expo:QA Madrid 2010
16 November 2010

Niels Malotaux

How to Improve the Result of Reviews and Inspections

N R Malotaux - Consultancy
The Netherlands
tel +31-30-2288868
fax +31-30-2288869
niels@malotaux.nl
www.malotaux.nl

Niels Malotaux

Niels Malotaux is an independent Project Coach and expert in optimizing project performance. He has 35 years experience in designing electronic and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading his own systems design company. Since 1998 he devotes his expertise to helping projects to deliver Quality On Time: accurately predicting what will be done when, delivering what the customer needs, when he needs it, to enable customer success. He also teaches testers how not to delay delivery. To this effect, Niels developed an approach to effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached well over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, India, Ireland, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less in the practice of real projects.

Niels puts development teams on the Quality On Time track and coaches them to stay there and deliver their quality software or systems on time, without overtime, without the need for excuses. Practical methods are developed, used, taught and continually optimized for:

- Evolutionary Project Management (Evo)
- Requirements Engineering and Management
- Reviews and Inspections

Within a few weeks of turning a development project into an Evo project, the team has control and can tell the customer when the required features will all be done, or which features will be done at a certain date. Niels enjoys greatly the moments of enlightenment experienced by his clients when they find out that they can do it, that they are really in control, for the first time in their lives.

expo:QA '10

How to Improve the Result of Reviews and Inspections

Niels Malotaux

expo:QA '10

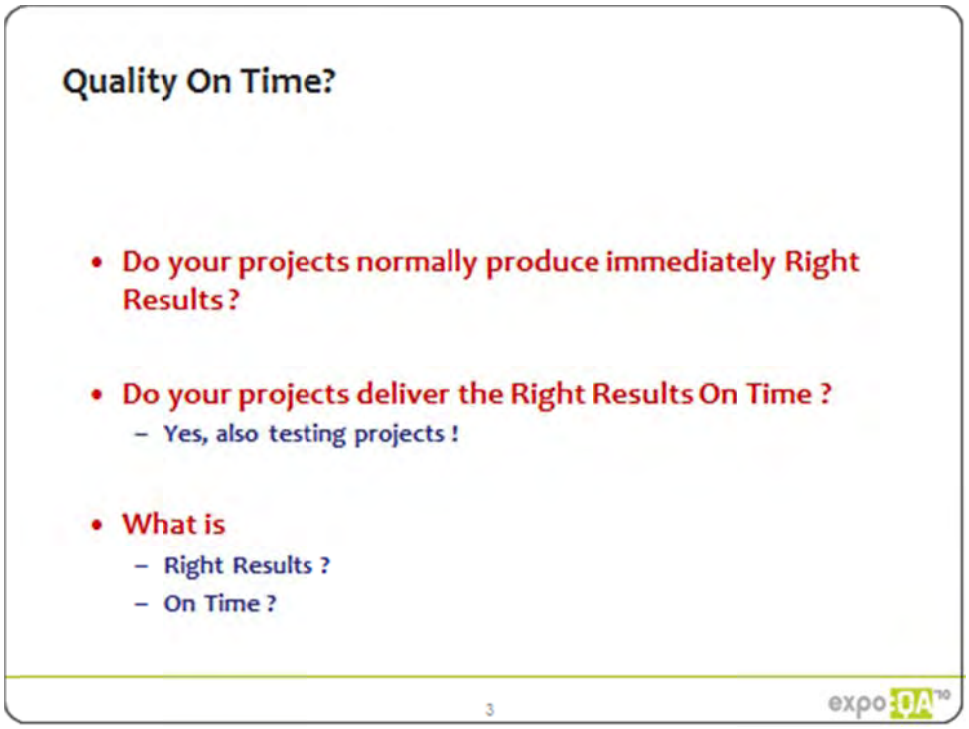
Niels Malotaux

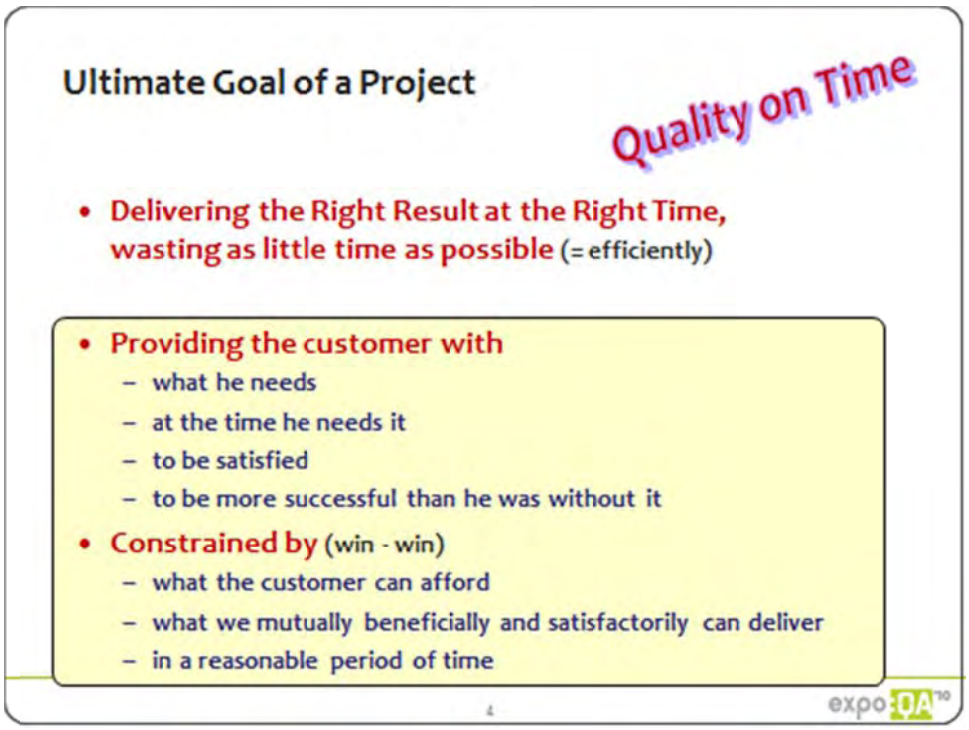
a recent design

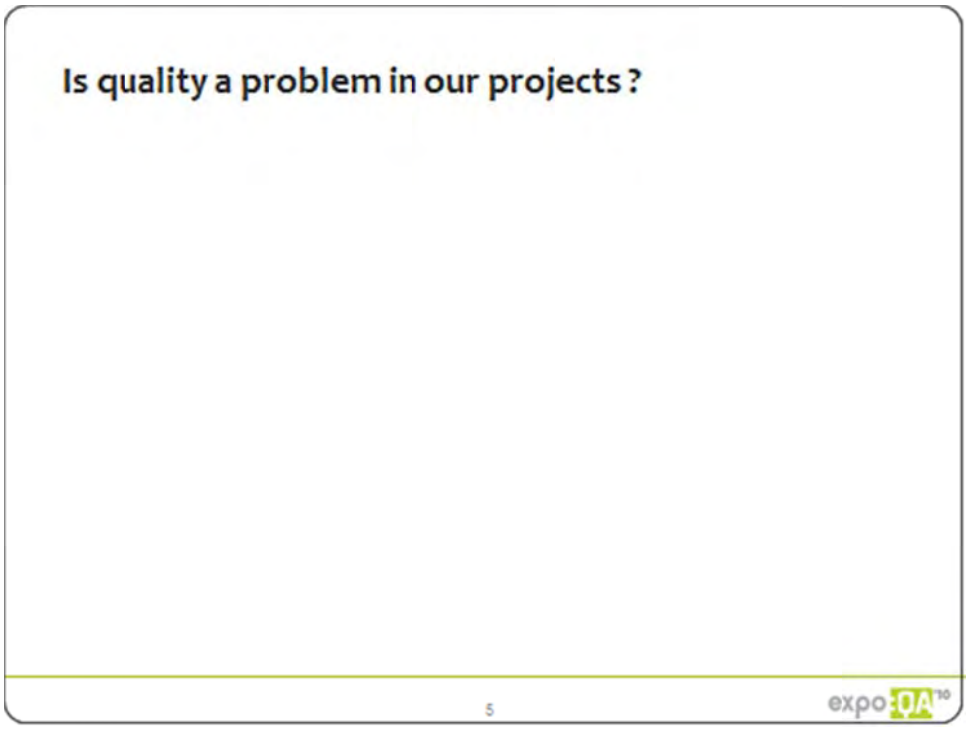
- Project Coach
  - Evolutionary Project Management (Evo)
  - Requirements Engineering
  - Reviews and Inspections
  - Reliable Embedded Systems
- More than 30 years Embedded Systems Design experience
- Expert in helping projects to be successful

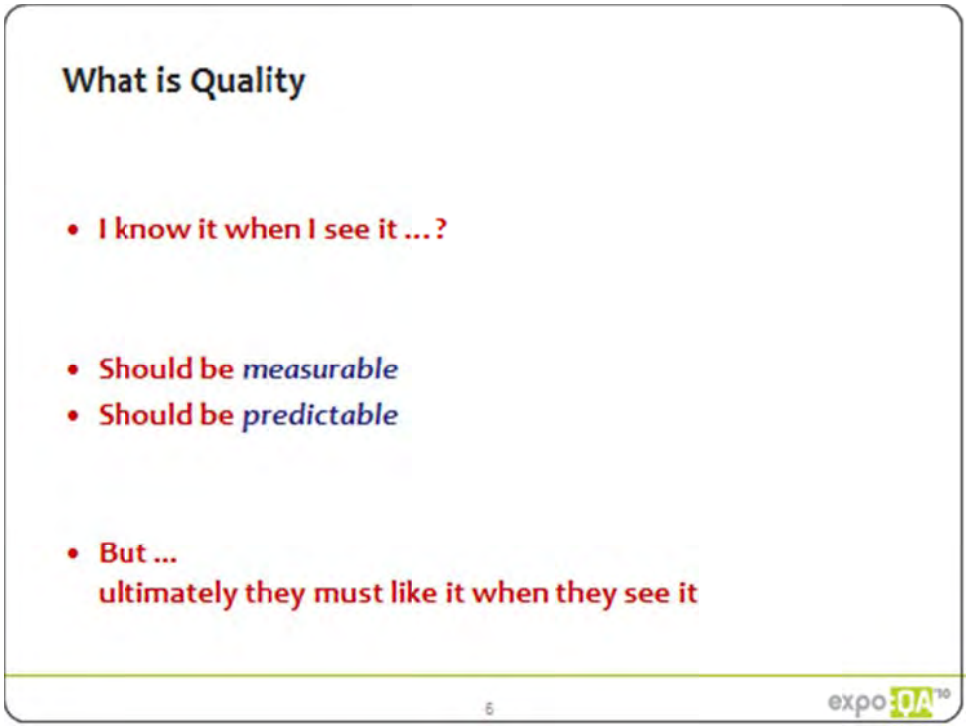
2

expo:QA '10


Quality On Time?
• Do your projects normally produce immediately Right Results?
• Do your projects deliver the Right Results On Time ?
– Yes, also testing projects !
• What is
– Right Results ?
– On Time ?
3
expo:QA


Ultimate Goal of a Project
Quality on Time
• Delivering the Right Result at the Right Time, wasting as little time as possible (= efficiently)
• Providing the customer with
– what he needs
– at the time he needs it
– to be satisfied
– to be more successful than he was without it
• Constrained by (win - win)
– what the customer can afford
– what we mutually beneficially and satisfactorily can deliver
– in a reasonable period of time
4
expo:QA


Is quality a problem in our projects?
5
expo:QA

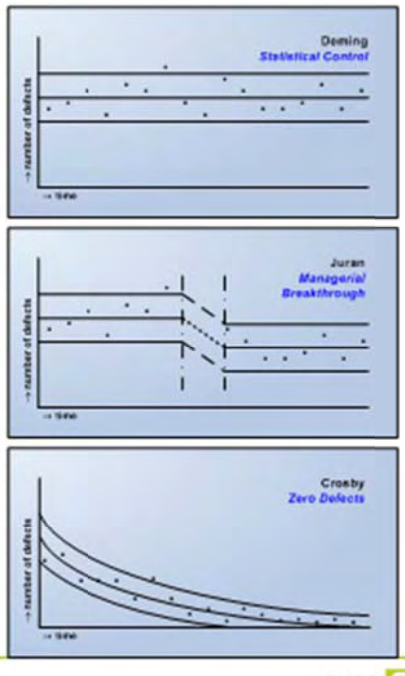

What is Quality
• I know it when I see it ...?
• Should be measurable
• Should be predictable
• But ...
ultimately they must like it when they see it
6
expo:QA

Quality guru's

- Shewhart - Economic Control of Quality 1930
- Deming - Japan 1950, Out of the crisis 1986
- Juran - Japan 1954, Quality handbook 1951
- Crosby - Zero Defects 1961, Quality is Free 1979
- Imai - Kaizen 1986, Gemba Kaizen 1997

7 expo:QA 10

Deming - Juran - Crosby



8 expo:QA 10

Absolutes of Quality

- Conformance to requirements
- Obtained through prevention
- Performance standard is zero defects
- Measured by the price of non-conformance (PONC)

Philip Crosby, 1970

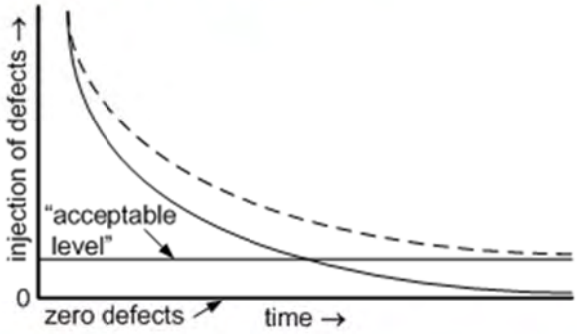
- The purpose is customer success (not customer satisfaction)

Added by Philip Crosby Associates, 2004

9 expo:QA'10

Is Zero Defects possible?

- Zero Defects is an asymptote



- When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately

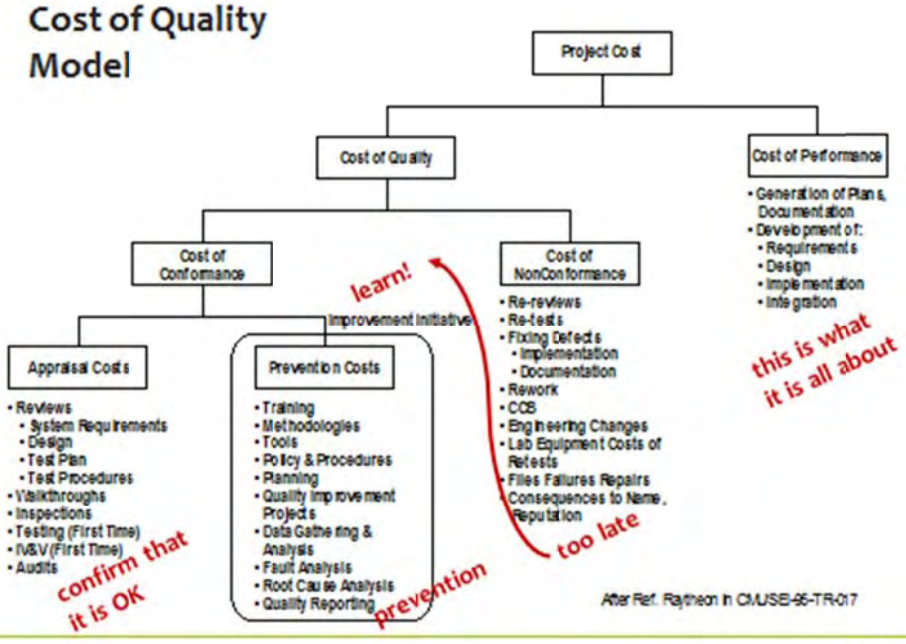
10 expo:QA'10

Attitude

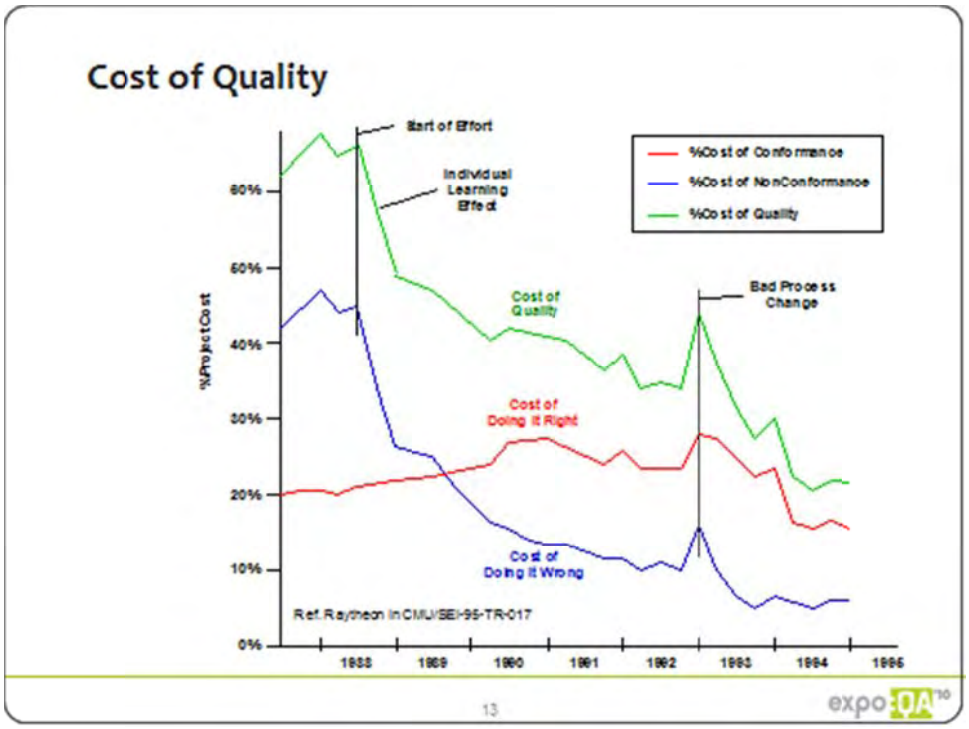
- As long as we think Zero Defects is impossible, we will keep producing defects
- From now on, we don't want to make mistakes any more
- We feel the failure (if we don't feel failure, we don't learn)
- If we deliver a result, we are sure it is OK and we'll be highly surprised when there proves to be a defect after all
- We do what we can to improve (continuous improvement)

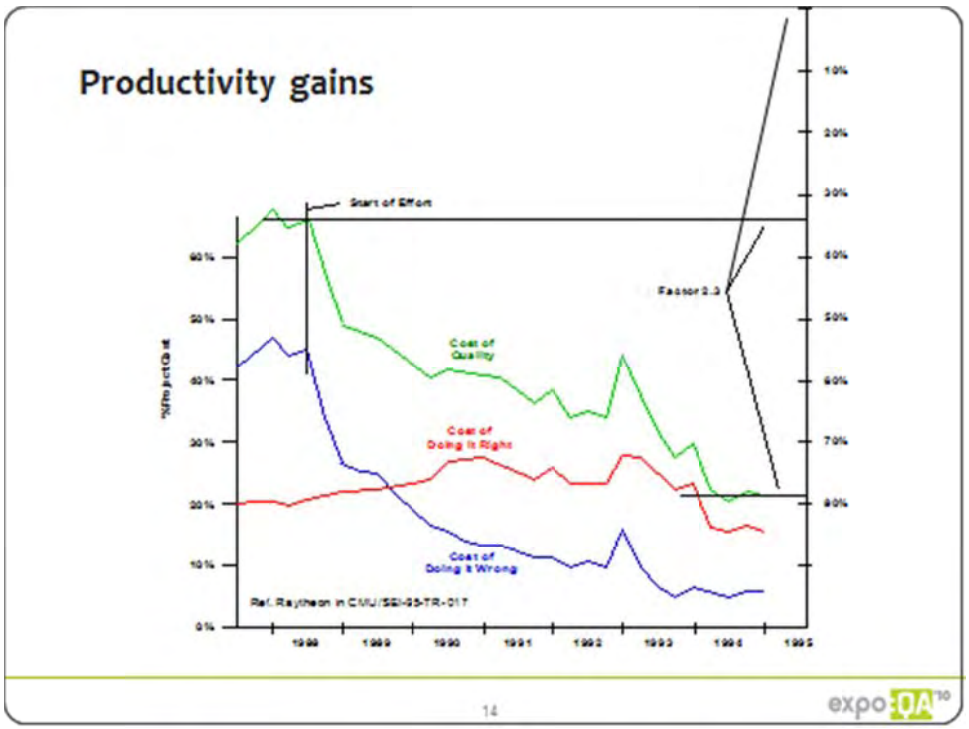
11 expo:QA

Cost of Quality Model



12 expo:QA


Cost of Quality
Start of Effort
Individual Learning Effect
%Cost of Conformance
%Cost of NonConformance
%Cost of Quality
Cost of Quality
Bad Process Change
Cost of Doing it Right
Cost of Doing it Wrong
%ProjectCost
60%
50%
40%
30%
20%
10%
0%
1988
1989
1990
1991
1992
1993
1994
1995
Ref. Raytheon in CMU/SEI-95-TR-017
13
expo:QA


Productivity gains
Start of Effort
Factor 2.3
Cost of Quality
Cost of Doing it Right
Cost of Doing it Wrong
%ProjectCost
Ref. Raytheon in CMU/SEI-95-TR-017
14
expo:QA

The Problem

- Still too many defects experienced by users

Apparently

- Still too many defects generated by developers
- Still too many defects remain undiscovered

However,

- There is a lot of knowledge how to reduce the generation and proliferation of defects

How much of your project will be spent on finding and fixing defects?

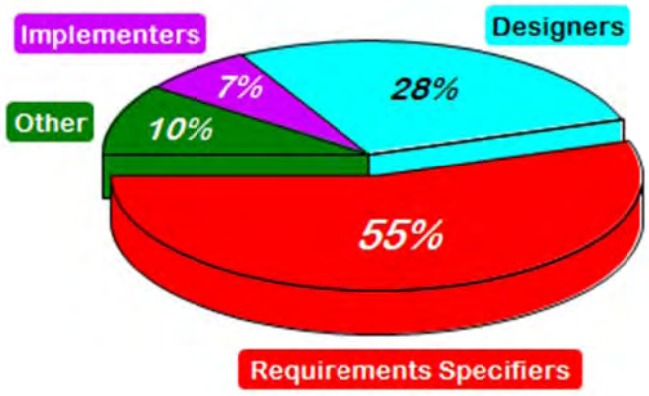
There is a large budget to do something about it:

??

- Some 50% of project time is consumed by all kinds of testing and repairing
- About 50% of developed software is never used
- Over 50% of delivered software is never used

15 expo:QA10

Typical Defect Injectors (cost breakdown)



After Bender Associates, 1996

16 expo:QA10

All we have to do ...

- A defect is
  the cause of a problem experienced by the users
- Making the customer more successful implies
  no defects
- All we have to do is delivering results without defects
- Do we?

