Inspection

PSU CS 300 Lecture 8-2

Bart Massey
Assoc Prof Computer Science
Portland State University
<bart@cs.pdx.edu>

Reviews

- **Managerial, Technical**
- Requirements, Design, **Code,** Tests, Changes...
- **Formal Inspection, Code Reading,** ``Structured Walkthrough'', Informal Walkthrough, Desk Checking, etc...

Code Reviews

- Perhaps least important phase to review (esp. if detailed design reviewed)
- Important to consider human capacities:
  - Input reasonably defect-free
  - Concentration on common mistakes
  - Concentration on semantic mistakes
  - Reasonable pace, expectations

Formal Technical Inspection of Code

- Methodology due to Fagan (IBM)
- Emphasis on detection, measurement
- Good information about helpfulness
- Slow to move into smaller SW projects

Code Inspection Roles

- Author: as passive as possible
  - answers questions
  - often reads, sometimes records
- Moderator: trained, sets pace, mediates
- Reader: reads code
- Recorder: records defects
- Reviewer: reviews the code

Code Inspection Preparation

- Assign code to reviewers
- Each reviewer
  - reads code carefully
  - completes any checklist(s)

Ideal Inspection Size

- Work
  - 200-500 lines of code (= 1 module!)
  - 1-3 hours
  - space between sessions
- People
  - 3-6
  - best with 3-4

Code Inspection Process

- Moderator instructs reader to proceed
- Reader reads line of code
- Reviewers signal any potential questions, defects
- Recorder records defects
- ...continue

Code Inspection Pitfalls

- Personalities
- Moderator rate/workload wrong
- Defect correction rather than detection
- Checklist issues
- Groupthink, wrong level of detail...
- Bad recording or followup

Code Reading

- Inspection without the meeting...
- Various levels of formality
- Good way to get code review started
- Can be almost as effective as inspection

Review and Code Quality

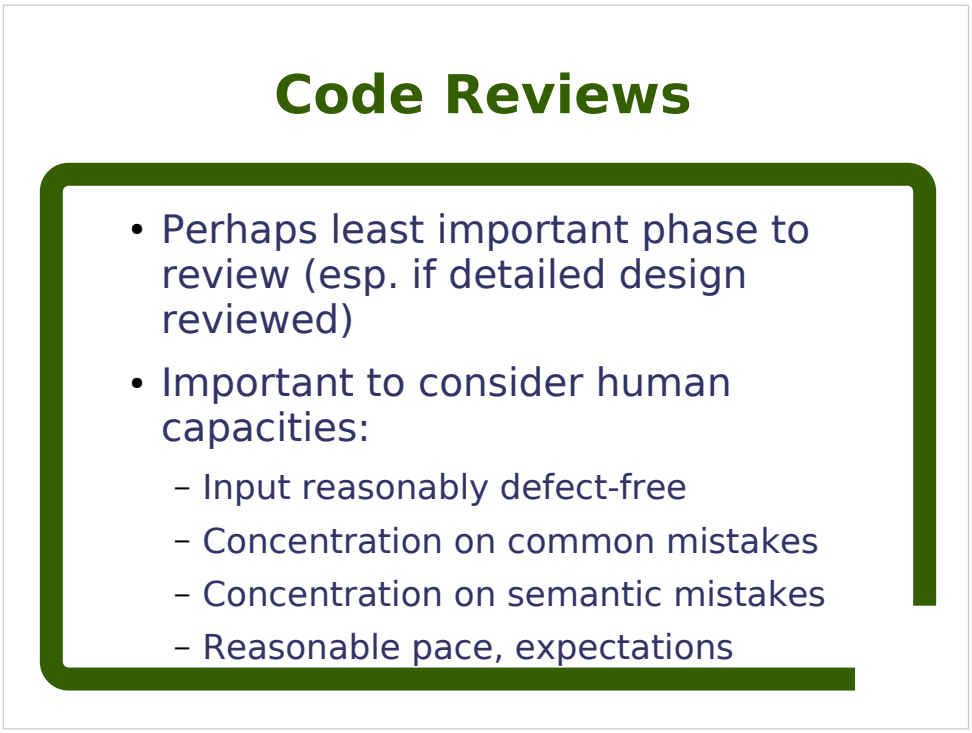
- Much easier to review
  - well-written code
  - designed code
- Much more effective to review low-defect code
  - modular
  - esp. ADT
  - not pretuned

