Robustness Anomalies:
A Catalogue

Eiji Adachi Barbosa

Alessandro Garcia

LES | DI | PUC-Rio – Brasil

OPUS Research Group

Robustness anomalies

- Robustness is the ability of a program to continue operating even in the presence of erroneous conditions (exceptions)
- Robustness anomalies are structures in the program that hamper proper handling of erroneous conditions
  - they might also simultaneously represent bugs, architectural violations (erosion), etc...

Exceptions

Unexpected conditions or events that occur at runtime and prevent programs to continue their normal execution flow

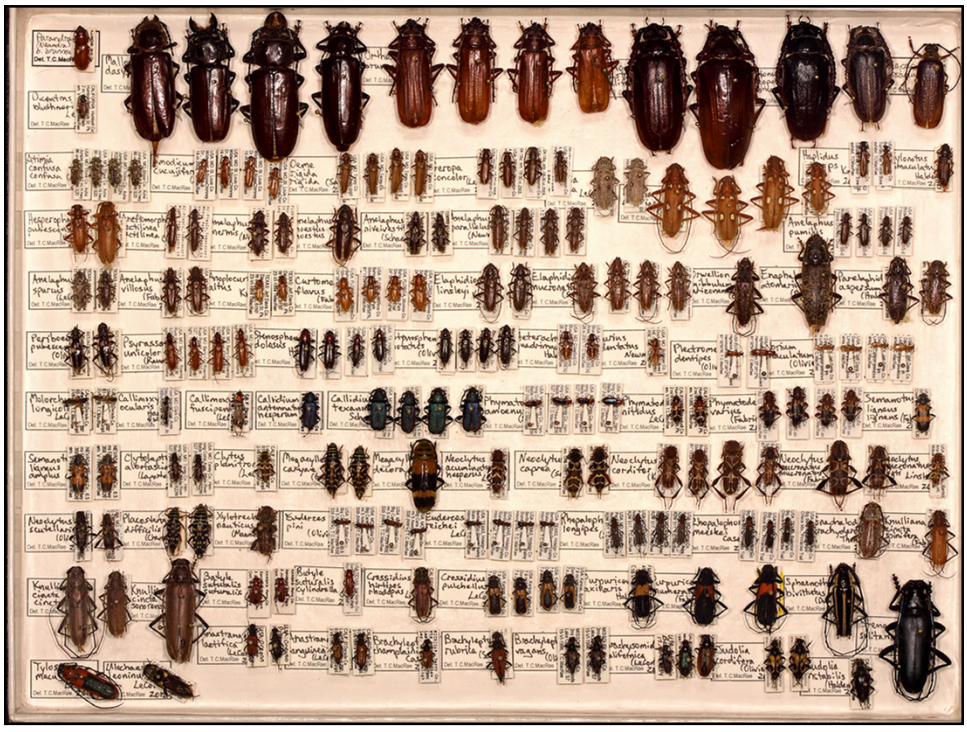
Press any key to continue _

Exception Handling Mechanisms

- **Intended to improve software robustness**
  - Ease the design and implementation of the exception handling in software systems
- Provided in most mainstream programming languages
  - Try – Catch – Throw

Problems with Exception Handling

- **Developers often neglect exception handling**
- The lack of proper handling actions is very common in real software systems
  - 40%-72% of handling actions are only-logging or empty handlers
- **Exception handling code** is more **error prone** and concentrates **more defects** than normal code
- **Developers introduce faults** in exception handling code even when they try to improve it



Study design

- Goal:
  - Gather deeper knowledge about the causes of faults related to exception handling reported in software systems
- To achieve our research goal, we performed a longitudinal evaluation in the context of two large software projects

