Electrical & Computer
ENGINEERING

18-642:

Peer Reviews

2/5/2018

The competent programmer is fully aware
of the strictly limited size of his own skull;
therefore he approaches the programming
task in full humility, and among other
things he avoids clever tricks like the
plague.

--*Edsger Dijkstra*

Edge
Case
Research

Carnegie
Mellon
University

1

Peer Reviews

■ **Anti-Patterns:**
- **No peer reviews**
- **Reviews too informal/too fast**
- **Reviews find <50% of all bugs**

■ **Fresh eyes find defects**
- Code and other document benefit from a second (and third) set of eyes
- Peer reviews find more bugs/$ than testing
  – And, they find them earlier when bugs are cheaper to fix
- Everything written down can benefit from a review

https://en.wikipedia.org/wiki/Peer_review#/media/File:ScientificReview.jpg

Most Effective Quality Practices

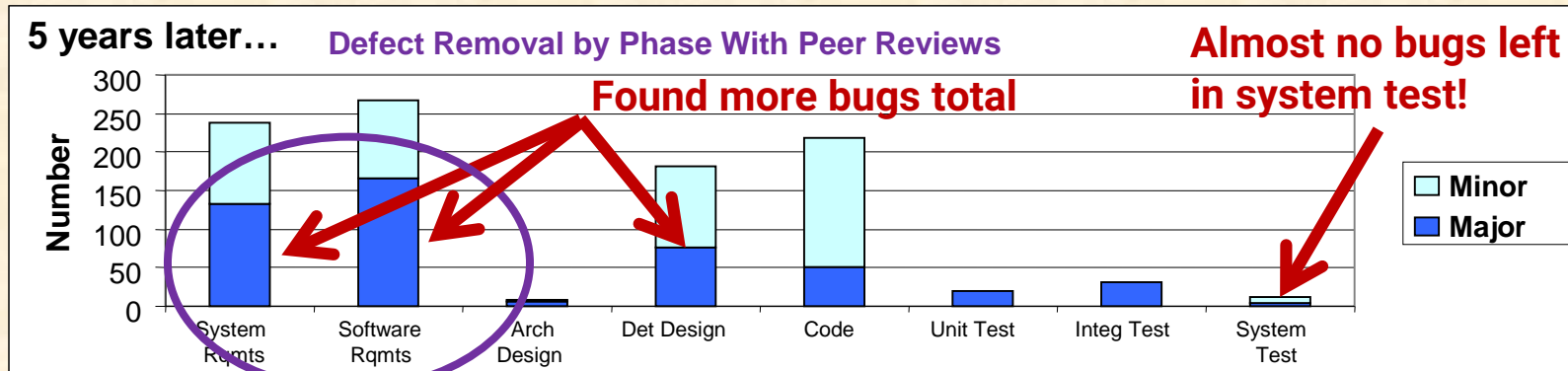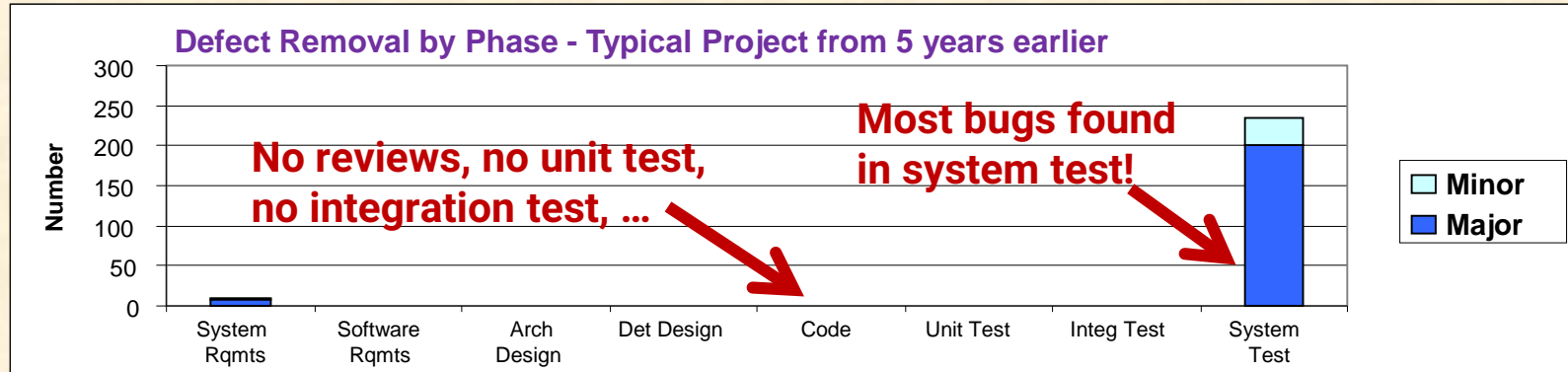Ebert & Jones, "Embedded Software: Facts, Figures, and Future," IEEE Computer, April 2009, pp. 42-52

