Reviews and Inspections used in various ways

Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/conferences

Niels Malotaux

- Independent Engineering and Team Coach
- Expert in helping teams and organizations to quickly become
  - More effective - doing the right things better
  - More efficient - doing the right things better in less time
  - More predictable - delivering as needed
- Getting projects back on track
- Embedded Systems architect (electronics/firmware)
- Project types
  electronics, firmware, software, space, road, rail, telecom, industrial control, parking system

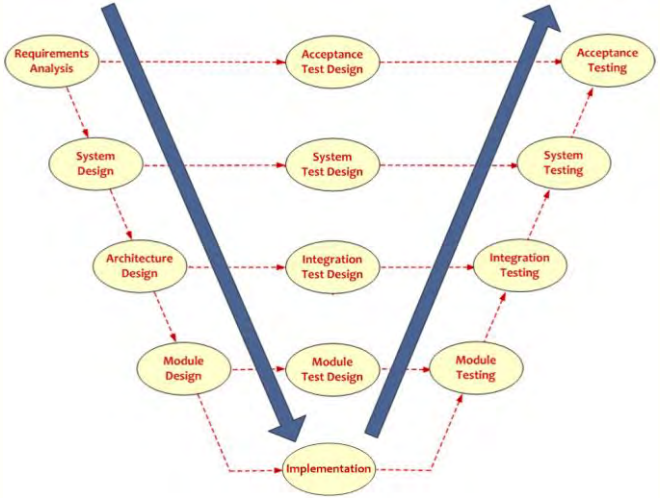
Delivering
Quality On Time
the Right Result
at the Right Time

Quality On Time

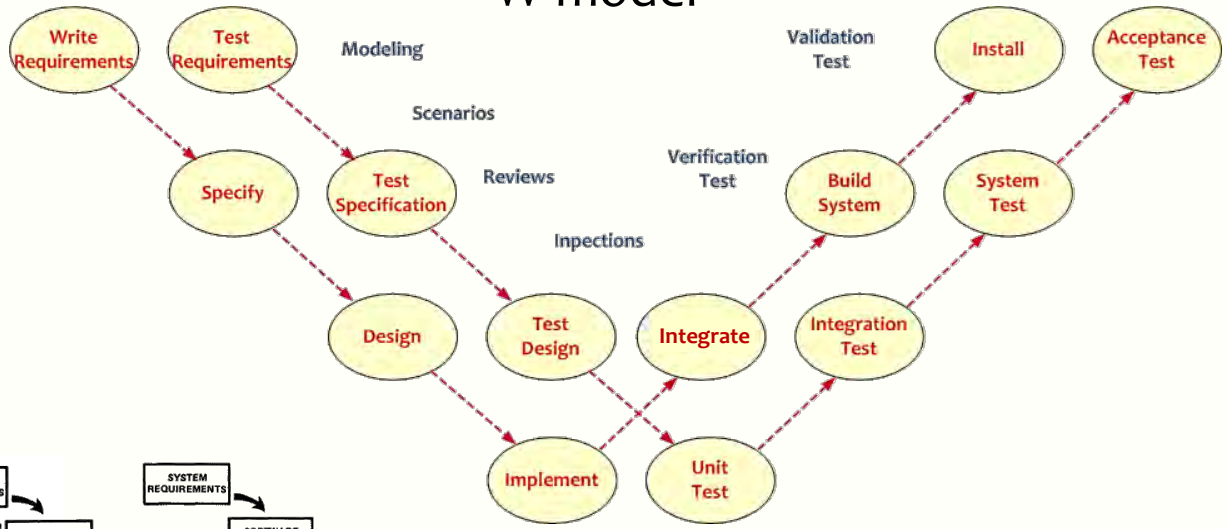
The most effective way of improving software productivity
and shortening project schedules
is to *reduce defect levels*

