

Wartbarkeit von Java Methoden: Ein quantitativer Ansatz

Seminar *Softwarequalität*

Andreas Heckl

20.07.2020

Inhalt

- 1) Motivation
- 2) Beispiele für Software Metriken
- 3) Existierende statische Analysetools
- 4) Eigener Prototyp

Motivation

- Viele Qualitätsstandards existieren, bieten allerdings nur abstrakte Beschreibungen
→ in Praxis nicht eindeutig umzusetzen oder messbar
- Ziel: mithilfe eines quantitativen Ansatzes Methoden nach „Inspektionswürdigkeit“ (im Sinne der Wartbarkeit) priorisierbar machen
- Warum Wartbarkeit ?
 - a) hoher Kostenfaktor
 - b) relevant für die meisten Systeme
- Warum Java Methoden ?
→ im Vergleich zu anderen „Ebenen“ existieren nur wenige Tools und Forschung für Methoden

Beispiele für Metriken

- Lines of Code
 - verschiedene Berechnungsmöglichkeiten
 - Beispiele im Kontext von Java:
 - inkl./exkl. Kommentarzeilen/Leerzeilen/Zeilen die nur Klammer enthalten
- Maintainability Index
 - zusammengesetzte, gewichtete Metrik
 - versucht, Wartbarkeit in einer einzigen Zahl auszudrücken
 - Kritik: Berechnung, Gewichtung und Interpretation nicht eindeutig

Existierende statische Analysetools (SATs)

Findings-orientiert



- Bieten verschiedene Sichten auf das System
- Modernes Design
- Bieten oft Sonderfunktionen wie Test Gap Analyse
- Darstellung von Metriken bis hinunter zur Klassenebene

Metriken-orientiert



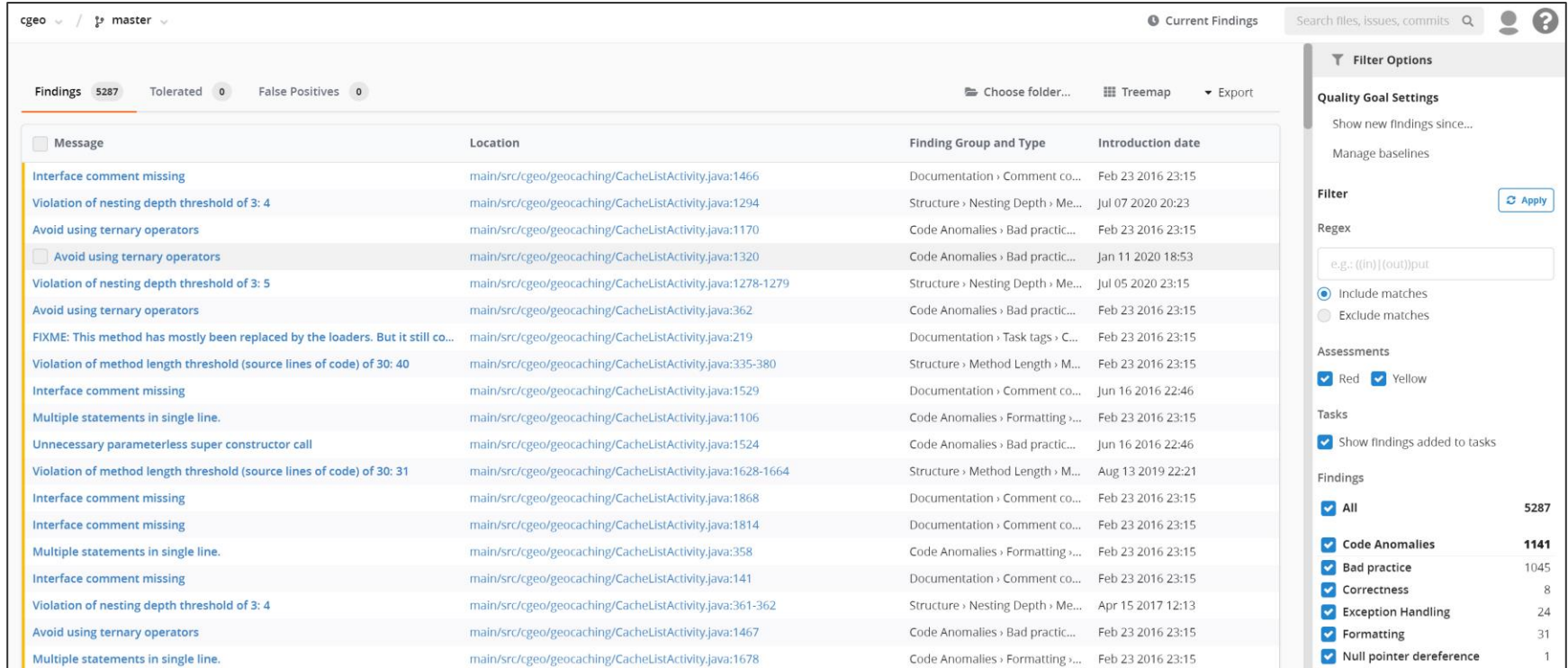
- nur wenige verfügbar
- vermutlich in Praxis nur selten eingesetzt
- Darstellung in Tabellenform
- Erlauben teilweise nur Export von „rohen“ Daten, z.B. als CSV oder XML → keine Analyse

