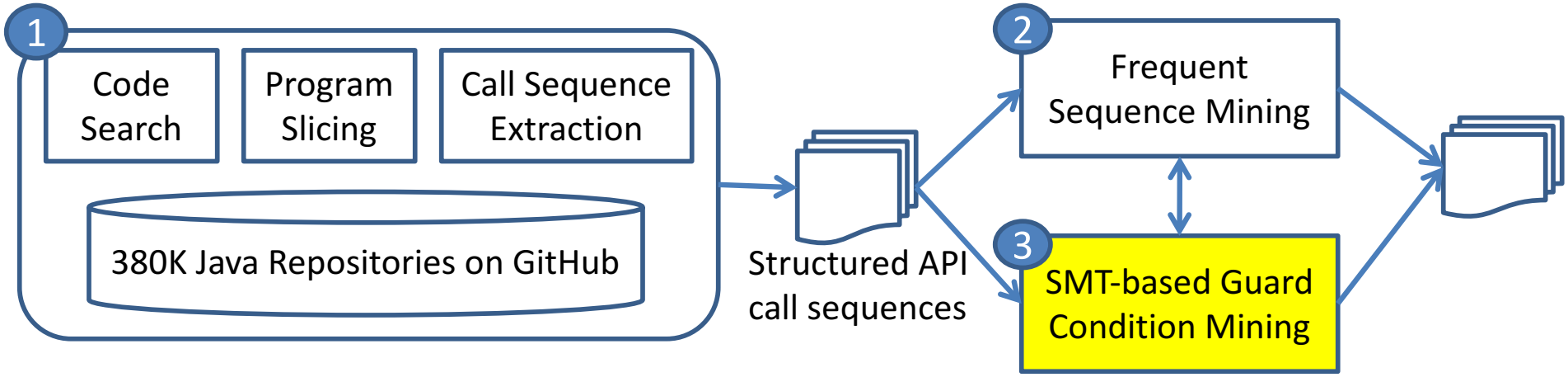


Two equivalent guard conditions for `String.substring`:

`arg0>=0 && arg0<=rcv.length()` ⇔ `arg0>-1 && arg0<rcv.length()+1`

Insight 4: Variations in Guard Conditions

- Guard conditions are canonicalized and grouped based on logical equivalence.



Two equivalent guard conditions for `String.substring`:

`arg0>=0 && arg0<=rcv.length()` ⇔ `arg0>-1 && arg0<rcv.length()+1`

Insight 4: Variations in Guard Conditions

- We use Z3 to prove the logic equivalence of guard conditions.

```
if (start>=0 && start<=s.length()) {
    s.substring(start);
}
```

```
p : arg0>=0 && arg0<=rcv.length()
```

```
if (i>-1 && i<log.length()+1) {
    log.substring(i);
}
```

```
q : arg0>-1 && arg0<rcv.length()+1
```

- $p \Leftrightarrow q$ is valid iff. $\neg((\neg p \vee q) \wedge (p \vee \neg q))$ is UNSAT.

Outline

- Problem Statement
- API usage mining from 380K Java Projects on GitHub
- An Empirical Study of API Misuse on Stack Overflow

Data Set: API Usage Patterns

- We learn 245 API usage patterns of 100 Java API methods.
- We manually inspect these patterns.
- 180 patterns can be confirmed by online documentation.

API method	Pattern
ArrayList.list	loop {; get(int)@arg0<rcv.size(); }
Scanner.nextLine	loop {; nextLine()@rcv.hasNextLine(); }
SQLiteDatabase.query	query(…)@true; close()@true
TypedArray.getString	getString(int)@true; recycle()@true
FileChannel.write	try {; write(ByteBuffer)@true; }; catch(IOException) {; }

