22-23 Oct 2007

Niels Malotaux

Reviews & Inspections

N R Malotaux - Consultancy
The Netherlands
+31-30-2288868
+31-30-2288869
niels@malotaux.nl
www.malotaux.nl/nrm/English

Niels Malotaux
Reviews & Inspections

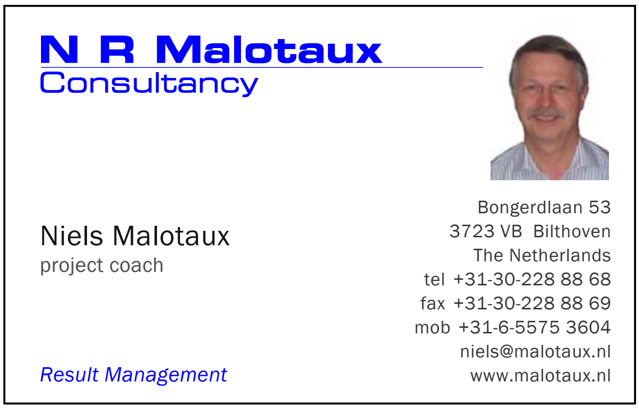
Niels Malotaux

Niels Malotaux is an independent Project Coach specializing in optimizing project performance. He has over 30 years experience in designing hardware 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: delivering what the customer needs, when he needs it, to enable customer success. To this effect, Niels developed an approach for effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached some 80 projects in 20+ organizations in the Netherlands, Belgium, Ireland, India, Japan and the US, which led to a wealth of experience in which approaches work better and which work less in practice.

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.



N R Malotaux
Consultancy

Niels Malotaux
project coach

Bongerdlaan 53
3723 VB Bilthoven
The Netherlands
tel +31-30-228 88 68
fax +31-30-228 88 69
mob +31-6-5575 3604
niels@malotaux.nl
www.malotaux.nl

Result Management

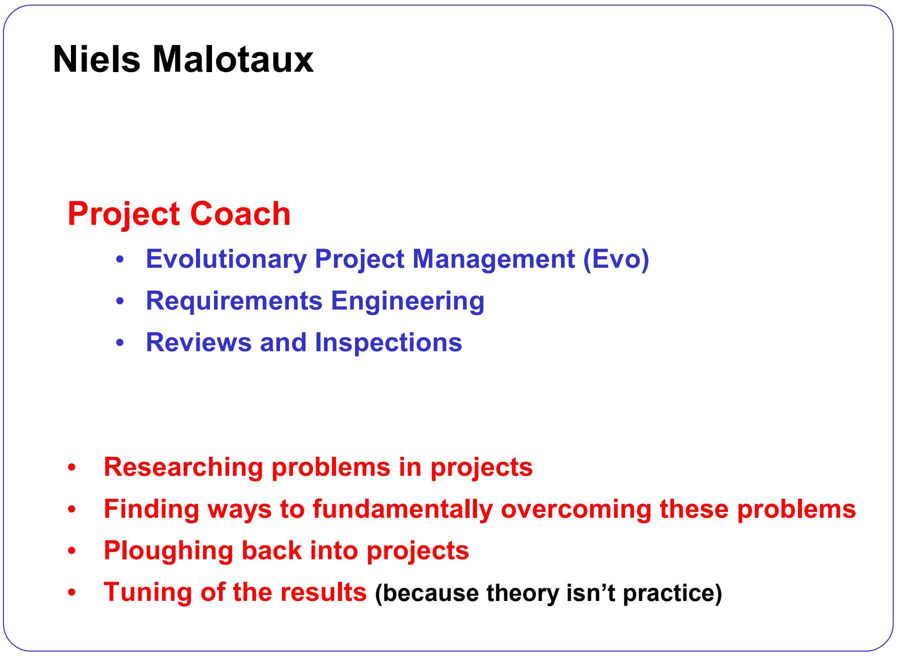
Reviews

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

1



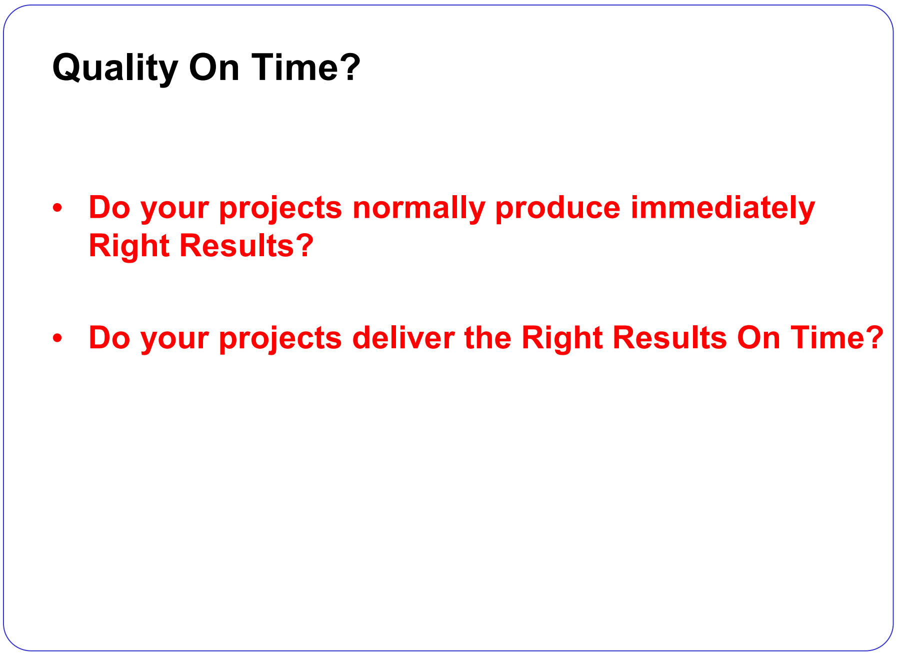
Niels Malotaux

Project Coach

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

- **Researching problems in projects**
- **Finding ways to fundamentally overcoming these problems**
- **Ploughing back into projects**
- **Tuning of the results (because theory isn't practice)**

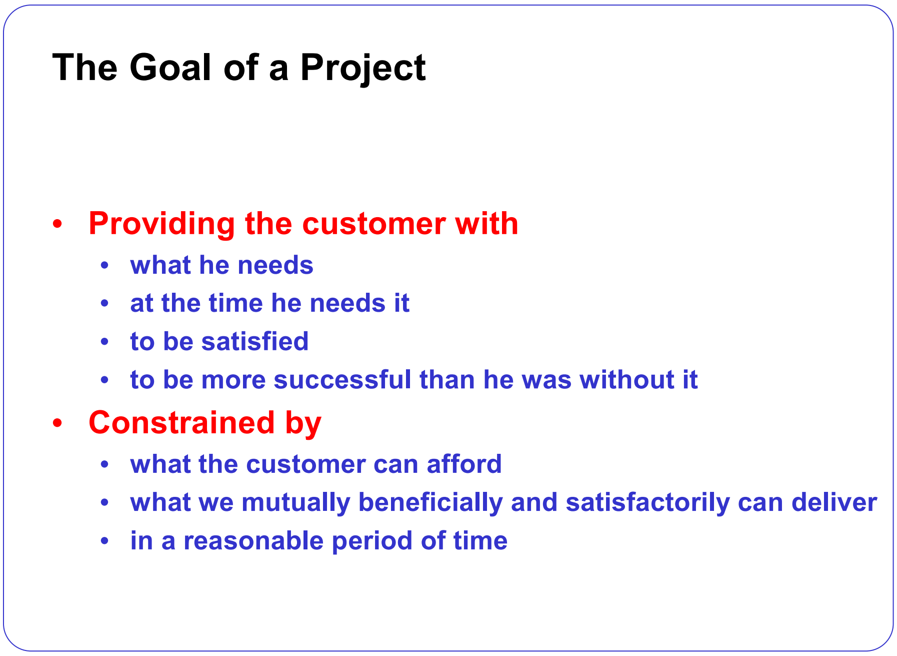
2



Quality On Time?

- Do your projects normally produce immediately Right Results?
- Do your projects deliver the Right Results On Time?

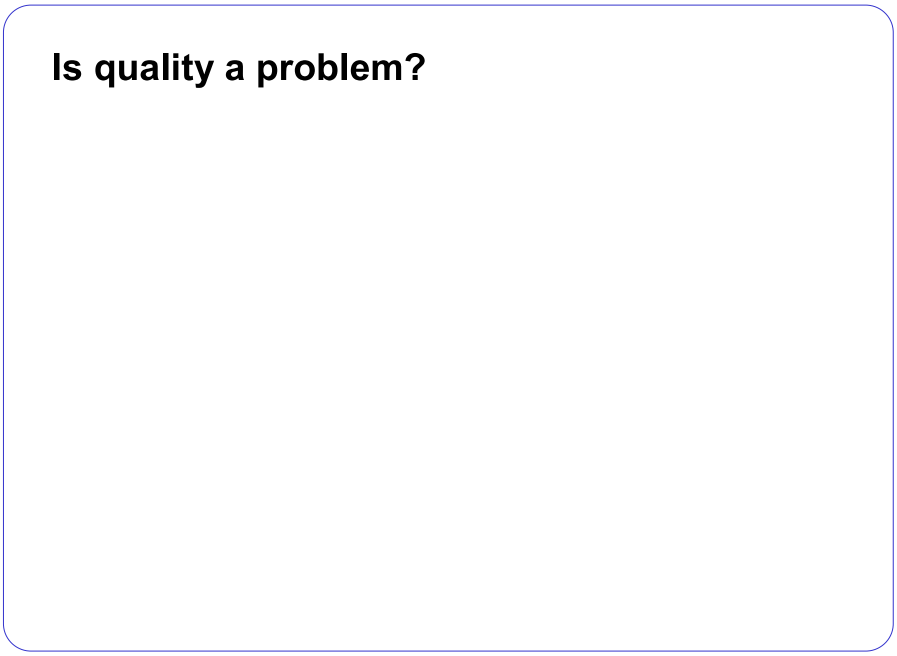
3



The Goal of a Project

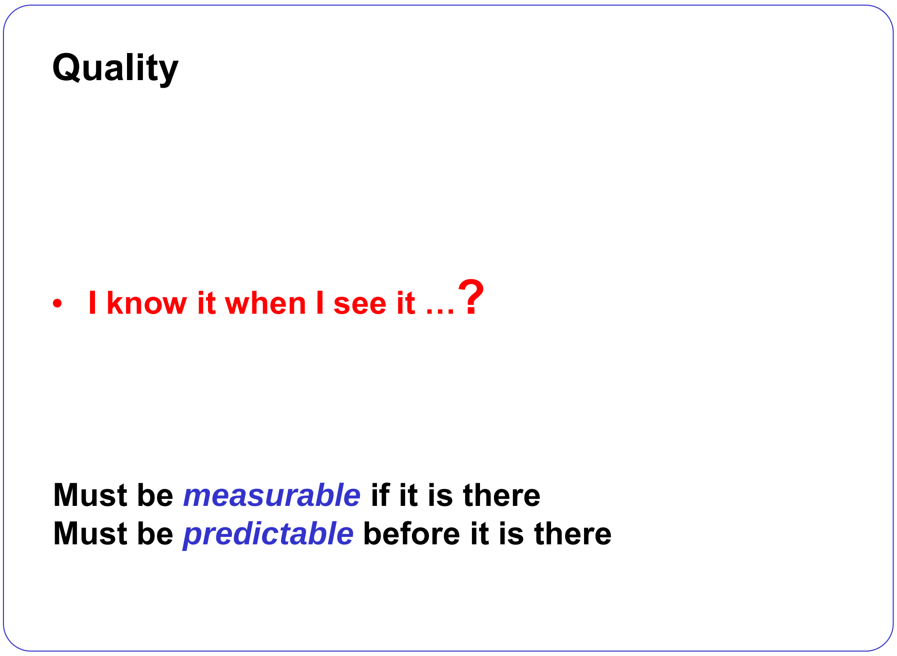
- Providing the customer with
  - what he needs
  - at the time he needs it
  - to be satisfied
  - to be more successful than he was without it
- Constrained by
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

4



Is quality a problem?

5



Quality

- I know it when I see it …?

Must be *measurable* if it is there
Must be *predictable* before it is there

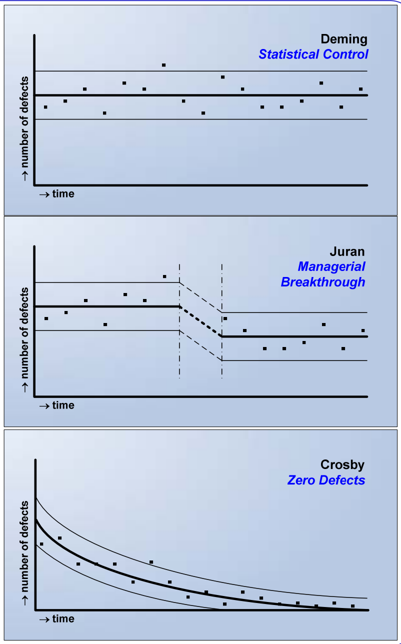
6

Product quality

- **Shewhart** - Economic Control of Quality 1930
- **Deming** - Japan 1949, Out of the crisis 1986
- **Juran** - Japan 1950, Quality handbook 1951
- **Crosby** - Zero Defects 1961, Quality is Free 1979

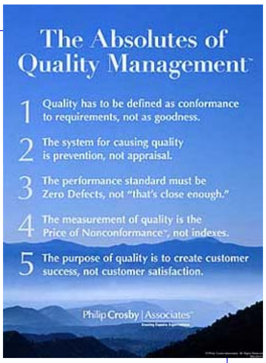
7

Deming - Juran - Crosby



8

Absolutes of Quality



The Absolutes of Quality Management™

1 Quality has to be defined as conformance to requirements, not as goodness.

2 The system for causing quality is prevention, not appraisal.

3 The performance standard must be Zero Defects, not "that's close enough."

4 The measurement of quality is the Price of Nonconformance™, not indexes.

5 The purpose of quality is to create customer success, not customer satisfaction.

Philip Crosby | Associates™

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

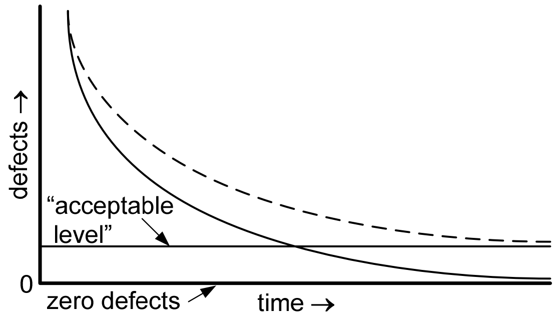
Philip Crosby, 1970

- The purpose is customer success
  (not customer satisfaction)

Added by Philip Crosby Associates, 2004

Is defect free software possible?

