UNIVERSITÉ Concordia UNIVERSITY

SOEN6611 Deliverable 2

Instructor: Dr. Pankaj Kamthan

Date: Aug. 05, 2012

Presented by: Team F

Team Member: Lu Ma, Meng Jia, Yuan Tao

Table of Contents

1. Part 1 3
1.1. WMC (Weighted Method Per Class) 3
1.1.1. Prime 3
1.1.2. Quadratic 4
1.1.3. Summary 5
1.2. CF (Coupling Factor) 5
1.2.1. Default package. 6
1.2.2. libs 6
1.2.3. ui 6
1.2.4. utils 7
1.2.5. Summery 7
1.3. LCOM*(LACK OF COHESION IN METHODS) 7
1.3.1. Prime 8
1.3.2. Quadratic 8
1.3.3. Summery 8
2. Part2 (Logical SLOC) 9
2.1. USC CodeCount: 9
2.1.1. MySqrt 9
2.1.2. Quardatic 10
2.1.3. Prime 10
2.2. LocMetrics 10
3. Part3 (Spearman’s Rank Correlation Coefficient) 12
3.1. Correlation between Quadratic and Prime classes 12
3.2. Correlation between three classes 12
3.2.1. WMC of MySqrt 13
3.3. Conclusion 14
Appendix A: Reference 15
Appendix B: Revision History 16

Table of Tables

Table 1-1 Attributes of Prime 8
Table 1-2 Attributes of Quadratic 8
Table 2-1 Overall metrics of MINSK 11
Table 2-2 LOC of classes 11
Table 3-1 Statistic between Quadratic and Prime 12
Table 3-2 Statistic between three classes 14

Table of Figures

Figure 1-1 WMC of Prime.. 4
Figure 1-2 WMC of Quadratic 5
Figure 1-3 UML diagram of libs package 6
Figure 1-4 UML diagram of ui package 7
Figure 3-1 WMC of MySqrt 13

1. Part 1

1.1. WMC (Weighted Method Per Class)

The equation to calculate WMC is:

$$\mathrm{WMC} = \sum_{i=1}^{\mathrm{n}} c_i(M_i)$$

We have to assume that the weight is not normalized. Therefore, the cyclomatic number of the methods suggests the value that can be used for Ci.

From last deliverable, we calculated the cyclomatic number for all the methods in 'Prime' and 'Quadratic' class. Here, we are going to make use of those calculated data.

1.1.1. Prime

There are totally two functions in the 'Prime' class.

1. checkInputs()

Cyclomatic number is 3. The weight for the checkInputs is set to be 3.

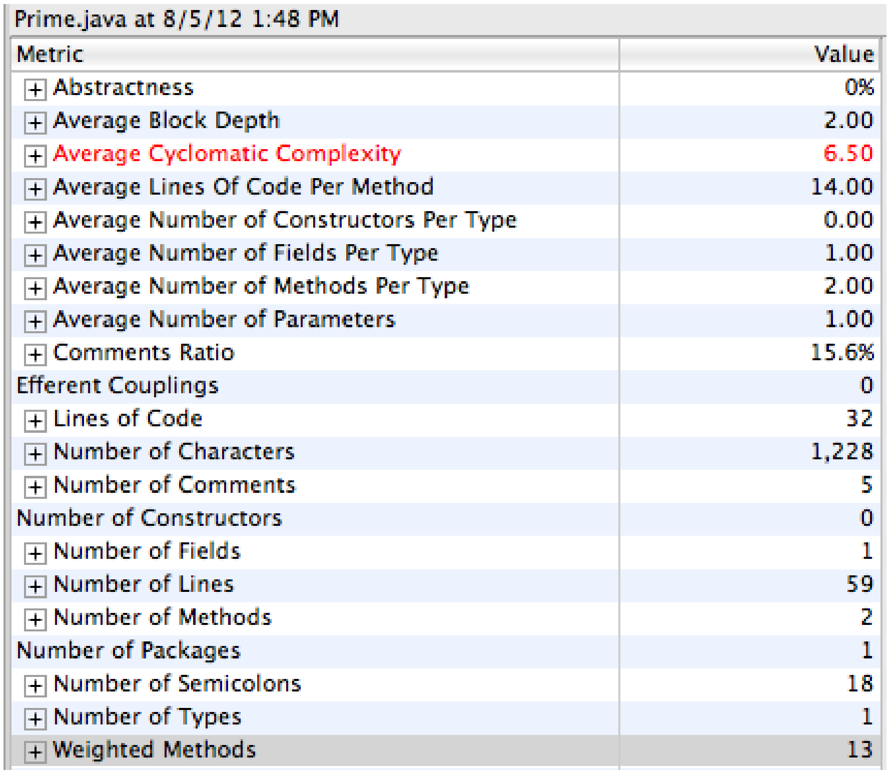
2. isPrime()

Cyclomatic number is 10. The weight for the isPrime is set to be 10.