- Is being late a defect?
- Is a failed experiment a defect?

17 expo:QA'10

Debugging ? ? ?

18 expo:QA'10

Bugs or Defects ?

- A design does not have bugs, it has *defects*
- Defects do not *emerge*
- People make errors, causing defects, causing problems
- Changing a requirement causes a lot of defects

19 expo:QA10

The process of defect injection

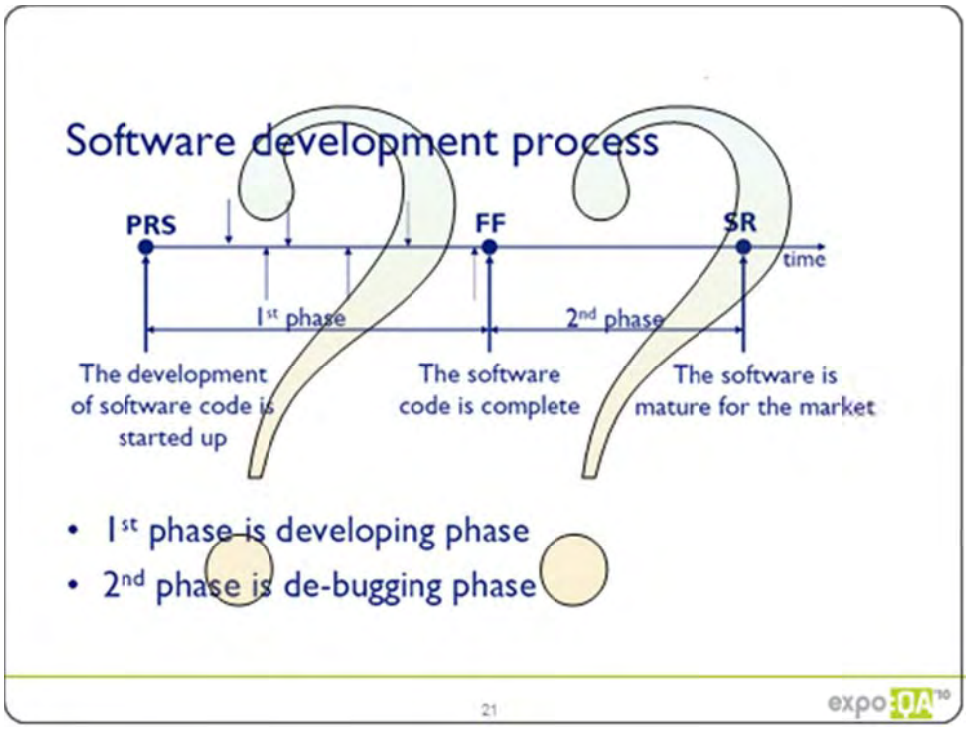
Conventional software development:

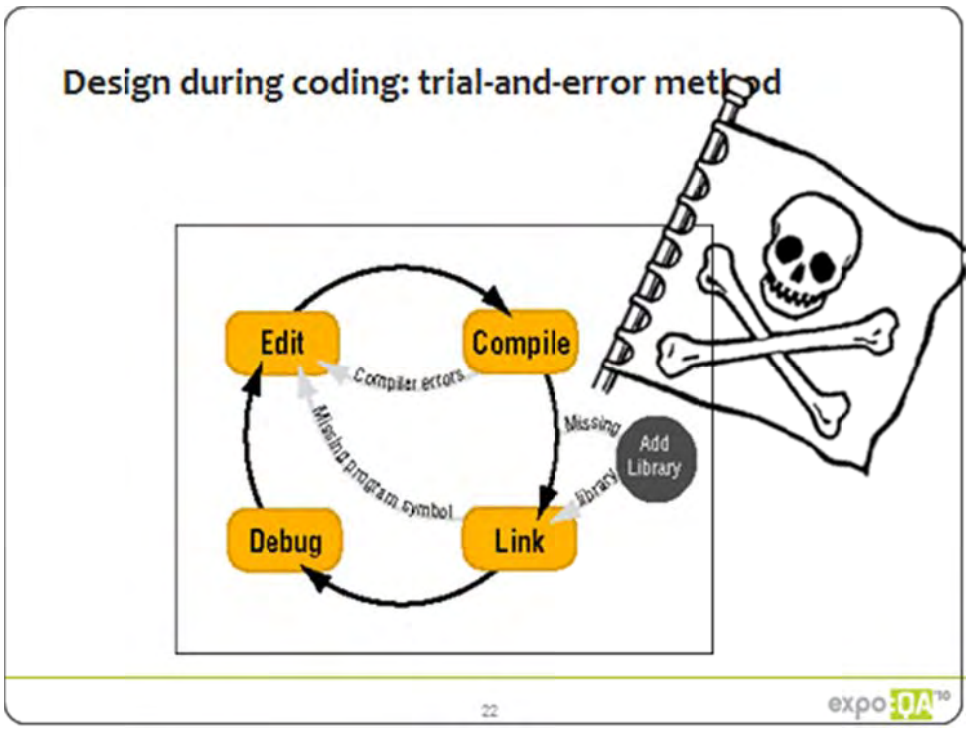
1. Development phase: inject bugs
2. Debugging or Testing phase: find bugs and fix bugs

Can't we do better ?

Does anybody mind ?

20 expo:QA10


Software development process
PRS
FF
SR
time
1st phase
2nd phase
The development of software code is started up
The software code is complete
The software is mature for the market
• 1st phase is developing phase
• 2nd phase is de-bugging phase
21
expo:QA


Design during coding: trial-and-error method
Edit
Compile
Compiler errors
Missing program symbol
Missing library
Add Library
Debug
Link
22
expo:QA

Bugs are so important, are they really?

- "Software without bugs is impossible"
- Bugs are counted
- We try to predict the number of bugs we will find
- It is suspect if we don't find the expected number
- Bugs are normal
- What would we do if there were no bugs any more?

➢ As long as we keep putting bugs in the center of the testing focus, there will be bugs

23 expo:QA

Defects found are symptoms of deeper lying problems

Repairing defects creates risks:

- Repair is done under pressure
- We think the problem is solved
- We introduce scars
- We keep repeating the same problems

→ Do Root Cause Analysis and make sure it never happens again

24 expo:QA

Defects typically overlooked

- **Functions that won't be used** (superfluous requirements)
  - Why to repair defects in the implementation of these requirements ?
  - The only defect is that it has been implemented
- **Nice things** (not checked, not paid for)
  Shouldn't be there in the first place
- **Missing quality levels** (should have been in requirements)
  Checking the implementation of the documented requirements won't help
- **Missing constraints** (should have been in requirements)
  Product could be illegal
- **Unnecessary constraints** (not required)
  What would testing say about these ?

25 expo:QA

Ways to achieve better quality ?

- Hope ??
- Test ?
- Debug ??
- Review ?
- Walkthrough ?
- Inspection ?

Prevention !!

26 expo:QA

Let's do a lot of testing !

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 bugs but it is hopelessly inadequate for showing their absence

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

27 expo:QA

Software testing

- 50% of defects is not found in test
- Repair of defects causes defects
- A compiler finds only 90% of syntax errors
- Of 4 defects:
  2 found by compiler, 1 at test and 1 by the customer
- How much %% of your projects is used for test, finding, repair, re-test?
- How much %% of the defects did you find and really fix?
- Now many of the defects *will be repeated*?

28 expo:QA

Who is the Customer of Testing and QA ?

Deming:

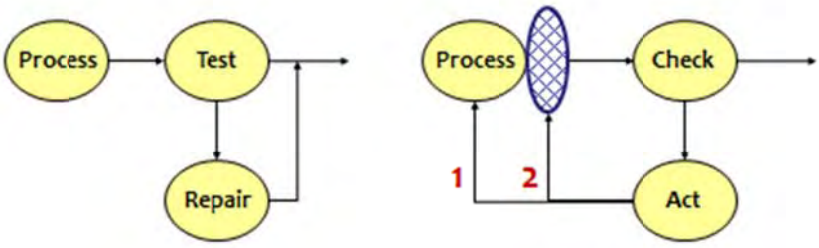
Quality comes not from testing, but from improvement of the development process. Testing does not improve quality, nor guarantee quality. It's too late. The quality, good or bad, is already in the product. You cannot test quality into a product.

→ Development is the customer

- Testing helps developers to become perfect
- Testing is a project to run alongside and synchronized to the development project
- Therefore, it must be organized like any other project

29 expo:QA'10

Testing is checking correctness



Usual Should be

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

30 expo:QA'10

Let's move

from

Fixation to Fix

to

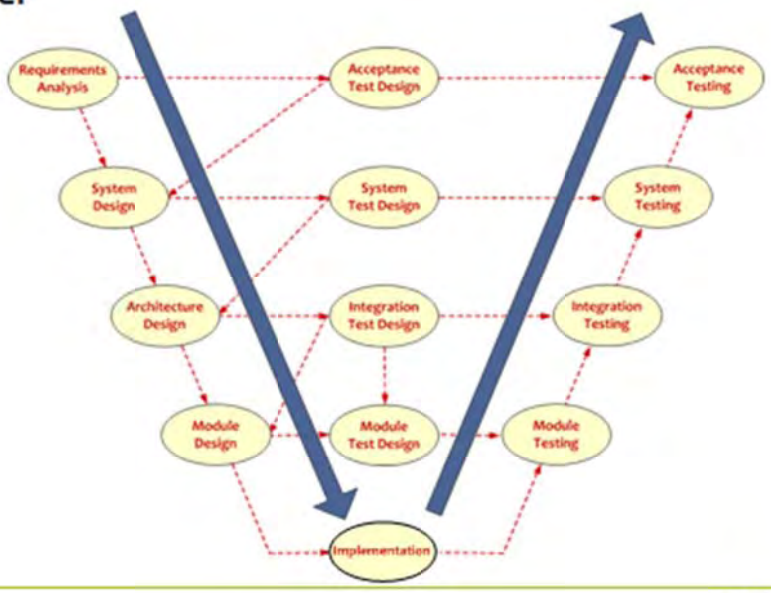
Attention to Prevention

- If we don't deal with the root, we will keep making the same mistakes over and over
- Toyota Production System: "Stop the Line"

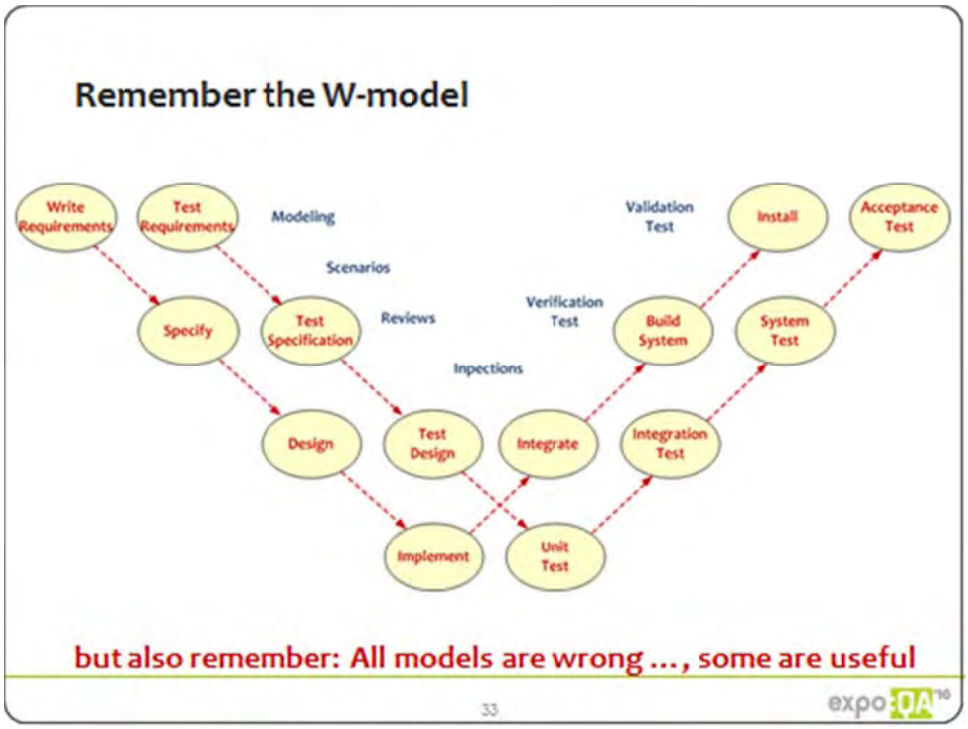
- Without feedback, we won't even know
- With quick feedback, we can put the repetition to a halt

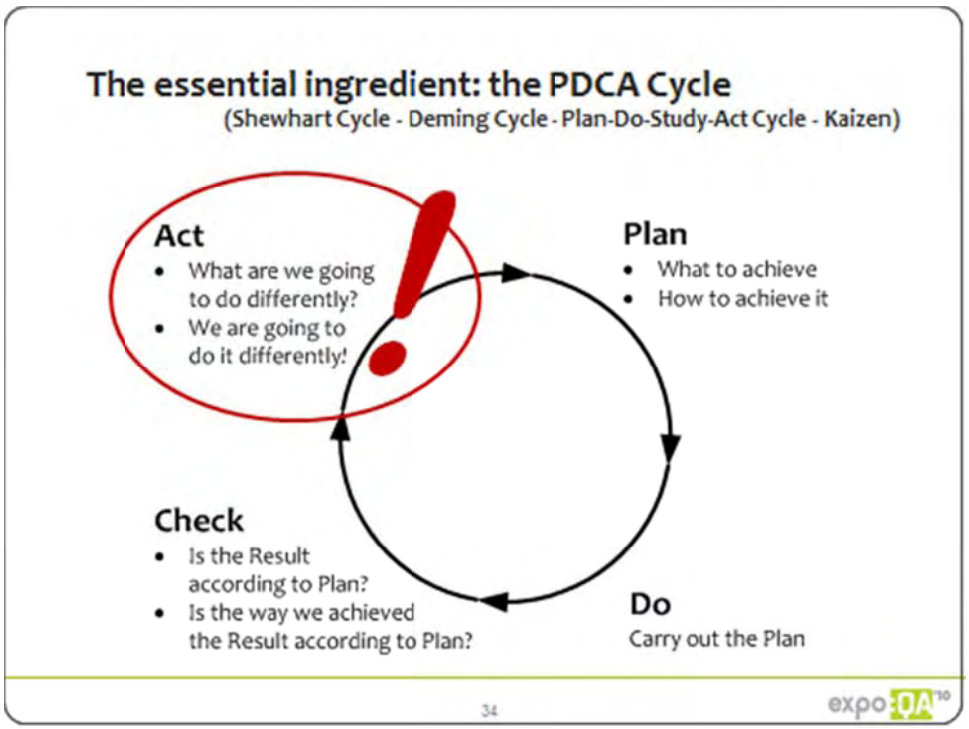
31

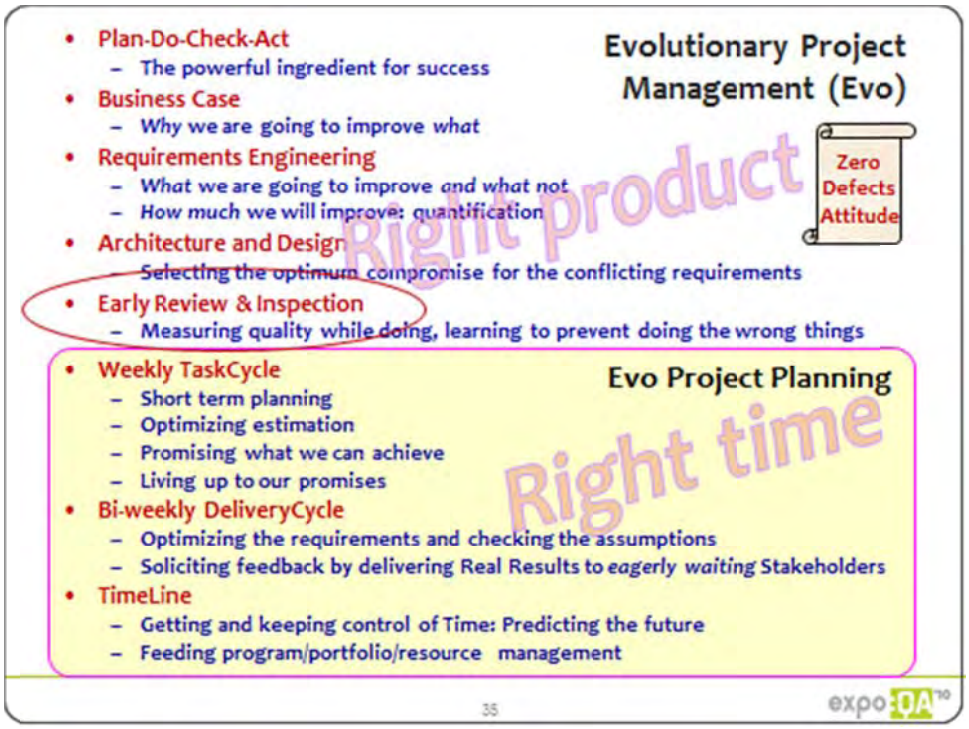
V-Model



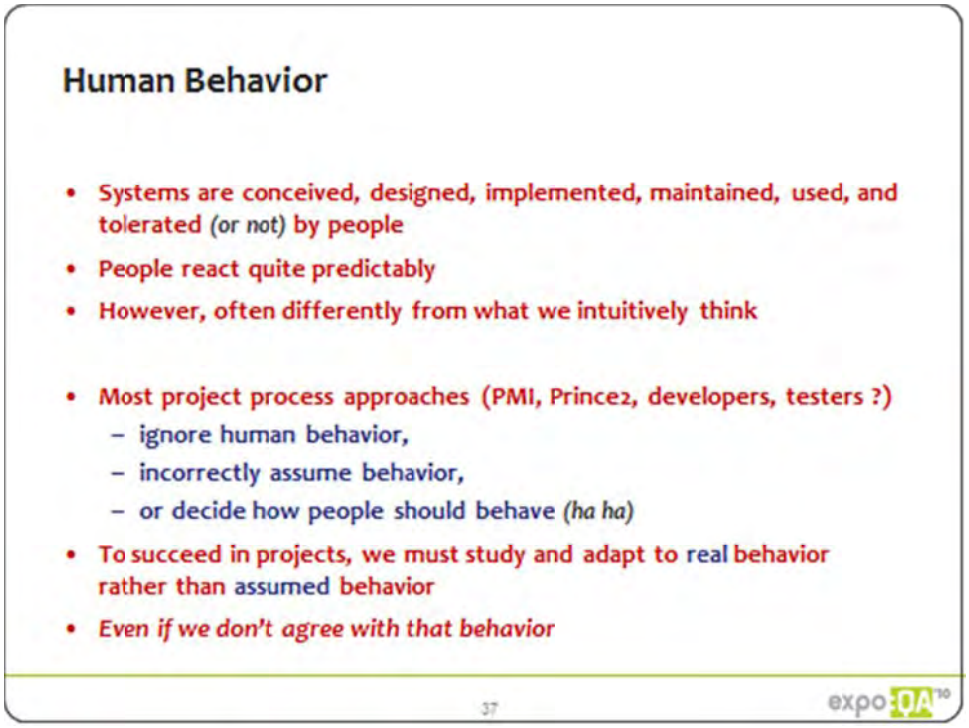
32


Remember the W-model
Write Requirements
Test Requirements
Modeling
Scenarios
Specify
Test Specification
Reviews
Design
Test Design
Inpections
Implement
Unit Test
Integrate
Verification Test
Build System
Integration Test
Validation Test
Install
System Test
Acceptance Test
but also remember: All models are wrong ..., some are useful
33
expo:QA

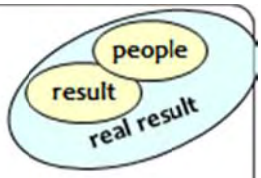

The essential ingredient: the PDCA Cycle
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)
Act
What are we going to do differently?
We are going to do it differently!
Plan
What to achieve
How to achieve it
Check
Is the Result according to Plan?
Is the way we achieved the Result according to Plan?
Do
Carry out the Plan
34
expo:QA


Evolutionary Project Management (Evo)
Plan-Do-Check-Act
The powerful ingredient for success
Business Case
Why we are going to improve what
Requirements Engineering
What we are going to improve and what not
How much we will improve: quantification
Architecture and Design
Selecting the optimum compromise for the conflicting requirements
Early Review & Inspection
Measuring quality while doing, learning to prevent doing the wrong things
Right product
Zero Defects Attitude
Evo Project Planning
Weekly TaskCycle
Short term planning
Optimizing estimation
Promising what we can achieve
Living up to our promises
Bi-weekly DeliveryCycle
Optimizing the requirements and checking the assumptions
Soliciting feedback by delivering Real Results to eagerly waiting Stakeholders
TimeLine
Getting and keeping control of Time: Predicting the future
Feeding program/portfolio/resource management
Right time
35
expo:QA


Human Behavior
36
expo:QA


Human Behavior
• Systems are conceived, designed, implemented, maintained, used, and tolerated (or not) by people
• People react quite predictably
• However, often differently from what we intuitively think
• Most project process approaches (PMI, Prince2, developers, testers ?)
– ignore human behavior,
– incorrectly assume behavior,
– or decide how people should behave (ha ha)
• To succeed in projects, we must study and adapt to real behavior rather than assumed behavior
• Even if we don't agree with that behavior
37
expo:QA

Is Human Behavior a risk?