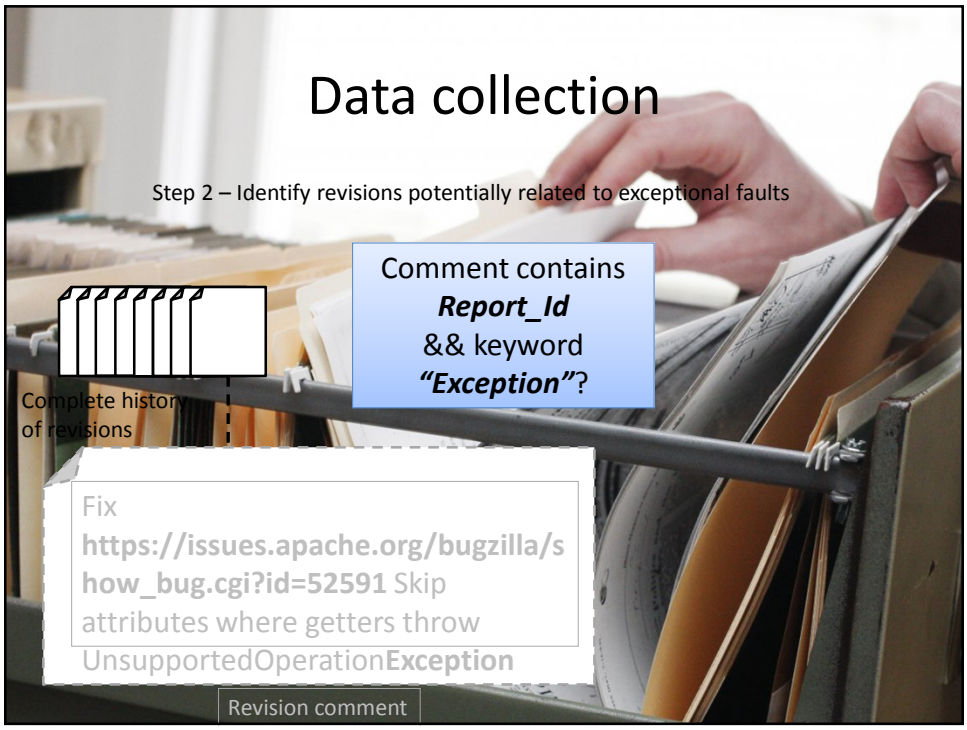
Target systems

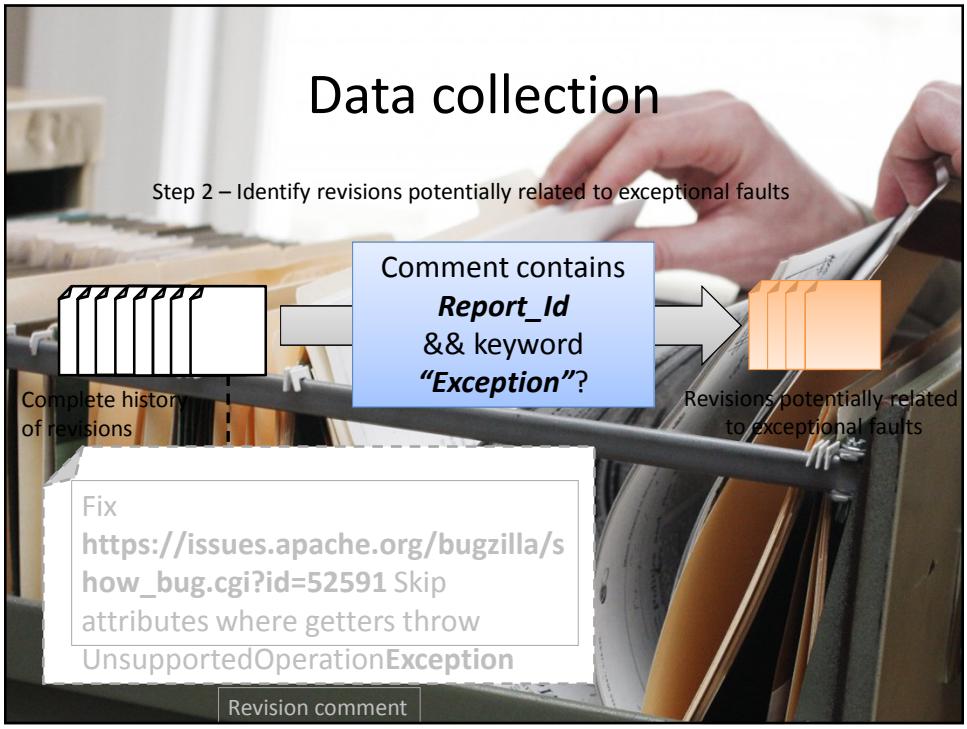
- Open-source projects
  - Publicly available bug reports and source code
- Possibility to map bug reports to corresponding fixing patches
  - *Bug-Report-Id-patterns* on commit logs
- Selected candidates:
  - Tomcat and Hadoop