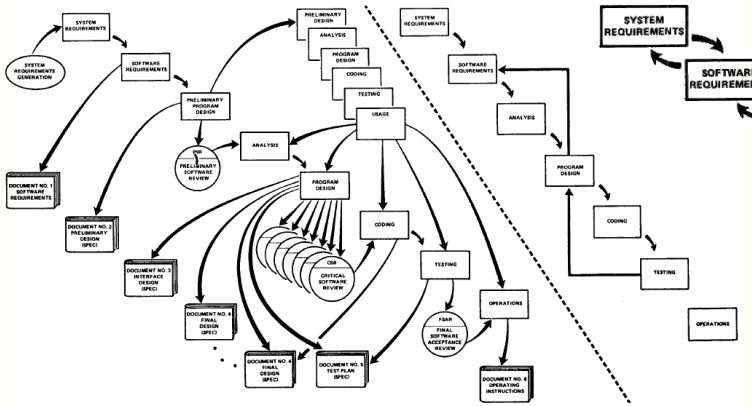
Capers Jones

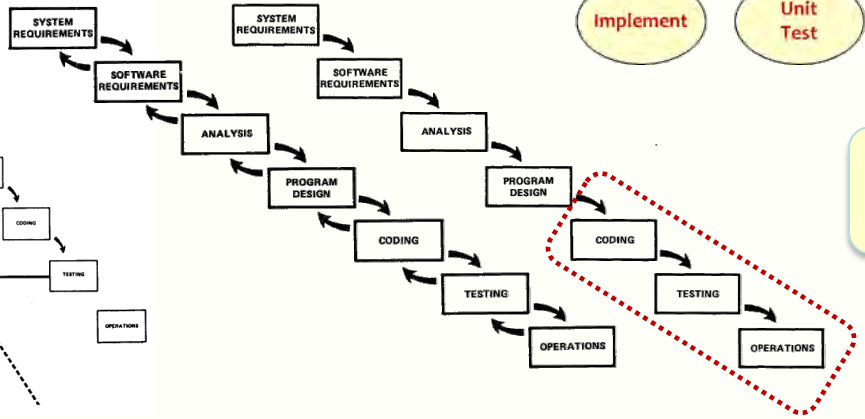
- Formal Inspections
  - prevented defects form occurring
  - removed defects that did occur
- Both Quality and On Time is improved if we work on reducing defect levels
- Why are testers so obsessed with finding bugs ?



W-model

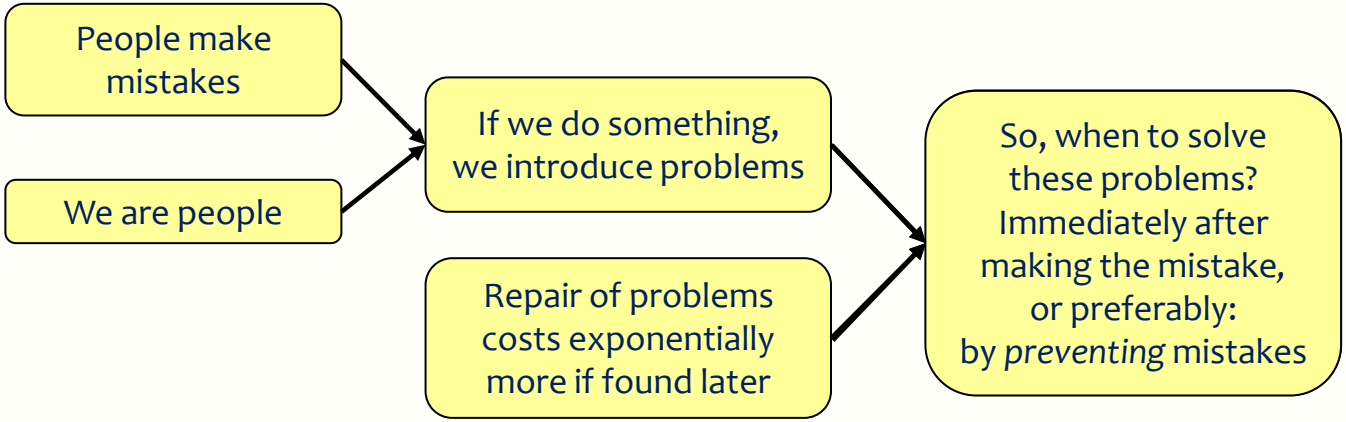






All models are wrong
Some are useful

We are people



Prevention costs much less than inject → find (?) → repair (?)