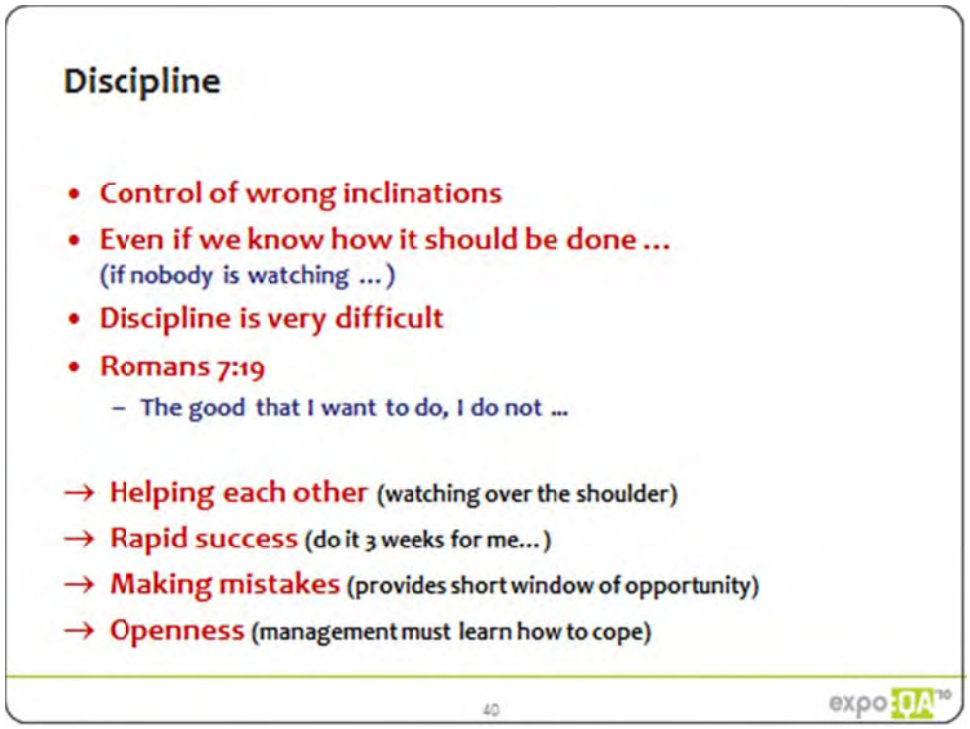
- Human behavior is a risk for the success of the system
  - When human behavior is incorrectly modelled in the system
  - Not because the humans are wrong
- Things that can go wrong
  - Customers not knowing well to describe what they really need
  - Users not understanding how to use or operate the system
  - Users using the system in unexpected ways
  - Incorrect modelling of human transfer functions within the system: ignorance of designers of systems engineers
- Actually, the humans aren't acting unpredictably
  - Because it happens again and again
  - Human error results from physiological and psychological limitations of humans

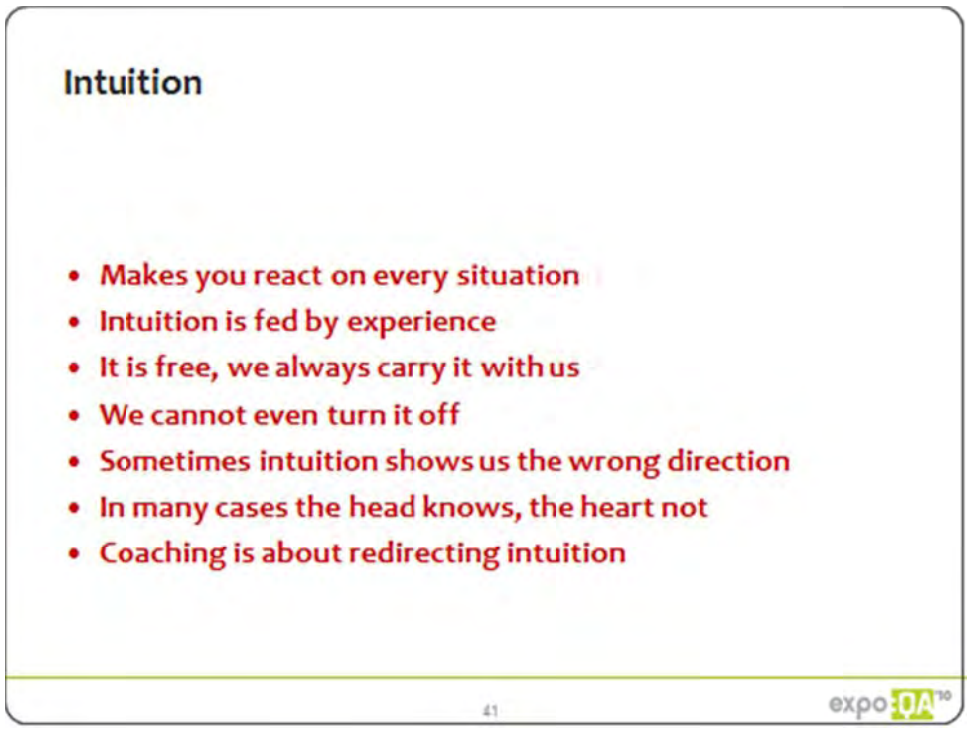
38 expo:QA10

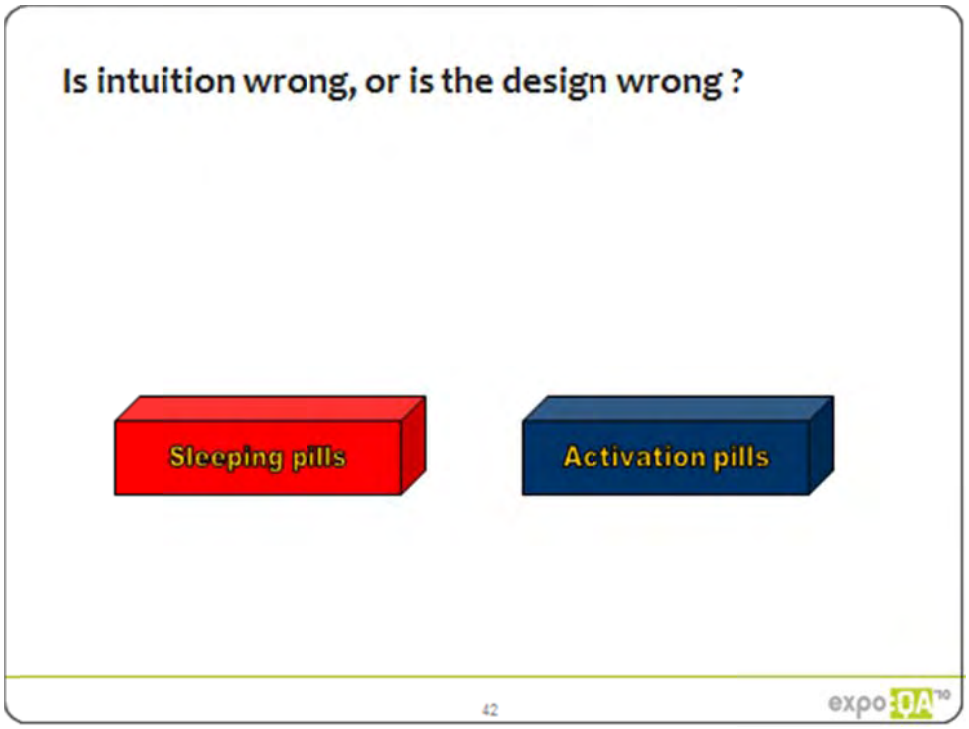
People responsible for success

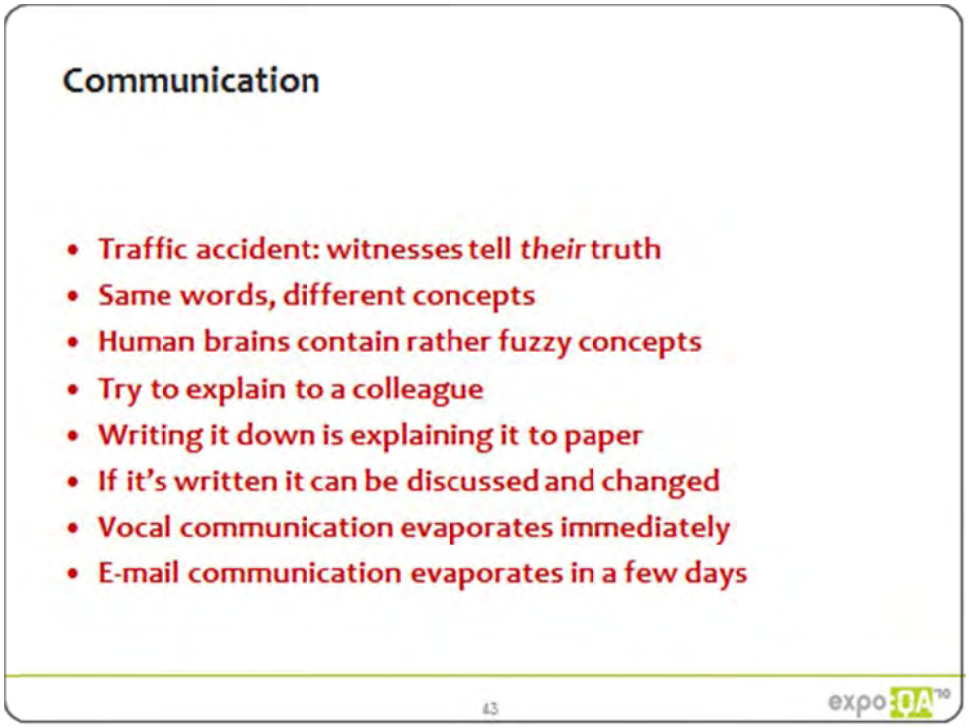
- *During* the project
  - Can still influence the performance of the project
  - First responsibility of the Project Manager
  - Actually responsibility of the whole development organization
- *After* the project, once the system is out there
  - No influence on the performance of the system any more
  - System must perform autonomously
  - So the performance must be there *by design*
  - Including appropriate interface with humans
  - Responsibility and required skill of Systems Engineers / Architects / Designers

39 expo:QA10


Discipline
• Control of wrong inclinations
• Even if we know how it should be done …
(if nobody is watching …)
• Discipline is very difficult
• Romans 7:19
– The good that I want to do, I do not …
→ Helping each other (watching over the shoulder)
→ Rapid success (do it 3 weeks for me…)
→ Making mistakes (provides short window of opportunity)
→ Openness (management must learn how to cope)
40
expo:QA


Intuition
• Makes you react on every situation
• Intuition is fed by experience
• It is free, we always carry it with us
• We cannot even turn it off
• Sometimes intuition shows us the wrong direction
• In many cases the head knows, the heart not
• Coaching is about redirecting intuition
41
expo:QA


Is intuition wrong, or is the design wrong ?
Sleeping pills
Activation pills
42
expo:QA


Communication
• Traffic accident: witnesses tell *their* truth
• Same words, different concepts
• Human brains contain rather fuzzy concepts
• Try to explain to a colleague
• Writing it down is explaining it to paper
• If it's written it can be discussed and changed
• Vocal communication evaporates immediately
• E-mail communication evaporates in a few days
43
expo:QA

Perception

- Quick, acute, and intuitive cognition (www.M-W.com)
- What people say and what they do is not always equal
- The head knows, but the heart decides
- Hidden emotions are often the drivers of behavior
- Customers who said they wanted lots of different ice cream flavours from which to choose, still tended to buy those that were fundamentally vanilla
- So, trying to find out what the real value to the customer is, can show many paradoxes
- Better not simply believe what they say: check!

44 expo:QA'10

People *like* change, if it's an improvement!

- People are not against change
- People (sub-consciously) don't like uncertainty
- Any project changes something and thus introduces uncertainty
- People can cope with uncertainty for a short time

45 expo:QA'10

Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures

- At the Fatal Day, any excuse is in vain: we *failed*
- Even if we "couldn't do anything about it"
- Failure is a very hard word. That's why we are using it !
- No pain, no gain
- We never say: "*You* failed", use: "*We* failed"
  - After all, we didn't help the person not to fail

46 expo:QA 10

We failed because of politics

- Good politics:
  - People decide differently based on different values
- Bad politics: hidden agenda's
  - Say this, mean that - often even unintentionally
  - Politics thrive by vagueness
  - Facts can make bad politics loose ground

- If you accepted the responsibility for the project, failure because of "politics" is just an excuse
- What did you really do about it ?

47 expo:QA 10

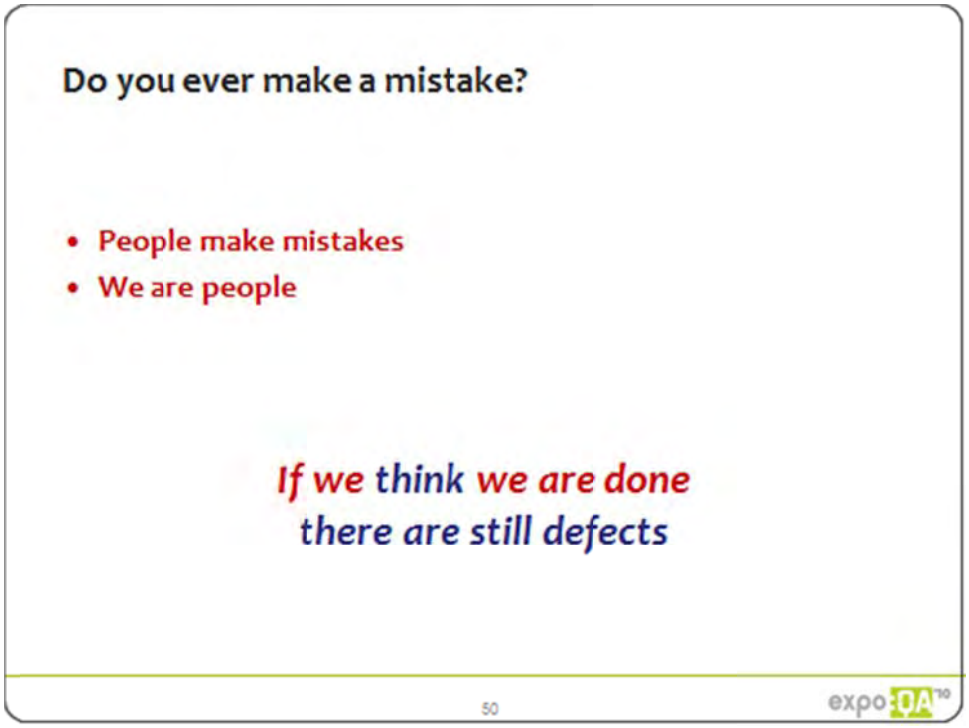
Culture

- It failed because of the existing culture

- Culture is the result of how people work together
- Culture can't be changed
- Culture *can* change
- By doing things differently

43


Reviews &
Inspections
49
expo:QA


Do you ever make a mistake?
• People make mistakes
• We are people
If we think we are done
there are still defects
50
expo:QA

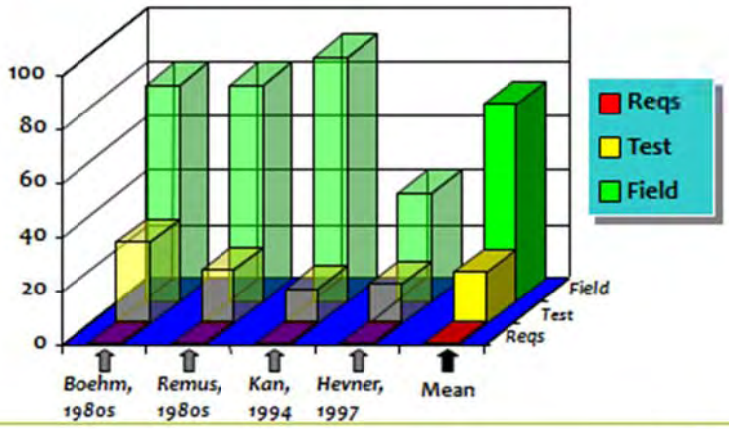
Costs of defects

The longer a defect stays in the system,
the more it costs to find and repair

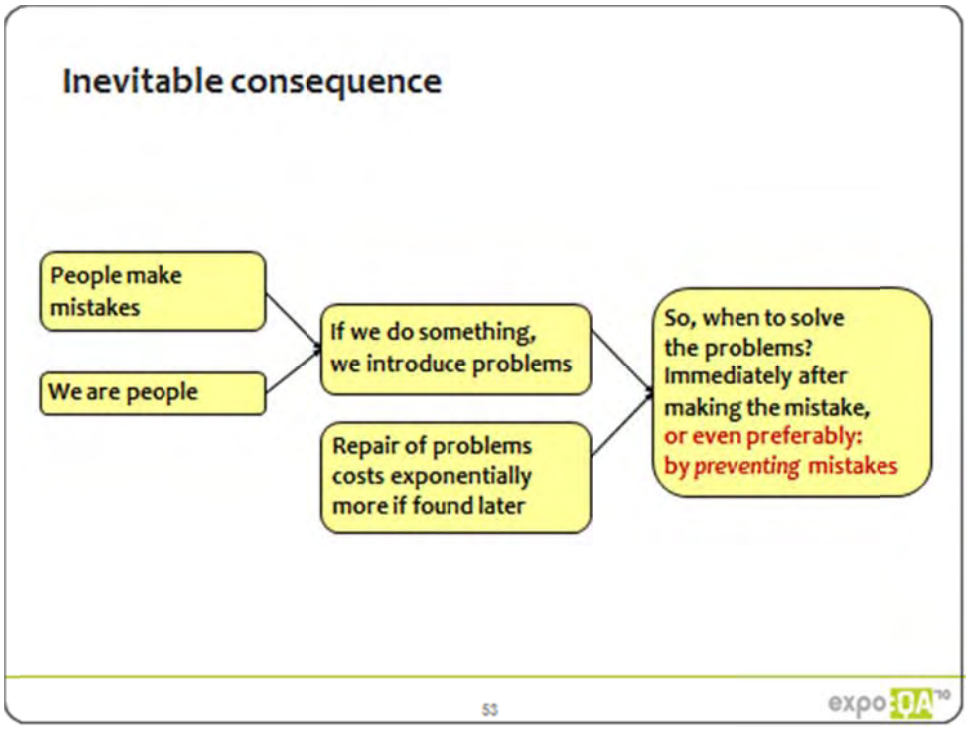
51 expo:QA

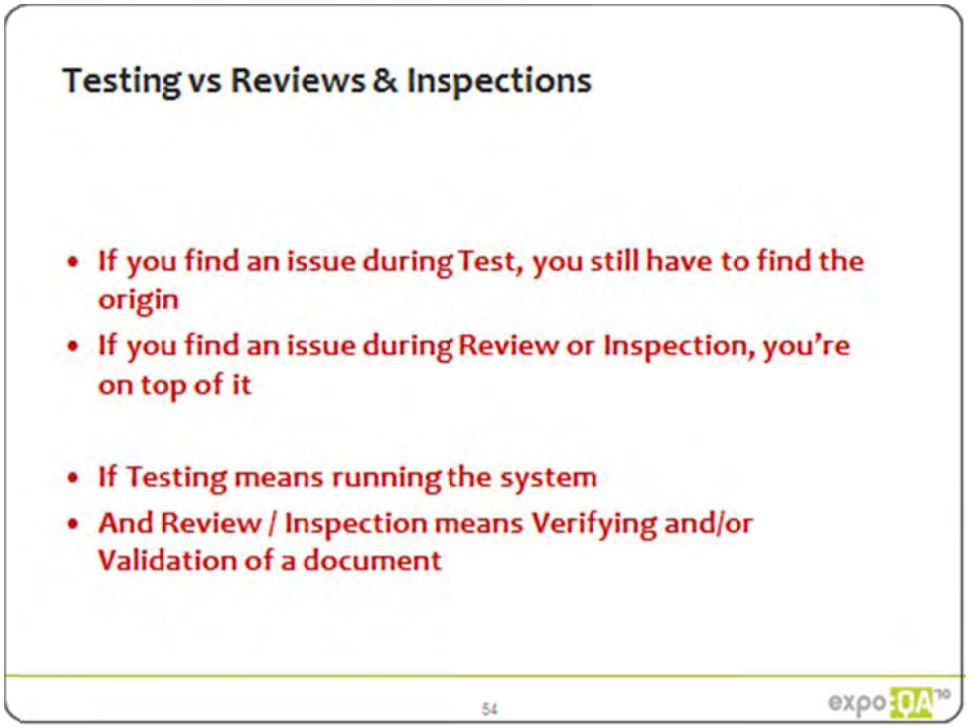
Cost of Requirements Defects

The longer a defect stays in the system,
the more it costs to repair



52 expo:QA


Inevitable consequence
People make mistakes
We are people
If we do something, we introduce problems
Repair of problems costs exponentially more if found later
So, when to solve the problems? Immediately after making the mistake, or even preferably: by preventing mistakes
53
expo:QA


Testing vs Reviews & Inspections
• If you find an issue during Test, you still have to find the origin
• If you find an issue during Review or Inspection, you're on top of it
• If Testing means running the system
• And Review / Inspection means Verifying and/or Validation of a document
54
expo:QA

Many types of Review to choose from

- Informal Review
- Pair Programming
- Technical Review
- Walkthrough
- Formal Inspection (Fagan type)
- Cleanroom Inspection
- Formal Inspection (Gilb/Graham type)
- Agile/Extreme/Lean/Early Inspection
- Gate Review
- Unit Test
- Debugging
- Test

55 expo:QA

Techniques

- Can you look at this ?
- Over the shoulder
- Pair Programming
- E-mail
- Tool
- On Screen
- Projector
- On Paper
- Formal process

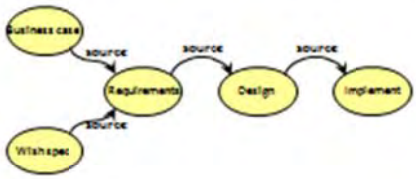
56 expo:QA

Formal Reviews (vs Ad-Hoc)

- Defined, repeatable process
- Measures effectiveness
- Continuous improvement
- Rules/checklists
- Feeds prevention process

57 expo:QA'10

Typical documents



• Wish specification	Thank you, nice input
• Business Case	Why are we doing it
• Requirements	What the project agrees to satisfy
• DesignLog	Selecting the 'optimum' compromise and how we arrived at this decision
• Specification	This is how we are going to implement it
• Implementation	Code, schematics, plans, procedures, hardware, documentation, training
• Process Log	Describing how and why we arrived at which current practices

58 expo:QA'10

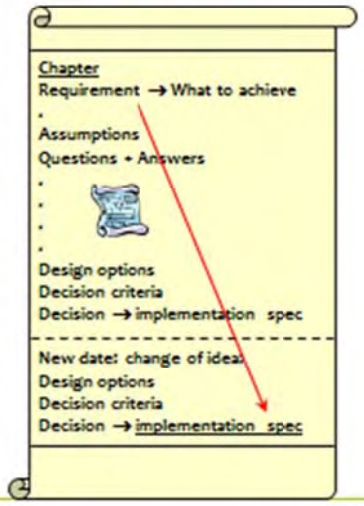
Does your Testing Project have all these documents ?

