Announcements

- Course information
  - Blackboard: discussion, turn-in
  - Web site: everything else
    - http://www.cs.cmu.edu/~aldrich/courses/654/
- First assignment is out
  - Topic: Specification Inspection, ODC, and Java
  - Mostly a group assignment
  - Due Monday, January 19, at 10:30am
- Next page: Policies

Policies

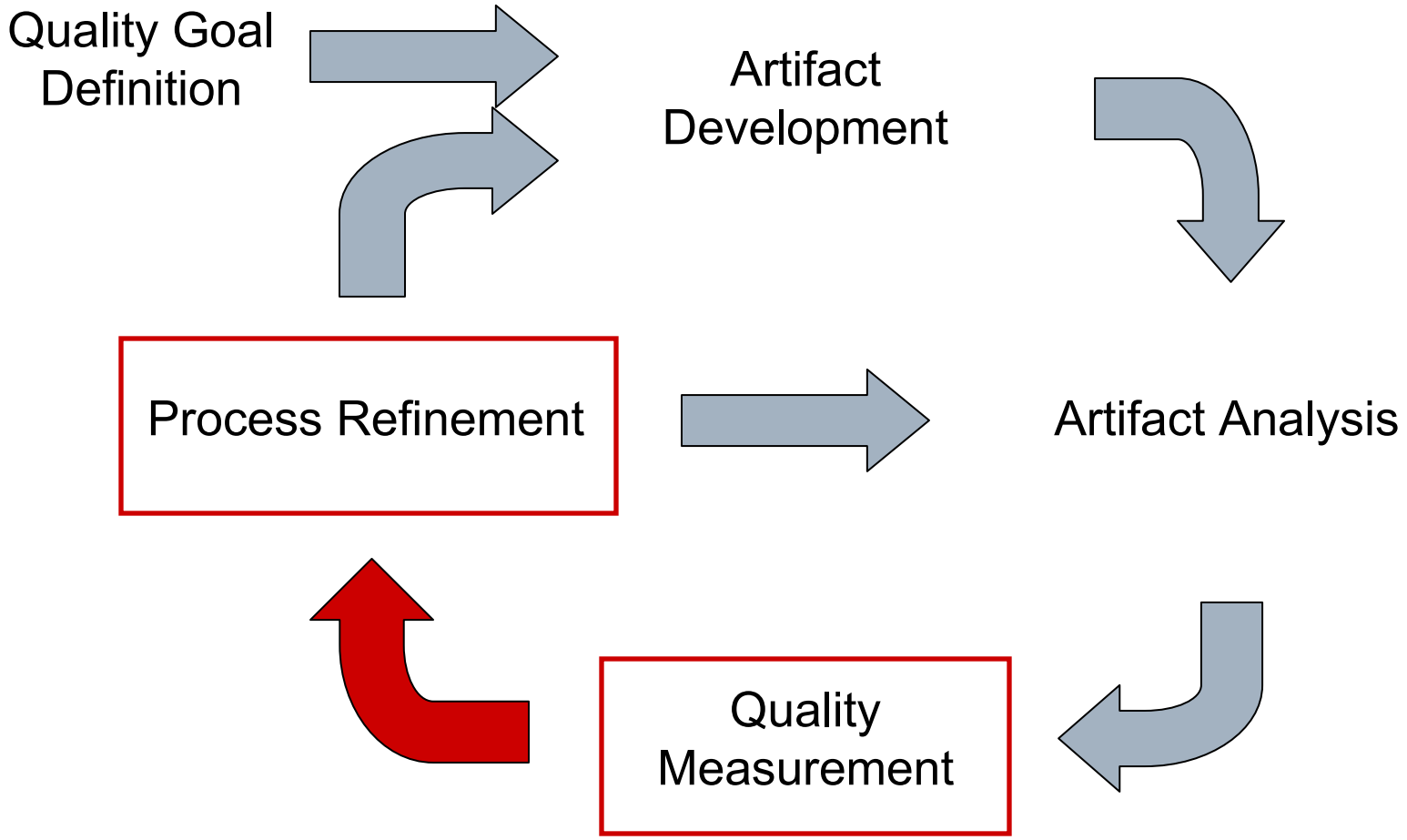
- Time Management
  - Keep track of time spent on each assignment
- Late Work
  - 5 free late days
    - can be used on non-critical path assignments only
  - No other late work except under extraordinary circumstances
- Collaboration Policy
  - You may discuss the lectures and assignments with others, and help each other with technical problems
  - Your work must be your own. You may not look at other solutions before doing your own. If you discuss an assignment with others, throw away your notes and work from the beginning yourself.
  - You must cite sources if you use or paraphrase any material
  - If you have any questions, ask the instructor or TAs

Analysis of Software Artifacts

Orthogonal Defect Classification (continued)

Jonathan Aldrich

Analysis in a Process Context



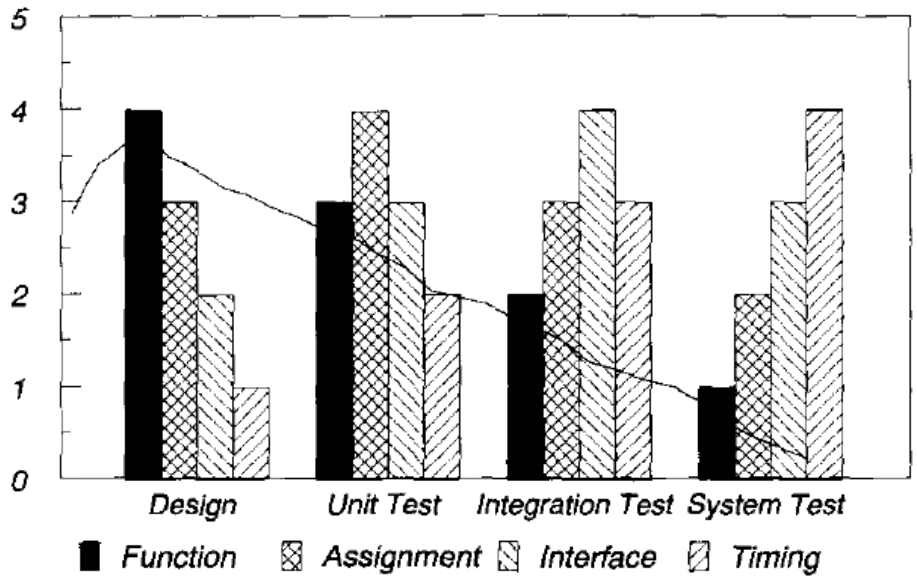
Principled Process Refinement: Orthogonal Defect Classification (ODC)