- Zero Defects is an asymptote



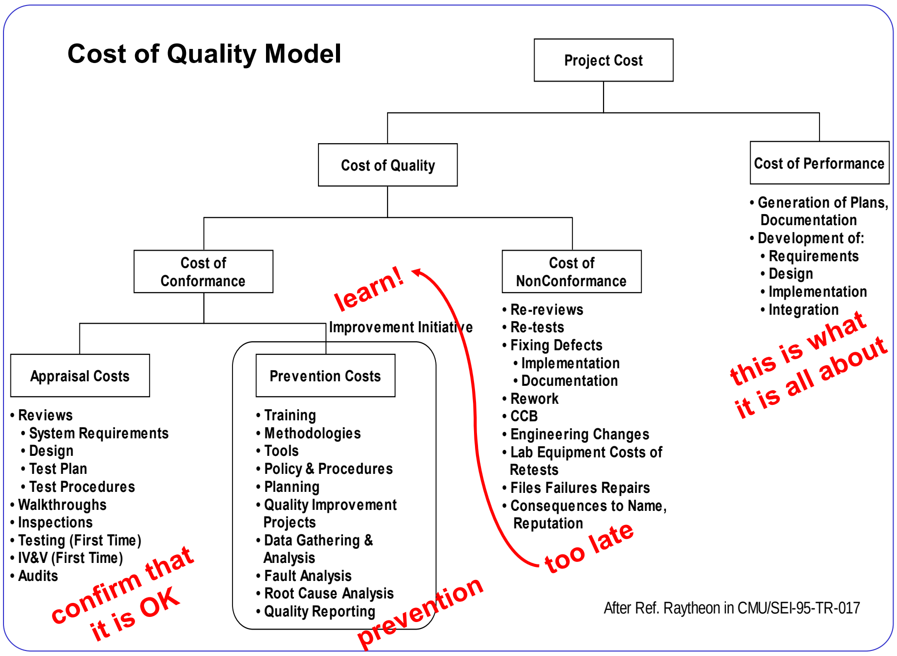
- When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately

Attitude

- As long as we think defect free software 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's OK and we'll be highly surprised when there still proves to be a defect
- We do all we can to improve

11

Cost of Quality Model



Project Cost

Cost of Quality

Cost of Performance
- Generation of Plans, Documentation
- Development of:
  - Requirements
  - Design
  - Implementation
  - Integration

Cost of Conformance

Cost of NonConformance
- Re-reviews
- Re-tests
- Fixing Defects
  - Implementation
  - Documentation
- Rework
- CCB
- Engineering Changes
- Lab Equipment Costs of Retests
- Files Failures Repairs
- Consequences to Name, Reputation

Appraisal Costs
- Reviews
  - System Requirements
  - Design
  - Test Plan
  - Test Procedures
- Walkthroughs
- Inspections
- Testing (First Time)
- IV&V (First Time)
- Audits

Improvement Initiative

Prevention Costs
- Training
- Methodologies
- Tools
- Policy & Procedures
- Planning
- Quality Improvement Projects
- Data Gathering & Analysis
- Fault Analysis
- Root Cause Analysis
- Quality Reporting

learn!

this is what it is all about

too late

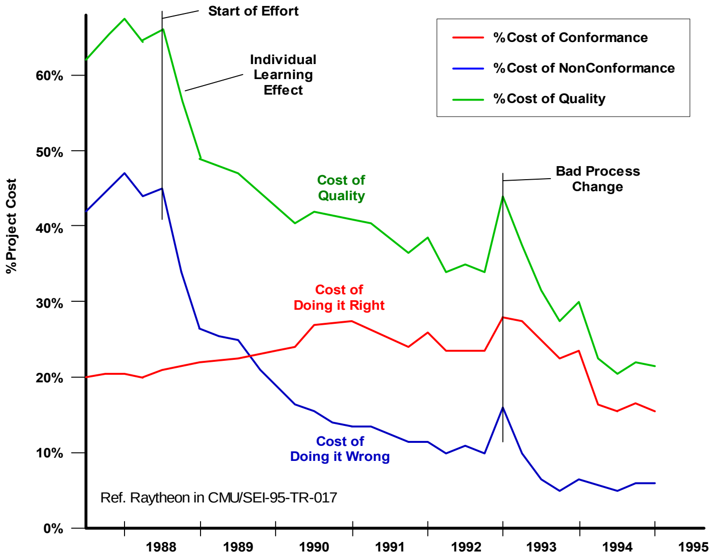
confirm that it is OK

prevention

After Ref. Raytheon in CMU/SEI-95-TR-017

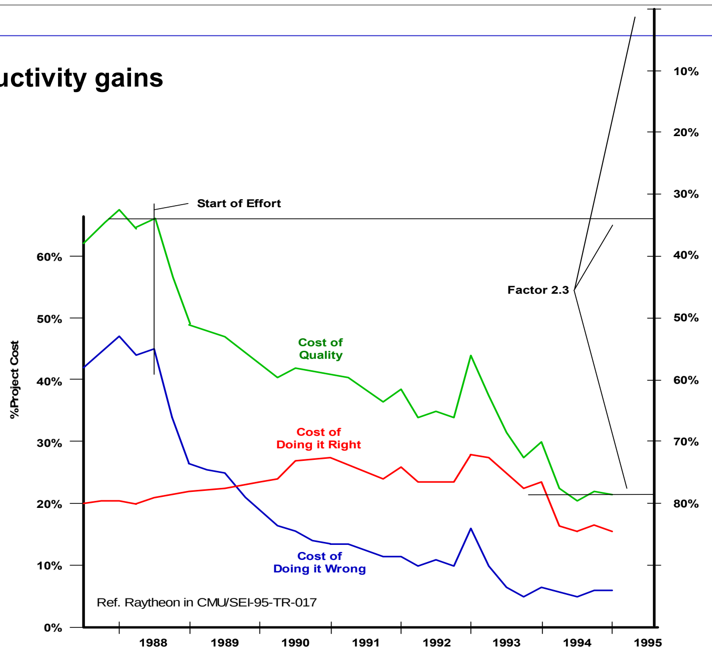
12

Cost of Quality



Ref. Raytheon in CMU/SEI-95-TR-017

Productivity gains



Ref. Raytheon in CMU/SEI-95-TR-017

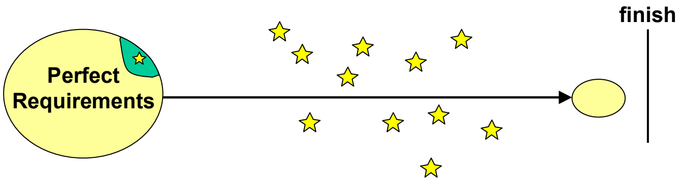
All we have to do ...

- A defect is the cause of a problem experienced by any of the stakeholders while relying on our results, ultimately affecting the customer
- 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?

15

Defects

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



16

Debugging???

17

The process of defect injection and detection

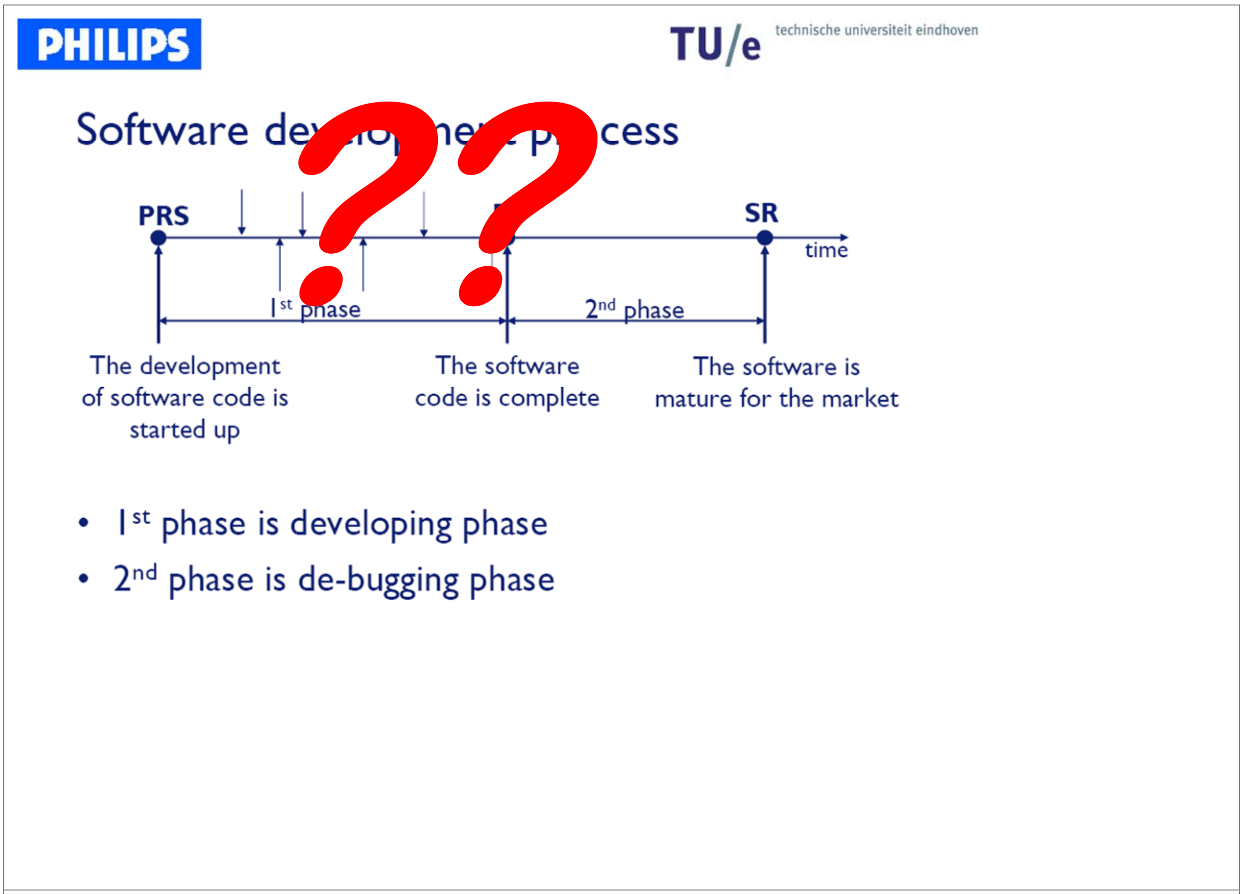
Conventional software development:

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

Can't we do better, or are we already doing things better?

**Real Engineering is
doing (most) things First Time Right**

18



PHILIPS

TU/e technische universiteit eindhoven

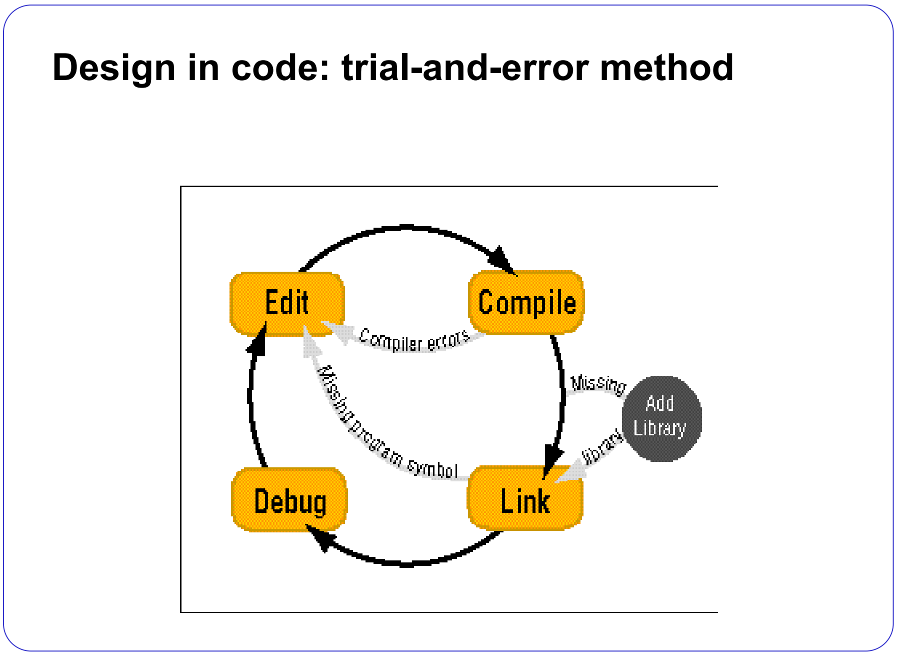
Software de[illegible]cess

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



Design in code: trial-and-error method

20

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

Defects found are symptoms of deeper problems

Repairing apparent defects creates several risks:

- Repair is done under pressure
- We think the problem is solved
- We introduce scars
- We keep repeating the same problems
- After finding the real cause, the redesign may make the repair redundant: time lost

Defects typically overlooked

- **Functions that won’t be used** (superfluous requirements)
  What’s the use of repairing defects in the code of these requirements?
- **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?

Are defects a problem for you?

- **Which types of defects?**
- **How do you know?**
- **Perhaps there are problems you don’t know?**
- **What can we do about it?**

Ways to achieve quality in software ?