- Testing is a Project, just as any other project

59

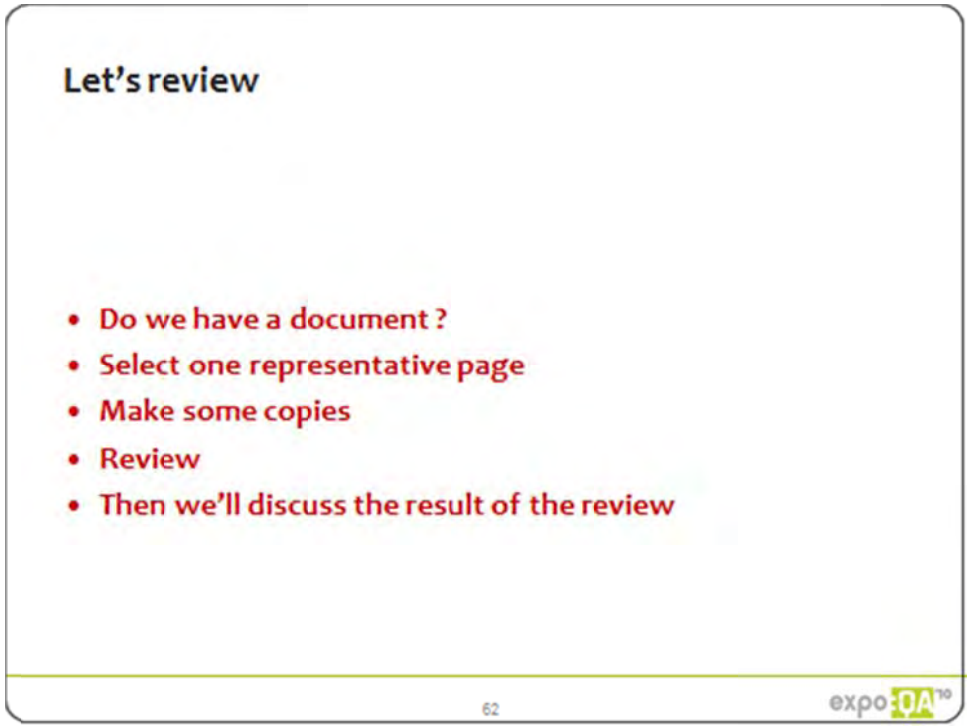
DesignLog (project level)

- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + reasoning:
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices
- Implementation



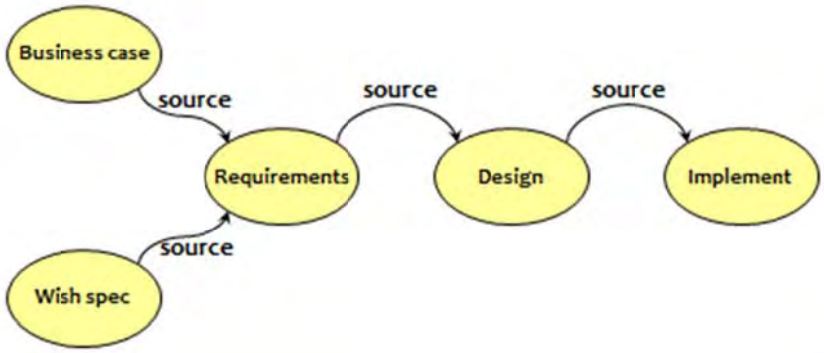
60


Did you ever do a Review?
61
expo:QA


Let's review
• Do we have a document?
• Select one representative page
• Make some copies
• Review
• Then we'll discuss the result of the review
62
expo:QA

Simple Rule for Reviews

"We don't review unless there is a source document"

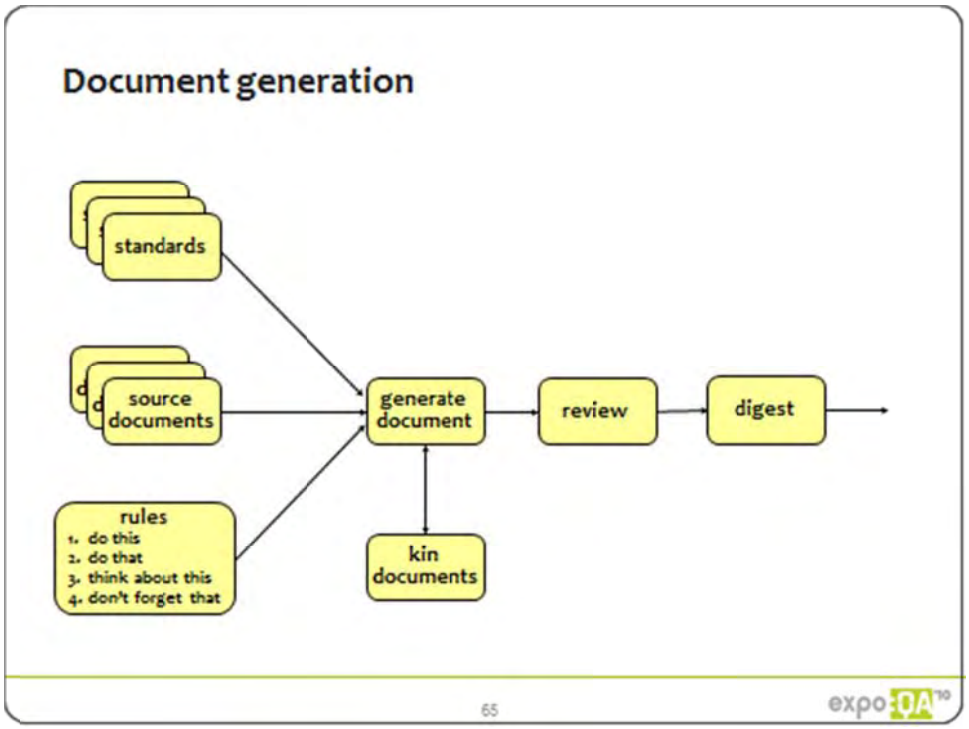


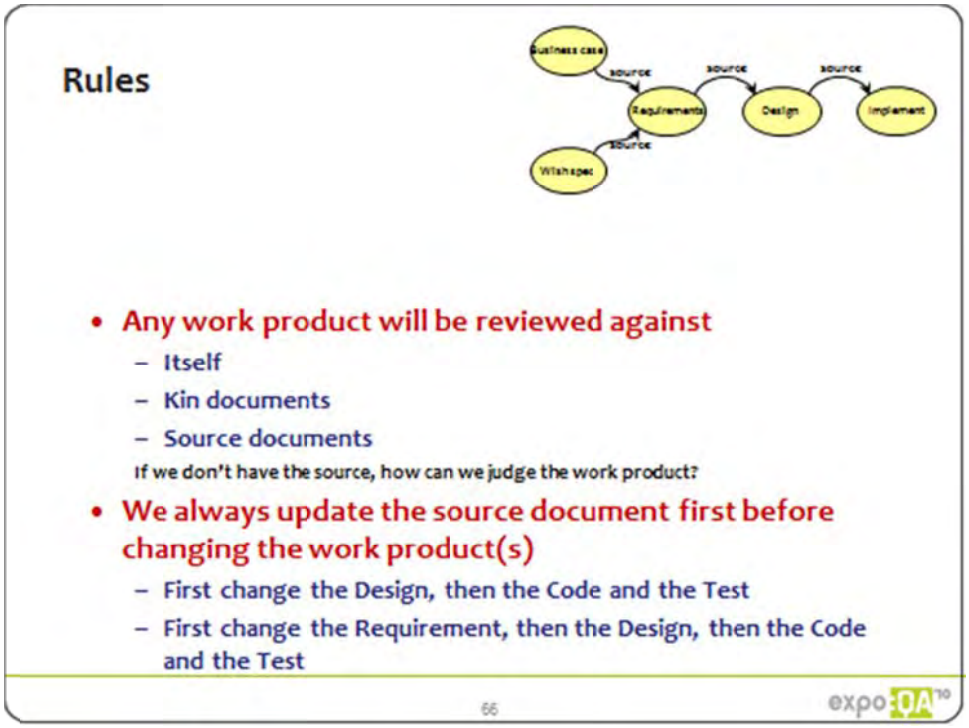
63

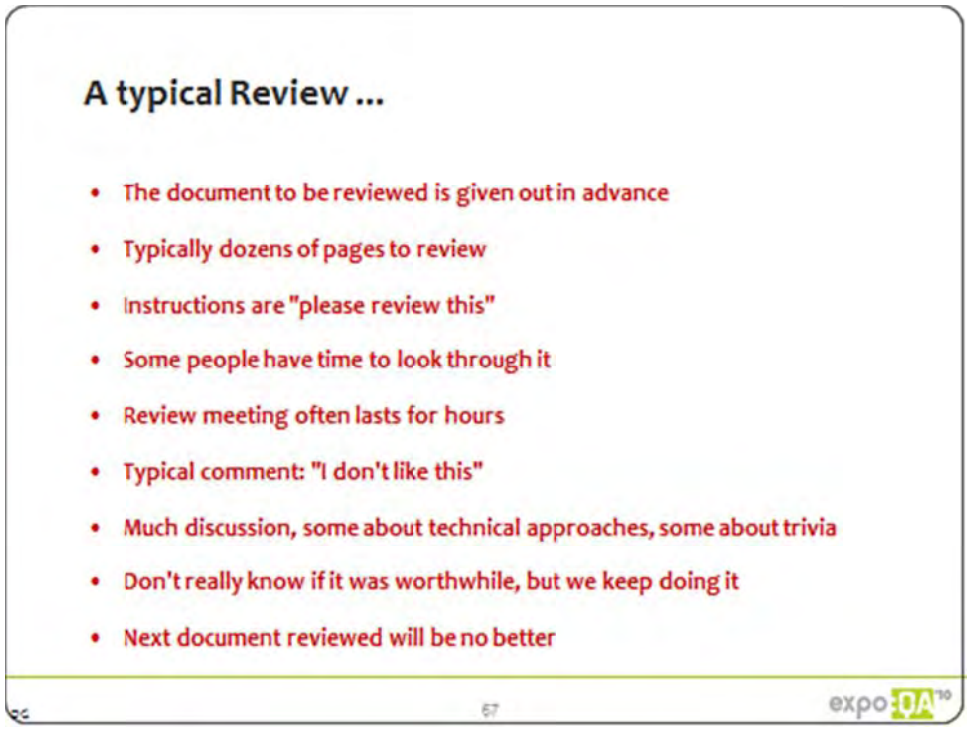
Now review again

- Any difference ?

64


Document generation
standards
source documents
rules
1. do this
2. do that
3. think about this
4. don't forget that
generate document
review
digest
kin documents
65
expo:QA


Rules
Business case
source
Requirements
Design
Implement
Wishspec
• Any work product will be reviewed against
– Itself
– Kin documents
– Source documents
If we don't have the source, how can we judge the work product?
• We always update the source document first before changing the work product(s)
– First change the Design, then the Code and the Test
– First change the Requirement, then the Design, then the Code and the Test
66
expo:QA

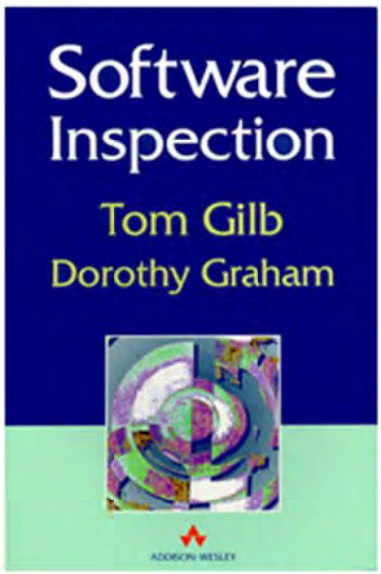

A typical Review ...
The document to be reviewed is given out in advance
Typically dozens of pages to review
Instructions are "please review this"
Some people have time to look through it
Review meeting often lasts for hours
Typical comment: "I don't like this"
Much discussion, some about technical approaches, some about trivia
Don't really know if it was worthwhile, but we keep doing it
Next document reviewed will be no better
67
expo:QA


Inspection is different
The document to be reviewed is given out in advance
not just product - rules to define defects, other docs to check against
Typically dozens of pages to review
chunk or sample
Instructions are "please review this"
training, roles
Some people have time to look through it
entry criteria to meeting, may be not worth holding
Review meeting often lasts for hours
2 hr max
Typical comment: "I don't like this"
Best Practice rules - Rules are objective, not subjective
Much discussion, some about technical approaches, some about trivia
no discussion, highly focused, anti-trivia
Don't really know if it was worthwhile, but we keep doing it
exit criteria - continually measure costs and benefits
Next document reviewed will be no better
most important focus is improvement in processes and skills
68
expo:QA

Inspection

- Most rigorous form of review
- Pioneered by Fagan (IBM) (paper 1976)
  - Locating all the defects in a work product, focus on code
- Inspection economics: Gilb/Graham (Software Inspection, 1993)
  - Quantifying the defect density of a work product and preventing poor quality work from moving downstream
- Early Inspection
  - Not waiting until the whole waterfall of the document is completed
- Is not the same as Review
- Use:
  - Walkthroughs for training
  - Technical Reviews for consensus
  - Inspections to improve the quality of the document and its process
  - Gate Reviews to decide what to do with it
- Would you like to base further work or decisions on a document of unknown quality ?

69 expo:QA'10



A ready to use recipe ...

70 expo:QA'10

16 page
Inspection
Manual

Inspection Manual
Procedures, rules, checklists and other texts
for use in Inspections

Version: 0.43 (Changed *Plan into Goal*)
Date: Oct 13, 2007
Owner: Niels Malotaux
Status: not inspected
Intended readership: anybody interested in or busy with inspections

Note: Most of these texts are originally taken from the book:
"Software Inspection" by Tom Gilb and Dorothy Graham
Addison Wesley, 1993. ISBN 0-201-63181-4, and from
web-sites, such as www.result-planning.com (Tom Gilbs web-site)
This is a starting point from which the procedures, rules, etc.
may be adapted to the local culture.

71

Inspection goals and effects

- Identify and correct major defects
- Most important:
  Identify and remove the *source* of defects
- Consequence:
  Education and interaction:
  How should we make documents in the first place?
- Interesting side-effect:
  People get to know each others documents efficiently

72

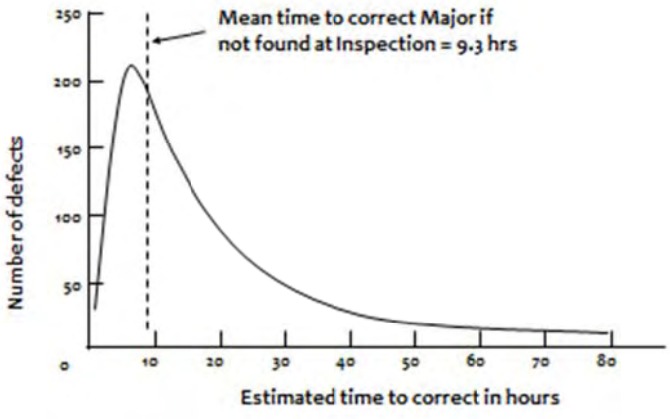
Defect classes

- Major defect
  - Defect probably has significantly increased costs to find and fix later (test, field)
    - 10 engineering hours lost extra
    - Average time in work-hours to find, log and fix a major defect by Inspection is 1 hour (observed by many sources)
- Minor defect
  - Not major (no significant impact on result)
- Super-major/critical
  - Order of magnitude more costly than major
  - Project threat

73

Cost of Repair

ref SI, fig 14.6, p315



74

Rules

- Rules are the law for documents
- Defect = Rule violation
  not: "I think this is wrong"
- Rule:
  All quality requirements must be expressed quantitatively
- Typical requirements found:
  The system should be extremely user-friendly
  The system must work exactly as the predecessor
  The system must be better than before

75 expo:QA'10

Generic Specification Rules (see Inspection Manual)

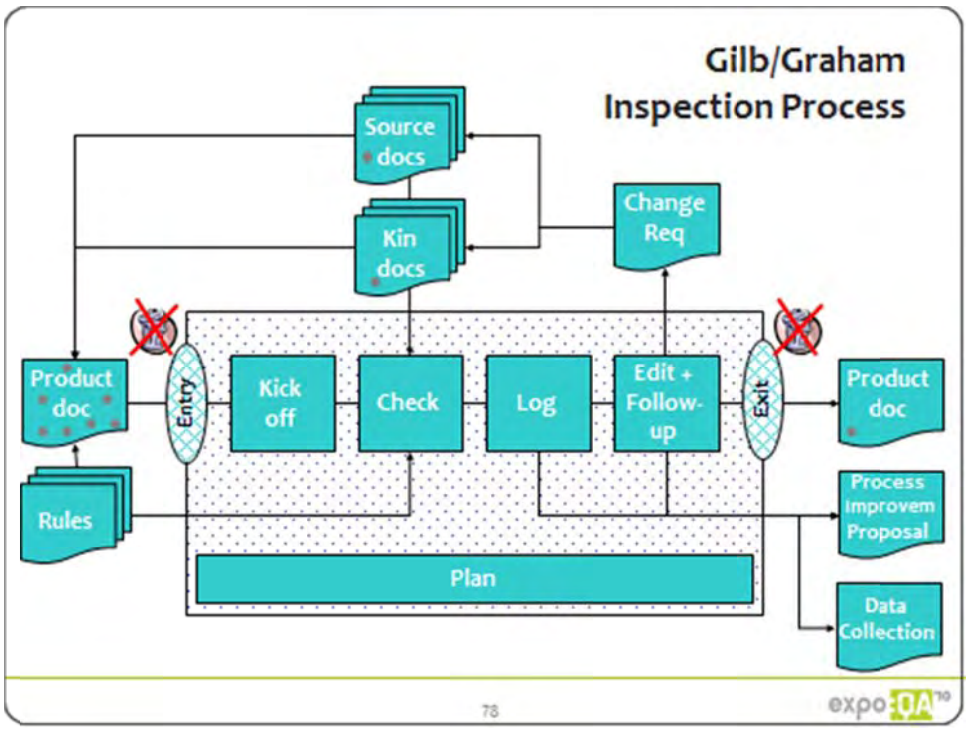
GE0	(def)	Generic engineering specification rules apply to all engineering documents as required best practices
GE1	(relevant)	All statements should be relevant to the subject
GE2	(complete)	There should not be any significant omissions
GE3	(consistent)	Statements should be consistent with other statements in the same or related documents
GE4	(unambiguous)	All specifications should be unambiguous to the intended readership
GE5	(note)	Comments, notes, suggestions, not official part of document shall be clearly marked ("", *ital*, /**/)
GE6	(brief)	All specifications shall be as brief as possible, to support their purpose, for the intended readership
GE7	(clarity)	All specifications shall result in clarity to the intended readership regarding it's purpose or intent (the burden is on author, not the reader) *Note: It is not enough that statements are unambiguous. They must contain clarity of purpose: why is it there?*
GE8	(elementary)	Statements shall be broken into their most elementary form *Note: This is so that they each can be cross-referenced externally (Traceability)*
GE9	(unique)	Specifications shall have a single instance in the entire project documentation
GE10	(source)	Statements shall have source info (spec ←source)
GE11	(risk)	The author should clearly indicate any information which is uncertain or poses any risk to the project, using indications like: {<vaguely defined>, ?, ??, 70% ±10, suitable comments or notes}
GE12	(verifiable)	All statements should be verifiable
GE13	(true)	The statement is simply not true

76 expo:QA'10

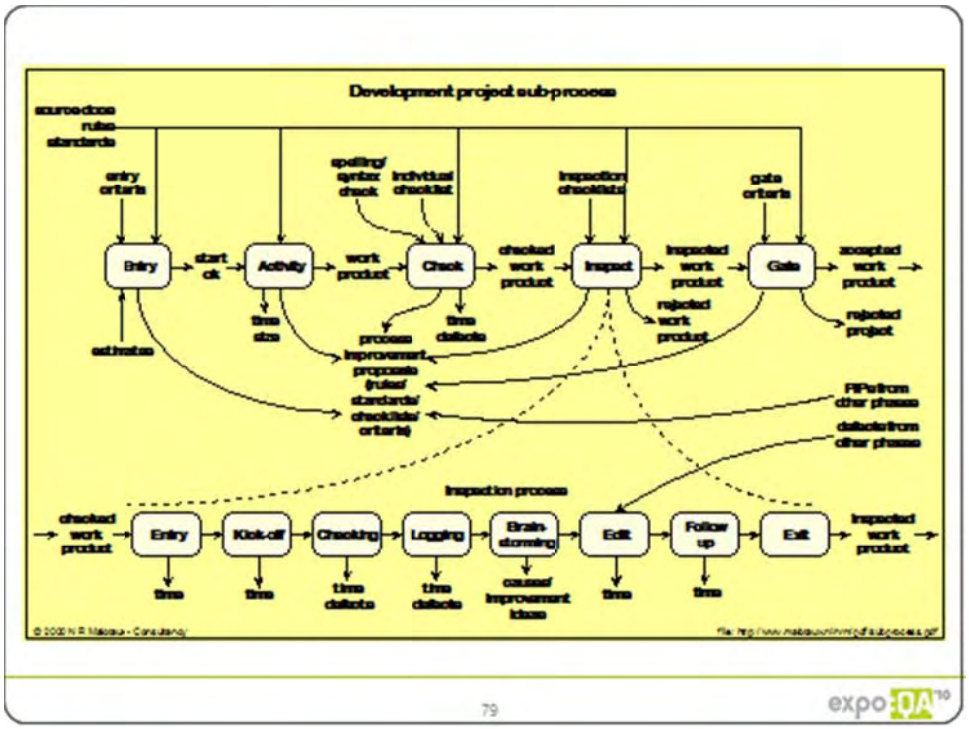
Check Lists

- Checklists contain interpretations of Rules to help reviewers to find more issues
- Rules are "The Law"
- Checklists provide "Jurisprudence"

77



78



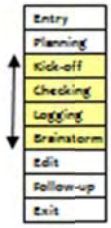
79

6 hour initial Inspection process

- 2 hr Kickoff
  - Why
  - How
  - What
- 2 hr Individual checking
  - 1 hr Whole document / relevant chapter
  - 1 hr 2 selected pages
- 2 hr Logging meeting
  - 1 hr Logging issues
  - ½ hr Discussion about Inspection process
  - ½ hr Discussion about what should have been in the document

80

4 hour mature Inspection process



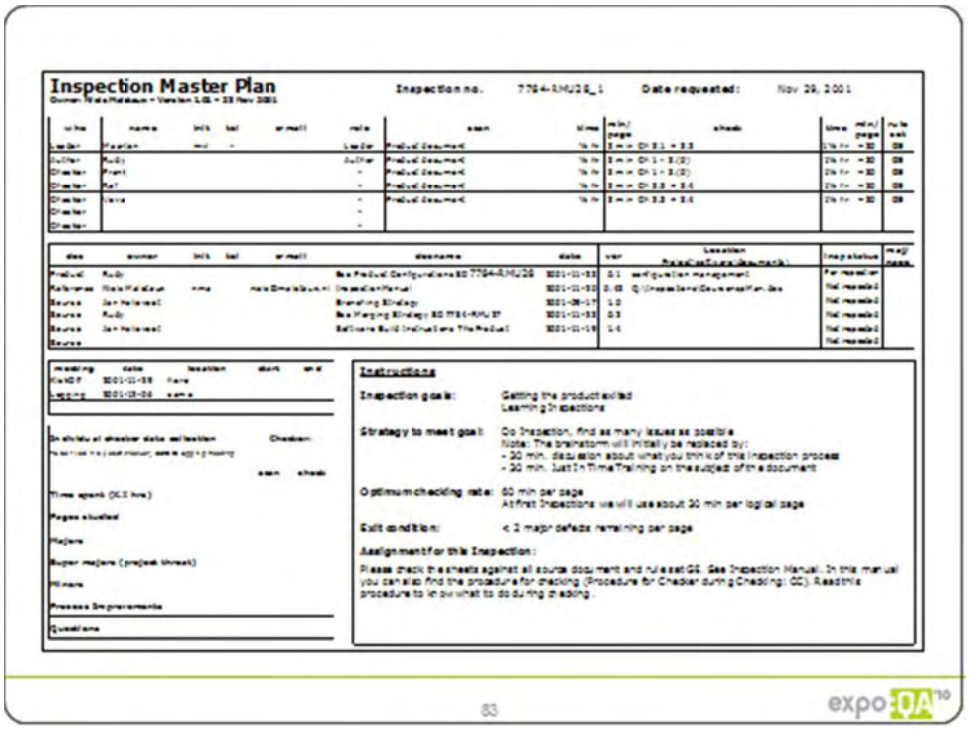
- 1/2 hr Kickoff
  - Why
  - How
  - What
- 2 hr Individual checking
  - 1 hr Whole document / relevant chapter
  - 1 hr 2 selected pages
- 1 ½ hr Logging meeting
  - 1 hr Logging issues
  - ½ hr Discussion about Inspection process
  - ½ hr Brainstorm