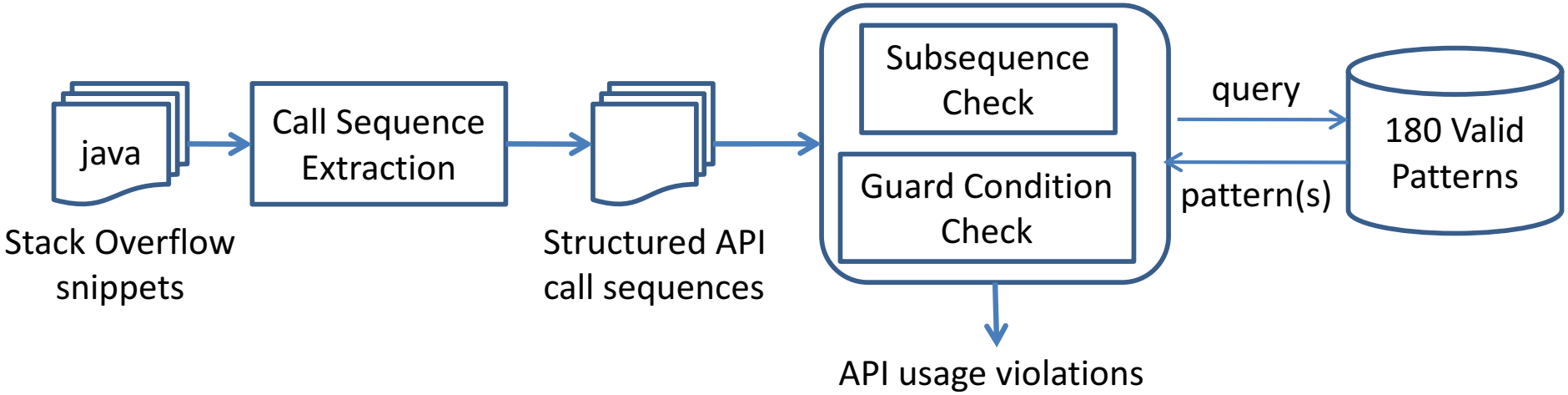
Data Set: SO Posts

- We extract code snippets in the markdown `<code>` from the SO posts with the `Java` tag.
- We parse and analyze these snippets using a partial program parsing and type resolution technique, JavaBaker.
- We find 217,818 SO posts with code snippets that use the 100 Java API methods in our study scope.

Siddharth Subramanian, Laura Inozemtseva, and Reid Holmes. Live API Documentation. ICSE 2014.

Method of API Misuse Detection

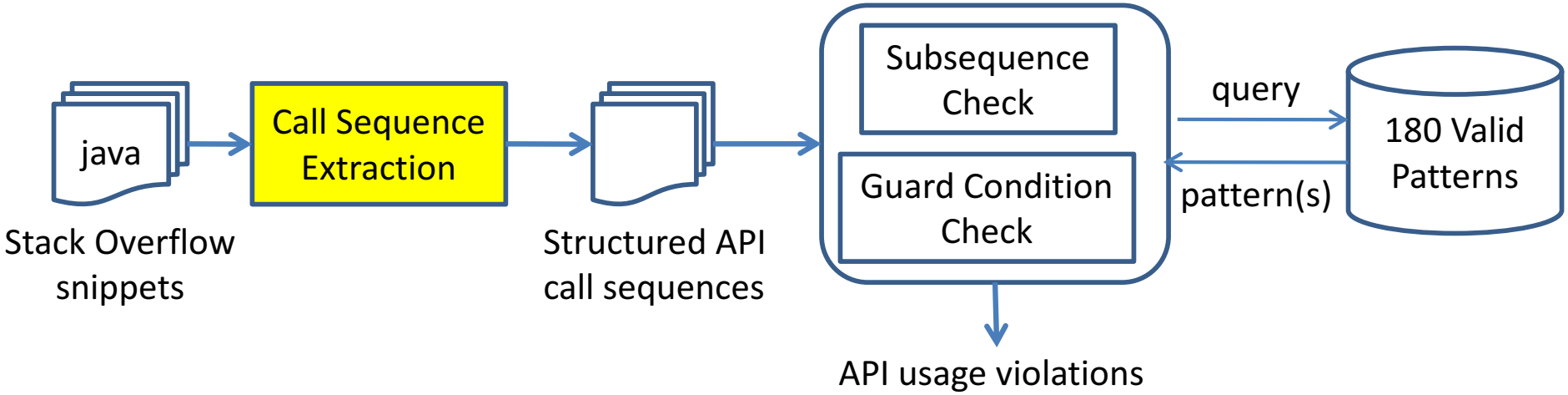
- We examine 220K SO posts with 180 confirmed patterns.



