Überdeckung der Schwachstellen aus OSGi Katalog mit SSISy

Ein Vergleich

Es gibt zwei Gesichter der Softwarequalität: die äußere Sicht, von Benutzer wahrgenommen, wird durch Funktionalität, Korrektheit, Bedienbarkeit und Performanz charakterisiert. Die innere Sicht dagegen wird von Entwickler wahrgenommen und durch Verständlichkeit, Wartbarkeit, Erweiterbarkeit und Änderbarkeit charakterisiert.

# SSISy

-steht für **S**tructural **I**nvestigation of **S**oftware **Sy**stems und ist eine Plattform, die zur Untersuchung von OO-Softwaresystemen dient. Es unterstützt die Sprachen Java, C++ und Delphi. Es kann Softwaremetriken berechnen, Codeduplicate und Vielzahl an Strukturprobleme erkennen. SSISy ist bereits in der Lage, mehr als 50 praxiserprobte Strukturprobleme zu erkennen und kann jederzeit durch eigene SQL-Anfragen erweitert werden.

Die Berechneten Softwaremetriken sind:

* Anzahl Codezeilen, Klassen, Methoden, Attribute
* Vererbungstiefe, zyklomatische Komplexität, Kopplung, Kohäsion. Die Problemmuster sind

Problemmuster

* Architekturebene: Paket- und Subsystemstruktur
  + Gottpacket
  + Importlüge
  + Verbotene Paketliebe
* Entwurfsebene: Vererbungshierarchien, Klasseninteraktion und –Verkapselung
  + Datenkapselaufbruch
  + Generationskonflikt
  + Gottklasse,
  + Tote Methode
  + Tote Attribute
* Implementierungsebene: innerhalb einer Methode
  + Labyrinthmethode
  + Lange Parameterliste
  + Versteckte Konstantheit

Codeduplikate

* Implementierungsebene: duplizierter Code
* Entwurfsebene: Signaturähnliche Klassen

Katalog der Strukturellen Problemen

1. Überschreibung von Attribute:   
   ein nicht statisches Attribut wird in einer Unterklasse überschrieben. Vielleicht bringt ein erwünschtes Effekt, es macht aber das Quellkode unübersichtlich
2. Zu komplexe Methode:  
   die Methode hat eine zyklomatische Komplexität höher als 10 (= mehr als 9 Sprünge in Kontrollfluss. Diese sind durch Anweisungen wie if, else, case und durch Schleifen verursacht). Es wird schwieriger, Tests für die Methode zu erstellen und Übersicht über den Kontrollfluss zu bewahren.
3. Neudefinierung von Konstanten
4. Zyklische Abhängigkeiten zwischen den Klassen
5. Zyklische Abhängigkeiten zwischen den Paketen
6. Zyklische Abhängigkeiten zwischen den Quellkodedateien
7. Zyklisches Aufrufen von Methoden
8. „tote“ Attribute (nicht referenzierte)
9. Totes Kode (nicht benutzt)
10. Tote Imports
11. Tote Methoden
12. Duplizierter Kode (ab 50 Zeilen)
13. Zu allgemeine Parameter ( wird in der Methode zu einer Subklasse gekastet)
14. Generationenkonflikt (mehr als 50% wird von Unterklassen überschritten
15. Gott-Klasse – Viele Attribute
16. Gott-Klasse – Viele Methoden
17. Gott-Datei – mehr als 2000 LOC
18. Gott-Methode – mehr als 200 LOC
19. Gott-Paket – mehr als 50 öffentliche, nicht verschachtelte Klassen oder Interfaces
20. Ignorierte Abstraktion
21. Importchaos
22. Ungeeignete Namenslänge
23. Unangemessene Kommentare
24. Inkonsistente Operationen – semantische Operationen nicht geeignet implementiert
25. Nicht formelle Kommentare (später nicht in Docs sichtbar)
26. Interface- Umgehung
27. Vorwissen von Unterklassen – bei der Kompilierung bhängigkeit von Unterklassen
28. Späte Abstraktion – eine Abstrakte Klasse hat eine direkte oder indirekte nicht Abstrakte Oberklasse
29. Lange Parameterliste
30. Mini-klassen (weniger als 3 Methoden und 3 Attribute)
31. Mini-Pakete (weniger als 3 Klassen und Interfaces)
32. Irreführende Namensgebung der Quellcodedateien
33. Objekt-Placebo – Attribute (Addressierung von statischen Attributen durch eine Instanz der Klasse
34. Objekt-Placebo – Methoden
35. Weise Geschwister – Attribute
36. Weise Geschwister – Methoden
37. Überfüllten Quellcodedateien (mehr als eine Klasse in einer Datei)
38. Tolerante Sichtbarkeit der Attributen
39. Tolerante Sichtbarkeit der Methoden
40. Polymorphes Aufruf im Konstruktor
41. Polymorphismus-Placebo
42. Abgelehnte Vererbung in der Implementierung
43. Abgelehnte Vererbung der Interface
44. Gegenläufige Paket- und Interfacehierarchie
45. Riskantes Code
46. Ähnliche aber nicht verwandte Abstraktionen
47. Simuliertes Polymorphismus
48. Gleicher Name der unterschiedlichen Typen
49. Unfertiger Quellcode
50. Variablen mit konstanten Wert
51. Missachtung der Datenverkapselung
52. Missachtung der Namenskonvertion