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

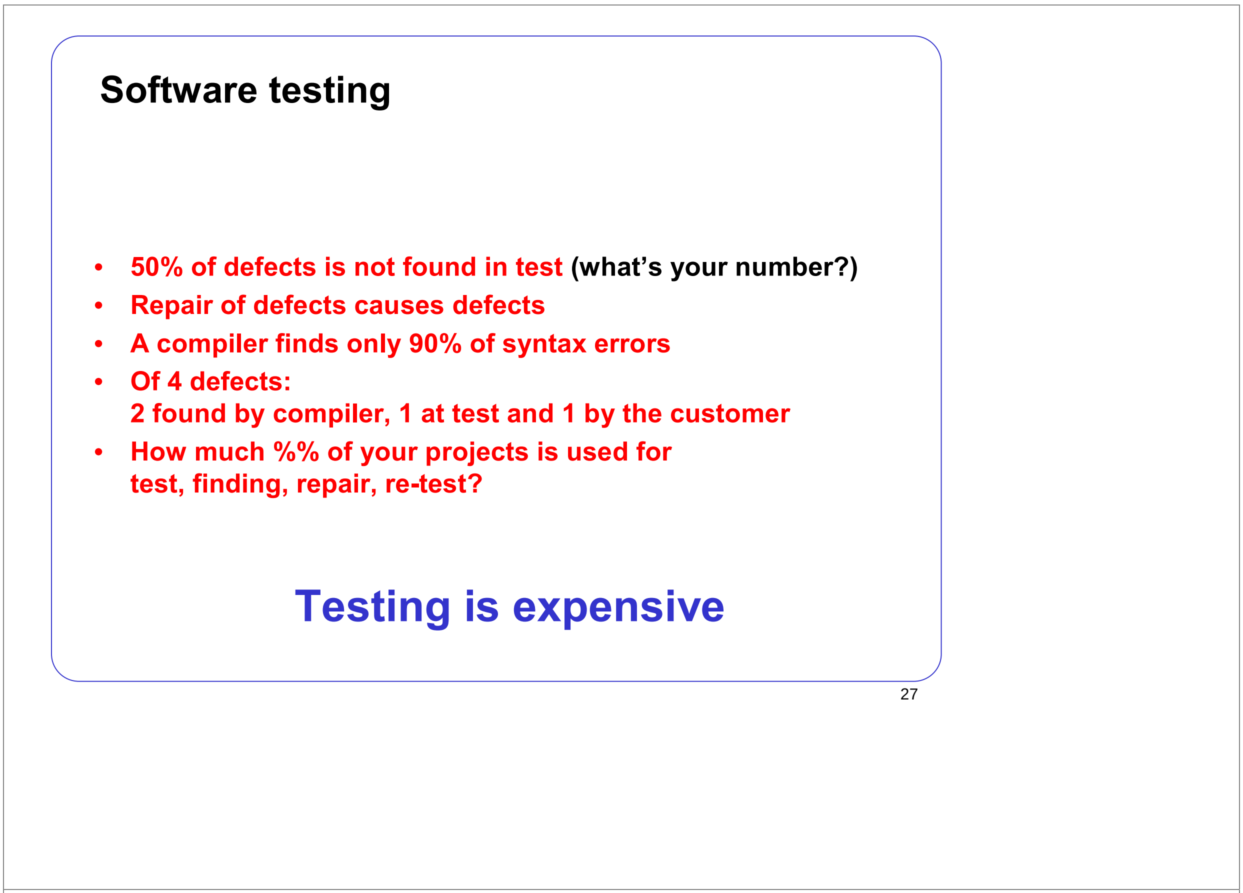
Prevention

25

Deming

- Quality comes not from inspection, but from improvement of the production process
- Inspection (testing) does not improve quality, nor guarantee quality
- Inspection is too late
- The quality, good or bad, is already in the product
- You cannot inspect quality into a product

26

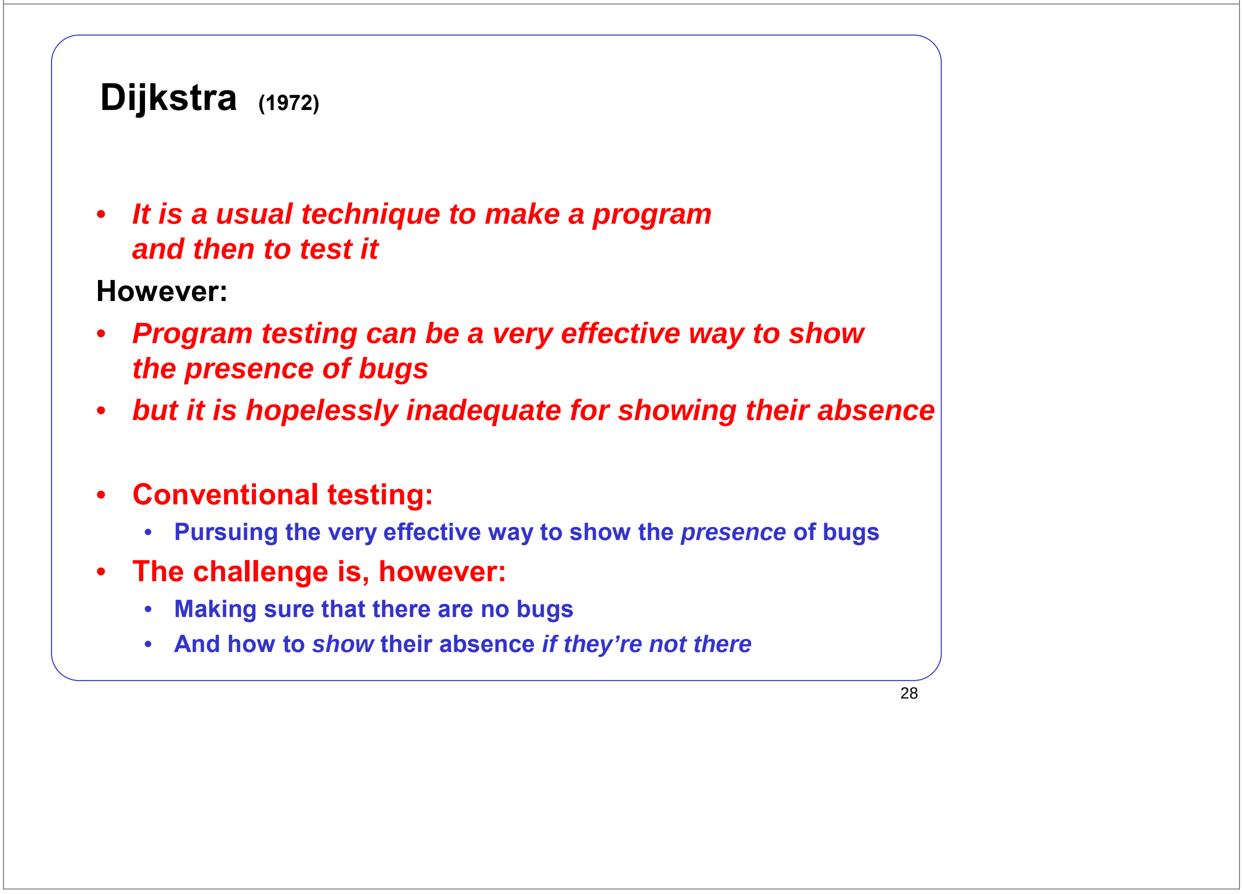


Software testing

- 50% of defects is not found in test (what's your number?)
- 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?

Testing is expensive

27



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 bugs
  - And how to *show* their absence *if they're not there*

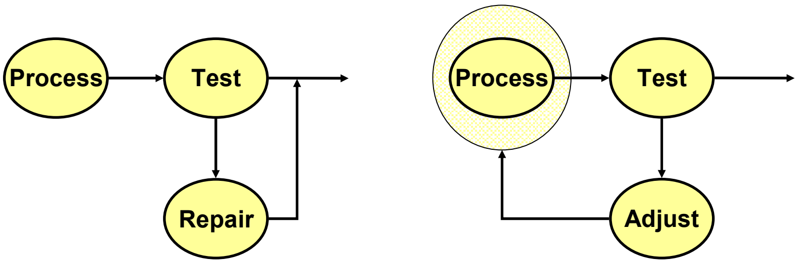
28

So, no testing?

- Testing is important
  however
- Goal should *not* be defect finding
- But rather measuring the quality of the production process

Testing is to check that it works correctly

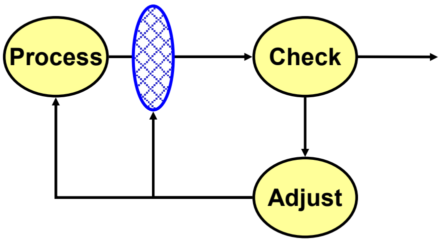
Testing is checking correctness



Usual

Should be

Testing is checking correctness



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

Let's move

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

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

Do you ever make a mistake?

- **People make mistakes**
- **We are people**

If we think we are done
there are still defects

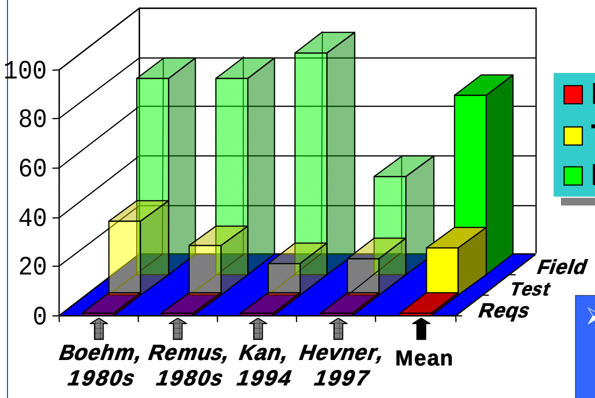
33

Costs of defects

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

34

Cost of Requirements Defects



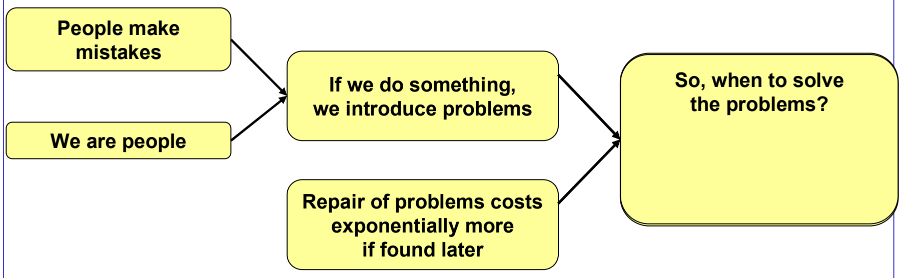
🟥 Reqs	$200
🟨 Test	$2,600 – $6,000
🟩 Field	$8,000 – $18,000

- *On average, finding a Requirements bug in Testing is 20 times as expensive.*
- *In field use: 75 times!*

DM

35

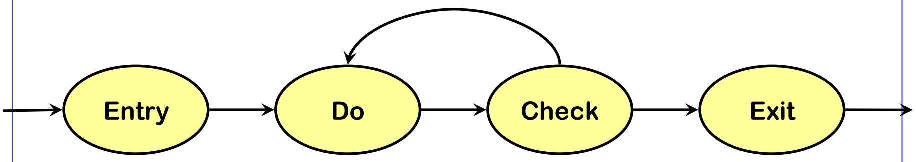
Inevitable consequence


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?

- Especially with safety and security issues, we don't want problems to happen
- We *have to* prevent
- During development we may let the problem happen, to learn, if we make sure that no real harm is done (simulation?)
- We may even *have to* let it happen, to check what really happens and whether our cure really works
- How else do we know that our assumptions are right?

36

Checking immediately after



Are you already doing this?

37

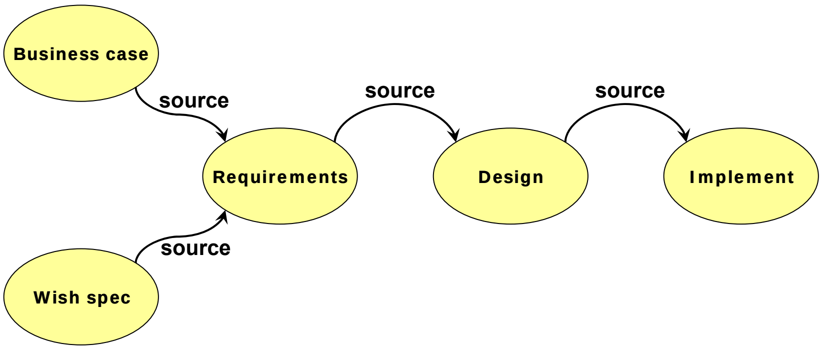
Exercise 1

- Review one page of one of your documents the way you are used to do the review
- Write down how you are doing it
- What are the results?

38

Simple Rule for Reviews

“We don’t review unless there is a source document”



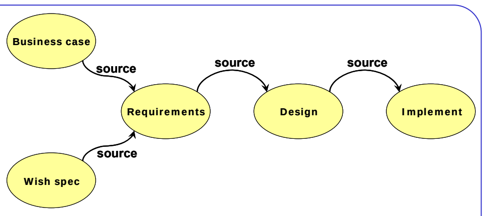
39

Exercise 2

- Review the same page with the “Source Document” rule
- What are the results?

40

Rules

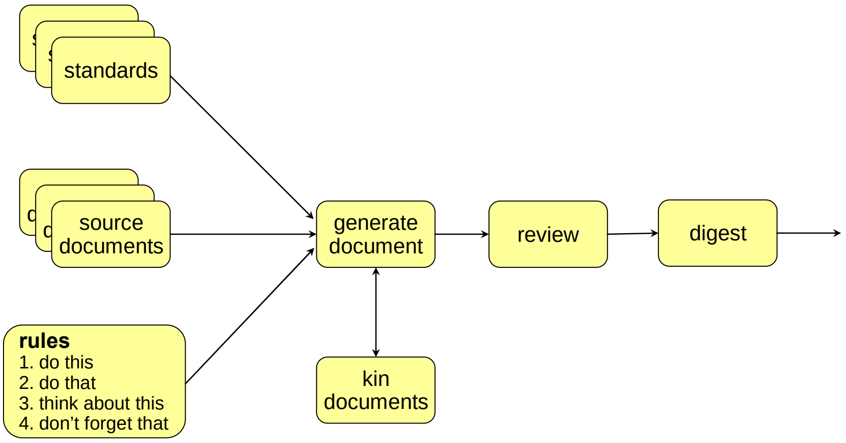


- Any workproduct will be reviewed against
  - Itself
  - Kin documents
  - Source documents