Dataset: http://web.cs.ucla.edu/~tianyi.zhang/examplecheck.html

Method of API Misuse Detection

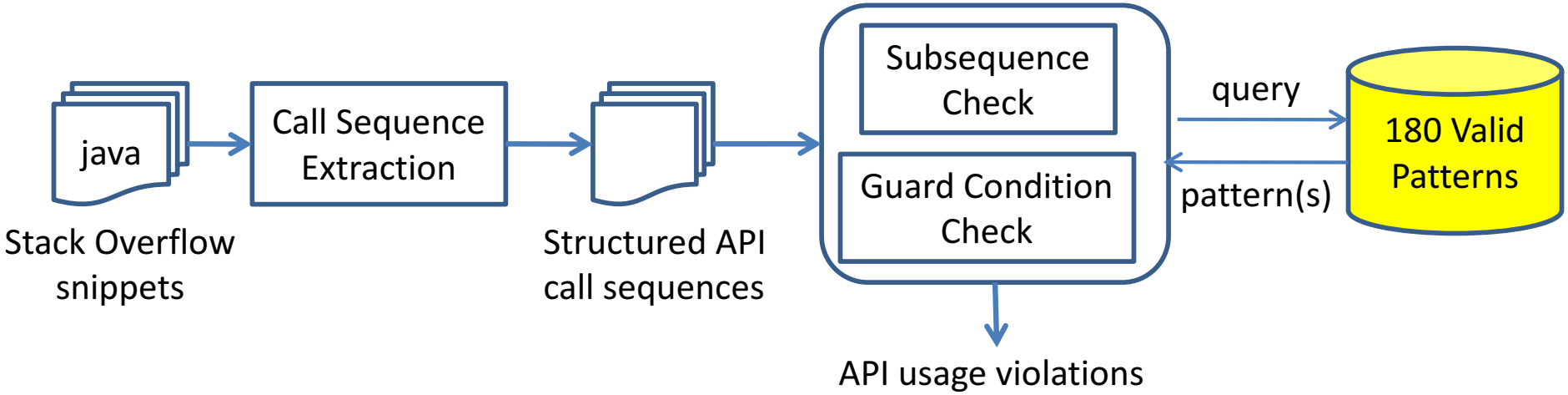
- We examine 220K SO posts with 180 confirmed patterns.



Dataset: http://web.cs.ucla.edu/~tianyi.zhang/examplecheck.html

Method of API Misuse Detection

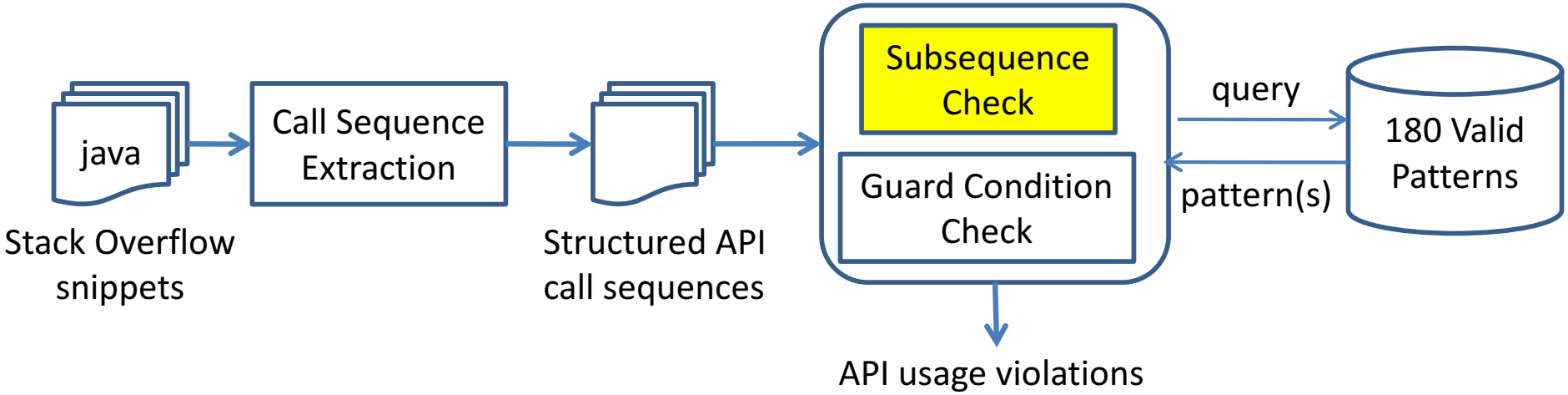
- We examine 220K SO posts with 180 confirmed patterns.