Dijkstra (1972)

It is a usual technique to make a program and then to test it

However:

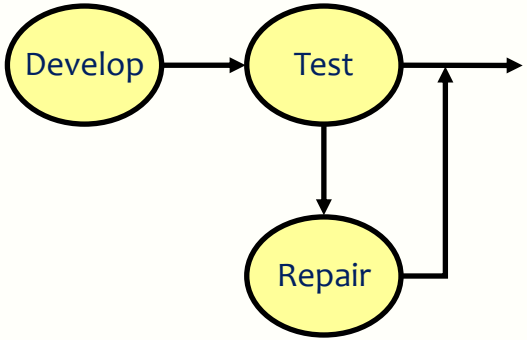
Program testing can be a very effective way to show the presence of defects

but it is hopelessly inadequate for showing their absence

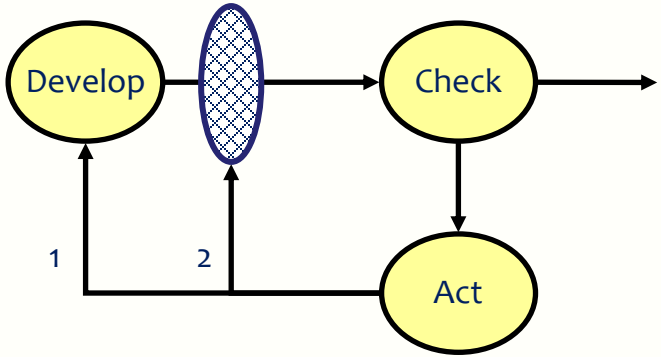
- Conventional testing:
  - Pursuing the very effective way to show the presence of defects
- The challenge is, however:
  - Making sure that there are no defects (development)
  - How to show their absence if they're not there (testing ?)

How can QA feed prevention ?

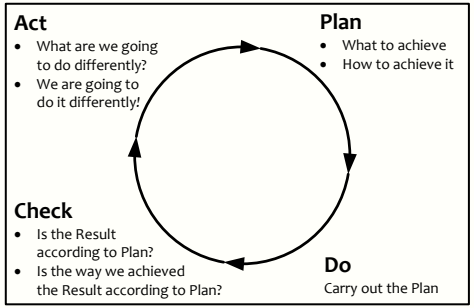
TestCon2018 - video:
How to move to Zero Defects
www.malotaux.eu/conferences



What we often see

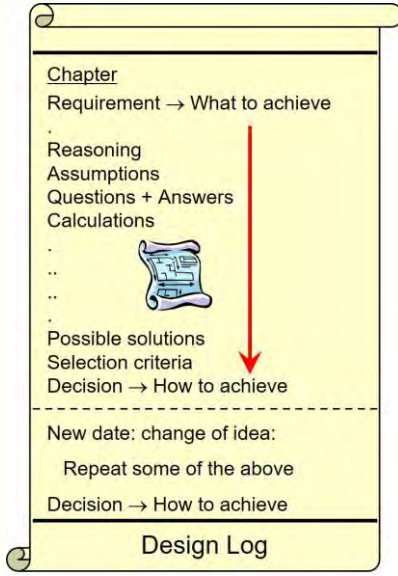


What we should expect

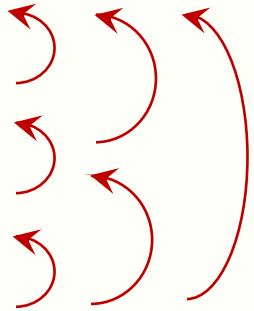


1. How can we prevent this ever happening again ?
2. Why did our earliest sieve not catch this defect ?

No proliferation of pollution



- Requirement
- Review
- Design
- Review
- Code
- Review

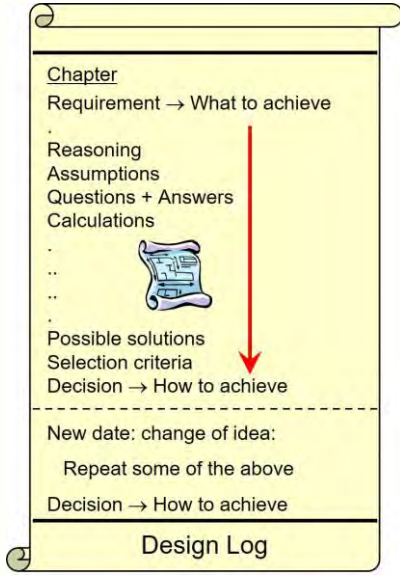


Iterate as needed

- Integration test (no questions, no issues)
- If issue in test: no Band-Aid: start all over again:
  Review: What's wrong with the design ?
- If there is no design: Reconstruct the design !
- QA to review the DesignLog for more efficiently helping the developers:
  Ask: "Can we see the DesignLog ?"

