

Technische Qualitätssicherung

XPUG FFM 57

Bastiaan Harmsen

03.03.2009

Inhalte

- Überblick Techniken
- Code Walks
- Werkzeuggestützte Techniken

„Technische“ QS?

- Maßnahmen, die „nah am Endprodukt“ durchgeführt werden können (und müssen)
- Maßnahmen, die zur Verhinderung von Katastrophen dienen
- Gegensatz zu: „QS für Dokumente“

Techniken

- Code Walks, Code Inspections (*)
- Design Inspections
- Architecture Inspections
- Werkzeuggestütztes (*)
 - Automatische Audits
 - Metriken
 - Copy&Paste Detector
 - Analyse Log-Files (z.B. von Application Servern)
 - Exceptions, Fehlermeldungen
 - Ressource quotas, Maschinenauslastung (Mem, Disk, Net)

Code Walks / Inspections

- „Gemeinsames lesen“ von Source Code
 - verschiedene „Formate“ (Anzahl Teilnehmer, ...)
- Bevorzugt Source Code, der vom Entwickler vorgeschlagen wird
- Nicht ausschließlich Source Code - weitere Artefakte (Design, Requirements)

Code Walks

- Vorbereitung
 - Terminabstimmung, Bereitstellung von Source Code (SC) durch Entwickler (richtige Version erwischen!), Lesen durch Reviewer, ausdrucken, markieren, kommentieren, in IDE Kontext browsen, Fragen im Vorfeld abstimmen, evtl. anderen SC wählen
- Walk the walk (walk your talk)
 - 1h üblicherweise, gemeinsam durch den SC (+Artefakte) gehen, sich erklären, was man versteht / nicht versteht, „Verstöße“ gegen Programmierrichtlinien feststellen, „Issues“ finden + dokumentieren
- Nachbereitung
 - Dokumentation, Protokollierung, Abstimmung Protokoll, Auswertung (z.B. welche Probleme, Smells, ... wurden gefunden), Tracing

Code Walks

Voraussetzungen (technisch / organisatorisch)

- Programmierrichtlinien (vorher publiziert + aktuell)
- Hinweise auf unerwünschte Smells
- Zugriff auf SC (unmittelbar, alle Teile+Versionen)
- Werkzeuge zur Analyse (IDE, Audits & Metriken, ...)
- Eingeplant im der Zeitrahmen der Entwicklung
- Rückhalt bei Projektleitung, Entwicklung
- „Heiliger Ort“ (alias „the place“)

Code Walks

Voraussetzungen (psychologisch)

- gefestigte Persönlichkeit
- mit Themen und Technik vertraut
- „egoless“
- sollte Hamburger, Sekt, Smarties, Filet mignon, Schnellpizza, Mager-Joghurt (alles durcheinander) ... nicht unbedingt mögen, aber verdauen können
- selbst korrekt in der Umsetzung

Exkurs: egoless programming

- 10 Gebote
 - Understand and accept that you will make mistakes
 - **You are not your code**
 - No matter how much "karate" you know, someone else will always know more
 - Don't rewrite code without consultation (**hmmm?**)
 - Treat people who know less than you with respect, deference, and patience
 - The only constant in the world is change
 - The only true authority stems from knowledge, not from position
 - Fight for what you believe, but gracefully accept defeat
 - Don't be "the guy in the room."
 - Critique code instead of people -- be kind to the coder, not to the code

Code Walks

Voraussetzungen (psychologisch)

- Respekt (vor Entwickler, Teamleitung, SC, NKdF)
- Kostenbewusstsein (es werden Kapazitäten gebunden)
- großzügig, geduldig, fehlertolerant, wenig pedantisch
- kein „Jäger“, kein Papst
- keine „offenen Rechnungen“
- neugierig, sehr offen
- „spiel-resistent“
- freundlich, nett, hilfsbereit - aber nicht blöd
- „gelockertes“ Vertrauensprotokoll

Code Walks

Details Durchführung

- Checkliste für Reviewer, nützliche Fragen
 - Mein persönlicher Zustand?
 - Meine Vorbehalte?
 - Ziele von diesem CW?
 - Ängste?
 - Schlechte Erfahrungen?
- Terminverlegung ist keine Schande (auch 5x)

Code Walks

Details Durchführung

- 1. Schritt CW (5 min.): gemeinsamer Vertrag
 - Wozu CW?
 - Eingrenzung (welche Themen, welche nicht, ...)
 - Verständnis für „Geschmäcker“ (Reviewer + Reviewee)
 - Vertraulichkeit (Beichtgeheimnis)
 - Offenheit im Umgang
 - Feedback über CW jederzeit gewünscht (Meta-Position)
 - Eskalation über PL möglich
 - Abbruch jederzeit möglich

Code Walks

Details Durchführung

- Protokoll
 - fixiert Erkenntnisse
 - wird vom Reviewer erstellt (zeitnah)
 - wird mit dem Reviewee abgestimmt (zeitnah)
 - keine „öffentliche“ Ablage
 - Falls bei Punkten keine Einigung
 - Hinterlegung verschiedene Ansichten im Protokoll
 - Faire Darstellung beider Ansichten (muss abgestimmt sein!)

