Relevante Metriken zur Bestimmung von Softwarequalität

Steffen Förster

05.09.2012

Definitionen

- Metrik
  - Eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.
    (IEEE 1061)

Definitionen

- Qualität
  - Unter Softwarequalität versteht man die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.
    (DIN ISO 9126)
  - Qualität ist die Übereinstimmung von Produkteigenschaften mit den Erwartungen des Kunden. (ISO 9000)

Agenda

1. Problemstellung
2. Bestandsaufnahme
3. Evaluation
4. Umsetzungskonzept
5. Implementierung
6. Ergebnisse
7. Fazit
8. Ausblick

1. Problemstellung

- Qualitätsmessung einer FM-Software
- Evaluierung relevanter Kriterien
- Aufstellung eines Qualitätsmodells

1. Problemstellung

- Etablierung automatisierter Messungen
- Qualitätsmessung bisher kein Entwicklungsbestandteil
- Anpassungen des Entwicklungsprozesses

2. Bestandsaufnahme

- kein vollständig dokumentierter Entwicklungsprozess mit integrierter Qualitätsanalyse
- keine bewusste Messung der erreichten Qualität
- sehr große evolutionär gewachsene Codebasis

2. Bestandsaufnahme

- in Ansätzen agile Entwicklungsmethoden
- Dokumentation in Redmine
- wenige automatisierte Softwaretests
- Durchführung eines Interviews mit Entwicklungsbeteiligten

3. Evaluation – Metriken

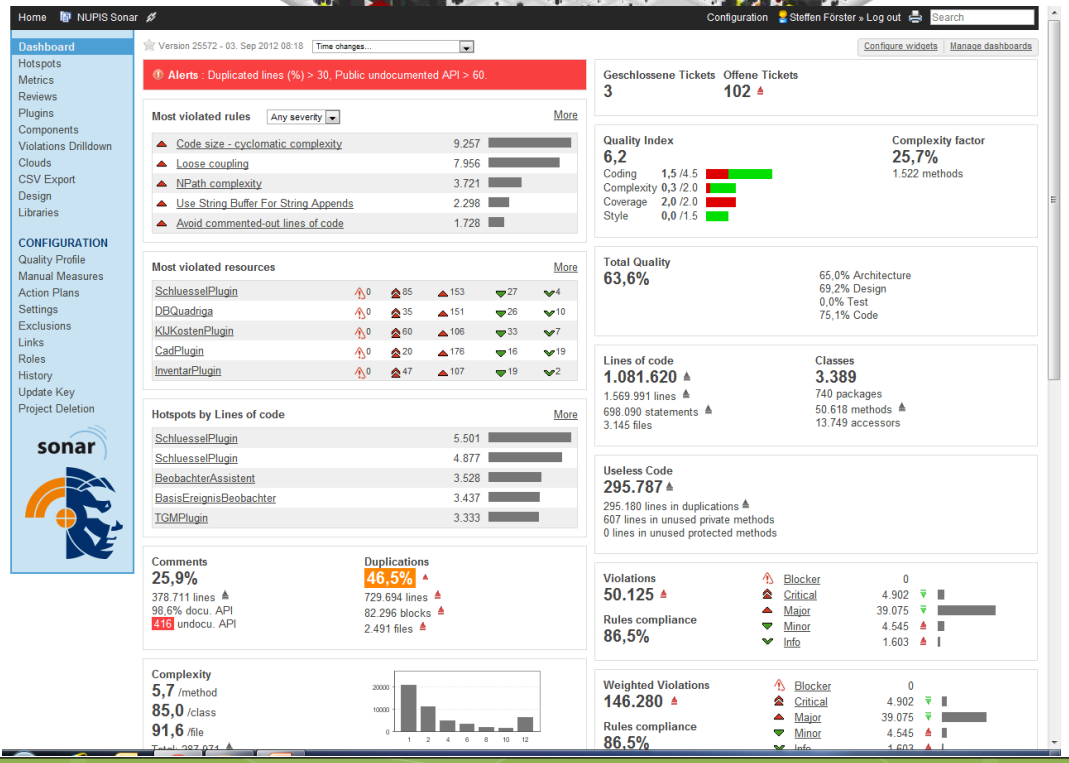
- Größenmetriken
  - LoC, NCSS
  - Function Points
- Komplexitätsmetriken
  - Halstead
  - Mc Cabe