Ranked by defect removal effectiveness in percent defects detectable at that stage that are removed.

"*" means exceptionally productive technique (more than 750+ function points/month)

- * 87% static code analysis ("lint" tools, compiler warnings)
- 85% design inspection
- 85% code inspection
- 82% Quality Function Deployment (requirements analysis)
- 80% test plan inspection
- 78% test script inspection
- * 77% document review (other documents)
- 75% pair programming (informal on-the-fly review)
- 70% bug repair inspection
- * 65% usability testing
- 50% subroutine testing (unit test)
- * 45% SQA (Software Quality Assurance) review
- * 40% acceptance testing

Peer Reviews Are Effective + Efficient

Defect Removal by Phase - Typical Project from 5 years earlier

No reviews, no unit test, no integration test, …

Most bugs found in system test!

Chart: Number (0–300) by phase: System Rqmts, Software Rqmts, Arch Design, Det Design, Code, Unit Test, Integ Test, System Test
- Minor
- Major

5 years later… **Defect Removal by Phase With Peer Reviews**

Almost no bugs left in system test!

Found more bugs total

Chart: Number (0–300) by phase: System Rqmts, Software Rqmts, Arch Design, Det Design, Code, Unit Test, Integ Test, System Test
- Minor
- Major

Found many bugs up front, where fixes are cheaper

[Source: Roger G., Aug. 2005]

Gold Standard: Fagan Style Inspections

■ **Methodical, in-person review meetings**
- Pre-meeting familiarity with project
- Producer explains item then leaves
- Moderator keeps things moving
- Reader (not author) summarizes as you go
- Reviewers go over every line, using <u>checklists</u> (perspective-based)
- Recorder takes written notes
- Result: written list of defects. The Producer fixes code off-line
- Re-inspection if the defect rate was too high

■ **Methodical reviews are the most cost effective**
- Important to measure bug discovery rate to ensure review quality

5

Rules for Successful Peer Reviews

- **Inspect the item, not the author**
  - Don't attack the author.
- **Don't get defensive**
  - Nobody writes perfect code. Get over it.
- **Find but don't fix problems**
  - Don't try to fix them; just identify them.
- **Limit meetings to two hours**
  - People are less productive after that point.
- **Keep a reasonable pace**
  - About 150 lines of code (or equivalent) per hour. Too fast and too slow are both bad.
- **Avoid "religious" debates on style**
  - Enforce conformance to your style guide. No debates on whether style guide is correct.
- **Inspect, early, often, and as formally as you can**
  - Keep records to document value (might take a while to mature).

严禁打闹蹦跳
No Playing& Jumping

严禁拍打
No Slapping

Example Light-Weight Review Report

Peer Review Template for Project X		
Date:	4/17/2011	
Artifact:	Xyzzy.cpp Functions: Foo(), Bar(), Baz()	
Reviewers:	Stella K., Joe B., Sam Q., Trish R.	
Size:	357	SLOC
Time Spent:	112	Minutes
# Issues:	3	
Outcome:	Re-Review of Bug Fixes Required	
Issue#	**Issue Description**	**Status**
1	Issue 1…..	Fixed
2	Issue 2….	Bugzilla
3	Issue 3….	Bugzilla
4	Issue 4….	Not a Bug
5		
6		
7		
8		
Status Key:	Fixed (trivial fix by author; no need to enter in defect list)	
	Bugzilla (entered into project defect system)	
	Not a Bug (false alarm)	

issues found is the most important item!

