Study on co-occurring critics

Ricardo Martins-Julien Le Van Suu

December 14, 2014

1 Introduction

A program which produces intelligent and non redundant critics should always be preferred. Having that kind of program can make a lot of difference in terms of productivity, cost efficiency,... In a first part we will study the issues in a simple way : that's say counting the critics and calculate the distance. For that, we will use two representatives packages. We will not put irrelevant issues, as theses should be always removed. The study will be done by method as this is more fitting with Checkstyle searching patterns. In every package, we pick only two classes as there are big classes which counts more than 500 errors.
For finishing, we will provide some high levels critics (For example sometimes some co occurring patterns may be great) and suggest a way to improve the CkeckStyle program. The automatic excel file is available here : `https://github.com/bigjulien/mmreport`

2 Study on the co occurring critics

2.1 Sorting the results : Statistics method

For finding the anomalies, we will use the following rules :

- The results in the range $]0; m - \sigma]$ (σ the standard deviation and m the mean of the distances between one rules and all others, except itself) will be highlighted in green with red font(because if there is a correlation, it's a strong one).
  The reason is a value outside $[m - \sigma; m + \sigma]$ is , in stats, a "abnormal" value.

- The values between $]m - \sigma; 0.9]$ will be also highlighted in green. The reason behind the value 0.9? It is recommended in the paper, and furthermore, it seems to don't miss something (a lot of values are = 0.9). Actually, if we would want to be sure to have max precision, we would just do all the pairs of issues(which correspond to a interval of [0 - 1].
  However, the following results gives us also great information about the strength of the correlations.

It's always a trade off between precision and fastening of the process.

- All the results in green (between 0 and 0.9) will be studied
- The values which are considered to be interesting patterns (everything but accidental correlations) will be in bold.

2.2 Results

2.2.1 First package: Root package

CC01 : Missing javadoc comment
CC02 : Reference to instance variable needs "this.".
CC03 : Variable should be declared final
CC04 : X is a magic number§
CC05 : Line longer than 80 characters
CC06 : Parameter should be final
CC07 : Method not designed for extension
CC08 : Executable count is X (max 30)
CC09 : Each variable declaration in its own statement
CC10 : Boolean expression complexity (max 3)
CC11 : variable hides a field
CC12 : Nested if-else depth (max 1)
CC13 : switch without default clause
CC14 : '++' not allowed
CC15 : Expected @param tag
CC16 : cyclomatic complexity (max 10)
CC17 : NCSS (max 50)
CC18 : Npath complexity (max 200)
CC19 : Uncommented main method
CC20 : Expression can be simplified
CC21 : Expected @param tag