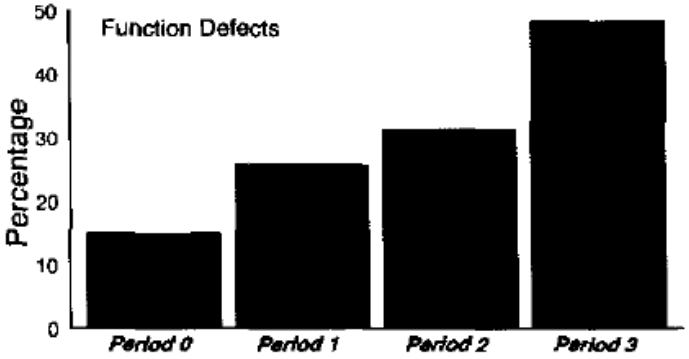
- Analyzing defect data to refine QA process
  - Need to know where defects are introduced and where found
- A defect's type is related to where it was introduced
  - Hypothesis: can estimate defect type with less bias than directly estimate of phase where defect was introduced

ODC Defect Types

Defect Type	Description	Source
Function	Capability affected requiring design change	Design
Interface	Error interacting with other components	Low-level design
Checking	Data not properly validated before use	Low-level design or code
Assignment	Assigned / initialized incorrect value	Code
Timing	Incorrect management of shared resources	Low-level design
Build	Mistakes in libraries, change management	Libraries/tools
Documentation	Incorrect documentation	Publications
Algorithm	Local problems that do not require design change	Low-level design

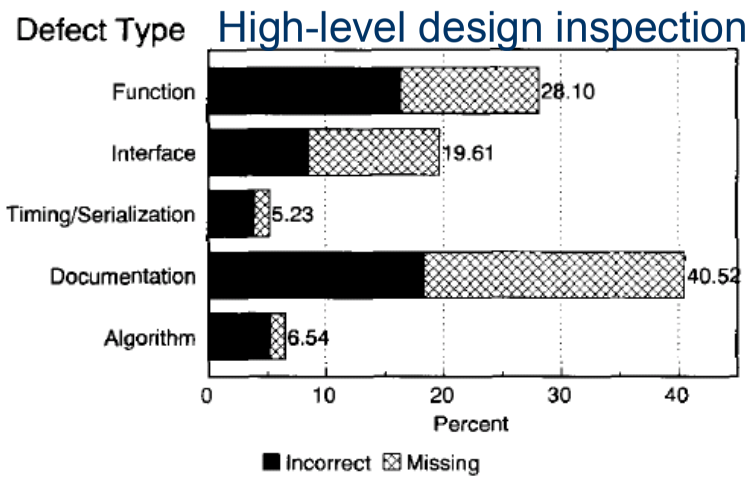
Example:
Analyzing Defect Type Distribution over Time

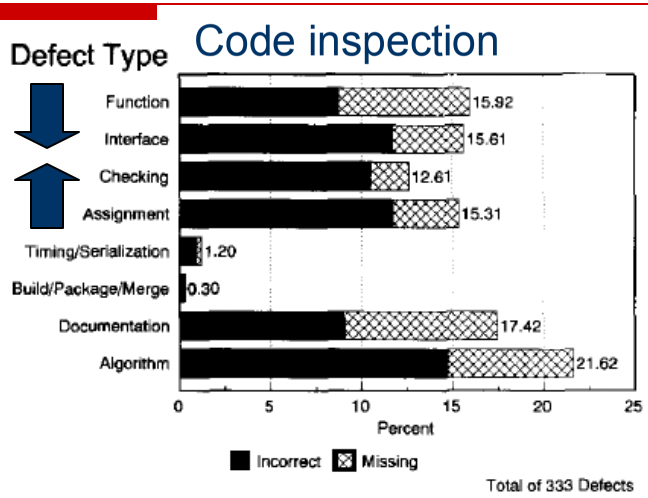


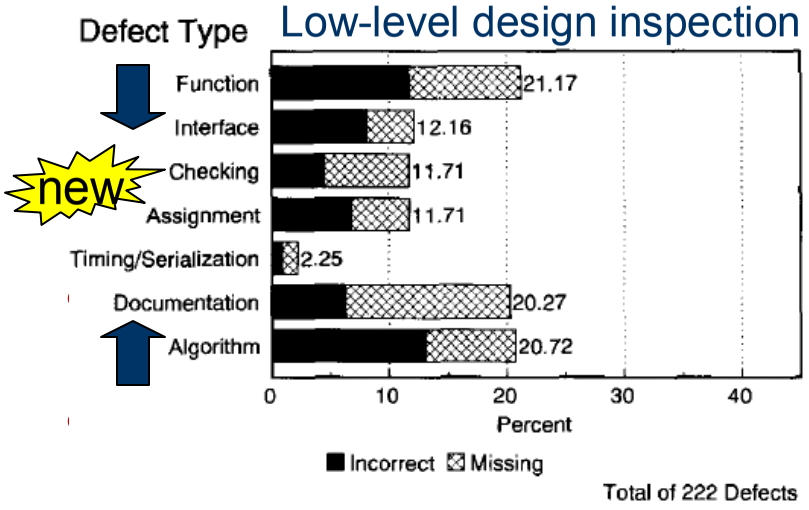


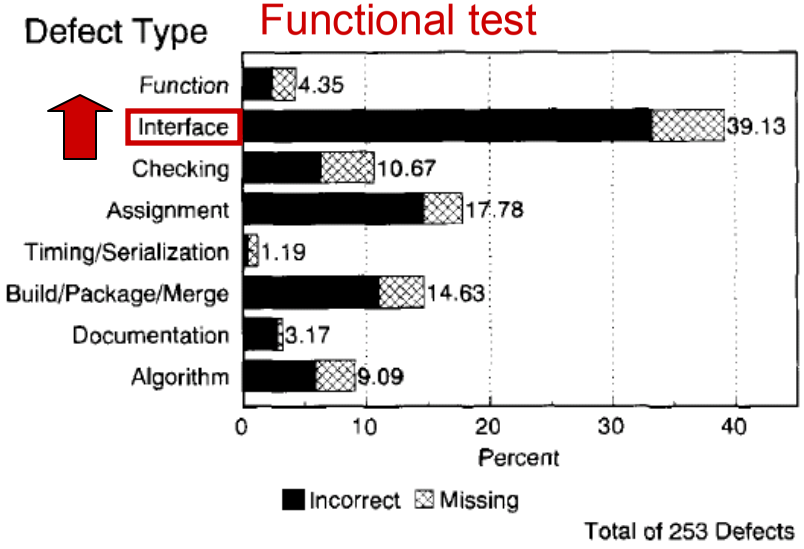
- Ideal curves
  - Function peaks in design inspection
  - Assignment peaks at unit test
  - Interface peaks at integration test
  - Timing peaks at system test
- Actual curve for function increases at each stage!
  - What does this tell us? What would you do about it?
    - Something was missed in design
    - Better to re-do design rather than keep testing