Free form text issue description

Just enter "fixed" if fixed within 24 hours

Perspective-Based Peer Reviews

- ■ **Perspective-based Peer Reviews are 35% more effective**
  [https://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.78.pdf]

- ■ **Mechanics of a Perspective-based review**
  - Divide a peer review checklist into three sections
  - Assign each participant a different section of the checklist
    - OK to notice other things, but primary responsibility is that section
    - Multiple sets of eyes + perspective breadth

- ■ **Example perspectives for a review:**
  - Control flow issues
  - Data handling issues
  - Style issues

Peer Review Checklist Template

Peer Review Checklist: Embedded C Code

Before Review:

| 0 | _____ | Code compiles clean with extensive warning checks (e.g. MISRA C rules) |

Reviewer #1:

1	_____	Comm...
2	_____	Style c...
3	_____	Proper...
4	_____	No orp...
5	_____	Condit...
6	_____	Parent...
7	_____	All swi...

Reviewer #2:

8	_____	Single point...
9	_____	Loop entry a...
10	_____	Conditionals...
11	_____	All functions...
12	_____	Use const an...
13	_____	Avoid use of...
14	_____	Strong typin...
15	_____	All variables...

Reviewer #3:

| 16 | _____ | Minimum scope for all functions and variables; essentially no globals |
17	_____	Concurrency (locking, volatile keyword, minimize blocking time)
18	_____	Input parameter checking (style, completeness)
19	_____	Error handling for function returns
20	_____	Handle null pointers, division by zero, null strings, boundary conditions
21	_____	Floating point issues (equality, NaN, INF, roundoff); use of fixed point
22	_____	Buffer overflow safety (bound checking, avoid unsafe string operations)

All Reviewers

| 23 | _____ | Does the code match the detailed design (correct functionality)? |
| 24 | _____ | Is the code as simple, obvious, and easy to review as possible? |