	CC01	CC02	CC03	CC04	CC05	CC06	CC07	CC08	CC09	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20	CC21	M-SD
CC01		0,765	0,615	0,636	0,625	0,833	0,923	0,818	0,889	1	0,9	0,909	1	1	1	0,889	0,889	1	**0,875**	0,875	1	0,745966896
CC02	0,765		0,85	0,75	**0,714**	0,7	0,643	0,875	1	1	0,933	0,938	1	1	0,933	0,929	0,929	0,923	1	1	0,846	0,77177199
CC03	0,615	0,85		0,786	0,588	**0,722**	0,857	0,846	0,8	0,9	**0,818**	1	1	1	0,917	0,909	0,909	1	0,9	0,9	1	0,745811177
CC04	0,636	0,75	0,786		0,895	0,75	1	0,909	1	1	0,889	1	1	0,857	0,75	1	1	1	1	1	1	0,794190907
CC05	0,625	0,714	0,588	0,895		0,65	0,889	**0,733**	**0,857**	1	0,867	**0,714**	0,929	1	0,938	**0,857**	**0,857**	0,929	0,929	0,929	1	0,716608183
CC06	0,833	0,7	0,722	0,75	0,65		0,813	0,875	0,846	1	0,857	0,938	0,923	0,923	**0,857**	0,929	0,929	1	0,923	0,923	0,846	0,765904296
CC07	0,923	0,643	0,857	1	0,889	0,813		1	1	1	1	1	1	1	1	1	1	1	1	1	0,667	0,827361196
CC08	0,818	0,875	0,846	0,909	0,733	0,875	1		0,6	1	1	**0,714**	1	1	0,857	0,6	0,6	0,8	0,8	0,8	1	0,702512492
CC09	0,889	1	0,8	1	0,857	0,846	1	0,6		1	1	1	1	1	1	0,66	0,66	1	0,5	**0,5**	1	0,684036449
CC10	1	1	0,9	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	0,97263932
CC11	0,9	0,933	0,818	0,889	0,867	0,857	1	1	1	1		1	1	0,667	0,8	1	1	1	1	1	1	0,842407398
CC12	0,909	0,938	1	1	0,714	0,938	1	0,714	1	1	1		1	0,75	1	**0,833**	**0,8**	**0,8**	0,75	1	1	0,794781821
CC13	1	1	1	1	0,929	0,923	1	1	1	1	1	1		1	1	1	1	1	1	1	1	0,969734188
CC14	1	1	1	0,857	1	0,923	1	1	1	1	0,667	0,75	1		0,666	1	1	1	1	1	1	0,829222672
CC15	1	0,933	0,917	0,75	0,938	0,857	1	0,857	1	1	0,8	1	1	0,666		0,75	0,75	0,667	1	1	1	0,772326116
CC16	0,889	0,929	0,909	1	0,857	0,929	1	0,6	0,66	1	1	0,833	1	1	0,75		0	**0,5**	0,5	**0,5**	1	0,530358387
CC17	0,889	0,929	0,909	1	0,857	0,929	1	0,6	0,66	1	1	0,8	1	1	0,75	0		**0,5**	0,5	**0,5**	1	0,52887101
CC18	1	0,923	1	1	0,929	1	1	0,8	1	1	1	0,8	1	1	0,667	0,5	0,5		1	1	1	0,739715375
CC19	0,875	1	0,9	1	0,929	0,923	1	0,8	0,5	1	1	0,75	1	1	1	0,5	0,5	1		0	1	0,56857079
CC20	0,875	1	0,9	1	0,929	0,923	1	0,8	0,5	1	1	1	1	1	1	0,5	0,5	1	0		1	0,579344375
CC21	1	0,846	1	1	1	0,846	0,667	1	1	1	1	1	1	1	1	1	1	1	1	1		0,882833306

There is a slight error because "Excepted @Param tag" appears twice, but it doesn't influence our final results.

2.2.2 Second package : Graphic2D

CC01 : Missing a Javadoc Comment
CC02 : Type Javadoc comment is missing null ta
CC03 : { should be on previous line
CC04 : Param should be final
CC05 : Method not designed for extension
CC06 : X is not followed by whitespace
CC07 : VAR should be final
CC08 : Return count is 3
CC09 : Don't Use trailing comments
CC10 : Line is longer than X
CC11 : Assignement of param is not allowed
CC12 : X is magic number
CC13 : reference needs this
CC14 : Unnecessary parenthesis
CC15 : Array should contain trailling comma
CC16 : Should be on same line
CC17 : Nested if else
CC18 : Must be private and have access method
CC19 : Static var def in wrong order
CC20 : Var explicilty instanced to null

