Learning to Find Bugs in Programs and their Documentation

Andrew Habib

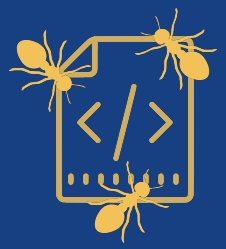
TU Darmstadt

software-lab.org

PhD Defense – December 14th, 2020

Software Bugs

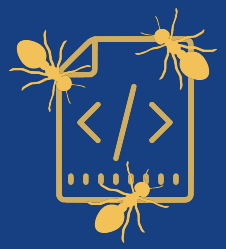
- **Bugs are everywhere**
  - 15-50 buggy LoC per 1,000 LoC[1]



1 Steve McConnell. Code Complete: A Practical Handbook of Software Construction

Software Bugs

- **Bugs are everywhere**
  - 15-50 buggy LoC per 1,000 LoC[1]



- **Software failures in 2018**[2]
  - Affect many people: ½ the world population
  - Costly: USD 1.7 trillions in assets

1 Steve McConnell. Code Complete: A Practical Handbook of Software Construction
2 The Software Fail Watch, 5th edition

Detecting Software Bugs

E.g.: Static analysis, dynamic analysis, model checking

Detecting Software Bugs

E.g.: Static analysis, dynamic analysis, model checking

- Far from being perfect
- Require high expertise
- Focus on traditional artifacts only
  E.g.: Source code, runtime information, predefined specifications

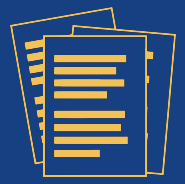
Detecting Software Bugs

E.g.: Static analysis, dynamic analysis, model checking

- Far from being perfect
- Require high expertise
- Focus on traditional artifacts only
  E.g.: Source code, runtime information, predefined specifications
- Ignore rich source of information: Documentation

Software Documentation

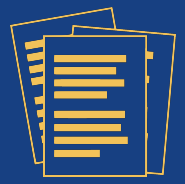
API documentation



- Developers intentions & assumptions
- Easy to understand by humans
- Abundance of information

Software Documentation

API documentation



- Developers intentions & assumptions
- Easy to understand by humans
- Abundance of information

Can we use software documentation to find bugs?

Using Documentation to Find Bugs

API Documentation Challenges

Using Documentation to Find Bugs

API Documentation Challenges

- Fuzzy natural language

Using Documentation to Find Bugs

API Documentation Challenges

- **Fuzzy natural language**
- **Code evolves rapidly, documentation:**
  - Missing information
  - Stale or outdated
  - Inconsistent

Using Documentation to Find Bugs

API Documentation Challenges

- **Fuzzy natural language**
- **Code evolves rapidly, documentation:**
  - Missing information
  - Stale or outdated
  - Inconsistent
- **Unclear documentation best practices**

API Documentation Issues Ex. (1)

E.g.: JDK format classes (`DateFormat`, `MessageFormat`, `NumberFormat`)

Bug report:[1]

> *"[...] Not being thread-safe is a significant limitation on a class (sic), with potentially dire results, and not documenting the classes as such is dangerous."*

1 https://bugs.java.com/view_bug.do?bug_id=4264153

API Documentation Issues Ex. (1)

E.g.: JDK format classes (`DateFormat`, `MessageFormat`, `NumberFormat`)

Bug report:[1]

> *"[...] Not being thread-safe is a significant limitation on a class (sic), with potentially dire results, and not documenting the classes as such is dangerous."*

Problem: Missing thread safety documentation

1 https://bugs.java.com/view_bug.do?bug_id=4264153

API Documentation Issues Ex. (1)

E.g.: JDK format classes (`DateFormat`, `MessageFormat`, `NumberFormat`)

Bug report:[1]

> *"[...] Not being thread-safe is a significant limitation on a class (sic), with potentially dire results, and not documenting the classes as such is dangerous."*

Problem: **Missing thread safety documentation**
Solution: **Document classes as <u>not</u> thread-safe**

1 https://bugs.java.com/view_bug.do?bug_id=4264153

API Documentation Issues Ex. (2)

E.g.: commons lang `StringUtils.unwrap`

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

[1] https://issues.apache.org/jira/browse/LANG-1475

API Documentation Issues Ex. (2)

E.g.: commons lang `StringUtils.unwrap`

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Invocation:[1]

```
unwrap("a", "a");   →   StringIndexOutOfBoundsException
```

1 https://issues.apache.org/jira/browse/LANG-1475

API Documentation Issues Ex. (2)

E.g.: commons lang `StringUtils.unwrap`

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Invocation:[1]

```
unwrap("a", "a");   →   StringIndexOutOfBoundsException
```

Problem: Unexpected (undocumented) exception

1 https://issues.apache.org/jira/browse/LANG-1475

API Documentation Issues Ex. (2)

E.g.: commons lang `StringUtils.unwrap`

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Invocation:[1]

```
unwrap("a", "a");   →   StringIndexOutOfBoundsException
```

Problem: Unexpected (undocumented) exception

Solution: Fix the corner case in the implementation

1 https://issues.apache.org/jira/browse/LANG-1475

PhD Thesis Statement

Automatically learning from programs and their documentation provides an effective means to prevent and detect software bugs.

PhD Thesis Statement

Automatically learning from programs and their documentation provides an effective means to prevent and detect software bugs.

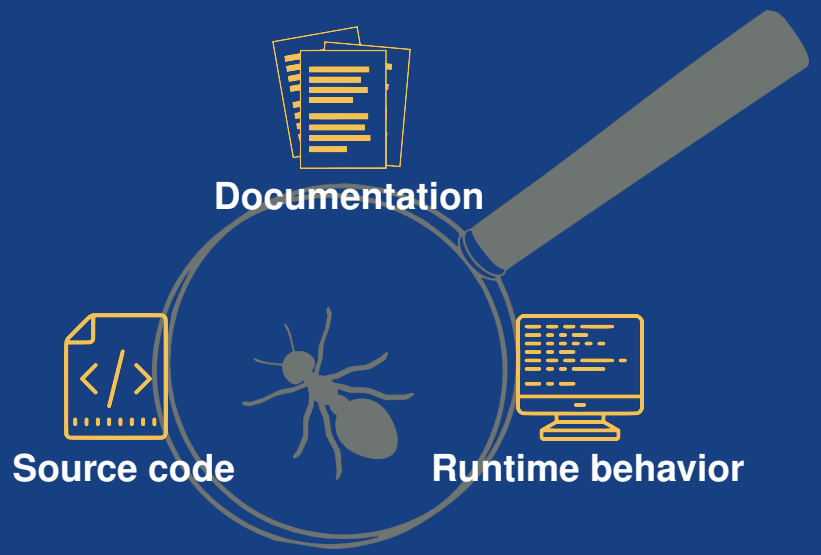
Documentation

Source code

Runtime behavior

PhD Thesis Statement

Automatically learning from programs and their documentation provides an effective means to prevent and detect software bugs.