Cleanroom

Concept: DesignLog



- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - Chapter per subject
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Reasoning
  - Assumptions
  - Questions
  - Calculations
  - Possible solutions
  - Selection criteria
  - Choices:
    - If rejected: why?
    - If chosen: why?
- Implementation specification

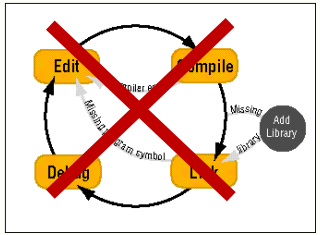
Case: In the pub

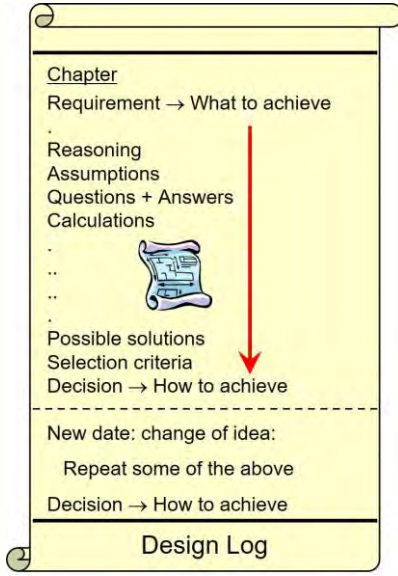
- James:

  Niels, this is Louise

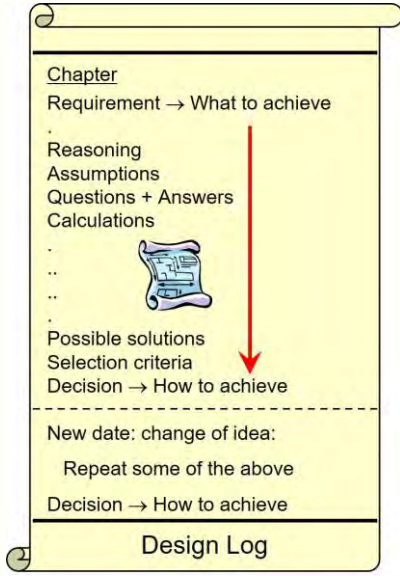
  Louise, this is Niels, who taught me about DesignLogging - Tell what happened !

- Louise:
  - We had only 7 days to finish some software
  - We were working hard, coding, testing, coding, testing
  - James said we should stop coding and go back to the design
  - "We don't have time !" - "We've only 7 days !"
  - James insisted
  - We designed, found the problem, corrected it, cleaned up the mess
  - Done in less than 7 days
  - Thank you!





What James told me later



- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a 'final' review, with another colleague
- Based on his expertise, the solution was completely reworked
- Actually, two features were delivered and deployed
  - One that was design and code reviewed had no issues after deployment
  - Other one, was the source of quite some defects
- From now on we use DesignLogs, to be reviewed before coding