Data collection
Step 1 – Gather complete history of revisions
Target system
source code repository


Data collection
Step 1 – Gather complete history of revisions
Target system
source code repository
Complete history of revisions

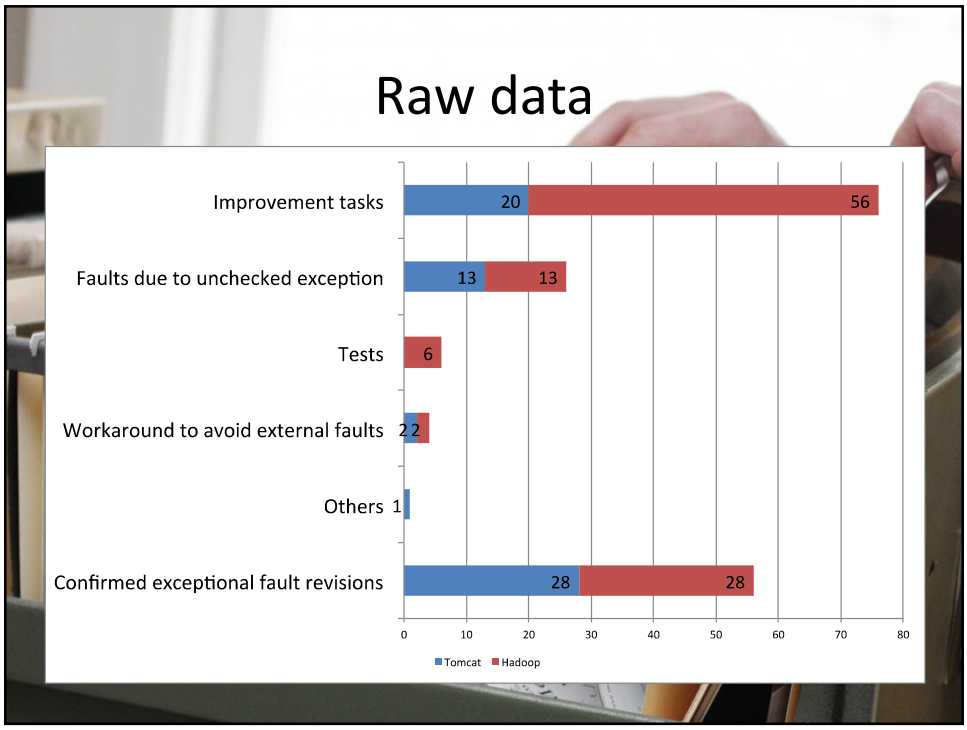

Data collection
Step 2 – Identify revisions potentially related to exceptional faults
Complete history of revisions
Fix https://issues.apache.org/bugzilla/show_bug.cgi?id=52591 Skip attributes where getters throw UnsupportedOperationException
Revision comment


Data collection
Step 2 – Identify revisions potentially related to exceptional faults
Comment contains Report_Id && keyword "Exception"?
Complete history of revisions
Fix https://issues.apache.org/bugzilla/show_bug.cgi?id=52591 Skip attributes where getters throw UnsupportedOperationException
Revision comment


Data collection
Step 2 – Identify revisions potentially related to exceptional faults
Comment contains
Report_Id
&& keyword
"Exception"?
Complete history of revisions
Revisions potentially related to exceptional faults
Fix **https://issues.apache.org/bugzilla/show_bug.cgi?id=52591** Skip attributes where getters throw UnsupportedOperation**Exception**
Revision comment

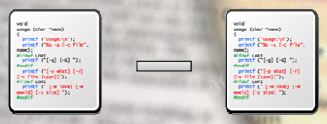

Data collection
Step 3 – Manual triage
Revision comment
Bug report
Revisions potentially related to exceptional faults
Source code diff


Data collection
Step 3 – Manual triage
Revision comment
Bug report
Source code diff
Revisions potentially related to exceptional faults
Revisions related to exceptional faults


Raw data
Improvement tasks
Faults due to unchecked exception
Tests
Workaround to avoid external faults
Others
Confirmed exceptional fault revisions
20
56
13
13
6
2
2
1
28
28
0
10
20
30
40
50
60
70
80
Tomcat
Hadoop