Dataset: http://web.cs.ucla.edu/~tianyi.zhang/examplecheck.html

Method of API Misuse Detection

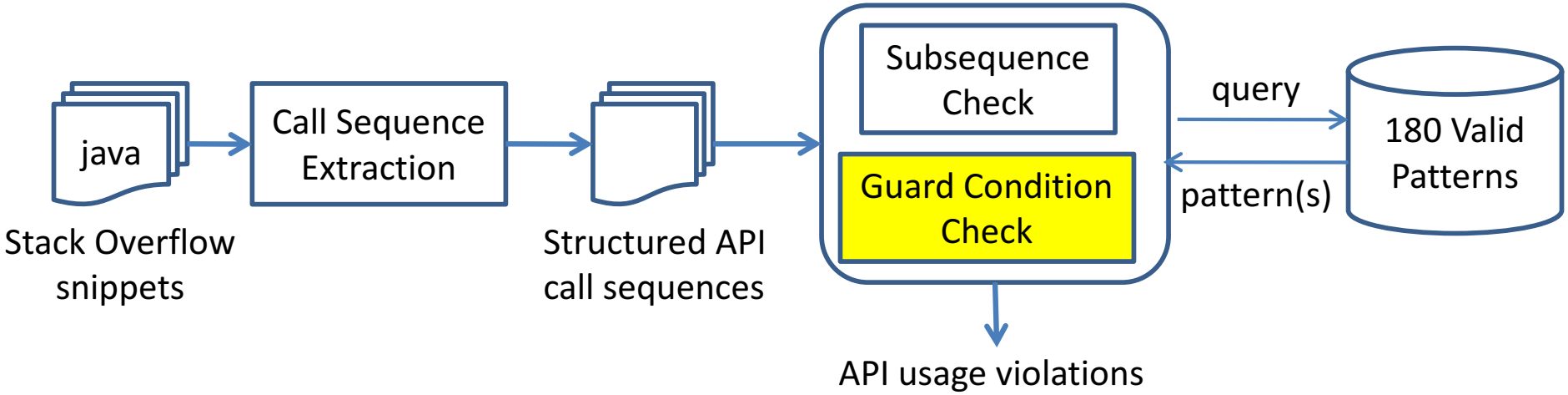
- We examine 220K SO posts with 180 confirmed patterns.



Dataset: http://web.cs.ucla.edu/~tianyi.zhang/examplecheck.html

Method of API Misuse Detection

- We examine 220K SO posts with 180 confirmed patterns.



Dataset: http://web.cs.ucla.edu/~tianyi.zhang/examplecheck.html

RQ1. Is API Misuse Prevalent on Stack Overflow?

- 31% of SO posts contain API usage violations.
- Two authors independently inspected 400 SO posts with reported API usage violations.
- 289 posts (72%) can negatively impact on production code.

```
public ArrayList get_user_by_id(String id) {
  ArrayList listUserInfo = new ArrayList();
  SQLiteDatabase db = this.getReadableDatabase();
  Cursor cursor = db.query(...);
  if (cursor != null) {
    while (cursor.moveToNext()) {
      UserInfo userInfo = new UserInfo();
      userInfo.setAppId(cursor.getString(cursor.getColumnIndex(
COLUMN_APP_ID)));
      // HERE YOU CAN MULTIPLE RECORD AND ADD TO LIST 11
      listUserInfo.add(userInfo);
    }
  }
  return listUserInfo;
}
```

A Cursor object is created but never released.

A code example of getting data from SQLite database in Android [Post ID 31531250]

```
public ArrayList get_user_by_id(String id) {
  ArrayList listUserInfo = new ArrayList();
  SQLiteDatabase db = this.getReadableDatabase();
  Cursor cursor = db.query(...);
  if (cursor != null) {
    while (cursor.moveToNext()) {
      UserInfo userInfo = new UserInfo();
      userInfo.setAppId(cursor.getString(cursor.getColumnIndex(
COLUMN_APP_ID)));
      // HERE YOU CAN MULTIPLE RECORD AND ADD TO LIST 11
      listUserInfo.add(userInfo);
    }
  }
  return listUserInfo;
}
```

A Cursor object is created but never released.