Einschub: Findings

- Findings sind konkrete potenzielle **Qualitätsmängel**, die von einem Tool gefunden wurden
- Sie können verschiedene Eigenschaften haben, z.B. Beschreibung, Kategorie, Einschätzung nach Schwere des Mangels etc.
- Zwei Beispiele für Findings:

<input type="checkbox"/> Message	Location	Finding Group and Type	Introduction date
Violation of file size threshold (source lines of code) of 750: 1709	main/src/cgeo/geocaching/CacheListActivity.java	Structure › File Size › Metric Violatio...	Feb 23 2016 23:15
Name <code>singleton</code> violates naming convention. Should be one of <code>[A-Z][_A-Z0-9]*</code>	main/src/cgeo/geocaching/sorting/VisitComparator.java:14	Naming › Java naming conventions ...	Feb 23 2016 23:15

Findings-orientierte Tools: Findings-Perspektive



The screenshot shows a web-based interface for finding code quality issues. The main area displays a table of findings, and the right sidebar contains filter options and a summary of findings counts.

Message	Location	Finding Group and Type	Introduction date
Interface comment missing	main/src/cgeo/geocaching/CacheListActivity.java:1466	Documentation › Comment co...	Feb 23 2016 23:15
Violation of nesting depth threshold of 3: 4	main/src/cgeo/geocaching/CacheListActivity.java:1294	Structure › Nesting Depth › Me...	Jul 07 2020 20:23
Avoid using ternary operators	main/src/cgeo/geocaching/CacheListActivity.java:1170	Code Anomalies › Bad practic...	Feb 23 2016 23:15
Avoid using ternary operators	main/src/cgeo/geocaching/CacheListActivity.java:1320	Code Anomalies › Bad practic...	Jan 11 2020 18:53
Violation of nesting depth threshold of 3: 5	main/src/cgeo/geocaching/CacheListActivity.java:1278-1279	Structure › Nesting Depth › Me...	Jul 05 2020 23:15
Avoid using ternary operators	main/src/cgeo/geocaching/CacheListActivity.java:362	Code Anomalies › Bad practic...	Feb 23 2016 23:15
FIXME: This method has mostly been replaced by the loaders. But it still co...	main/src/cgeo/geocaching/CacheListActivity.java:219	Documentation › Task tags › C...	Feb 23 2016 23:15
Violation of method length threshold (source lines of code) of 30: 40	main/src/cgeo/geocaching/CacheListActivity.java:335-380	Structure › Method Length › M...	Feb 23 2016 23:15
Interface comment missing	main/src/cgeo/geocaching/CacheListActivity.java:1529	Documentation › Comment co...	Jun 16 2016 22:46
Multiple statements in single line.	main/src/cgeo/geocaching/CacheListActivity.java:1106	Code Anomalies › Formatting ›...	Feb 23 2016 23:15
Unnecessary parameterless super constructor call	main/src/cgeo/geocaching/CacheListActivity.java:1524	Code Anomalies › Bad practic...	Jun 16 2016 22:46
Violation of method length threshold (source lines of code) of 30: 31	main/src/cgeo/geocaching/CacheListActivity.java:1628-1664	Structure › Method Length › M...	Aug 13 2019 22:21
Interface comment missing	main/src/cgeo/geocaching/CacheListActivity.java:1868	Documentation › Comment co...	Feb 23 2016 23:15
Interface comment missing	main/src/cgeo/geocaching/CacheListActivity.java:1814	Documentation › Comment co...	Feb 23 2016 23:15
Multiple statements in single line.	main/src/cgeo/geocaching/CacheListActivity.java:358	Code Anomalies › Formatting ›...	Feb 23 2016 23:15
Interface comment missing	main/src/cgeo/geocaching/CacheListActivity.java:141	Documentation › Comment co...	Feb 23 2016 23:15
Violation of nesting depth threshold of 3: 4	main/src/cgeo/geocaching/CacheListActivity.java:361-362	Structure › Nesting Depth › Me...	Apr 15 2017 12:13
Avoid using ternary operators	main/src/cgeo/geocaching/CacheListActivity.java:1467	Code Anomalies › Bad practic...	Feb 23 2016 23:15
Multiple statements in single line.	main/src/cgeo/geocaching/CacheListActivity.java:1678	Code Anomalies › Formatting ›...	Feb 23 2016 23:15