For TWO Reviewers assign items: Reviewer#1: 1-11; 23-24   Reviewer#2: 12-24

■ **Customize as needed**

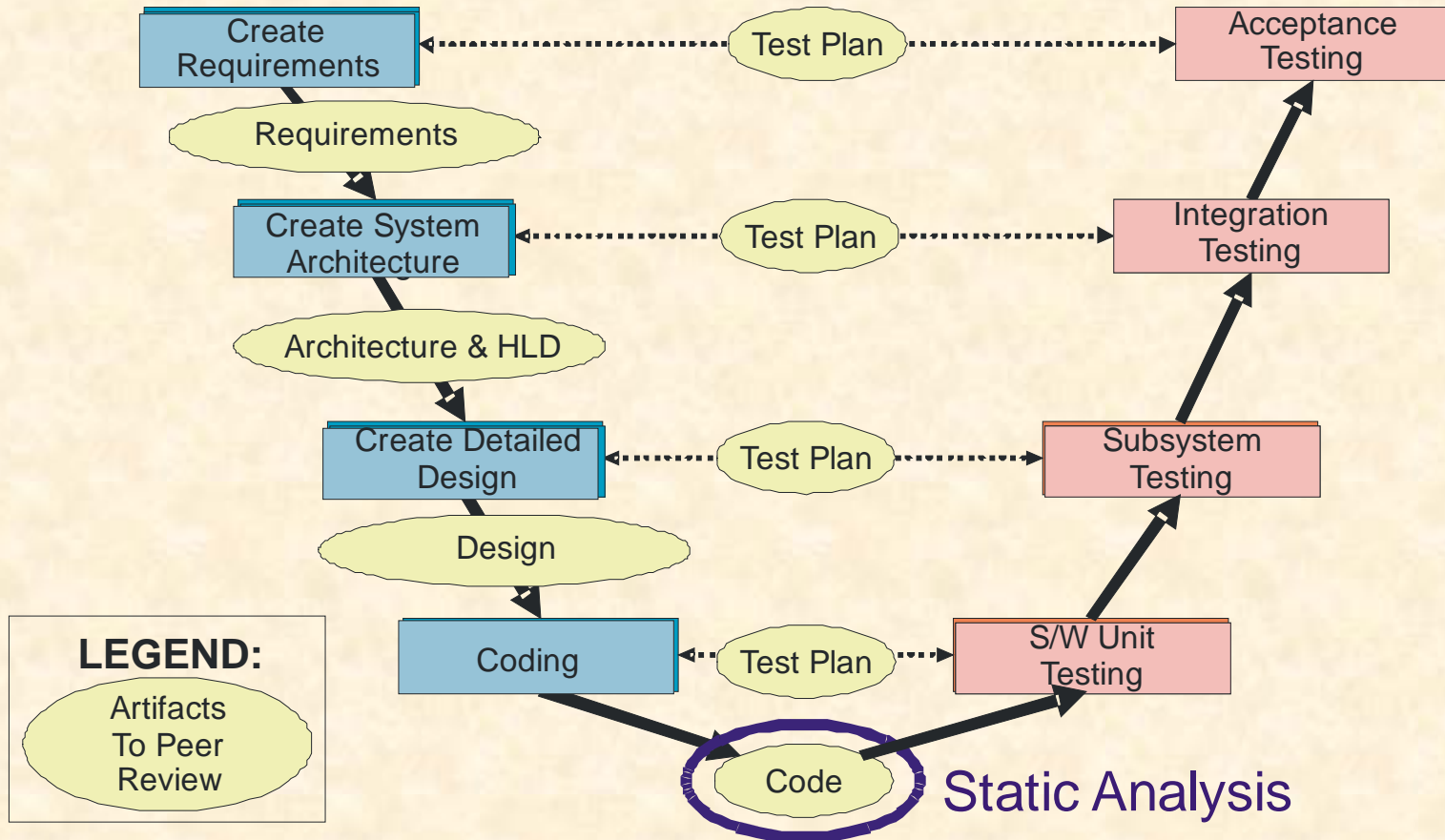# Before The Spreadsheet (Ineffective Reviews)

Spring 2010 18-649 Student Hours
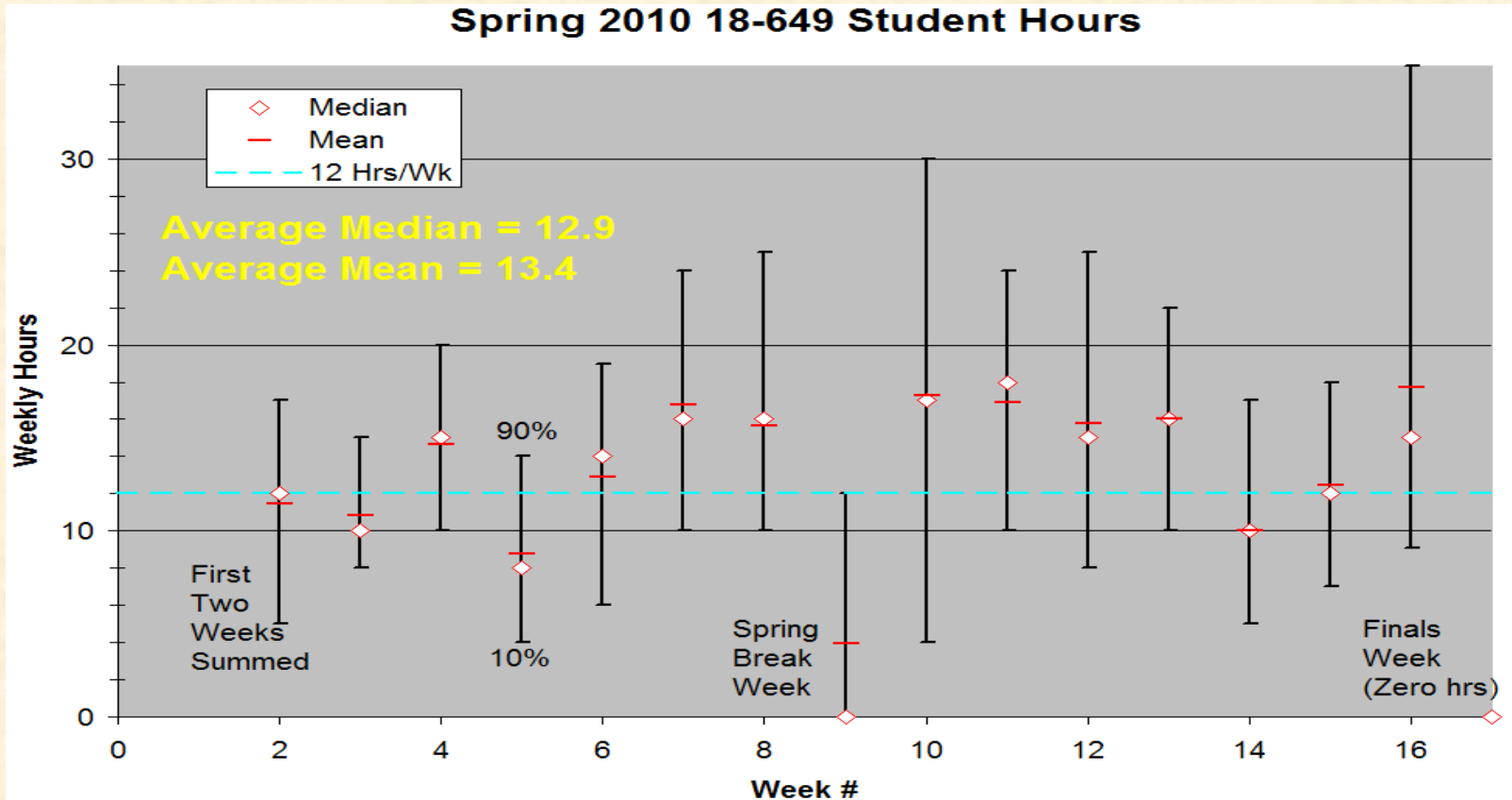
After Spreadsheet & Weekly Defect Reporting