3. Evaluation – Metriken

- Architekturmetriken
  - CK-Suite, LK-Suite
  - MOOD, Martin Metriken
- Sonstige Metriken
  - Anforderungsmetriken
  - Aufwände
  - Testabdeckung

3. Evaluation – Tools

- JDepend
- iPlasma und inFusion
- CodeCity
- STAN4J
- Checkstyle
- Findbugs
- PMD
- Sonar



3. Evaluation – Modelle

- CMMI als Referenzrahmen der Prozesse
- SPICE als Bewertungsmaßstab für den Entwicklungsprozess
- SQALE als Qualitätsmodell
- GQM als Methode ein Qualitätsmodell aufzustellen

4. Umsetzungskonzept

- Entscheidung für GQM
- Interview mit Stakeholdern
- Aufstellung des GQM Modells
- Implementierung einer Continous Inspectation
- Etablierung einer stetigen Anpassung des Modells

4. Umsetzungskonzept – Qualitätsmodell

- Ziele:
  - *Welche Produktqualität erreicht unser Produkt unter dem aktuellen Mitteleinsatz?*
  - *Welche Produktqualität kann unser Produkt unter dem aktuellen Mitteleinsatz erreichen?*
- *20 Fragen*
- *26 Metriken*

4. Qualitätsmodell - Auszug

- **Frage 9:** *Wie gut halten sich die Entwickler an die vorgegebenen Programmierrichtlinien?*
  - ***Metrik:*** *Richtlinienverletzung(Anzahl), Richtlinieneinhaltung (%)*
- **Frage 15:** *Wie komplex sind die einzelnen Artefakte?*
  - ***Metrik:*** *Komplexität/Klasse, Komplexität/Methode, WMC, COF*