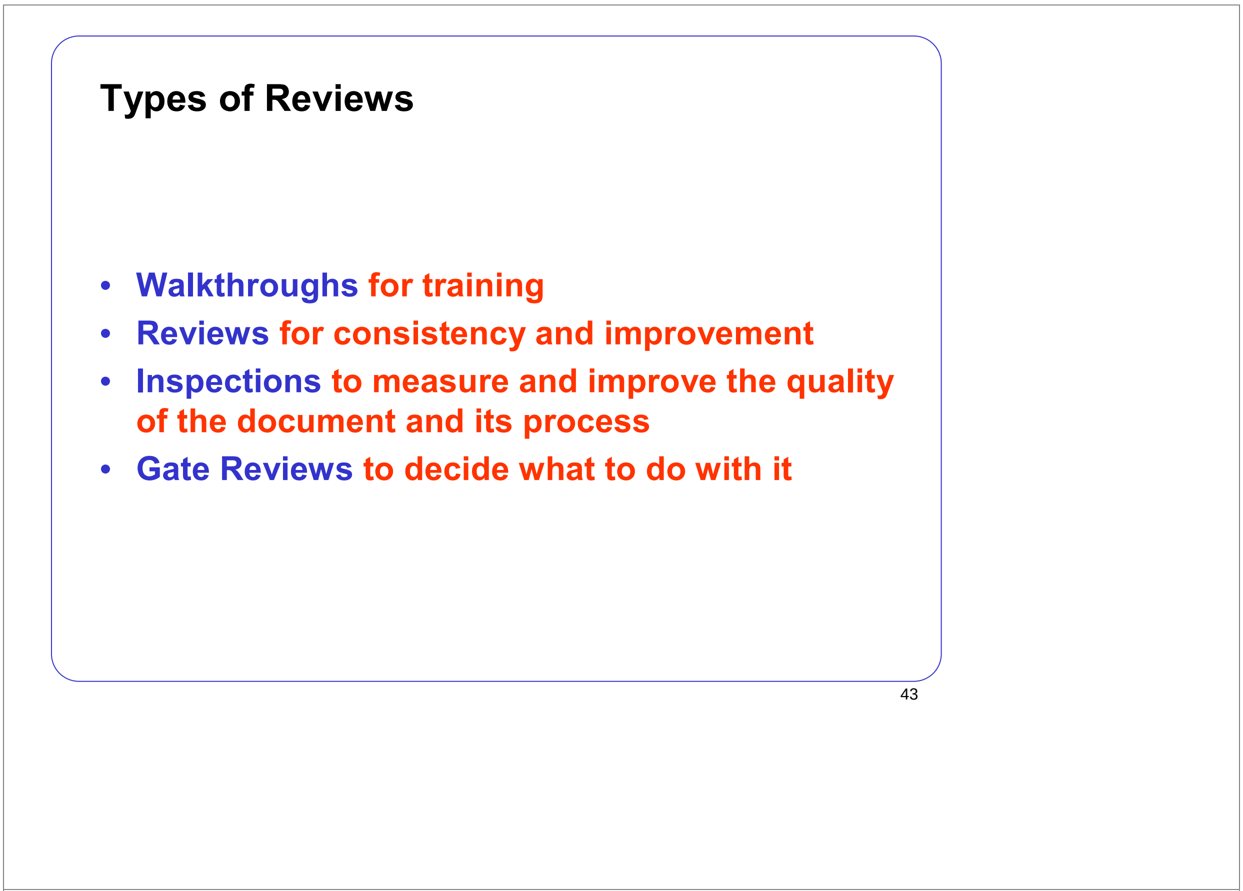
  If we don't have the source, how can we judge the workproduct?
- We always update the source document first before changing the workproduct(s)
  - First change the Design, then the Code and the Test
  - First change the Requirement, then the Design, then the Code and the Test

41

Document generation



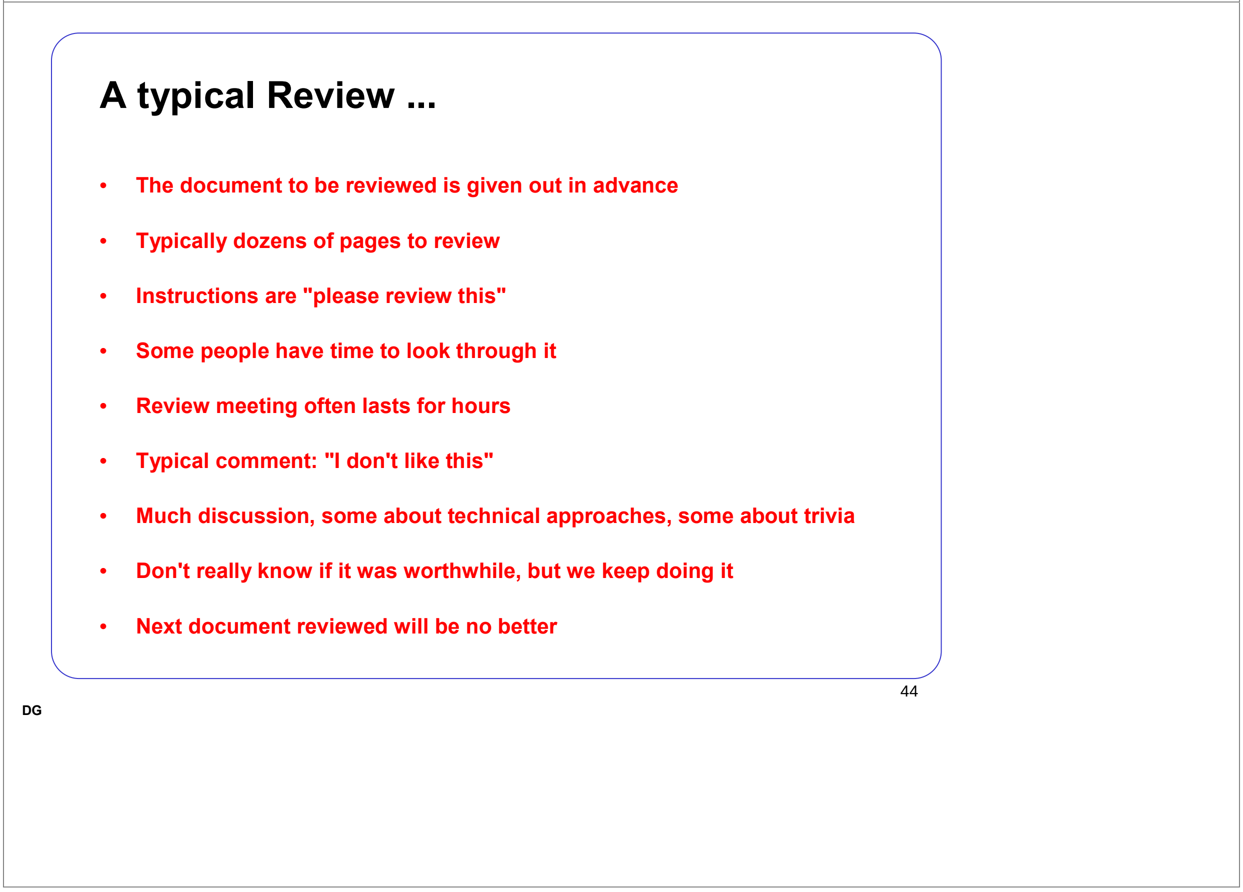
42



Types of Reviews

- Walkthroughs for training
- Reviews for consistency and improvement
- Inspections to measure and improve the quality of the document and its process
- Gate Reviews to decide what to do with it

43



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

44

DG

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

45

DG

Rules

- Rules are the law for documents
- Defect = Rule violation

- 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

46

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

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

47

TG

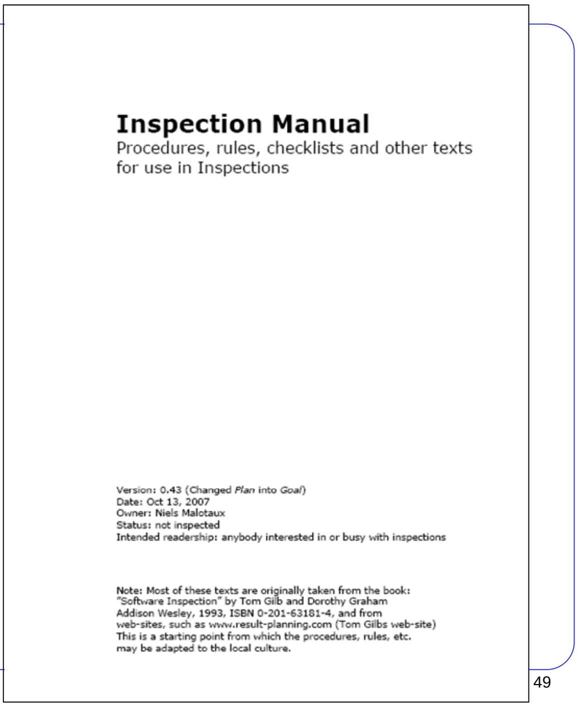
Use the three rules on this Requirement

It shall be possible to easily extend the system's functionality
on a modular basis, to implement specific (e.g. local) functionality

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

48

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.

49

Choosing the top 3 Rules (Kommsi)

- Three most effective rules
  - cover 60–80% of problems
  - different top 3 Rules for different projects
- Most economical to inspect issues that:
  - support the goal of the product
  - are likely to harm the project in the current development phase
- Rules by risk analysis on documents

Source: Marko Komssi, Increasing responsiveness and economy of Software Inspection, EuroStar 2004

50

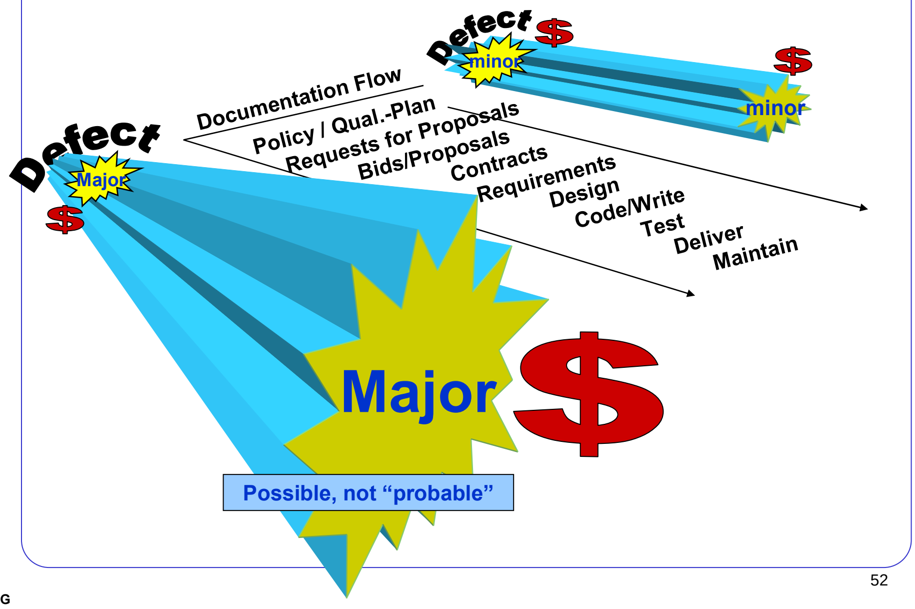
DG

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

51

Major ↔ minor Severity Concepts



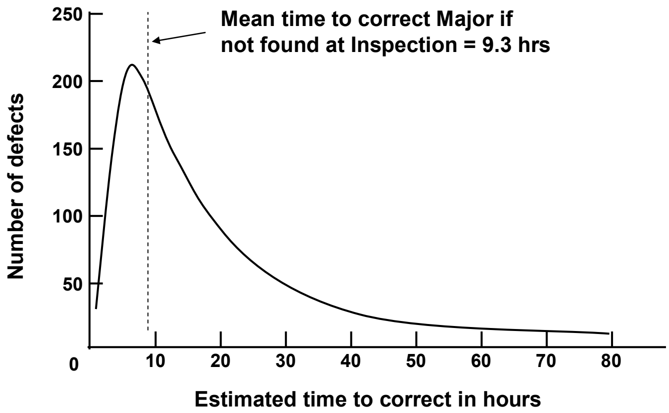
Possible, not "probable"

52

TG

Cost of Quality

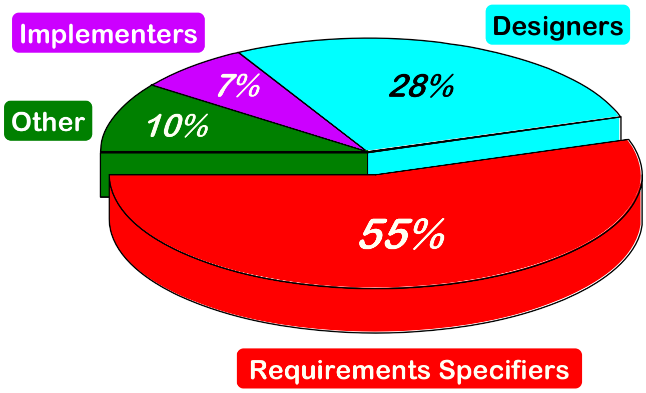
ref SI, fig 14.6, p315



Mean time to correct Major if not found at Inspection = 9.3 hrs

53

Typical Defect Injectors (*cost* breakdown)

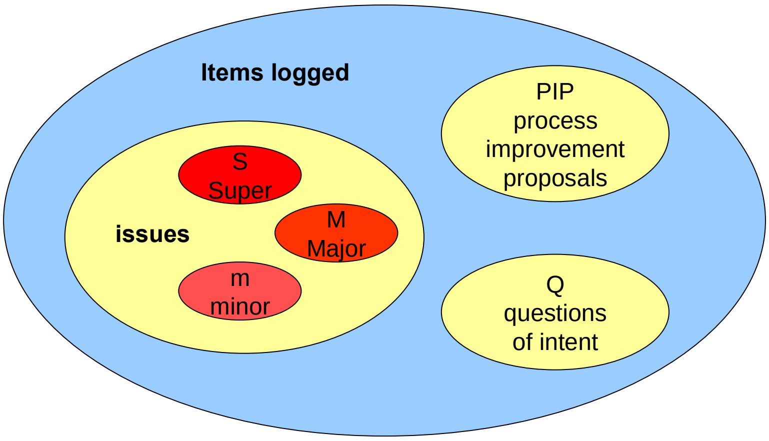


After Bender Associates, *1996*

54

DM

What are we looking for?



55

Optimum Checking Rate

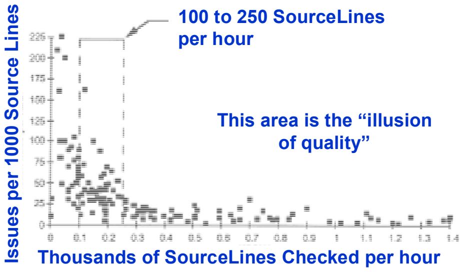
Optimum Checking Rate

- The most **Effective** individual speed for 'checking a document against all related documents' in page/hr or in hr/page

Notes

- Not 'reading' speed
- 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")



100 to 250 SourceLines per hour

This area is the "illusion of quality"

Raytheon, CMU/SEI-95-TR017

56

TG

Ref. Dorothy Graham

At first glance ..