Analysis method

Step 1 – Diff description



V2

V1

Analysis method

Step 1 – Diff description

V2

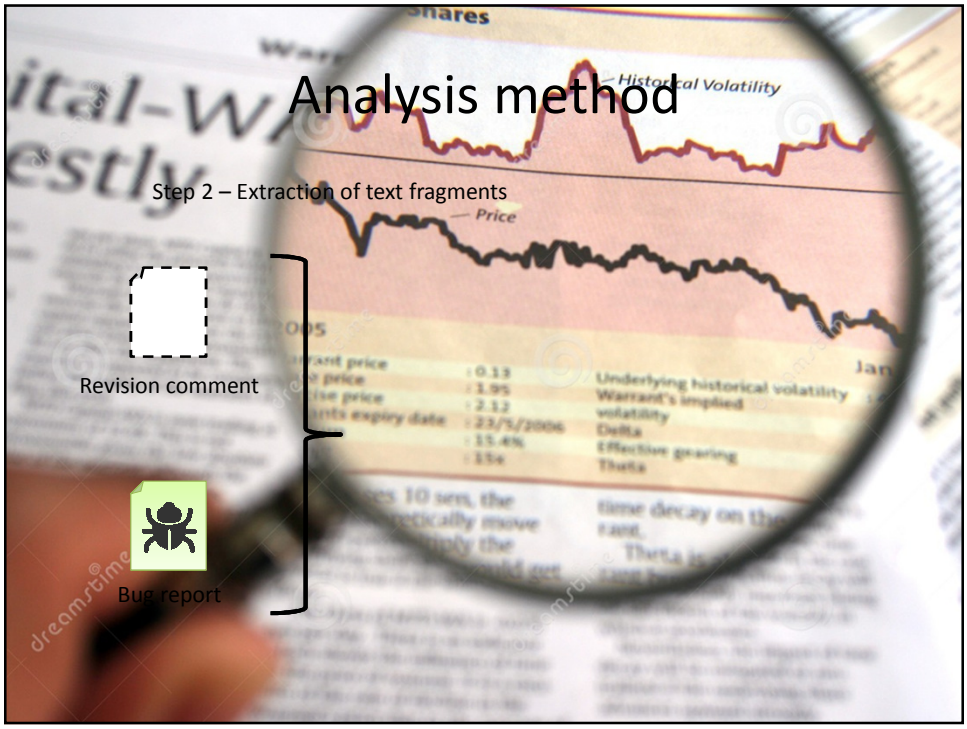
V1

"Changed the argument of a catch from InvalidProtocolBufferException to Throwable. Also changed parameters of exception instantiation within catch block."

