CptS422 Software Engineering Principles II (Fall 2018) Course Project Topics

1. General Description

The project consists of building an Eclipse plugin to detect structural or lexical metrics. The project will thus be implemented in Java. The metrics must be implemented as Checkstyle checks (http://checkstyle.sourceforge.net/) by extending the Eclipse CheckStyle plugin (http://eclipse-cs.sourceforge.net/#!/).

2. Suggested Topics

Two suggested topics are provided below, each focusing on one of two kinds of metrics (structural/lexical).

2.1 Structural metrics:

- *Halstead Length* is the sum of the total number of operators and operand [1,2]
- *Halstead Vocabulary* is the sum of the number of unique operators and unique operands [1,2]
- *Halstead Volume* is the program length (N) times the $\log_2$ of the program vocabulary (n) [1,2] : Volume = N $\log_2$ n
- *Halstead Difficulty* is half of the unique operators multiplied by the total number of operands, divided by the number of distinct operators [1,2]
- *Halstead Effort* is the difficulty multiplied by the volume. Effort = DV. Effort was intended as a suggestion for how long code review might take [1,2]
- *Number of comments* [3]
- *Number of lines of comments* [3]
- *Number of looping statements* [3]
- *Number of operators* [3]
- *Number of operands* [3]
- *Number of expressions* [3]
- *Number of variable declarations* [4]
- *External Method references* are the number of methods called from an external class [4]
- *Local method references* are the number of methods called from the same class [4]
- *Number of casts*
- *Maintainability Index* measures how maintainable the source code is.
  The maintainability index is calculated as a factored formula consisting of Lines of Code, Cyclomatic Complexity and Halstead volume [5]:
  MI = 171 - 5.2 * log2(V) - 0.23 * G - 16.2 * log2 (LOC) + 50 * sin (sqrt(2.4 * CM)), where:

  - V = Halstead Volume
  - G = Cyclomatic Complexity
  - LOC = count of source Lines Of Code (SLOC)
  - CM = percent of lines of Comment (optional)

2.2 Lexical metrics [6,7]:

- Extreme contraction
- Inconsistent identifier use
- Meaningless terms
- Misspelling
- Odd grammatical structure
- Overloaded identifiers
- Useless type indication
- Whole-part
- Synonyms and similar identifiers
- Terms in wrong context
- No hyponymy/hypernymy in class hierarchies
- Identifier construction rules

[1] M. Halstead. Elements of software science. Elsevier New York, 1977.

[2] Posnett, D., Hindle, A., and Devanbu, P. T. (2011). A simpler model of software readability. In Proceedings of the Working Conference on Mining Software Repositories (MSR), pp. 73--82. URL: http://dl.acm.org/citation.cfm?id=1985454

[3] Buse, R. and Weimer, W. (2010). Learning a metric for code readability. IEEE Transactions on Software Engineering (TSE), vol. 36, pp. 546–558. URL: http://ieeexplore.ieee.org/document/5332232/

[4] Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6), pp. 476-493.

[5] Oman, P. and Hagemeister, J. (1992). Metrics for assessing a software system's maintainability. In Proceedings of the International Conference on Software Maintenance (ICSM), pp. 337-344. URL: http://ieeexplore.ieee.org/document/242525/

[6] Abebe, S. L., Haiduc, S., Tonella, P. and Marcus, A. (2009). Lexicon bad smells in software. In Proceedings of the Working Conference on Reverse Engineering (WCRE), pp. 95–99. URL: http://ieeexplore.ieee.org/document/5328733/

[7] Lexicon Bad Smells Wiki: http://selab.fbk.eu/LexiconBadSmellWiki/

3. Self-Proposed Topics

If you would like to work on a topic other than the two suggested above, you may propose your own project topic. However, for self-proposed topics, you need to inform and convince the instructor, before the Deliverable/Milestone 0 deadline, that: (1) the project can be realistically finished up to initial implementation before the Deliverable 1 deadline, and (2) the project will be readily testable (i.e., testing would not be too hard) and there will be significant amount of work for the testing phase (i.e., the testing would not be trivial).