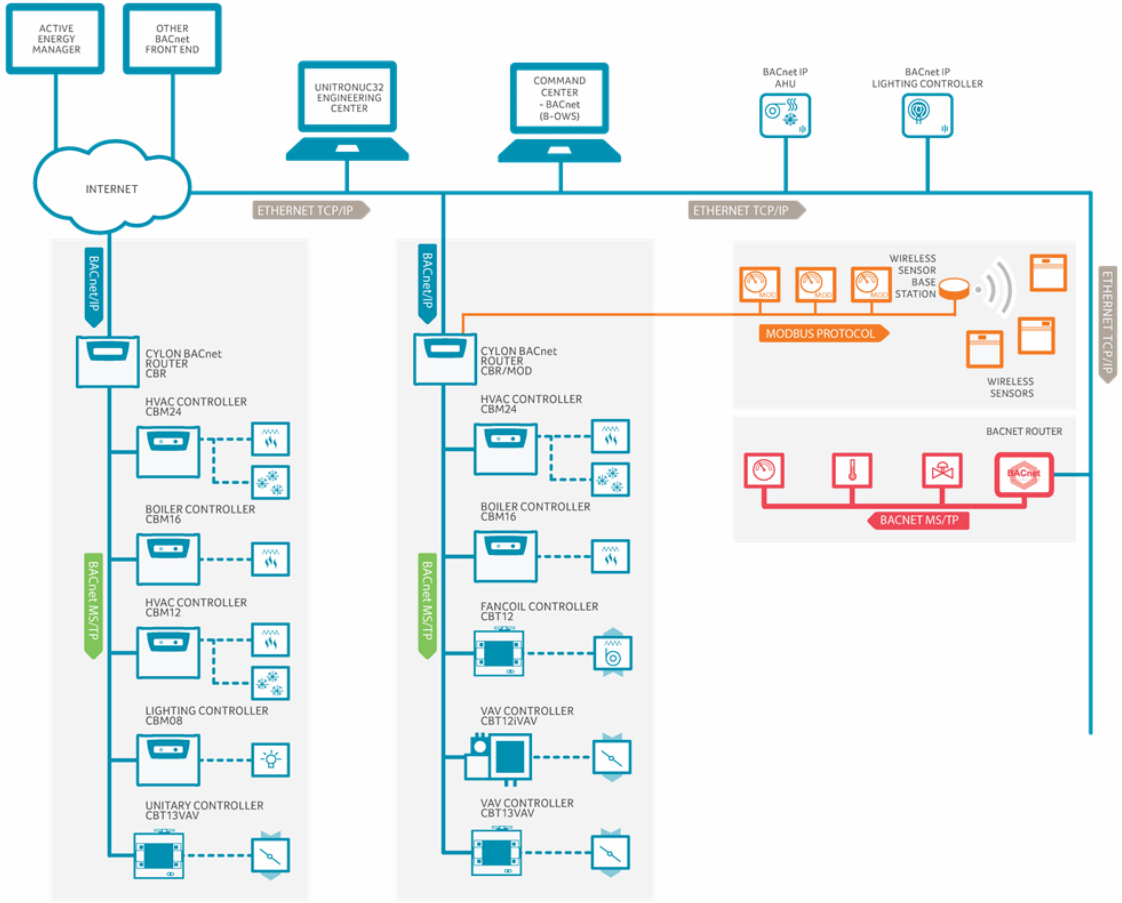
Using Inspection in various ways

- The review caused them not to implement a bad solution
- Problems of the original solution would not have been found out until much later
- The whole process allowed them to deliver well before the deadline rather than after

Case: Can you teach Inspections ?

- Short intro
- Are you regularly reviewing ?
- Let's do it: baseline
  - Take a document
  - Reproduce one page
  - Do review
  - No issues
- One rule ('source')
  - Many issues

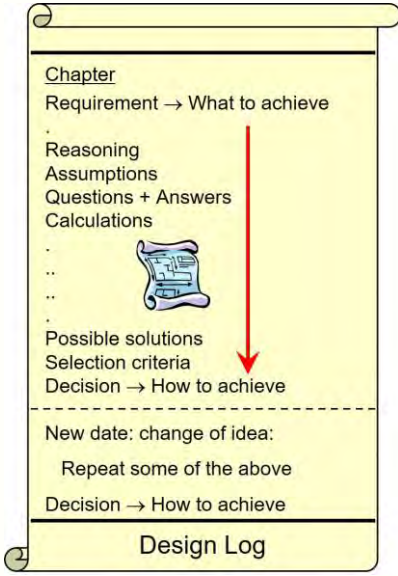
Datalog function improvement



'glitches'

Results

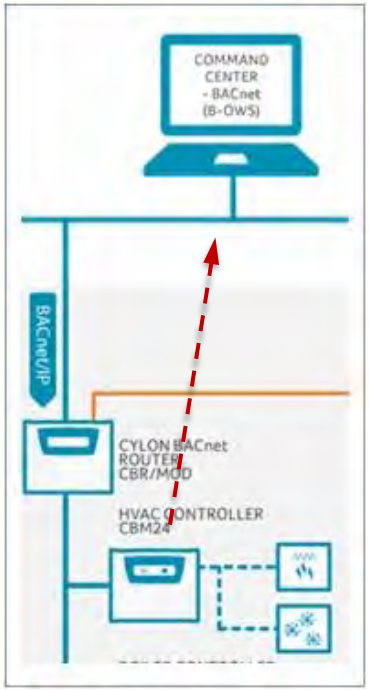
- No code until design-log is reviewed
- You’re delaying my project !
- Example
- Solution
- Thanks, you saved my project
- Did I do the same ?
- Telling people to change: resistance
- How to let people change themselves …



From his DesignLog

- A number of Firmware based methods of removing the glitches from the datalog reading process have been investigated

- but it has been decided to go with a mechanism implemented in the external system reading the datalog to remove the glitches



Using Inspection in various ways

- What if I would have suggested that the document wasn't right ?
- They showed themselves that it wasn't right
- Instead of proceeding further with Inspections, they first learnt what design is about