PhD Thesis

Learning from Programs and their Documentation

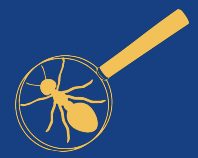
Source code

Documentation

PhD Thesis

Learning from Programs and their Documentation

Chapter 2
y of state-of-the-art tools



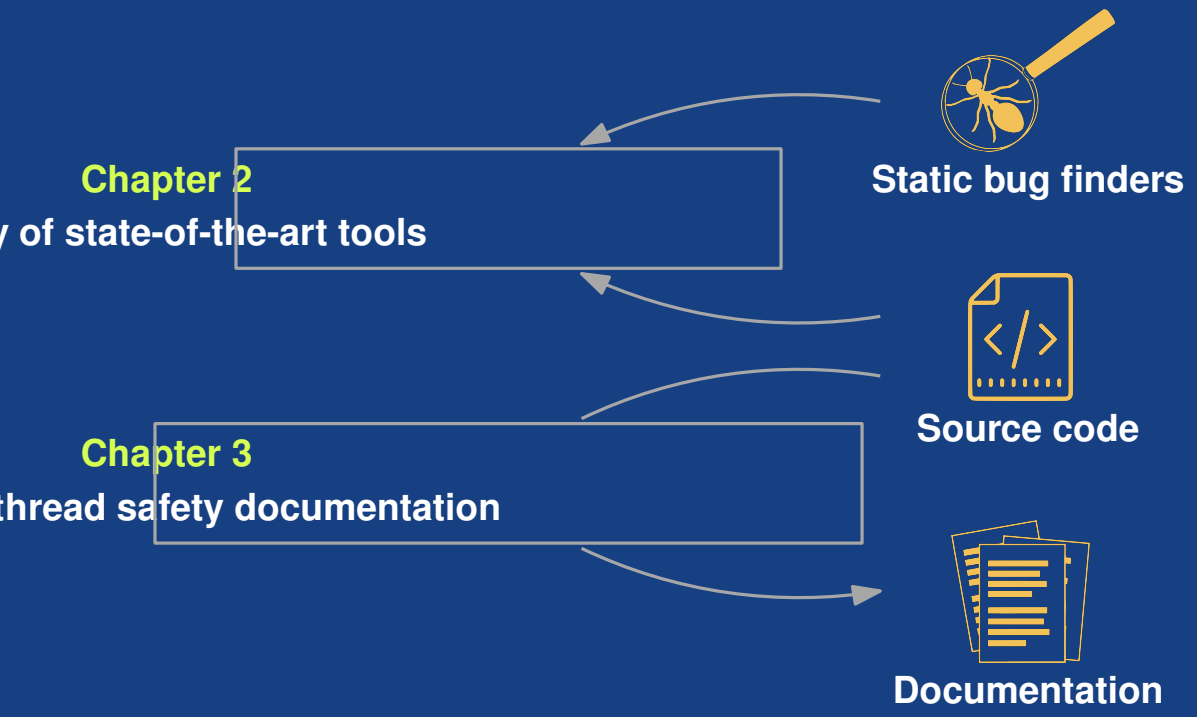
Static bug finders

Source code

Documentation

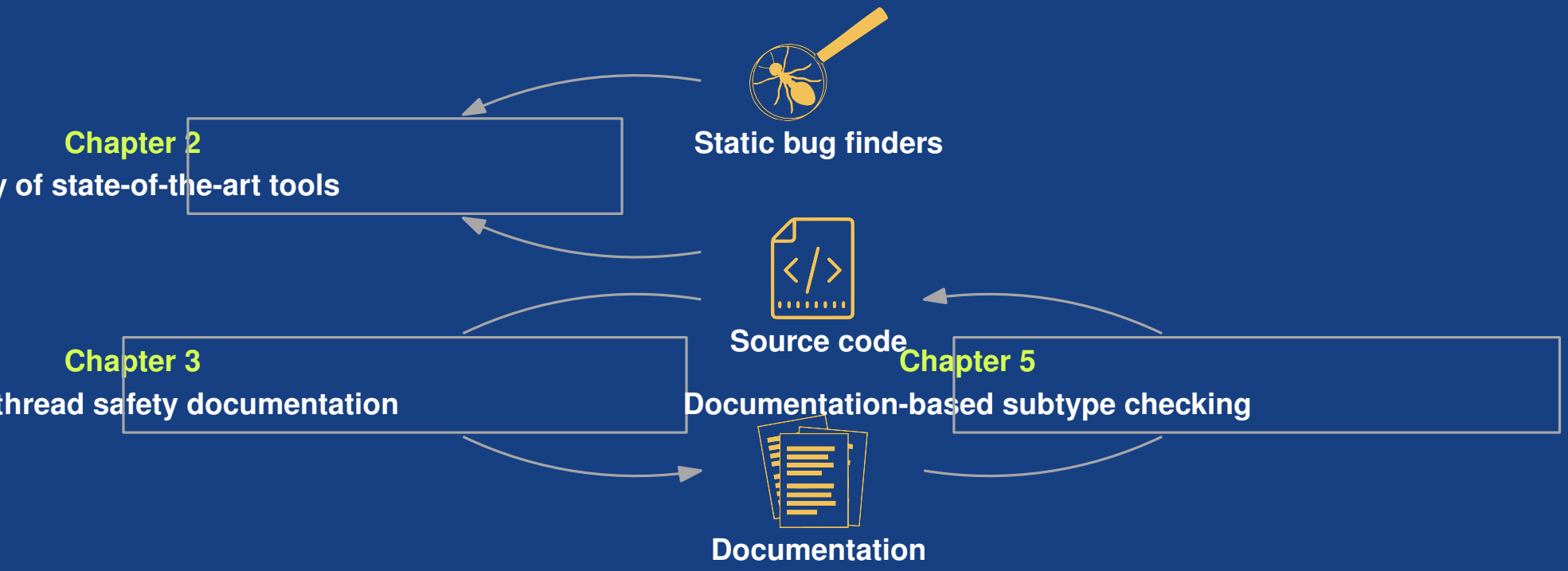
PhD Thesis

Learning from Programs and their Documentation



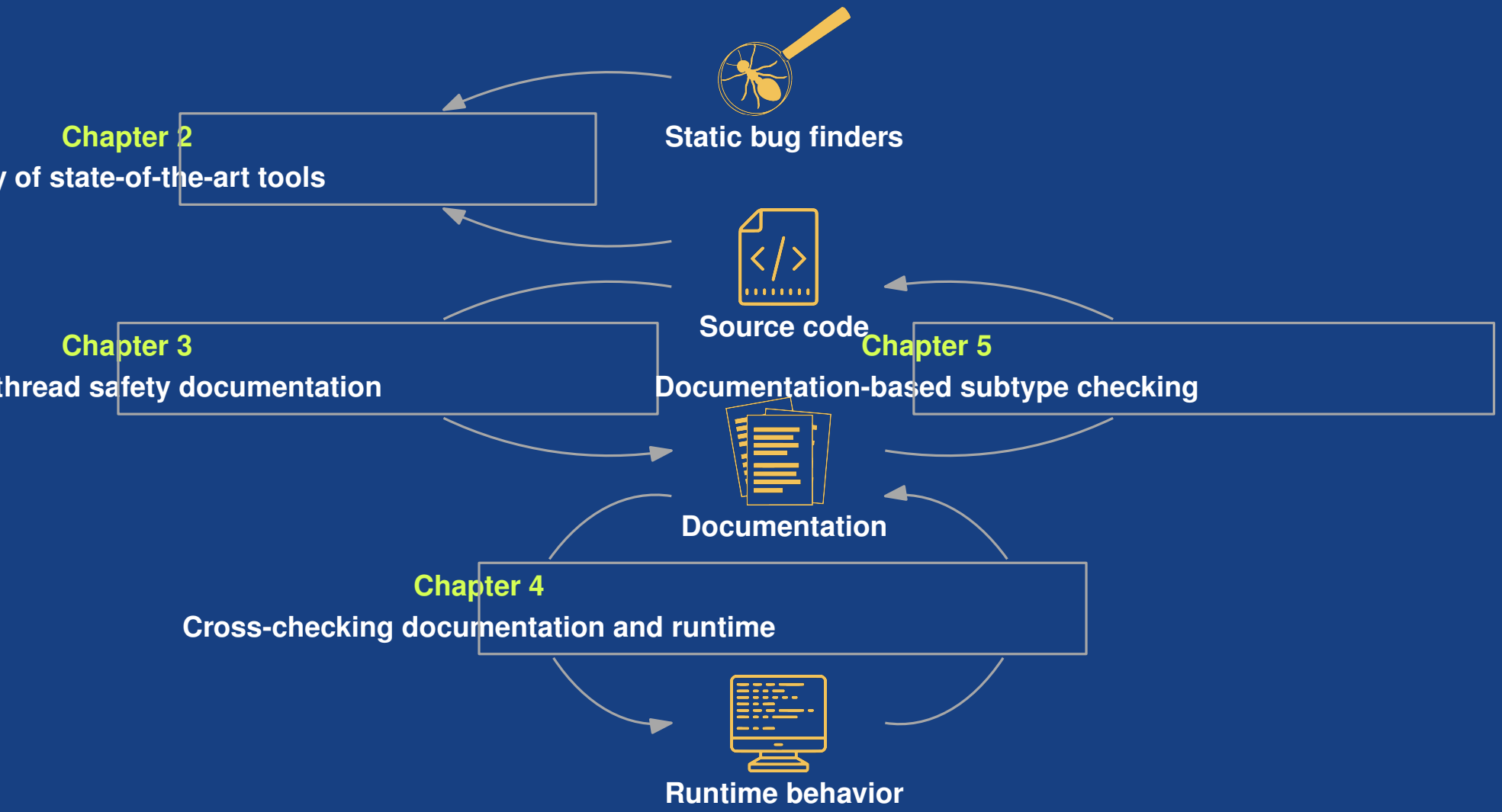
PhD Thesis

Learning from Programs and their Documentation



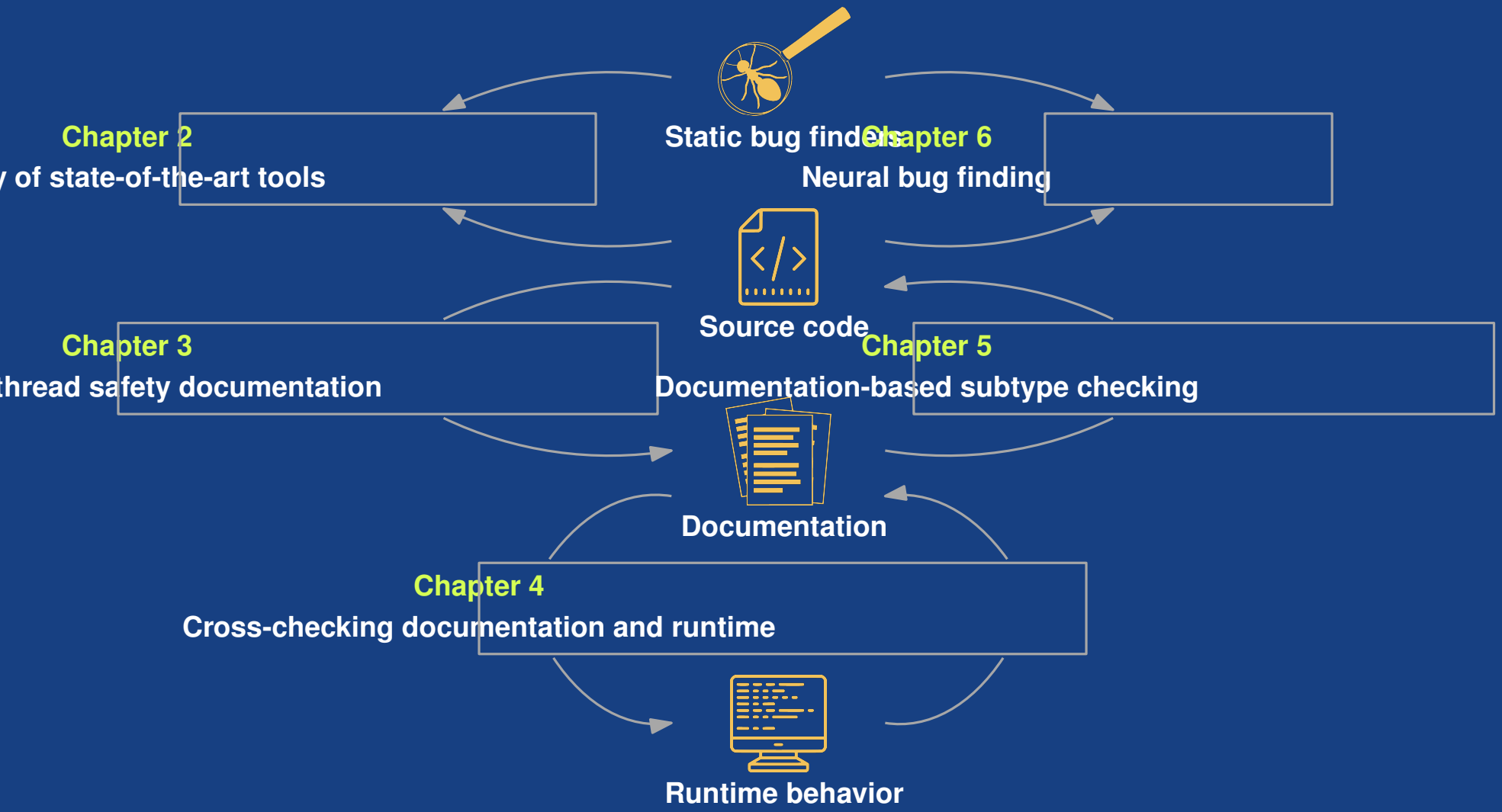
PhD Thesis

Learning from Programs and their Documentation



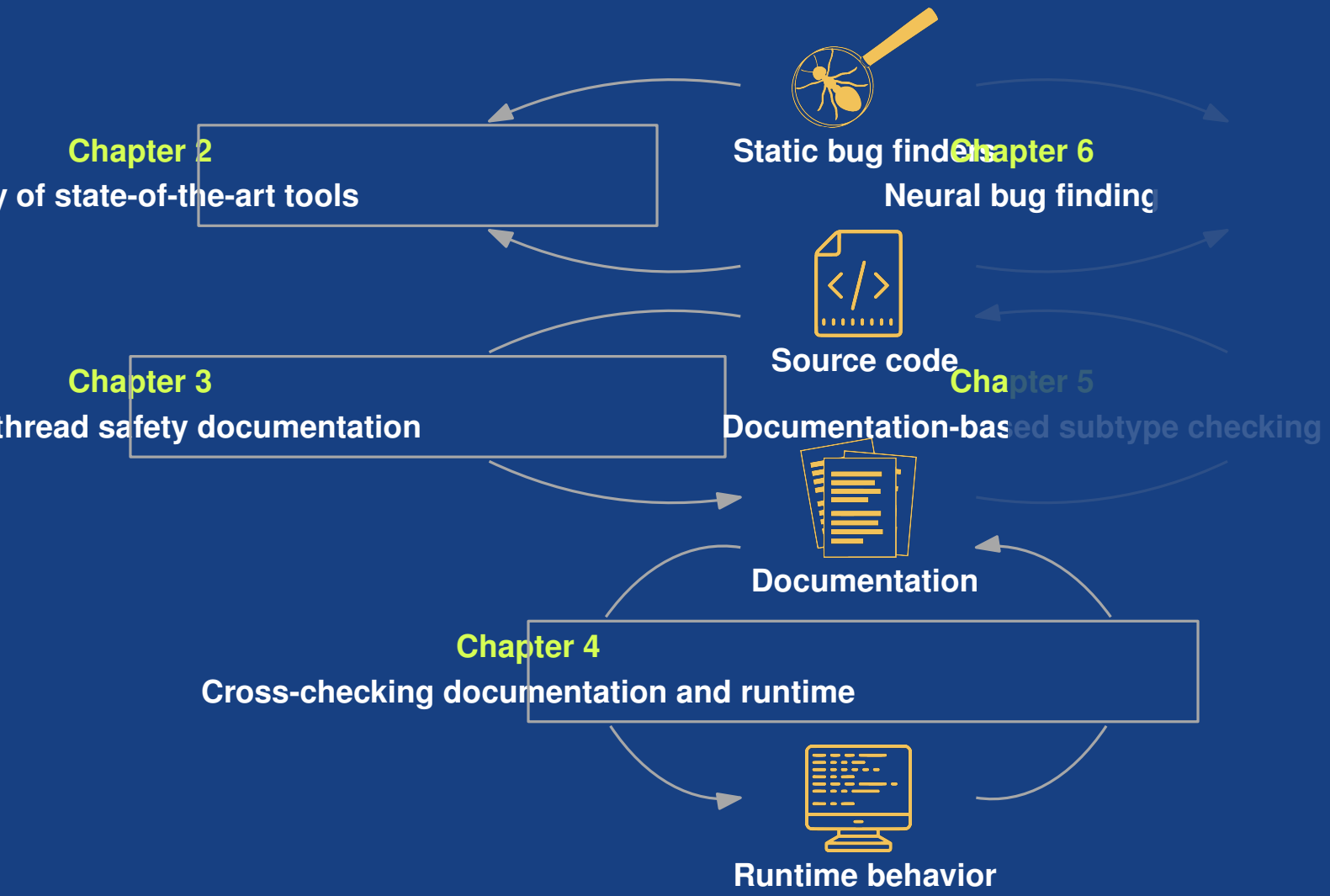
PhD Thesis

Learning from Programs and their Documentation



PhD Thesis

Learning from Programs and their Documentation



What is the state of static bug finding?

How Many of All Bugs Do We Find? A Study of Static Bug Detectors.
Andrew Habib and Michael Pradel. IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018
(Chapter 2)

Static Bug Detectors

Google
Error Prone

facebook
Infer

SpotBugs

Static Bug Detectors

Google
Error Prone

facebook
Infer

SpotBugs

How Many Bugs Do They Find?

Static Bug Detectors

Google
Error Prone

facebook
Infer

SpotBugs

~~How Many Bugs Do They Find?~~

Given a representative set of real-world bugs, how many of them do static bug detectors find?

Real-World Bugs: Defects4J

594 bugs from 15 popular Java projects

- Extended version of Defects4J data set[1]

1 Just et al., Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs. ISSTA'14

Real-World Bugs: Defects4J

594 bugs from 15 popular Java projects

- Extended version of Defects4J data set[1]

Why this set?