Example:
Analyzing Defect Type Distribution over Time

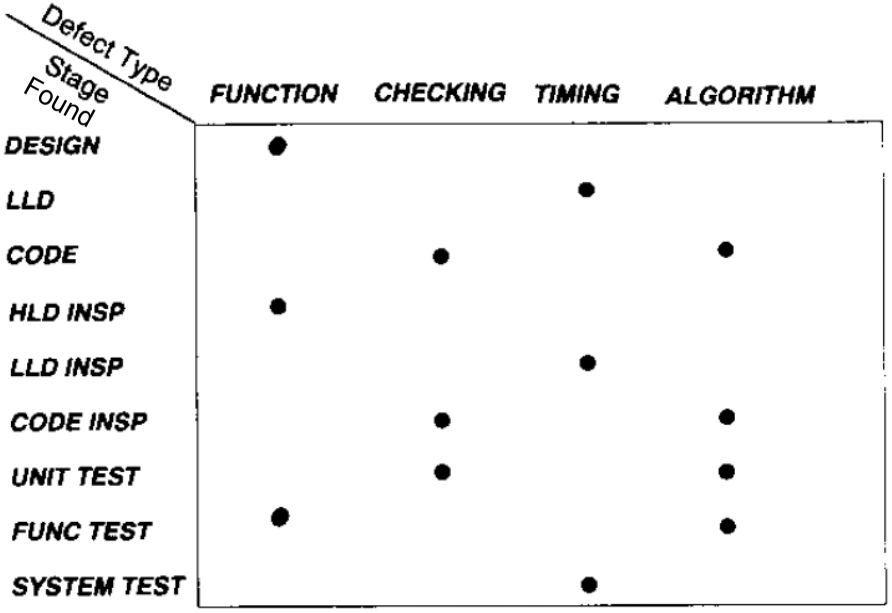


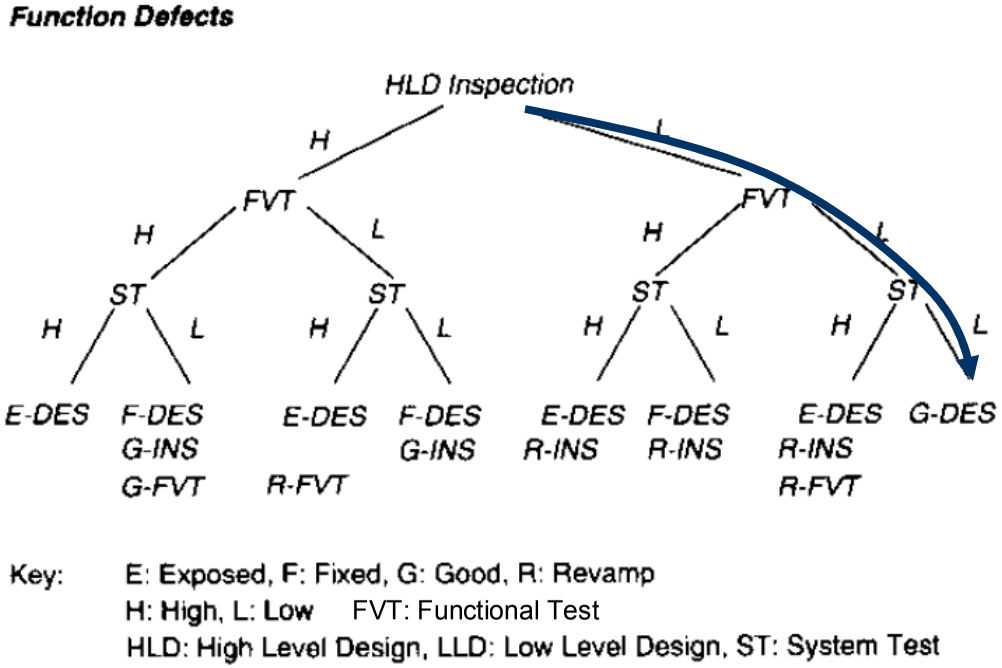






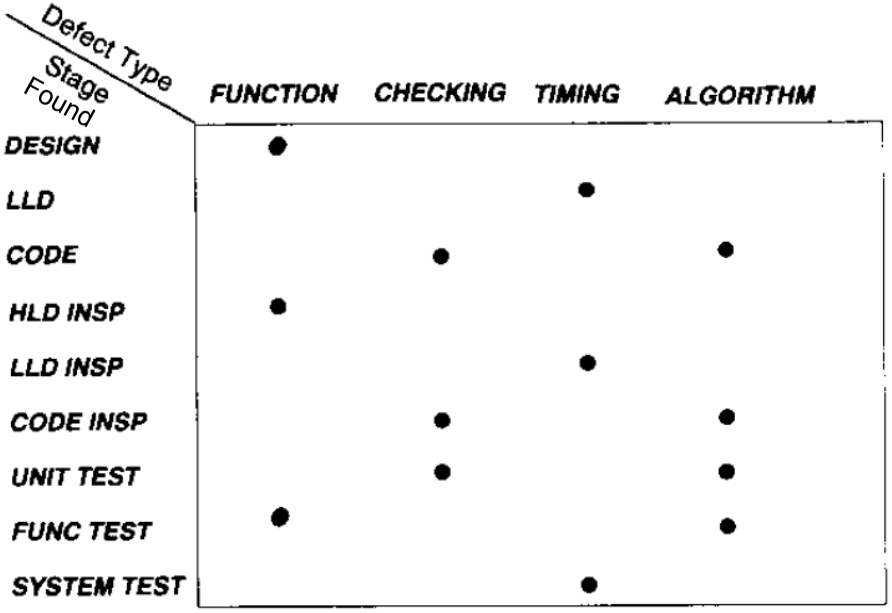
Process Inferences

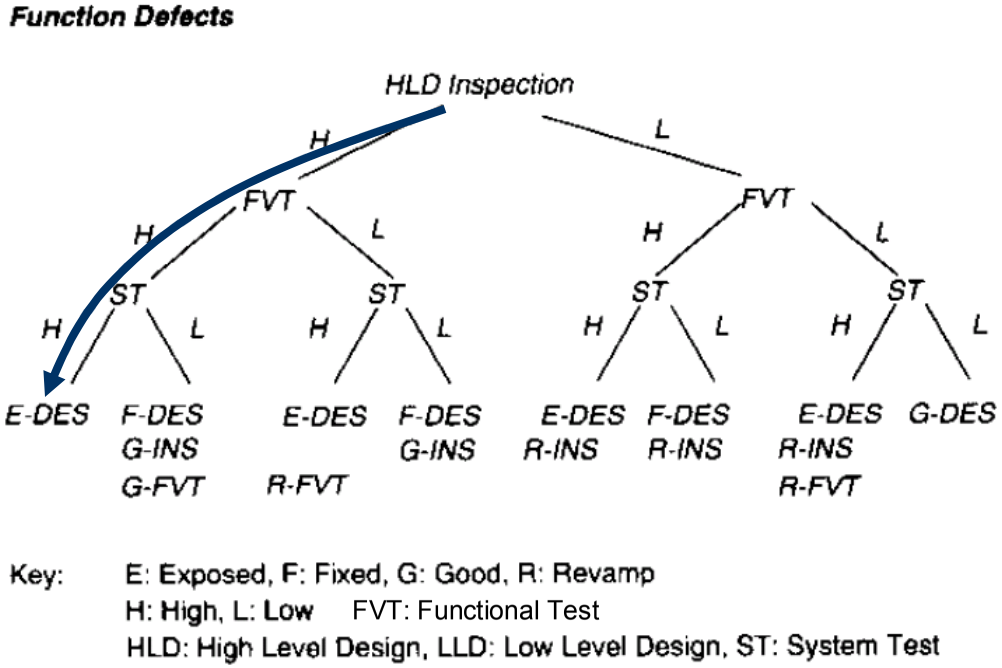




- No errors => good work

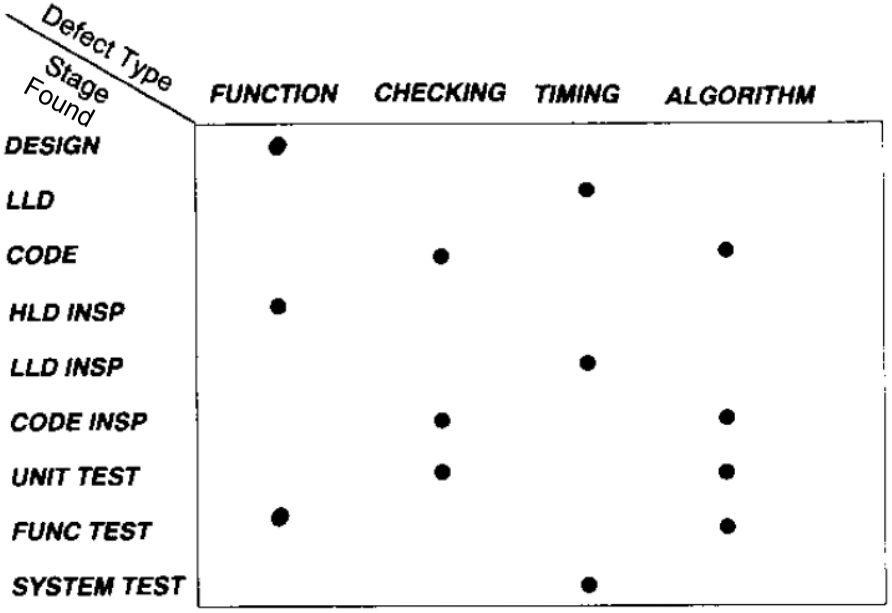
Process Inferences

