A Survey on Metric of Software Complexity

Sheng Yu, Shijie Zhou
University of Computer Science and Technology of China
Paper’s Purpose

- Survey on the research and the development of the metric of software complexity

Survey Topics

- Lines of Code - length
- Halstead Complexity Metric - volume
- Cyclomatic Complexity Metric - structure
Definition

- **Software metric** is the measurement to quantify some characteristics or attributes of a software entity
  - Quality of source code
  - Development process
  - Quality of application

- **Metric of software complexity** focuses on the quality of source codes
Types of Software Complexity

- **Essential**
  - Problem software is trying to solve

- **Selecting**
  - Programming languages, the problem modeling methods and the software design methods

- **Incidental**
  - The quality of the involved implementer
Focus of Software Complexity

- Mange and reduce the *incidental* complexity
- Improve the development of the software
- The software itself
## Classification of Complexity by Software timeline

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Design</td>
</tr>
<tr>
<td>McCabe</td>
<td>*</td>
</tr>
<tr>
<td>Halstead</td>
<td></td>
</tr>
<tr>
<td>Lines of Code</td>
<td>*</td>
</tr>
<tr>
<td>Error Count</td>
<td></td>
</tr>
<tr>
<td>Object Oriented Class Metrics</td>
<td>*</td>
</tr>
<tr>
<td>Software Package Metrics</td>
<td>*</td>
</tr>
<tr>
<td>Cohesion</td>
<td>*</td>
</tr>
<tr>
<td>Coupling</td>
<td>*</td>
</tr>
</tbody>
</table>
## Classification of Complexity by Calculating Basis

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Target</th>
<th>Logic Structure</th>
<th>Source Codes</th>
<th>User Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>McCabe</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halstead</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Lines of Code</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Error Count</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Object Oriented Class Metrics</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Package Metrics</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cohesion</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coupling</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Classic Metrics of Software Complexity and Their Variations

- Lines of Code - length
- Halstead Complexity Metric - volume
- Cyclomatic Complexity Metric - structure
Lines of Code is based upon its relationship with bug density

<table>
<thead>
<tr>
<th>LOC</th>
<th>63</th>
<th>100</th>
<th>158</th>
<th>251</th>
<th>398</th>
<th>630</th>
<th>1000</th>
<th>&gt;1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug Density</td>
<td>1.5</td>
<td>1.4</td>
<td>0.9</td>
<td>0.5</td>
<td>1.1</td>
<td>1.9</td>
<td>1.3</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Advantages:
• Easy to understand
• Fast to count
• Independent of the programming language

Disadvantages:
• Ignores difficulty complexity of each line of code
• Ignores the structure of the code
• Ignores programmer ability
Classic Metrics of Software Complexity and Their Variations

- Halstead Complexity Metric – volume

- Volume: length of a program times the minimum number of bits necessary to represent the operators and operands

\[ V = N \times \log(n) \]

- Length \( N = n_1 \log(n_1) + n_2 \log(n_2) \)
- \( n_1 \) – distinct operators
- \( n_2 \) – distinct operands

Maurice Howard Halstead - Software Science
Classic Metrics of Software Complexity and Their Variations

- **Advantages**
  - Deep analysis not needed
  - Easy to calculate
  - Program language independent

- **Disadvantages**
  - Ignores complexity of control stream
  - Constant value used for programmers ability

- **Variations**
  - Weighted Halstead (WHCM)
  - Data and Control stream (DCM)
Classic Metrics of Software Complexity and Their Variations

- Cyclomatic Complexity Metric - structure
  
  - Represents the topological structure of the software
  - Measures the complexity of a module's decision structure graph
    - Number of linearly independent paths
  
  $V(G) = e - n + 2$
The Trend of the Metric of Software Complexity and Their Variations

- **Advantages**
  - Close relationship with the bug density
    - if $0 < \text{CCM} < 20$ then low bug density
    - else if $\text{CCM} \geq 100$ then 60% of new bug

- **Disadvantages**
  - Ignores complexity of data flow (Used with HCM)
  - Ignores complexity of nested code
  - Does not distinguish the complexities of different kinds of control flow.

- **Variation**
  - Pseudo-path metric model (PPMM) - Gives different control structures different weights $\sum V(P_i)$
# Comparison

<table>
<thead>
<tr>
<th>Metric</th>
<th>Usability</th>
<th>Effect</th>
<th>Popularity</th>
<th>Theoretical Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>Easy</td>
<td>Good</td>
<td>Wide</td>
<td>No</td>
</tr>
<tr>
<td>HCM</td>
<td>Medium</td>
<td>Good</td>
<td>Wide</td>
<td>No</td>
</tr>
<tr>
<td>WHCM</td>
<td>Medium</td>
<td>Bad</td>
<td>Narrow</td>
<td>No</td>
</tr>
<tr>
<td>DCM</td>
<td>Hard</td>
<td>Bad</td>
<td>Narrow</td>
<td>No</td>
</tr>
<tr>
<td>CCM</td>
<td>Medium</td>
<td>Good</td>
<td>Wide</td>
<td>Yes</td>
</tr>
<tr>
<td>PPMM</td>
<td>Medium</td>
<td>Bad</td>
<td>Narrow</td>
<td>No</td>
</tr>
</tbody>
</table>
Correlations

- Halstead Volume = 45 * LOC – 428
- CC = 0.22 * LOC – 1.9
- Software defect prediction (2 or 3)
- Software defect prediction metric varies by the application
Trend

- Merging several different metrics
- Combining metrics of software complexity with data mining techniques
End

- Questions/Comments