



J-Spring

16 april 2008 Spant! - Bussum



Hoe overleef ik een code review?

Mark van Holsteijn

Competitive Software Engineering

www.cswe.nl

Medeoprichter en lid van het Know IT kennisnetwerk www.know-it.nl



Agenda

- Redenen voor een code review
- Systeemeigenschappen
 - Betrouwbaarheid / Onderhoudbaarheid
- Indicatoren
 - Failure rate / Defect Density
- Code Reviews
- Defect Removal Technieken
- Extreme Inspecties
- Oproep



Redenen voor een code review

- Onafhankelijke In-process kwaliteitstoets – Fabrieksinspectie
 - Gaat het eindproduct voldoen aan onze kwaliteitseisen.
 - Is het een slim idee?
 - Wordt het goed uitgevoerd?
 - Zijn er zaken die beter kunnen?
- Project met faalsymptomen – Krijg ik mijn auto nog?
 - Slippende deadlines
 - Kostenoverschrijding
 - Sterk oplopend aantal bevindingen
 - “redesigns” door de leverancier
 - Ingraven van leverancier
 - “Dat is een functionele eis van de klant”
 - “Dat is nergens formeel vereist”
- Van Ontwikkeling naar Beheer – De Custom Car in eigen onderhoud.
 - Het projectteam wordt ontmantelt
 - Expertise vertrekt
 - Onderhoud en beheer gaat door anderen gedaan worden
 - Eigen beheer, derde partij
 - Onderhoudbaarheid
- Stabiliteitsproblemen na oplevering – De maandagmorgenauto
 - Gebruikers klagen over de ontbrekende “kwaliteit”
 - Performance
 - Stabiliteit
 - Gebruiksgemak



Centrale vragen

- De vraag die de klant beantwoordt wil zien als resultaat van de review.
- Bijvoorbeeld:
 - Is het systeem van voldoende kwaliteit zodat wij dit in eigen beheer kunnen nemen?
 - Voldoet de implementatie aan de standaarden en richtlijnen voor Enterprise Java applicaties?
 - Wat is de oorzaak van de voortdurende kostenoverschrijding door het project?
 - Kan dit systeem de verwachte capaciteitsgroei aan?
 - ...
- Vaak onuitgesproken
 - Kunnen we de leverancier aansprakelijk stellen?
- Vragen meestal gerelateerd aan “The Big Five”



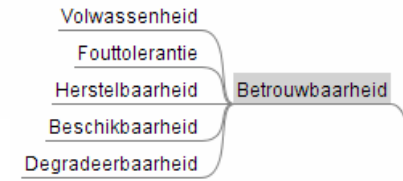
De Trofeeën – the big five eu.. six



- **Betrouwbaarheid**
 - De mate van vertrouwen in het systeem.
- **Onderhoudbaarheid**
 - De mate van inspanning die nodig is om wijzigingen aan te brengen.
- **Efficiëntie**
 - De ratio tussen de gebruikte resources en de geleverde “performance”.
- **Bruikbaarheid**
 - De mate waarin het systeem bruikbaar wordt geacht.
- **Functionaliteit**
 - De mate waarin het systeem de benodigde functies levert.
- **Portabiliteit**
 - De mate van inspanning die nodig is om een systeem over te brengen van de ene naar de andere omgeving.



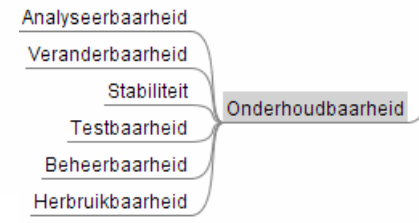
Betrouwbaarheid



- Volwassenheid
 - Frequentie van fouten.
- Fouttolerantie
 - De mate waarin het systeem bestand is tegen het optreden van fouten.
- Herstelbaarheid
 - De mate waarin het systeem in staat is om te herstellen van fouten.
- Beschikbaarheid
 - De percentage van de tijd waarin de functies van het systeem beschikbaar zijn.
- Degradeerbaarheid
 - De inspanning die nodig is om de essentiële functies te herstellen na een breakdown.