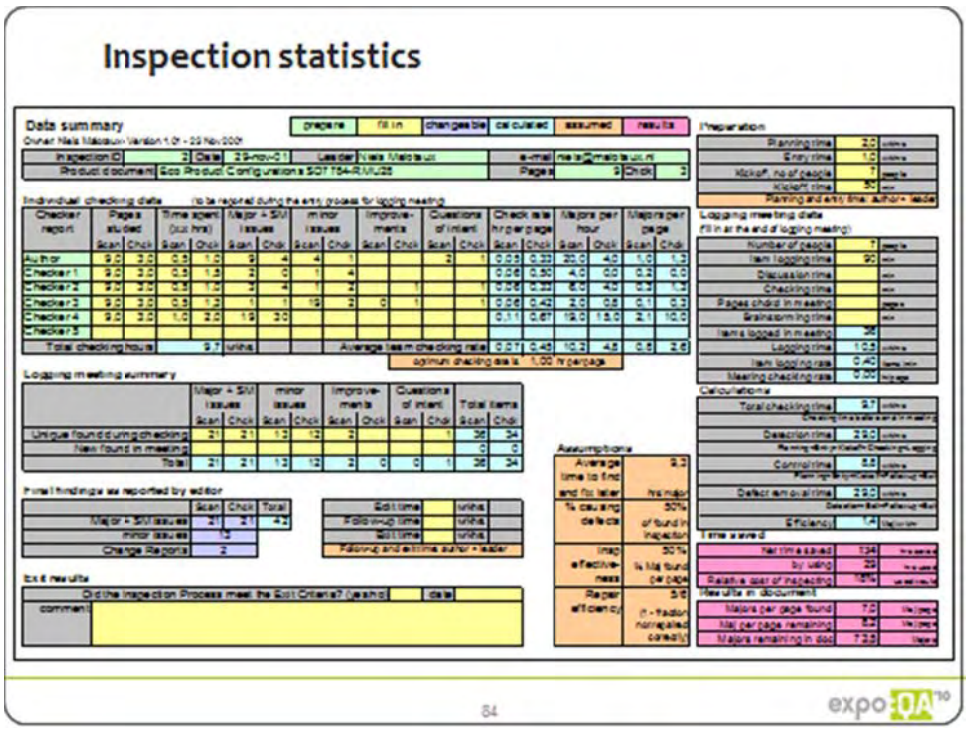
81 expo:QA'10

What do you need

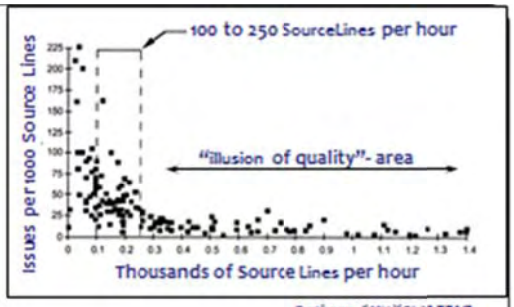
- Trained Inspection leader
- Inspection Manual
  - Rules, Procedures
- Documents + owners
- Checkers
- Inspection Master Plan template
  - Who, What, Where
- Presentation for the Kick-off meeting
  - Why, How, What
- Inspection metrics template
  - Data collection
  - Issue collection
  - (Brainstorm - fruits collection)

82 expo:QA'10


Inspection Master Plan
Instructions
Inspection goals:
Strategy to meet goal:
Optimum checking rate:
Exit condition:
Assignment for this Inspection:
83
expo:QA


Inspection statistics
Data summary
Individual checking data
Logging meeting summary
Assumptions
84
expo:QA

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")

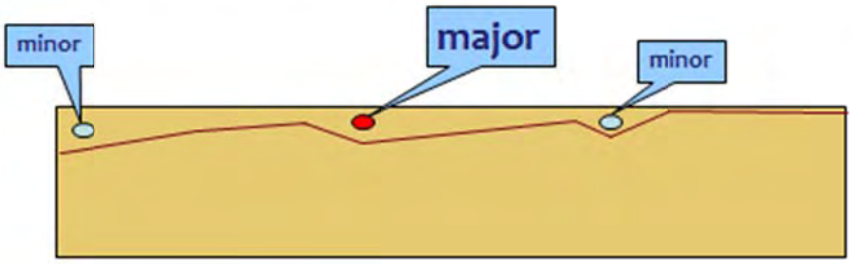
Optimum checking rate

Ref. Dorothy Graham

Here's a document: review this (or Inspect it)

Review "Thoroughness"?

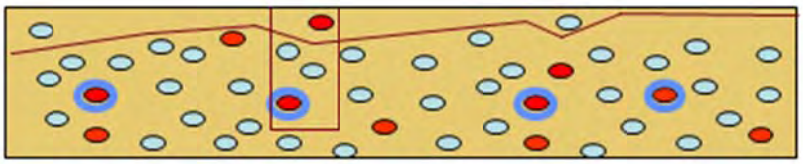
Ref. Dorothy Graham



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

Inspection Thoroughness

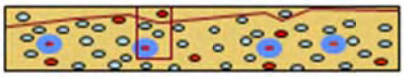
Ref. Dorothy Graham



- 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 realize it !

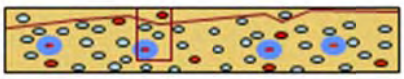
Gilb/Graham Inspection



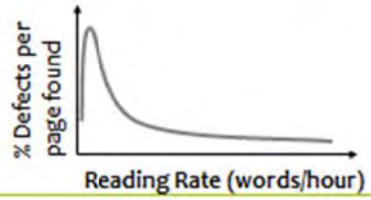
- Gilb/Graham inspection differs from other types of inspection in some or all of these ways:
  - Purpose:
    Quantifying quality, not searching for all defects
  - Controlled reading rate:
    The material being inspected is read very thoroughly in order to identify as many defects as possible (deep vs shallow sample)
  - Sampling:
    Only samples are inspected to optimize time and effort investment while maintaining the reading rate
  - Entry/Exit Criteria:
    Quantified entry and exit criteria used to guide the inspection effort
  - Rules:
    Written rule sets used to locate and classify defects

ES 89 expo:QA

Gilb/Graham Concepts
Reading Rate



- Default recommended reading rate is one logical page per hour, lower than in many other inspection methods
- This ensures adequate time to locate the vast majority of latent defects in the specification
- Supporting documents, rules, etc. can be read at any speed



Read too fast and you will miss most of the defects!

ES 90 expo:QA

Exit criteria: estimating remaining majors (after fixing)

Entry
Planning
Kick-off
Checking
Logging
Brainstorm
Edit
Follow-up
Exit

- You are about to Inspect your own document
- What is acceptable exit level?
  - 1000 estimated Major defects remaining per page ?
  - 100 ?
  - 10 ?
  - 1 ?
- What exit criteria will you use today?
  - I will accept no more than _____ estimated remaining major defects per page
- How much %% of defects do you think you'll find?
  - I will find _____ % of the defects

91 expo:QA'10

Undetected defects

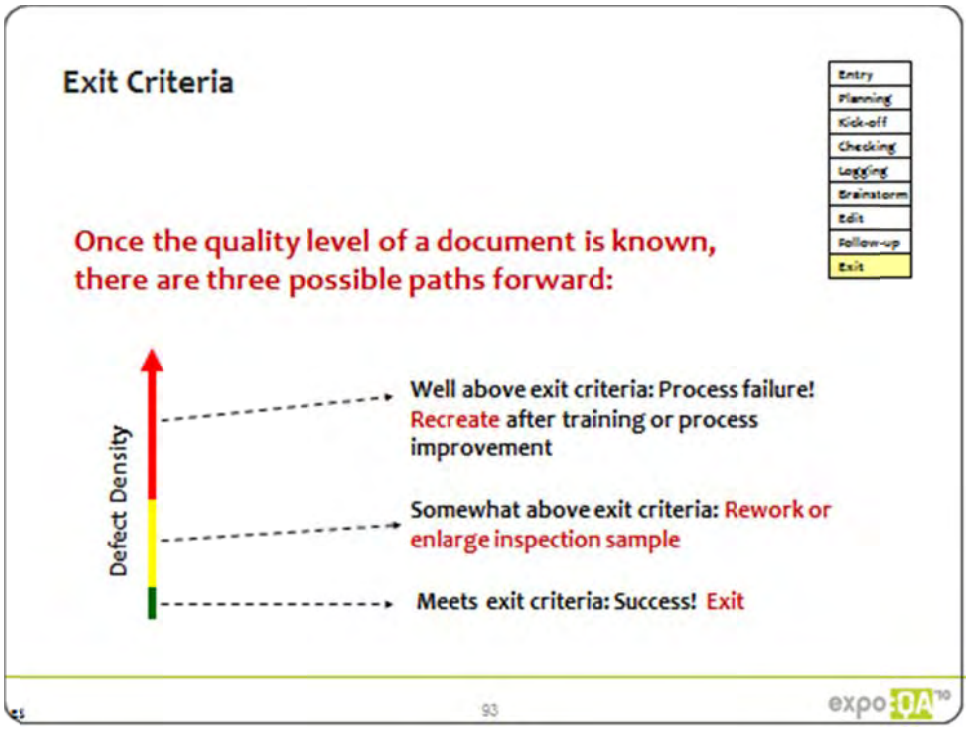
Entry
Planning
Kick-off
Checking
Logging
Brainstorm
Edit
Follow-up
Exit

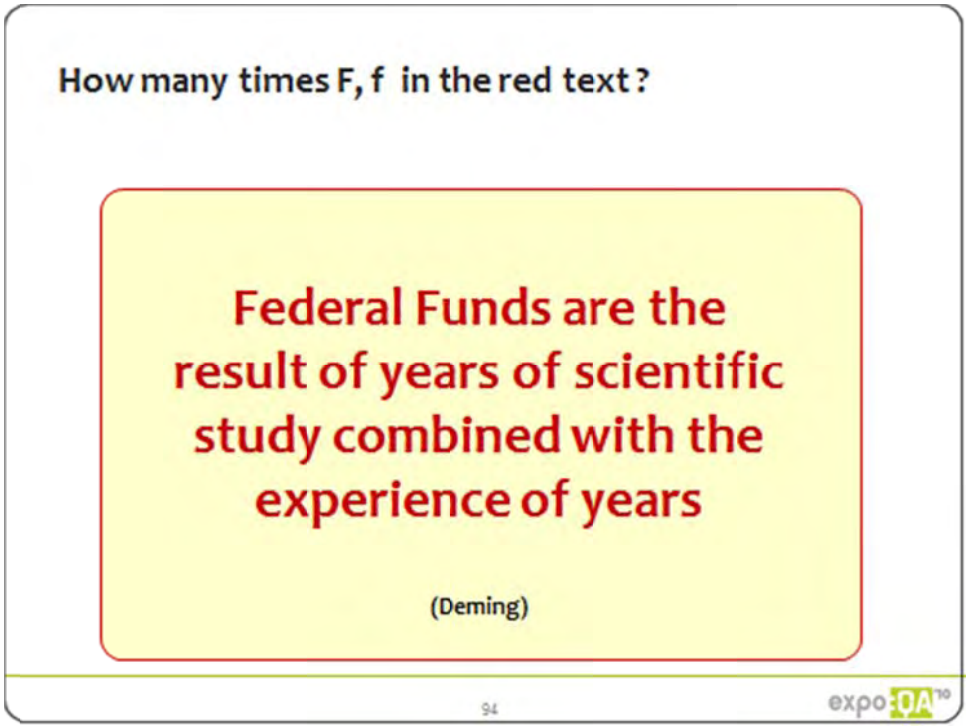
Defects present but not yet detected by Inspection

Mature Inspection process	undetected defects	yield	99% yield
– Pseudo code	20%	80%	
– Module and interface design	12%	88%	
– Source code	40%	60%	7 x
Immature Inspection process			
– All documents	70%	30%	12 x

(Lindner 1992)

92 expo:QA'10


Exit Criteria
Entry
Planning
Kick-off
Checking
Logging
Brainstorm
Edit
Follow-up
Exit
Once the quality level of a document is known, there are three possible paths forward:
Defect Density
Well above exit criteria: Process failure! Recreate after training or process improvement
Somewhat above exit criteria: Rework or enlarge inspection sample
Meets exit criteria: Success! Exit
93
expo:QA


How many times F, f in the red text?
Federal Funds are the result of years of scientific study combined with the experience of years
(Deming)
94
expo:QA

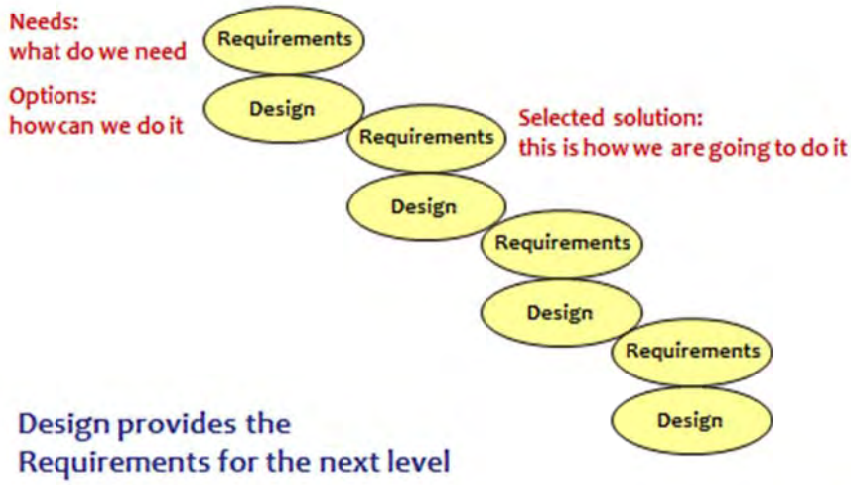
Simple Rules for Requirements

1. Unambiguous to the intended readership
   Two designers arrive at the same result
2. Clear enough to test
   Two testers get same result
3. No design mixed in requirements

Requirement: What the acquirer cares about: 'how good it should be'
vs
Design: Set of decisions made by development: 'how to be good'

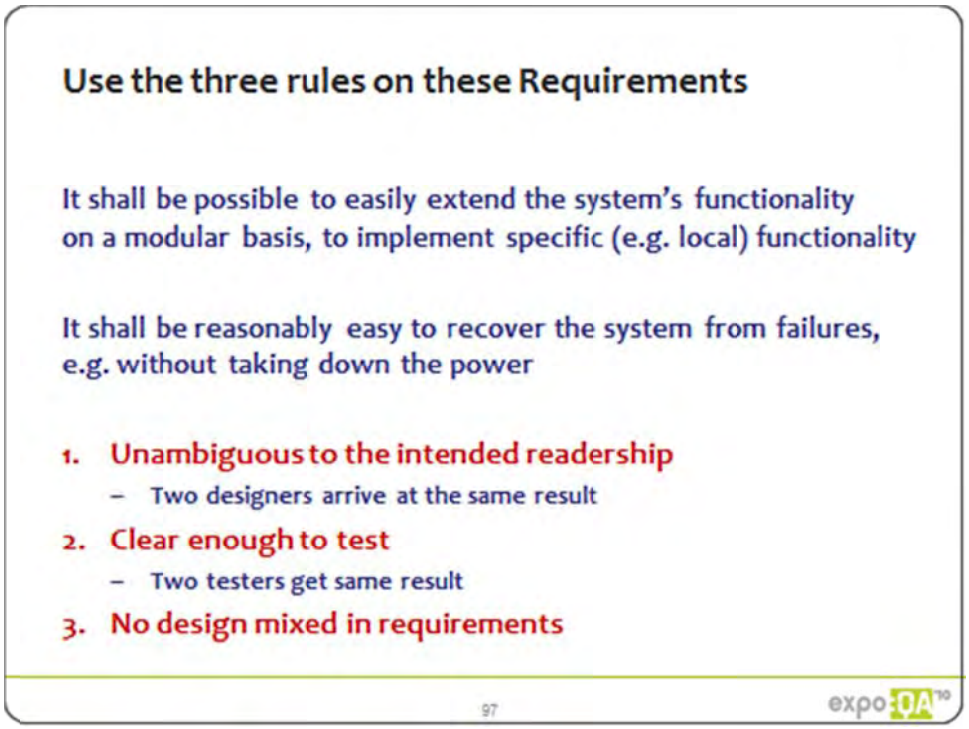
95

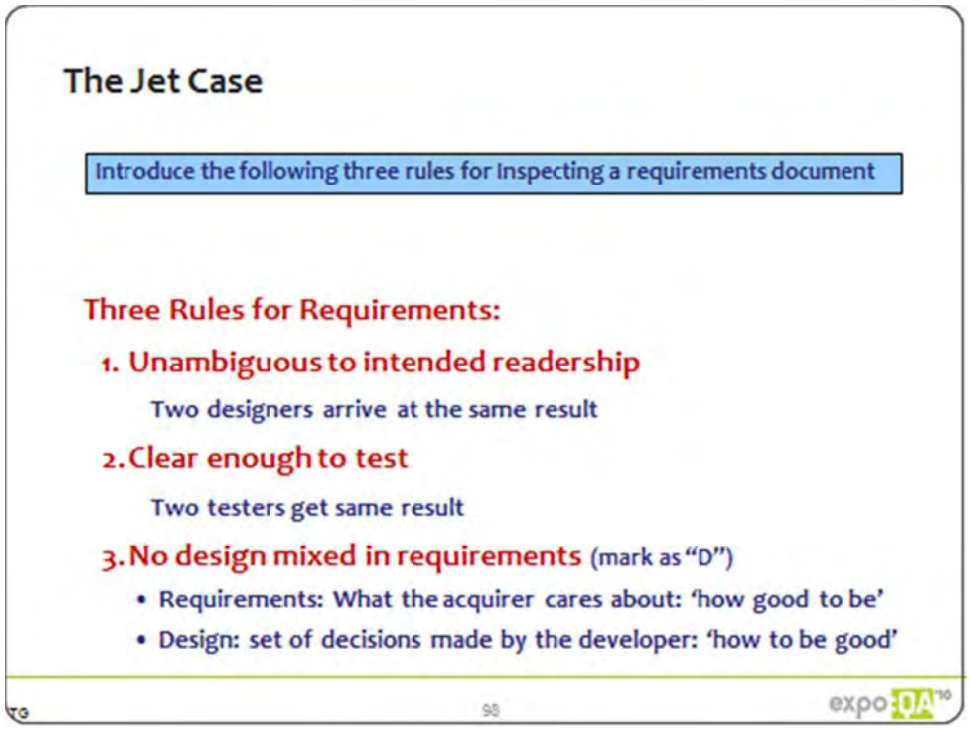
No Design in the requirements, but ...

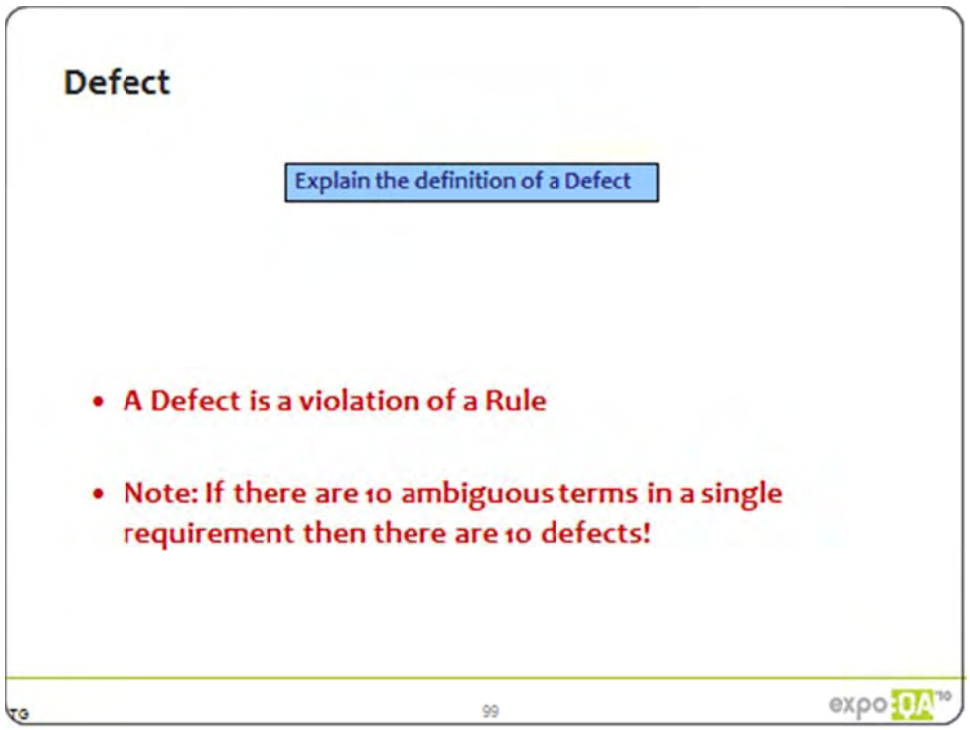


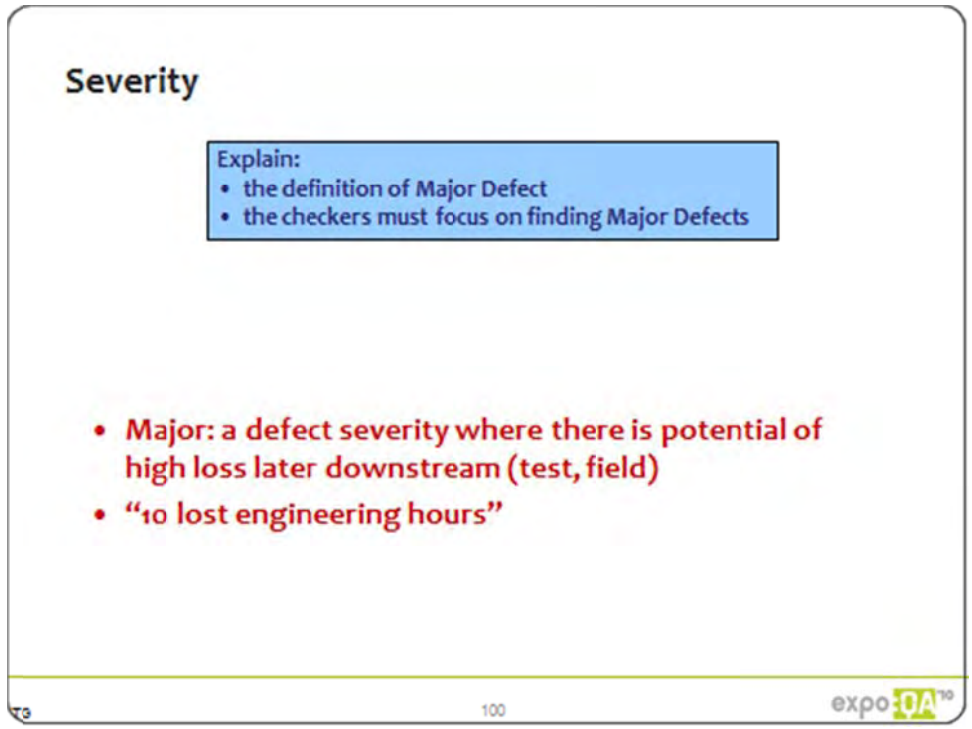
Design provides the
Requirements for the next level