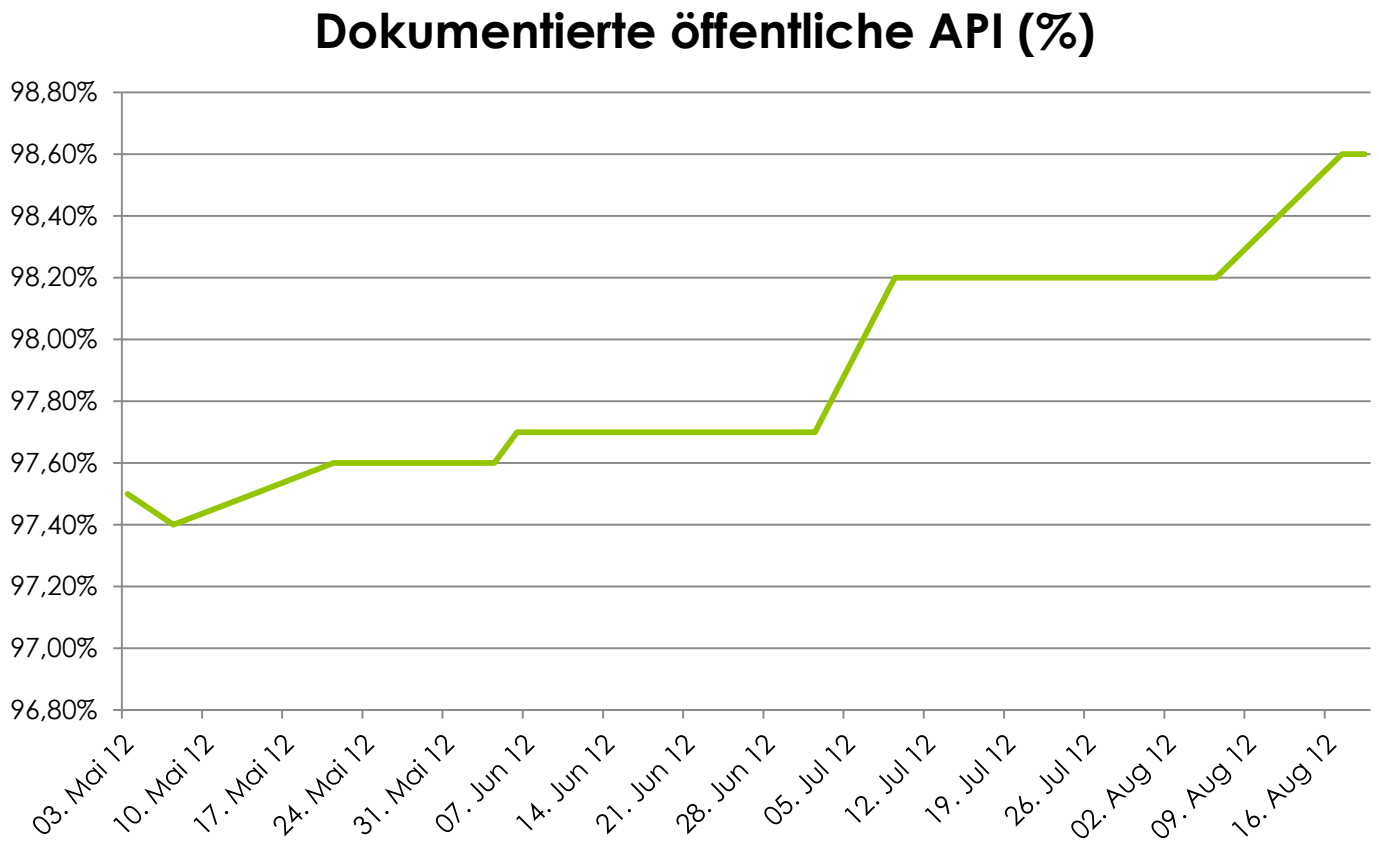
5. Implementierung

- Ausbau Nutzung von Continuous Integration
- Integration der täglichen Codeanalyse
- Einführung der Continuous Inspectation mit Sonar
- einsehbare Ergebnisse für alle Entwickler