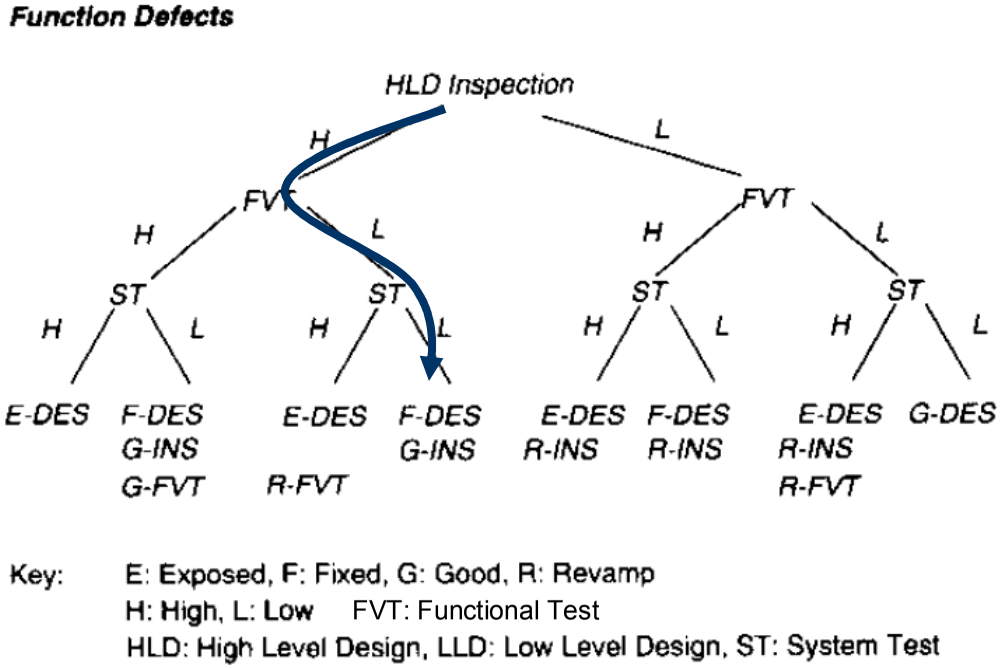




- No errors => good work
- High errors throughout => trouble

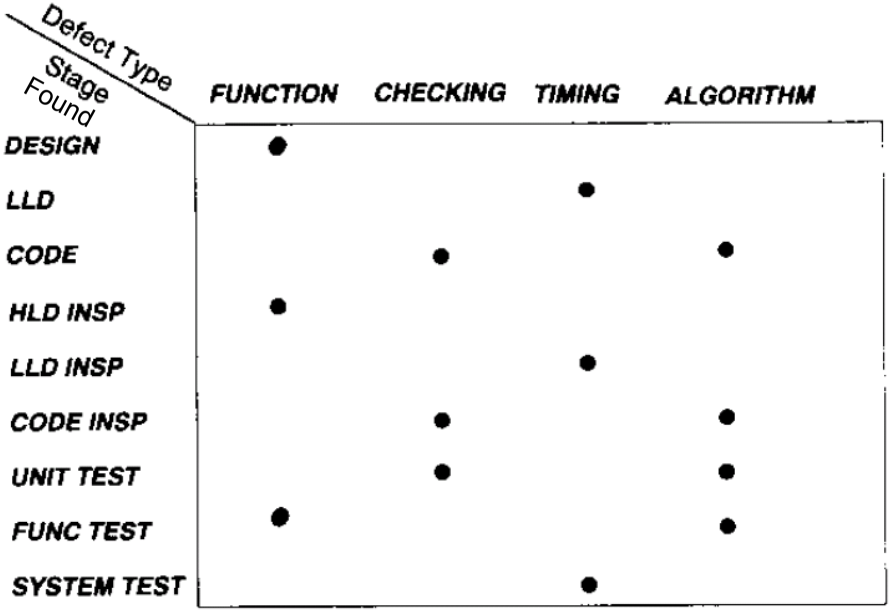
Process Inferences

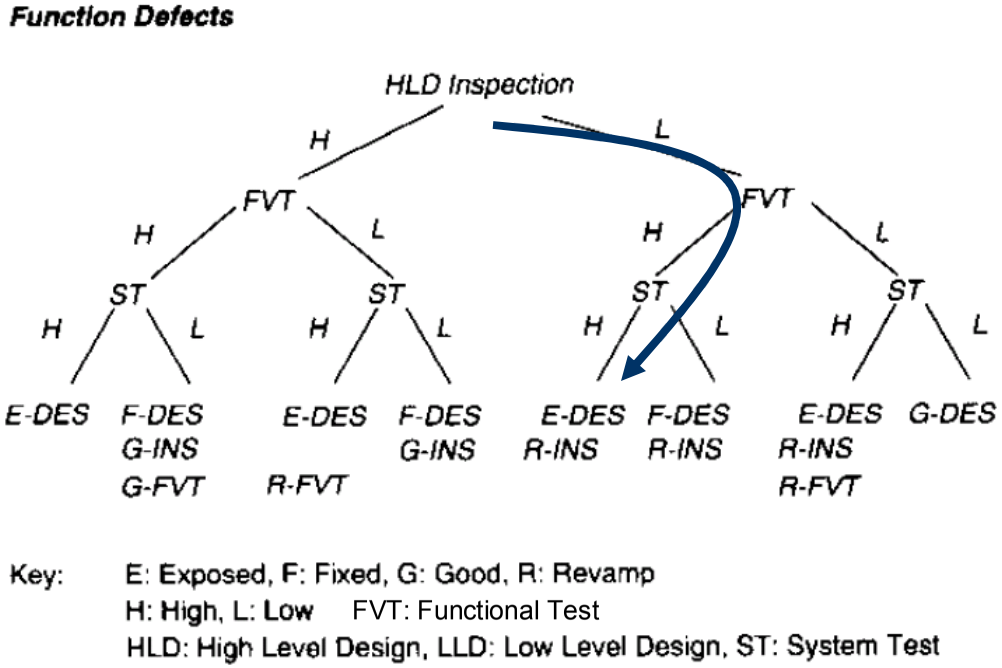




- No errors => good work
- High errors throughout => trouble
- High errors then low => fixed w/ good stage

Process Inferences