Filter Options

Quality Goal Settings

Show new findings since...
Manage baselines

Filter Apply

Regex
e.g.: ((in)|(out))put

☒ Include matches
☐ Exclude matches

Assessments

☒ Red ☒ Yellow

Tasks

☒ Show findings added to tasks

Findings

☒ All **5287**

☒ Code Anomalies **1141**

☒ Bad practice 1045

☒ Correctness 8

☒ Exception Handling 24

☒ Formatting 31

☒ Null pointer dereference 1

- Vorteil ggü. Metriken-orientierten Tools: konkrete Qualitätsdefekte werden angezeigt und es ist klar, wo und wie sie zu beheben sind.

Erläuterungen zur Findings-Perspektive

- In der Liste steht jede Zeile für ein Finding
- In der rechten Leiste können einige Filter gesetzt werden, z.B. nach Kategorie
- Ein Klick auf einen Eintrag in der Liste zeigt die entsprechende Stelle im Code an
- In dieser Perspektive findet die eigentliche Wartung statt: Wartungsbeauftragte können die Findings Liste Schritt für Schritt durchgehen und die Qualitätsdefekte beheben

Metriken-orientierte Tools: Plugin für Sonarqube

Method																		
Number of rows: 5		←	→	page 17/2152		Filter: <input type="text"/>		Column filter: <input type="text"/>										
		Name ↓	HVOL	MISM	NII	LOC	LLOC	McC	NUMPAR	NOS	CD	CLOC	DLOC	NLE	CCO	CI	CLLC	NOI
81	Java	AntClassLoader2()	8	104.07	0	2	2	1	0	0	0.33	1	1	0	0	0	0	1
82	Java	AntClassLoader5(ClassLoader parent, Project project, Path classpath, boolean parentFirst)	57.36	91.89	0	4	4	1	4	1	0.8	16	16	0	0	0	0	1
83	Java	AntFTPFile(AntFTPFile parent, String path)	1168.79	31.81	1	43	42	10	2	25	0.13	6	5	5	4	1	0.4	8
84	Java	AntFTPFile(AntFTPFile parent, String path)	1201.34	31.69	0	43	42	10	2	26	0.13	6	5	5	4	1	0.38	8
85	Java	AntFTPFile(FTPClient client, FTPFile ftpFile, String curpwd)	109.39	83.89	2	5	5	1	3	3	0.55	6	6	0	0	0	0	0

- Zeilen stehen für Methoden, Spalten für Metriken.
- Qualitätsbewertung: rote Einträge verletzen Grenzwerte, grüne nicht. Schwarze Werte unterliegen keiner Bewertung.
- Mit einem Klick auf Spaltenköpfe können die Metriken sortiert werden.
- Ein Klick auf die Methode führt zum Code der zugehörigen Klasse.