Onderhoudbaarheid



- Analyseerbaarheid
 - De inspanning die nodig is voor het analyseren van een fout
- Veranderbaarheid
 - De mate van inspanning die gepaard gaat met het wijzigen van het systeem of het verwijderen van fouten.
- Stabiliteit
 - Het aantal geïntroduceerde fouten per release.
- Testbaarheid
 - De inspanning die nodig is het systeem te valideren.
- Beheerbaarheid
 - De inspanning die nodig is om het systeem (weer) operationeel te krijgen.
- Herbruikbaarheid
 - De mate waarin delen van het systeem (gedeeltelijk) gebruikt kunnen worden in een ander systeem.



Betrouwbaarheid en Onderhoudbaarheid

- Betrouwbaarheid
 - De effecten van de aanwezigheid van fouten.
- Onderhoudbaarheid
 - Opsporen, verwijderen, herstellen en voorkomen van fouten.
- Twee belangrijke indicatoren
 - Failure Rate / MTTF – Mean Time To Failure .
 - Defect Density - #aantal fouten per eenheid software.
- Definities
 - Defect
 - Een potentiële fout in het systeem.
 - Failure
 - Het optreden van een fout.



Failure Rate



- Een indicatie voor Betrouwbaarheid (volwassenheid)
- Frequentie waarmee een component faalt.
- $MTTF = 1 / \text{Failure Rate}$
- Dat is een leuke eenheid voor Betrouwbaarheid
- Maar...
 - Wat betitelen we als een Failure?
 - Crash? error message? Een issue?



Defect Density

- Defect Density is een mooie indicator
- met een hele nare eenheid
 - #aantal fouten per eenheid code.
 - Fouten zitten “verstopt” in de code.
 - manifestatie pas bij gebruik.
 - De eenheid is niet gestandaardiseerd (KLoC, KLoS, Functiepunt, taalafhankelijk).
 - Geen courante statistische gegevens voor “hedendaagschse” projecten.
 - Geen standaard meetapparatuur.



Function Point Defect Rates	
CMM Level	Rate
1	0.75
2	0.44
3	0.27
4	0.14
5	0.05

source: Software Assessments, Benchmarks and Best Practices, 2001, Caper Jones

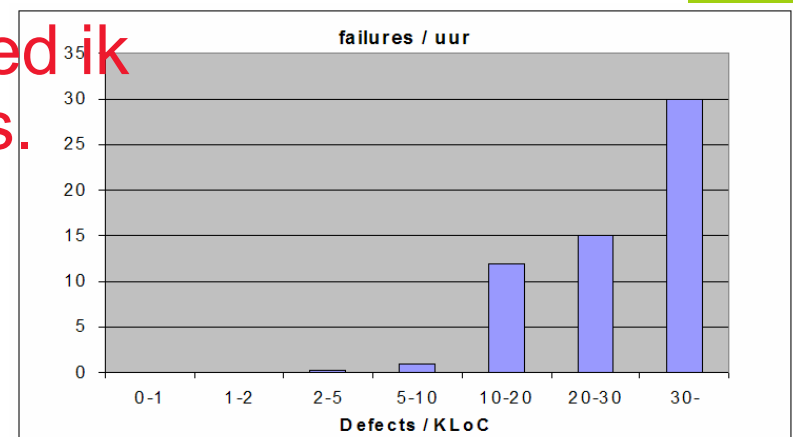
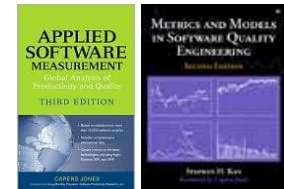


Relatie defect density en failure rate

- Is er een relatie?
- Volgens S. Kan is er geen wiskundige relatie.
- Volgens C. Jones is er een empirische relatie ...
 - van een mainframe systeem software lang, lang geleden
- Maar... persoonlijk vermoed ik dat die relatie er ook wel is.
- Combinatie
 - Geeft een goede indruk

defects/KLoC	MTTF
0-1	Onbepaald
1-2	24-160 uur
2-5	4-24 uur
5-10	1-4 uur
10-20	5-60 min
20-30	4-15 min
30-	minder 2 min

bron: Applied Software Measurement: Assuring Productivity and Quality, Jones, C. ISBN 0-07-032813-7





Code Review - de sporen

- De code
 - Versie management systeem
 - Meeste gewijzigde sources
 - High Risk Modules 10% -> 80%
- Issue management systeem
 - Ontdekte defects
- Applicatie (server) log files
 - "Geaccepteerde" Failures
- Runtime monitoring systemen
 - Unreported Failures in productie
- Unit test & coverage reports
 - Defect removal efficiency
- Build systeem
 - Continuous integration
 - OTAP omgeving?