Diff description

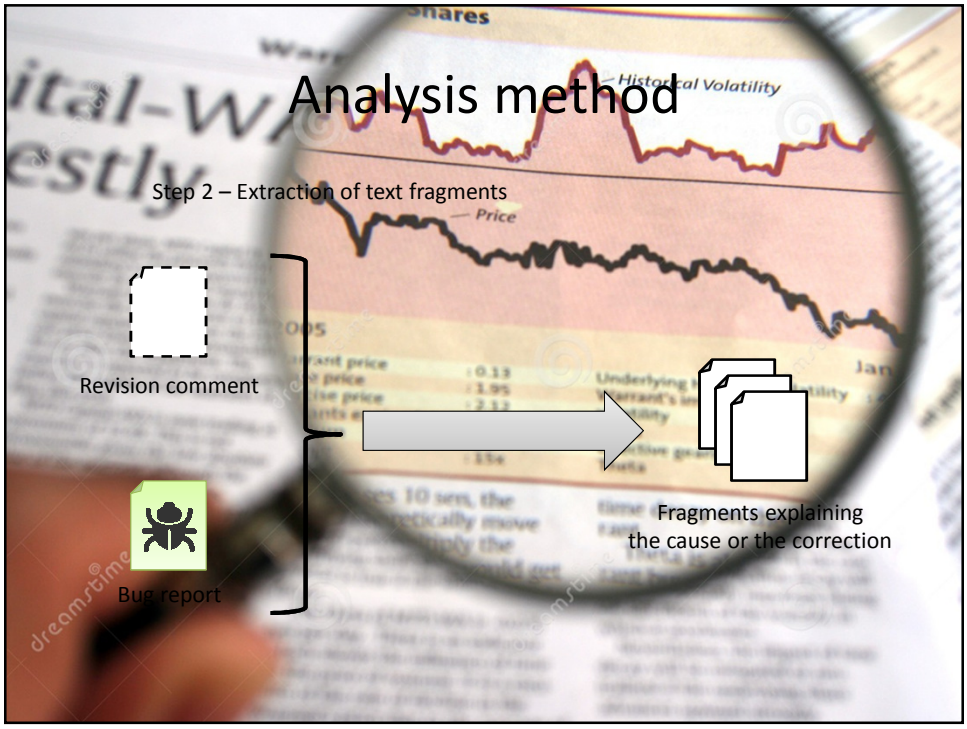
Analysis method

Step 2 – Extraction of text fragments



Analysis method

Step 2 – Extraction of text fragments



Analysis method

Step 2 – Extraction of text fragments

So, two conclusions:

1. The failure occurs above response.encodeURL("j_security_check")" call.

2. I suspect that _jspx_page_context is null. In that case the Throwable in the catch block is silently swallowed.

Example of fragment extracted from bug report

Analysis method

Step 3 – Explaining the cause

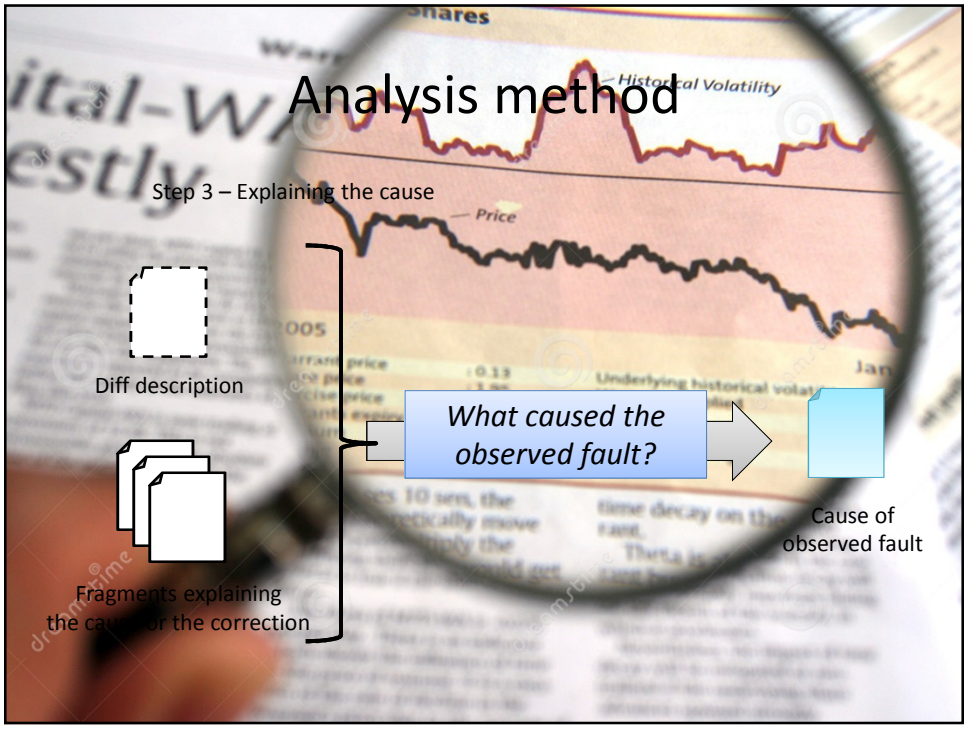
Diff description

What caused the observed fault?