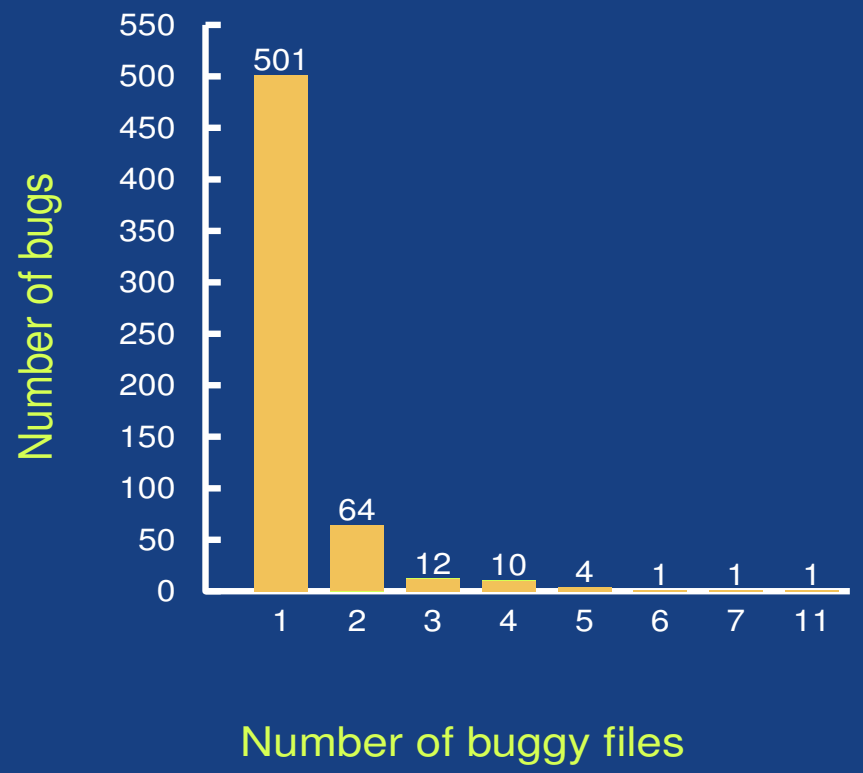
- Gathered independently
- Used in other bug-related studies[2]
- Contains real fixes by developers

1 Just et al., Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs. ISSTA'14
2 Just et al., 2014 (mutation testing); Shamshiri et al., 2015 (test generation); Pearson et al., 2017 (fault localization); Martinez et al., 2017 (program repair)

Real-World Bugs: Defects4J

594 bugs from 15 popular Java projects

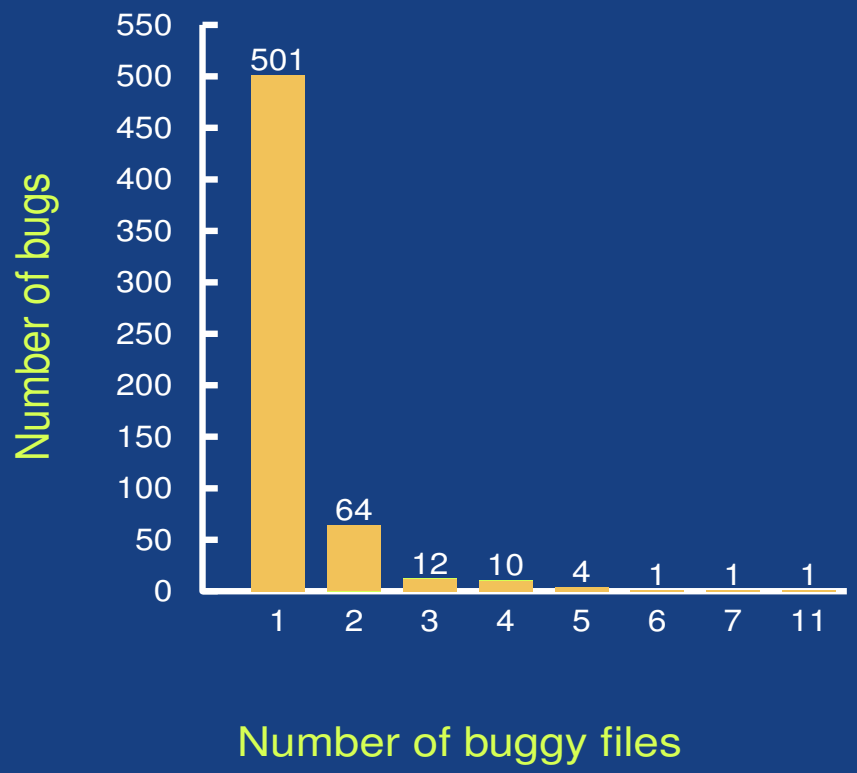
- Extended version of Defects4J data set

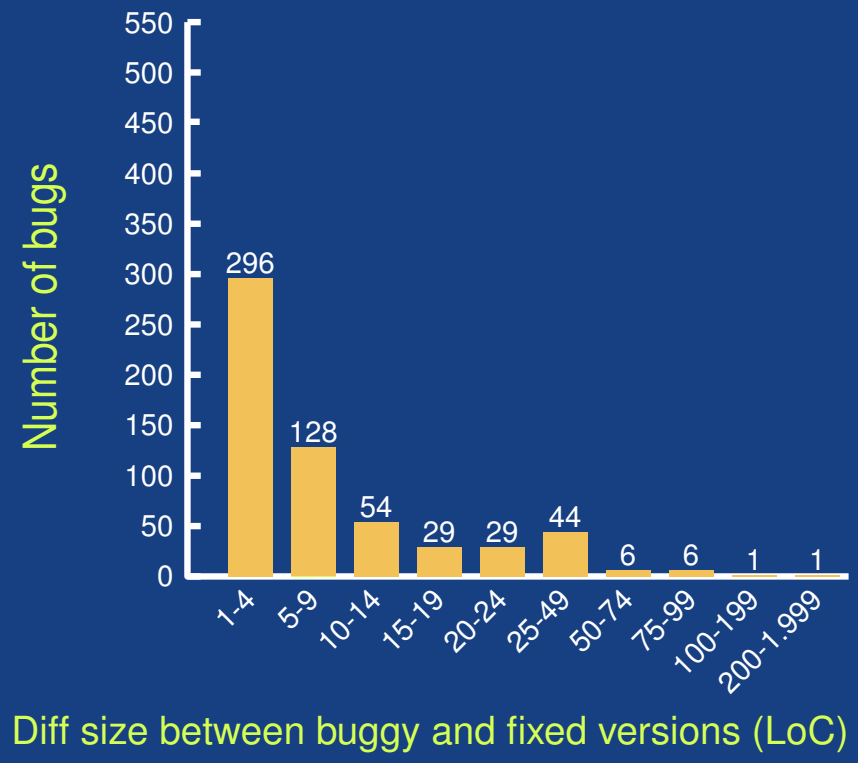


Real-World Bugs: Defects4J

594 bugs from 15 popular Java projects

- Extended version of Defects4J data set





How to Determine Detected Bugs?

Bugs + fixes

Bug detectors

How to Determine Detected Bugs?

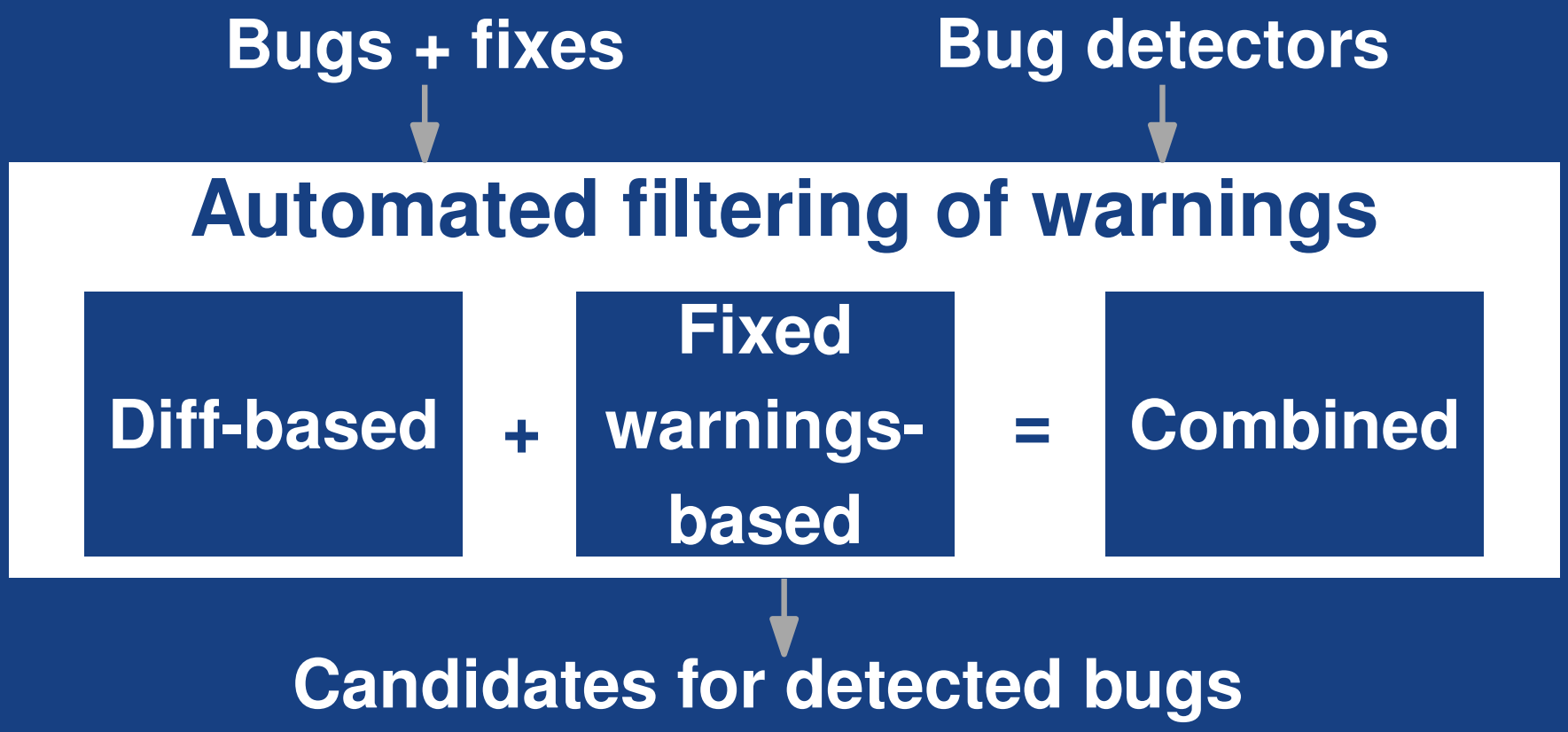
Bugs + fixes

Bug detectors

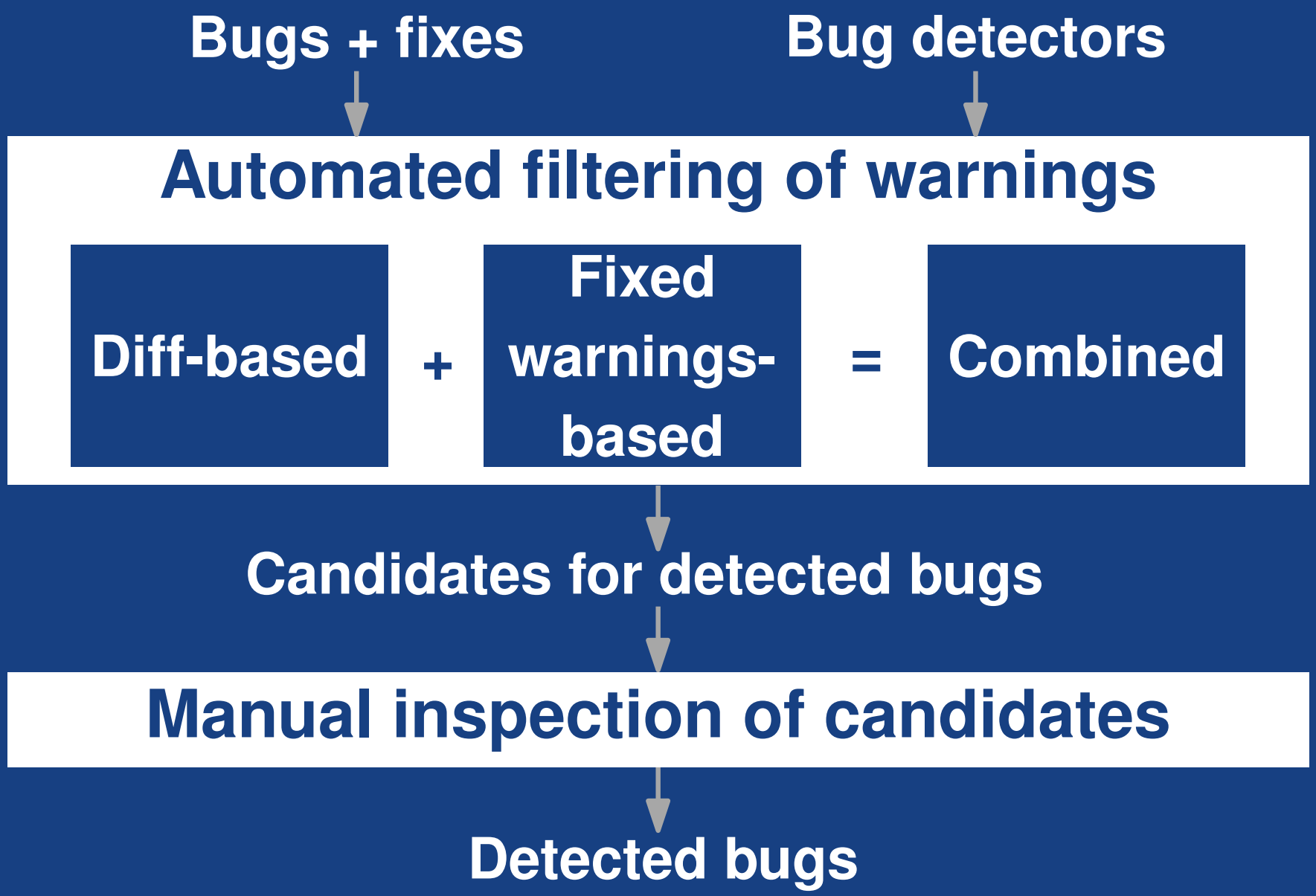
Automated filtering of warnings

Diff-based	+	Fixed warnings-based	=	Combined

How to Determine Detected Bugs?