Case: City of Amsterdam

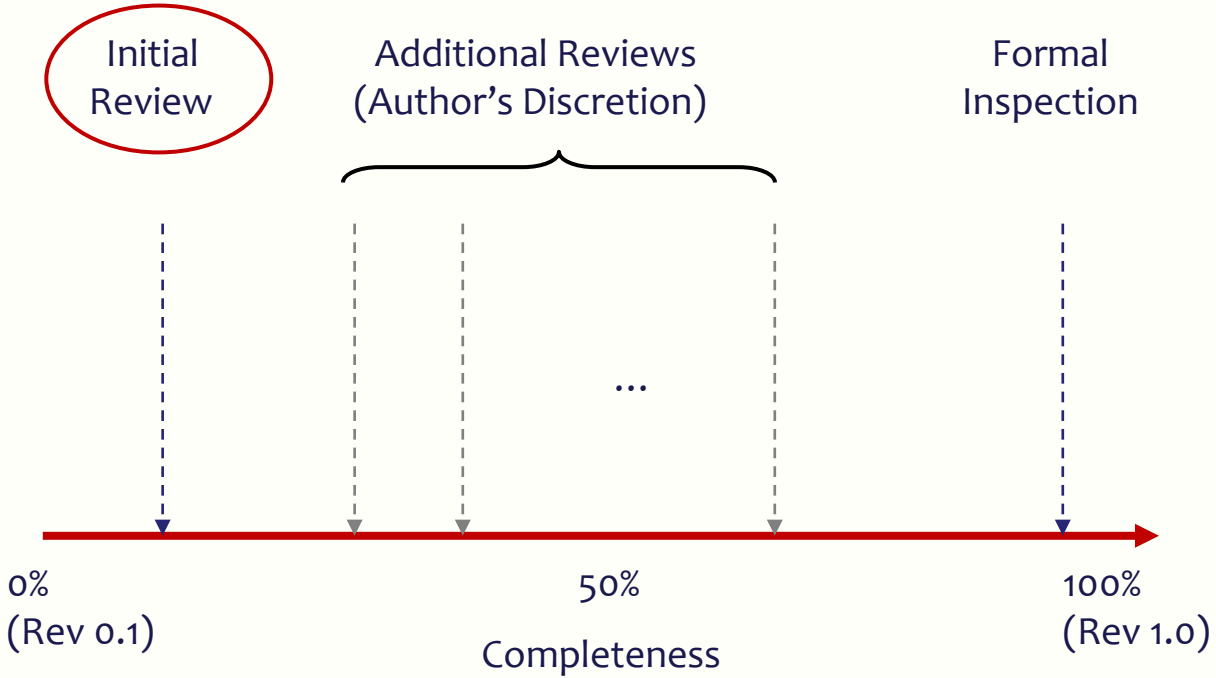
- Request for Proposal
- Can you teach Inspections ?
- You'll ditch the document after the course !
- Ha ha
- Of course they did
- Even redefined the whole project …

Using Inspection in various ways

- What if I would have suggested that the document wasn't right ?
- They showed themselves that it wasn't right
- They learnt so much about their project, that they redefined the project completely
- What would have happened if the Request for Proposal was sent out to suppliers
- Suppliers love unclear documents, allowing them to sell many more hours
- The City would have gotten a great solution for the wrong problem

Early Inspection

Prevention costs less than Repair



Case: Early Inspection on Requirements

- Large e-business application with 8 requirements authors
  - Each sent the first 8-10 requirements, of ~100 requirements per author
    (table format, about 2 requirements per page including all data)
  - Initial reviews completed within a few hours of submission
  - Authors integrated the suggestions and corrections, then continued to work
  - Some authors chose additional reviews, others did not
  - Inspection performed on document to assess final quality level