Eigener Prototyp

Erinnerung: Ziel ist, Priorisierung von Methoden zu erlauben

Batch Skript

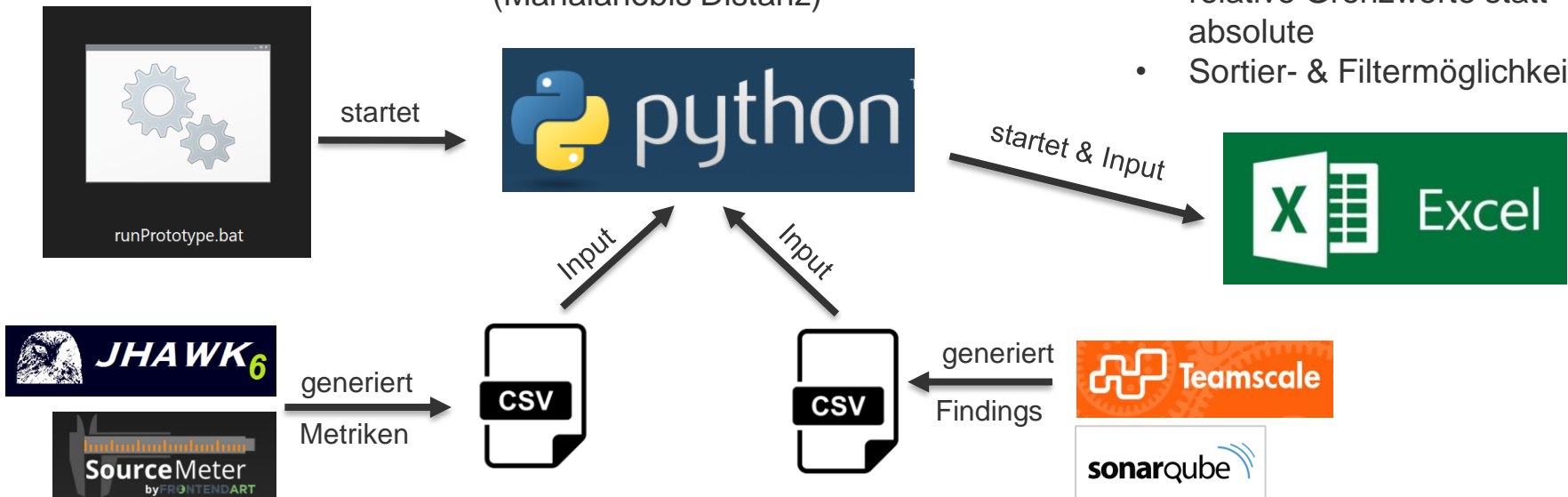
startet Python Skript,
dann Excel Datei

Python Skript

- Inputs: 2 CSV Dateien mit Metriken auf Methodenebene und Findings
- Datenbereinigung
- ordnet Methoden Findings zu
- Normierung von Werten zw. 0 und 1
- berechnet multivariates Distanzmaß (Mahalanobis Distanz)

Excel Datei

- Pivottabelle zur eigentlichen Analyse
- Datenbasis: von Python Skript bereitgestellte Daten → automatisierter Import und Aktualisierung beim Öffnen
- relative Grenzwerte statt absolute
- Sortier- & Filtermöglichkeiten