Code Walks

Entwickler-Persönlichkeiten

- offen, kooperativ
„gut, dass ich da mal drüber reden kann“
- verschlossen
„das ist MEIN Source Code!“
- arrogant, rechthaberisch
„DAS MACHT MAN ABER SO!“

- Rückschlüsse auf Fähigkeiten, Qualität, ... dadurch meistens nicht möglich
- **Blinde Flecke ... sieht man nicht!**

Code Walks

Variationen

- „Peer reviews“
 - hört sich gut an - ist aber oft problematisch
 - ungenügende Trennung der Peers
 - Betriebsblindheit
- Zentralisiertes Qualitätsteam
 - schwierig, Vertrauensbasis aufzubauen
 - „zu breite“ Sicht
 - nicht Teil des Teams („wir“ <> „die“)
 - nicht Teil des Produkts

Code Walks

Gern gesehen

- Verständliche, gute Namen (für alles ...)
- Kurzer, übersichtlicher SC
- DRY (Don't repeat yourself)
- Intuitiv nachvollziehbarer SC
- Brauchbare Kommentare - wenn komplizierter SC
- SOC (Separation of concerns)
- TDA (Tell, don't ask)

Code Walks

Nicht gern gesehen

- Cobol-Style in Java-Klassen
- Obfuscated SC
- Schweizer Taschenmesser
- Supercoole Tricks
- „Manager“-Klassen
- „Datenträger“-Klassen (keine Methoden nötig ...)
- „Doing it my way“ (z.B. Logging)
- Mangelhaftes Exception-Handling
- Mißbrauch von Sprachkonzepten
- Feature envy

Code Walks

- Reflexion über Software
- Fragen
 - was macht der SC?
 - was macht der SC **wirklich**?
 - was macht der SC **sonst noch**?
- Langfristige Verhaltensänderung / Verbesserung Q
 - Entwickler weiß, dass er immer wieder CWs macht
 - Anmerkungen bleiben hängen, werden evtl. umgesetzt
 - Code Walks fördern die Auseinandersetzung auf der Meta-Ebene

Werkzeuggestützes

Audits & Metriken

- Messen, zählen, wiegen ...
 - einfach (meistens zu einfach), deswegen sehr beliebt
- Bedürfen **immer** der Interpretation
- Müssen „getailored“ werden
 - nicht alles zählen, nicht alles ist wirklich wichtig
- Vollständiger Regelkreis ist wichtig:
messen **und** regeln
- Erst Wiederholung macht den Meister (verbessert die Metriken)

Audits & Metriken

Beispiele für Findings (Java)

- ECB: Empty catch blocks
 - `void ExceptionIgnorer.ignore()`
- BDCB: Brain-dead catch blocks
- HIA: Hiding inherited attributes
- HN: Hiding names
- CC: Cyclomatic complexity zu hoch (> 50)
- ILTD: Indentation level too deep (> 7)
- NOO: Number of operators too high (> 30)

Audits & Metriken

Werkzeuge

- Desktop / Server
 - Borland Together Architect
 - pmd & cpd (gleiche Quelle)

Werkzeuggestützes

CPD (Copy&Paste Detector)

- „Ist das wirklich alles neu“?
 - Wenn die bevorzugte Metrik „Code lines / €“ ... gute Frage!
- Wie groß ist das Wartungs-Monster, das wir vor uns haben?
 - Wenn SC kopiert wurde und nicht tot ist, wo muss bei einem Bug Fix geändert werden? Erwischt man alle relevanten Stellen?
- Messinstrument für DRY
- Erschreckende Zahlen
- Gute Erklärungen (Wartung)
- Technik (Robin-Karp-Algorithmus)

Katastrophenverhinderung

- Regelmäßig Log-Files lesen
- Exceptions auswerten (Java)
- Spürnase
 - Source Code, der AppServer beim Start behindert!
 - Fehler im Memory Management
 - „SQL-Gefrimmel“
 - Zugriffe auf Backends
 - Caching desperately needed!

Information Management

- Welche Probleme kennen wir?
- Was wurde wie von wem wann behoben?
 - Warum tritt es wieder auf?
- Versionsverwaltung
- Bug Tracking (mit „wie wurde das Problem behoben“)
- Was äußert sich wie? Internet ist unbedingte Voraussetzung
- Dokumentieren, dokumentieren, dokumentieren ...
und suchbar machen / halten!
 - Ausschließlich elektronisch - alle andere findet man in Hektik nicht
- Bugs keep coming back - sometimes in weird dress
- There is always another bug - but not all bugs are really dangerous

Literatur

- **Code Clean (Robert C. Martin, „Uncle Bob“)**
- Software Inspection (Tom Gilb, Dorothy Graham)
- The Psychology of Computer Programming (Gerald M. Weinberg)
- Are Your Lights On? (Donald C. Gause, Gerald M. Weinberg)

Links

- http://en.wikipedia.org/wiki/Separation_of_concerns
- <http://c2.com/cgi/wiki?TellDontAsk>
- <http://c2.com/cgi/wiki?LawOfDemeter>
- <http://www.industriallogic.com/papers/smellstorefactorings.pdf>
- <http://www.codinghorror.com/blog/archives/000584.html>
- <http://www.artima.com/designtechniques/compoinh.html>
- <http://de.wikipedia.org/wiki/Rabin-Karp-Algorithmus>
- http://en.wikipedia.org/wiki/Cyclomatic_complexity