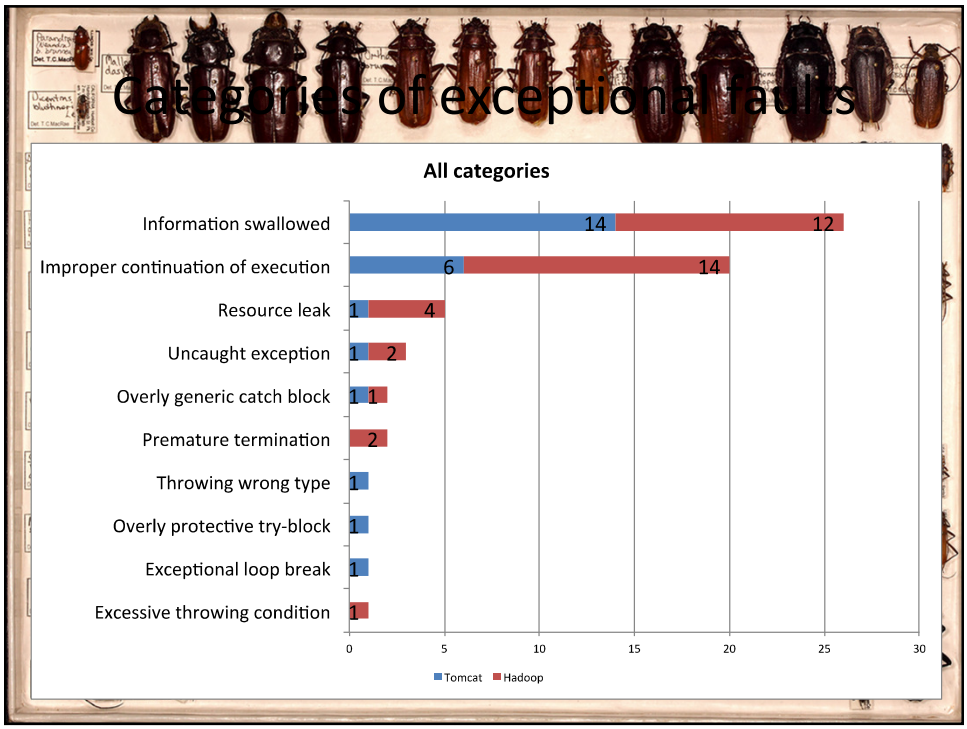
Fragments explaining the cause or the correction

Analysis method

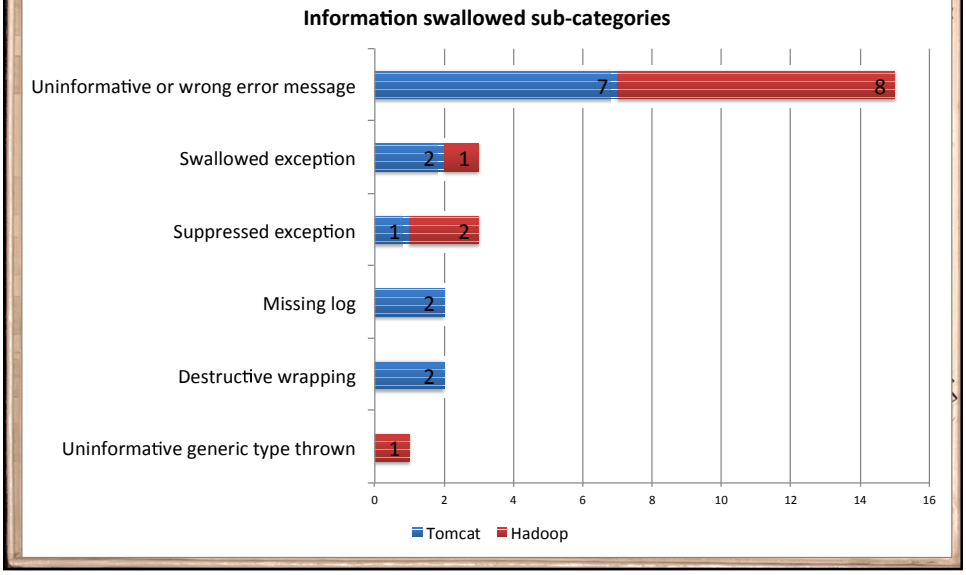
Step 3 – Explaining the cause



Categories of exceptional faults



Categories of exceptional faults



Category: Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
       closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
       closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
       closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
       closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
       closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
        closeStream( out );
    }
}
```

Category – Suppressed exception

```
public void copyBytes() throws IOException{
    try{
        //manipulate bytes
    }
    finally {
        closeStrea
    }
}
```

Original exception does not flow out of the method
AND
Exception raised on the finally block does not contain information about the original exception

Category – Destructive remapping