WMC=10+3=13.

The result from CodePro[1]:

[1] https://developers.google.com/java-dev-tools/download-codepro



Prime.java at 8/5/12 1:48 PM

Metric	Value
⊞ Abstractness	0%
⊞ Average Block Depth	2.00
⊞ Average Cyclomatic Complexity	6.50
⊞ Average Lines Of Code Per Method	14.00
⊞ Average Number of Constructors Per Type	0.00
⊞ Average Number of Fields Per Type	1.00
⊞ Average Number of Methods Per Type	2.00
⊞ Average Number of Parameters	1.00
⊞ Comments Ratio	15.6%
Efferent Couplings	0
⊞ Lines of Code	32
⊞ Number of Characters	1,228
⊞ Number of Comments	5
Number of Constructors	0
⊞ Number of Fields	1
⊞ Number of Lines	59
⊞ Number of Methods	2
Number of Packages	1
⊞ Number of Semicolons	18
⊞ Number of Types	1
⊞ Weighted Methods	13

Figure 1-1 WMC of Prime

1.1.2. Quadratic

There are totally three functions in the ‘Quadratic’ class.

1. validation()

Cyclomatic number is 2. The weight for validation is set to be 2.

2. howManyRoots()

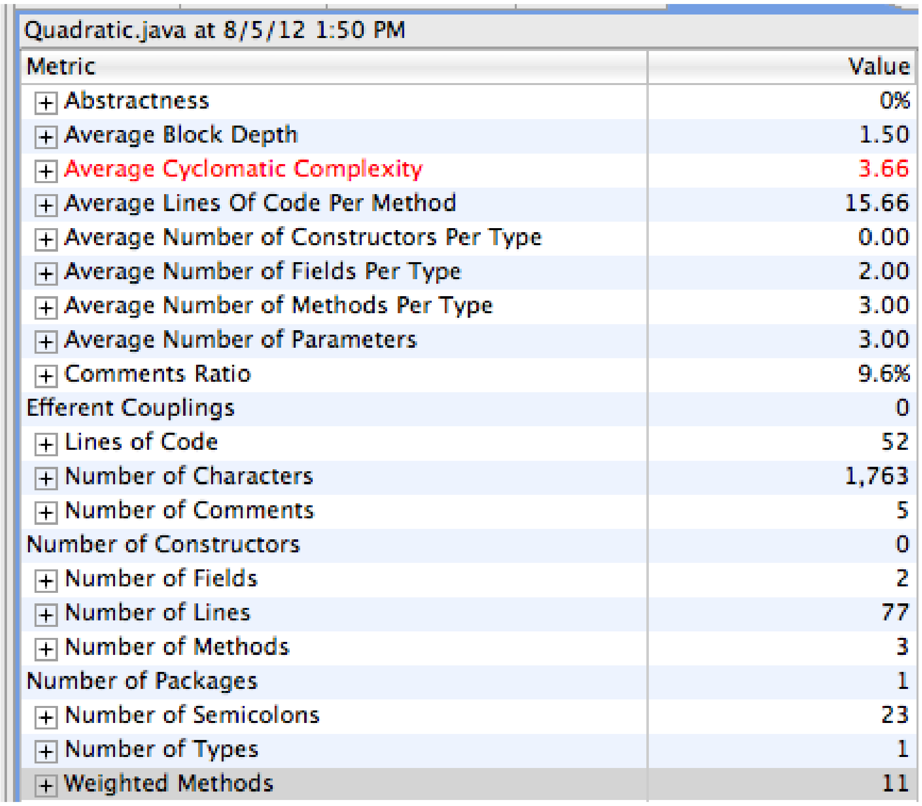
Cyclomatic number is 3. The weight for howManyRoots is set to be 3.

3. solve()

Cyclomatic number is 6. The weight for solve is set to be 6.

WMC=2+3+6=11.