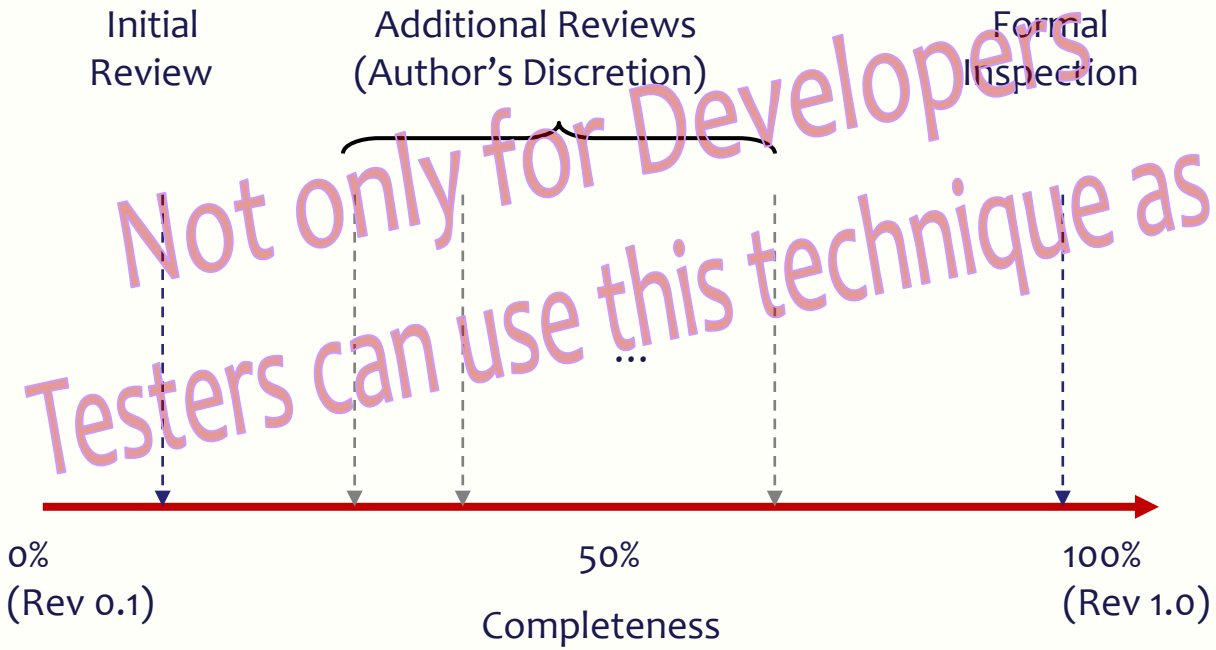
Results

Average major defects per requirement in initial review	8
Average major defects per requirement in final document	3

- Time investment: 26 hr
  - 12 hours in initial review (1.5 hrs per author)
  - About 8 hours in additional reviews
  - 6 hours in final inspection (2 hrs, 2 checkers, plus prep and debrief)
- Major defects prevented: 5 per requirement in ~750 total
- Saved 5 x 750 x 10 hr = 37500 hr / 3 = 12500 x $50 = $625000

Early Inspection

Prevention costs less than Repair



Not only for Developers

Testers can use this technique as well !

Case: Test Cases

- A tester's improvement writing successive test plans
  - Early Inspection used on an existing project to improve test plan quality
  - Test plan nearly "complete", so we *simulated* Early Inspection
  - First round: inspected 6 randomly-selected test cases
  - Author notes systematic defects in the results, reworks the whole document accordingly (~32 hrs)
  - Second round: inspected 6 more test cases: quality vastly improved
  - Test plan exits the process and goes into production
  - The author goes on to write another test plan

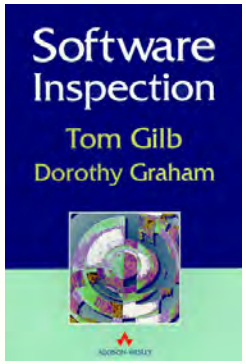
Results

First round	6 major defects per test case
Second round	0.5 major defects per test case

- Time investment: 2 hours in initial review,
  36 hours total in final formal inspection, excluding rework
  (2 inspections, 4 hrs each, 4 checkers, plus preparation and debrief)
- Historically about 25% of all defects found by testing were closed as "functions as designed", still 2-4 hrs spent on each to find out
- This test plan yielded over 1100 software defects with only 1 defect (0.1 %) closed as "functions as designed"
- Time saved on the project: 500 - 1000 hrs (25% x 1100 x 2-4 hrs)

Defect Prevention in action: First inspection of this tester's next test plan:
0.2 major defects per test case