A code example of getting data from SQLite database in Android [Post ID 31531250]

```
public ArrayList get_user_by_id(String id) {
  ArrayList listUserInfo = new ArrayList();
  SQLiteDatabase db = this.getReadableDatabase();
  Cursor cursor = db.query(...);
  if (cursor != null) {
    while (cursor.moveToNext()) {
      UserInfo userInfo = new UserInfo();
      userInfo.setAppId(cursor.getString(cursor.getColumnIndex(
COLUMN_APP_ID)));
      // HERE YOU CAN MULTIPLE RECORD AND ADD TO LIST 11
      listUserInfo.add(userInfo);
    }
  }
  return listUserInfo;
}
```

A Cursor object is created but never released.

A code example of getting data from SQLite database in Android [Post ID 31531250]

```
public ArrayList get_user_by_id(String id) {
  ArrayList listUserInfo = new ArrayList();
  SQLiteDatabase db = this.getReadableDatabase();
  Cursor cursor = db.query(...);
  if (cursor != null) {
    while (cursor.moveToNext()) {
      UserInfo userInfo = new UserInfo();
      userInfo.setAppId(cursor.getString(cursor.getColumnIndex(
COLUMN_APP_ID)));
      // HERE YOU CAN MULTIPLE RECORD AND ADD TO LIST 11
      listUserInfo.add(userInfo);
    }
  }
  return listUserInfo;
}
```

A Cursor object is created but never released.

A code example of getting data from SQLite database in Android [Post ID 31531250]

```
public ArrayList get_user_by_id(String id) {
  ArrayList listUserInfo = new ArrayList();
  SQLiteDatabase db = this.getReadableDatabase();
  Cursor cursor = db.query(...);
  if (cursor != null) {
    while (cursor.moveToNext()) {
      UserInfo userInfo = new UserInfo();
      userInfo.setAppId(cursor.getString(cursor.getColumnIndex(
COLUMN_APP_ID)));
      // HERE YOU CAN MULTIPLE RECORD AND ADD TO LIST 11
      listUserInfo.add(userInfo);
    }
  }
  return listUserInfo;
}
```

A Cursor object is created but never released.

A code example of getting data from SQLite database in Android [Post ID 31531250]

RQ1. Is API Misuse Prevalent on Stack Overflow?

- Many SO snippets use hardcoded input for illustration.
- They may crash with real-world input data.

```
String text = "<img src=\"mysrc\" width=\"128\" height=\"92\"
border=\"0\" alt=\"alt\" /><p><strong>";
text = text.substring(text.indexOf("src=\""));
text = text.substring("src=\"".length());
text = text.substring(0, text.indexOf("\""));
System.out.println(text);
```

A code example of extracting the `src` field from a html string [Post ID 12742734]

RQ1. Is API Misuse Prevalent on Stack Overflow?