Result from CodePro:



Quadratic.java at 8/5/12 1:50 PM

Metric	Value
⊞ Abstractness	0%
⊞ Average Block Depth	1.50
⊞ Average Cyclomatic Complexity	3.66
⊞ Average Lines Of Code Per Method	15.66
⊞ Average Number of Constructors Per Type	0.00
⊞ Average Number of Fields Per Type	2.00
⊞ Average Number of Methods Per Type	3.00
⊞ Average Number of Parameters	3.00
⊞ Comments Ratio	9.6%
Efferent Couplings	0
⊞ Lines of Code	52
⊞ Number of Characters	1,763
⊞ Number of Comments	5
Number of Constructors	0
⊞ Number of Fields	2
⊞ Number of Lines	77
⊞ Number of Methods	3
Number of Packages	1
⊞ Number of Semicolons	23
⊞ Number of Types	1
⊞ Weighted Methods	11

Figure 1-2 WMC of Quadratic

1.1.3. Summary

We want WMC to be low as it measures the size and the complexity of the class. The low value of WMC indicates the class is easy to maintain and reuse. In our project, the WMC for 'Prime' and 'Quadratic' is not very small, because those two classes involve a lot of logical determination and algorithm. We want to balance the algorithm efficiency and the program complexity.

1.2. CF (Coupling Factor)

The equation to calculate CF is:

$$\mathrm{CF} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\mathrm{IsClient}(C_i, C_j)\right)}{\mathrm{n}^2 - \mathrm{n}}$$

CF is one coefficient that associated with packages. We have four packages in our MINSK. Below is the CF calculation for each package.

1.2.1. Default package.

We have only one class in this package, which is MinskDriver. Therefore, there is no IsClient relationship inside this package. CF=0.

1.2.2. libs

This package contains our main logic of the MINSK project. We have five classes in this package. N=5.

CF = [3+2+1+0+0] / (5*5 – 5)

CF = 6 / 20

CF = 0.3

Here is the UML diagram of the package:

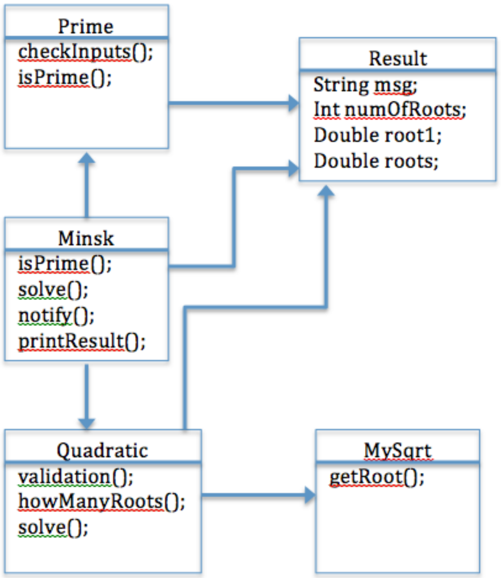


Figure 1-3 UML diagram of libs package

1.2.3. ui

This package contains two classes, that is n=2. UIBasicInf is a interface that UIconsole implements it. This relationship is not inheritance.

CF = [0+1] / (2*2 – 2)

CF = 1 / 2

CF = 0.5

Right is the UML diagram of the package:

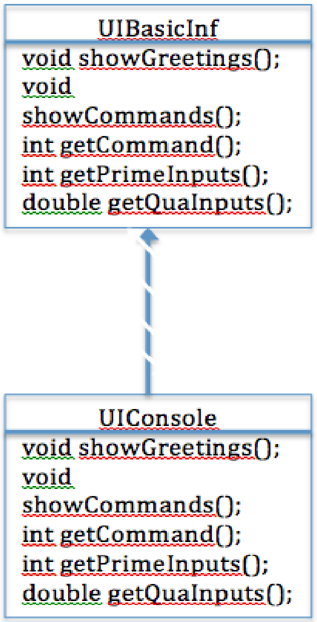


Figure 1-4 UML diagram of ui package

1.2.4. utils

We have only one class in this package, which is SysLogger. Therefore, there is no IsClient relationship inside this package. CF=0.

1.2.5. Summery

For each of our package, we have our coupling factor less than or equal to 0.5, which is not a large value. And our goal is to achieve a small number of CF. The CF value shows our project is easy to maintain.