- No errors => good work
- High errors throughout => trouble
- High errors then low => fixed w/ good stage
- Low then high => revamp stage

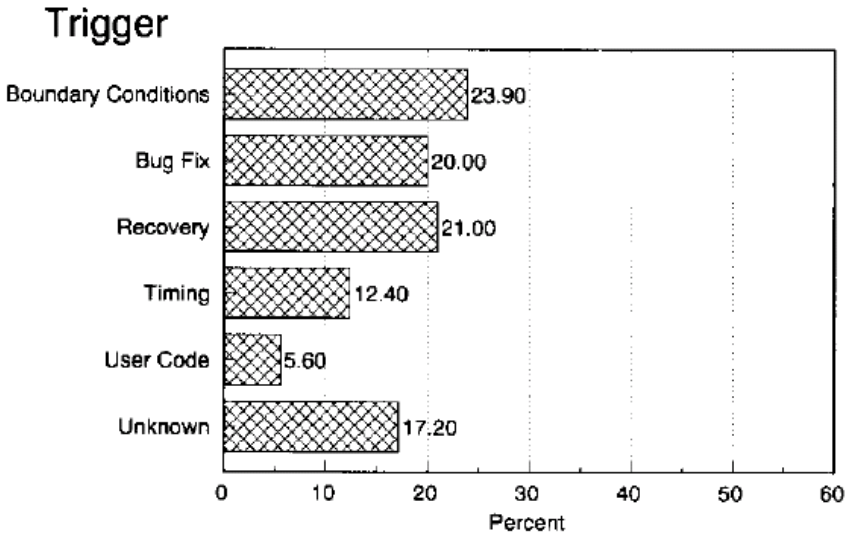
Defect Triggers

- Trigger – a condition that allows a defect to surface

- Examples
  - bug fix
  - boundary conditions
  - exception handling
  - timing
  - workload