How to Determine Detected Bugs?



Warnings to Inspect

	All warnings			
	Per bug			
Tool	Min	Max	Avg	Total
Error Prone	0	148	7.58	4,402
Infer	0	36	0.33	198
SpotBugs	0	47	1.1	647
Total				**5,247**

Warnings to Inspect

	All warnings				
	Per bug				Candidates
Tool	Min	Max	Avg	Total	**only**
Error Prone	0	148	7.58	4,402	53
Infer	0	36	0.33	198	32
SpotBugs	0	47	1.1	647	68
Total				**5,247**	**153**

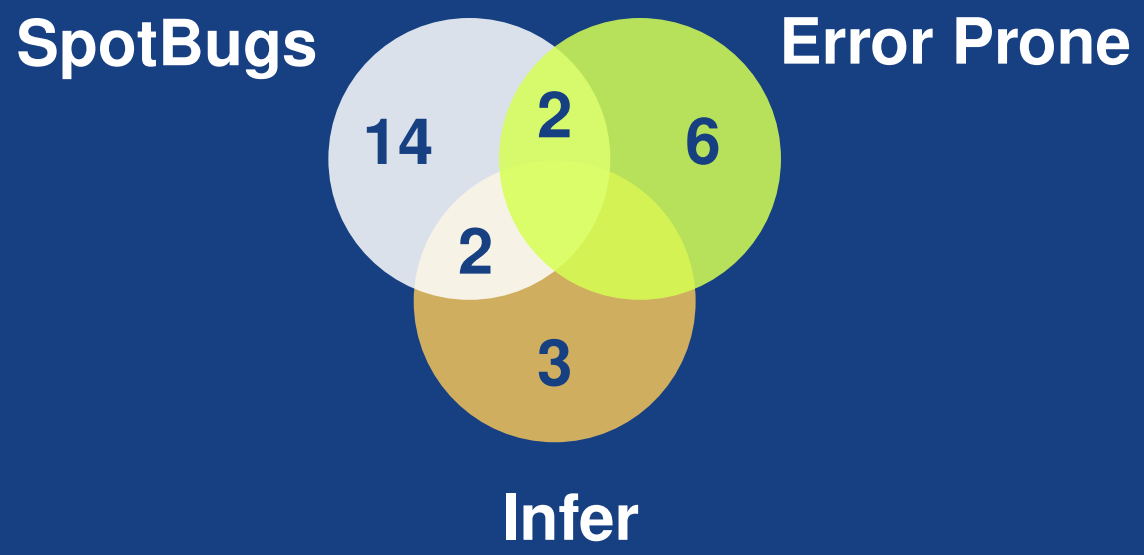
Warnings to Inspect

	All warnings				
	Per bug				Candidates
Tool	Min	Max	Avg	Total	**only**
Error Prone	0	148	7.58	4,402	53
Infer	0	36	0.33	198	32
SpotBugs	0	47	1.1	647	68
Total				**5,247**	**153**

97% of all warnings are removed by the automated filtering step

Our Findings

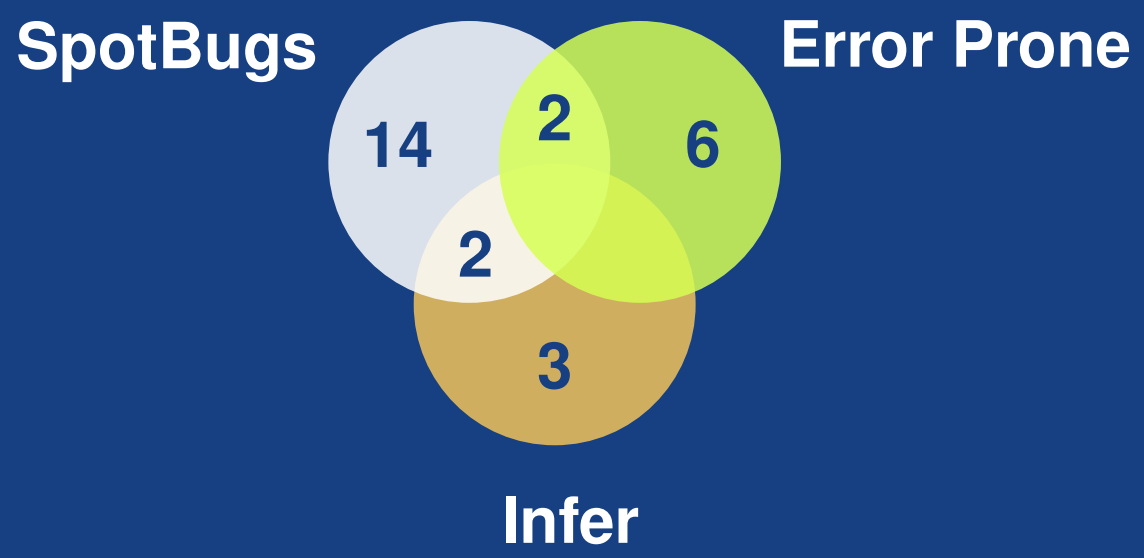
Our Findings



The three tools together:

Detect 27 out of 594 bugs

Our Findings



The three tools together:

Detect 27 out of 594 bugs

95.5% of the bugs are missed!

Our Findings (2)

Sample of 20 missed bugs:

Our Findings (2)

Sample of 20 missed bugs:

- **14/20 require domain-specific knowledge, e.g.:**
  - Joda-Time leap year
  - Closure compiler specific function declaration

Our Findings (2)

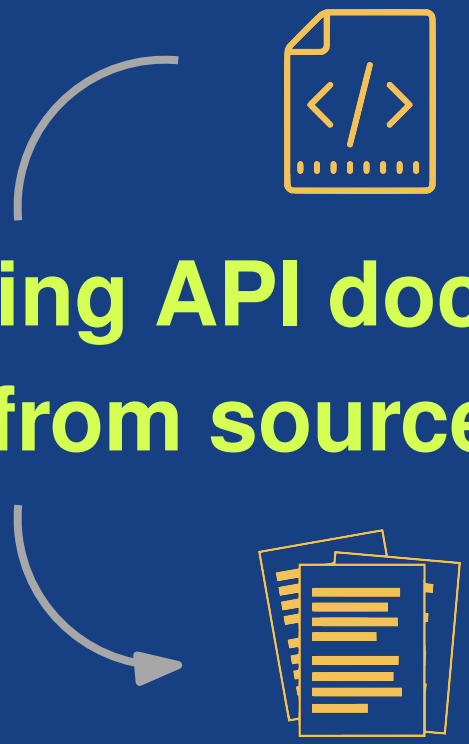
Sample of 20 missed bugs:

- **14/20 require domain-specific knowledge, e.g.:**
  - Joda-Time leap year
  - Closure compiler specific function declaration
- **6/20 are near missess, e.g.:**
  - index out-of-bounds checker, but only intra-procedural
  - index out-of-bounds checker, but only for arrays, not ArrayLists

Our Findings (2)

Sample of 20 missed bugs:

- **14/20 require domain-specific knowledge, e.g.:**
  - Joda-Time leap year
  - Closure compiler specific function declaration
- **6/20 are near missess, e.g.:**
  - index out-of-bounds checker, but only intra-procedural
  - index out-of-bounds checker, but only for arrays, not ArrayLists
- **Huge room for improvement**

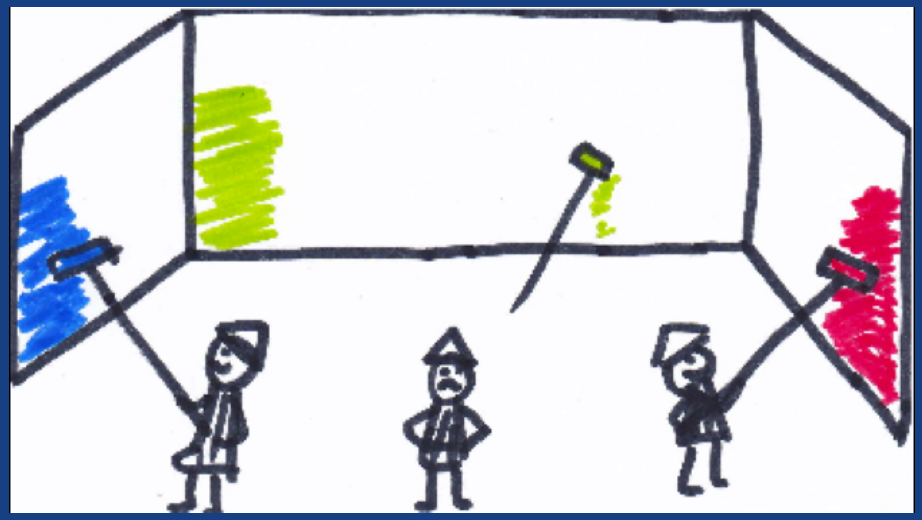


Learning API documentation from source code

Is This Class Thread-Safe? Inferring Documentation using Graph-Based Learning. Andrew Habib and Michael Pradel. IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018

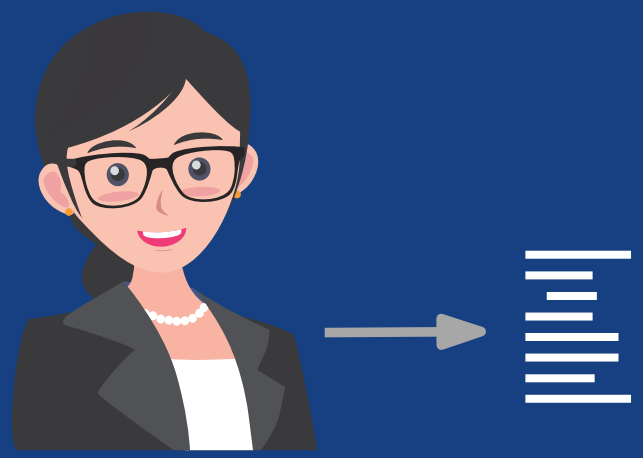
(Chapter 3)

Thread-Safe Classes

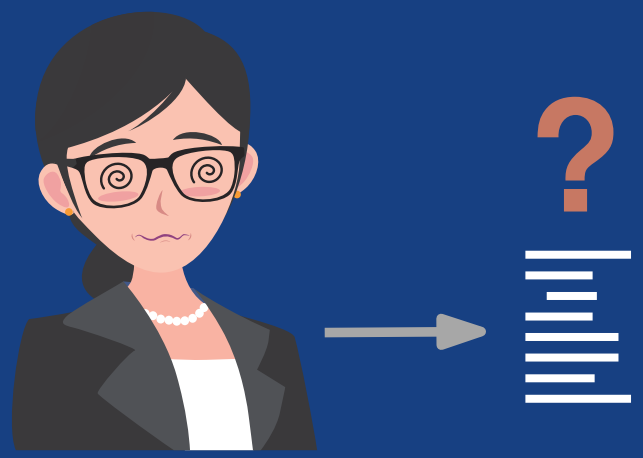


Thread-Safe Classes

Thread-Safe Classes



Thread-Safe Classes

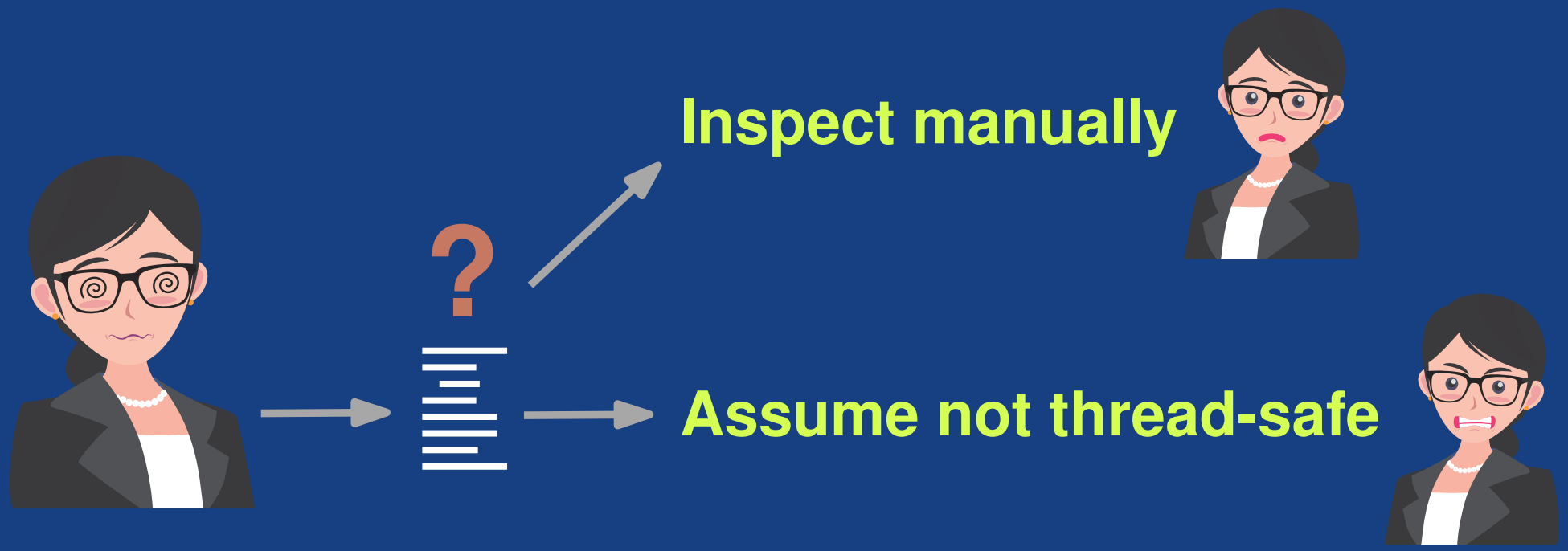


Is this class thread-safe?