1.3. LCOM*(LACK OF COHESION IN METHODS)

The equation to calculate CF is:

$$\mathrm{LCOM^*} = \frac{\frac{1}{\mathrm{a}}\left(\sum_{i=1}^{\mathrm{a}} \mu(A_i)\right) - \mathrm{m}}{1 - \mathrm{m}}$$

1.3.1. Prime

This class contains 2 methods. That m=2.
Below table is the summery of attributes:

Attribute Name:	Accessed by:
ret	isPrime
retTrue	isPrime
retFalse	isPrime

Table 1-1 Attributes of Prime

From above table, we can see that a=3, and each attribute is only accessed by one method. This situation is when LCOM* is the maximum.
LCOM*=1.

1.3.2. Quadratic

This class contains 3 methods. That is m=3.
Below table is the summery of attributes:

Attribute Name:	Accessed by:
mySqrt	solve
myRes	solve
delta	howManyRoots

Table 1-2 Attributes of Quadratic

From above table, we can see that a=3, and each attribute is only accessed by one method. This situation is when LCOM* is the maximum.
LCOM*=1.

1.3.3. Summery

Ideally, we want LCOM* to be low, but in our project, we have a very high value. Because that class ‘Prime’ and ‘Quadratic’ both don’t have a lot of methods. And each method does independent functions, which will increase the LCOM*.

2. Part2 (Logical SLOC)

In MINSK, there are three main classes which contain the functional algorithms.
There are three classes including:

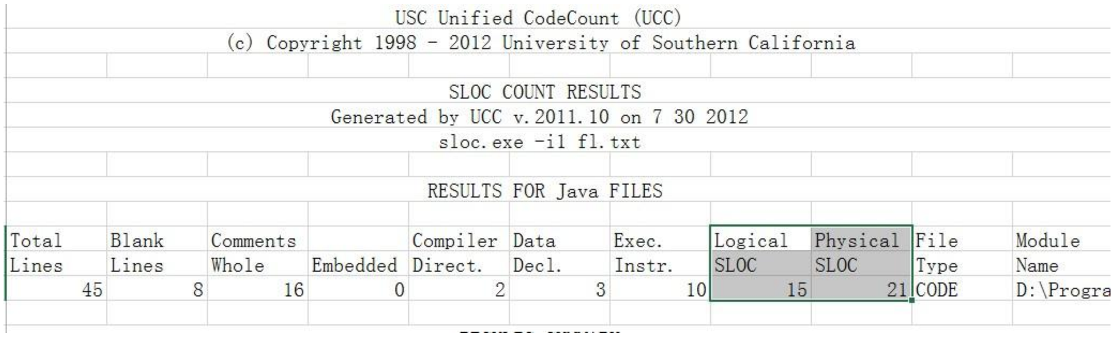
- MySqrt
- Quadratic
- Prime

As required, we have used USC CodeCount [2] to calculate logical SLOC of each class. Besides, we also use LocMetrics[3] as another tool to testify the results of the UCC.

2.1. USC CodeCount:

For calculating logical SLOC for each class and the results are the following ones:

2.1.1. MySqrt



USC Unified CodeCount (UCC)
(c) Copyright 1998 - 2012 University of Southern California

SLOC COUNT RESULTS
Generated by UCC v.2011.10 on 7 30 2012
sloc.exe -i1 fl.txt

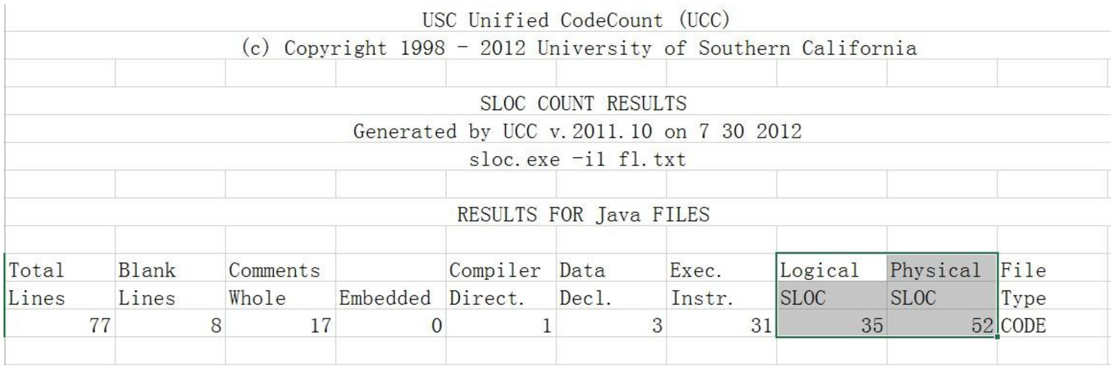
RESULTS FOR Java FILES

Total	Blank	Comments		Compiler	Data	Exec.	Logical	Physical	File	Module
Lines	Lines	Whole	Embedded	Direct.	Decl.	Instr.	SLOC	SLOC	Type	Name
45	8	16	0	2	3	10	15	21	CODE	D:\Progra

