Software Quality Measurement

Gruppe H

5. Januar 2018

Inhaltsverzeichnis

1 Software Quality Measurement

Für `mallet`[1], eine Verarbeitungssoftware für natürliche Sprache, sollen beispielhaft Kenngrößen für Softwarequalität ermittelt und ausgewertet werden. Zur Berechnung der Kenngrößen wird `ckjm`[2] verwendet, das alle sechs Kenngrößen (Metriken zum Messen von Softwarequalität) nach Chidamber and Kemerer sowie zwei zusätzliche ermittelt. Die Ausgaben von `ckjm` werden mit Hilfe eines zur Verfügung gestellten, auf `matplotlib` basierenden Pythonskripts aufbereitet und in Form von Histogrammen dargestellt.

Listing 1.1: Beispiele für den Output von mallet

```
1 cc.mallet.extract.test.TestDocumentExtraction 7 0 0 20 31 21
      0 7
2 cc.mallet.pipe.SerialPipes$Predicate 2 1 0 2 3 1 1 2
3 cc.mallet.classify.BalancedWinnowTrainer 6 0 0 11 21 11 1 6
4 cc.mallet.util.CharSequenceLexer 20 1 0 1 45 94 14 16
5 ...
```

1.1 CBO - Kopplung zwischen Objektklassen

Ein hoher Wert des CBO (Coupling between object classes) ist erwünscht. Das bedeutet, dass Methoden mit vielen Instanzvariablen eng gekoppelt sind, was sich wiederum positiv auf die Softwarequalität auswirkt.

Ein kleiner Wert ist unerwünscht, da dies bedeutet, dass alle Methoden der Klasse eigene unabhängige Instanzvariablen benutzen. Dies wirkt sich negativ auf die Softwarequalität aus. Über 10% der Klassen haben einen CBO von 0, die Tabelle der schlechtesten Klassen zeigt also nur eine Teilmenge der Klassen mit diesem Wert.

CBO: schlechteste Klassen	
`cc.mallet.util.Lexer`	0
`cc.mallet.topics.TopicReports`	0
`cc.mallet.cluster.util.PairwiseMatrix`	0

CBO: beste Klassen	
`cc.mallet.classify.tests.TestNaiveBayes`	33
`cc.mallet.fst.semi_supervised.tui.SimpleTaggerWithConstraints`	34
`cc.mallet.fst.tests.TestCRF`	51

[1] Mallet auf github
[2] ckjm auf github, Manual

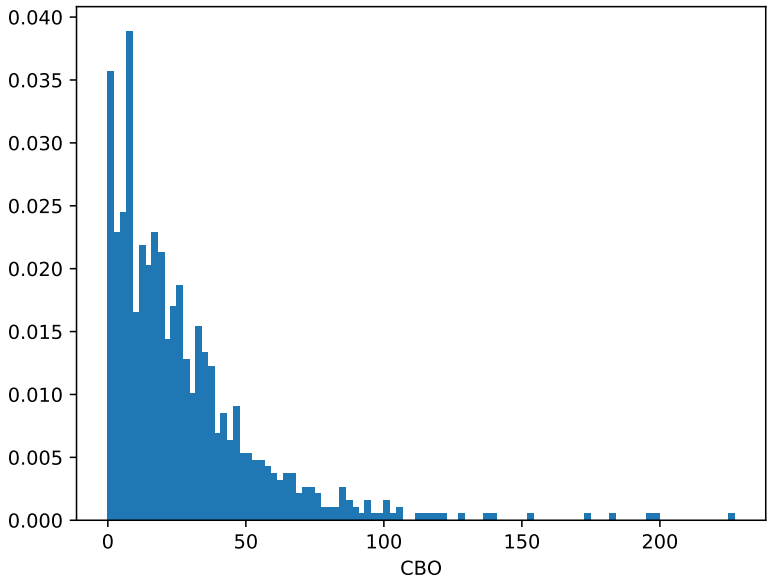

Abbildung 1.1: CBO: Coupling between Object Classes

Das Histogramm (Abb. 1.1) zeigt die CBO-Werte aller getesteten Klassen. Die CBO-Werte liegen im Bereich von 0 bis 51.

