A study on the practicality of three complexity metrics
Software Metrics

- Software metrics is a function where the input is software data and the output is a value which decide how the given attribute affect the software.
Software Complexity

• A measure of the resources expended by a system while interacting with a piece of software to perform a given task. If the interacting system is a computer, then complexity is defined by the execution time and storage required to perform the computation. If the interacting system is a programmer, then complexity is defined by the difficulty of performing tasks such as coding, debugging, testing or modifying the software.
Software Complexity Cont.

- Two categories:
  - Design Complexity
  - Code Complexity
Usage of Code Complexity Metrics

- Predict critical information about reliability, portability, maintainability etc.

- Design and execute test suites.

- Provide continuous feedback throughout a software project.

- Reward good programmers.
Research Gap

- Measuring software complexity is still a vast research subject.

- Many approaches suggested over time:
  - Halstead metrics
  - McCabe cyclomatic complexity
  - McClure metrics etc.

- Only a few studies have been conducted to find out the applicability of proposed metrics.

- Practical applicability of a metric is normally assessed by checking whether it satisfies the five properties proposed by Briand et al.

- Practical applicability have not been checked from the perspective of the real world software engineers.
Research Objective

- Compare the results of three complexity metrics for ten Java programs:
  - Halstead’s metric
  - McCabe metric
  - Shao & Wangs’ metric

- With the help of real world software engineers, identify which metric is the most suitable metric that can be used in the current state-of-art.
Literature Review

- McCabe’s Cyclomatic Complexity (1976)
- Mayer’s Complexity Metrics (1977)
- Gilb’s Logical Complexity Metrics (1977)
- Halstead Software Science (1977)
- McClure’s Invocation Complexity Metrics (1978)
- Hansen’s Hybrid Complexity Measure (1978)
- M.R. Woodward and his companions Knot Measure (1979)
- Schneidewind and Hoffmann’s Empirical Study (1979)
Literature Review

- S.H. Kan’s Complexity Metrics (1980)
- Oviedo’s Model (1980)
- Belady & Evangelists’ System-Partitioning Measures (1981)
- Iyengar and his companions’ Metric (1982)
- Selig’s Complexity Metrics (1990)
Literature Review

• Moreau and Dominck (1990) – The number of messages and the level of inheritances.


• Dumke, Neumann, and Stoeffler (1992) – Metrics for system level, class level and the method level.

• J.Y. Chen and J.F. Liu (1993) – Method to measure the complexity, reusability, and attributes of a class of object-oriented software.
Literature Review

- Kapsu, Yeongil and Chisu (1995) – Complexity Model based on Shannon’s Entropy Concept


- Douce, Layzell & Buckley (1999) – Set of spatial measures of software complexity
Literature Review


• Sanjay Misra (2006) – Cognitive Weight Complexity Measure


• Benjapol Auprasert and Yachai Limpiyakorn (2008) - A method to compute the cognitive complexity measure using combinatorial rules

• Usha Chhillar and Shuchita Bhasin (2011) – Weighted Complexity measure
Methodology

- Understood and applied the 3 complexity metrics.

- Used ten freely available java programs (based on the object oriented concept) as the base:

- Compared the derived complexities against the complexity scores given by programming experts.
Results

- SW’s CFS is the best metric that can be used in the real world.

<table>
<thead>
<tr>
<th>Expert Ranking</th>
<th>MaCabe's CC Metric</th>
<th>Halsted's Metric</th>
<th>Shao and Wangs' Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spearman's Rank Correlation Coefficient</td>
<td>.752</td>
<td>.851</td>
<td>.863</td>
</tr>
<tr>
<td>Significance Level (p-value)</td>
<td>.012</td>
<td>.002</td>
<td>.001</td>
</tr>
</tbody>
</table>
Thank you!