Spring 2011 18-649 Student Hours
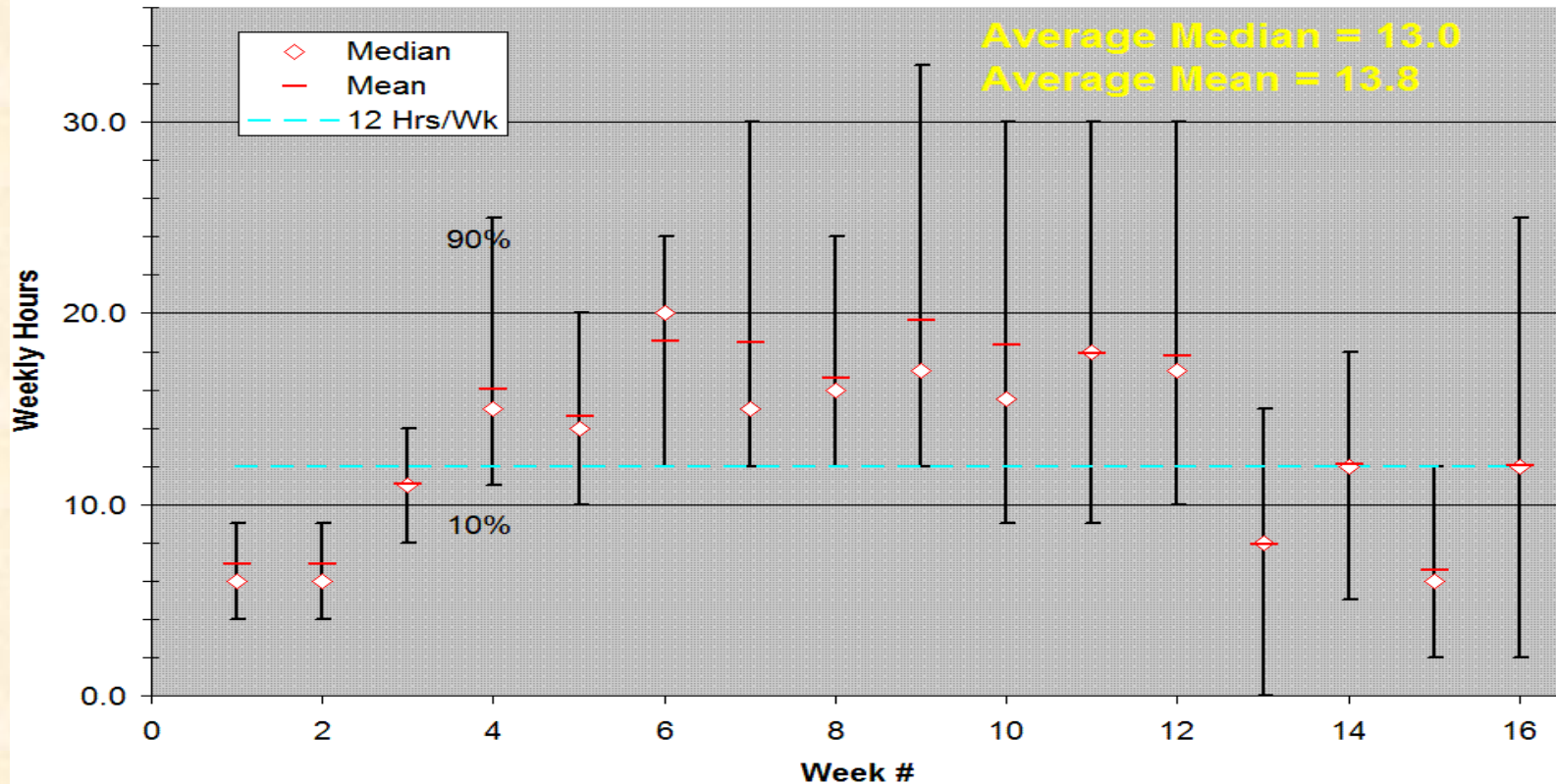
Review More Than Just The Code

