EXPERIENCE-BASED MODEL REFINEMENT
G. Holzmann
G. Holzmann

model checking

model

automatic translation

code
Singleton design pattern
+singleton +pattern

+singleton +pattern +bad

+singleton +pattern +hate

+singleton +pattern +evil

+singleton +pattern +danger

760 000

230 600
Difficult tests

Creating sub-classes is difficult

Code maintenance is harder

Hidden dependencies
Difficult tests

Creating sub-classes is difficult

Code maintenance is harder

EXTREMELY error prone

Hidden dependencies
Difficult tests

Creating sub-classes is difficult

Hidden dependencies

Code maintenance is harder

EXTREMELY error prone
class Singleton {
    private static Singleton instance = null;

    private Singleton() {}

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
  private static Singleton instance = null;

  private Singleton() {};

  public static Singleton getInstance() {
    if (instance == null) {
      instance = new Singleton();
    }
    return instance;
  }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}

class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {
    }

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```
It just works!
It just works!

... or not
Multithreading

singleton
Multithreading
Multithreading

singleton
Multithreading

singleton
Multithreading
Multithreading

singleton
Multithreading
class Singleton {
    private static Singleton instance = null;

    private Singleton() {
    }

    public static synchronized Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
A good solution

class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static synchronized Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            synchronized (this) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
}
The "Double-Checked Locking is Broken" Declaration

David Bacon et al.
Do we modify the model...
Do we modify the model... again?
Do we modify the model... again?

The transformation is abstracting the bugs!
model

code

context
model

code

context

Language
VM / Platform
OS
Hardware
Cosmic rays
...

17
Language
VM / Platform
OS
Hardware
Cosmic rays
...

model

code

context
“Relevant correctness requirements”

G. Holzmann
Relevant correctness requirements

In-vivo and in-vitro executions

G. Candea

G. Holzmann
Fool me once, shame on you;
fool me twice, shame on me
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```
```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

    public static Singleton getInstance() {
        if (instance == null) {
            synchronized(this) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
}
```
<table>
<thead>
<tr>
<th>Risk involved</th>
<th>Program</th>
<th>Expected behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multithreading</td>
<td>![Cup of Coffee]</td>
<td>Multiple singletons</td>
</tr>
<tr>
<td>Multithreading</td>
<td>![Cup of Coffee]</td>
<td>Delays</td>
</tr>
<tr>
<td>Execution in Windows</td>
<td>![Cup of Coffee]</td>
<td>OS crash</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Risk involved</td>
<td>Program</td>
<td>Expected behavior</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Multithreading</td>
<td><img src="image" alt="Coffee" /></td>
<td>Multiple singletons</td>
</tr>
<tr>
<td>Multithreading</td>
<td><img src="image" alt="Coffee" /></td>
<td>Delays</td>
</tr>
<tr>
<td>Execution in Windows</td>
<td><img src="image" alt="Coffee" /></td>
<td>OS crash</td>
</tr>
</tbody>
</table>

Witnesses database
Recycling of Program Correctness Proofs

A. Podelski
Recycling of Program Correctness Proofs

“A crazy idea”
Witnesses database
Spec

Witnesses database

Execution context

Program
Select the relevant witnesses
Select the relevant witnesses

Define the Code2Model translation
Select the relevant witnesses
Define the Code2Model translation
Check the translation w.r.t the witnesses
Select the relevant witnesses  
Define the Code2Model translation  
Check the translation w.r.t the witnesses
Select the relevant witnesses

Define the Code2Model translation

Check the translation w.r.t the witnesses

Witnesses covered

Witness uncovered

Check the program
Select the relevant witnesses

Define the Code2Model translation

Check the translation w.r.t. the witnesses

Witness uncovered

Witnesses covered

Check the program
Select the relevant witnesses

Define the Code2Model translation

Check the translation w.r.t the witnesses

Check the program

Fix the program

Witnesses covered

Witness uncovered

Bug found

Spec

Witnesses database

Execution context

Program
Select the relevant witnesses

Define the Code2Model translation

Check the translation w.r.t. the witnesses

Witnesses covered

Check the program

No bugs found

Bug found

Fix the program

Witness uncovered

Program

Spec

Witnesses database

Execution context
Select the relevant witnesses

Define the Code2Model translation

Check the translation w.r.t. the witnesses

Witness uncovered

Witnesses covered

Check the program

No bugs found

Bug found

Fix the program

Spec

Witnesses database

Execution context

Program
Experience-based model refinement
Experience-based model refinement

Not automatic!
Thank you!