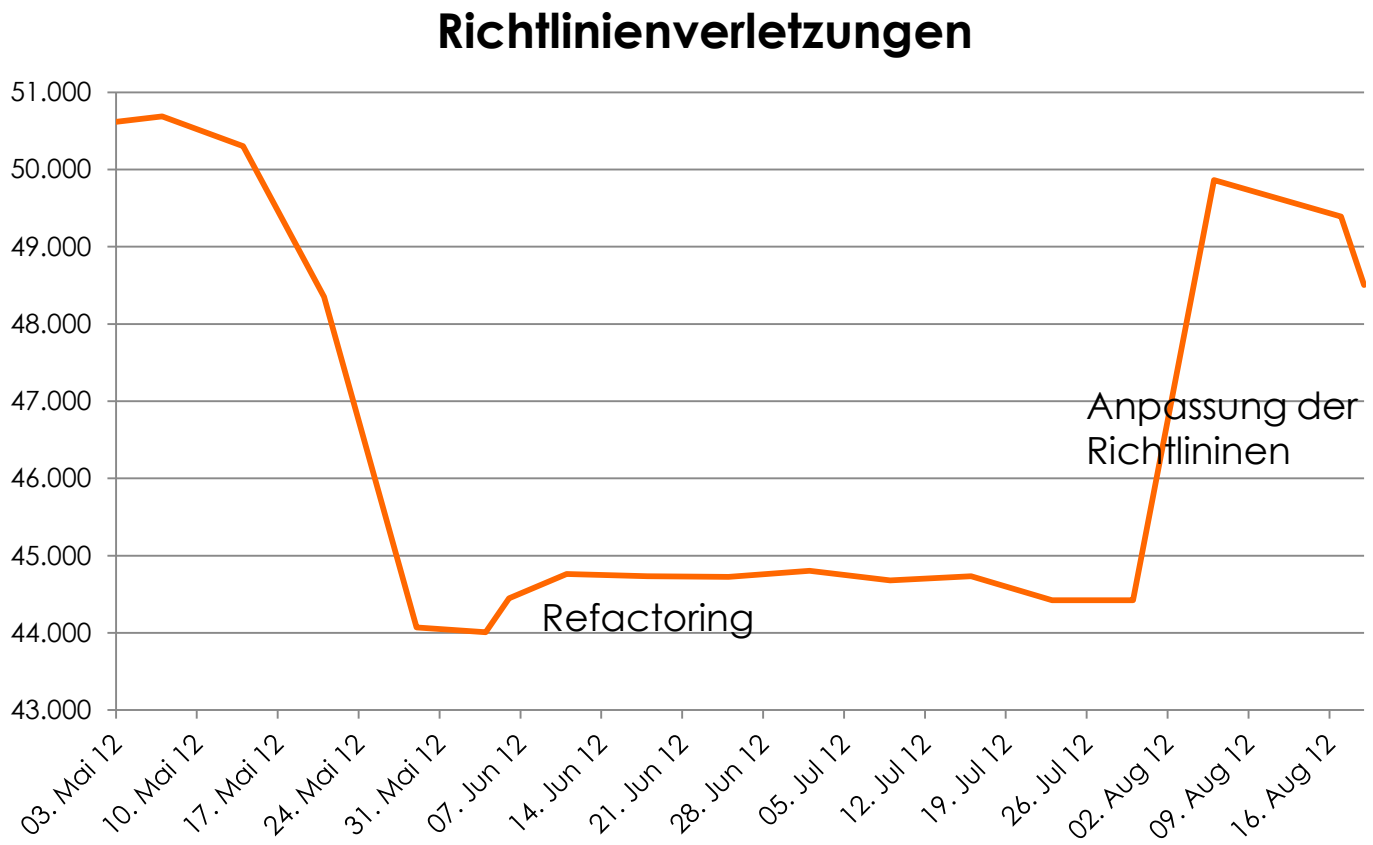
6. Ergebnisse

- höhere Dokumentationsdisziplin
- Entwicklung eines besseren Qualitätsbewusstseins
- Start zu einer nachhaltigen Qualitätsverbesserung
- Erkennung der Schwachstellen in der Codebasis

6. Ergebnisse



6. Ergebnisse



7. Fazit

- Für die Bestimmung von Softwarequalität sind Metriken ein geeignetes Mittel.
- CMMI kann den Rahmen für eine Strukturierung bilden.
- Zur Erstellung eines Qualitätsmodells ist GQM im Kontext von CMMI zielführend.
- Der Reifegrad konnte nicht gesteigert werden.
- Einige Capability Levels konnten verbessert werden.

7. Fazit

- Besonders relevante Metriken
  - WMC zusammen mit LCOM4
  - Richtlinieneinhaltung
  - Test-Abdeckung
  - Technische Schuld
- Es konnte eine durchschnittliche bis gute Gesamtqualität gemessen werden
- Continous Inspectation hilft zukünftig bei der Trendanalyse

8. Ausblick

- Regelmäßige Anpassungen des Qualitätsmodells planen
- Etablierung einer unternehmensweiten Messung
- Verantwortung verteilen

8. Ausblick

- Optimale Nutzung der vorhandenen Daten durch besseres Reporting
- Einbindung historischer Aufwandsdaten zur Abschätzung zukünftiger Entwicklungen
- Bessere Integration zwischen Sonar und Redmine

Metrik

Fragen?

Qualität

Diskussion

- Warum wurden fast ausschließlich Codemetriken in Betracht gezogen?
- Welche Bereiche sollten noch betrachtet werden?
- Ist ein generisches Qualitätsmodell sinnvoll?

Thesen

1. Für die Bestimmung von Softwarequalität sind Metriken ein geeignetes Mittel.
2. Schlechte Codequalität sorgt für Toleranz gegenüber neuem schlechten Code.
3. Die Auswahl von Softwaremetriken muss dem gewünschten Umfang der Qualitätsaussage entsprechen.
4. Die Werte von Softwaremetriken müssen dem Kontext entsprechend interpretiert werden.
5. Goal Question Metric ist eine geeignete Methode um im Rahmen von Capability Maturity Model Integration ein Qualitätsmodell zu definieren.
6. Die Nutzung von Continuous Integration unterstützt die Nutzung von Softwaremetriken in Rahmen von Continuous Inspectation.
7. Transparente Verfahren und offene Tools verhelfen der Qualitätsmessung zu einer hohen Akzeptanz unter den Entwicklern.
8. Eine institutionalisierte regelmäßige Messung kann die Produktivität eines Entwicklungsteams steigern.