Testing of Java Web Services for Robustness

Chen Fu, Ana Milanova,
David Wonnacott, Barbara Ryder
Availability of Internet Services

- Internet Service: New Kid in 24x7 domain
  - Public Telephone System: 99.999%
  - Internet Services: 99% ~ 99.9%

- Why?
  - Hardware:
    - Heterogeneous Cluster-based, complex system
  - Software
    - Short lifecycle caused by market pressure.
    - Components from various vendors.
  - Faults are unavoidable (Disk/Network/OS)
Fault Injection

Motivation:
- Redundant components are used to mask individual faults.
  - But would the software be able to take advantage of that?
- Testing program reaction to hardware/software problems
  - Disk Crash, Network congestion, OS resources depletion, OS bug ...
- Waiting for real fault to see the reaction of the system?
  - Actual Problems happen in rare basis

Solution
- Special software components to simulate “faulty conditions”.
Fault Injection – Current Approach

- Stochastic process
  - Distribution – data from real system:
- Fault Coverage:
  - Probability that a fault will be handled correctly

Fault Injection Engine

![Diagram]

- Application
- Java Runtime
- OS
- Device

s.read()
Fault Injection – White box test?

```java
try{
    ...
    process(...) ...
} catch (IOException e){
    // recovery code
}
```

Fault Injection Engine

Application
Java Runtime
OS
Device
try {
  ...
} catch (IOException e) {
  ...
}

try {
  ...
  ...
} catch (IOException e) {
  ...
}

try {
  ...
} catch (IOException e) {
  ...
}

try {
  ...
  ...
} catch (IOException e) {
  ...
}

read()/receive():

Diskwrite:
Coverage Metric

Exception Def-Catch Coverage is: \( \frac{|E|}{|F|} \)

- **Static**
  - \( F \) – Set of possible e-c links (starting from a set of fault-sensitive operations)

- **Dynamic**
  - \( E \) – Set of e-c links that are actually experienced in a set of test runs (\( E \cap F \))
Framework

Compile time

Tester provided Fault set

Java Application

Exception-Catch Link Analysis

Possible E-C links

Run time

Fault Injector-Mendosus

Instrumented Java Program

Observed E-C links

Measured Exception Def-Catch Coverage
Analysis: Finding e-c links

Exception-Flow Analysis → possible e-c links

DataReach Filter → possible e-c links
Exception-flow Analysis

Finding e-c links

```java
void foo() throws Exception {
    ...
    try {
        bar();
    } catch (IOException ioe) {...}
}
```

Set of throws can be handled here?

Set of throws that can reach `bar()` without being handled?

ReachingThrown
Exception-flow Analysis

```java
void foo() throws Exception {
    ...
    try{
        bar();
    } catch (IOException ioe) {...}
}

void bar() throws Exception {
    ...
    throw new SocketException();
    ...
    throw new OtherException();
    ...
}
```

\[ RT(j) = \bigcup_{t \in T} (\text{gen}(t) \sqcap \text{kill}(\text{trynest}(t))) \bigcup_{cs \in CS} \bigcup_{m \in \text{target}(cs)} (RT(m) \sqcap \text{kill}(\text{trynest}(cs))) \]
Exception-flow Analysis

- Dataflow Problem defined on call graph (backward)
- Varies call graph algorithm can be used:
  - CHA, RTA, Points-To (context insensitive, context sensitive)

SocketException thrown in bar

catch (IOException ioe) in foo

Call Chain
void readFile(String s){
    byte[] buffer = new byte[256];
    try{
        InputStream f =new FileInputStream(s);
        InputStream source=new BufferedInputStream(f);
        for (...) 
            c = source.read(buffer);
    }catch (IOException e){ ...}
}

void readNet(Socket s){
    byte[] buffer = new byte[256];
    try{
        InputStream n =s.getInputStream();
        InputStream source=new BufferedInputStream(n);
        for (...) 
            c = source.read(buffer);
    }catch (IOException e){ ...}
Data-Reach ___ Motivation

readFile

FilterInputStream.read(byte[])
BufferedInputStream.read(byte[],int,int)
BufferedInputStream.fill()