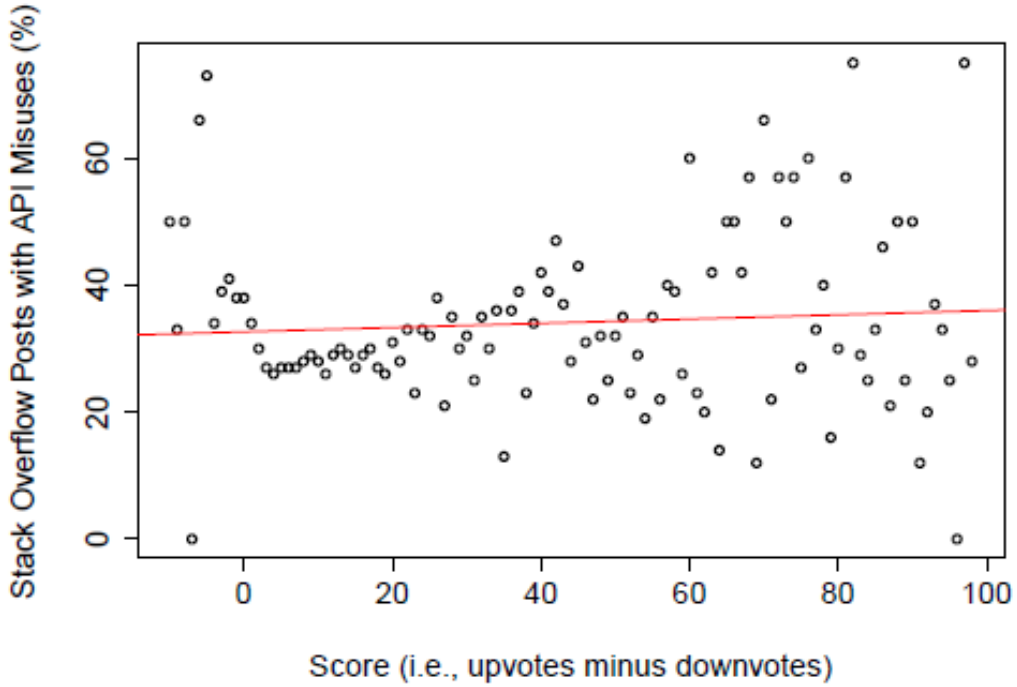
- Many SO snippets use hardcoded input for illustration.
- They may crash with real-world input data.

```
String text = "<img src=\"mysrc\" width=\"128\" height=\"92\"
border=\"0\" alt=\"alt\" /><p><strong>";
text = text.substring(text.indexOf("src=\""));
text = text.substring("src=\"".length());
text = text.substring(0, text.indexOf("\""));
System.out.println(text);
```

A code example of extracting the `src` field from a html string [Post ID 12742734]

RQ2. Are highly voted posts more reliable?

- Highly-voted posts are not necessarily more reliable in terms of correct API usage.



RQ3. What are characteristics of API misuse?

- API misuse is caused by three main reasons---*missing control structures, missing or incorrect order of API calls,* and *incorrect guard conditions.*

```
Cursor emails = db.query(Email.CONTENT_URI,...);
while (emails.moveToNext()) {
   String email = emails.getString(..);
}
emails.close();
```

Missing finally block [Post ID 31427468]

RQ3. What are characteristics of API misuse?

- API misuse is caused by three main reasons---***missing control structures***, *missing or incorrect order of API calls,* and *incorrect guard conditions*.

```
Cursor emails = db.query(Email.CONTENT_URI,...);
while (emails.moveToNext()) {
   String email = emails.getString(..);
}
emails.close();
```

Missing finally block [Post ID 31427468]

RQ3. What are characteristics of API misuse?

- API misuse is caused by three main reasons---*missing control structures,* ***missing or incorrect order of API calls,*** and *incorrect guard conditions*.

```
ByteBuffer bb = ByteBuffer.allocate(4);
bb.put(newArgb);
int i = bb.getInt();
```

Missing an API call, `bb.flip()` [Post ID 12100651]

RQ3. What are characteristics of API misuse?