Optimum Checking Rate

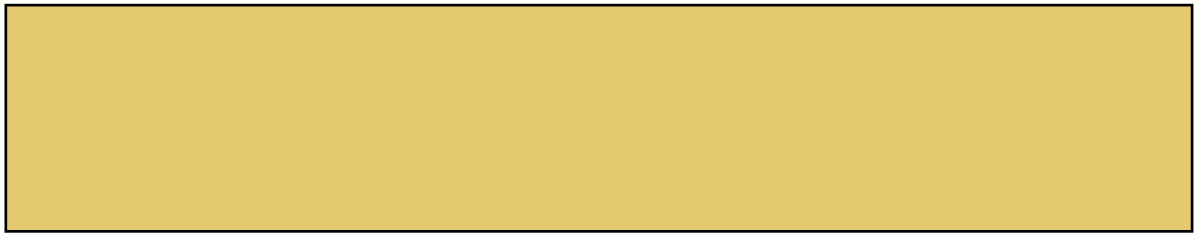


- The most effective individual speed for ‘checking a document against all related documents’ in page/hr
- Not ‘reading’ speed, but rather correlation speed
- Failure to use it, gives ‘bad estimate’ for ‘Remaining defects’

- 100~250 SLoC per hour
- 1 page of 300 words per hour (“logical page”)

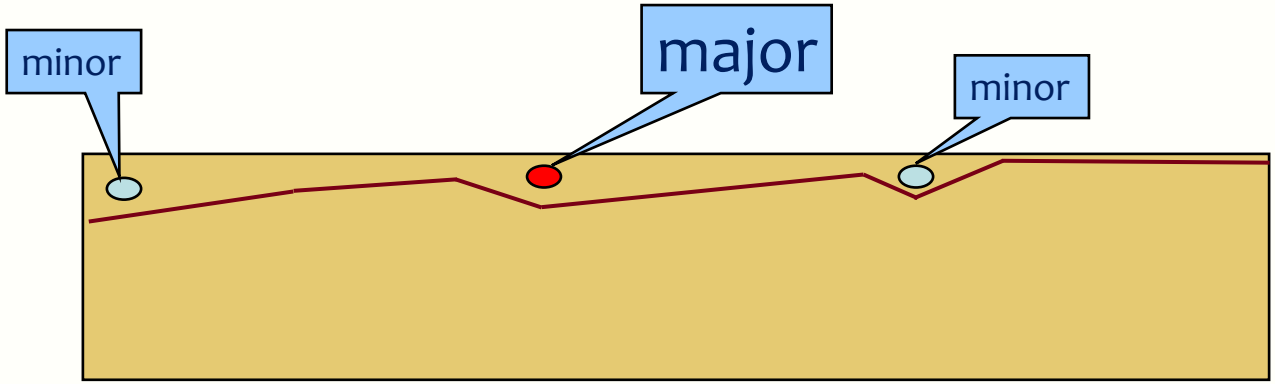
Here’s a document: review it

Ref. Dorothy Graham



Typical Review

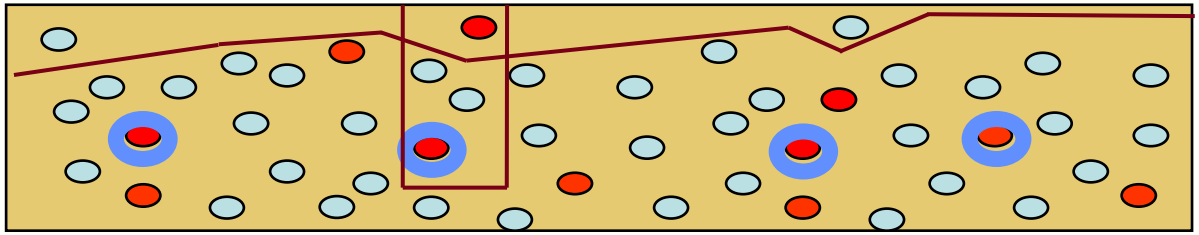
Ref. Dorothy Graham



- Find some defects, one Major
- Fix them
- Consider the document now corrected and OK ...

Taking a sample

Ref. Dorothy Graham



- Inspection can find deep-seated defects
- All of that type can be corrected
- Needs optimum checking rate

- In the above case we are clearly taking a sample
- In the "shallow" case we were also taking a sample, however, *we didn't feel it !*

Various ways

- DesignLog concept
- Case: In the pub: reviewing caused a design change
- Case: datalog: reviewing caused a concept change
- Case: reviewing caused ditching the document, and redefining the project completely
- Early Inspection
  - Case: Preventing requirements defects saved half a million
  - Case: Test case review vastly improved test case design
- Optimum checking rate to find deep seated defects

Reviews and Inspections used in various ways

Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/conferences