[2] http://sunset.usc.edu/research/CODECOUNT/

[3] http://www.locmetrics.com/

2.1.2. Quardatic



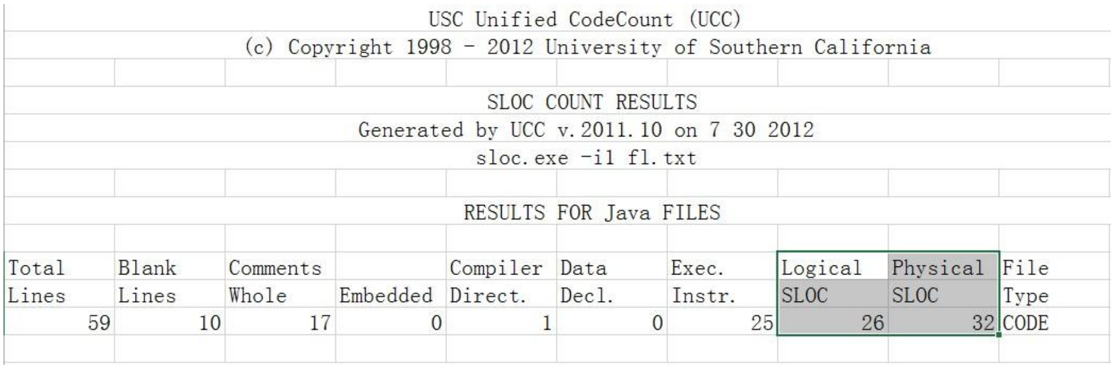
USC Unified CodeCount (UCC)
(c) Copyright 1998 - 2012 University of Southern California

SLOC COUNT RESULTS
Generated by UCC v.2011.10 on 7 30 2012
sloc.exe -i1 fl.txt

RESULTS FOR Java FILES

Total	Blank	Comments		Compiler	Data	Exec.	Logical	Physical	File
Lines	Lines	Whole	Embedded	Direct.	Decl.	Instr.	SLOC	SLOC	Type
77	8	17	0	1	3	31	35	52	CODE

2.1.3. Prime



USC Unified CodeCount (UCC)
(c) Copyright 1998 - 2012 University of Southern California

SLOC COUNT RESULTS
Generated by UCC v.2011.10 on 7 30 2012
sloc.exe -i1 fl.txt

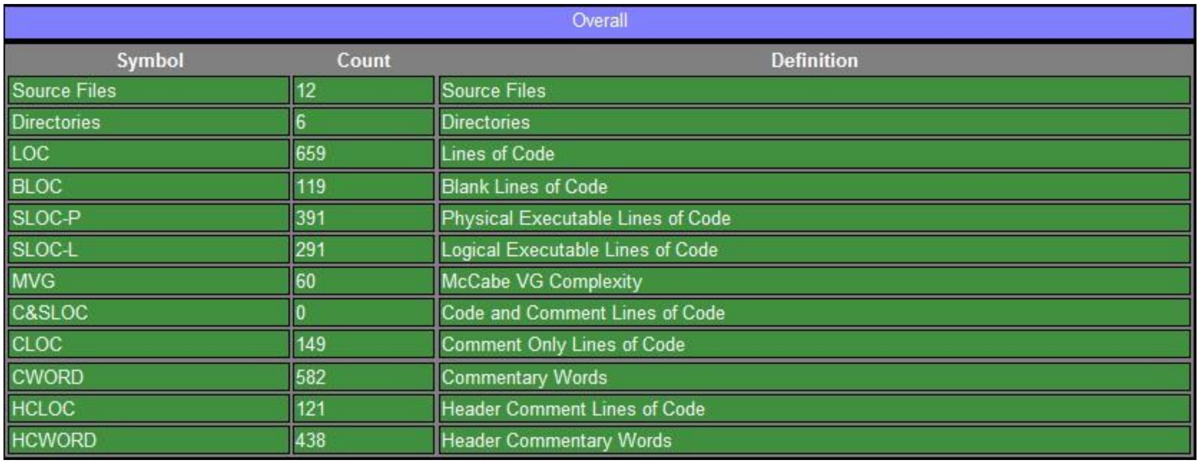
RESULTS FOR Java FILES

Total	Blank	Comments		Compiler	Data	Exec.	Logical	Physical	File
Lines	Lines	Whole	Embedded	Direct.	Decl.	Instr.	SLOC	SLOC	Type
59	10	17	0	1	0	25	26	32	CODE