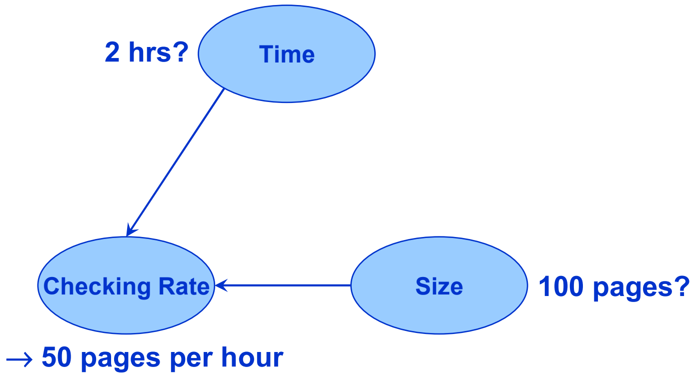
Here's a document: review this (or Inspect it)

57

DG

Ref. Dorothy Graham

Time and size determine rate



2 hrs?

Time

Checking Rate

Size

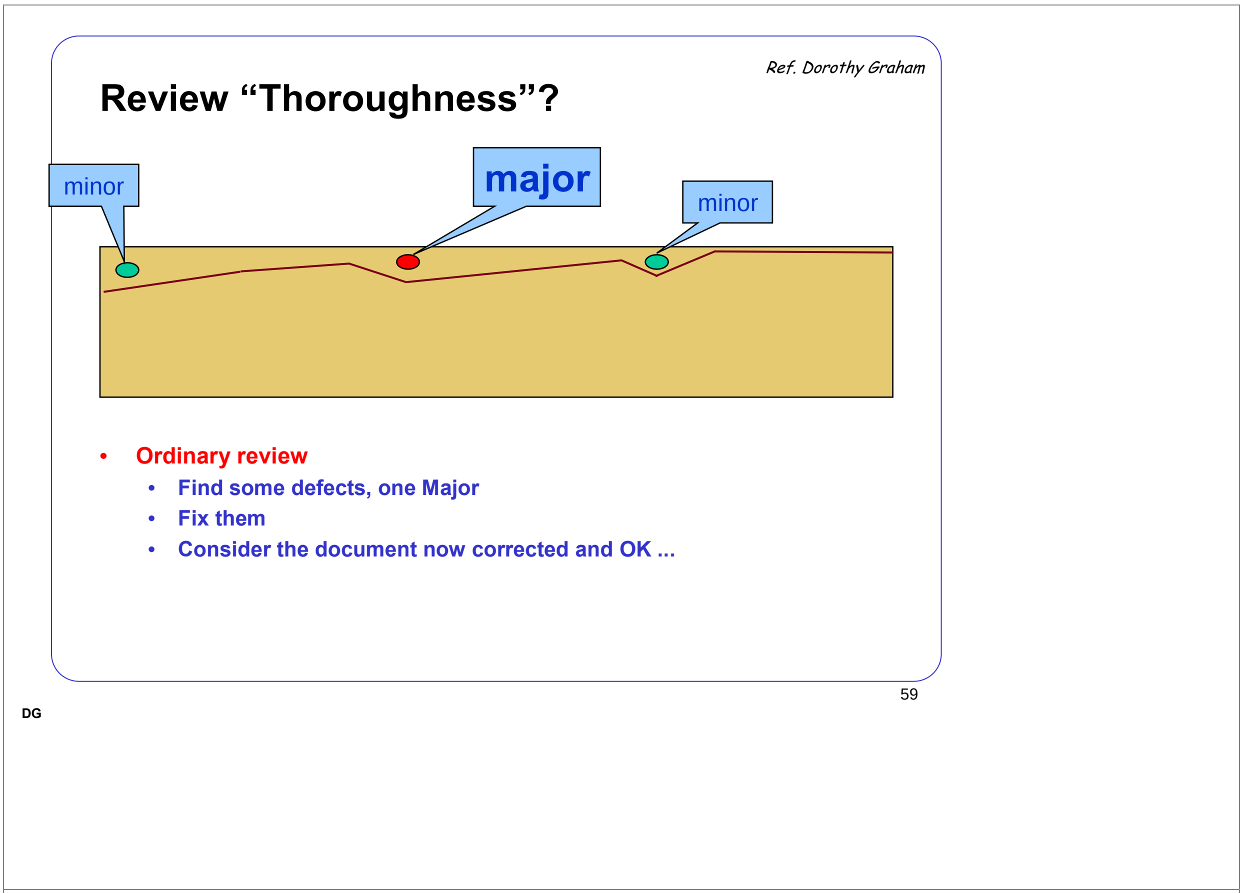
100 pages?

→ 50 pages per hour

Reviews: time and size determine rate

58

DG



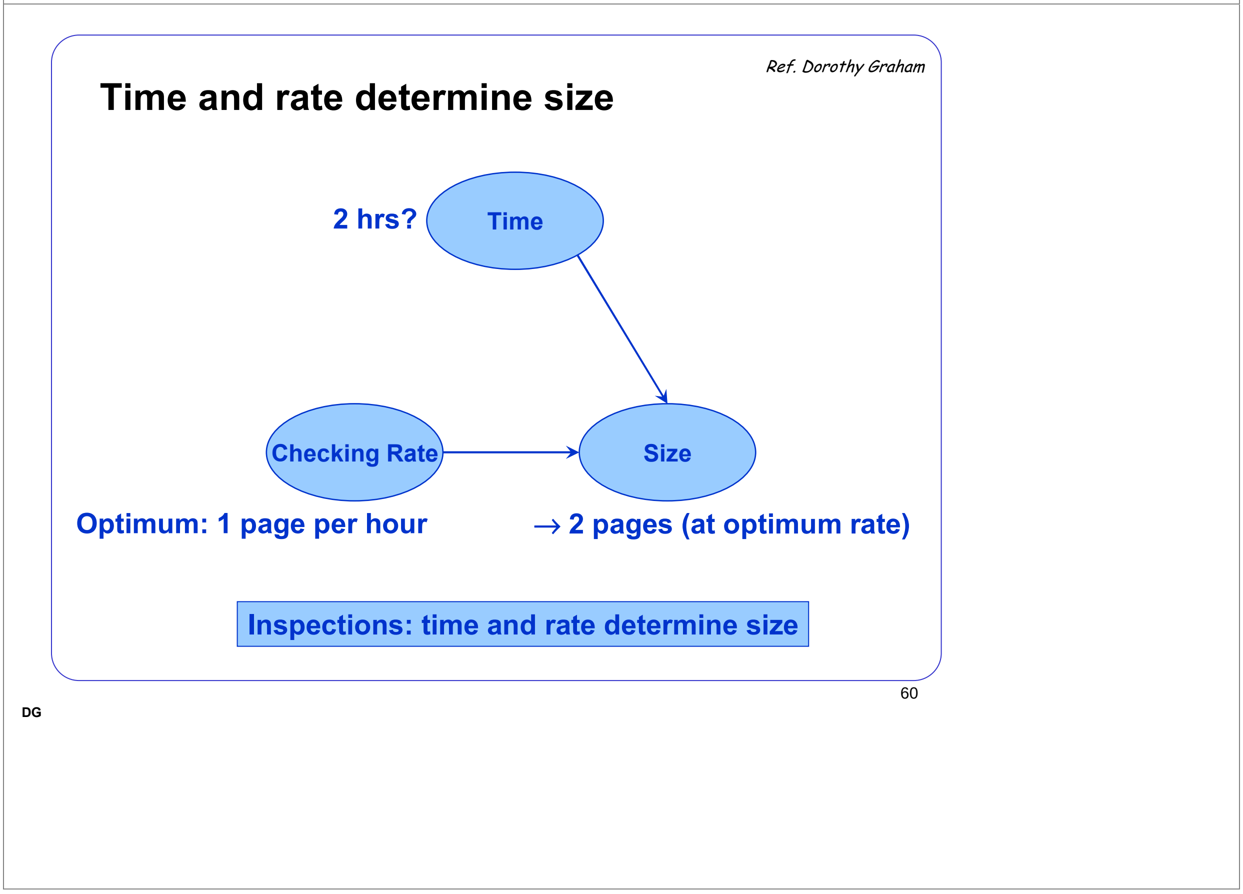
Ref. Dorothy Graham

Review “Thoroughness”?

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

DG

59



Ref. Dorothy Graham

Time and rate determine size

2 hrs? Time

Checking Rate → Size

Optimum: 1 page per hour

→ 2 pages (at optimum rate)

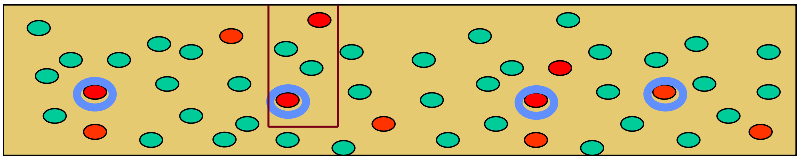
Inspections: time and rate determine size

DG

60

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 *!*

61

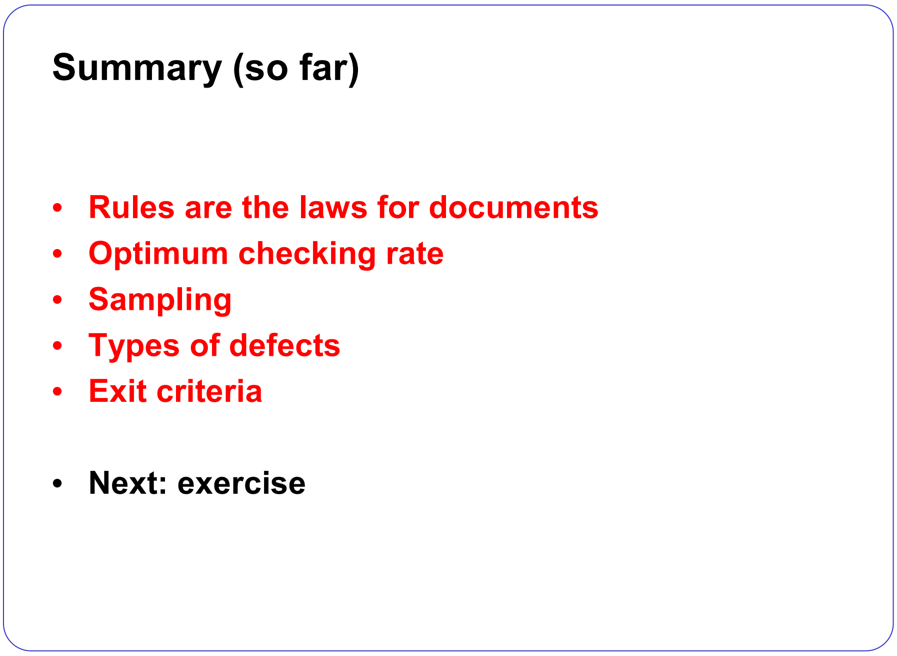
DG

Exit criteria: estimating remaining majors (after fixing)

- You are about to Inspect your own document
- What is acceptable exit level? Is it OK to have:
  - 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

62

(DG)

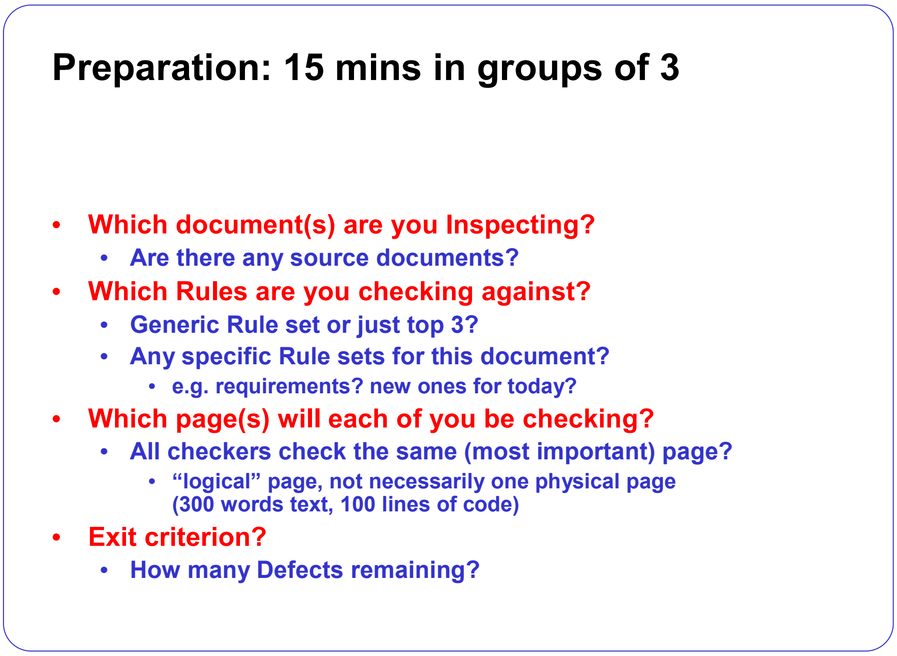


Summary (so far)

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

- Next: exercise

63

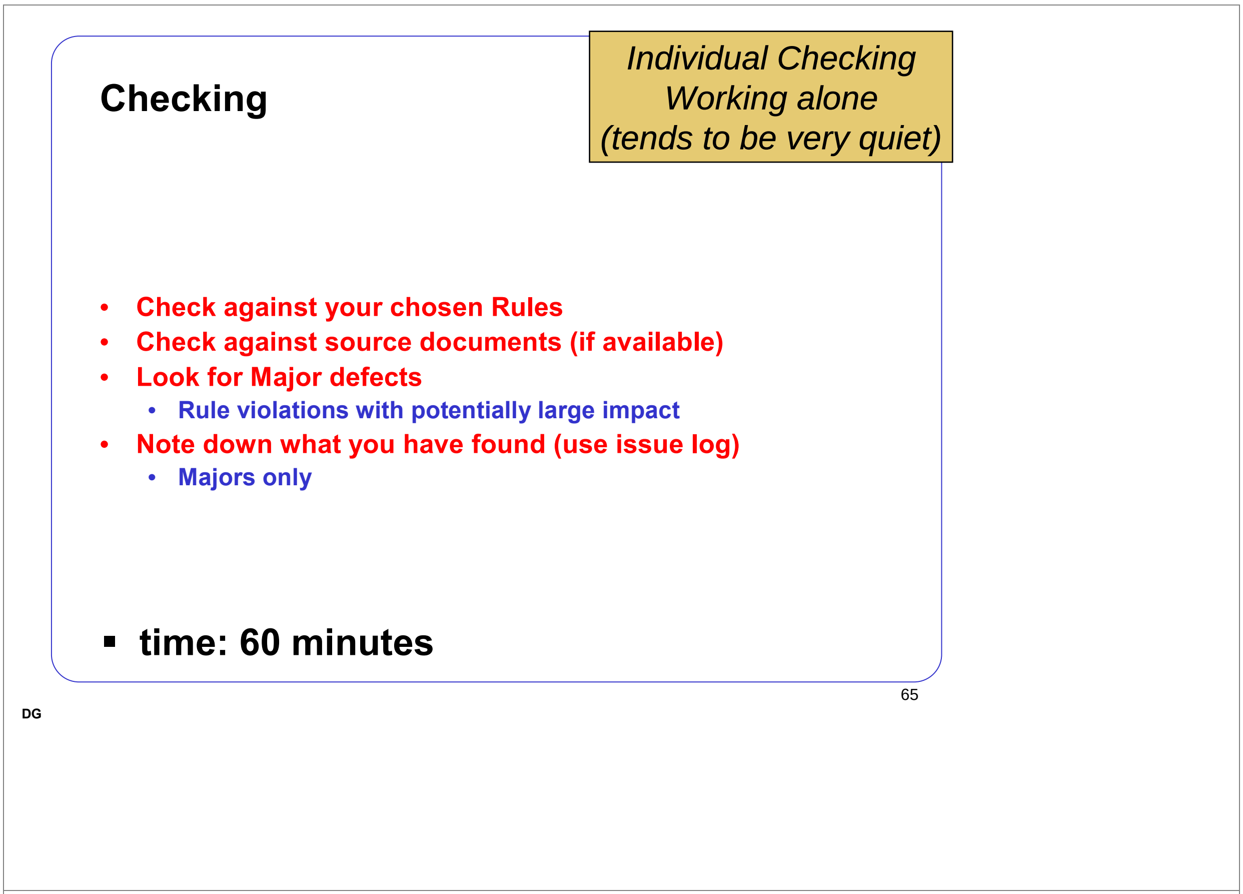


Preparation: 15 mins in groups of 3

- 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?