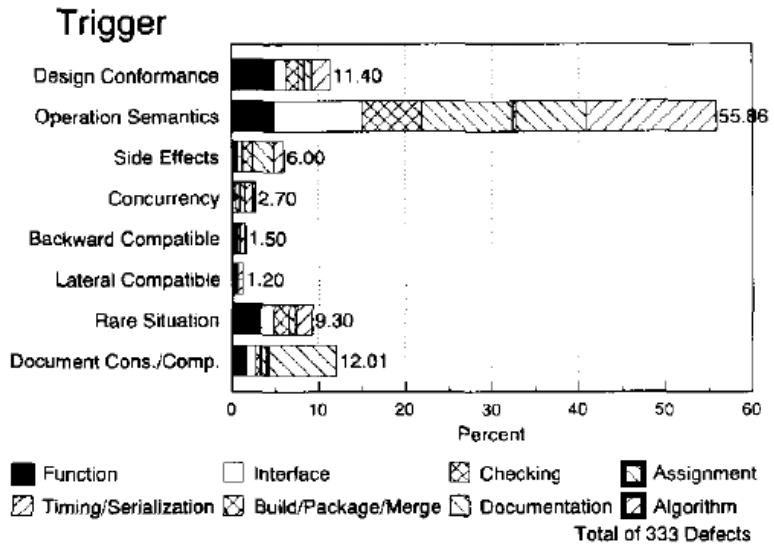
- Why useful?
  - If defects in the field differ from defects found in test, that points out an inadequacy of the test suite

Defect Triggers



- Example from pilot
  - Most defects triggered by boundary conditions
  - Intervention: invest more time in inspections looking at boundary conditions

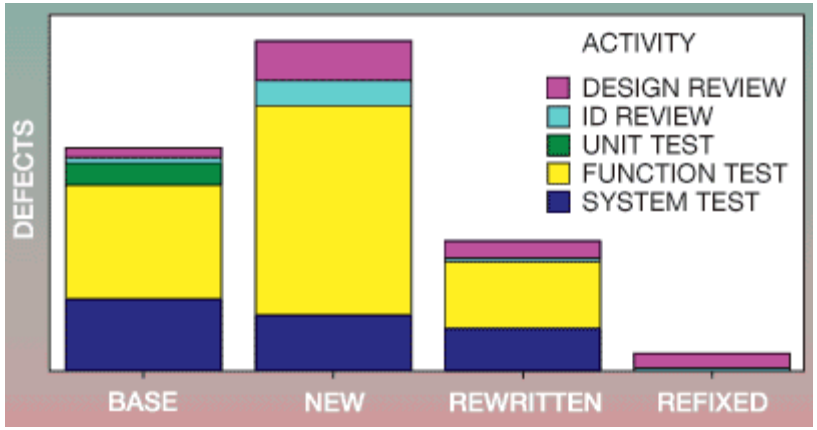
Defect Triggers

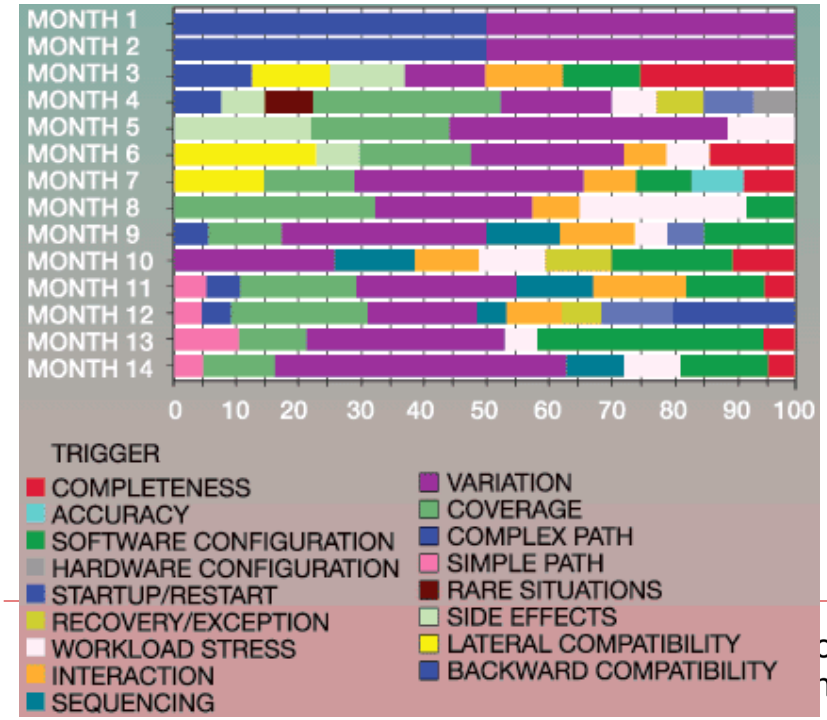


- Another pilot
  - Context: significant interaction with other components
  - Observation: few lateral compatibility triggers
  - Suggests poor review process (or extraordinarily talented team)

ODC: Case Study 1

Source: Butcher, Munro, and Kratschmer. Improving Software Testing via ODC: Three Case Studies. IBM Systems Journal 41(1), 2002.





- Surprisingly many errors in base code
- Majority should have been caught in function test
- Response: more regression testing

- Many field defects had "variation" trigger
  - different inputs for a function
- Response: more variation testing

ODC: Case Study 2 & 3

Source: Butcher, Munro, and Kratschmer. Improving Software Testing via ODC: Three Case Studies. IBM Systems Journal 41(1), 2002.

- Case study 2
  - Good signs led to early entry into system test
    - broad range of triggers & some complex triggers: coverage, variation, sequence, interaction, recovery/exception, startup/restart
  - Field triggers included a lot of software configuration
    - do more configuration testing
  - A lot of "checking" defects from missing code
    - broaden testing with code coverage tool and analyzing test cases for trigger coverage
- Case study 3
  - Assessed a project as not ready for release
    - Fewer than expected defects found in functional testing relative to GUI review
      - functional testing inadequate
    - Simple defects found in functional testing (coverage, variation triggers)
      - more interesting defects not exposed yet!
    - Majority of defects found in old code
      - should have been caught before!

ODC Summary

- A principled approach to tracking defects can provide insight into improving process
- Orthogonal Defect Classification
  - Objective defect types used to estimate defect introduction phase
  - Defect trigger distribution used to identify weaknesses in testing
- ODC provided useful feedback in case studies

Analysis of Software Artifacts

Inspection

Jonathan Aldrich

The Computer's Perspective

http://www.xkcd.com/371/

used by permission

Inspection – The Big Questions

1. **What is inspection?**
   - And what are the benefits?
2. **When are inspections better than testing?**
   - What kind of attributes?
   - What is the typical experience of firms with inspection?
3. **Are there different kinds of inspections?**
   - What are the relative benefits of each?
4. **Who are the inspection participants?**
   - Roles played and their benefits
5. **How is the inspection process accomplished?**
   - What are summary guidelines for the meetings?
6. **What gets inspected?**
   - And when to do inspections?