```
public void foo(){
    try{
        // do something
    }
    catch ( IOException e ){
       String s = e.getMessage();
       throw new BusinessException( s );
    }
}
```

Category – Destructive remapping

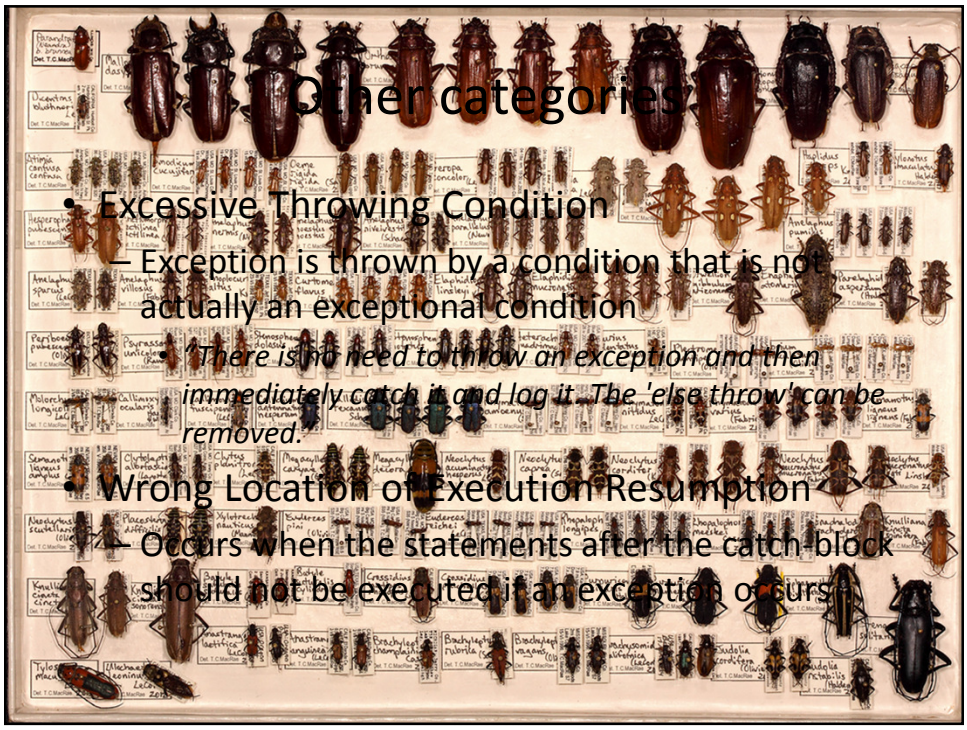
Extracted from bug report:

"(The error is) the fact that the catch block that logs the exception is swallowing the original exception and (its) stack trace."