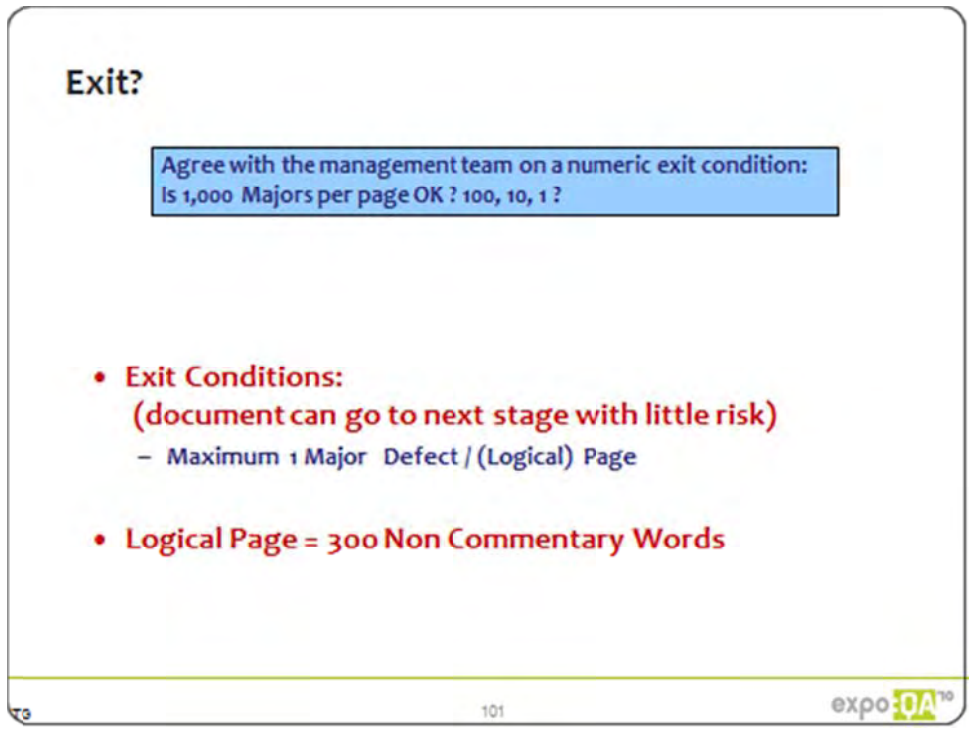
96


Use the three rules on these Requirements
It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality
It shall be reasonably easy to recover the system from failures, e.g. without taking down the power
1. Unambiguous to the intended readership
– Two designers arrive at the same result
2. Clear enough to test
– Two testers get same result
3. No design mixed in requirements
97
expo:QA


The Jet Case
Introduce the following three rules for Inspecting a requirements document
Three Rules for Requirements:
1. Unambiguous to intended readership
Two designers arrive at the same result
2. Clear enough to test
Two testers get same result
3. No design mixed in requirements (mark as "D")
• Requirements: What the acquirer cares about: 'how good to be'
• Design: set of decisions made by the developer: 'how to be good'
98
expo:QA


Defect
Explain the definition of a Defect
• A Defect is a violation of a Rule
• Note: If there are 10 ambiguous terms in a single requirement then there are 10 defects!
99
expo:QA


Severity
Explain:
• the definition of Major Defect
• the checkers must focus on finding Major Defects
• Major: a defect severity where there is potential of high loss later downstream (test, field)
• "10 lost engineering hours"
100


Exit?
Agree with the management team on a numeric exit condition:
Is 1,000 Majors per page OK ? 100, 10, 1 ?
• Exit Conditions:
(document can go to next stage with little risk)
– Maximum 1 Major Defect / (Logical) Page
• Logical Page = 300 Non Commentary Words
101

The Job

- You have up to 15 minutes for checking
  One Requirements Logical page from the 82 pages document
- Count all Rule Violations → Defects
- Classify Major and minor

102

Report for Page 81

	Total	Major	Design
1.	24	15	5
2.	44	15	9
3.	55	20	4
4.	22	4	2

Inspection results on requirements document, 4 managers

Defect Density

- Total for group 20 x 2 = 40 Majors assume are unique
- If 33% effective, total in page = 3 x 40 = 120
- Of which 2/3 or 80 were not yet found
- If we attempt to fix the 40 found, and correctly fix 5/6 then 7 are failed fixes, so:
- Total remaining after Inspection and editing = 80 + 7 = 87 Majors per page

103

Report for Page 82

	Total	Major	Design
1.	41	24	1
2.	33	15	5
3.	44	30	10
4.	24	3	5

Inspection results on requirements document, 4 other managers

Defect Density

- Total for group 30 x 2 = 60 Majors assume are unique
- If 33% effective, total in page = 3 x 60 = 180
- Of which 2/3 or 120 were not yet found
- If we attempt to fix the 60 found, and correctly fix 5/6 then 10 are failed fixes, so:
- Total remaining after Inspection and editing = 10 + 120 = 130 Majors per page

T3 104 expo:QA 10

Extrapolation to Whole Document

- Page 81: 120 majors/p
- Page 82: 180 Majors/p
- Average 150 Majors/page x 82 page = 12300 Majors in the document.

- If a Major has 1/3 chance of causing loss (12300 / 3 = 4100)
- And each loss is avg 10 hours then total project Rework cost is about 41000 hours loss
- (This project was over a year late and expected one more year)
  - 1 year = 2000 hour x 10 people = 20000 hours

T3 105 expo:QA 10

Summary (so far)

- Rules are the laws for documents
- Optimum checking rate
- Sampling
- Types of defects
- Exit criteria
- Measuring the benefit

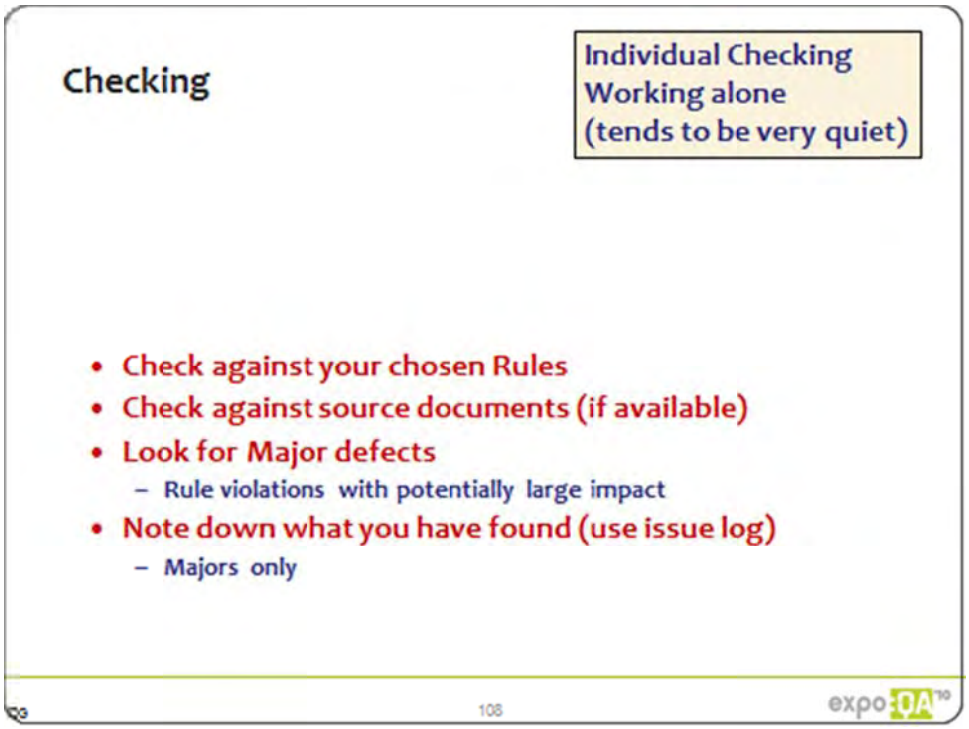
- Next: exercise

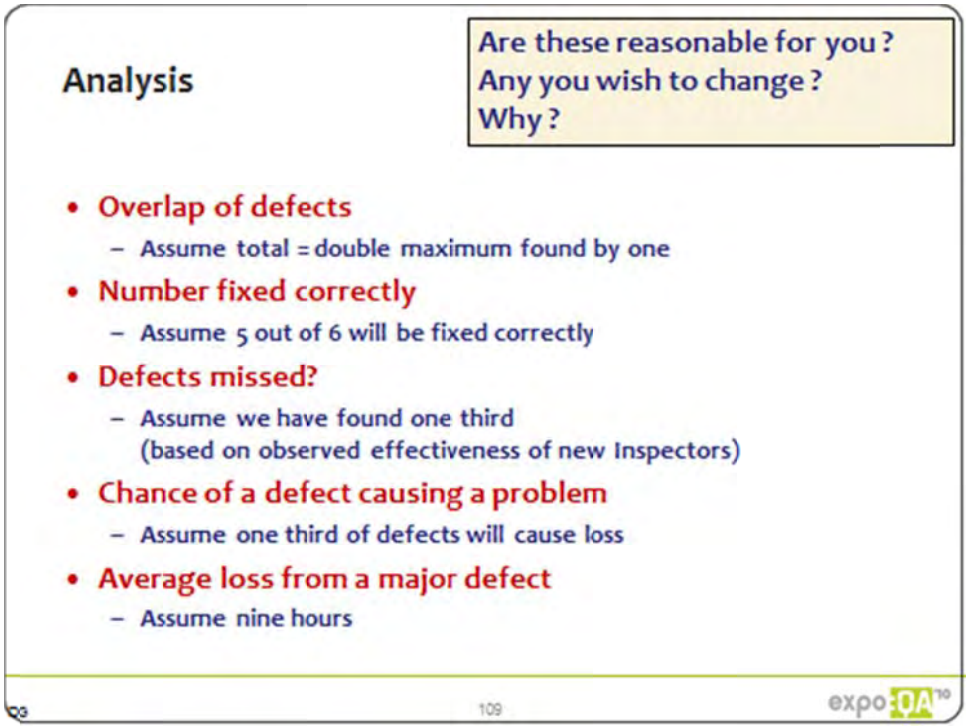
106 expo:QA'10

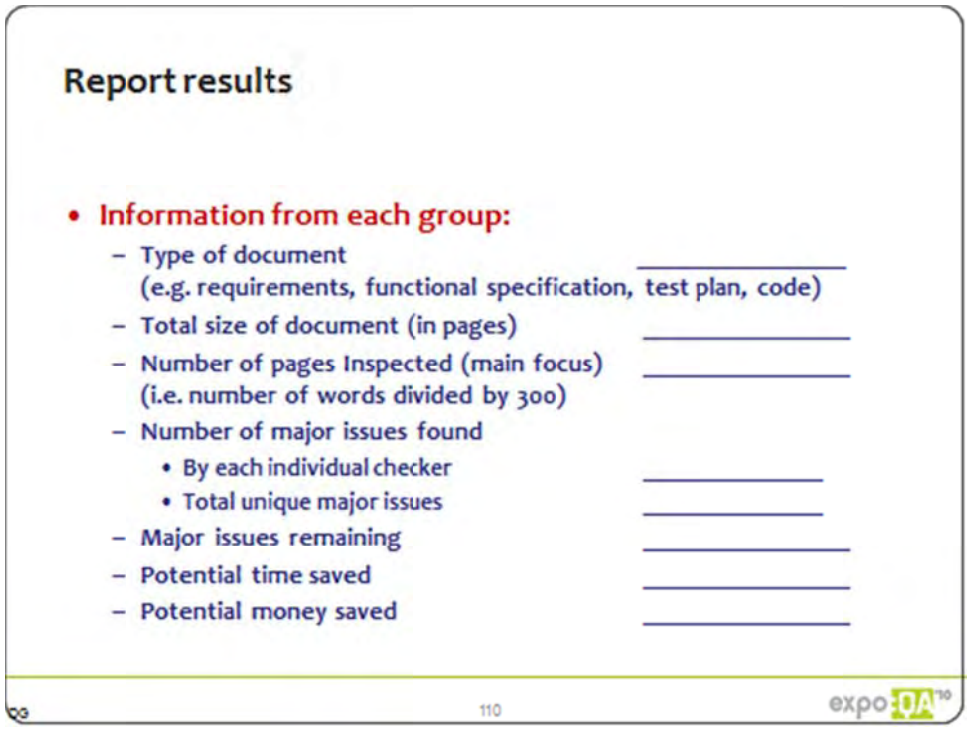
Preparation: 15 mins in groups of 3

- Which document(s) are you Inspecting ?
  - Are there any source documents ?
- Which Rules are you checking against?
  - Generic Rule set or just top 3 ?
  - Any specific Rule sets for this document ?
    - e.g. requirements ? new ones for today ?
- Which page(s) will each of you be checking ?
  - All checkers check the same (most important) page ?
    - "logical" page, not necessarily one physical page (300 words text, 100 lines of code)
- Exit criterion?
  - How many Defects remaining ?

107 expo:QA'10


Checking
Individual Checking
Working alone
(tends to be very quiet)
• Check against your chosen Rules
• Check against source documents (if available)
• Look for Major defects
– Rule violations with potentially large impact
• Note down what you have found (use issue log)
– Majors only
108
expo:QA


Analysis
Are these reasonable for you?
Any you wish to change?
Why?
• Overlap of defects
– Assume total = double maximum found by one
• Number fixed correctly
– Assume 5 out of 6 will be fixed correctly
• Defects missed?
– Assume we have found one third
(based on observed effectiveness of new Inspectors)
• Chance of a defect causing a problem
– Assume one third of defects will cause loss
• Average loss from a major defect
– Assume nine hours
109
expo:QA

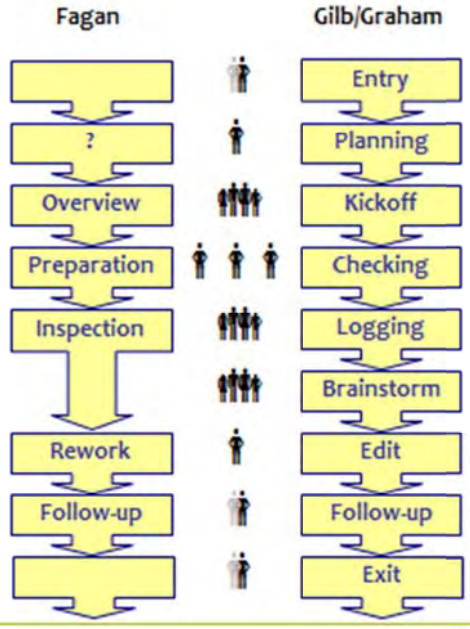

Report results
• Information from each group:
– Type of document
(e.g. requirements, functional specification, test plan, code)
– Total size of document (in pages)
– Number of pages Inspected (main focus)
(i.e. number of words divided by 300)
– Number of major issues found
• By each individual checker
• Total unique major issues
– Major issues remaining
– Potential time saved
– Potential money saved
110
expo:QA

Fagan Inspections

111

expo:QA

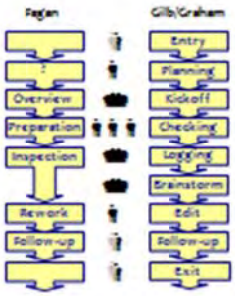
Inspection Process Steps



112

expo:QA

Fagan Inspections



- Objective: finding errors
- Based on publication in IBM Journal
- Emphasis on inspecting code
- If more than 5% reworked: 100% re-inspection
- If less than 5%: moderator decides
- *All* modifications better be inspected (even 1 line change)
- Most defects found during the meeting
- Typical defect list obtained used for prevention
- Typical defect list obtained used for next inspection
- Learn how to look for defects

113

Errors found in Inspection and in Test

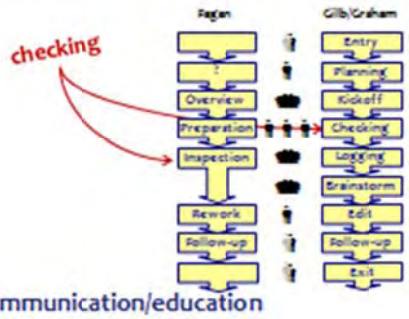
Table 1 Error detection efficiency

Process Operations	*Errors Found per K.NCSS*	*Percent of Total Errors Found*
Design		
I_1 inspection		
Coding	38*	82
I_2 inspection		
Unit test		
Preparation for acceptance test	8	18
Acceptance test	0	
Actual usage (6 mo.)	0	
Total	46	100

*51% were logic errors, most of which were missing rather than due to incorrect design.

114

Fagan Process



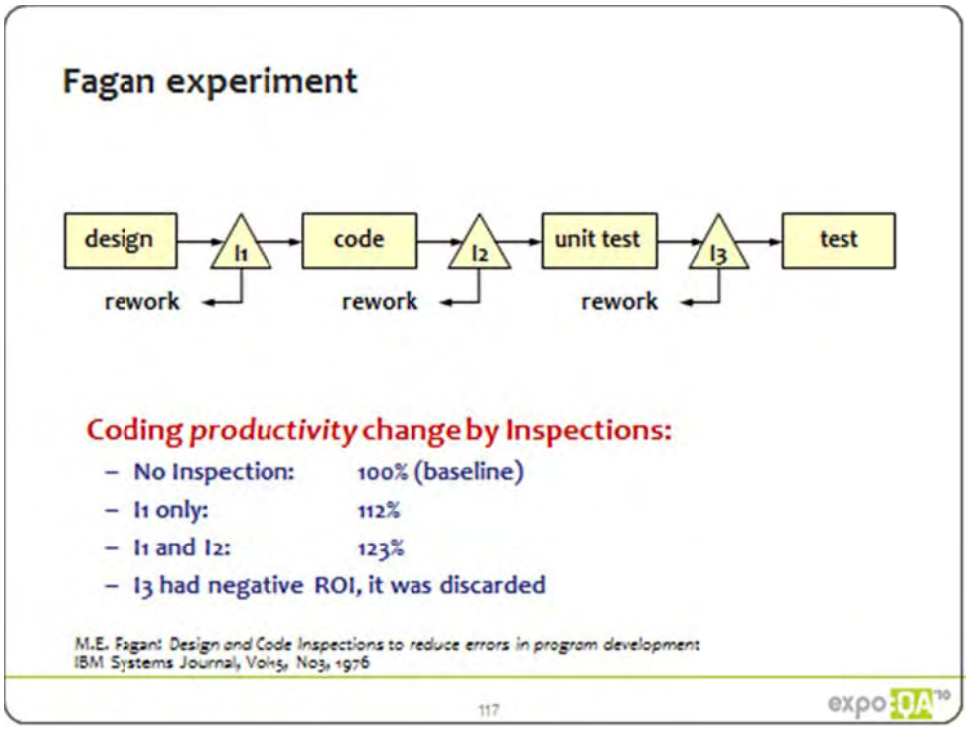
- Steps
  - Overview team Communication/education
  - Preparation individual Education
  - Inspection team Finding errors (no discussion)
  - Rework author Resolving errors and problems
  - Follow-up moderator Decision - analyse - process
- *What* to look for in Inspection
  Errors classified by type, ranked by frequency,
- *How* to look for presence of errors (education!)
- Analyse results for prevention

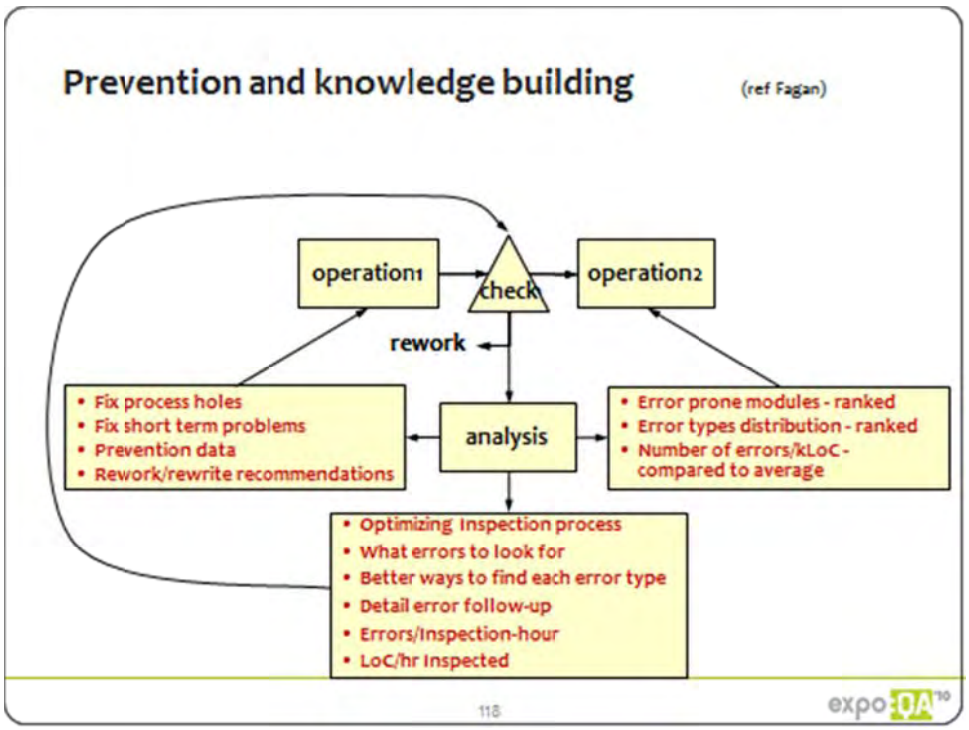
115 expo:QA10

Fagan roles

- Moderator (specially trained)
- Designer (source document)
- Coder/Implementer (current document)
- Tester (testability)

116 expo:QA10


Fagan experiment
design
I1
code
I2
unit test
I3
test
rework
rework
rework
Coding productivity change by Inspections:
– No Inspection: 100% (baseline)
– I1 only: 112%
– I1 and I2: 123%
– I3 had negative ROI, it was discarded
M.E. Fagan: Design and Code Inspections to reduce errors in program development
IBM Systems Journal, Vol15, No3, 1976
117
expo:QA


Prevention and knowledge building
(ref Fagan)
operation1
check
operation2
rework
Fix process holes
Fix short term problems
Prevention data
Rework/rewrite recommendations
analysis
Error prone modules - ranked
Error types distribution - ranked
Number of errors/kLoC - compared to average
Optimizing Inspection process
What errors to look for
Better ways to find each error type
Detail error follow-up
Errors/Inspection-hour
LoC/hr Inspected
118
expo:QA