Software Inspections

1. What are software inspections (reviews)?
   - Meetings (real or virtual) during which designs and code are reviewed by people other than the original developer.

- What are the benefits of inspections?
  - New perspective
    - Finding defects may be easier for people who haven't seen the artifact before and don't have preconceived ideas about its correctness
  - Knowledge sharing
    - Regarding designs and specific software artifacts
    - Regarding defect detection practices
  - Find flaws early
    - Can dramatically reduce cost of fixing them
    - During detail design – even before code is written
    - Or code that does not yet have a test harness
    - Or code in which testing has found flaws but root causes are not understood
  - Reduce rework and testing effort
    - Can reduce overall development effort

Source material
Peer Reviews in Software: A Practical Guide.
Karl E. Wiegers.
Additional material from William Scherlis.

Inspections vs. Testing

2. What attributes are well-handled by inspections but not testing?

- Characteristics of code
  - Maintainability, evolvability, reusability
- Other properties tough to test
  - Scalability, efficiency
  - Security, integrity
  - Robustness, reliability, exception handling
- Requirements, architecture, design documents
  - Cannot “execute” these as a test

Experience with inspection

- Raytheon
  - Reduced "rework" from 41% of cost to 20% of cost
  - Reduced effort to fix integration problems by 80%
- Paulk et al.: cost to fix a defect in space shuttle software
  - $1 if found in inspection
  - $13 during system test
  - $92 after delivery
- IBM
  - 1 hour of inspection saved 20 hours of testing
  - Saved 82 hours of rework if defects in released product
- IBM Santa Teresa Lab
  - 3.5 hours to find bug with inspection, 15-25 through testing
- C. Jones
  - Design/code inspections remove 50-70% of defects
  - Testing removes 35%
- R. Grady, efficiency data from HP
  - System use 0.21 defects/hour
  - Black box 0.28 defects/hour
  - White box 0.32 defects/hour
  - Reading/inspect 1.06 defects/hour
- Your mileage may vary
  - Studies give different answers
  - These results show what is possible

Kinds of Inspections

Inspections / Formal Technical Reviews

- Participation defined by policy
  - Developers
  - Designated key individuals – peers, QA team, Review Board, etc.
- Advance preparation by participants
  - Typically based on checklists
- Formal meeting to discuss artifact
  - Led by moderator, not author
  - Documented process followed
  - May be virtual or conferenced
- Formal follow-up process
  - Written deliverable from review
  - Appraise product

Walkthroughs

- No advance preparation
- Author leads discussion in meeting
- No formal follow-up
- Low cost, valuable for education

Other review approaches

- Pass-around – preparation part of an inspection
- Peer desk check – examination by a single reviewer (like pair programming)
- Ad-hoc – informal feedback from a team member

There are tradeoffs among the techniques

- Formal reviews typically find more bugs
  - Ford Motor: 50% more bugs found
- But they also cost more

Inspection – The Big Questions

1. **What is inspection?**
   - And what are the benefits?
2. **When are inspections better than testing?**
   - What kind of attributes?
   - What is the typical experience of firms with inspection?
3. **Are there different kinds of inspections?**
   - What are the relative benefits of each?
4. **Who are the inspection participants?**
   - Roles played and their benefits
5. **How is the inspection process accomplished?**
   - What are summary guidelines for the meetings?
6. **What gets inspected?**
   - And when to do inspections?

Review Roles: **Moderator** and **Recorder**

4. Who are the stakeholders in inspection?

Moderator

- Organizes review
  - Keeps discussion on track
  - Ensures follow-up happens
- Key characteristics
  - Good facilitator
  - Knowledgeable
  - Impartial and respected
  - Can hold participants accountable and correct inappropriate behavior

Recorder

- Captures a log of the inspection process

Review Roles: **Reader**

Reader

- Presents material
  - Describes interpretation of each point
  - Discuss different interpretations by other team members
- Why should the Reader be different from the Author?
  - Reveals ambiguities
    - If author were to present, others might not mention that their interpretation was different
- Why not just ask for comments section by section?
  - Can be faster
  - Downside: does not capture differing perspectives as effectively

Review Roles: **Author**

Author

- Describes rationale for work
- Not moderator or reader
  - Conflict between objectivity required of moderator/reader and advocacy for the author's own work
  - Others raise issues more comfortably
- Not recorder
  - Temptation to not write down issues the author disagrees with
- Why should the Author attend? Are there downsides?
  - Gain insight from others' perspectives
  - Can answer questions
  - Can contribute to discussion based on knowledge of artifact
  - Potential downside: meeting may be confrontational

Inspection – The Big Questions

1. **What is inspection?**
   - And what are the benefits?
2. **When are inspections better than testing?**
   - What kind of attributes?
   - What is the typical experience of firms with inspection?
3. **Are there different kinds of inspections?**
   - What are the relative benefits of each?
4. **Who are the inspection participants?**
   - Roles played and their benefits
5. **How is the inspection process accomplished?**
   - What are summary guidelines for the meetings?
6. **What gets inspected?**
   - And when to do inspections?

Process: Planning

5. How is the inspection process accomplished?

Planning

- Determine objectives
- Choose moderator
- Identify inspectors
  - Good to involve people with connection to artifact
    - e.g. depends on, interfaces with
- Schedule meeting(s)
  - General guideline: 150-200 SLOC/hour, or 3-4 pages/hour
- Prepare and distribute inspection package
  - Deliverable, supporting docs, checklists
  - Cross-reference specs, standards

Process

Overview meeting

- Informal meeting
- Goal: go over features, assumptions, background, context
- Optional stage
  - May be able to use paper overview or shared context

Preparation (*Why?*)

- Inspectors examine deliverable
  - Defects: cause an error in the product
  - Non-defects: improvements, clarification, style, questions
    - May want to list typos/spelling/format/style separately and not discuss during the meeting
  - Conformance to standards & specification
  - Often use checklist
- General guideline
  - prep time ~ meeting time

Process: Meeting

The Meeting

- *Reader* describes one segment at a time
  - *Inspectors* respond: defects, questions, suggestions
- *Recorder* writes down each defect, suggestion, issue
  - This is the primary deliverable
- *Moderator*
  - Avoid problem solving (***why?***), inappropriate behavior, lack of participation
  - At conclusion: prepares report with appraisal and data
- Outcomes: Appraisal of product
  - Accepted (minor changes, no follow up)
  - Accepted conditionally (minor changes, verification)
  - Reinspect following rework (major changes)
  - Inspection not completed
- Outcomes: Input on improving inspection process
- *Variant*: reviewers make comments on electronic bulletin board
  - Cost is lower
  - Lose benefits of direct meeting (face to face, telephone)
    - Synergy - new bugs found (4%? 25%?)
    - Learning by participants
    - Communication about product