1.2 LCOM - Mangel an Abgeschlossenheit

Im Vergleich zum CBO-Wert verhält sich der LCOM-Wert komplementär. Ein hoher Wert steht für den Mangel an Kohäsion (Zusammenhang zwischen den Methoden) und wirkt sich somit negativ auf die Softwarequalität aus. Ein niedriger Wert bedeutet keinen Mangel und ist somit positiv für die Softwarequalität.

LCOM: beste Klassen	
`cc.mallet.fst.semi_supervised.GELatticeTask`	0
`cc.mallet.optimize.OptimizerEvaluator$ByBatchGradient`	0
`cc.mallet.util.ProgressMessageLogFormatter`	0

LCOM: schlechteste Klassen	
`cc.mallet.types.InstanceList`	2365
`cc.mallet.types.MatrixOps`	1911
`cc.mallet.fst.CRF`	1578

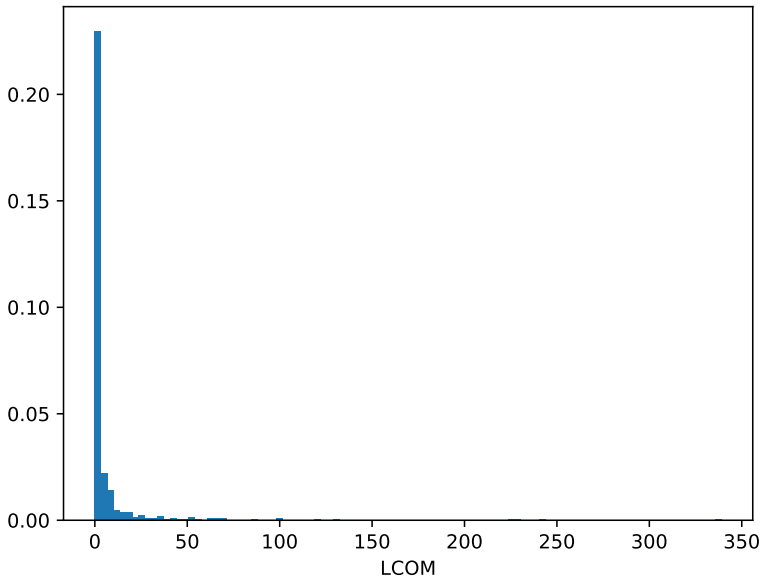

Abbildung 1.2: LCOM: Lack of cohesion in methods

1.3 Weitere Kenngrößen und graphische Darstellung

Die Kenngrößen werden als normierte Histogramme dargestellt, auf der x-Achse sind Bereiche möglicher Werte der Kenngröße aufgetragen. Auf der y-Achse der Anteil der Klassen, die diesem Wertebereich zugeordnet sind.

Neben den in den vorherigen Kaptieln bereits vorgestellten Kenngrößen wurden folgende weitere von `ckjm` ermittelt:

WMC: Weighted methods per class (Abb. 1.3 li.) stellt der Wert einfach nur die Anzahl der Methoden einer Klasse dar, weil die Gewichtung (entsprechend der Komplexität) aller Methoden auf 1 gesetzt wurde. Als Grund für diesen festen Wert werden die einfache Implementierung und die Kompatibilität zu Chidamber und Kemerer genannt. Für die Verständlichkeit und Erweiterbarkeit ist es von Vorteil, wenn die Anzahl der Methoden einer Klasse in einem überschaubaren Rahmen bleiben. Bei der betrachteten Software haben alle Klassen 0–5 Methoden, was sehr überschaubar ist.

Größere Werte sind negativ. Kleinere Werte sind positiv.