Cleanroom Inspections

119

Cleanroom expectations

NASA Satellite control system
- 40kLoC FORTRAN
- Testing found 4.5 defect/kLoC
- 60% of programs compiled successfully first time

IBM decision support program
- 107kLoC various languages, 50 person team
- Testing found 2.6 defect/kLoC
- 5 of 8 components: no defects found, no defects found in use

IBM tape drive controller, real time data stream control
- 86kLoC, C-code, 50 person
- Testing found 1.2 defect/kLoC

Ericsson Telecom operating system
- 350kLoC, assembler and C, 70 person, 18 months
- Testing found 1 defect/kLoC

120

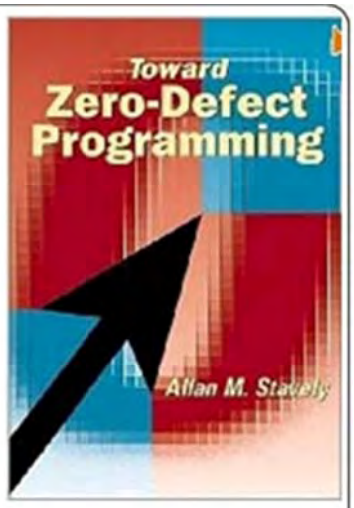
Cleanroom benefits

- Zero failures in field use
- Short development cycles
- Long product life

Quality is cheaper

121 expo:QA

Cleanroom

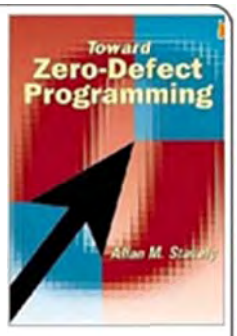


Allan M. Stavely
Toward Zero Defect Programming

There are more books, but Stavely explains it very pragmatically

122 expo:QA

Cleanroom Software Development



- Design (Mathematical proof)
- Verification (by others)
- Implementation
- Verification (by others)
- No unit test
- Only Integration Test (by others)
  (Test is Running Code)

- *Verification* is for finding defects
- *Testing* is for not finding defects

123 expo:QA 10

Cleanroom fundamentals

- Design principle
  - Designers can and should produce systems free of defects *before* testing
- Testing principle
  - The purpose of testing is to *measure* quality
- Main development model
  - Incremental (Cleanroom) / Evolutionary (Gilb) / Cyclic (TSP)
    - Each increment is a working subset of the final product
    - Stable requirements for each increment
    - No eleventh hour integration

124 expo:QA 10

Cleanroom Principles

- Incremental development
  - User verifiable increments
- Team organisation
  - 4~8 people
- Formal methods of specification and design
  - Level of formalism varies even within project
- Intense review
  - Mathematical proof of correctness
  - Verifying individual control structures
- No unit test
  - No testing infinite number of paths, infinite combination of data
- Statistical testing as reliability measurement
  - Testing is not suitable for bug-hunting

125 expo:QA'10

Cleanroom Inspections

- The purpose of Inspection is to eliminate defects
- Exit criterion for design:
  - One design statement materializes as 3 to 10 code statements
- Checklists of typical errors we make
  - Listed in order of frequency
- No Unit Test - Developer does not 'try' software!
- Testing:
  - Finding as many of the remaining defects as possible
  - Too many errors discovered
    → previous steps are not being done properly
    → redo previous steps (do not "repair")

126 expo:QA'10

Cleanroom: Slowest reviewer sets the pace

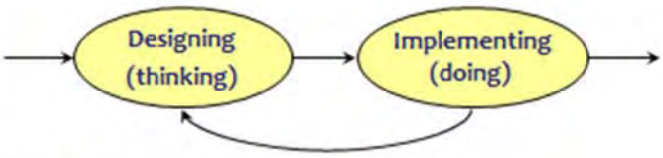
- Wrong: Does anyone consider this incorrect?
  (dreamers won't answer)
- Better: Does everybody agree that this is correct?
  (attention is required)
- A team does not consider a verification condition proven until the slowest person to respond has expressed agreement

It is important to resist taking shortcuts here

127 expo:QA

Getting stuck somewhere?

- Getting stuck in implementation? Back to the design!



- Getting stuck in Inspection? Back to the design!
- Getting stuck in Testing? Back to the design!

- *Why* do we get stuck?
- Root cause analysis!

128 expo:QA

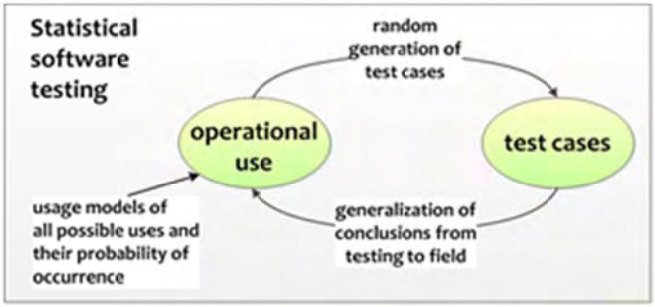
Testing in Cleanroom

- Testing is an important part of the process, but it is done only after verification (by Inspection) is successfully completed
- Testing is done:
  - Primarily to *measure* quality
  - Secondarily to find defects that escaped detection during verification
- Number of bugs per thousand lines of code <10 after verification, compilation and syntax checking
- Very good teams produce 2.3 defects per kLoC and reject code with 4 or 5 defects per kLoC
- *No attempt is done to try to salvage rejected code by debugging*
  - The code is sent back to the developers to be rewritten and reverified
  - Then it is tested as a completely new product
- Usage based testing – statistical testing
- Risk based testing – high risk, low probability will still be checked !

129 expo:QA'10

Statistical Testing

You need also other forms of testing !



130 expo:QA'10

No Unit Testing in Cleanroom

- We should avoid any kind of private testing, whether it is unit testing or some other kind
- We may experiment for various reasons, but we must resist the temptation to test our actual code

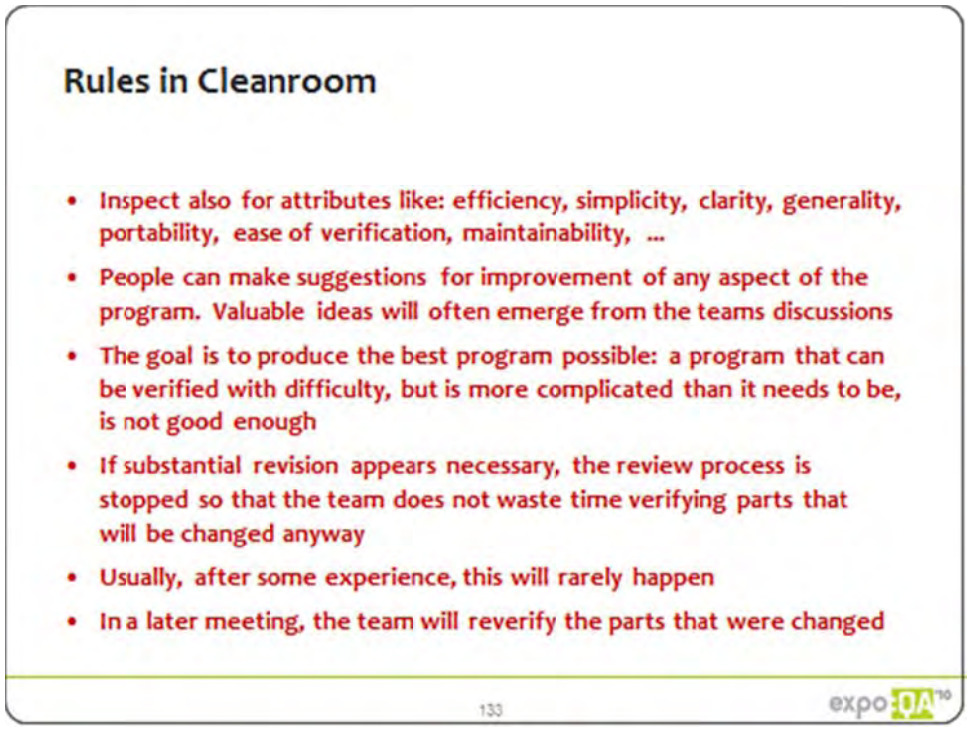
131 expo:QA

Philosophy behind Cleanroom

- To avoid dependence on costly defect-removal processes
- By writing code increments right the first time and
- Verifying their correctness *before* testing

(Linger, 1994)

132 expo:QA

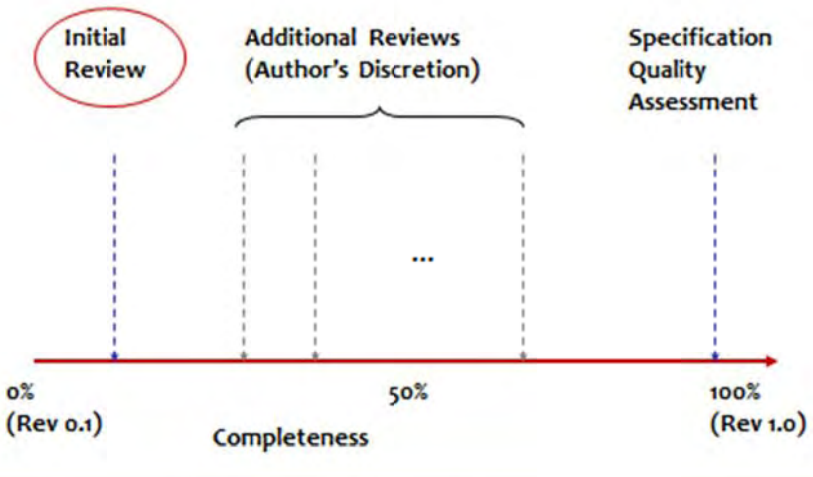

Rules in Cleanroom
• Inspect also for attributes like: efficiency, simplicity, clarity, generality, portability, ease of verification, maintainability, ...
• People can make suggestions for improvement of any aspect of the program. Valuable ideas will often emerge from the teams discussions
• The goal is to produce the best program possible: a program that can be verified with difficulty, but is more complicated than it needs to be, is not good enough
• If substantial revision appears necessary, the review process is stopped so that the team does not waste time verifying parts that will be changed anyway
• Usually, after some experience, this will rarely happen
• In a later meeting, the team will reverify the parts that were changed
133
expo:QA

Early Inspections

134

Early Inspection

Prevention costs less than Repair



135

Initial Review

Purpose: Locating mistakes and tendencies that could lead to injecting major defects if not corrected

When: As soon as the author has completed a small representative portion of the specification, typically a few pages or 600-1200 words (e.g. few requirements)

Who: Individual or small team (1 or 2)
- Expertise in the subject matter
- Expertise in generic principles (such as requirements engineering, design, specific language)

What: Detailed review of the specification against rules and checklists for known error conditions and dangerous tendencies; formal inspection may be used

Duration: Because the sample is small, the initial review takes only 1-2 hr

The earlier it's reviewed, the more defects we can *prevent*

ES 136 expo:QA'10

Initial Review Checklist

- ✓ Use a small team of experienced reviewers
- ✓ Schedule the review to minimize author waiting time
- ✓ Focus on issues that are or will cause major defects
- ✓ Avoid elements of style
- ✓ Be constructive at all times
- ✓ Focus on the work product, and never on the author
- ✓ Maintain confidentiality!
  The review is for the author's benefit

Reviewers: Your job is to make the author look like a hero

ES 137 expo:QA'10

Case Study 1 - Situation

- Large e-business integrated application with 8 requirements authors, varying experience and skill
  - Each sent the first 8-10 requirements of estimated 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

ES 138 expo:QA 10

Case Study 1 - Results

Average major defects per requirement in initial review	8
Average major defects per requirement in completed 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

ES 139 expo:QA 10

Why Early Inspection Works

- Many defects are repetitive and can be prevented
  - Early review allows an author to get independent feedback on individual tendencies and errors
  - By applying early learning to the rest (~90%) of the writing process, many defects are prevented before they occur
  - Reducing rework in both the document under review and all downstream derivative work products

ES 140 expo:QA'10

Case Study 2 - Situation

- 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 simulated Early Inspection
  - First round, inspected 6 randomly-selected test cases
  - Author notes systematic defects in the results, reworks the 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 on the next project...

ES 141 expo:QA'10

Case Study 2 - Results

First round inspection	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 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
- 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

ES 142 expo:QA 10

Early Detection vs. Prevention

Denise Leigh (Sema group, UK), British Computer Society address, 1992:

An eight-work-year development, delivered in five increments over nine months for Sema Group (UK), found:

- 3512 defects through inspection
- 90 through testing
- and 35 (including enhancement requests) through product field use

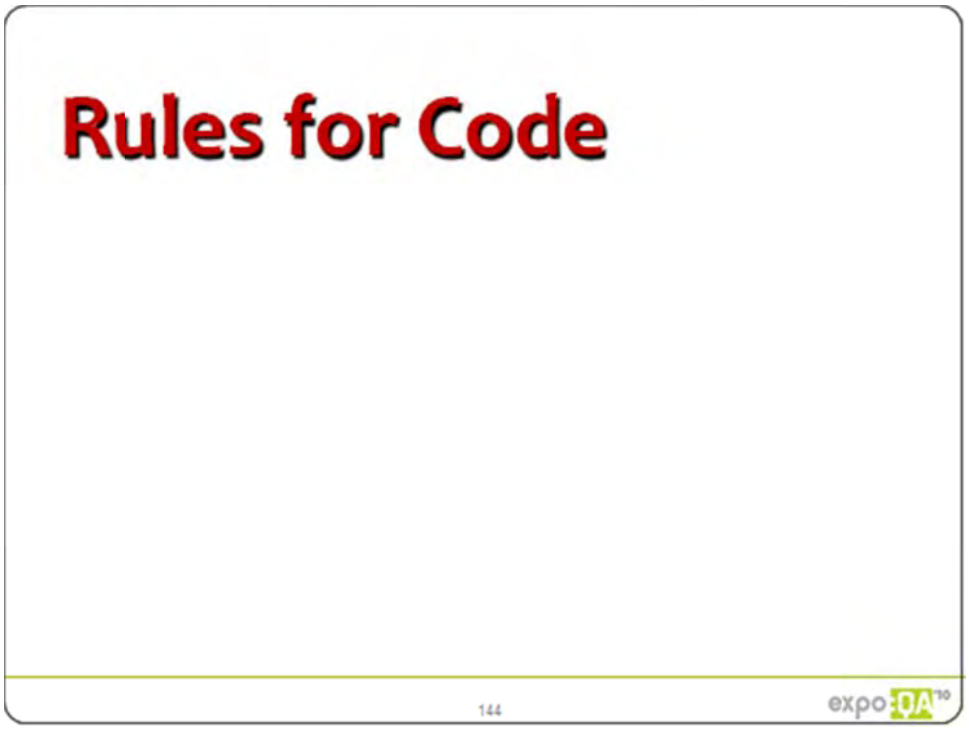
After two evolutionary deliveries, *unit testing of programs was discontinued because it was no longer cost-effective*

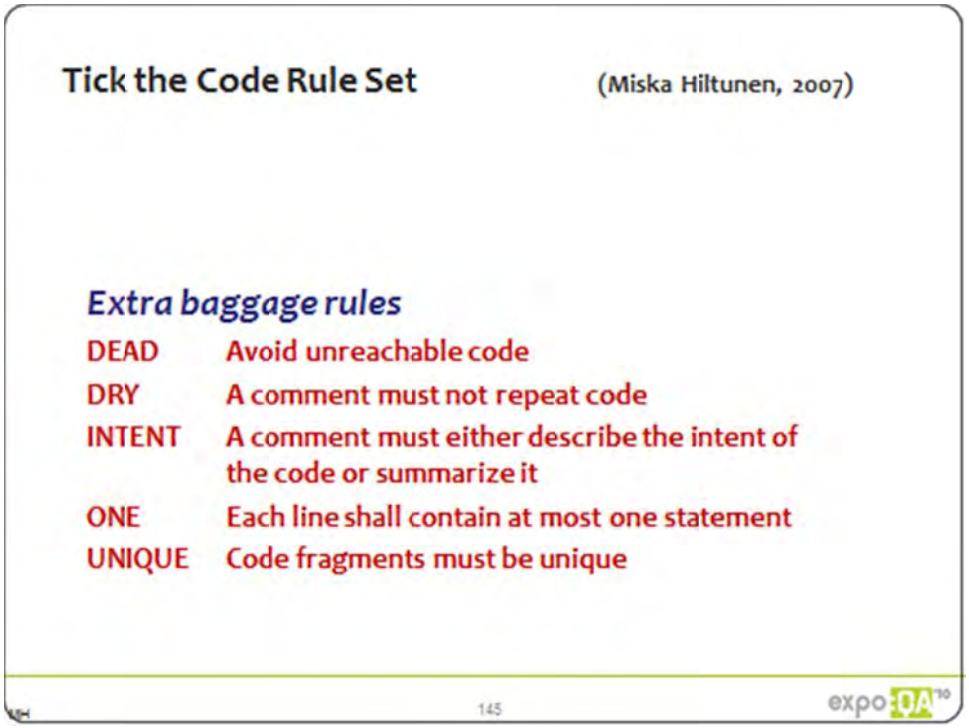
Nice job! Early detection has big benefits - BUT...

How many of the 3512 defects found in end-of-line inspections could have been completely prevented by Early Inspection?

Cost-effective defect prevention is the bottom line

ES 143 expo:QA 10


Rules for Code
144
expo:QA


Tick the Code Rule Set
(Miska Hiltunen, 2007)
Extra baggage rules
DEAD Avoid unreachable code
DRY A comment must not repeat code
INTENT A comment must either describe the intent of the code or summarize it
ONE Each line shall contain at most one statement
UNIQUE Code fragments must be unique
145
expo:QA

Tick the Code Rule Set (Miska Hiltunen, 2007)

Missing info rules

DEFAULT	A 'switch' must always have a 'default' clause
ELSE	An 'if' always has an 'else'
MAGIC	Do not hardcode values
PTHESES	Parenthesize amply
TAG	Forbidden: marker comments
ACCESS	Variables must have access routines
HIDE	Direct access to global and member variables is forbidden

146 expo:QA

Tick the Code Rule Set (Miska Hiltunen, 2007)

Chaos-inducers

CALL	Call subroutines where feasible
NAME	Bad names make code bad
RETURN	Each routine shall contain exactly one 'return'
SIMPLE	Code must be simple
FAR	Keep related actions together
DEEP	Avoid deep nesting
FOCUS	A routine shall do one and only one thing

147 expo:QA

Tick the Code Rule Set

(Miska Hiltunen, 2007)

Risky assumptions

CHECK-IN	Each routine shall check its input data
NEVERNULL	Never access a 'NULL' pointer or reference
NULL	Set freed or invalid pointers to 'NULL'
CONST 1ST	Put constants on the left side in comparisons
ZERO	Never divide by zero
PLOCAL	Never return a reference or pointer to local data
ARRAY	Array accesses shall be within the array
VERIFY	Setter must check the value for validity

148 expo:QA'10

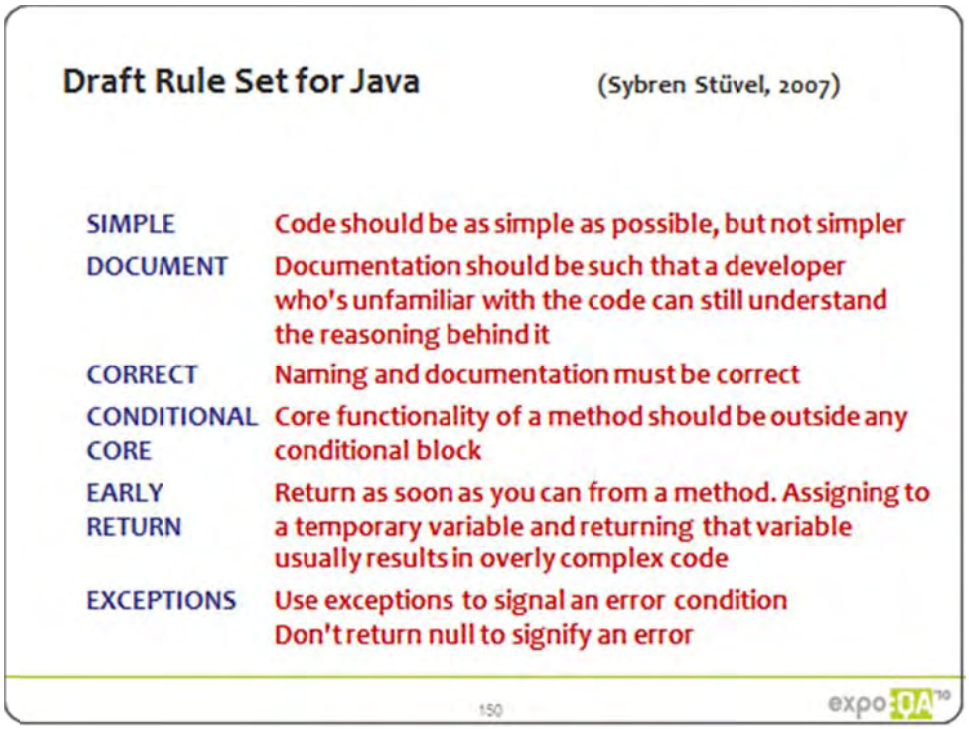
Tick the Code Speed

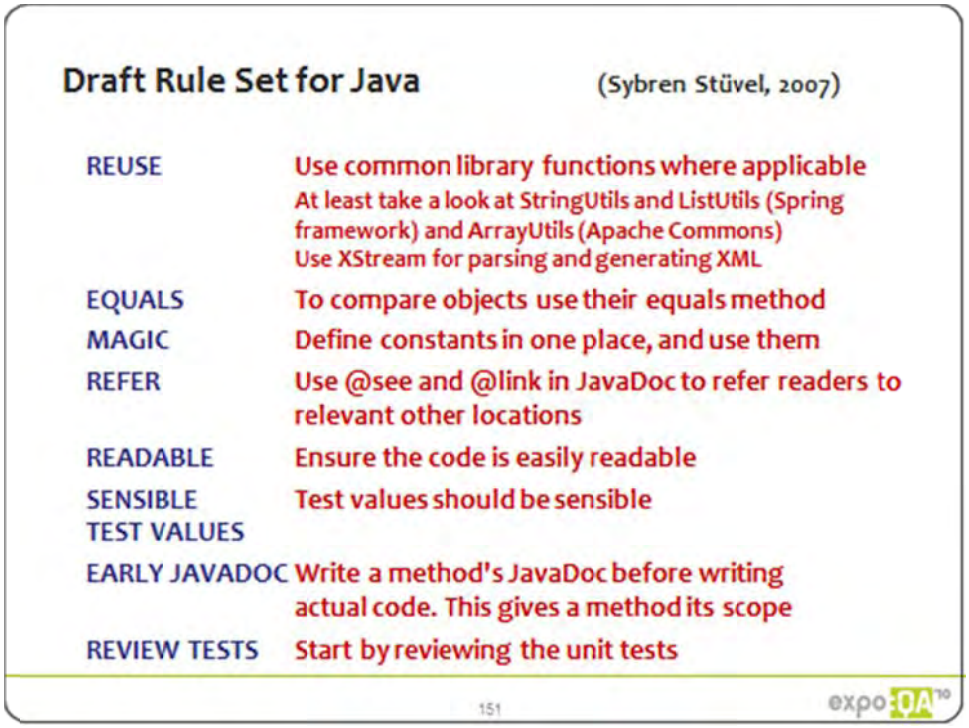
(Miska Hiltunen, 2007)