64

DG



Checking

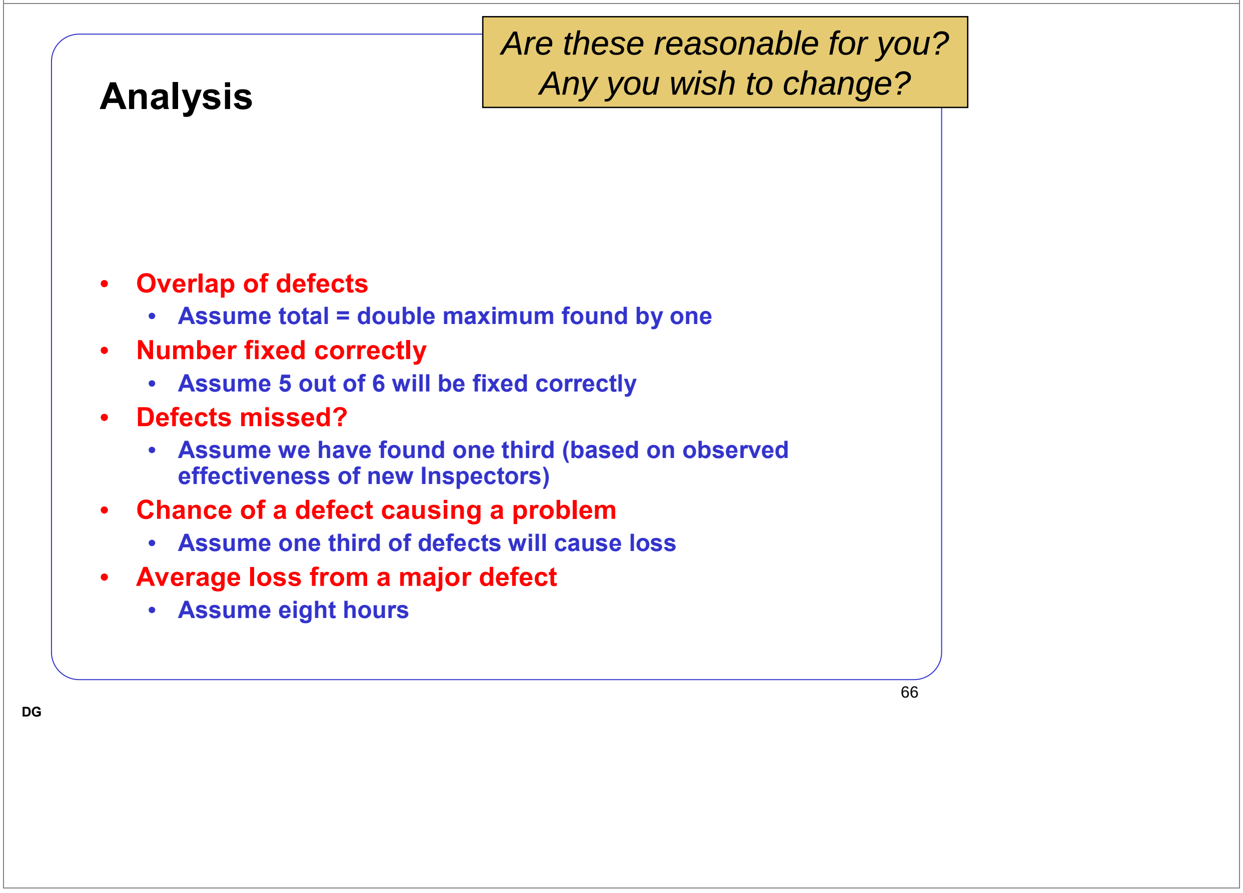
Individual Checking
Working alone
(tends to be very quiet)

- 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

- time: 60 minutes

65

DG



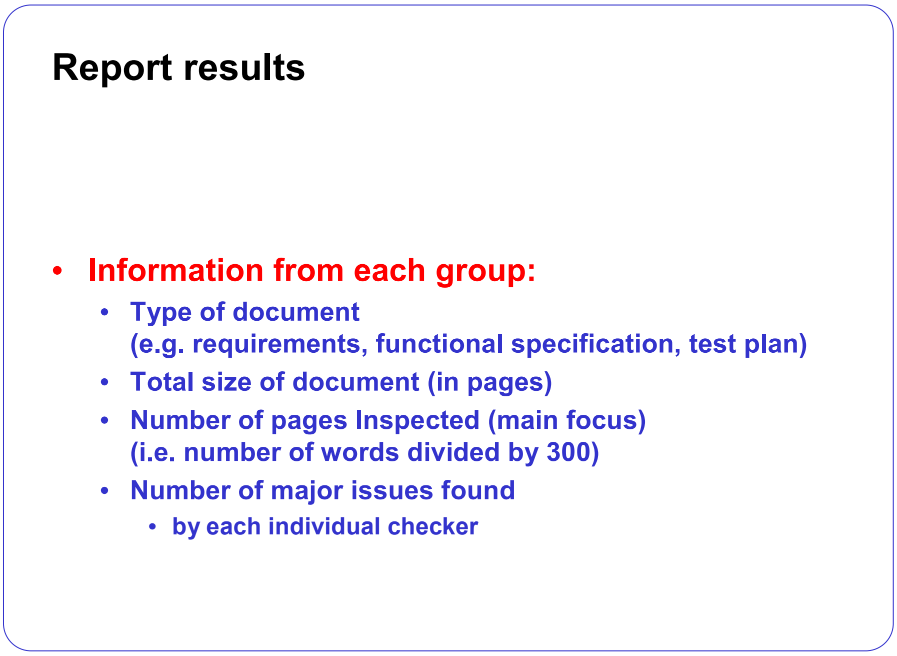
Analysis

Are these reasonable for you?
Any you wish to change?

- 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 eight hours

66

DG

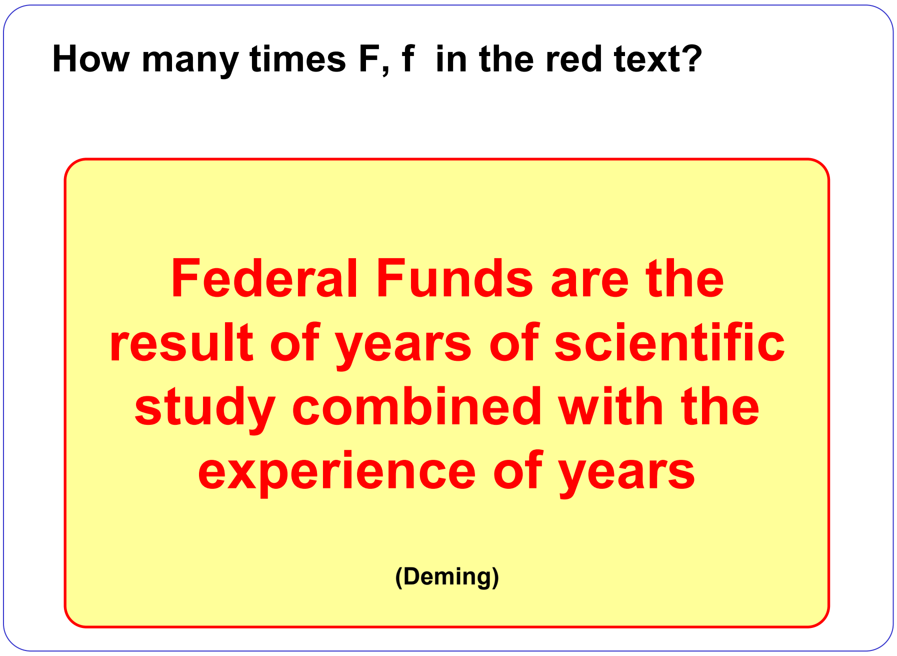


Report results

- Information from each group:
  - Type of document
    (e.g. requirements, functional specification, test plan)
  - 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

67

DG

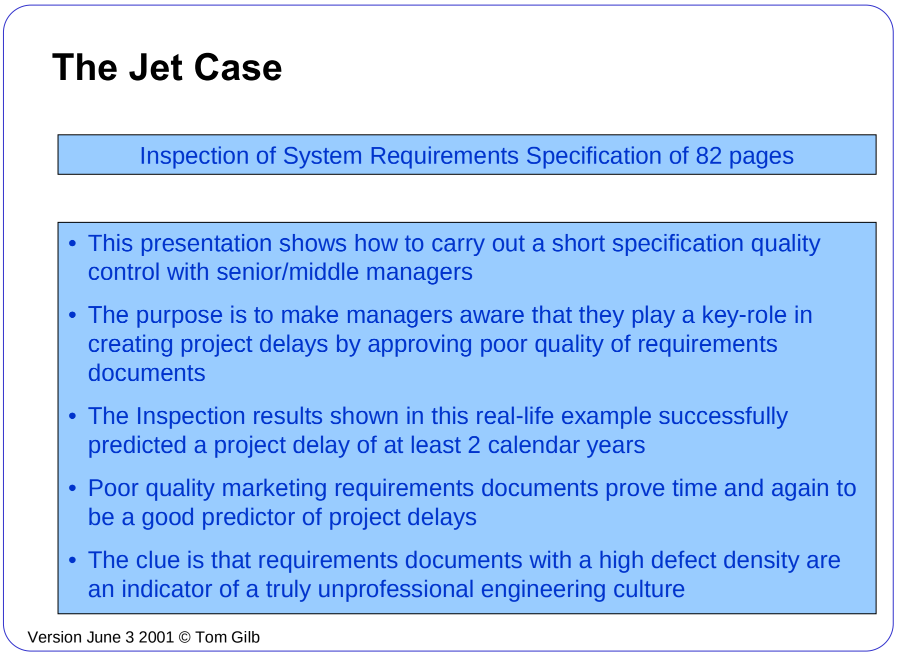


How many times F, f in the red text?

Federal Funds are the result of years of scientific study combined with the experience of years

(Deming)

68



The Jet Case

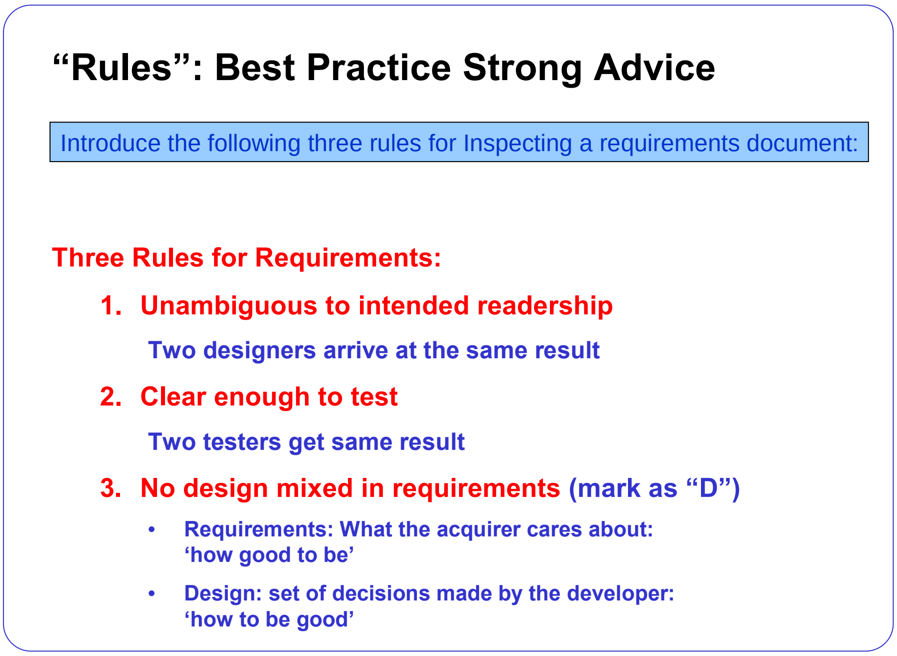
Inspection of System Requirements Specification of 82 pages

- This presentation shows how to carry out a short specification quality control with senior/middle managers
- The purpose is to make managers aware that they play a key-role in creating project delays by approving poor quality of requirements documents
- The Inspection results shown in this real-life example successfully predicted a project delay of at least 2 calendar years
- Poor quality marketing requirements documents prove time and again to be a good predictor of project delays
- The clue is that requirements documents with a high defect density are an indicator of a truly unprofessional engineering culture