2.2. LocMetrics

LocMetrics counts total lines of code (LOC), blank lines of code (BLOC), comment lines of code (CLOC), lines with both code and comments (C&SLOC), logical source lines of code (SLOC-L), McCabe VG complexity (MVG), and number of comment words (CWORDS). Physical executable source lines of code (SLOC-P) is calculated as the total lines of source code minus blank lines and comment lines. Counts are calculated on a per file basis and accumulated for the entire project. LocMetrics also generates a comment word histogram.

In LocMetrics, it generated all metrics information of the whole project. The Table below is the overall metrics information of the MINSK.

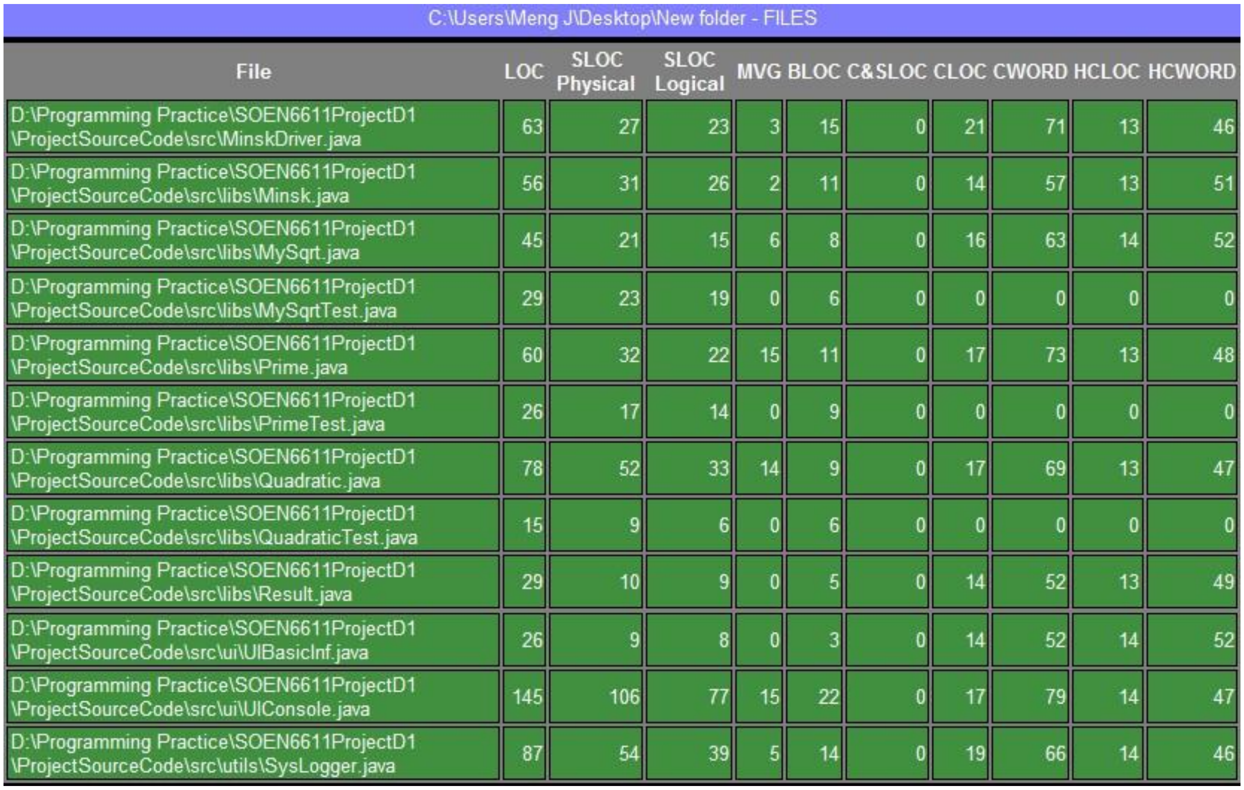


Overall

Symbol	Count	Definition
Source Files	12	Source Files
Directories	6	Directories
LOC	659	Lines of Code
BLOC	119	Blank Lines of Code
SLOC-P	391	Physical Executable Lines of Code
SLOC-L	291	Logical Executable Lines of Code
MVG	60	McCabe VG Complexity
C&SLOC	0	Code and Comment Lines of Code
CLOC	149	Comment Only Lines of Code
CWORD	582	Commentary Words
HCLOC	121	Header Comment Lines of Code
HCWORD	438	Header Commentary Words