Rule	Call	Check-In	Dead	Deep	Default	Dry	Else
Ticks/hr	46	82	45	76	11	53	322
Rule	Hide	Magic	Name	NeverNull	Tag	Unique	
Ticks/hr	186	516	93	90	18	20	

- Average number of ticks found per hour per rule
- Software developers could find this many violations in one hour in the code they produce
- 144 developers checked for 108h to create the data

149 expo:QA'10


Draft Rule Set for Java
(Sybren Stüvel, 2007)
SIMPLE
Code should be as simple as possible, but not simpler
DOCUMENT
Documentation should be such that a developer who's unfamiliar with the code can still understand the reasoning behind it
CORRECT
Naming and documentation must be correct
CONDITIONAL CORE
Core functionality of a method should be outside any conditional block
EARLY RETURN
Return as soon as you can from a method. Assigning to a temporary variable and returning that variable usually results in overly complex code
EXCEPTIONS
Use exceptions to signal an error condition
Don't return null to signify an error
150
expo:QA


Draft Rule Set for Java
(Sybren Stüvel, 2007)
REUSE
Use common library functions where applicable
At least take a look at StringUtils and ListUtils (Spring framework) and ArrayUtils (Apache Commons)
Use XStream for parsing and generating XML
EQUALS
To compare objects use their equals method
MAGIC
Define constants in one place, and use them
REFER
Use @see and @link in JavaDoc to refer readers to relevant other locations
READABLE
Ensure the code is easily readable
SENSIBLE TEST VALUES
Test values should be sensible
EARLY JAVADOC
Write a method's JavaDoc before writing actual code. This gives a method its scope
REVIEW TESTS
Start by reviewing the unit tests
151
expo:QA

MISRA C

- MISRA: Motor Industry Software Reliability Association
- MISRA C (1998) has 127 rules
- Providing a set of guidelines to restrict features in the ISO C language of known undefined or otherwise dangerous behaviour
- Of these, 93 are required and the remaining 34 are advisory
  - Rule 104 (required): Non-constant pointers to functions shall not be used

Version	Rules	Sections	Pages
MISRA C 1998	127	17	69
MISRA C 2004	141	21	111

152 expo:QA'10

MISRA C

Rule 59 (required): The statement forming the body of an "if", "else if", "else", "while", "do ... while", or "for" statement shall always be enclosed in braces

```
if (x == 0)
{
y = 10;
z = 0;
}
else
y = 20;
z = 1;
```

153 expo:QA'10

MISRA C

Rule 33 (required):
The right hand side of a
"&&" or "||" operator
shall not contain side effects

```
if ((x == y) || (*p++ == z))
{
/* do something */
}
```

```
if (x == y)
{
doSomething = 1;
}
else if (*p++ == z)
{
doSomething = 1;
}

if (doSomething)
{
/* do something */
}
```

154 expo:QA

MISRA C

Motor Industry Software Reliability Association

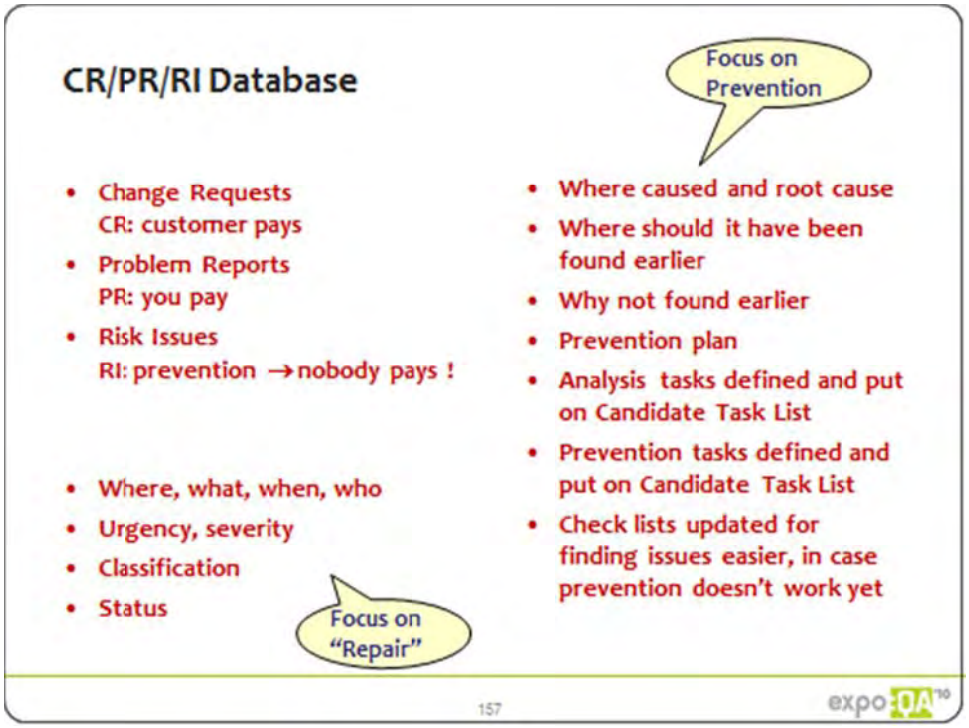
```
a[i] = ++i;
```
happens once in every 7,000 lines in C

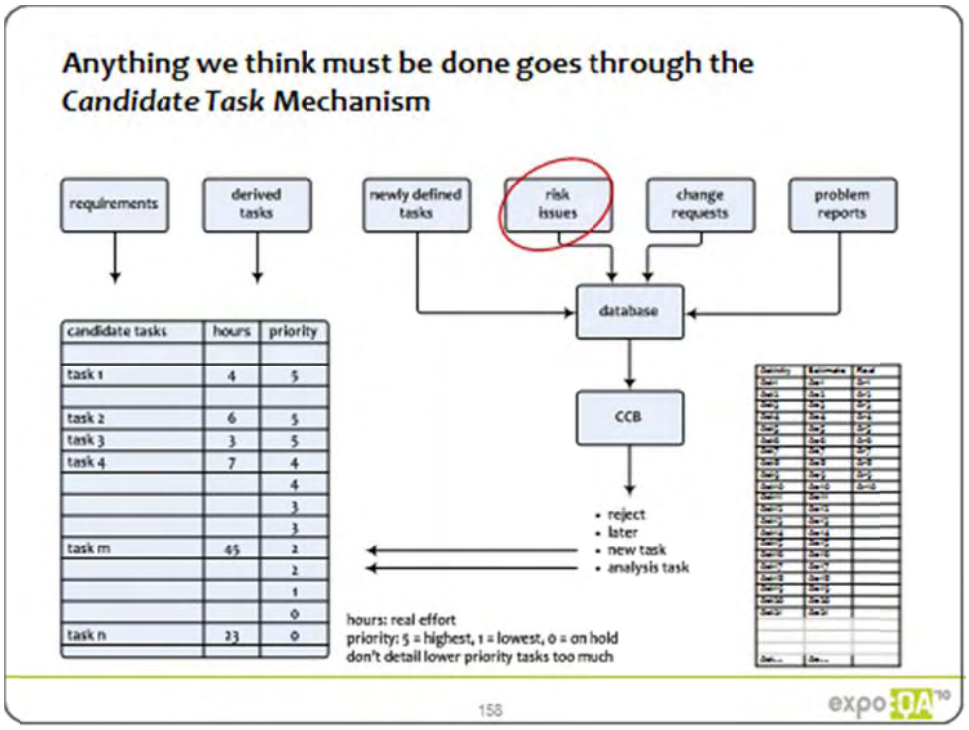
```
c == d;

if (c=d)
{
}
```

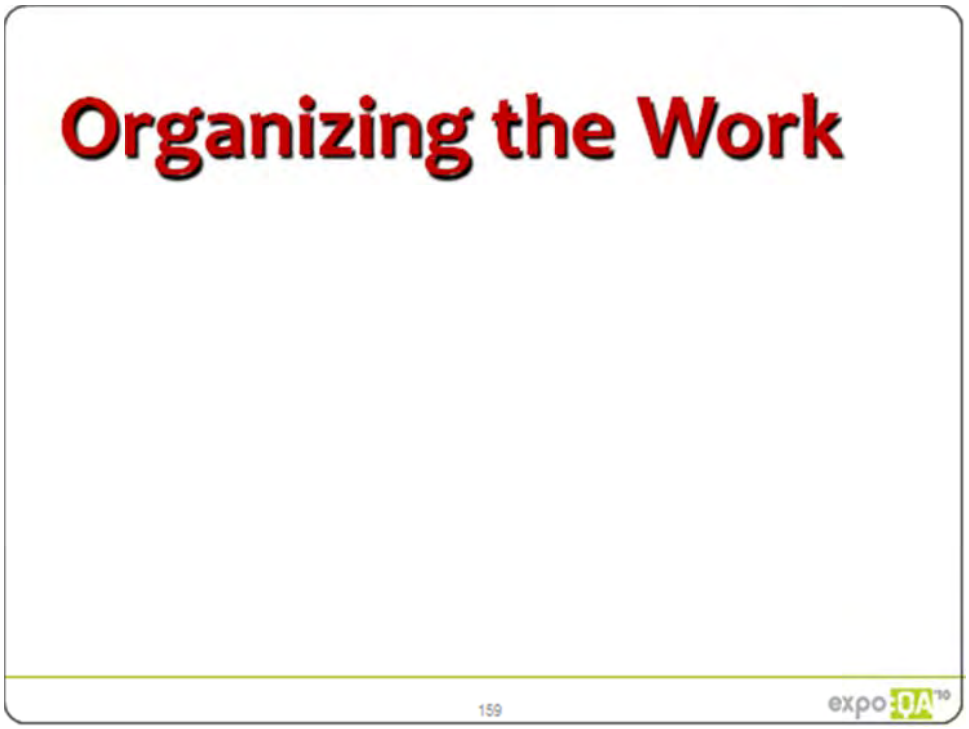
Put on checklist

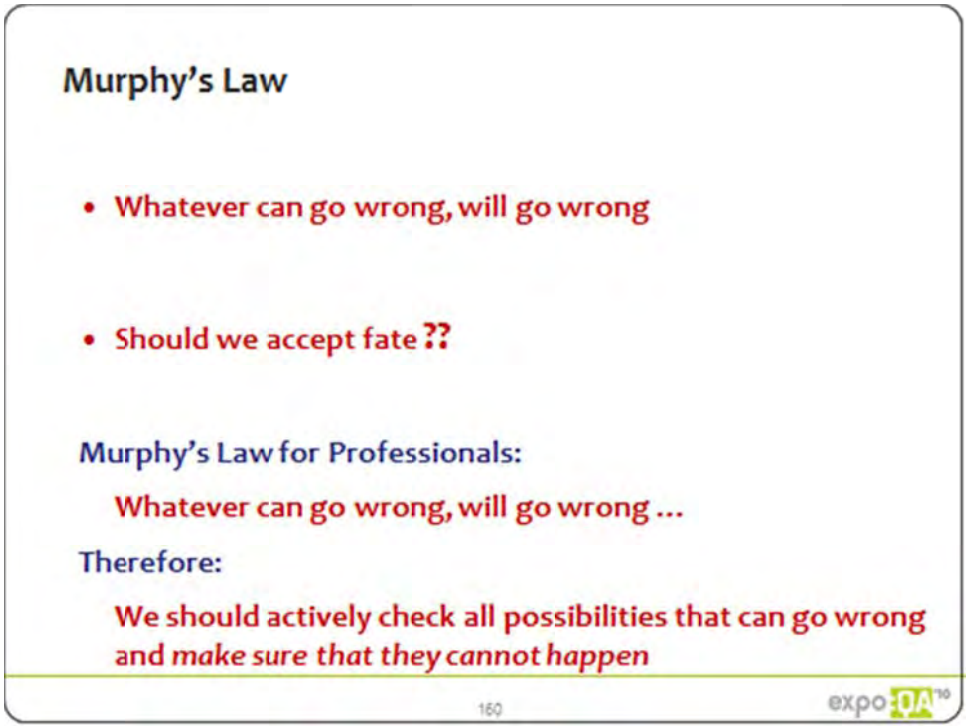
155 expo:QA


CR - PR - RI
Database
156
expo:QA


CR/PR/RI Database
Change Requests
CR: customer pays
Problem Reports
PR: you pay
Risk Issues
RI: prevention → nobody pays !
Where, what, when, who
Urgency, severity
Classification
Status
Focus on "Repair"
Focus on Prevention
Where caused and root cause
Where should it have been found earlier
Why not found earlier
Prevention plan
Analysis tasks defined and put on Candidate Task List
Prevention tasks defined and put on Candidate Task List
Check lists updated for finding issues easier, in case prevention doesn't work yet
157
expo:QA


Anything we think must be done goes through the *Candidate Task* Mechanism
requirements
derived tasks
newly defined tasks
risk issues
change requests
problem reports
database
CCB
• reject
• later
• new task
• analysis task
candidate tasks
hours
priority
task 1
4
5
task 2
6
5
task 3
3
5
task 4
7
4
4
3
3
task m
45
2
2
1
0
task n
23
0
hours: real effort
priority: 5 = highest, 1 = lowest, 0 = on hold
don't detail lower priority tasks too much
158
expo:QA


Organizing the Work
159
expo:QA


Murphy's Law
• Whatever can go wrong, will go wrong
• Should we accept fate ??
Murphy's Law for Professionals:
Whatever can go wrong, will go wrong ...
Therefore:
We should actively check all possibilities that can go wrong and make sure that they cannot happen
160
expo:QA

Preflection, foresight, prevention

Insanity is doing the same things over and over again
and hoping the outcome to be different *(let alone better)*
Albert Einstein 1879-1955, Benjamin Franklin 1706-1790, it seems Franklin was first

Only if we change our way of working,
the result may be different

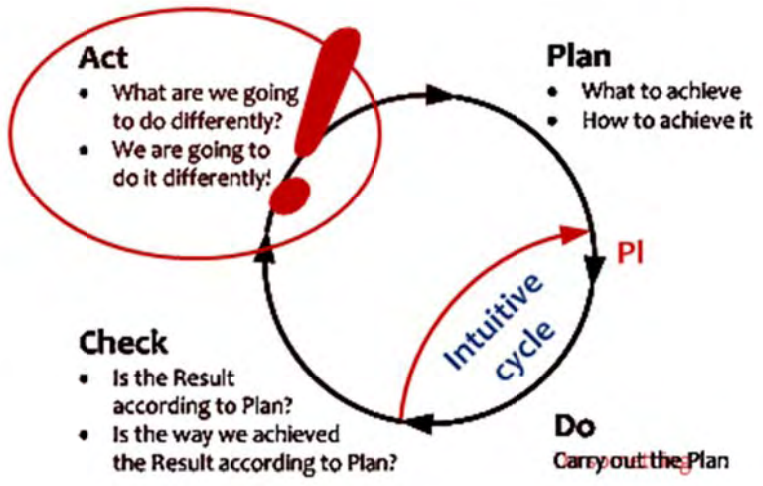
- Hindsight is easy, but reactive
- Foresight is less easy, but proactive
- Reflection is for hindsight and learning
- Preflection is for foresight and prevention

Only with *prevention* we can save precious time
This is used in the Deming or Plan-Do-Check-Act cycle

161 expo:QA 10

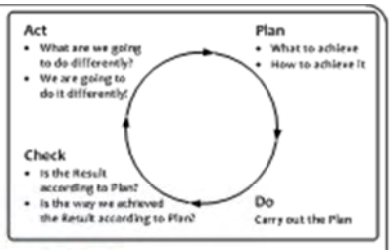
The essential ingredient: the PDCA Cycle

(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)



162 expo:QA 10

Evo



- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle
  actively, deliberately, rapidly and frequently,
  for *Product*, *Project* and *Process*, based on RoI and highest *value*
- Combining Planning, Requirements- and Risk-Management into
  Result Management
- We know we are not perfect, but the customer shouldn't find out
- Evo is about delivering Real Stuff to Real Stakeholders
  doing Real Things *"Nothing beats the Real Thing"*
- Projects seriously applying Evo, routinely conclude
  successfully on time, or earlier

163 expo:QA'10

Evolutionary Project Management (Evo)

- Plan-Do-Check-Act
  - The powerful ingredient for success
- Business Case
  - Why we are going to improve *what*
- Requirements Engineering
  - What we are going to improve *and what not*
  - How much we will improve: quantification
- Architecture and Design
  - Selecting the optimum compromise for the conflicting requirements
- Early Review & Inspection
  - Measuring quality while doing, learning to prevent doing the wrong things

Right product

Zero Defects Attitude

Evo Project Planning

- Weekly TaskCycle
  - Short term planning
  - Optimizing estimation
  - Promising what we can achieve
  - Living up to our promises
- Bi-weekly DeliveryCycle
  - Optimizing the requirements and checking the assumptions
  - Soliciting feedback by delivering Real Results to *eagerly waiting* Stakeholders
- TimeLine
  - Getting and keeping control of Time: Predicting the future
  - Feeding program/portfolio/resource management

Right time

164 expo:QA'10

Conventional Evo Testing

- Final validation shouldn't find any problems
- Earlier verifications mirror quality level to developers: how far from the goal and what still to learn
- Evo has *no debugging phase!*

165

Developers are constantly optimizing

- The product
  how to arrive at the most effective product (goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

166

Testers are constantly optimizing

- The product
  how to arrive at the most effective product (goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

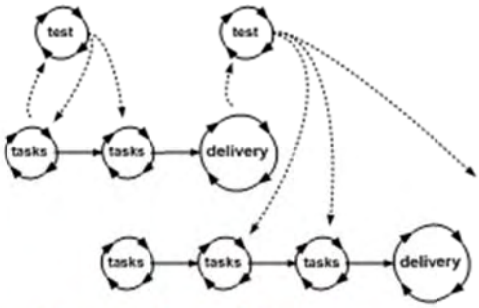
167 expo:QA

Further Improvement

- Tester's customer is "the developers"
- Finding defects is not the goal (except if they're there)
- Project Success is
- Testers select and use *any method* appropriate
- Testers check work in progress *before* it is finished
- Testers solve the Review and Inspection organizing problem
- Testing is organized the Evo way, entangling intimately with the development process

168 expo:QA

How to start doing it



- Testers organize their work in weekly TaskCycles
- DeliveryCycle is the Test-Feedback cycle
- Testers use their own TimeLine, synchronized with the developers TimeLine
- Testers conclude their work in sync with developers
- Testers *know* what they are supposed to test
- Testers check work in progress *even before* it is finished

169 expo:QA 10

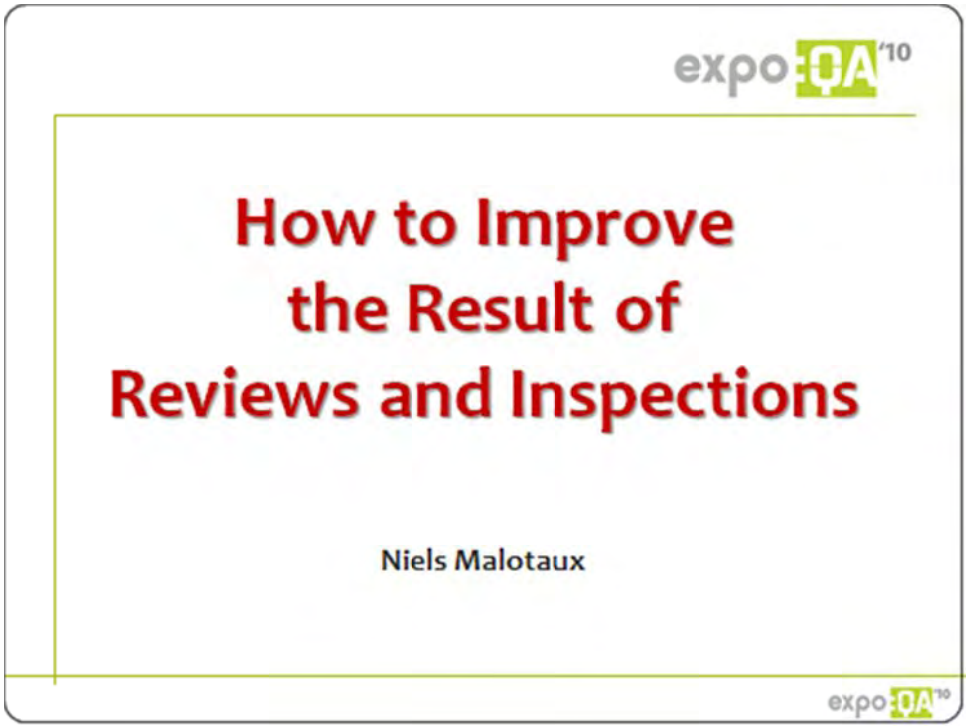
The aim of Testing

- Being done as soon as the development is done
- Well, almost

- Excuses, excuses, excuses
  - The developers are always late
    (Evo developers live up to their promises)
  - The developers don't take us seriously
    (Evo developers ask testers for help)
  - The developers don't inject enough defects
    (now testing becomes a challenge)
- Helping development to be successful

170 expo:QA 10


www.malotaux.nl/Booklets
1 Evolutionary Project Management Methods (2001)
Issues to solve, and first experience with the Evo Planning approach
More
2 How Quality is Assured by Evolutionary Methods (2004)
After a lot more experience: rather mature Evo Planning process
3 Optimizing the Contribution of Testing to Project Success (2005)
How Testing fits in
3a Optimizing Quality Assurance for Better Results (2005)
Same as Booklet 3, but for non-software projects
4 Controlling Project Risk by Design (2006)
How the Evo approach solves Risk by Design (by process)
5 TimeLine: How to Get and Keep Control over Longer Periods of Time (2007)
Replaced by Booklet 7, except for the step-by-step TimeLine procedure
6 Human Behaviour in Projects (APCOSE 2008)
Human Behavioural aspects of Projects
7 Evolutionary Planning, or How to Achieve the Most Important Requirement (2008)
Planning of longer periods of time, what to do if you don't have enough time
8 Help! We have a QA Problem! (2009)
Use of TimeLine technique: How we solved a 6 month backlog in 9 weeks
RS Measurable Value with Agile (Ryan Shriver - 2009)
Use of Evo Requirements and Prioritizing principles
www.malotaux.nl/nrm/Insp
Inspection pages
171
expo:QA'10


expo:QA'10
How to Improve the Result of Reviews and Inspections
Niels Malotaux
expo:QA'10