

**Die verrückte Sightseeingtour
Metrik-Analyse**

Version 1.1

Die verrückte Sightseeingtour	Version: 1.1
Metrik-Analyse	Date: 6/18/2013

Revision History

Date	Version	Description	Author
6/18/2013	1.1	2 Metriken analysiert	Janina Schilling, Christiane Helmchen, Yvonne Meininger

Die verrückte Sightseeingtour	Version: 1.1
Metrik-Analyse	Date: 6/18/2013

Table of Contents

Revision History	2
Table of Contents	3
Metrik-Analyse	4
1. Screenshots von der Analyse mit Sonar	4
2. Erläuterungen der Metriken	5
3. Analyse von einzelnen Metriken	5
3.1 Cyclomatic Complexity mit Sonar	5
3.2 Depth of Inheritance Tree mit Metrics-Tool für Eclipse (DIT)	6

Die verrückte Sightseeingtour	Version: 1.1
Metrik-Analyse	Date: 6/18/2013

Metrik-Analyse

1. Screenshots von der Analyse mit Sonar



Abbildung 1: Dashboard



Abbildung 2: Hotspots



Abbildung 3: Word-Cloud zeigt alle Klassen (je größer desto öfter referenziert)

Die verrückte Sightseeingtour	Version: 1.1
Metrik-Analyse	Date: 6/18/2013

2. Erläuterungen der Metriken

Englische Erläuterungen zu den von Sonar gemessenen Metriken zu finden unter <http://docs.codehaus.org/display/SONAR/Metric+definitions>

3. Analyse von einzelnen Metriken

3.1 Cyclomatic Complexity mit Sonar

Erläuterung:

Anzahl der verschiedenen Wege durch eine Methode

Je größer die Anzahl, desto unübersichtlicher sind die verschiedenen Endzustände und es ist schwer ersichtlich, welche Fälle abgedeckt werden.



Abbildung 4: Klassen, welche mindestens eine Methode mit einer hohen Cyclomatic Complexity enthält



Abbildung 5: Klasse SpielplattenEinschieber mit der relevanten Methode

Interpretation:

- Problem: mehrere IF-Anweisungen mit jeweils mehreren mit OR verknüpften Bedingungen

Idee zur Verbesserung:

- innerhalb der IF-Bedingung eine Methode aufrufen, die die verschiedenen Optionen ermittelt und einen Boolean zurückliefert

Die verrückte Sightseeingtour	Version: 1.1
Metrik-Analyse	Date: 6/18/2013

3.2 Depth of Inheritance Tree mit Metrics-Tool für Eclipse (DIT)

Erläuterung:

Maximale Pfadlänge von der Wurzel bis zur betrachteten Klasse (Vererbungstiefe)



Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Methods (avg/max per type)	204	4.08	4.617	23	/DVST/src/com/dhbw/dst/models/Spielplatte.java	
Nested Block Depth (avg/max per method)		1.228	0.55	3	/DVST/src/com/dhbw/dst/activities/SpielActivity.java	openKartenAnkündigung
Depth of Inheritance Tree (avg/max per type)	2.4	1.709		6	/DVST/src/com/dhbw/dst/activities/SpielerAnlegen...	
Number of Packages	6					
Referent Coupling (avg/max per packageFragen)		12.167	9.045	25	/DVST/src/com/dhbw/dst/models	
Number of Inheritance Facets (max inheritanceFacet)	7	1.147	1.690	7	/DVST/src/com/dhbw/dst/Classes	

Abbildung 6

Mittel	Standardabweichung	Maximum	Klasse mit höchstem DIT
2,4	1,709	6	SpielerAnlegenActivity

Interpretation:

- recht hohe Pfadlänge durch weitere Vererbung der von Android zur Verfügung gestellten Klassen
 - wir haben selbst nur eine Vererbungsstufe eingebaut (SpielerAnlegenActivity erbt von SpielerActivity)
 - aber SpielerActivity muss von der Android-Klasse Activity erben
 - alle Oberklassen von Activity stellt Android automatisch

Idee zur Verbesserung:

- Activity muss verwendet werden, somit kaum Verbesserung möglich
- zusätzlich eingebaute Vererbungsstufe ist hilfreich zur Vermeidung von Coderedundanz, da die SpielerBearbeitenActivity ähnliche Funktionen hat (Problem: starke Abhängigkeit von der Android Activity)