Homework 4

Team 9 - John McFarren, Erica O'Kelly, Devila Bakrania, Matthew Monaco

"I pledge my honor that I have abided by the Stevens Honor System." - JM, EO, DB, MM

Summary: This assignment sets the team on a mission to source a software complexity analysis tool, and to use it to analyze the complexity of three different programs. The team sourced three programs from the past assignments of one of its members. The programs are called "hashit", "puzzleSolver", and "reciprocalCycles". The team compared the results from the tool with their intuition, and determined that for the three chosen programs, LOC alone is an adequate indicator of software complexity.

1. First, manually decide, by reading the code, which piece of software program you think is the most complex and why.

After reading the code, the team decided that the "puzzleSolver" program is probably the most complex, since it is about twice as large as the other two programs, in terms of LOC. Also, the puzzle solver program calls several libraries at the beginning of the code, more than any other programs, leading the team to think that having this many libraries would indicate higher complexity. Additionally, each chunk of code appears to have more lines with further indentation, like the content within the "else if" portions, suggesting that the code is more layered and therefore complex.

2. Run the selected complexity analysis tool(s) on the 3 programs, and collect the following metrics LOC, CC, CK OO metrics (if classes are used), MI – whichever ones the tools automatically calculate. Document any assumptions in using the metrics and Graph/ analyze results.

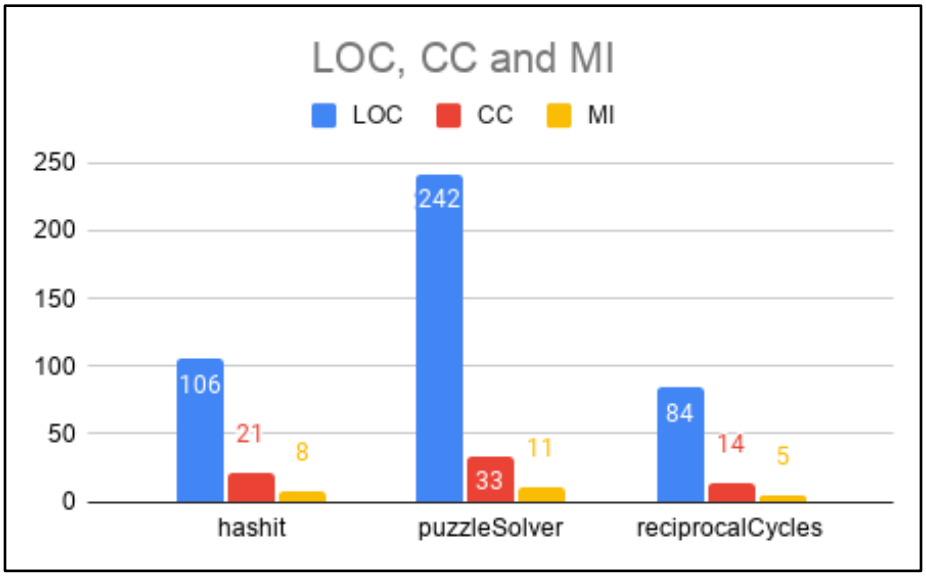
hashit:

LOC - 106
CC - 21
MI - 8

puzzleSolver:

LOC - 242
CC - 33
MI - 11

reciprocalCycles:



LOC - 84
CC - 14
MI - 5

When the results given by the complexity analysis tool (Metrix++) were graphed, the team noticed a consistency across the metrics. The program with the most lines of code had the highest cyclomatic complexity, so there appears to be some level of correlation between the two metrics. The team assumes that the program is ignoring all comment lines and empty lines within the software programs for its LOC calculations. The Maintainability Index score appears to be correlated with the other two metrics, and the team assumes that the program uses a simple variant of the Maintainability Index, employing a formula similar to Visual Studio's formula that we used in class. The Metrix++ website suggests that the MI calculation considers CC and LOC.

3. Compare the tool results from step 4 with your subjective judgment from Step 3. How do the tool results compare to your intuition? Are they consistent? What did you think of the code analysis tool? Good or not so Good? Why?

The tool results support the intuition we had in deciding which program would be the most complex. Our reasoning for step 3 on which program would be the most complex mainly relied on how long the program looked as well as how large the different regions within the program appeared. This rationale was proven correct when using the Metrix++ tool to analyze the program because from our analysis the more LOC for a program proved to have a greater CC and MI. We enjoyed using Metrix++ as our code analysis tool because we were able to choose which metric to analyze within the program allowing for a condensed output with only the information we needed. We did not have to scan a full analysis in order to find the metrics we needed for this assignment.

4. Based on your analysis, which of the following two viewpoints do you support? 1) LOC is adequate as an indicator of software complexity. 2) CC is more useful than LOC in measuring software complexity. Provide clear reasons for or against a view.

Based on the team's analysis, we would support the statement that LOC is an adequate indicator of software complexity. Each metric appeared to rise and fall alongside LOC, and given that the programs were all created by the same developer, the team has no reason to believe that this would be a coincidence or that it might be explainable by pointing to different coding styles used. The team could not obtain enough information from the analyzed programs to say definitively that CC is more useful than LOC in measuring complexity. From the reading, the

team gets the impression that CC often runs into issues involving its lack of standardization in counting rules. Because LOC is a simpler metric, it may be adequate for simpler programs, like those we analyzed.

