Visualization of Exception Handling Constructs to Support Program Understanding

Hina Shah

Carsten Görg and Mary Jean Harrold

Supported by NSF under CCR-0205422, CCF-0429117, CCF-0541049, and CCF-0725202, and Tata Consultancy Services, Ltd.

Presenter supported by SIGSOFT CAPS
(patents pending)
## Existing Techniques

### Analyses
- simplify exception structure [Robillard and Murphy 2000]
- identify exception flow [Sinha et al. 2004]
- identify re-thrown exception chains [Fu and Ryder 2007]

### Visualizations
- ExPo: throw-catch pairs as flow graph. [Sinha et al. 2004]
- EXTEST: exception propagation and navigation path as tree [Fu, Ryder 2005]
- ExceptionBrowser: exception propagation as tree [Chang et al. 2002]

### Remarks
- focus on improving analysis techniques
- provide no visual representation
- focus on low-level abstraction
- provide no system-wide view
- provide no context
Existing Techniques

Goal
Create Visualization that provides
- different perspectives for viewing exception-handling information
- at different levels of detail
- along with context information

Result
Visualization with three views
- shows static-analysis information of exception-handling constructs
- at different levels of detail
- along with context
Outline

- Exception handling in Java
- Survey
- Visualization
- Evaluation
- Conclusion
Outline

- Exception handling in Java
- Survey
- Visualization
- Evaluation
- Conclusion
Exception Handling in Java

Exception Handling constructs
- exception types
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
MyException e = new MyException();
try{
    ...
    if(flag==null){
        throw e;
    }else{
        // normal execution
    }
    // normal execution
}
catch(MyException e){
    // recovery code
}
finally{
    // clean up code
}
```
Exception Handling in Java

Exception Handling constructs
- exception types
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
    MyException e = new MyException();
    try{
        ...
        if(flag==null){
            throw e;
        }else{
            // normal execution
        }
    }catch(MyException e){
        // recovery code
    }
    finally{
        // clean up code
    }
```
Exception Handling in Java

Exception Handling constructs
- exception *types*
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
MyException e = new MyException();
try{
   ...
   if(flag==null){
      throw e;
   }else{
      // normal execution
   }
   // normal execution
}
}catch(MyException e){
   // recovery code
}
finally{
   // clean up code
}
```
Exception Handling in Java

Exception Handling constructs
- exception types
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
    MyException e = new MyException();
    try{
        ...
        if(flag==null){
            throw e;
        }else{
            // normal execution
        }
        // normal execution
    }catch(MyException e){
        // recovery code
    }
    finally{
        // clean up code
    }
```
Exception Handling in Java

Exception Handling constructs
- exception types
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
  MyException e = new MyException();
  try{
      ...
      if(flag==null){
        throw e;
      }else{
        // normal execution
      }
      // normal execution
  }
  catch(MyException e){
    // recovery code
  }
  finally{
    // clean up code
  }
}
```
Exception Handling in Java

Exception Handling constructs
- exception types
- try blocks
- throw statements
- catch blocks
- finally blocks