Version June 3 2001 © Tom Gilb

69

TG



“Rules”: Best Practice Strong Advice

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**

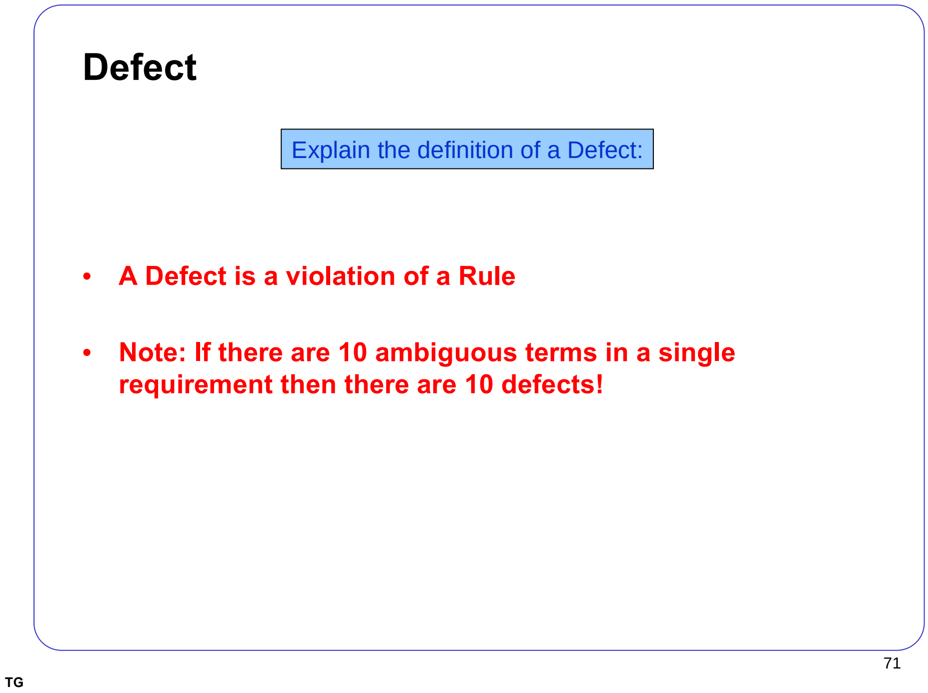
   Two testers get same result

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’**

70

TG



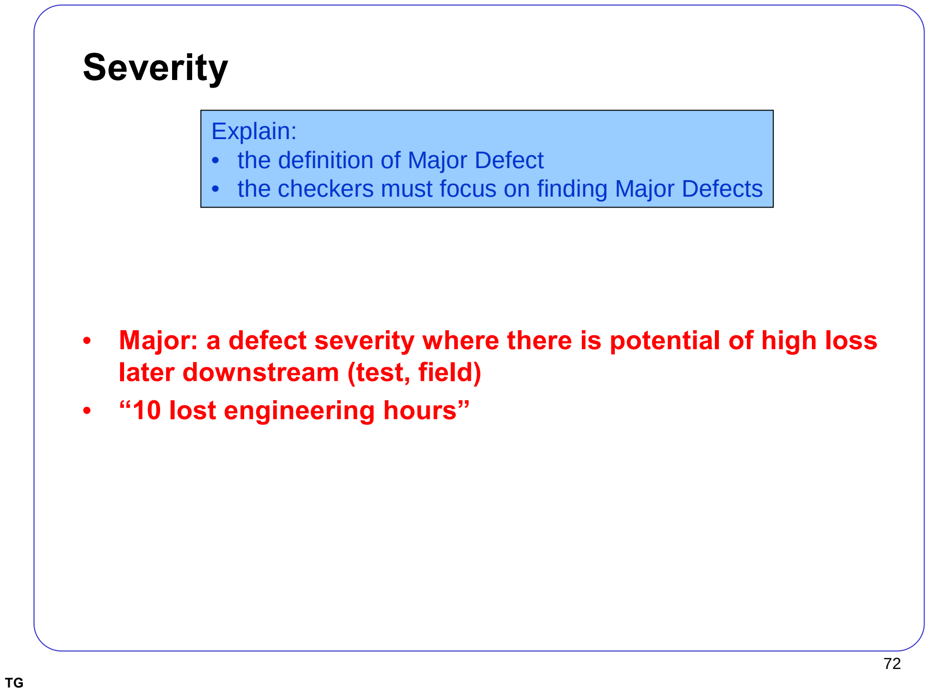
Defect

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!

TG

71



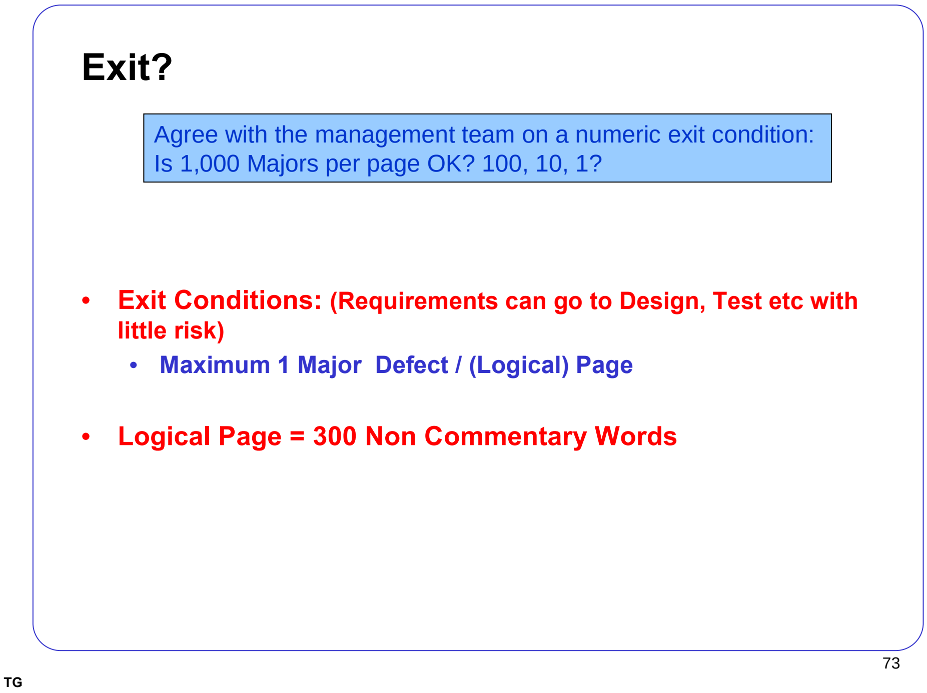
Severity

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”

TG

72



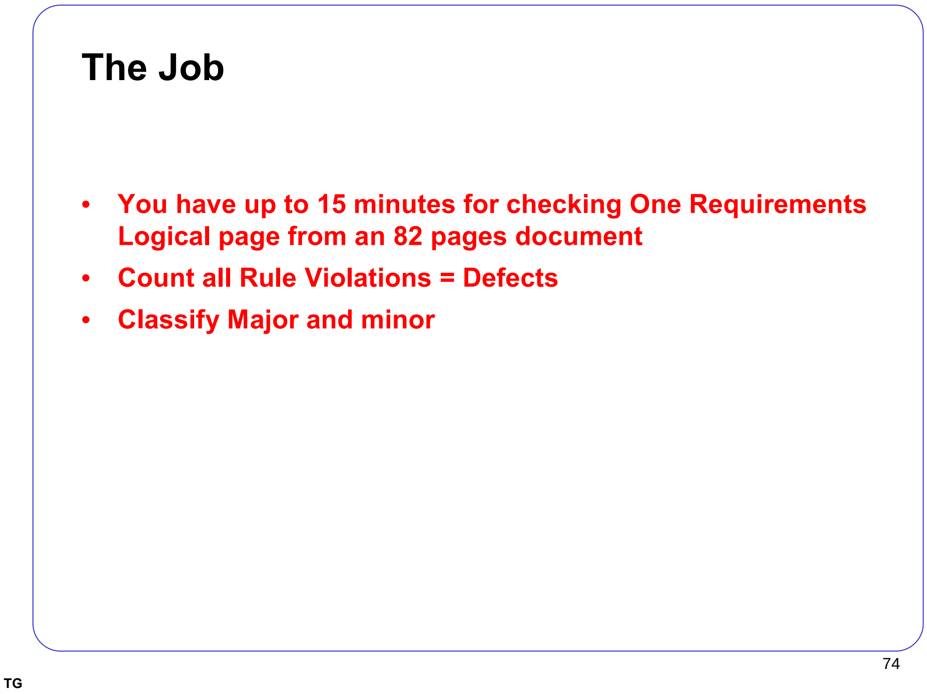
Exit?

Agree with the management team on a numeric exit condition:
Is 1,000 Majors per page OK? 100, 10, 1?

- **Exit Conditions: (Requirements can go to Design, Test etc with little risk)**
  - **Maximum 1 Major Defect / (Logical) Page**
- **Logical Page = 300 Non Commentary Words**

TG

73



The Job

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

TG

74

Report for Page 81

Inspection results on requirements document, 4 managers

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

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

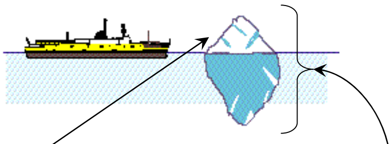
75

TG

Report for page 82

Inspection results on requirements document, 4 other managers

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

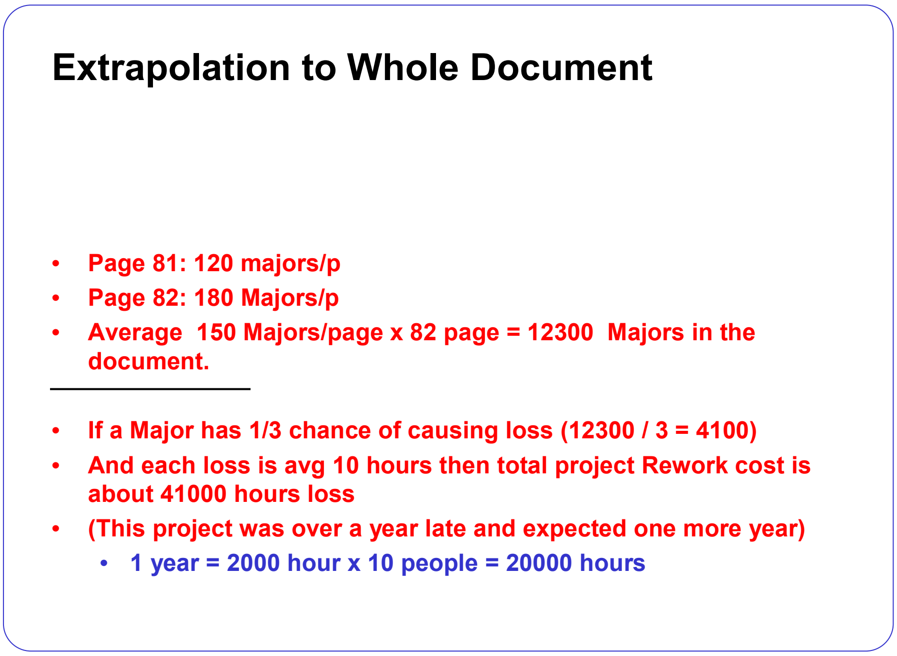


Defect Density

- 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:
- The total remaining after Inspection and editing = 10 + 120 = 130 Majors per page

76

TG



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

77

TG

Inspection

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

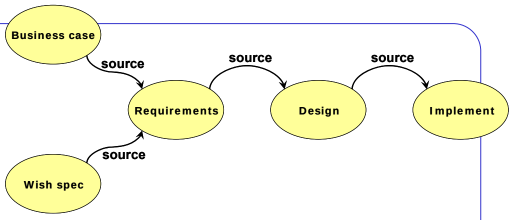
78

Types of documents

- Quotation
- Contract
- Requirements
- Architecture
- Design
- Software code
- Test plan
- Test scripts
- Electronic schematic diagram
- Mechanical drawing

79

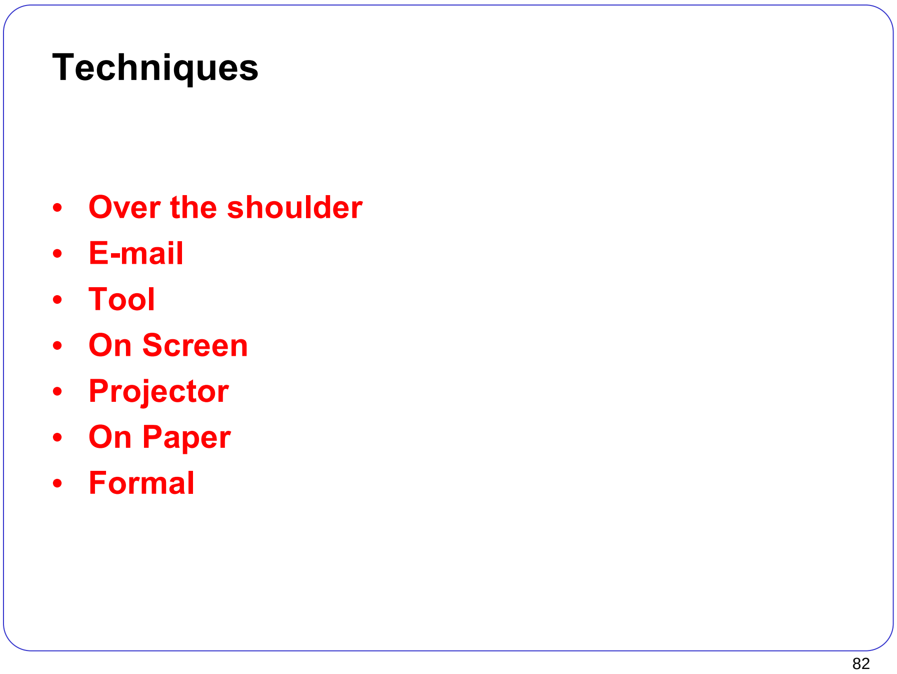
Documentation in a project



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

Many types of Review to choose from

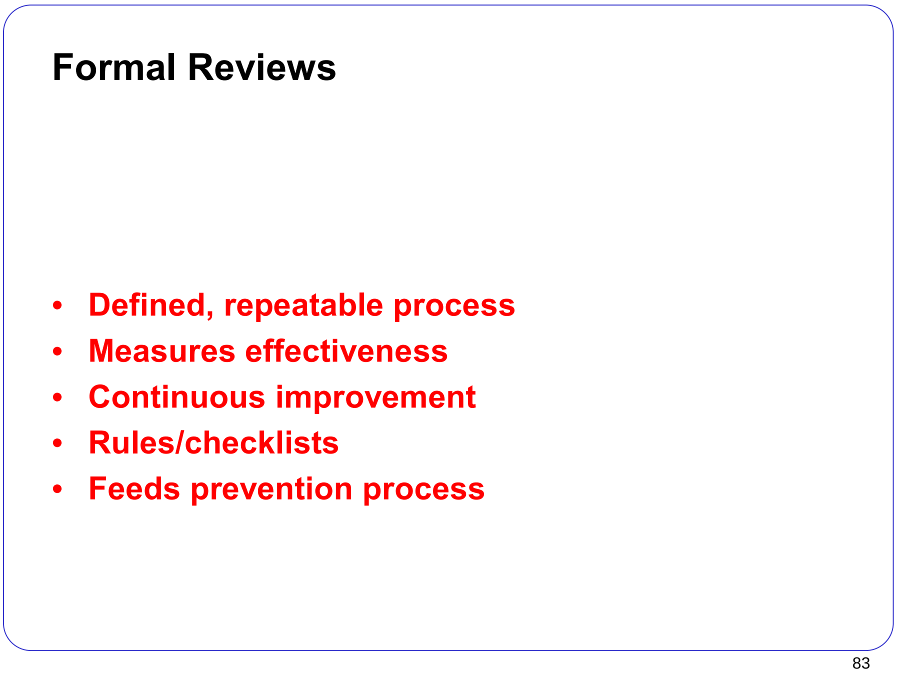
- Informal Review
- Pair Programming
- Technical Review
- Walkthrough
- Formal Inspection (Fagan type)
- Cleanroom Inspection
- Formal Inspection (Gilb/Graham type)
- Agile/Extreme/Lean Inspection
- SQC
- Gate Review
- Unit Test
- Debugging
- Test



