Is This Class Thread-Safe?  
Inferring Documentation using Graph-Based Learning

Andrew Habib, Michael Pradel  
TU Darmstadt, Germany  
software-lab.org
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?

Inspect manually
Thread-Safe Classes

Is this class thread-safe?

Inspect manually

Assume not thread-safe
Thread-Safe Classes

Is this class thread-safe?

- Assume not thread-safe
- Assume thread-safe
- Inspect manually

Created by Freepik
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

<table>
<thead>
<tr>
<th>Documented as</th>
<th>Count</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread-safe</td>
<td>11</td>
<td>9.2%</td>
</tr>
<tr>
<td>Not thread-safe</td>
<td>12</td>
<td>10.0%</td>
</tr>
<tr>
<td>Conditionally thread-safe</td>
<td>2</td>
<td>1.7%</td>
</tr>
<tr>
<td>No documentation</td>
<td>95</td>
<td>79.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total inspected classes</th>
<th>120</th>
<th>100.0%</th>
</tr>
</thead>
</table>
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

<table>
<thead>
<tr>
<th>Documented as</th>
<th>Count</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread-safe</td>
<td>11</td>
<td>9.2%</td>
</tr>
<tr>
<td>Not thread-safe</td>
<td>12</td>
<td>10.0%</td>
</tr>
<tr>
<td>Conditionally thread-safe</td>
<td>2</td>
<td>1.7%</td>
</tr>
<tr>
<td>No documentation</td>
<td>95</td>
<td>79.2%</td>
</tr>
</tbody>
</table>

**Total inspected classes** 120 100.0%
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

<table>
<thead>
<tr>
<th>Documented as:</th>
<th>Count</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread-safe</td>
<td>11</td>
<td>9.2%</td>
</tr>
<tr>
<td>Not thread-safe</td>
<td>12</td>
<td>10.0%</td>
</tr>
<tr>
<td>Conditionally thread-safe</td>
<td>2</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

By extrapolation:

% of documented classes = 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

This talk: TSFinder
Machine learning approach to infer thread-safety documentation
Overview of TSFinder

Training

Labeled training classes → Extracted graphs → Graph kernel matrix → SVM model
Overview of TSFinder

Training

Labeled training classes → Extracted graphs → Graph kernel matrix → SVM model

Classification

New class → Extracted graphs → Feature vector

Thread-safe ✓  Thread-unsafe
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;
    }
}
Field-Focused Graphs

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

    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;
    }
}
```
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;
   }
}
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;
    }
}
Field-Focused Graphs

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

    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;
    }
}
```

**Diagram**

```
+-----------+      +-----------+
|           |      |           |
|   m       |      |   f       |
|           |      |           |
|   isMax()  |      | seq       |
|           |      |           |
| Reads     |      |           |
```

public class Sequence {
    private volatile int seq;

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

    boolean isMax () {
        return seq > MAX;
    }

    void reset () {
        seq = 0;
    }
}
public class Sequence {
    private volatile int seq;

    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;
    }
}
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;
    }
}
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;
    }
}
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;
    }
}
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;
    }
}

boolean isMax() {
    return seq > MAX;
}
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;
    }
}

Build the rest of the graph as before
Class to Vector

Known classes

New class $C$
Class to Vector

Known classes

New class $C$

Graph kernel: $K(\text{known class}, \text{new class}) = k \in [0, 1]$

Similarity score

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

Known classes

New class $C$

Graph kernel: $K(\text{known classes}, C) = k \in [0, 1]$

Summary of similarity of $C$ to known classes

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

Known classes

New class $C$

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

Summary of similarity of $C$ to known classes

Vector representation of class $C$

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

230 Java classes from the JDK

- Explicit thread safety documentation

<table>
<thead>
<tr>
<th>Classes</th>
<th>Count</th>
<th>Fields</th>
<th></th>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
</tr>
<tr>
<td>TS</td>
<td>115</td>
<td>1</td>
<td>64</td>
<td>8.7</td>
<td>2</td>
</tr>
<tr>
<td>not TS</td>
<td>115</td>
<td>0</td>
<td>55</td>
<td>4.3</td>
<td>1</td>
</tr>
<tr>
<td>All</td>
<td>230</td>
<td>0</td>
<td>64</td>
<td>6.4</td>
<td>1</td>
</tr>
</tbody>
</table>
## Evaluation: Setup