Table 2-1 Overall metrics of MINSK

The Table below is the detailed LOC information of each class. Take a look at the three classes we calculated with UCC, MySqrt, Quadratic and Prime. In LocMetrics, the logical SLOC results of the classes match the ones in UCC.



C:\Users\Meng J\Desktop\New folder - FILES

File	LOC	SLOC Physical	SLOC Logical	MVG	BLOC	C&SLOC	CLOC	CWORD	HCLOC	HCWORD
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\MinskDriver.java	63	27	23	3	15	0	21	71	13	46
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\Minsk.java	56	31	26	2	11	0	14	57	13	51
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\MySqrt.java	45	21	15	6	8	0	16	63	14	52
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\MySqrtTest.java	29	23	19	0	6	0	0	0	0	0
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\Prime.java	60	32	22	15	11	0	17	73	13	48
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\PrimeTest.java	26	17	14	0	9	0	0	0	0	0
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\Quadratic.java	78	52	33	14	9	0	17	69	13	47
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\QuadraticTest.java	15	9	6	0	6	0	0	0	0	0
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\libs\Result.java	29	10	9	0	5	0	14	52	13	49
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\ui\UIBasicInf.java	26	9	8	0	3	0	14	52	14	52
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\ui\UIConsole.java	145	106	77	15	22	0	17	79	14	47
D:\Programming Practice\SOEN6611ProjectD1 \ProjectSourceCode\src\utils\SysLogger.java	87	54	39	5	14	0	19	66	14	46

Table 2-2 LOC of classes

3. Part3 (Spearman's Rank Correlation Coefficient)

3.1. Correlation between Quadratic and Prime classes

Let Xi be the rank of Logic SLOC for each class.
Let Yi be the rank of WMC for each class.
Let Di be the result of Xi – Yi.

According to the data we obtained from previous chapter, we can get the statistics:

	Logic SLOC(x_i)	Rank(x_i)	WMC(y_i)	Rank(y_i)	$d_i = x_i - y_i$	$(d_i)^2$
Quadratic	35	2	11	1	1	1
Prime	26	1	13	2	-1	1

Table 3-1 Statistic between Quadratic and Prime

Now $\sum_{i=1,2}(d_i)^2 = 1 + 1 = 2$ and n = 2. Thus $r_s = 1 – (6*2 / (2^3 – 2)) = -1$.

"The value r_s varies from -1 to 1. The values near 1 indicate a strong positive correlation, the values near -1 indicate a **strong negative correlation**, and the values near 0 indicate very weak or no correlation."[4]

As a result, the correlation between Logic SLOC and WMC for classes Quadratic and Prime has a **strong negative correlation**.