DIT: Depth of Inheritance Tree (Abb. 1.3 re.) DIT misst die Tiefe des Vererbungsbaums. Je tiefer dieser Baum ist, desto negativer wirkt sich das auf die Softwarequalität aus, weil es dann schwieriger ist, die Vererbungsstruktur bis in die Blätter nachzuvollziehen. D.h. in unserem Histogramm sind solche Klassen gut, die den Wert 0 haben, weil die Tiefe des Baums

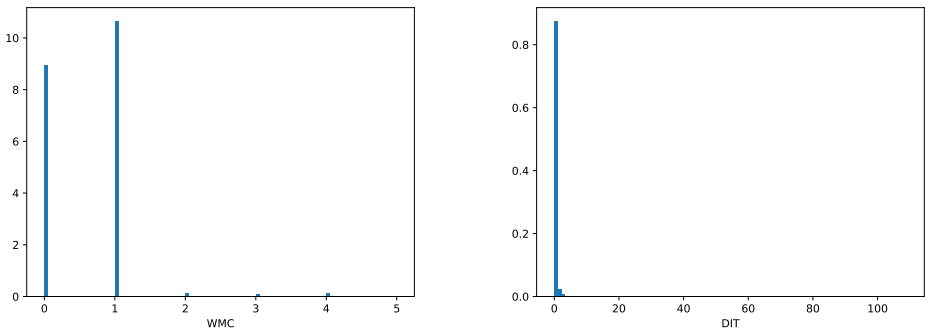

Abbildung 1.3: WMC (WB: 0-5) und DIT (WB: 0-109)

NOC: Number of Children (Abb. 1.4 li.) NOC misst die Anzahl der Subklassen, die direkt unter der gemessenen Klasse existieren. Ein hoher Wert bedeutet, dass diese Klasse oft wiederverwendet wird und somit Code gespart wird. Das bedeutet aber auch, dass eine Klasse mit hohem Wert gut getestet muss, um die Softwarequalität zu steigern. Im Vergleich zu DIT misst NOC die Breite des Baums, statt der Tiefe.

RFC: Response for a Class (Abb. 1.4 re.) steht für die Anzahl aller möglichen auszuführenden Methoden innerhalb der Klasse. Je höher der Wert, desto höher die Komplexität der Klasse. Also ist ein hoher Wert nicht erwünscht.

Ca: Afferent coupling (nicht C&K-Metrik, Abb. 1.5 li.) gibt an, wie viele Klassen von dieser Klasse abhängen. Eine Klasse mit einem hohen Ca-Wert hat eine hohe Verantwortung innerhalb der Software, weil sie Funktionalitäten für viele andere Klassen zur Verfügung stellt. Weit über die Hälfte der Klassen in `mallet` weist hier eher geringe Werte unter 10 auf. Es gibt aber auch mehrere Klassen, die von über 20 und bis zu 68 anderen Klassen genutzt werden, das heißt dass eine Änderung einer dieser Klassen in sehr vielen anderen Klassen Auswirkungen haben kann. Diese Auswirkungen sind extrem schlecht zu überschauen, die Bearbeitung einer solchen Klasse führt zu nicht abschätzbaren Folgen und jeden abhängige Klasse bringt ein weiteres Risiko, dass die Software nicht mehr funktioniert, mit sich.

NPM: Number of Public Methods for a class (nicht C&K-Metrik, Abb. 1.5 re.) bestimmt die Anzahl der public methods innerhalb der Klasse. Sie ist ein Maß dafür, wie umfangreich die API ist, die eine Klasse, bzw. in Summe ihr Package zur Verfügung stellt. Öffentliche Methoden sollten bewusst und sparsam eingesetzt werden, sie brauchen eine gute Dokumentation und dürfen in ihre nach außen sichtbare Funktionalität darf nicht ohne Weiteres verändert werden. Sie stellen also ein Hindernis für Veränderungen in der Klasse dar und erschweren ihre Wartung. Ein hoher Wert für NPM kann ein Hinweis darauf sein, dass das Prinzip der Kapselung von Klassen nicht eingehalten wurde.