mögliche Ansicht beim ersten Öffnen

	A	B	C	D	E	F	G	H	I	J
1							10			
2					Top x% (x in G1 eingeben) anzeigen und überdurchschn. Werte fett anzeigen			Normed Score anzeigen/ausblenden		
3										
4										
5										
6	Zeilenbeschriftungen	Summe von HDFI	Summe von McC	Summe von NL	Summe von CD	Summe von LLOC	Summe von NOS	Summe von Findings	Summe von MahalanobisDistance	Summe von NormedScore
7	L1000***int getColumnCount()***	2	1	0	0	4	1	0	4,833798082	0,856224311
8	L1001***Object getValueAt(int row, int column)***	13,7143	2	1	0	6	3	3	4,582767534	0,830007259
9	L10015***String getName()***C:\	4	1	0	0,2	4	1	0	2,792852637	1,044629675
10	L10017***int getDesiredComponentCount()***	3,5	1	0	0,2	4	1	0	3,091569312	1,018669922
11	L10019***void filterImage(ComplexImageFilter filter)***	25,8333	3	1	0,272727	16	11	2	21,35133136	1,043189206
12	L10025***float[] createMask(double[] mask)***	48,5395	8	4	0,0222222	44	37	9	50,6411245	0,96030256
13	L10027***float[] findOutline(ComplexImageFilter filter)***	77,1597	33	8	0,0649351	72	72	34	109,4044974	1,441856512
14	L10029***void applyOutline(ComplexImageFilter filter)***	36	3	2	0,0769231	12	12	3	12,07393434	0,945971001
15	L10034***boolean isOutline(float[] mask)***	19,5278	7	1	0,0588235	16	13	5	8,08560481	0,848904142
16	L1004***String getColumnNames()***	7,2	2	0	0	4	1	1	4,309063916	0,832483116
17	L10042***void drawOutlineSpot(int x, int y)***	28,7778	8	3	0,047619	20	18	8	45,58273389	0,915481225
18	L1006***boolean isCellEditable(int row, int column)***	4,375	1	0	0	4	1	0	4,252742085	0,830952584
19	L10066***float calcMaskValue(double[] mask)***	15,1667	3	1	0	9	5	3	6,632791393	0,811266014
20	L10071***Property[] getPropertyNames()***	9,06667	1	0	0	10	1	0	12,13177688	0,703139994
21	L10072***Property(String name, float value)***	6,42857	1	0	0,333333	6	3	0	5,51252154	1,118533463
22	L10074***void writeToStream(DataOutputStream out)***	20,0556	1	0	0,0833333	11	8	0	6,370740678	0,825084971
23	L10078***void initFromStream(DataInputStream in)***	20,9	2	1	0,0909091	10	7	1	2,762213688	0,885422489
24	L10083***String getName()***C:\	4	1	0	0,2	4	1	0	2,685801097	1,044629675
25	L10085***void filterImage(ComplexImageFilter filter)***	57,0323	6	3	0,166667	30	33	6	33,54381056	1,102440163
26	L10091***float getBrightness()***	3,5	1	0	0,2	4	1	0	3,887985624	1,021891016
27	L10093***TextureParameter[] getParameters()***	6	1	0	0,2	4	1	0	3,878952599	1,022361625
28	L10095***TextureParameter(Object obj)***	6	1	0	0	10	7	1	13,74538753	0,712462488
29	L10098***Property[] getPropertyNames()***	6	1	0	0	5	1	0	4,05441584	0,796563155
30	L10099***void writeToStream(DataOutputStream out)***	8,125	1	0	0,2	4	1	0	3,808495075	1,032205443
31	L10103***void initFromStream(DataInputStream in)***	6,875	1	0	0,2	4	1	0	3,475286385	1,032790549
32	L10108***String getName()***C:\	4	1	0	0,2	4	1	0	2,72993503	1,044629675
33	L10110***void filterImage(ComplexImageFilter filter)***	14,4	1	0	0,461538	7	4	0	12,97964418	1,239468713
34	L10116***void filterComponent(ComplexImageFilter filter)***	31,5	3	2	0,1	9	11	4	12,52957841	0,985262103
35	L10122***Property[] getPropertyNames()***	6	1	0	0	5	1	0	4,475655093	0,793699287
36	L10124***void writeToStream(DataOutputStream out)***	18,7	1	0	0,125	7	4	0	6,360175538	0,916207614
37	L10128***void initFromStream(DataInputStream in)***	12,25	1	0	0,2	4	1	0	4,095368833	1,032574777
38	L10135***HIDRIImage createlImage(int width, int height)***	81,2313	27	6	0,06	141	111	17	274,2187162	1,42539619
39	L10157***String readLine(InputStream in)***	26	3	1	0,0909091	10	5	0	10,1021913	0,920950399
40	L1017***ValueSelector(double value)***	17,6957	1	0	0,37931	18	14	0	78,02093162	1,0783084
41	L10175***HIDRIImage[byte[] r, byte[] g, byte[] b]***	9,28571	1	0	0,142857	6	3	0	23,50818372	0,924614573
42	L10184***void writelImage(ComplexImageFilter filter)***	53,7778	4	3	0,0285714	34	32	5	24,8497754	0,926724876
43	L1020***Component getTableCellAt(int row, int column)***	15,4375	2	0	0	6	3	2	21,97399441	0,795206929
44	L10202***int getERGB()***C:\User\	23,92	4	1	0,0666667	14	14	5	11,00508162	0,845573882
45	L10226***ImageMap()***C:\User\	2,5	1	0	0	4	1	1	4,351995251	0,840509639
46	L10228***void buildMipMaps(byte[] data)***	58,1678	20	4	0,0747664	99	110	13	194,7426708	1,20696465

