Software Change Classification using Hunk Metrics

Javed Ferzund, Syed Nadeem Ahsan, Franz Wotawa
Institute for Software Technology
Graz University of Technology, Graz, Austria
Contents

- Background
- Introduction
- Study Approach
- Results
- Conclusion
- Future Work
Software needs to be continuously changed for prolonged usability

Changes are required to:
1. fix bugs
2. add new functionality
3. enhance performance
4. improve compatibility

Bug: A programming mistake or error in code that results in program failure
Background

- Multiple developers make changes at different times
- Most of the time developers are not aware of each others code
- Changes are made in short time periods
- Sometimes these changes introduce bugs into the source code
Bug-Introducing changes can be extracted
Changes are made in small chunks of code called hunks
It is observed that not all hunks in a change contribute to bugs
Introduction

- Prediction of Bugs
  a) Granularity Levels
  b) Metrics Types
  c) Technique Types
Introduction

- Granularity Level
  1. Module
  2. File
  3. Class
  4. Method
  5. Change

- Metrics Types
  1. Process Metrics
  2. Product Metrics
  3. Combination
Introduction

- Technique Types
  - Statistical Methods
    1. Multiple Linear regression
    2. Logistic Regression
    3. Principal Component Analysis
  - Machine Learning Techniques
    1. Decision Tree
    2. Support Vector Machine
    3. Neural Networks
    4. Random Forests
    etc.
Introduction

- Software Repositories
  1. CVS
  2. SVN
- Bug Databases
  Bugzilla
- Metrics Tool
Study Approach

- SCM Repository
- Bug Database
- Change Extractor
- Log Extractor
- Bug Report Extractor
- Hunk Preprocessing
- Hunk Labeling
- Metrics Calculation
- Hunk Classification
- Fact Database
Study Approach

Figure: Log Information
Study Approach

Figure: Difference output
### Study Approach

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Revision</th>
<th>Stamp</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>1.38</td>
<td>(pmulet 21-Jul-04)</td>
<td>if (this.bootclasspath != null &amp;&amp; this.bootclasspath.size() != 0) {</td>
</tr>
<tr>
<td>111</td>
<td>1.6</td>
<td>(othomann 05-Sep-02)</td>
<td>/*</td>
</tr>
<tr>
<td>112</td>
<td>1.6</td>
<td>(othomann 05-Sep-02)</td>
<td>* Set the bootclasspath for the Eclipse compiler.</td>
</tr>
<tr>
<td>113</td>
<td>1.6</td>
<td>(othomann 05-Sep-02)</td>
<td>*/</td>
</tr>
<tr>
<td>115</td>
<td>1.38</td>
<td>(pmulet 21-Jul-04)</td>
<td>cmd.createArgument().setPath(this.bootclasspath);</td>
</tr>
<tr>
<td>116</td>
<td>1.29</td>
<td>(othomann 29-Sep-03)</td>
<td>} else {</td>
</tr>
<tr>
<td>117</td>
<td>1.38</td>
<td>(pmulet 21-Jul-04)</td>
<td>this.includeJavaRuntime = true;</td>
</tr>
</tbody>
</table>

**Figure:** Annotations for revision 1.65
Study Approach

<table>
<thead>
<tr>
<th>Line Diff</th>
<th>Line Code Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>52c52</td>
<td><code>this.attributes.log(AntAdapterMessages.getString( ant.jdtadapter.info.usingJDTCompiler ), Project.MSG_VERBOSE); //NON-NLS-1</code></td>
</tr>
<tr>
<td>62c62</td>
<td><code>if (!resultValue &amp;&amp; this.verbose) {</code></td>
</tr>
<tr>
<td>82c82</td>
<td><code>if (this.bootclasspath != null &amp;&amp; this.bootclasspath.size() != 0) {</code></td>
</tr>
<tr>
<td>87c87</td>
<td><code>cmd.createArgument().setPath(this.bootclasspath);</code></td>
</tr>
<tr>
<td>89c89</td>
<td><code>this.includeJavaRuntime = true;</code></td>
</tr>
<tr>
<td>92c92</td>
<td><code>Path classpath = new Path(this.project);</code></td>
</tr>
<tr>
<td>99c99</td>
<td><code>addExtdirs(this.extdirs, classpath);</code></td>
</tr>
</tbody>
</table>

Figure: Changed hunks in revision 1.38
Hun Metrics

- No. of Loops (NOL)
- No. of Function Calls (NOFC)
- No. of Conditions (NOCN)
- No. of Variable Declarations (NOV)
- No. of Assignments (NOA)
- No. of Relational Operators (NORO)
- No. of Null Statement (NON)
- No. of Logical Operators (NOLO)
- No. of Pointers (NOP)
- No. of Return Statements (NORS)
- No. of Arrays (NOAR)
- No. of Case Statements (NOCS)
Hunk Metrics

• No. of Break Statements (NOB)
• No. of Goto Statements (NOG)
• No. of Include Statements (NOI)
• No. of Define statements (NOD)
• No. of Structures (NOS)
• No. of Function Declarations (NOFD)
• No. of Classes (NOC)
• No. of Object Instantiations (NOO)
• No. of Imports (NOIP)
• No. of Inheritance Statements (NOIH)
• No. of Exception Handlers (NOE)
• No. of Throw statements (NOTH)
• No. of Assertions (NOAS)
Hunk Metrics

- Total Hunks (NOH)
- No. of Previous Buggy Hunks (NOBH)
Results

Accuracies

Accuracies using Random Forest and Logistic Regression

Classification Accuracy

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

Apache Eclipse Epiphany Evolution Mozilla-C Mozilla-J Nautilus PostgreSQL

Random Forests
Logistic Regression

Project

- Random Forests
- Logistic Regression
Results

- **Buggy Hunk Precision**

Buggy Hunk Precision using Random Forests and Logistic Regression

<table>
<thead>
<tr>
<th>Project</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>0.7</td>
</tr>
<tr>
<td>Eclipse</td>
<td>0.8</td>
</tr>
<tr>
<td>Epiphany</td>
<td>0.6</td>
</tr>
<tr>
<td>Evolution</td>
<td>0.7</td>
</tr>
<tr>
<td>Mozilla-C</td>
<td>0.8</td>
</tr>
<tr>
<td>Mozilla-J</td>
<td>0.7</td>
</tr>
<tr>
<td>Nautilus</td>
<td>0.6</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Legend:
- Random Forests
- Logistic Regression
Results

- Buggy Hunk Recall

Buggy Hunk Recall using Random Forests and Logistic Regression

![Bar chart showing recall for different projects](chart.png)

- Apache
- Eclipse
- Epiphany
- Evolution
- Mozilla-C
- Mozilla-J
- Nautilus
- PostgreSQL

Recall

Project
Conclusion

- Hunk classification approach works for smallest level of granularity
- Hunk metrics are easy to calculate
- Results are available immediately after a hunk is made
- NOCN, NOA, NOFC, NORS, NOBH and NOH are important predictors for buggy hunks
Conclusion

- Random forest is a better predictor than logistic regression
- Average accuracy > 80\%, average buggy hunk precision > 75\% and average buggy hunk recall > 65\%
Future Work

- Other machine learning methods
- Other metrics
Questions?

- Thanks for your attention...