Attachments:

```
StevensUser@DESKTOP-95C1RO8 ~/git_assignments/cs370/hashIt
$ python ../../../metrixplusplus-1.7.0/metrix++.py view
[LOG]: WARNING: Logging enabled with INFO level
[LOG]: INFO:    Processing: ./
./:: info: Overall metrics for 'std.code.complexity:cyclomatic' metric
        Average         : 3.5
        Minimum         : 1
        Maximum         : 7
        Total           : 21.0
        Distribution    : 6 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     1 : 0.333 : 0.333 : 2      |||||||||||||||||||||||||||||||||
                     2 : 0.167 : 0.500 : 1      ||||||||||||||||
                     5 : 0.333 : 0.833 : 2      |||||||||||||||||||||||||||||||||
                     7 : 0.167 : 1.000 : 1      ||||||||||||||||

./:: info: Overall metrics for 'std.code.lines:code' metric
        Average         : 13.25
        Minimum         : 1
        Maximum         : 32
        Total           : 106.0
        Distribution    : 8 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     1 : 0.125 : 0.125 : 1      ||||||||||||
                     4 : 0.125 : 0.250 : 1      ||||||||||||
                     6 : 0.125 : 0.375 : 1      ||||||||||||
                     8 : 0.250 : 0.625 : 2      |||||||||||||||||||||||||
                    20 : 0.125 : 0.750 : 1      ||||||||||||
                    27 : 0.125 : 0.875 : 1      ||||||||||||
                    32 : 0.125 : 1.000 : 1      ||||||||||||

./:: info: Overall metrics for 'std.code.mi:simple' metric
        Average         : 1.0
        Minimum         : 1
        Maximum         : 1
        Total           : 8.0
        Distribution    : 8 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     1 : 1.000 : 1.000 : 8      ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||||||||||||||||
```

```
StevensUser@DESKTOP-95C1RO8 ~/git_assignments/cs370/scrambleSquares
$ python ../../../metrixplusplus-1.7.0/metrix++.py view
[LOG]: WARNING: Logging enabled with INFO level
[LOG]: INFO:    Processing: ./
./:: info: Overall metrics for 'std.code.complexity:cyclomatic' metric
        Average         : 4.71428571
        Minimum         : 0
        Maximum         : 16
        Total           : 33.0
        Distribution    : 7 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     0 : 0.143 : 0.143 : 1      ||||||||||||||
                     1 : 0.143 : 0.286 : 1      ||||||||||||||
                     4 : 0.571 : 0.857 : 4      ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||
                    16 : 0.143 : 1.000 : 1      ||||||||||||||

./:: info: Overall metrics for 'std.code.lines:code' metric
        Average         : 26.88888889
        Minimum         : 3
        Maximum         : 68
        Total           : 242.0
        Distribution    : 9 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     3 : 0.111 : 0.111 : 1      |||||||||||
                     4 : 0.111 : 0.222 : 1      |||||||||||
                     5 : 0.111 : 0.333 : 1      |||||||||||
                     6 : 0.111 : 0.444 : 1      |||||||||||
                    33 : 0.111 : 0.556 : 1      |||||||||||
                    34 : 0.111 : 0.667 : 1      |||||||||||
                    40 : 0.111 : 0.778 : 1      |||||||||||
                    49 : 0.111 : 0.889 : 1      |||||||||||
                    68 : 0.111 : 1.000 : 1      |||||||||||

./:: info: Overall metrics for 'std.code.mi:simple' metric
        Average         : 1.22222222
        Minimum         : 1
        Maximum         : 3
        Total           : 11.0
        Distribution    : 9 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     1 : 0.889 : 0.889 : 8      ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||||||||||||||||||||||
                     3 : 0.111 : 1.000 : 1      |||||||||||

./:: info: Directory content:
        File            : puzzlesolver.cpp
```

```
StevensUser@DESKTOP-95C1RO8 ~/git_assignments/cs370/reciprocalCycles
$ python ../../../metrixplusplus-1.7.0/metrix++.py view
[LOG]: WARNING: Logging enabled with INFO level
[LOG]: INFO:    Processing: ./
./:: info: Overall metrics for 'std.code.complexity:cyclomatic' metric
        Average         : 4.66666667
        Minimum         : 3
        Maximum         : 7
        Total           : 14.0
        Distribution    : 3 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     3 : 0.333 : 0.333 : 1      |||||||||||||||||||||||||||||||||
                     4 : 0.333 : 0.667 : 1      |||||||||||||||||||||||||||||||||
                     7 : 0.333 : 1.000 : 1      |||||||||||||||||||||||||||||||||

./:: info: Overall metrics for 'std.code.lines:code' metric
        Average         : 16.8
        Minimum         : 2
        Maximum         : 32
        Total           : 84.0
        Distribution    : 5 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     2 : 0.400 : 0.400 : 2      |||||||||||||||||||||||||||||||||||||||||
                    23 : 0.200 : 0.600 : 1      |||||||||||||||||||||
                    25 : 0.200 : 0.800 : 1      |||||||||||||||||||||
                    32 : 0.200 : 1.000 : 1      |||||||||||||||||||||

./:: info: Overall metrics for 'std.code.mi:simple' metric
        Average         : 1.0
        Minimum         : 1
        Maximum         : 1
        Total           : 5.0
        Distribution    : 5 regions in total (including 0 suppressed)
          Metric value : Ratio : R-sum : Number of regions
                     1 : 1.000 : 1.000 : 5      |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||||||||||||||||

./:: info: Directory content:
        File            : ReciprocalCycles.java
```