Inspection

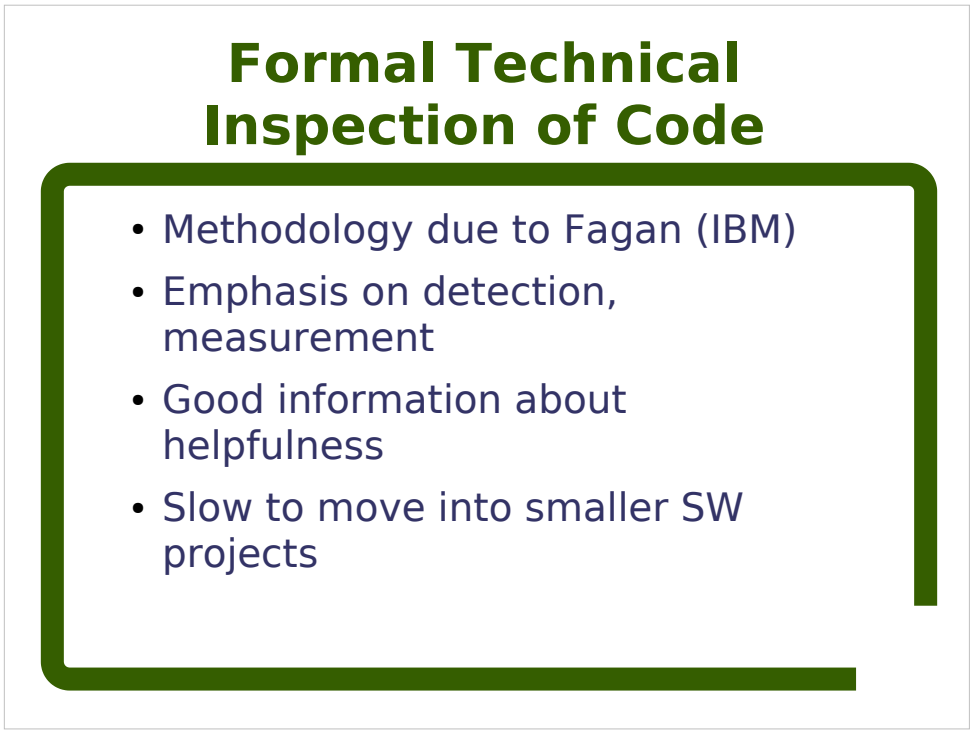
PSU CS 300 Lecture 8-2

Bart Massey
Assoc Prof Computer Science
Portland State University
<bart@cs.pdx.edu>

The goal of a technical review is to address specific, quantifiable, technical quality issues.
Most reports from review projects show fairly good results regardless of the actual type of review. This should encourage a sensible organization to do *something* rather than the current default...



Reviewing is one of those activities that is a moderate mismatch with human strengths. Thus, one must be careful to focus the review on things humans can do, in manageably sized chunks.



Neither traditional CS degrees nor smaller shops normally have any training in inspection, which perpetuates the lack of it in many modern environments.

Code Inspection Roles

- Author: as passive as possible
  - answers questions
  - often reads, sometimes records
- Moderator: trained, sets pace, mediates
- Reader: reads code
- Recorder: records defects
- Reviewer: reviews the code

