

# **Reduceng Erors & Imprvoing Quilaty Throuhg Reviewws**

**David A. Cook, Ph.D.  
Stephen F. Austin State University  
Nacogdoches, TX  
cookda@sfasu.edu**



## Please note...

- ◆ Stephen F. Austin State University is NOT in Austin – it is in Texas' oldest city, Nacogdoches – in the northeast corner of the state.



# Enough reviews for you????

- ◆ Review
- ◆ Management Review
- ◆ Technical Review
- ◆ Inspection
- ◆ Peer Review
- ◆ Walk-Through
- ◆ Audit
- ◆ “Skim” Review
- ◆ Disciplined Document Review
- ◆ Desk Check
- ◆ Personal Document Review
- ◆ Personal Code Review
- ◆ Code Review
- ◆ Design Review
- ◆ Formal Qualification Review
- ◆ Requirements Review
- ◆ Test Readiness Review
- ◆ Functional Configuration Audit
- ◆ Physical Configuration Audit
- ◆ Etc.

# What reviews give you

## ◆ Direct Benefits

- Improved code quality
- Fewer Defects
- Improved communication about code content
- Education of new/junior developers

## ◆ Indirect benefits

- Shorter and more effective testing
- Less maintenance
- Improved customer satisfaction
- More maintainable code

# Quality is the goal

- ◆ Quality is NOT free
- ◆ “Cost” of Quality includes
  - Review costs
  - Tests cost
  - All defect prevention costs (training)
- ◆ Savings from Quality include
  - Decreased costs of product failure
    - ☞ Help Desk
    - ☞ Customer defect repair
  - Shorter test cost
  - Shorter development time

# Return on Investment

- ◆ Boeing – 33:1 savings from reviews
- ◆ HP – 10:1 saving \$21 million a year
- ◆ Space Shuttle – \$1 if error found in inspection, \$13 during test, \$92 after delivery
- ◆ IBM – each hour of inspection saved 20 hours of testing, and 82 hours of rework (for each error that would have made it to delivery)
- ◆ AT&T – 22:1 savings if errors found early, reduced cost of finding errors by 10:1

# More savings

- ◆ Maintenance costs are typically 50% less (values of 90% have been reported)
- ◆ Litton Data Systems – 3% increase in costs due to inspections, number of errors found during system and integration testing dropped 30%



# Reviews vs. Testing

- ◆ Testing is a discrete activity, reviews should be continuous
- ◆ Each testing stage only removes about 35% of errors present
- ◆ GOOD Reviews and Inspections typically remove 50%
- ◆ Testing can give poor code coverage, and will always give poor coverage of documentation

# What can be reviewed?

- ◆ ?? (fill in the blanks)

# What can be reviewed?

- ◆ ?? (fill in the blanks)

# What can't be reviewed?

# Management Involvement is limited

- ◆ Measurement dysfunction – when managers use review data to evaluate. This produces inconsistent results and bizarre behavior.
- ◆ Leads to inaccurate data, invalid reviews, and the use of reviews to grind “personal” issues.
- ◆ Management involvement should be limited to “edited” and “sanitized” summarization of the final results

# Management Commitment

- ◆ Provide resources (time and space)
- ◆ Setting policies and goals
- ◆ Maintaining reviews even when under a time crunch
- ◆ Require schedules to include review time
- ◆ Providing training
- ◆ Not using results to evaluate
- ◆ Holding people accountable for participation and contributions

# Management Commitment (cont.)

- ◆ Rewarding early adopters
- ◆ Running interference with challengers
- ◆ Respecting review team's appraisal
- ◆ Asking for status reports, showing how the program is working, what it costs, and the benefits (and deficits)

# Consequences of Misapplication of Inspection Data

- ◆ Developers might not submit products
- ◆ Developers might not agree to review peer's work
- ◆ Defects brought up after the review, not during
- ◆ Pre-reviews to prepare
- ◆ Too much debate on what is a defect
- ◆ Review of small products – wasting time

# Ego-less programming

- ◆ Need to stress the benefits of reviews to all levels of management
  - Less time in rework
  - Increased productivity
  - Education and learning
  - Better able to meet deadlines
  - Better risk management
- Not “extra time”, but reallocation of effort



# Reviews are NOT milestones

- ◆ Milestones are a “time”
- ◆ Reviews are a “process”
  
- ◆ Milestones occur AFTER a review, and involve a go/no-go decision

# Principles for a review

1. Check egos at the door
2. Keep the review team small
3. Find problems, don't solve them
4. Limit review time
5. Require preparation

# Peer Review Spectrum

- ◆ Inspection
- ◆ Team Review
- ◆ Walkthrough
- ◆ Pair Programming
- ◆ Peer Deskcheck
- ◆ Ad Hoc

Most formal



Least formal

# Typical Activities

<u>REVIEW TYPE</u>	<u>Planning</u>	<u>Preparation</u>	<u>Meeting</u>	<u>Corrections</u>	<u>Verification</u>
Inspection	Yes	Yes	Yes	Yes	Yes
Team Review	Yes	Yes	Yes	Yes	No
Walkthrough	Yes	No	Yes	Yes	No
Pair Programming	Yes	No	N/A	Yes	Yes
Peer Deskcheck	No	Yes	Maybe	Yes	No
Ad Hoc	No	No	Yes	Yes	No
Individual	No	No	No	Yes	Always

# Which type of review for you?

- ◆ Depends upon
  - Criticality of application
  - Skill of individual reviewer
  - Needs of the organization
  - Maturity of the organization