FileInputStream.read(...)

readNet

SocketInputStream.read(...)
BufferedInputStream.read1(byte[],int,int)
Feasibility of a call chain

Objects Reaching \( t \):
1. Through \( s \) and \( b \)
   field loads in “reachable methods”
2. Through global variables accessed in
   “reachable methods”
   field loads in “reachable methods”
3. Created in “reachable methods”

Solution
Collect these objects using Points-to Graph
If none of them has appropriate type
\( \Rightarrow \) infeasible
try{
    ... inject_fault();
    process(...)
    ...
    cancel_fault();
} catch (IOException e){
    record_current_fault();
    // recovery code
}
## Benchmarks

<table>
<thead>
<tr>
<th>Name</th>
<th>Classes</th>
<th>Methods</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPD</td>
<td>11(1407)</td>
<td>128(7479)</td>
<td>2783</td>
</tr>
<tr>
<td>JNFS</td>
<td>56(1664)</td>
<td>447(9603)</td>
<td>10478</td>
</tr>
<tr>
<td>Haboob</td>
<td>338(1403)</td>
<td>1323(7432)</td>
<td>39948</td>
</tr>
<tr>
<td>Muffin</td>
<td>278(1365)</td>
<td>2080(7677)</td>
<td>32892</td>
</tr>
</tbody>
</table>
Configurations

- CHA
- RTA
- PTA
- InPTA
- PTA-DR
- InPTA-DR

Exception-Flow Analysis

possible e-c links

DataReach Filter

possible e-c links
Coverage

- FTPD
- JNFS
- Haboob
- Muffin

Bars for each application represent coverage percentages across different models:
- CHA
- InPTA
- RTA
- PTA-DR
- InPTA-DR
<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Inline</th>
<th>PTA</th>
<th>Exception-Flow</th>
<th>DataReach</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPD</td>
<td>3.9</td>
<td>56.7</td>
<td>15.4</td>
<td>144.6</td>
<td>220.6</td>
</tr>
<tr>
<td>Haboob</td>
<td>4.2</td>
<td>53.7</td>
<td>12.1</td>
<td>155.8</td>
<td>221.6</td>
</tr>
<tr>
<td>JNFS</td>
<td>4.2</td>
<td>67.7</td>
<td>16.8</td>
<td>71.7</td>
<td>160.4</td>
</tr>
<tr>
<td>Muffin</td>
<td>6.7</td>
<td>62.2</td>
<td>12.5</td>
<td>97.2</td>
<td>171.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2,419.2</td>
<td>3,103.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3,490.4</td>
<td>3,961.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2,618.0</td>
<td>2,954.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,581.3</td>
<td>4,907.7</td>
</tr>
</tbody>
</table>
Thanks!
Configurations -- CHA

AST

CHA → Call Graph

Exception-Flow Analysis

possible e-c links
Configurations -- RTA

- AST
- Call Graph
- Exception-Flow Analysis
- possible e-c links
Configurations -- PTA

- AST
- Call Graph
- Exception-Flow Analysis
- Context Insensitive Points-to Analysis

possible e-c links
Configurations -- PTA-DR

- AST
- Call Graph
- Context Insensitive Points-to Analysis
- Point-to Graph

Exception-Flow Analysis

possible e-c links

DataReach Filter

possible e-c links
Configurations -- InPTA

AST

Call Graph

Context Insensitive Points-to Analysis

Constructor Inlining

Exception-Flow Analysis

possible e-c links
Configurations -- InPTA-DR

- AST
- Call Graph
- Context Insensitive Points-to Analysis
- Constructor Inlining
- Exception-Flow Analysis
- DataReach Filter
- possible e-c links