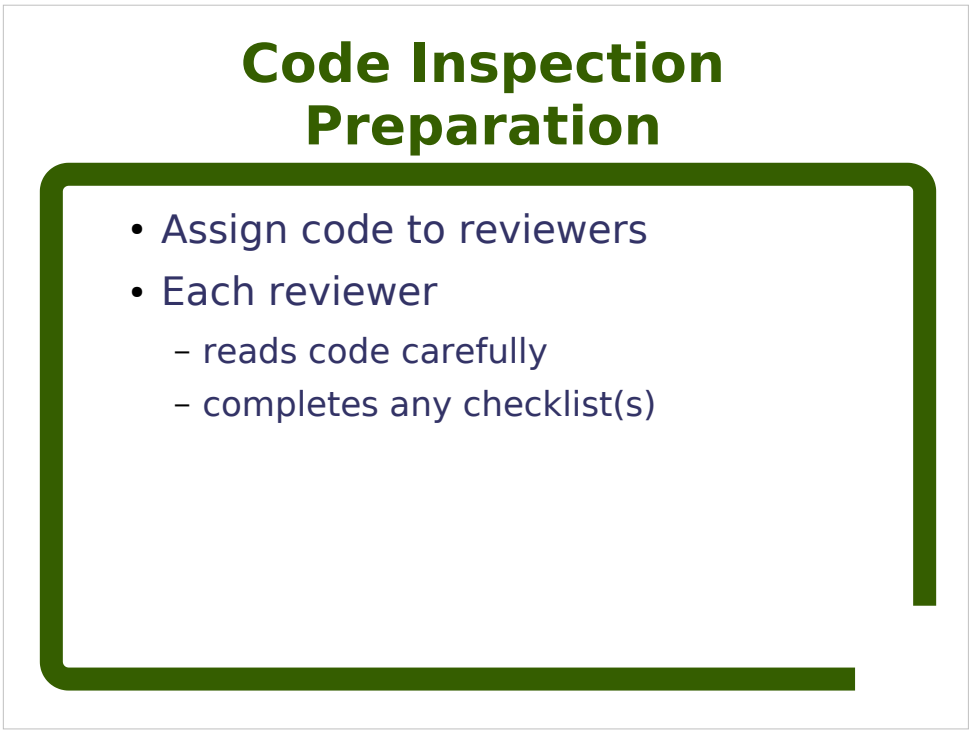
The author should *not*

volunteer defect information
suggest corrections on line
accept praise or blame

The reviewers should *not*

``attack''
suggest corrections on line
stray from the task at hand (*e.g.* design comments)

The recorder must note as much information as possible about each defect, and as little else as possible!

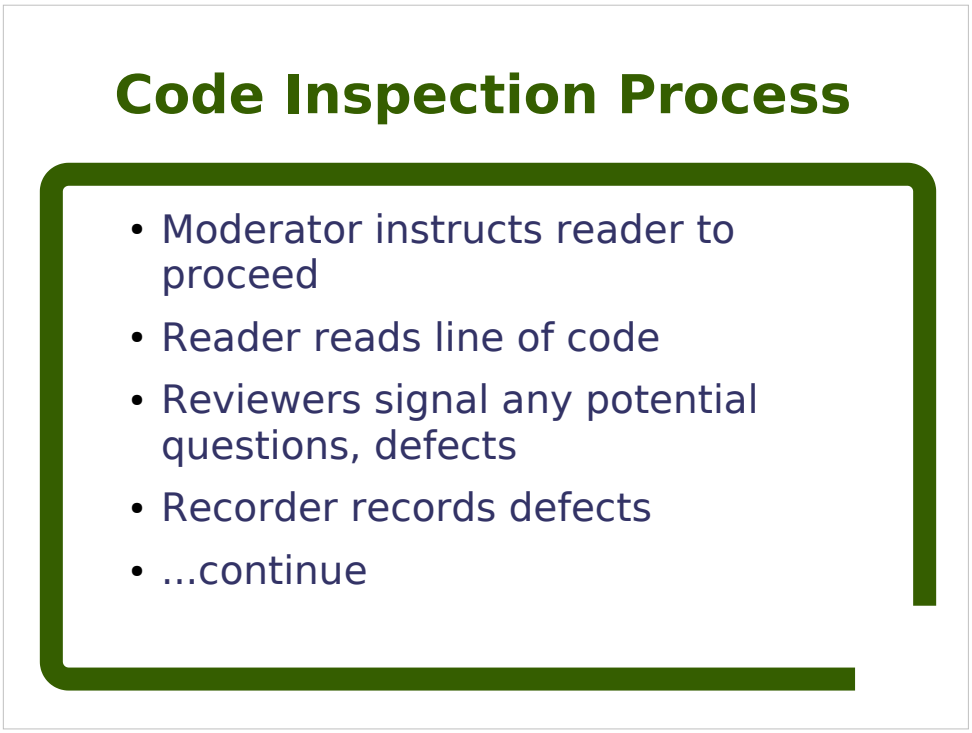


Checklists should not be used as a substitute for a brain. Besides, if done correctly, they will be too short to even look like one.