```
public vo
    try{
    }
    catch
       String s = e.getMessage();
       throw new BusinessException( s );
    }
}
```

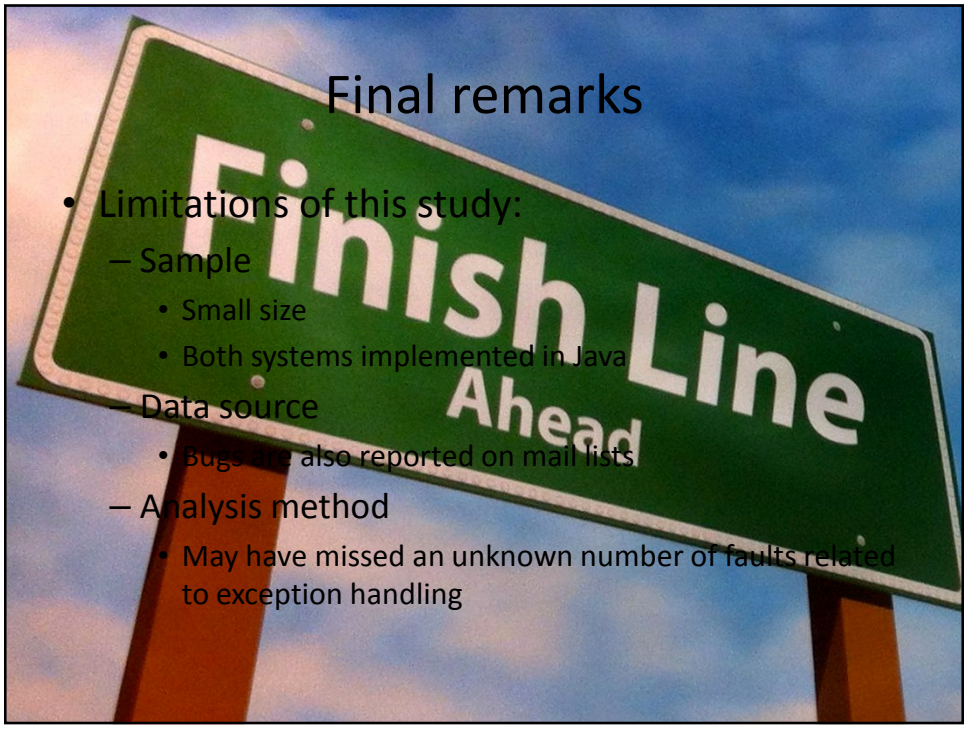
Other categories

- Premature termination
  - Catch block terminates without retrying
    - *"I now think the best solution is catch all IOExceptions and then retry once."*
- Overly protective try-block
  - Try-block is very long and inadvertently guards the occurrence of many different exceptions.
    - *"I believe the scope for which the FileNotFoundException" block applies is too great."*



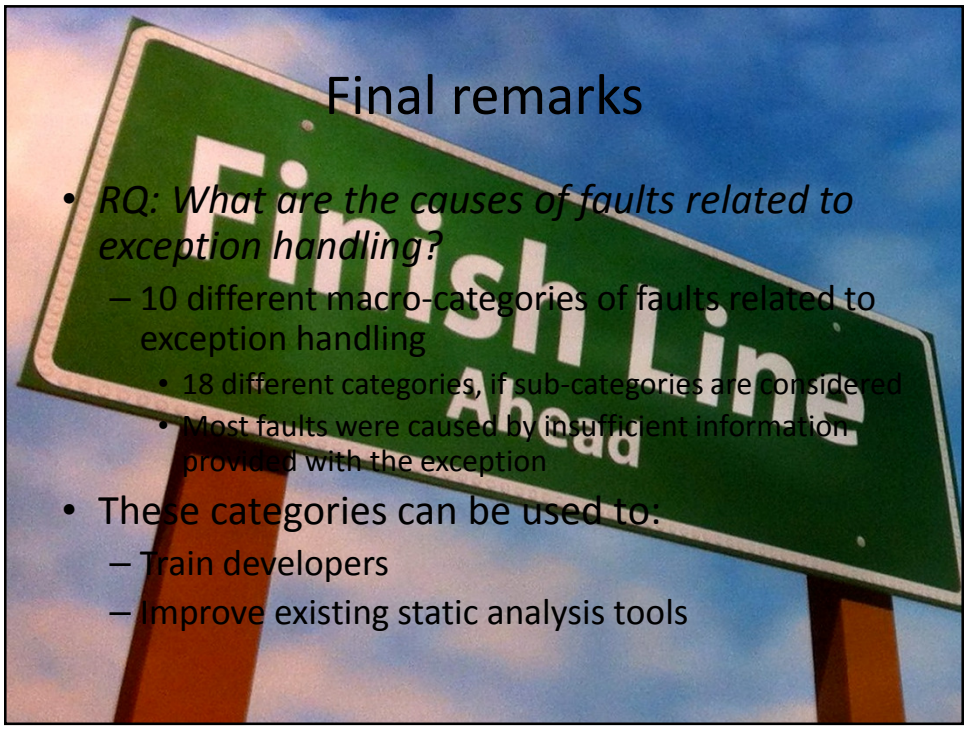
Other categories

- Excessive Throwing Condition
  - Exception is thrown by a condition that is not actually an exceptional condition
    - *"There is no need to throw an exception and then immediately catch it and log it. The 'else throw' can be removed."*
- Wrong Location of Execution Resumption
  - Occurs when the statements after the catch-block should not be executed if an exception occurs.



Final remarks

- Limitations of this study:
  - Sample
    - Small size
    - Both systems implemented in Java
  - Data source
    - Bugs are also reported on mail lists
  - Analysis method
    - May have missed an unknown number of faults related to exception handling



Final remarks

- *RQ: What are the causes of faults related to exception handling?*
  - 10 different macro-categories of faults related to exception handling
    - 18 different categories, if sub-categories are considered
    - Most faults were caused by insufficient information provided with the exception
- These categories can be used to:
  - Train developers
  - Improve existing static analysis tools

Final remarks

- Future:
  - Replicate this study with other systems, programming languages, etc
    - We invite you to replicate our study and contribute with new fault categories
  - Study how faults are corrected