Process: Rework and Follow-up

Follow-up by author

- *Author* addresses each item
  - Ensure understanding of issue
    - Is it a defect or not? Is it a feature request or requirement change?
  - Fixes defects and makes improvements
    - Uncorrected/unverified defects go into defect tracking system
- Deliverables
  - Corrected work product
  - Response to each issue and rationale for action
- Moderator (or verifier) meets with author
  - Check resolution of issues
  - Examine corrected deliverable
- Author checks in code

Process: Analysis

Analysis

- Causal analysis
  - Analyze root causes of defects
- Make improvements to development and QA processes
  - Add issue to checklist
  - Change testing approach
  - Develop or purchase new static analysis
- Measuring effectiveness
  - Percentage of bugs found during inspection
    - vs. found by other means or afterwards (test, customer)
- Measuring efficiency
  - “Defects per hour”
  - Will decrease as your process improves

Meetings: Review Guidelines

- Build reviews into your schedule
  - Otherwise unexpected and viewed as intrusion
  - Recognize that reviews can accelerate schedule by reducing other V&V activities
- Keep review team small
  - General guidelines: 3-7 participants
    - 3 is minimum for formal process to work
    - Below 3, too few perspectives besides author
    - Above 7, work may be slowed by process, scheduling
  - Smaller groups for code, larger groups for other documents
    - Knowledge is spread around more, more stakeholders
    - Particular for requirements
- Find problems, but don't try to solve them
  - Typically less expensive to address 1-on-1
  - Guideline: halt solution discussion after 1-3 minutes
- Limit meetings to 2 hours maximum
  - Attention span gets lost beyond this
- Require advance preparation
  - Provides much of the value of a (formal) review

Discussion: Checklists

- What makes a good checklist?
  - Illustrate the principle with an example checklist item
- Principles
- Examples

Checklist Items from the Web

- Specification
  - Is documentation complete, including DBC or Error checking specs as appropriate?
- Design
  - Can better data structures or more efficient algorithms be used?
  - Are error messages comprehensive and provide guidance as to how to correct the problem?
  - Is there duplicate code that could be replaced by a call to a function that provides the behavior of the duplicate code?
  - Do any derived classes have common members that should be in the base class?
- Coding
  - Have all array (or other collection) indexes been prevented from going out-of-bounds?
  - Is integer arithmetic, especially division, used appropriately to avoid causing unexpected truncation/rounding?
  - Are all files closed properly, even in the case of an error?
  - Are all object references initialized before use?
  - In a switch statement, are all cases by break or return?
  - Are all objects (including Strings) compared with "equals" and not "=="?
- Style
  - Are descriptive variable and constant names used in accord with naming conventions?
  - Are there literal constants that should be named constants?

- I think the above are good examples (but not comprehensive). Sources:
  - http://users.csc.calpoly.edu/~jdalbey/205/Resources/InspectChecklist.html
  - http://undergraduate.csse.uwa.edu.au/units/CITS2220/assign2/JavaInspectionCheckList.pdf

Customizing Checklists

- What should be included in a checklist for a:
  - Operating system?
  - Online store?
  - Word processor?
  - Aircraft flight control system?
  - Real-time system?
  - Concurrent system?

Meetings: Checklists

- Benefits of checklists
  - Focus on likely sources of error
  - Form quality standard that aids preparers
  - Can bring up issues specific to a product
- Should be short
  - About seven items
    - If more, group and do multiple passes
- Focus
  - Priority issues
  - Issues unlikely to be found other ways
  - Historical problems
  - Issues specific to the document
- Start with checklist from well-known source
  - Refine based on experience

- Pitfall: overemphasis on style issues
  - It's good to find style issues in inspections, but other issues are higher priority – specification, design, correctness, security, …

People: Social Aspects of Reviews

- Reviews are challenging
  - Authors invest self-worth in product
  - Encourages you to avoid letting others find errors
- For Authors
  - Recognize value of feedback
  - Place value in making code easy to understand
  - Don’t take criticism of code personally
- For reviewers
  - Don’t show off how much better/smarter you are
  - Be sensitive to colleagues
    - Bad: "you didn't initialize this variable“
    - Good: "I didn't see where this variable was initialized"

Review Pitfalls

- Letting reviewers lead the quality process
  - Attitude: “why fix this, the reviewers will find it“
  - Responsibility for quality is with author, not reviewers
    - Reviewers help
- Insisting on perfection/completion before review
  - Makes harder to accept suggestions for change
- Using review statistics for HR evaluation
  - Real world example:
    - Manager decides "finding more than 5 bugs during an inspection would count against the author” [Weigers '02]
  - Negative effects
    - Avoid submitting for inspection
    - Submit small pieces at a time
    - Avoid pointing out defects in reviews (thus missing them)
    - Holding "pre-reviews" that waste time and skew metrics

Inspection – The Big Questions

1. **What is inspection?**
   - And what are the benefits?
2. **When are inspections better than testing?**
   - What kind of attributes?
   - What is the typical experience of firms with inspection?
3. **Are there different kinds of inspections?**
   - What are the relative benefits of each?
4. **Who are the inspection participants?**
   - Roles played and their benefits
5. **How is the inspection process accomplished?**
   - What are summary guidelines for the meetings?
6. **What gets inspected?**
   - And when to do inspections?

What to Inspect

- *First*, requirements documents; *second*, design documents
  - Difficult to validate in other ways
  - May have high associated risk
    - Especially important to get right
    - Cheaper to fix earlier on in process
  - Many different perspectives are helpful
  - Need involvement of multiple stakeholders
- *Third*, critical or uncertain pieces of code
  - Security-critical code
  - Safety-critical code
- Start inspections at the earliest stages of process
  - Catch mistakes early, when easy to fix
  - Allow rest of system to be built with knowledge gained
- Sample segments when there is a large body of work
  - Consider what are good “coverage” criteria

Questions?

Resources

- Wiegers text
  - **Peer Reviews in Software: A Practical Guide**
- A Microsoft perspective
  - http://msdn.microsoft.com/en-us/library/cc265075.aspx