© 2017 Philip Koopman **12**

The Economics Of Peer Review

- **Peer reviews provide more eyeballs to find bugs in an affordable way**
  - Good embedded coding rate is <u>1-2 lines of code/person-hr</u>
    - (Across entire project, including reqts, test, etc.)
  - A person can review 50-100 times faster than they can write code
    - If you have 4 people reviewing, that is still >10x faster than writing!
  - How much does peer review cost?
    - 4 people * 100-200 lines of code reviewed per hour
    - E.g., 300 lines; 4 people; 2 hrs review+1 hr prep = <u>25 LOC/person-hr</u>
  - **Reviews are only about 5%-10% of your project cost**
- **Good peer reviews find at least half the bugs!**
  - And they find them early, so total project cost can be reduced

- **Why is it folks say they don't have time to do peer reviews?**
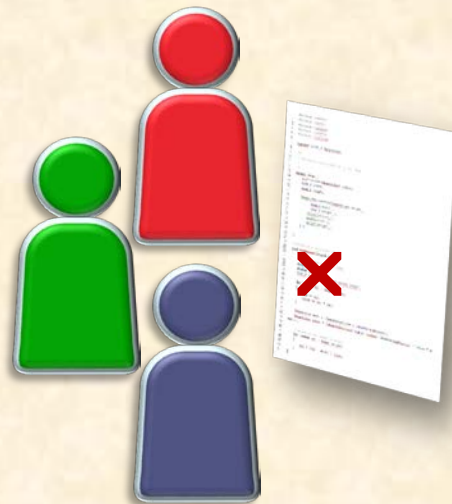
13

Peer Review Best Practices

- **Formal reviews (inspections) optimize bugs/$**
  - **Target 10% of project effort to find 50% of bugs**
    - You can review 100x faster than write code; it's cheap
  - Review everything written down, not just code
  - Use a perspective-based checklist to find more bugs
- **Review pitfalls**
  - If your reviews find <50% of defects, they are **BROKEN**
    - The 80/20 rule does NOT apply to review formality! Formal reviews are best.
    - You can't review at end; need to review throughout project
- **Review tools**
  - On-line review tools are OK, but not a substitute for in-person meeting
  - Static analysis tools are great – but not a review!