<u>Review Objective</u>	<u>Inspection</u>	<u>Team Review</u>	<u>Walkthrough</u>	<u>Pair Programming</u>	<u>Peer Deskcheck</u>	<u>Passaround</u>
Find defects	X	X	X	X	X	X
Conformance to specs	X	X			X	X
Verify complete and correct	X		X			
Assess understandability and maintainability	X	X		X		X
Demonstrate quality	X					
Collect data for improvement	X	X				

<u>Review Objective</u>	<u>Inspection</u>	<u>Team Review</u>	<u>Walkthrough</u>	<u>Pair Programming</u>	<u>Peer Deskcheck</u>	<u>Passaround</u>
Measure quality	X					
Education of team members		X	X	X		X
Reach consensus on approach		X	X	X		X?
Ensure changes of fixes made correctly		X	X		X	
Explore alternative approaches			X	X		
Simulate execution of a program			X			
Minimize review cost					X	

# Common Misconception

- ◆ Peer reviews are a luxury
- ◆ TRUTH: Peer reviews, when intelligently applied, shorten development and testing. In fact, some testing steps may be skipped (or will be so small they are almost a formality)



# How fast to review

- ◆ Studies show that 200 LOC/hour is close to optimal
  - More, and you miss errors
  - Less, and you get diminishing returns
- ◆ With 200 LOC/hour, defects will be reduced to around 20 per 1000 LOC

# Rules for reviews

- ◆ Schedule no less than a week in advance, to give participants time to prepare
- ◆ No more than one inspection per day for any one participant (including the moderator)
- ◆ No “lunch” inspections
- ◆ No “3 PM Friday” inspections
- ◆ Coffee and donuts are a necessity
- ◆ Have a time limit – and **STICK TO IT!!** End when the time is up

## Before the review – perform “Skim Review”

- ◆ Brief one-time reading (similar to reading a novel)
- ◆ Guidelines for “skim review”
  - Don’t depend on ad-hoc, skim reviews to find all (or even most of) the defects
  - Use them to overview document
  - Use them to check that entrance criteria for review have been met (e.g., not more than 3 major defects found in 10 minutes)

# During any structured review

- ◆ Have recorder!!!
- ◆ Have a recorder who knows how to record!!!
- ◆ Use semi-formal & formal documents to record errors (location, side effects, any other specifics)
- ◆ Use the same documentation to provide accountability and reduce need for follow-up (although spot-checking of follow-up is **HIGHLY** recommended)

# Seven “Truths” about Reviews \*

- ◆ Peer reviews can take many forms
- ◆ Inspections are a software industry best practice
- ◆ There is no one true inspection method
- ◆ Peer reviews complement testing
- ◆ Peer reviews are both technical and social activities
- ◆ Managers can make or break a review program
- ◆ A peer review program doesn't run itself

\* This slide and the next from Karl Wieggers' Book  
Reviews Cook 2010

# Comparison of methods

<u>Element</u>	Fagan Method	Gilb/Graham Method
Process Steps	<ul style="list-style-type: none"> <li>•Planning</li> <li>•Overview</li> <li>•Preparation</li> <li>•Inspection Meeting</li> <li>•Rework</li> <li>•Follow-up</li> <li>•Causal Analysis</li> </ul>	<ul style="list-style-type: none"> <li>•Planning</li> <li>•Kickoff Meeting</li> <li>•Individual Checking</li> <li>•Logging Meeting</li> <li>•Editing</li> <li>•Follow-up</li> <li>•Process Brainstorming Meeting</li> </ul>
Roles	<ul style="list-style-type: none"> <li>•Author</li> <li>•Moderator</li> <li>•Reader</li> <li>•Recorder</li> <li>•Inspector</li> </ul>	<ul style="list-style-type: none"> <li>•Author</li> <li>•Inspection Leader</li> <li>•Scribe</li> <li>•Checker</li> </ul>
Defect-Detection Techniques	<ul style="list-style-type: none"> <li>•Defect Checklists</li> </ul>	<ul style="list-style-type: none"> <li>•Rule Sets</li> <li>•Checklists</li> </ul>
Emphasis	<ul style="list-style-type: none"> <li>•Removing Defects</li> </ul>	<ul style="list-style-type: none"> <li>•Document Quality</li> <li>•Measurement</li> <li>•Process Improvement</li> </ul>

# Remember - to make reviews work...

- ◆ No discipline or rigor is normally associated with informal reviews, so effective leaders and checklists must be used to achieve useful results
- ◆ To make reviews useful, members of the the review team must be objective
  - Make sure that some members of the review team have different backgrounds
  - Make sure that some members of the review team have no direct involvement with the product being reviewed
  - Political agendas need to be left at the door
- ◆ Make sure that reviewers understand the requirements
  - If necessary, present requirements in a number of different ways
  - Simply reading the requirements documents is probably insufficient
  - The brain can only keep so many requirements “active”

# Questions???



# References

- ◆ *Peer Reviews in Software: A Practical Guide* by Karl E. Wieggers (Addison-Wesley, 2002).
- ◆ *Best Kept Secrets of Peer Code Review* by Jason Cohen (2006, free, from [smartbearsoftware.com](http://smartbearsoftware.com))
- ◆ *Peer Reviews*, a STSC Workshop presented at Orlando Naval Air Warfare Center, Training Systems Division, (May 2008) by Dr. David A. Cook
- ◆ *Managing the Software Process* by Watts Humphrey (Addison-Wesley, 1989)
- ◆ Fagan, M. "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, Vol. 15, No. 3 (1976), pp. 182-211.