	CC01	CC02	CC03	CC04	CC05	CC06	CC07	CC08	CC09	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20	M-SD
CC01		0,778	0,55	0,588	0,941	0,941	0,786	1	1	0,8	1	0,8	0,917	0,9	0,889	0,889	0,778	0,889	0,889	0,889	0,7329654464
CC02	0,778		0,9	0,938	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0,9255159101
CC03	0,55	0,9		0,478	0,55	0,739	0,667	0,95	0,9	0,739	0,955	0,905	0,857	0,952	0,95	0,95	**0,9**	0,95	0,95	0,95	0,673041877
CC04	0,588	0,938	0,478		0,8	0,737	**0,647**	1	0,867	0,5	**0,8**	0,875	0,733	0,867	0,933	0,933	0,938	1	1	1	0,6545571519
CC05	0,941	1	0,55	0,8		0,714	0,938	**0,889**	1	1	1	1	0,818	1	1	1	1	1	1	1	0,8063066837
CC06	0,941	1	0,739	0,737	0,714		0,786	0,889	1	0,714	0,8	0,909	0,818	0,9	1	1	0,9	1	1	1	0,7762203467
CC07	0,786	1	0,667	0,647	0,938	0,786		0,875	0,75	0,3	0,9	0,9	1	0,75	0,875	0,875	1	1	1	1	0,6702756409
CC08	1	1	0,95	1	0,889	0,889	0,875		1	1	1	1	1	1	1	1	1	1	1	1	0,9362185449
CC09	1	1	0,9	0,867	1	1	0,75	1		**0,778**	0,75	1	1	1	1	1	1	1	1	1	0,8589487626
CC10	0,8	1	0,739	0,5	1	0,714	0,3	1	0,778		0,667	0,8	0,917	0,778	0,889	0,889	1	1	1	1	0,6394953897
CC11	1	1	0,955	0,8	1	0,8	0,9	1	0,75	0,667		0,8	0,833	0,8	0,75	0,75	0,8	0,75	0,75	0,75	0,7288189673
CC12	0,8	1	0,905	0,875	1	0,909	0,9	1	1	0,8	0,8		0,833	0,75	0,667	0,667	0,75	0,667	0,667	0,667	0,6970498529
CC13	0,917	1	0,857	0,733	0,818	0,818	1	1	1	0,917	0,833	0,833		1	1	1	0,8	1	1	1	0,8311101411
CC14	0,9	1	0,952	0,867	1	0,9	0,75	1	1	0,778	1	0,75	1		0,5	0,5	1	1	1	1	0,7281821143
CC15	0,889	1	0,95	0,933	1	1	0,875	1	1	0,889	1	0,667	1	0,5		0	1	1	1	1	0,6336605003
CC16	0,889	1	0,95	0,933	1	1	0,875	1	1	0,889	1	0,667	1	0,5	0		1	1	1	1	0,6336605003
CC17	0,778	1	0,9	0,938	1	0,9	1	1	1	1	1	0,75	0,8	1	1	1		0,5	0,5	0,5	0,6738190839
CC18	0,889	1	0,95	1	1	1	1	1	1	1	1	0,667	1	1	1	1	0,5		0	0	0,4740316051
CC19	0,889	1	0,95	1	1	1	1	1	1	1	1	0,667	1	1	1	1	0,5	0		0	0,4740316051
CC20	0,889	1	0,95	1	1	1	1	1	1	1	1	0,667	1	1	1	1	0,5	0	0		0,4740316051

The results are perhaps a bit surprising because of the existance of some 0's outside the diagonal .

The explanation is that the rarer errors, in terms of occurrences, are in the bottom of the table (because we found them after the others, this is logical in a probabilistic point of view). Theses errors are likely to have strange values, because of the few cases they are implied in.
However, we won't discuss them (as advised in the code critics documents) because there is almost no chance to discover an interesting pattern.

3 Classification on those critics : Package 1

3.1 Redundant Critics

CC16,CC17 - CC18 "Cyclomatic" - "NCSS" & "NPath"
The two complexity issues seem to be correlated to the NPath but not correlated together. Why not just delete the NPath, which overlaps those two ?

3.2 Overlap Requires Splitting

CC09 - CC20 "Each var declaration in own statement" - " Expression can be simplified"
We can split the" expression can be simplified" in : cc09 :"each var declaration in own statement" and "too much or/and in the expression"

3.3 Overlap Requires Merging

CC07 - CC16,17,18 "Executable count is X" - "Complexities"
Of course if there is a lot of executable statement the complexity is bad. As this is calculated in the complexity, we should fusion in a rule "X complexity is too high because of X executable statements)

3.4 Same niche

CC01 - CC19 "Missing Javadoc Comment" - "Uncommented Main method"
Theses two are speaking of the same entity : comments so it's likely that the developer who didn't think it was useful to comment his main, had the same line of tought when it comes to commenting the overall class.

CC03 - CC11 "Variable should be declared Final" - "Variable hides a field"
These issues are about variables.

CC06 - CC15 "Parameter should be final" - "Missing @parameter"
Trivial

3.5 Almost subset

CC02 - CC05 "Needs this" - "Line longer than 80 characters" As there is a lot of very long class names in this project, using the name class instead of this will case an increase of the number of characters

CC05 - CC09 "Line is longer than" - "Each variable declaration in its own statement"
If a variable is declared like that it will likely increase the number of characters.

CC05 - CC12 "Line is longer than" - "nested if depth"
The algorithm is counting tabulations and spaces, we can see when there is a lot of indentation due to ifs, it will also produce a too long line issue.

3.6 High level critics

CC03 - CC06 "Var should be declared final" - "Parameter should be declared final"
This is almost trivial : When the developer wants to make calculations, rename variable in the body of the method, we will not "cast" the parameter into a final one.