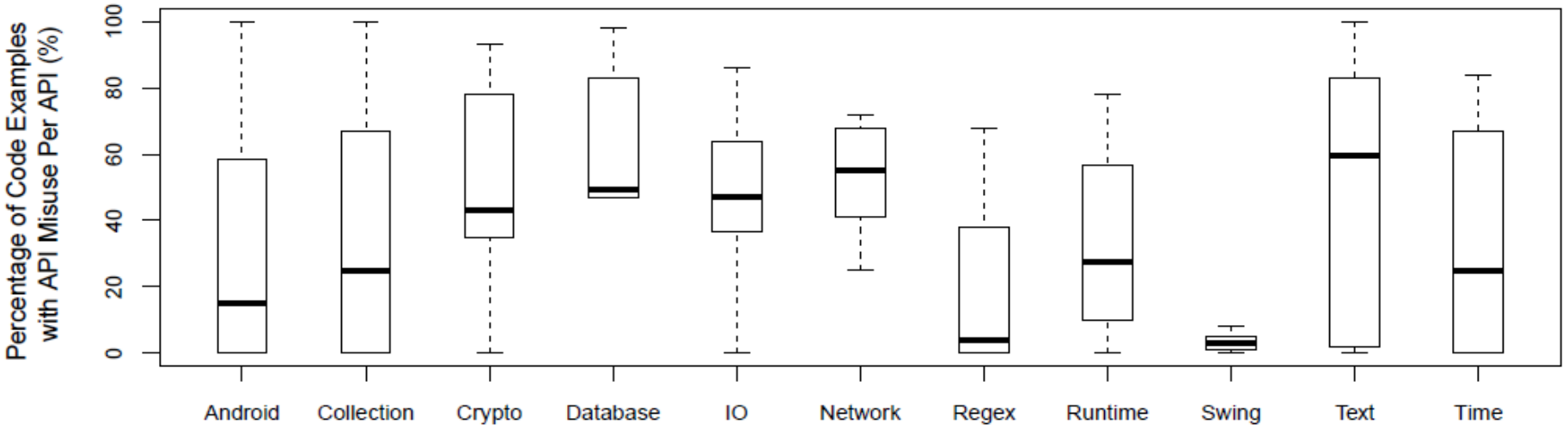
- API misuse is caused by three main reasons---*missing control structures, missing or incorrect order of API calls,* and ***incorrect guard conditions***.

```
TreeMap map = new TreeMap();
//OR SortedMap map = new
//TreeMap();
map.firstKey();
```

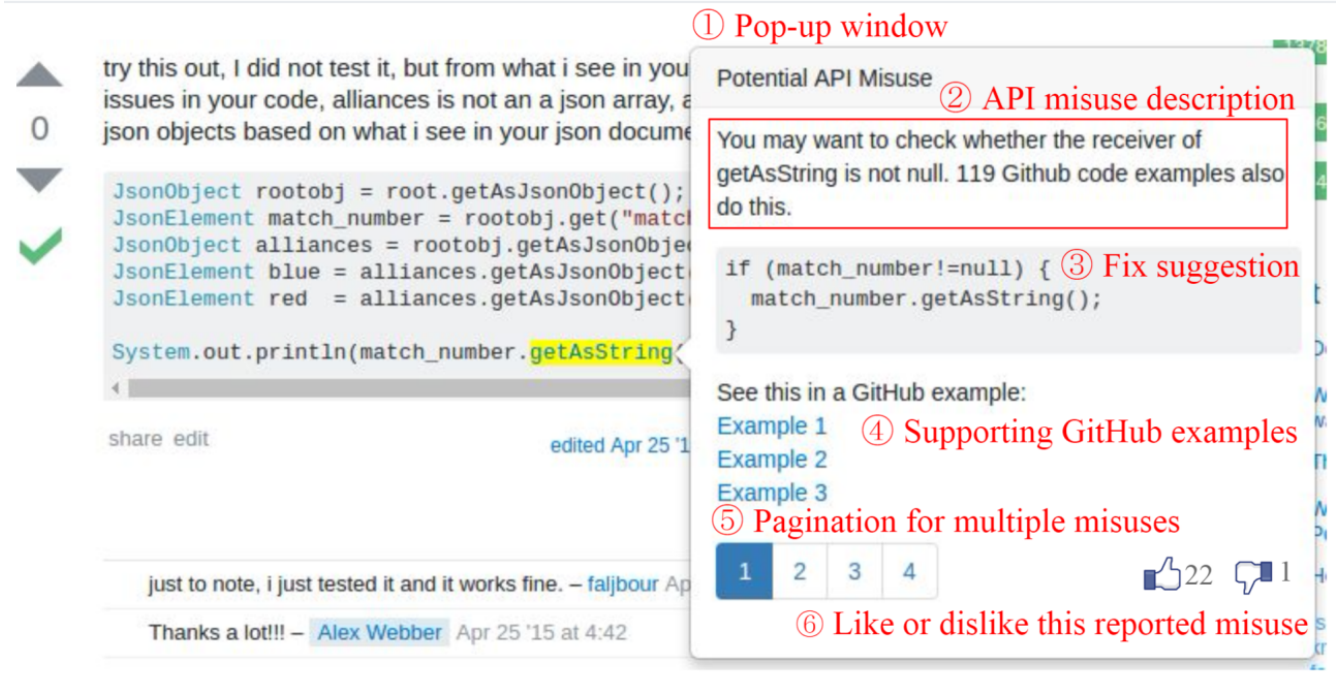
Incorrect guard condition, `!map.isEmpty()` [Post ID 21983867]

RQ3. What are characteristics of API misuse?