Sortiert nach Anzahl Findings

	A	B	C	D	E	F	G	H	I	J
1							10			
2					Top x% (x in G1 eingeben) anzeigen und überdurchschn. Werte fett anzeigen			Normed Score anzeigen/ausblenden		
3										
4										
5										
6	Zeilenbeschriftungen	Summe von HDIF	Summe von McC	Summe von NL	Summe von CD	Summe von LLOC	Summe von NOS	Summe von Findings	Summe von MahalanobisDistance	Summe von NormedScore
7	L18441***TriangleMesh subdivide	234,237	95	7	0,0954654	379	387	112	2801,386751	3,162158749
8	L30503***void doSimplification()	179,945	95	6	0,0666667	322	267	102	1531,154057	2,672155951
9	L41533***void importFile(BFrame	248,773	96	11	0,0736041	365	287	97	3801,339487	3,166223535
10	L41728***void writeObjects(Scene	116,215	47	12	0,0904393	352	327	91	2811,278758	2,442314863
11	L18443***TriangleMesh subdivide	181,909	81	7	0,109181	359	323	89	1263,576898	2,774996163
12	L32900***void compile(String mac	94,9	92	6	0,132159	197	185	80	906,5964756	2,164571932
13	L42110***void processMessage(in	141,633	52	8	0,363636	252	184	79	2600,078968	2,405542232
14	L37452***void renderTriangleHyb	215,583	54	8	0,01059	654	661	76	2995,524141	3,519016705
15	L18538***void doSubdivide(Triang	133,121	51	9	0,0615385	305	237	75	1786,999996	2,211464593
16	L30014***TriangleMesh bevelFace	142,066	92	7	0,156667	253	245	69	1586,900515	2,51983061
17	L30016***TriangleMesh bevelEdge	147,98	117	8	0,0943878	355	332	69	4021,965869	2,93978672
18	L37416***void renderTriangleGou	211,14	40	6	0,011041	627	645	65	2892,218238	3,277700401
19	L30334***TriangleMesh doJoinBo	124,331	65	5	0,0576923	147	145	64	420,9757471	1,853017947
20	L35845***void processMessage(in	116,402	32	9	0,333333	164	98	63	845,7080101	1,965615218
21	L15265***RenderingMesh getRend	155,674	54	7	0,0807692	239	205	61	666,9748832	2,154212703
22	L4396***void addTracks()***C:\Us	160,299	39	5	0,0359712	134	128	61	334,72219	1,734109606
23	L37491***void renderTrianglePho	215,327	56	7	0,0124113	557	535	60	1871,911101	3,181762125
24	L41465***void setVertexParamete	167,585	39	5	0,0352941	164	152	58	2614,285987	1,827458063
25	L32916***BufferedImage executel	104,015	94	4	0,238683	185	170	56	1154,13664	2,218554101
26	L35865***void download(BFrame	8,25	1	0	0,272727	8	4	54	6516,985454	1,0278247
27	L12167***void actionPerformed(C	94,359	76	21	0	195	154	54	2315,752757	2,38005496
28	L39280***double spawnRay(Rende	193,151	71	5	0,116959	302	233	53	732,7605241	2,478574368
29	L15007***void splitFaces(Vector<	119,631	83	4	0,0495868	230	191	53	825,9369563	2,067022995
30	L41011***Object3D extrudeMesh	205,976	64	4	0,1	225	241	51	831,5943923	2,347627874
31	L39192***void run()***C:\Users\A	92,7229	38	6	0,132948	150	150	50	2111,825582	1,67681896
32	L9111***TriangleMesh getMesh(ir	168,286	86	9	0,0882353	186	151	48	570,4229219	2,368038665
33	L9005***int findClickTarget(Point	166,574	90	5	0,0298507	195	153	48	1330,891901	2,20340581
34	L34267***void renderFlatTriangle	88,9142	57	8	0,00324675	307	266	46	2356,596973	2,067836219
35	L41462***void writeScene(Scene t	140,264	57	7	0,0660377	198	175	46	373,7943414	2,019528169
36	L36102***void readInfoFromDocu	110,262	60	4	0,134387	219	199	45	660,6186489	1,960181164
37	L35912***void ok(CommandEvent	25,575	3	2	0	27	18	45	3505,343902	0,771551807
38	L14167***void setFaceParameters	135,88	49	6	0,0440752	152	147	45	232,5780992	1,811142169