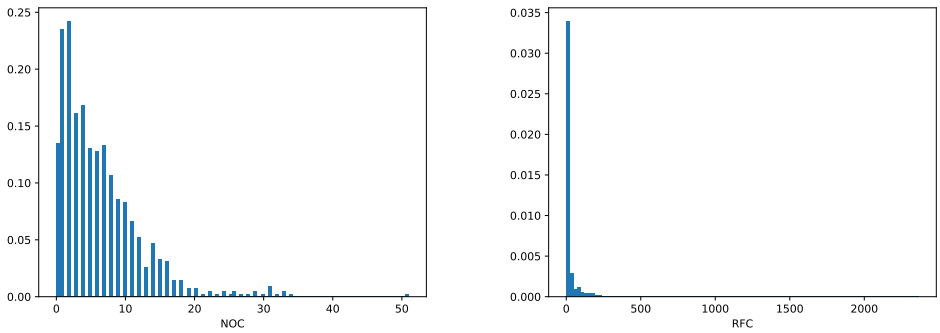

Abbildung 1.4: NOC (WB: 0–51) und RFC (WB: 0–2356)

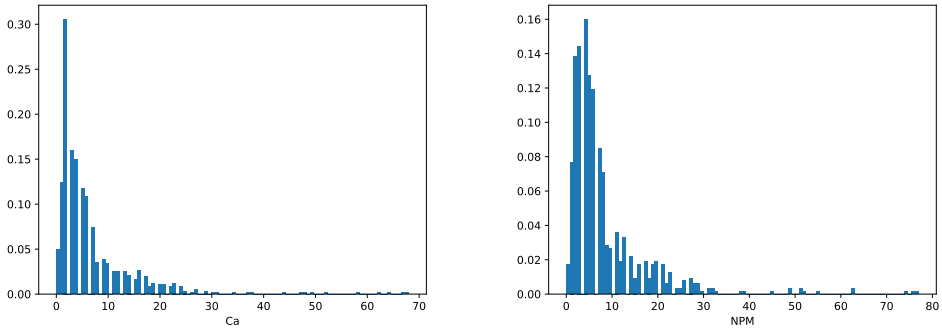

Abbildung 1.5: Ca (WB: 0-68) und NPM (0-77)

1.4 Vergleich mit einer abgeänderten Version von mallet

Dem Programm `mallet` werden zwei weitere Klassen `NewParallelTopicModel.java` und `TopicInferencerInterface.java` hinzugefügt. Nach dem Kompilieren wird die Analyse von `mallet` wiederholt und es werden die Kennzahlen der Klassen `NewParallelTopicModel` und `ParallelTopicModel` verglichen:

Klasse	WMC	DIT	NOC	CBO	RFC	LCOM	Ca	NPM
`NewParallelTopicModel`	68	1	0	22	233	800	0	61
`ParallelTopicModel`	63	1	2	21	227	627	11	58

Im Vergleich zur ursprünglichen Version der von `ParallelTopicModel` hat sich WMC um den Wert 5 erhöht, das heißt die Anzahl der Methoden der Klasse hat zugenommen, der um 3 erhöhte Wert für NPM zeigt, dass davon nur 3 Methoden public waren. Die Tiefe des Vererbungsbaums (DIT) ist unverändert, die Anzahl der Kinder (NOC) um 2 auf 0 verringert. Die Kopplung zwischen Objektklassen (CBO) hat sich kaum verändert, während der Mangel an Abgeschlossenheit (LCOM) um über 25% angewachsen ist. Der Ca-Wert, der die Anzahl der Klassen, die von dieser Klassen abhängen, angibt, hat sich laut Berechnung von 11 auf 0 reduziert.