Code review – de tools

- **Het blote oog**
 - Langzaam
 - Beperkte focus
 - Goed in het vinden van fouten in de logica
 - Vereist kennis en ervaring
- **Statische code analyse tools**
 - Snel
 - Grondig
 - Goed in het vinden van “triviale” fouten
- **Ervaring**
 - Visuele en automatische inspectie leverde gelijke Defect Densities op





ADHD Code Review – aan het werk!

- Een code review
 - Op basis van de representatieve steekproef.
 - Van twee systemen A & B.
 - A – `serveResource()`
 - B – `getPackAsStream()`
 - Spreek je vertrouwen uit op basis van de code in de steekproef:

• Geen vertrouwen	20	defects
• Wantrouwig	10-20	defects
• Goed van vertrouwen	2-10	defects
• Vol vertrouwen	0-2	defects
 - Neem 1 minuut per pagina
 - Markeer “defects” en tel ze.
 - Geef aan op basis waarvan je oordeelt.
 - Aantal defects



ADHD Code review – de uitslag op 17 april 2008

- | | | |
|-----------------------|------|------|
| – Geen vertrouwen | A:6 | B:45 |
| – Wantrouwig | A:40 | B:30 |
| – Goed van vertrouwen | A:43 | B:2 |
| – Vol vertrouwen | A:2 | B:1 |



Code review – Tomcat 4.1.x



- Goede reputatie
- Een steekproefgewijze review levert met moeite defects op
- In officiële bug tracking database Module Catalina, had 264 bugs tussen okt 2001 en sept 2006 (0.15 fout per dag, geschat MTTF 210 uur)
- Grove schatting :
defect density < 1 defect/KLoC

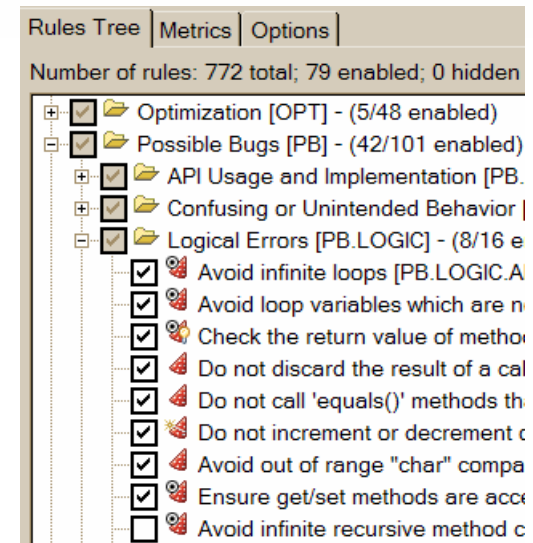
defects/KLoC	MTTF
0-1	Onbepaald
1-2	24-160 uur
2-5	4-24 uur
5-10	1-4 uur
10-20	5-60 min
20-30	4-15 min
30-	minder 2 min

bron: Applied Software Measurement:
Assuring Productivity and Quality, Jones, C.
ISBN 0-07-032813-7



Code Review – Statische Analyse

- Met Parasoft JTest
- Bevat bijna 800 regels
- In verschillende rulesets, waaronder
 - Must have - 80 actieve regels
 - Should have – 315 actieve regels
 - Nice to have – 453 actieve regels
 - “This is where formatting kicks in”
 - Maar ook Code Smell, Effective Java, Core J2ee Patterns
- Maakt het bruikbaar in bestaande projecten