Thread-Safe Classes

Is this class thread-safe?

Thread-Safe Classes



Is this class thread-safe?

Thread-Safe Classes

Inspect manually

Assume not thread-safe

Assume thread-safe

Is this class thread-safe?

Documentation of Thread Safety

Case study: The Qualitas Corpus

- **112 Java projects**
- **179,239 classes**

Documentation of Thread Safety

Case study: The Qualitas Corpus

- **112 Java projects**
- **179,239 classes**
- **Search: "concu", "thread", "sync", "parallel"**
  - 8,655 search hits
  - Randomly sample 120 hits
  - Manually inspect random sample

Documentation of Thread Safety

Case study: The Qualitas Corpus

- **Search: "concu", "thread", "sync", "parallel"**
  - 8,655 search hits (from 179,239 classes)
  - Manually inspect random sample of 120 hits

Documented as:	Count	%
Thread-safe	11	9.2%
Not thread-safe	12	10.0%
Conditionally thread-safe	2	1.7%
No documentation	95	79.2%
Total inspected classes	**120**	**100.0%**

Documentation of Thread Safety

Case study: The Qualitas Corpus

- **Search: "concu", "thread", "sync", "parallel"**
  - 8,655 search hits (from 179,239 classes)
  - Manually inspect random sample of 120 hits

Documented as:	Count	%
Thread-safe	11	9.2%
Not thread-safe	12	10.0%
Conditionally thread-safe	2	1.7%
No documentation	95	79.2%
Total inspected classes	**120**	**100.0%**

21%

Documentation of Thread Safety

Case study: The Qualitas Corpus

- **Search: "concu", "thread", "sync", "parallel"**
  - 8,655 search hits (from 179,239 classes)
  - Manually inspect random sample of 120 hits

Documented as:	Count	%
Thread-safe	11	9.2%
Not thread-safe	12	10.0%
Conditionally thread-safe	2	1.7%

21%

By extrapolation:

% of documented classes in corpus = 1.004%

Is This Class Thread-Safe?

Given an object-oriented class with unknown multi-threading behaviour, infer whether it is supposed to be thread-safe or not

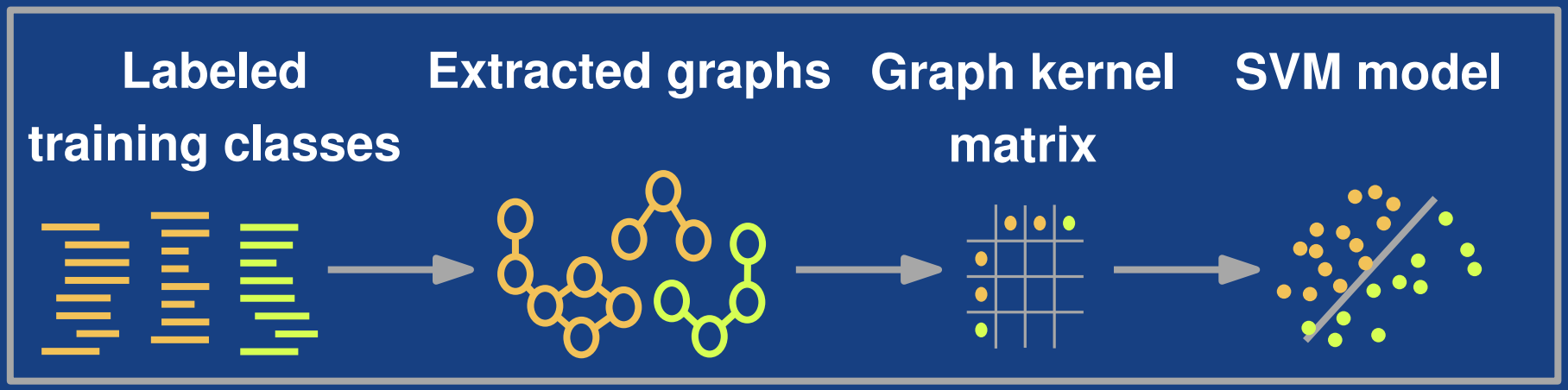
Is This Class Thread-Safe?

Given an object-oriented class with unknown multi-threading behaviour, infer whether it is supposed to be thread-safe or not

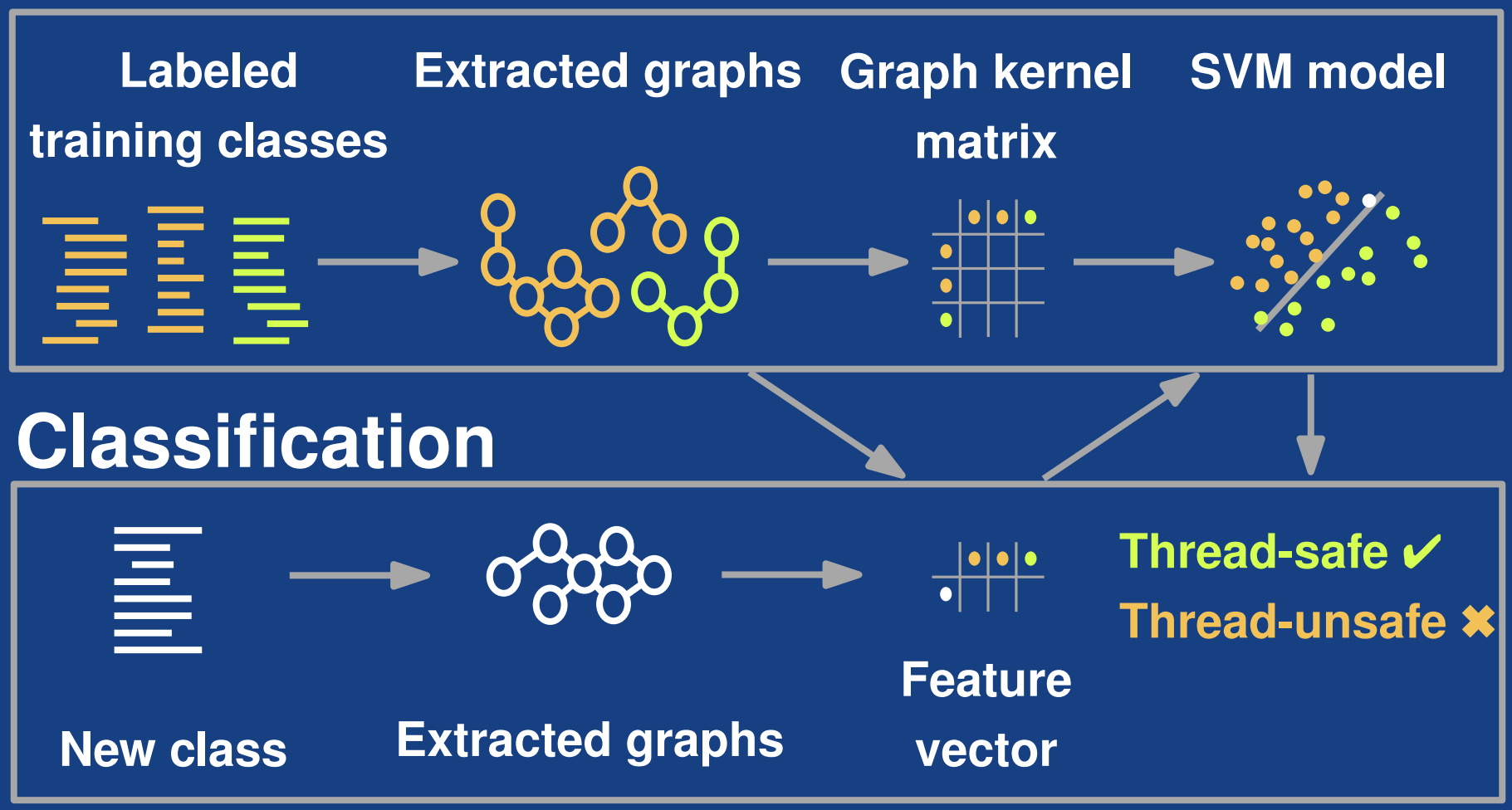
TSFinder: Machine learning-based approach to infer thread safety documentation

Overview of TSFinder

Training



Overview of TSFinder



Field-Focused Graphs

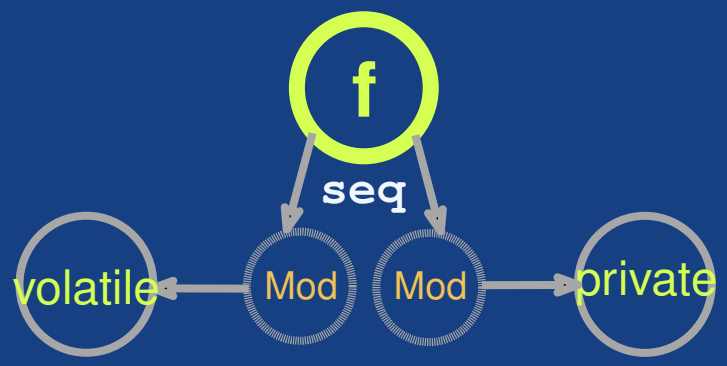
```
public class Sequence {

}
```

Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;

}
```

Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;
```

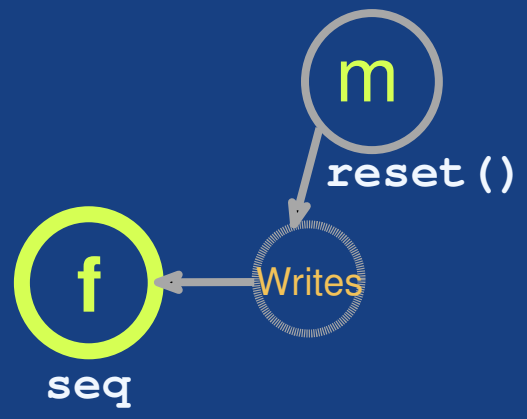



```
}
```

Mod: Modifier

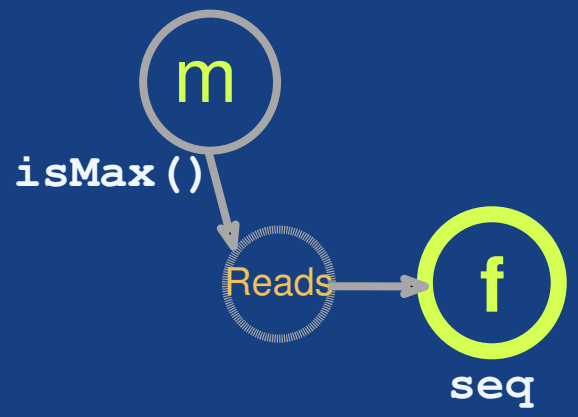
Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;

  void reset() {
    seq = 0;
  }
}
```



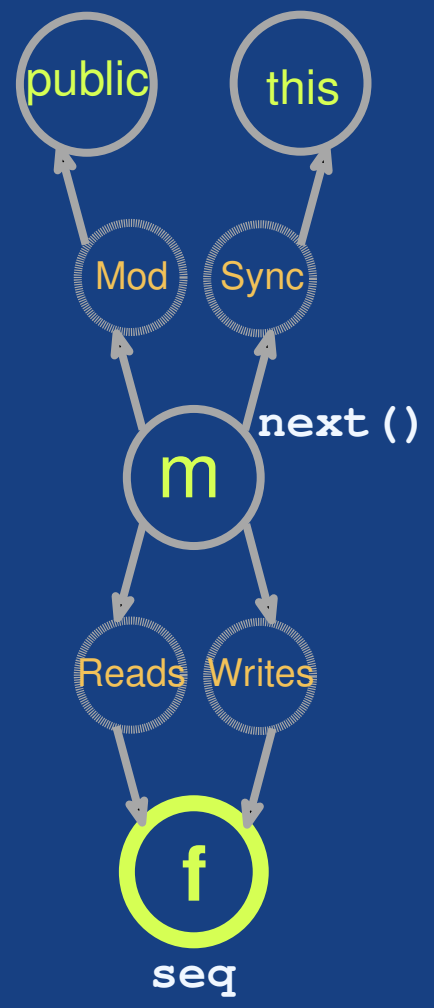
Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;

  boolean isMax() {
    return seq > MAX;
  }

}
```