- Network, database, IO, crypto, string manipulation APIs are more likely to be misused.

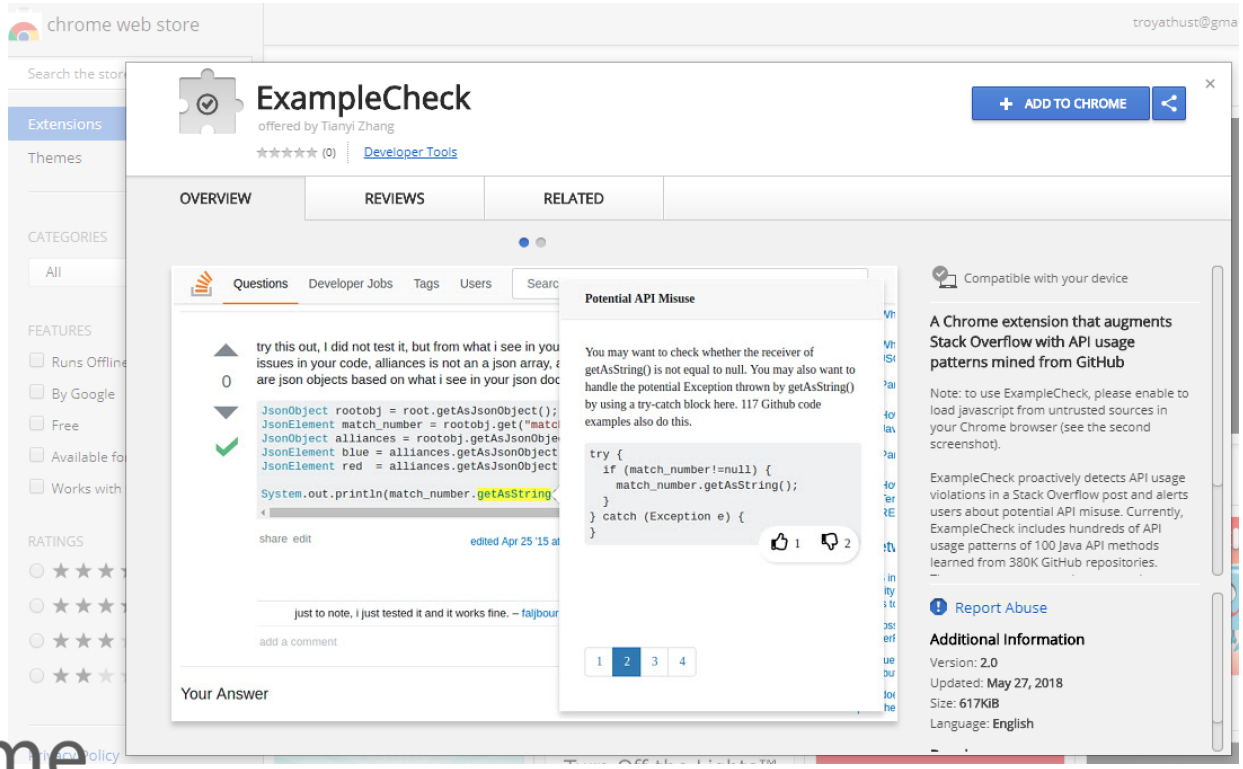


ExampleCheck: Checking API Misuse in Stack Overflow using Patterns Mined from GitHub



chrome

ExampleCheck: Checking API Misuse in Stack Overflow using Patterns Mined from GitHub



chrome

https://chrome.google.com/webstore/detail/examplecheck/amliempebckaiaklimcpopomlnklkioe

Examplore: Visualizing API Usage Examples at Scale [CHI 2018]

- Examplore visualizes API usage features in hundreds of code examples along with their statistical distribution in histograms.

Tool: http://examplore.cs.ucla.edu:3000/

Our Contributions

- An API usage mining technique that extracts patterns from over 380K GitHub projects
- A large-scale empirical study of API misuse in 220K SO posts
- A Chrome extension that augments Stack Overflow with API usage patterns mined from GitHub