Code review – resultaten statische analyse

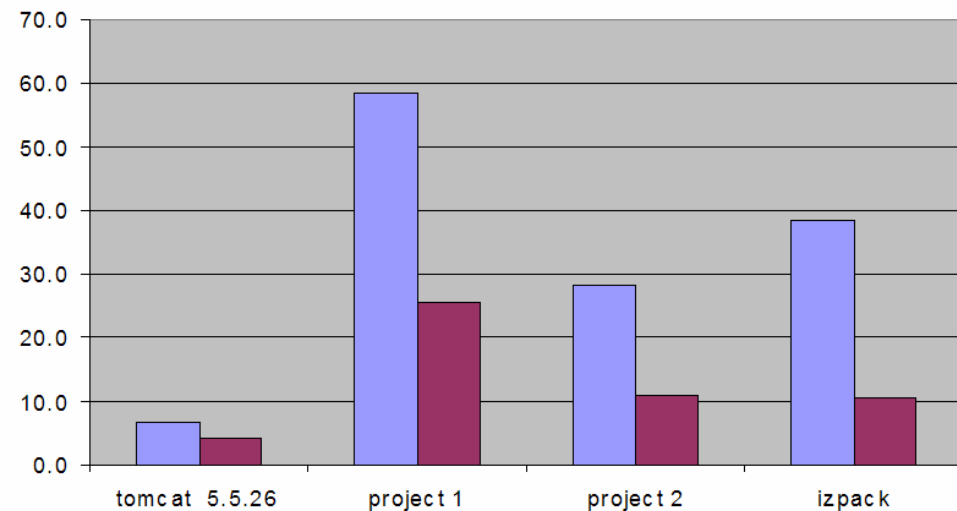


- Op 4 projecten
 - Tomcat 5.5.26
 - IzPack 3.11.0
 - Twee “commerciele” projecten

project	distinct	total	MTTF	Defects/KLoC
tomcat 5.5.26	69	1370		4
project 1	97	3547	55 min	26
project 2	88	3817	2 uur	11
izpack	124	2642		11

defects/KLoC	MTTF
0-1	Onbepaald
1-2	24-160 uur
2-5	4-24 uur
5-10	1-4 uur
10-20	5-60 min
20-30	4-15 min
30-	minder 2 min

bron: Applied Software Measurement:
Assuring Productivity and Quality, Jones, C.
ISBN 0-07-032813-7





Defect Removal Technieken

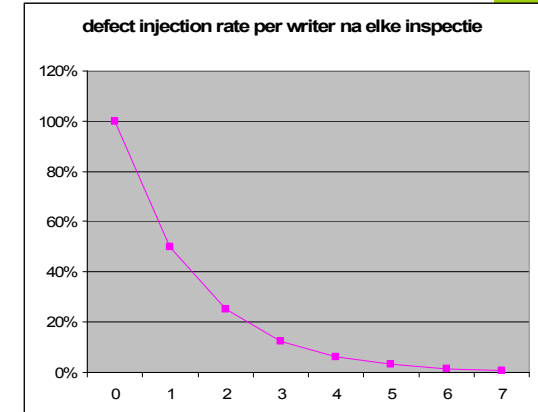
- Hoe verminder je de defect density?
 - Defect Removal Efficiency = $\text{Aantal gevonden defects} / \text{Aantal aanwezige defecten} * 100\%$
- Beste technieken
 - High volume Beta test
 - Formele Inspecties
- >95% DRE
 - Combinatie van 8-10 technieken

Defect Removal Efficiency		
Activity	min	max
Informal design reviews	25%	40%
Formal design inspections	45%	65%
Informal code reviews	20%	35%
Formal code inspections	45%	70%
Unit test	15%	50%
New function test	20%	35%
Regression test	15%	30%
Integration test	25%	40%
Performance test	20%	40%
System test	25%	55%
Acceptance test (1 client)	25%	35%
Low-volume Beta test (< 10 clients)	25%	40%
High-volume Beta test (>1000 clients)	60%	85%



Inspecties

- Effectief
 - Defect Removal Efficiency tussen 65-85%.
- Preventief
 - Feedback van inspecties vermindert het aantal geinserte defects.
- Tijd en kostenbesparend
 - Hoe eerder in het proces, hoe beter
 - Schatting 1:9.3
 - Elke cycle, reduceert de defect injection rate met 50%
 - Goedkoper dan testen
- Maar..
 - Formele inspecties zijn uit
 - Vereist inzet, discipline en doorzettingsvermogen
 - Kan dat nou niet beter....