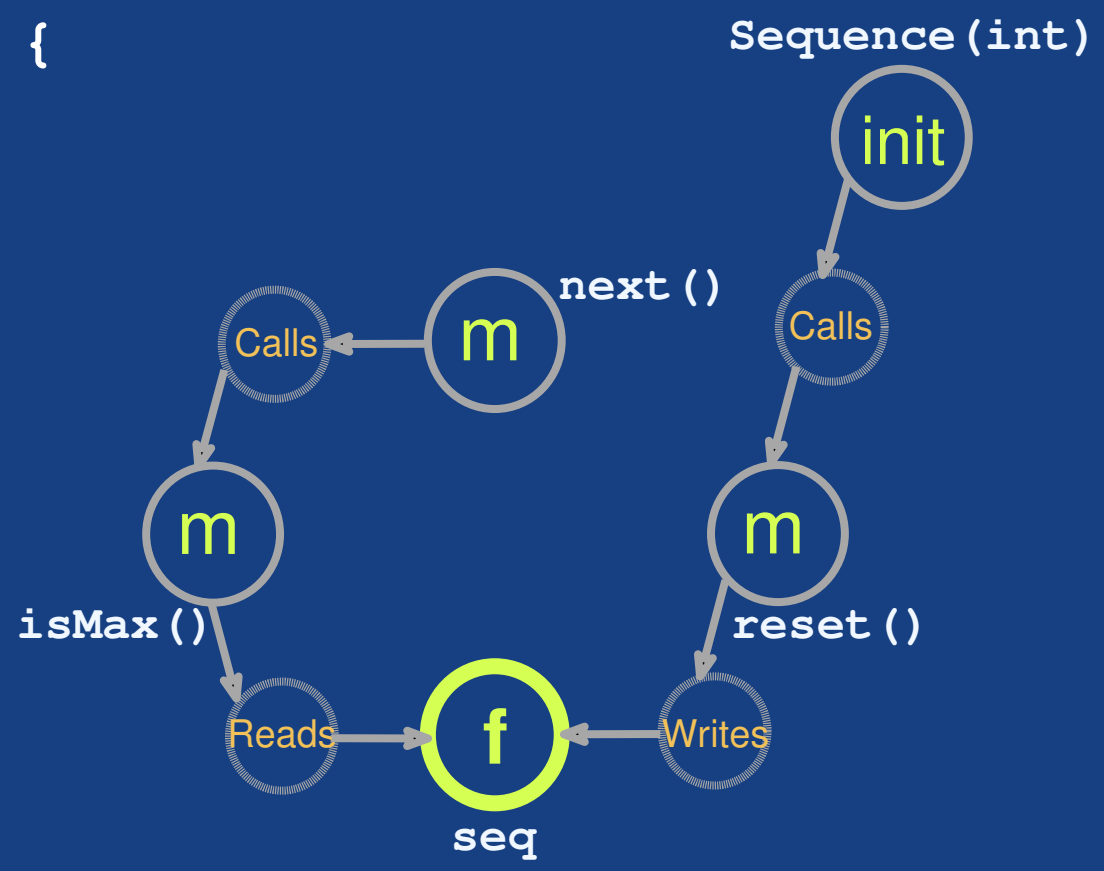
Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;

  synchronized
  public int next() {
    if(!isMax())
      return seq++;
    return -1;
  }

}
```



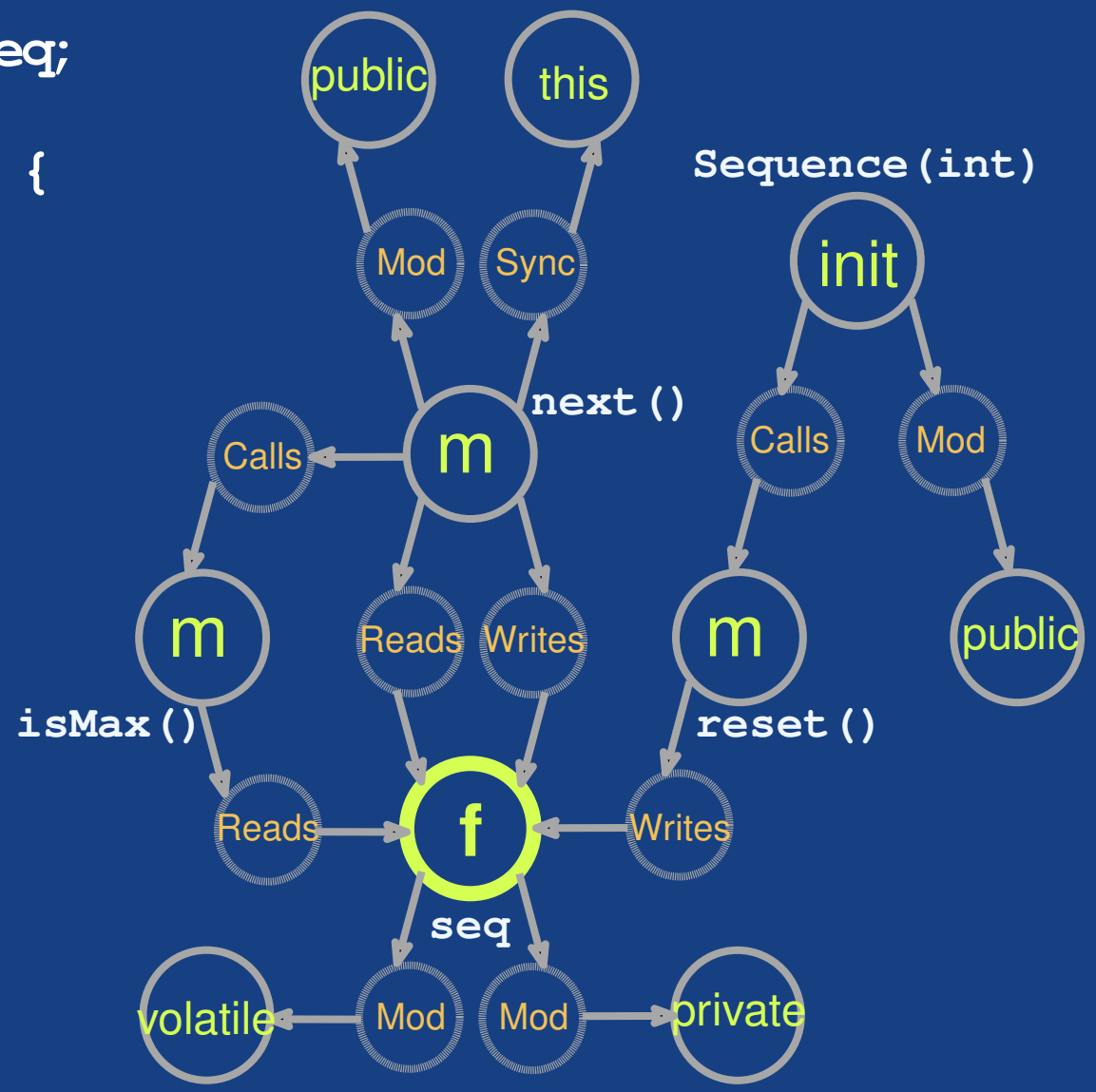
Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;

  public Sequence(int m) {

    reset();
  }
  synchronized
  public int next() {
    if(!isMax())


  }

}
```



Field-Focused Graphs

```
public class Sequence {
  private volatile int seq;
  private int MAX;
  public Sequence(int m) {
    MAX = m;
    reset();
  }
  synchronized
  public int next() {
    if(!isMax())
      return seq++;
    return -1;
  }
  boolean isMax() {
    return seq > MAX;
  }
  void reset() {
    seq = 0;
  }
}
```



Mod: Modifier

Class to Vector

Known classes

New class C

Class to Vector

Known classes

New class C

Graph kernel:[1] $K($  ,  $) = k \in [0, 1]$

Similarity score

1 We use the *Weisfeiler-Lehman Graph Kernels* [Shervashidze et al., 2011]

Class to Vector

Known classes

New class C

Graph kernel:[1] $K($  $,$  $) = k \in [0, 1]$

Similarity score

Summary of similarity of C to known classes

1 We use the *Weisfeiler-Lehman Graph Kernels* [Shervashidze et al., 2011]

Class to Vector

Known classes

New class C

Graph kernel:[1] $K($  $,$  $) = k \in [0, 1]$ **Similarity score**

Summary of similarity of C to known classes

Vector representation of class C

1 We use the *Weisfeiler-Lehman Graph Kernels* [Shervashidze et al., 2011]

Effectiveness of TSFinder

- 230 JDK classes with documented thread safety (balanced number of examples)
- Two-class SVM with SGD[1] and hinge loss
- 10-fold cross-validation

1 Stochastic gradient descent

Effectiveness of TSFinder

- 230 JDK classes with documented thread safety (balanced number of examples)
- Two-class SVM with SGD[1] and hinge loss
- 10-fold cross-validation

	Thread-Safe		Not Thread-Safe	
Accuracy	**Prec.**	**Rec.**	**Prec.**	**Rec.**
94.5%				

1 Stochastic gradient descent

Effectiveness of TSFinder

- 230 JDK classes with documented thread safety (balanced number of examples)
- Two-class SVM with SGD[1] and hinge loss
- 10-fold cross-validation

	Thread-Safe		Not Thread-Safe	
Accuracy	**Prec.**	**Rec.**	**Prec.**	**Rec.**
94.5%	**94.9%**	**94.0%**	**94.2%**	**95.0%**

1 Stochastic gradient descent

Effectiveness of TSFinder

- 230 JDK classes with documented thread safety (balanced number of examples)
- Two-class SVM with SGD[1] and hinge loss
- 10-fold cross-validation

	Thread-Safe		Not Thread-Safe	
Accuracy	**Prec.**	**Rec.**	**Prec.**	**Rec.**
94.5%	**94.9%**	**94.0%**	**94.2%**	**95.0%**

Most predictions are correct!

1 Stochastic gradient descent

TSFinder Against Baseline

Naive classifier using simple class feautres, e.g.:

- % of volatile fields
- % of synchronized methods

TSFinder Against Baseline

Naive classifier using simple class feautres, e.g.:

- % of volatile fields
- % of synchronized methods

	Accuracy	
Classifier	TSFinder	Naïve
SVM (SGD[1] with hinge loss)	94.5%	75.0%
Random forest	94.1%	79.3%
SVM (SMO[2])	92.5%	70.6%
SVM (SGD with log loss)	92.0%	74.3%
Additive logistic regression	92.8%	74.5%

1 Stochastic gradient descent
2 Sequential minimal optimization

TSFinder Against Baseline

Naive classifier using simple class feautres, e.g.:

- % of volatile fields
- % of synchronized methods

	Accuracy	
Classifier	TSFinder	Naïve
SVM (SGD[1] with hinge loss)	94.5%	75.0%
Random forest	94.1%	79.3%
SVM (SMO[2])	92.5%	70.6%
SVM (SGD with log loss)	92.0%	74.3%
Additive logistic regression	92.8%	74.5%

1 Stochastic gradient descent
2 Sequential minimal optimization

Our Contributions

- State of the practice for documenting thread safety is poor
- New program representation: Field-focused graphs
- TSFinder uses machine learning to infer documentation
- TSFinder infers thread safety documentation with accuracy of 94.5%

Cross-checking documentation against observed exceptional runtime

(Chapter 4)

How to Judge Observed Behavior?

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

How to Judge Observed Behavior?

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Is this the correct behavior?

How to Judge Observed Behavior?

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

How to Judge Observed Behavior?