```java
method()
MyException e = new MyException();
try{
    ...
    if(flag==null){
        throw e;
    }else{
        // normal execution
    }
    // normal execution
}
}catch(MyException e){
    // recovery code
}
finally{
    // clean up code
}
```
Survey

Goal
- understand the needs of developers
- inform the design of our visualization system

Process
- created initial questionnaire design
- conducted pilot runs
- revised questionnaire based on feedback
- conducted the actual survey
  - demographics (34 participants)
  - exception-related questions
### Demographics (34 Participants)

#### Current Role
- Software Developer: 17 participants
- Project Manager: 1 participant
- Test Engineer: 5 participants
- Student: 13 participants

#### Professional Experience
- < 1 yr: 6 participants
- 1-2 yrs: 6 participants
- 2-3 yrs: 4 participants
- 4-5 yrs: 6 participants
- > 5 yrs: 12 participants

#### Project Size
- Small (> 5 KLOC): 7 participants
- Medium (5-50 KLOC): 19 participants
- Large (> 50 KLOC): 9 participants
## Demographics (34 Participants)

- **Current Role**
  - Software Developer: 17 participants
  - Project Manager: 1 participant
  - Test Engineer: 5 participants
  - Student: 13 participants

- **Professional Experience**
  - < 1 yr: 6 participants
  - 1-2 yrs: 6 participants
  - 2-3 yrs: 4 participants
  - 4-5 yrs: 6 participants
  - > 5 yrs: 12 participants

- **Project Size**
  - small (> 5 KLOC): 7 participants
  - medium (5-50 KLOC): 19 participants
  - large (> 50 KLOC): 9 participants
## Demographics (34 Participants)

<table>
<thead>
<tr>
<th>Current Role</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Developer</td>
<td>17</td>
</tr>
<tr>
<td>Project Manager</td>
<td>1</td>
</tr>
<tr>
<td>Test Engineer</td>
<td>5</td>
</tr>
<tr>
<td>Student</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Professional Experience</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1 yr</td>
<td>6</td>
</tr>
<tr>
<td>1-2 yrs</td>
<td>6</td>
</tr>
<tr>
<td>2-3 yrs</td>
<td>4</td>
</tr>
<tr>
<td>4-5 yrs</td>
<td>6</td>
</tr>
<tr>
<td>&gt; 5 yrs</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Size</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>small (&gt; 5 KLOC)</td>
<td>7</td>
</tr>
<tr>
<td>medium (5-50 KLOC)</td>
<td>19</td>
</tr>
<tr>
<td>large (&gt; 50 KLOC)</td>
<td>9</td>
</tr>
</tbody>
</table>
### Demographics (34 Participants)

- **Current Role**
  - Software Developer: 17 participants
  - Project Manager: 1 participant
  - Test Engineer: 5 participants
  - Student: 13 participants

- **Professional Experience**
  - < 1 yr: 6 participants
  - 1-2 yrs: 6 participants
  - 2-3 yrs: 4 participants
  - 4-5 yrs: 6 participants
  - > 5 yrs: 12 participants

- **Project Size**
  - small (> 5 KLOC): 7 participants
  - medium (5-50 KLOC): 19 participants
  - large (> 50 KLOC): 9 participants

---

Options:
- Exception
- Ha.
- Survey
- Visualization
- Evaluation
- Conclusion
Exception-Related Questions

Number of Participants

<table>
<thead>
<tr>
<th>Question</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is 'number of throw-catch pairs' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'exception concentration/distribution' information useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'exception-dependency information' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'number of exceptions of a type information' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'where exception originated' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'where exception caught' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'what type of exception occurred' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'complete exception flow' useful?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Yes
- No
- Maybe
Exception-Related Questions

Number of Participants

Is 'number of throw-catch pairs' useful?

Is 'exception concentration/distribution' information useful?

Is 'exception-dependency information' useful?

Is 'number of exceptions of a type' information useful?

Is visually representing 'where exception originated' useful?

Is visually representing 'where exception caught' useful?

Is visually representing 'what type of exception occurred' useful?

Is visually representing 'complete exception flow' useful?

Yes  No  Maybe
## Exception-Related Questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
<th>Maybe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is 'number of throw-catch pairs' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'exception concentration/distribution' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'exception-dependency information' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is 'number of exceptions of a type information' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'where exception originated' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'where exception caught' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'what type of exception occurred' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is visually representing 'complete exception flow' useful?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Number of Participants

<table>