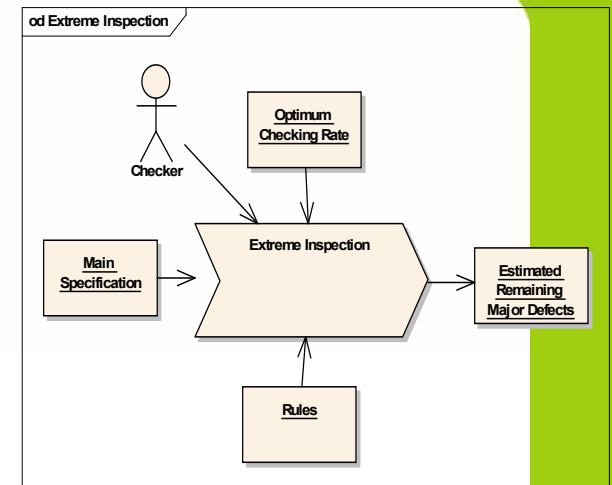


Bron: Competitive Engineering, Gilb. T.

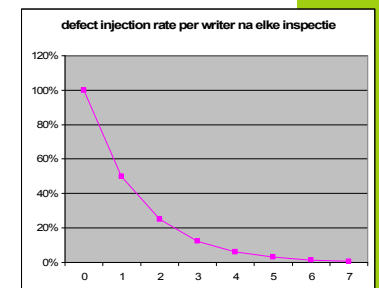


Extreme Inspecties

- Extreme inspecties
 - Steeksproefgewijze inspectie
 - Op representatieve steekproef
- Definities
 - Rule
 - Een standaard voor de productie van geschreven proces output. Een rule is een objectieve maat voor kwaliteit (“defect-freeness”)
 - Defect
 - A schending van een formele, geschreven regel.
 - Major Defect
 - een Defect waarvan de kosten om deze te verwijderen in een later fase een orde grootte hoger zullen zijn.
 - Optimum Checking rate
 - De gemiddelde snelheid waarmee je de meeste major defects vindt. Effectieve inspecties gaan heel langzaam (1 uur per pagina).
 - Remaining major defects
 - geschat Aantal major defects per volume. Berekent mbv totaal gevonden defects en % effectivens.



Bron: Competitive Engineering, Gilb. T.





Extreme Inspection Process

P1: Identificeer Checkers.

P2: Selecteer Rules.

P3: Selecteer een steekproef van 1 pagina (300 woorden).

P4: Instrueer checkers.

P5: Check het document op major defects gedurende 10-30 min.

P6: Rapporteer resultaten.

P7: Analyseer resultaat.

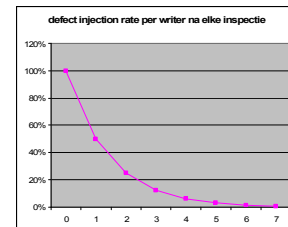
P8: Besluit actie (Exit/Rework/Rewrite).

P9: Suggereer oorzaak.



Hoe overleef je een code review?

- Je overleeft het meestal wel
 - De kosten van totale rework zijn vaak te hoog
 - Maar je reputatie loopt wel een deuk op
 - Je kunt ze wel voorkomen...
- Voer ze zelf extreme inspecties uit.
 - Stel rules op.
 - Wekelijks.
 - Doe het steekproefsgewijs.
- Gebruik statische analyse tools.
 - Als entry criteria
 - “don’t waste the checkers time”
- Houd je rules up-to-date bij!
 - Zo leren je van elkaar





Oproep

- Voor een NLJUG SIG
 - Doel: verbeteren van de kwaliteit van JEE systemen
- Verzamelen van indicatoren
 - Failure Rate.
 - Defect Density.
 - Load en responsetijden.
 - Op basis van een standaard tool & rule set.
- En gebruiken als publieke benchmark

.nl.
jug



Vragen?



Bronnen

- Kwaliteit van software producten, van Zeist et.al., ISBN 90-267-2430-6
- Applied Software Measurement, Assuring Productivity and Quality, Jones, C., ISBN 0-07-032813-7
- Metrics and Models in Software Quality Engineering Kan, S. ISBN-0-201-72915-6
- www.flickr.com - ksr8s
- Conflict and litigation Between software clients and developers, Version 10 – April 13, 2001, Caper Jones
- Competitive Engineering, Gilb T., ISBN 0-7506-6507-6