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Documentation:

```
public static Strin
Unwraps a given
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Is this the correct behavior?

No, w.r.t. documentation

Overview of DocRT

- Classify observed exceptional behavior based on documentation

Overview of DocRT

Classify observed exceptional behavior based on documentation

Runtime Behavior
+
Documentation

→ DocRT →

Correct ✔
Buggy ✖

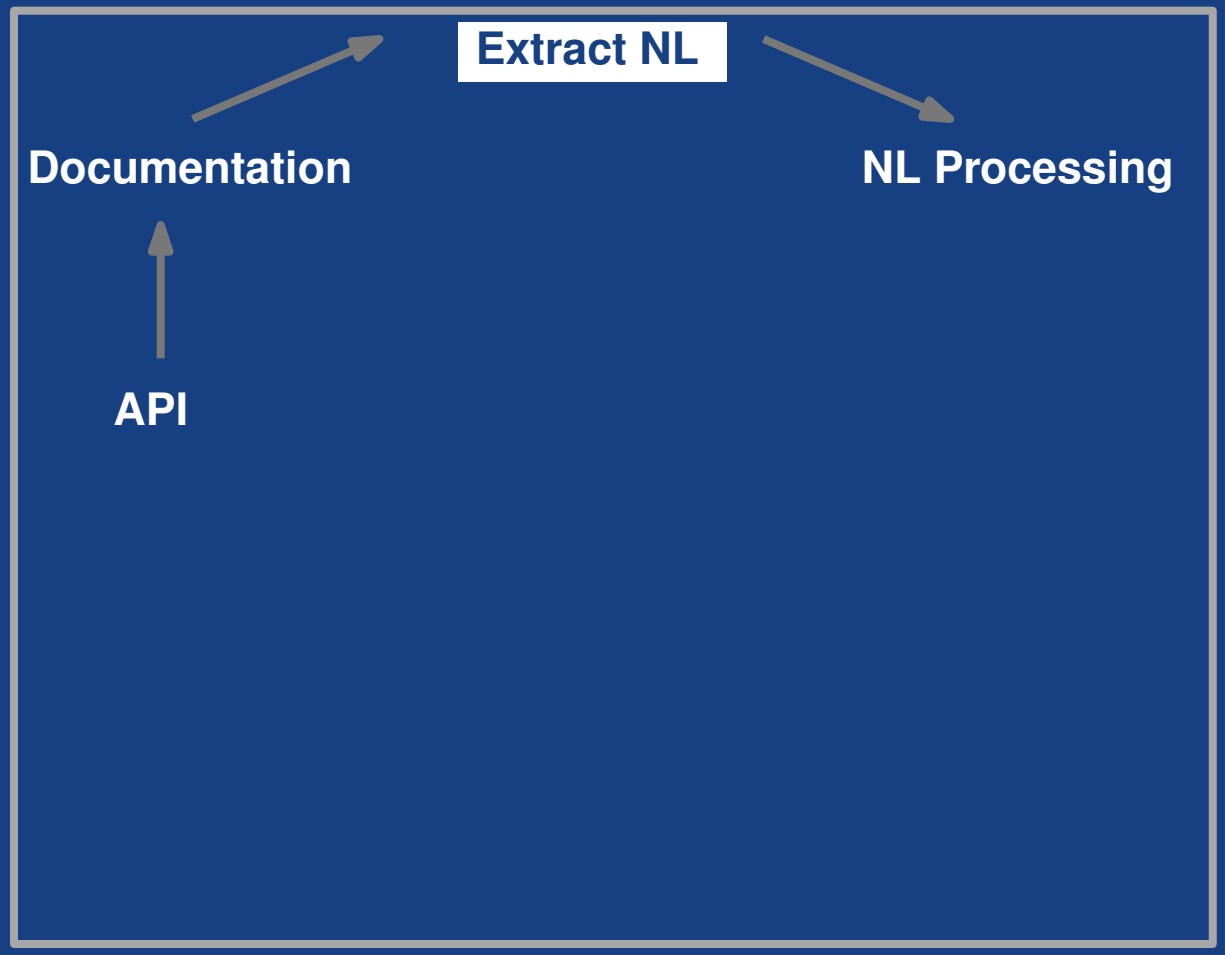
Overview of DocRT

Data Generation

API

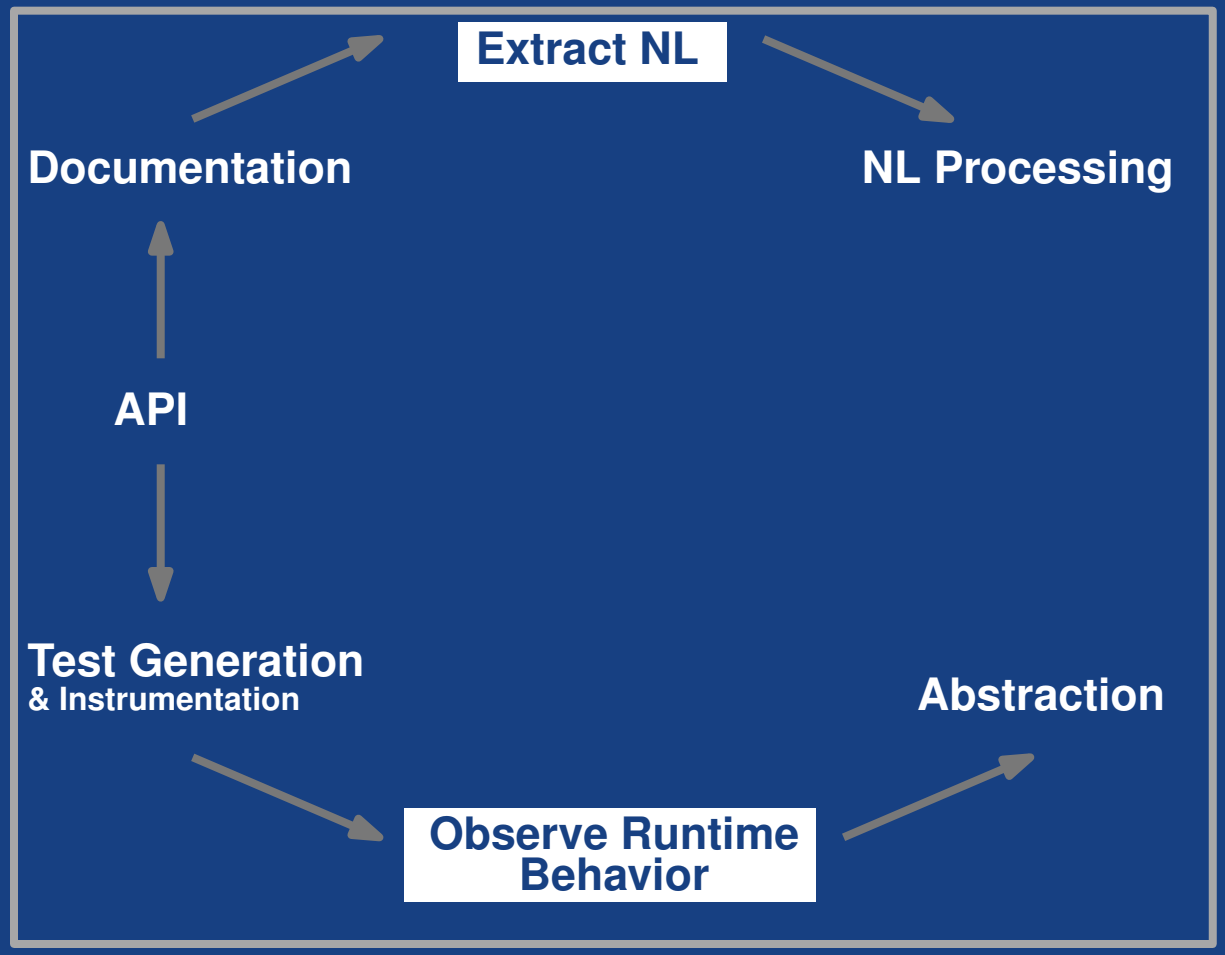
Overview of DocRT

Data Generation



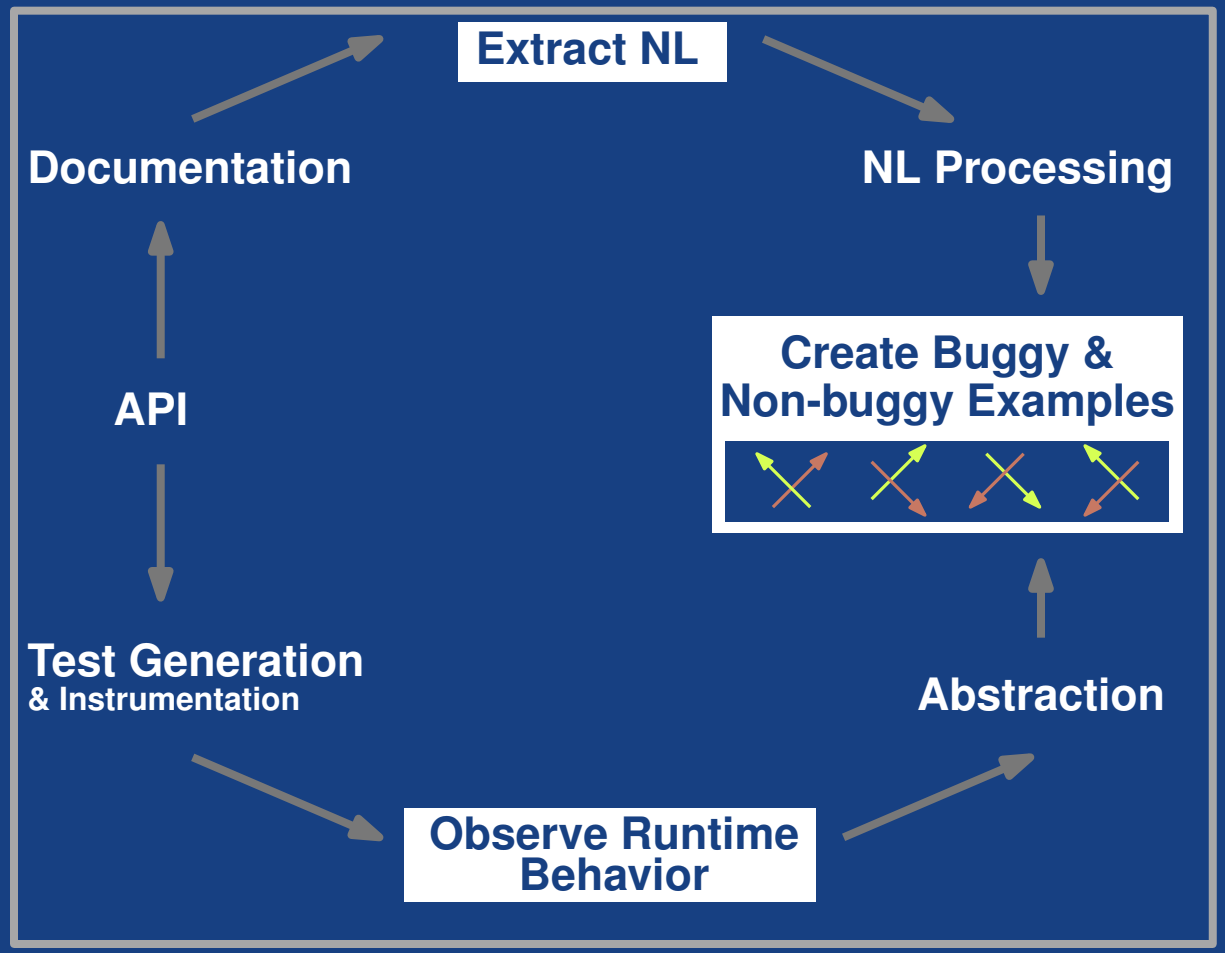
Overview of DocRT

Data Generation

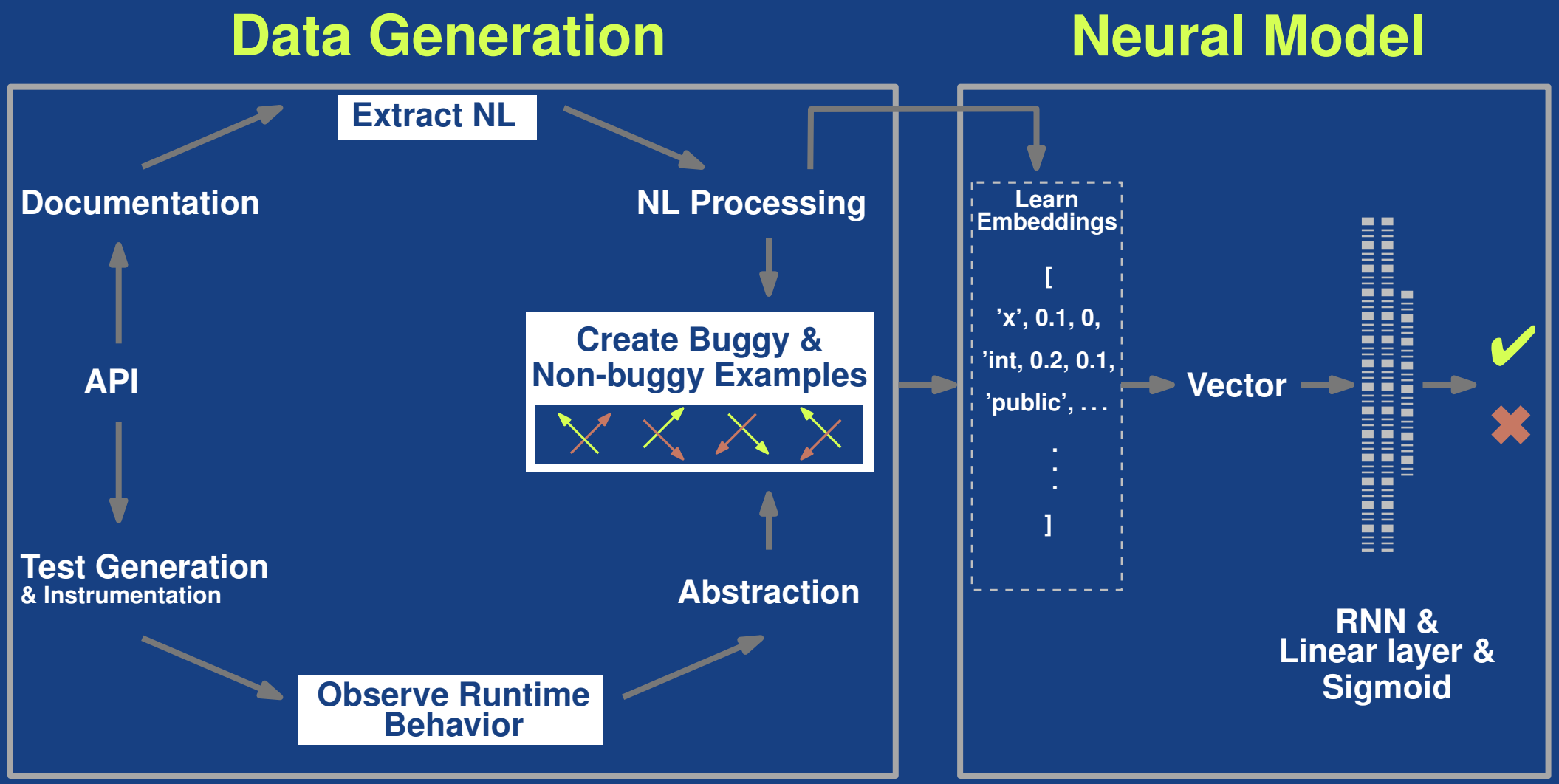


Overview of DocRT

Data Generation



Overview of DocRT



Extracting NL Information

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Extracting NL Information

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Names of the method & its parameters

Extracting NL Information

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

Types of return value & parameters & exceptions

Extracting NL Information

Documentation:

```
public static String unwrap(String str, String wrapToken)
Unwraps a given string from another string.  ...
Parameters:
str - the String to be unwrapped, can be null
wrapToken - the String used to unwrap
Returns:
unwrapped String or the original string if it is
    not quoted properly with the wrapToken
```

NL Descriptions

the method & its paramaters & return value & exceptions

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Abstract behavior through:

Pre- & Post-states

- Base object
- Arguments

Result

- Return value
- Thrown exception

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Base object: None here, static method

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Base object: None here, static method

Parameters: `str:"a", wrapToken:"a"`

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Base object: None here, static method

Parameters: `str:"a"`, `wrapToken:"a"`

Return: None here, runtime exception

Extracting Runtime Information

Invocation:

```
commons.lang3.StringUtils.unwrap("a", "a");
```

Observed behavior:

```
Exception in thread "main"
    java.lang.StringIndexOutOfBoundsException:
                                String index out of range: -1
```

Base object: None here, static method

Parameters: `str:"a"`, `wrapToken:"a"`

Return: None here, runtime exception

Runtime Exception:

`java.lang.StringIndexOutOfBoundsException`

How to Obtain Buggy Examples?

- **Assume all collected pairs of documentation and behavior are correct.**

How to Obtain Buggy Examples?

- **Assume all collected pairs of documentation and behavior are correct.**
- **Mutate the collected pairs.**

  Behavior mutations:

  - Raise random exception
  - Remove thrown exception
  - Replace thrown exception by random one

  NL mutations:

  - Remove thrown exception from documentation

How to Obtain Buggy Examples?

- **Assume** all collected pairs of documentation and behavior **are correct**.

Invocation:

```
commons.lang3.StringUtils.unwrap("xyx", "x");
```

Observed behavior: return value: `"y"`

Correct example

Mutation: Raise random exception

How to Obtain Buggy Examples?

- **Assume** all collected pairs of documentation and behavior **are correct**.

Correct example

Invocation:

```
commons.lang3.StringUtils.unwrap("xyx", "x");
```

Observed behavior: return value: `"y"`

Mutation: Raise random exception

Buggy example

Invocation:

```
commons.lang3.StringUtils.unwrap("xyx", "x");
```

Observed behavior:

```
Exception in thread "main"
  java.lang.NullPointerException
```

Evaluation: Setup

- **5,000 Java projects from Maven**
- **207,455 Public methods with doc**
- **Randoop**[1] generates 146,397 tests for 25,076 methods
- **292,782 pairs of buggy & non-buggy method doc. and behavior (almost balanced)**
- **70, 15, 15 % split for training, validation, and testing**

1 Pacheco et al."Feedback-Directed Random Test Generation". ICSE'07

Evaluation: Setup

- **5,000 Java projects from Maven**
- **207,455 Public methods with doc**
- **Randoop**[1] generates 146,397 tests for 25,076 methods
- **292,782 pairs of buggy & non-buggy method doc. and behavior (almost balanced)**
- **70, 15, 15 % split for training, validation, and testing**

1 Pacheco et al."Feedback-Directed Random Test Generation". ICSE'07

Training The Neural Model

- Two Bi-directional LSTM layers (size=100)
- Dropout of 0.3
- Batch size of 1k
- 50 epochs
- Train using the Adam optimizer
- Decaying learning rate of 0.0001

Effectiveness of DocRT

On the test set:

Precision: 81%

Accuracy: 87%

Recall: 97%

F1: 88%

Effectiveness of DocRT

On the test set:

Precision: 81% **Accuracy: 87%**

Recall: 97% **F1: 88%**

In practice:

18 bugs reports from popular projects, e.g.:

- 11 Undocumented `NullPointerException`
- 3 Unexpected `IndexOutOfBounds` in corner cases
- 1 Undocumented `IllegalArgumentException`

Effectiveness of DocRT

On the test set:

Precision: 81% **Accuracy: 87%**

Recall: 97% **F1: 88%**

In practice:

18 bugs reports from popular projects, e.g.:

- 11 Undocumented `NullPointerException`
- 3 Unexpected `IndexOutOfBounds` in corner cases
- 1 Undocumented `IllegalArgumentException`

DocRT detects all of them.

Detecting New Bugs

Sample top 50 buggy predictions in the test set

Analyzed method calls	21,592
Calls where DocRT reports a warning	4,829
Inspected calls with warning	50
True positives	45
Undocumented exception	39
Wrong type of exception documented	1
False positives	5
Indirect description of the exceptional behavior	4
Exceptional behavior is fully documented	1

Detecting New Bugs

Sample top 50 buggy predictions in the test set

Analyzed method calls	21,592
Calls where DocRT reports a warning	4,829
Inspected calls with warning	**50**
True positives	**45**
Undocumented exception	**39**
Wrong type of exception documented	**1**
False positives	**5**
Indirect description of the exceptional behavior	4
Exceptional behavior is fully documented	1

Detecting New Bugs: Example

Documentation:

```
public byte byteAt(int i)
Gets a byte within this sequence of bytes
Parameters:
i - index into sequence
Returns:
byte
Throws:
IllegalArgumentException - if i is out of range
```

Invocation:

```
org.apache.fluo.api.data.Bytes bytes = new
                                org.apache.fluo.api.data.Bytes();
bytes.byteAt(-1);
```

Observed behavior:

```
java.lang.IndexOutOfBoundsException: i < 0, -1
```

Detecting New Bugs: Example

Documentation:

```
public byte byteAt(int i)
Gets a byte within this sequence of bytes
Parameters:
i - index into sequence
Returns:
byte
Throws:
IllegalArgumentException - if i is out of range
```

Bug

Invocation:

```
org.apache.fluo.api.data.Bytes bytes = new
                                org.apache.fluo.api.data.Bytes();
bytes.byteAt(-1);
```

Observed behavior:

```
java.lang.IndexOutOfBoundsException: i < 0, -1
```

Detecting New Bugs: Example

Documentation:

```
public byte byteAt(int i)
Gets a byte within this sequence of bytes
Parameters:
i - index into sequence
Returns:
byte
Throws:
java.lang.IndexOutOfBoundsException: - if i is out of range
```

Fix

Invocation:

```
org.apache.fluo.api.data.Bytes bytes = new
                                org.apache.fluo.api.data.Bytes();
bytes.byteAt(-1);
```

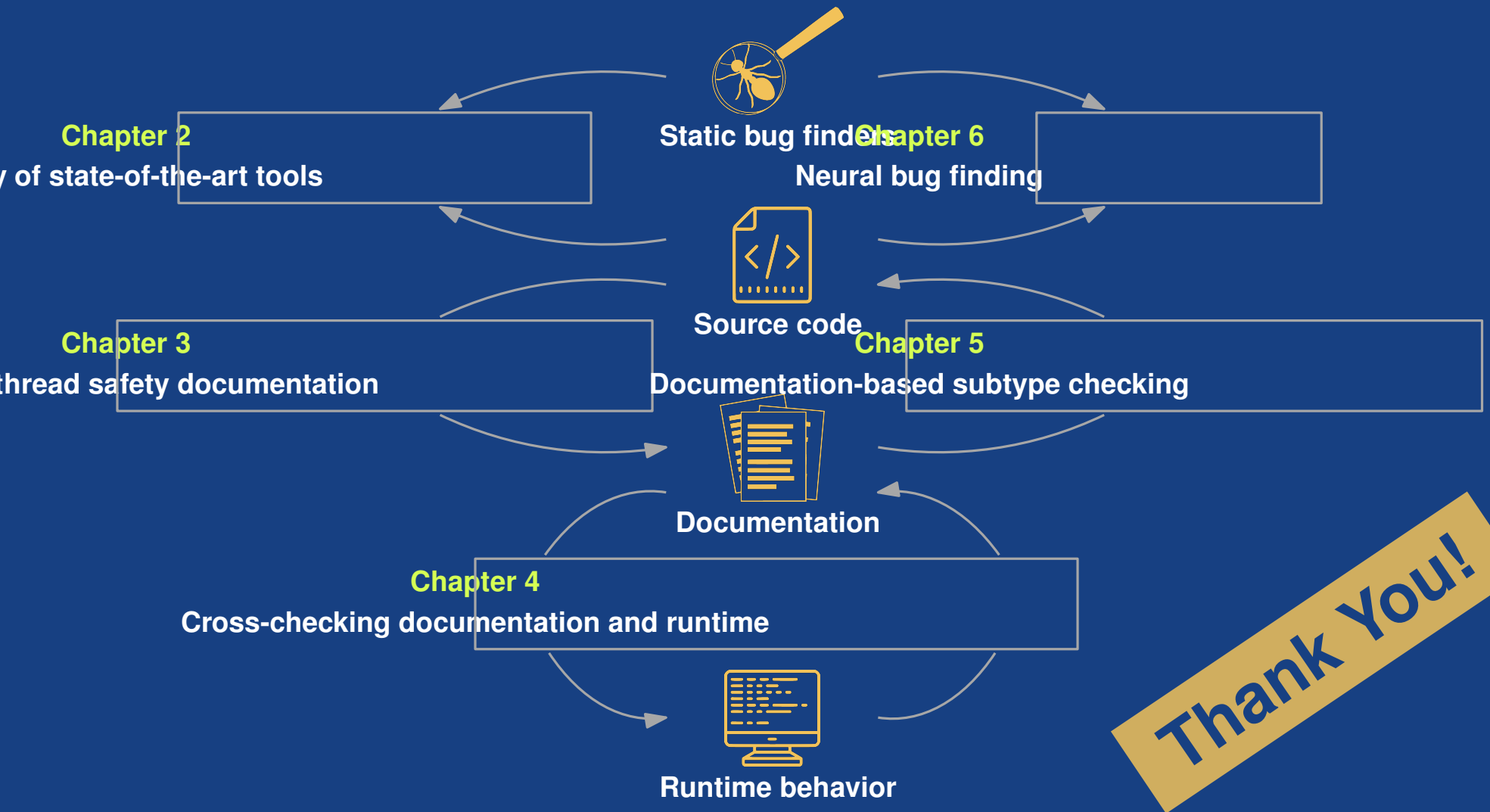
Observed behavior:

```
java.lang.IndexOutOfBoundsException: i < 0, -1
```

Our Contributions

- Novel learning task for the test oracle problem
- DocRT leverages documentation to check runtime behavior
- DocRT detects inconsistent documentation w.r.t. runtime behavior with accuracy of 87%

Learning from Programs & their Documentation



Thank You!

Characters images from Freepik, https://www.freepik.com/

Future Work

- **Utilizing documentation in more SE tasks**
  e.g., in program repair
- **Extracting and utilizing more domain-knowledge from other NL sources**
  e.g., bug reports, issue trackers, commit messages
- **Richer source code representation for ML-based SE tasks**
- **Abstraction (representation) of runtime behavior**
- **Learning from runtime**

Publications

- Andrew Habib and Michael Pradel. *How many of all bugs do we find? a study of static bug detectors.* IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018.
- Andrew Habib and Michael Pradel. *Is this class thread-safe? inferring documentation using graph-based learning.* IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018.
- Andrew Habib, Avraham Shinnar, Martin Hirzel, and Michael Pradel. *Type Safety with JSON Subschema.* CoRR abs/1911.12651 (2019).
- Andrew Habib and Michael Pradel. *Neural Bug Finding: A Study of Opportunities and Challenges.* CoRR abs/1906.00307 (2019).