3.2. Correlation between three classes

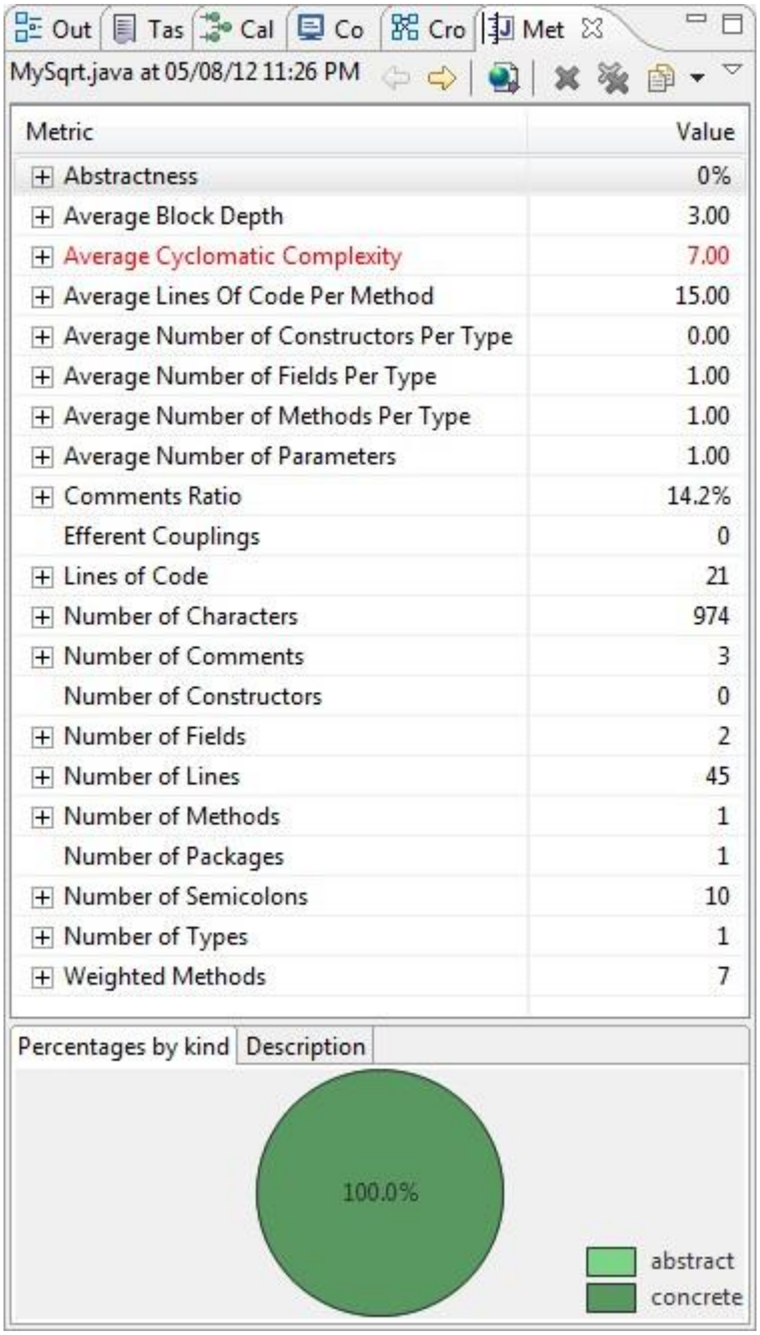
Since there are only two classes, the correlation is either strong negative correlation or strong positive correlation. We decide to add MySqrt class for analysis.

[4] software_measurement_data_analysis.pdf, Dr. Pankaj Kamthan, 2012

3.2.1. WMC of MySqrt

From CodePro, we get the following data of MySqrt Class:

- Average Cyclomatic Complexity: 7.00
- Weighted Methods: 7



MySqrt.java at 05/08/12 11:26 PM

Metric	Value
Abstractness	0%
Average Block Depth	3.00
Average Cyclomatic Complexity	7.00
Average Lines Of Code Per Method	15.00
Average Number of Constructors Per Type	0.00
Average Number of Fields Per Type	1.00
Average Number of Methods Per Type	1.00
Average Number of Parameters	1.00
Comments Ratio	14.2%
Efferent Couplings	0
Lines of Code	21
Number of Characters	974
Number of Comments	3
Number of Constructors	0
Number of Fields	2
Number of Lines	45
Number of Methods	1
Number of Packages	1
Number of Semicolons	10
Number of Types	1
Weighted Methods	7

Figure 3-1 WMC of MySqrt

Here is the statistic of three classes:

	Logic SLOC(x_i)	Rank(x_i)	WMC(y_i)	Rank(y_i)	$d_i = x_i - y_i$	$(d_i)^2$
Quadratic	35	3	11	2	1	1
Prime	26	2	13	3	-1	1
MySqrt	15	1	7	1	0	0

Table 3-2 Statistic between three classes

Now $\sum_{i=1,2}(d_i)^2 = 1 + 1 + 0 = 2$ and $n = 3$. Thus $r_s = 1 – (6*2 / (3^3 – 3)) = 0.5$.

The result shows that between Quadratic, Prime and MySqrt classes, there is a **positive correlation,** which is opposite to the previous result.

3.3. Conclusion

The correlation of Logic SLOC and WMC is quite different between two classes and three classes. In order to get more meaningful results, we need to do the analysis with more number of classes.

Appendix A: Reference

1. https://developers.google.com/java-dev-tools/download-codepro
2. http://sunset.usc.edu/research/CODECOUNT/
3. http://www.locmetrics.com/
4. software_measurement_data_analysis.pdf, Dr. Pankaj Kamthan, 2012

Appendix B: Revision History

Version	Date	Author	Remark
V0.1	Aug. 05, 2012	Team F	