Sortiert nach Mahalanobis Distanz

	A	B	C	D	E	F	G	H	I
1					Top x% (x in G1 eingeben) anzeigen und überdurchschn. Werte fett anzeigen		10	Normed Score anzeigen/ausblenden	
2									
3									
4	Path	(Alle)							
5									
6	Zeilenbeschriftungen	Summe von HDIF	Summe von MCCC	Summe von NL	Summe von CD	Summe von LLOC	Summe von NOS	Summe von Findings	Summe von MahalanobisDistance
7	L35865***void download(BFrame	8,25	1	0	0,272727	8	4	54	6516,985454
8	L15088***void splitOneFace(Vect	114,601	98	10	0,0431034	444	287	33	5665,487473
9	L30016***TriangleMesh bevelEdge	147,98	117	8	0,0943878	355	332	69	4021,965869
10	L41533***void importFile(BFrame	248,773	96	11	0,0736041	365	287	97	3801,339487
11	L37262***ComplexImage createFii	41	10	6	0,0454545	42	40	31	3655,742976
12	L42504***void setFont(Font font)	4,66667	1	0	0,97351	4	2	0	3612,309779
13	L35912***void ok(CommandEvent	25,575	3	2	0	27	18	45	3505,343902
14	L37452***void renderTriangleHybri	215,583	54	8	0,01059	654	661	76	2995,524141
15	L37416***void renderTriangleGou	211,14	40	6	0,011041	627	645	65	2892,218238
16	L41728***void writeObjects(Scene	116,215	47	12	0,0904393	352	327	91	2811,278758
17	L18441***TriangleMesh subdivide	234,237	95	7	0,0954654	379	387	112	2801,386751
18	L39104***void setConfiguration(Si	17,0308	30	29	0	63	60	31	2741,18761
19	L14165***void setVertexParamete	167,585	39	5	0,0352941	164	152	58	2614,285987
20	L42110***void processMessage(in	141,633	52	8	0,363636	252	184	79	2600,078968
21	L34267***void renderFlatTriangle(88,9142	57	8	0,00324675	307	266	46	2356,596973
22	L12167***void actionPerformed(C	94,359	76	21	0	195	154	54	2315,752757
23	L34279***void renderSmoothTriar	99,6809	32	5	0,00273224	365	362	32	2243,15766
24	L39192***void run()***C:\Users\	92,7229	38	6	0,132948	150	150	50	2111,825582
25	L37491***void renderTrianglePhoi	215,327	56	7	0,0124113	557	535	60	1871,911101
26	L18538***void doSubdivide(Triang	133,121	51	9	0,0615385	305	237	75	1786,999996
27	L38046***void generatePhotons(P	20,8	2	1	0,421053	11	7	18	1743,239149
28	L823***String text(String name)**	7	2	1	0,153846	11	3	0	1703,04643
29	L38780***void addObject(ObjectI	288,194	53	5	0,015625	189	137	42	1677,721249
30	L30014***TriangleMesh bevelFace	142,066	92	7	0,156667	253	245	69	1586,900515

Fazit

- State-of-the-Art SATs existieren, geben allerdings keine Informationen auf Methodenebene.
- Tools, die Informationen auf Methodenebene geben, haben deutliche Schwächen ggü. den o.g. SATs.
- Das entwickelte Tool erlaubt es, Methoden miteinander zu vergleichen und im Kontext der Wartung zu priorisieren
- Es ist vorstellbar, die Konzepte des Tools als Plugin in ein SAT umzusetzen → Kombination des Findings-orientierten und Metriken-orientierten Ansatzes