CC05 - CC08 "Line longer than" - "Exec count is X"
This means than the code is probably too complex. Theses two alerts together are giving us a hint to the code complexity, which is great.

CC05 - CC16,17: "Line longer than" - "Complexity matters"
Same like above.

CC12 - CC16,17,18 "Nested if depth" - "Complexities"
Excepted because if there is a lot of nested ifs this increase the complexity. We should not recommend to merge them because it gives us additional information about the source of the complexity.

3.7 Accidental Correlation

Just one example will be putted there, as this is all the relations CC(i)-CC(j) we haven't mentioned before.
We won't highlight this kind of issues in excel as they are not interesting.

CC01 - CC03 "Missing Javadoc Comment" - " } Should be on previous line"
Obviously, these two have nothing related. This is not even speaking of same entities.

3.8 Noisy correlation

We won't highlight in the excel file this kind of issues. Most of the noisy correlations will be also accidental correlation.

CC04 - CC05,06 "Param should be final" - "Method not designed for extension"&"Line is longer than"

4 Classification on those critics : Package 2

We will only put there new correlations founded.

4.1 Almost subset

CC04 - CC11 "Param should be final" - "Assignement of param is not allowed"
If we resolve the first we resolve also the second. Actually, we don't recommend to delete one of those two.

CC09 - CC10 "Trailing comments" - "Line is longer than"
If there is comments the line will be likely longer ...

4.2 High level critics

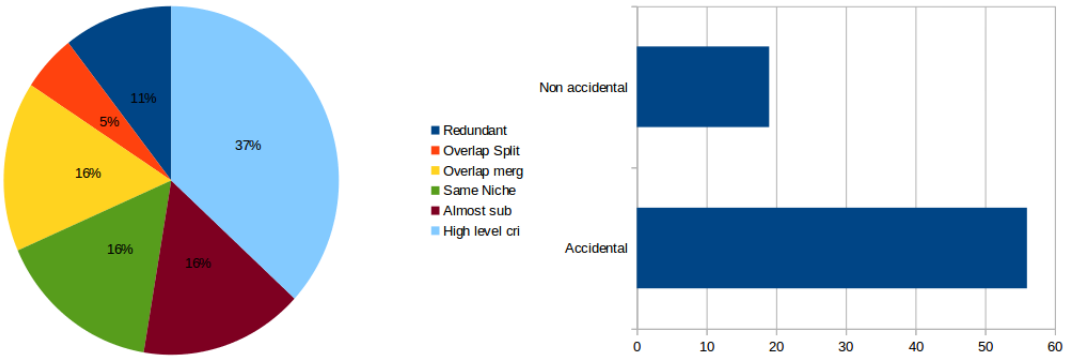
CC03 - CC17 "{ Should be on previous line" - "Nested if else"
In if there is likely a { so theses two are correlated, but it's actually good.

CC05 - CC08 "Method not designed for extension" - "Return count is 3"
This is actually giving us a hint of why this return count is hight : The method is not designed to be extended. Perhaps we should re factor and do inheritance. This is two good critics.

5 Global improvements and comments on co occurring critics

5.1 Statistics

We will present statistics about the errors founded in the first package.



Number of critics in the interval $]0; m - \sigma] = 9$
Number of critics in the interval $]m - \sigma; 0, 9] = 19$

Theses are revealing some hints :

- Half of the correlation are "strong" ones, in the interval $]0; m - \sigma]$

- There is no ill defined critics : This is normal, because ill defined critics always produce false positives in our case (ex: "switch not allowed"), so we don't put those critics in the excel sheet.
- There is a lot of high level critics which are for the most expected/good critics.
- Most of the correlations are accidental, which reveals a great quality of the program.
- The "Same niche" correlations should not be considered as a problem.

5.2 Global recommendations

We can see that most of the problematic correlations can be resolved by merging.
We would recommend to merge into more consequent part the problems and so, the program should gain in readability, although losing a little precision .
Many of the correlations are due to the different complexities measures, so this should be the first thing to look at.

On the overall, the program is doing well and the correlations should not be a problem. Theses great results are probably due to a big community which is testing and enhancing this tool.

6 Conclusion

For a measure program developer, the just middle between be too fined or too coarse is very hard to find. This is also normal to find correlations, a serious measure program without is impossible.
Some are even great, because it gives us more informations and permits to identify more quickly the root of the problem.