### 230 Java classes from the JDK

- Explicit thread safety documentation

<table>
<thead>
<tr>
<th>Classes</th>
<th>Count</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
<th>Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>115</td>
<td>13</td>
<td>4,264</td>
<td>430.2</td>
<td>1,989</td>
</tr>
<tr>
<td>not TS</td>
<td>115</td>
<td>7</td>
<td>1,931</td>
<td>219.7</td>
<td>2,871</td>
</tr>
<tr>
<td>All</td>
<td>230</td>
<td>7</td>
<td>4,264</td>
<td>323.1</td>
<td>4,860</td>
</tr>
</tbody>
</table>
Effectiveness of TSFinder

- Two-class SVM with SGD*
- 10-fold cross-validation
- 230 labeled JDK classes

* Stochastic Gradient Descent
Effectiveness of TSFinder

- Two-class SVM with SGD*
- 10-fold cross-validation
- 230 labeled JDK classes

<table>
<thead>
<tr>
<th></th>
<th>Thread-Safe</th>
<th>Not Thread-Safe</th>
</tr>
</thead>
</table>

* Stochastic Gradient Descent
Effectiveness of TSFinder

- Two-class SVM with SGD*
- 10-fold cross-validation
- 230 labeled JDK classes

<table>
<thead>
<tr>
<th></th>
<th>Thread-Safe</th>
<th>Not Thread-Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>94.5%</td>
<td>94.2%</td>
</tr>
<tr>
<td>Prec.</td>
<td>94.9%</td>
<td>94.2%</td>
</tr>
<tr>
<td>Rec.</td>
<td>94.0%</td>
<td>95.0%</td>
</tr>
</tbody>
</table>

* Stochastic Gradient Descent
Effectiveness of TSFinder

- Two-class SVM with SGD*
- 10-fold cross-validation
- 230 labeled JDK classes

<table>
<thead>
<tr>
<th></th>
<th>Thread-Safe</th>
<th></th>
<th>Not Thread-Safe</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>94.5%</td>
<td>Prec. 94.9%</td>
<td>Rec. 94.0%</td>
<td>Prec. 94.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rec. 95.0%</td>
</tr>
</tbody>
</table>

Most predictions are correct!

* Stochastic Gradient Descent
Comparison with Baseline

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

- % of volatile fields
- % of synchronized methods
Comparison with Baseline

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

- % of volatile fields

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuray</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (SGD* with hinge loss)</td>
<td>94.5%</td>
</tr>
<tr>
<td>Random forest</td>
<td>94.1%</td>
</tr>
<tr>
<td>SVM (SMO**)</td>
<td>92.5%</td>
</tr>
<tr>
<td>SVM (SGD with log loss)</td>
<td>92.0%</td>
</tr>
<tr>
<td>Additive logistic regression</td>
<td>92.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TSFinder</th>
<th>Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (SGD* with hinge loss)</td>
<td>94.5%</td>
<td>75.0%</td>
</tr>
<tr>
<td>Random forest</td>
<td>94.1%</td>
<td>79.3%</td>
</tr>
<tr>
<td>SVM (SMO**)</td>
<td>92.5%</td>
<td>70.6%</td>
</tr>
<tr>
<td>SVM (SGD with log loss)</td>
<td>92.0%</td>
<td>74.3%</td>
</tr>
<tr>
<td>Additive logistic regression</td>
<td>92.8%</td>
<td>74.5%</td>
</tr>
</tbody>
</table>

* Stochastic Gradient Descent
** Sequential Minimal Optimization
Comparison with Baseline

**Naive classifier using simple class features, e.g.:**

- % of volatile fields

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TSFinder</th>
<th>Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (SGD* with hinge loss)</td>
<td>94.5%</td>
<td>75.0%</td>
</tr>
<tr>
<td>Random forest</td>
<td>94.1%</td>
<td><strong>79.3%</strong></td>
</tr>
<tr>
<td>SVM (SMO**)</td>
<td>92.5%</td>
<td>70.6%</td>
</tr>
<tr>
<td>SVM (SGD with log loss)</td>
<td>92.0%</td>
<td>74.3%</td>
</tr>
<tr>
<td>Additive logistic regression</td>
<td>92.8%</td>
<td>74.5%</td>
</tr>
</tbody>
</table>

* Stochastic Gradient Descent
** Sequential Minimal Optimization
Efficiency of TSFinder

- **Training**
  - One-time effort
  - All steps: 11.7 minutes
  - Model graphs (230 classes): 0.6 MB

- **Classifying new class**
  - On average over 230 classes: 3 seconds
  - Graphs extraction dominates classification
Conclusion

- State-of-the-art of thread-safety documentation is poor
- **TSFinder** uses machine learning to infer documentation
- **TSFinder** infers thread safety documentation with accuracy of 94.5%