<thead>
<tr>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

### Contextual

- Is visually representing 'where exception originated' useful?
- Is visually representing 'where exception caught' useful?
- Is visually representing 'what type of exception occurred' useful?
- Is visually representing 'complete exception flow' useful?

### Quantitative

- Is 'number of throw-catch pairs' useful?
- Is 'exception concentration/distribution' information useful?
- Is 'exception-dependency information' useful?
- Is 'number of exceptions of a type information' useful?
## Exception-Related Questions

<table>
<thead>
<tr>
<th>Quantitative</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is 'number of throw-catch pairs' useful?</td>
<td>0 5 10 15 20 25 30</td>
</tr>
<tr>
<td>Is 'exception concentration/distribution' information useful?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is 'exception-dependency information' useful?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is 'number of exceptions of a type information' useful?</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contextual</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is visually representing 'where exception originated' useful?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is visually representing 'where exception caught' useful?</td>
<td>No</td>
</tr>
<tr>
<td>Is visually representing 'what type of exception occurred' useful?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is visually representing 'complete exception flow' useful?</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Examples
• which type of exception reaches which throw statement
• which throw statement reaches which catch block
Visualization

Color Scheme
- Packages
- Classes
- Methods
- Exception-handling constructs

Example
- NanoXML
  - 2700 LOC
  - 3 packages
  - 5 classes
  - 85 methods

(Downloaded from: http://nanoxml.sourceforge.net/original)
Quantitative View

Exception Ha..  |  Survey  |  Visualization  |  Evaluation  |  Conclusion
Quantitative View - NanoXML

The diagram shows the number of throw-catch pairs at the PACKAGE level for different libraries and configurations. The library configurations are:

- (default)
- nanoxml
- nanoxml.sax

The count is 11 pairs, with the throw in nanoxml and the catch in nanoxml.sax.
method()
MyException e = new MyException();
    try{
        ...
        if(flag==null){
            throw e;
        }else{
            // normal execution
        }
        // normal execution
    }
}catch(MyException e){
    // recovery code
}
finally{
    // clean up code
}
method() {
    MyException e = new MyException();
    try {
        ...
        if (flag == null) {
            throw e;
        } else {
            // normal execution
        }
    } catch (MyException e) {
        // recovery code
    }
    finally {
        // clean up code
    }
}
Flow View - NanoXML

XMLParseException

Exception Handling
Survey
Visualization
Evaluation
Conclusion
Contextual View - NanoXML
Three Visualizations
EnHanCe Visualization

Program Analysis tool
eptools [Sinha et al. 2000]

Program analysis tool X

Program analysis tool Y

Exception Handling Centric (EnHanCe) Visualization

Eclipse
EnHanCe Screenshot
Evaluation

Goals

Understand

- *current approach* - how developers deal with exception-handling constructs
- *visualization usefulness* - whether visualization will help better understand exception-handling

Process

- created interview guide
- conducted pilot study with three participants (1-4 years experience)
- conducted detailed study with eight participants (1-10 years experience)
Results

- Current approach
  - “ignore-for-now” approach
  - primarily for debugging
  - perceived as forced

- Visualization usefulness
  - flow view reveals interesting patterns
  - contextual view simplifies search tasks
  - quantitative view less clear
    - information about finally, rethrown exceptions absent
  - questions about scalability
Conclusion and Future Work

- Improvements
  - scalability
  - quantitative view redesign

- Enhancements
  - show *finally* related information
  - show rethrown *exceptions* information
  - show runtime exception-handling

- Additional studies
Contributions

1. Conducted survey to understand developers needs
2. Designed a visualization with three views
   - Quantitative View
   - Flow View
   - Contextual View
3. Implemented visualization as an Eclipse plugin → EnHanCe
4. Conducted studies to evaluate the visualization tool

QUESTIONS!
THANK YOU!
Extra --- EnHanCe Patterns

Patterns

- Type centric
  - Subgraph consisting of type definition (under consideration), reachable throws, and reachable catches

- Throw centric
  - Subgraph consisting of reachable type, throw statement (under consideration), and reachable catches

- Catch centric
  - Subgraph consisting of reachable type, reachable throws, and catch (under consideration)

Helps in testing and refactoring
Extra --- Quantitative View Concerns

• Quantitative view
  • is not intuitive (difficult to understand the information it is representing, without being explained); instruction info will help
  • Blue shades need to be changed
  • Confused with the columns and row reading