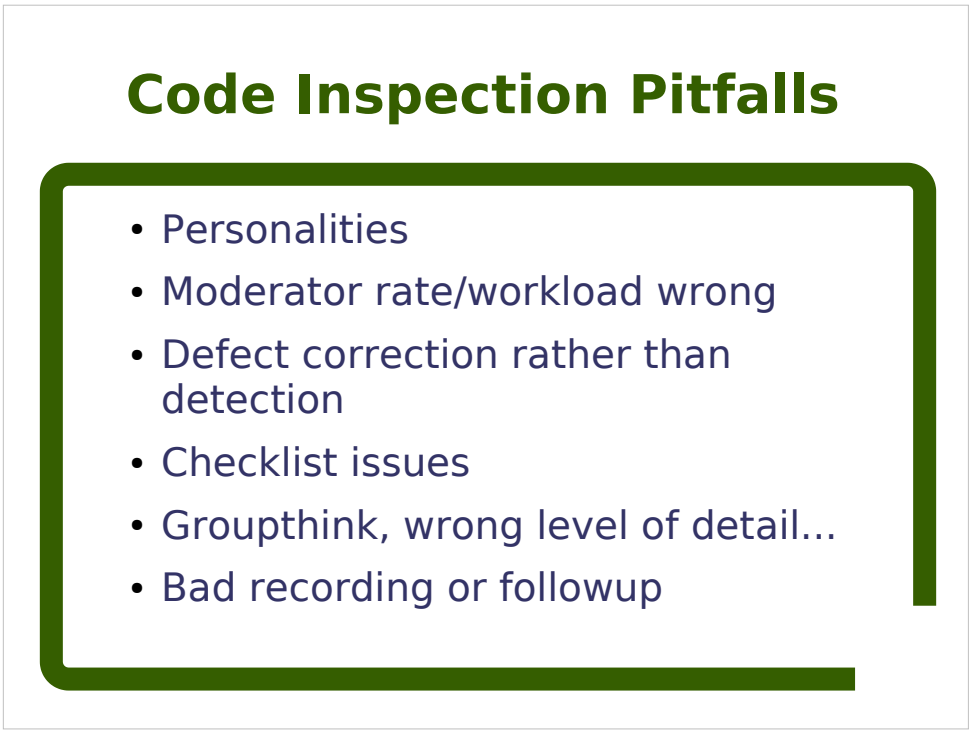
Ideal Inspection Size

- Work
  - 200-500 lines of code (= 1 module!)
  - 1-3 hours
  - space between sessions
- People
  - 3-6
  - best with 3-4

These numbers are of course approximate. The advice of your text is good on this: figure out what should work in advance, and *stick to it!*



I hold with the view that anything that looks like a defect is: it may not be a functional defect, but it's at least a coding style or documentation defect.
Note that if this is done properly, there will be no place for *egos* to play. The process is deliberately rigged to be as impersonal as possible, to avoid this. The moderator must stop arguments, and really even limit discussion, as quickly as possible: there'll be plenty of time for that after the meeting.



Checklists are great, but they should be short and useful. Whenever adding to one, drop something off as well.

Code Reading

- Inspection without the meeting...
- Various levels of formality
- Good way to get code review started
- Can be almost as effective as inspection

Note that this lecture focuses on code reading for review of putatively good code; there's also the art of reading legacy code in useful ways...

For an organization that has no code review at all, a good way to start is by mandating that no code will be shipped until it has been read formally by at least *k* of the staff (probably *k=1* or *2*) other than the author. This can be a tool for usefully introducing checklists as well.

Review and Code Quality

- Much easier to review
  - well-written code
  - designed code
- Much more effective to review low-defect code
  - modular
  - esp. ADT
  - not pretuned

Thus, all the other stuff we've talked about so far interacts with the review process. It is thus important to get it all right as much as possible.