Techniques

- Over the shoulder
- E-mail
- Tool
- On Screen
- Projector
- On Paper
- Formal

82



Formal Reviews

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

83

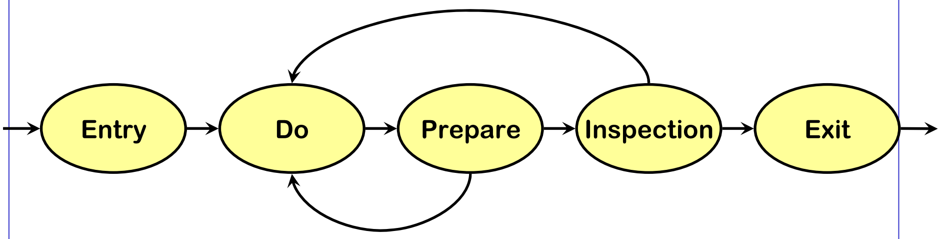
Inspection

- Most rigorous form of review
- Pioneered by Fagan (IBM) (paper 1976)
  - Locating all the defects in a work product
- Introduction of Inspection economics: Gilb/Graham (Software Inspection, 1993)
  - Quantifying the defect density of a work product and preventing poor quality work from moving downstream
- 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

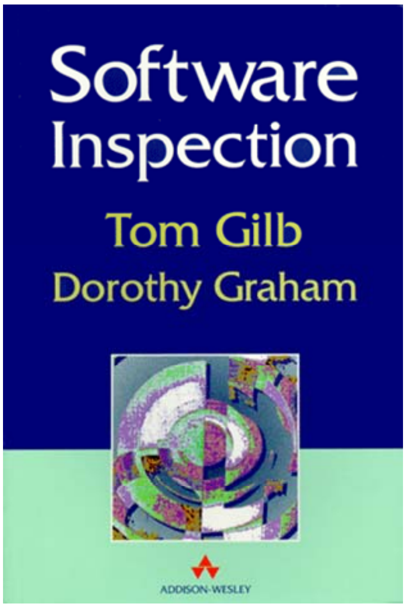
Would you like to base further work or decisions on a document of unknown quality?

84

Preparation for Code Inspection



85



Software Inspection

Tom Gilb
Dorothy Graham

ADDISON-WESLEY

A ready to use recipe …

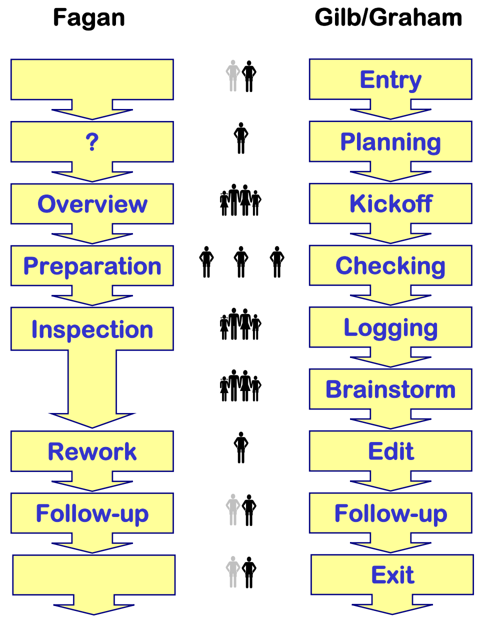
86

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 generate documents in the first place?
- Interesting side-effect:
  People get to know each others documents efficiently

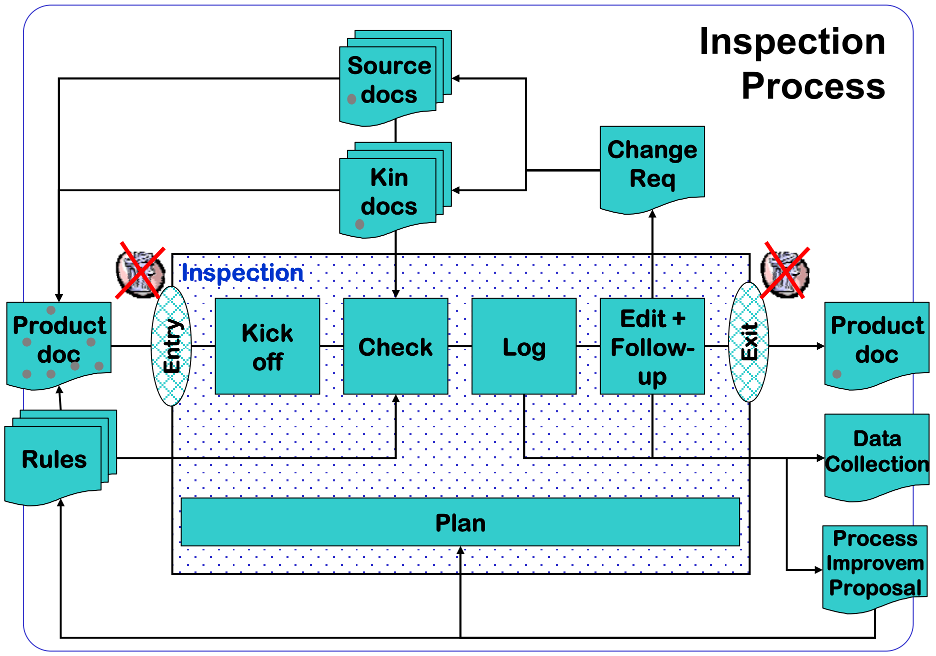
87

Inspection Process Steps



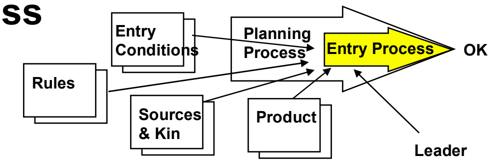
88

Inspection Process



89

Entry Process



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

- Purpose **Why?**
  - **To avoid continuing a costly process which is doomed to failure (no exit, product not released)**
  - **To permit correction of fail-prone conditions**
    - **before using time and people fruitlessly**
  - **To permit continuous process improvement by learning which entry conditions are worth checking**
- Organization **How!**
  - **The Team Leader checks all conditions in the generic and product-specific entry Conditions**
  - **Anyone can suggest improved Entry Conditions to process owner (Entry is part of process definition)**
  - **“Failed” entry conditions are dealt with (corrected, waived) before entering**

90

TG

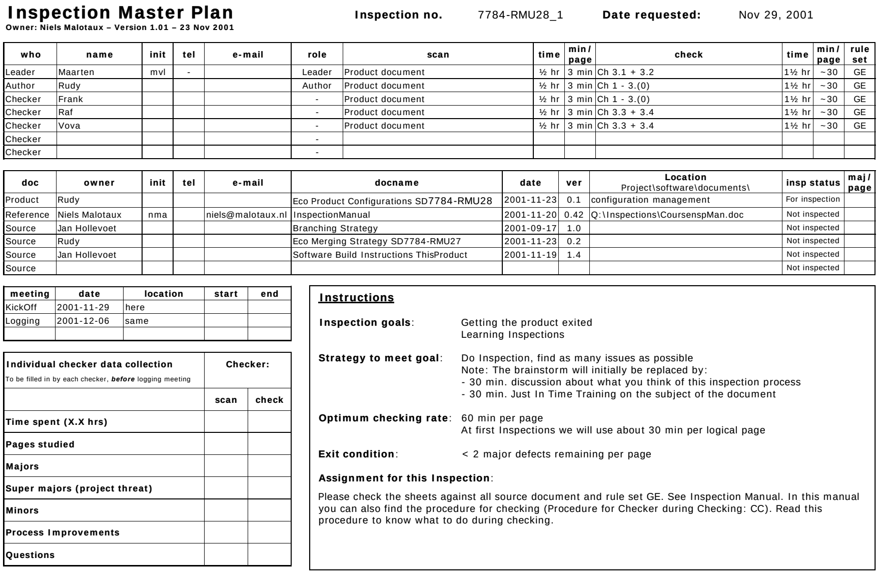
Generic Inspection Entry criteria

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

GEC1	**(author)**	**The author can decide not to enter any substage of inspection**
GEC2	**(leader)**	**The leader can decide not to enter any substage of inspection**
GEC3	**(writing)**	**All source documents are in writing and successfully exited**
GEC4	**(rules)**	**Generic and specific rule sets for the task are available in writing**
GEC5	**(plan)**	**A master plan has been made with checking rate of one page per hour**
GEC6	**(trained)**	**The leader has been trained and certified as Inspection leader**
GEC7	**(exam)**	**A cursory (< 5 min) examination of a sample shows < 1 major/page**
GEC8	**(checks)**	**Possible machine checks are done**
GEC9	**(participate)**	**The author agrees to participate as checker**

91

TG



Inspection Master Plan

Owner: Niels Malotaux – Version 1.01 – 23 Nov 2001

Inspection no. 7784-RMU28_1 **Date requested:** Nov 29, 2001

who	name	init	tel	e-mail	role	scan	time	min / page	check	time	min / page	rule set
Leader	Maarten	mvl	-		Leader	Product document	½ hr	3 min	Ch 3.1 + 3.2	1½ hr	~30	GE
Author	Rudy				Author	Product document	½ hr	3 min	Ch 1 - 3.(0)	1½ hr	~30	GE
Checker	Frank				-	Product document	½ hr	3 min	Ch 1 - 3.(0)	1½ hr	~30	GE
Checker	Raf				-	Product document	½ hr	3 min	Ch 3.3 + 3.4	1½ hr	~30	GE
Checker	Vova				-	Product document	½ hr	3 min	Ch 3.3 + 3.4	1½ hr	~30	GE
Checker					-							
Checker					-							

doc	owner	init	tel	e-mail	docname	date	ver	Location Project\software\documents\	insp status	maj / page
Product	Rudy				Eco Product Configurations SD7784-RMU28	2001-11-23	0.1	configuration management	For inspection	
Reference	Niels Malotaux	nma		niels@malotaux.nl	InspectionManual	2001-11-20	0.42	Q:\Inspections\CoursenspMan.doc	Not inspected	
Source	Jan Hollevoet				Branching Strategy	2001-09-17	1.0		Not inspected	
Source	Rudy				Eco Merging Strategy SD7784-RMU27	2001-11-23	0.2		Not inspected	
Source	Jan Hollevoet				Software Build Instructions ThisProduct	2001-11-19	1.4		Not inspected	
Source									Not inspected	

meeting	date	location	start	end
KickOff	2001-11-29	here		
Logging	2001-12-06	same		

Individual checker data collection To be filled in by each checker, ***before*** logging meeting	**Checker:**	
	scan	**check**
Time spent (X.X hrs)		
Pages studied		
Majors		
Super majors (project threat)		
Minors		
Process Improvements		
Questions		

Instructions

Inspection goals: Getting the product exited
Learning Inspections

Strategy to meet goal: Do Inspection, find as many issues as possible
Note: The brainstorm will initially be replaced by:
- 30 min. discussion about what you think of this inspection process
- 30 min. Just In Time Training on the subject of the document

Optimum checking rate: 60 min per page
At first Inspections we will use about 30 min per logical page

Exit condition: < 2 major defects remaining per page

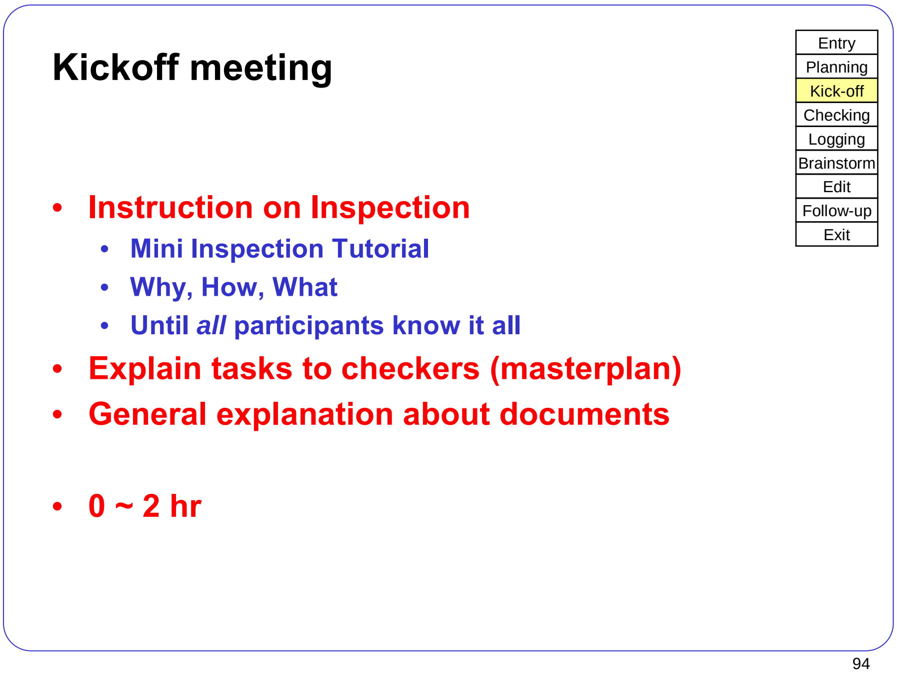
Assignment for this Inspection:

Please check the sheets against all source document and rule set GE. See Inspection Manual. In this manual you can also find the procedure for checking (Procedure for Checker during Checking: CC). Read this procedure to know what to do during checking.

Checking roles

- User
- Tester
- System
- Quality
- Service
- Source documents
- Rules

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



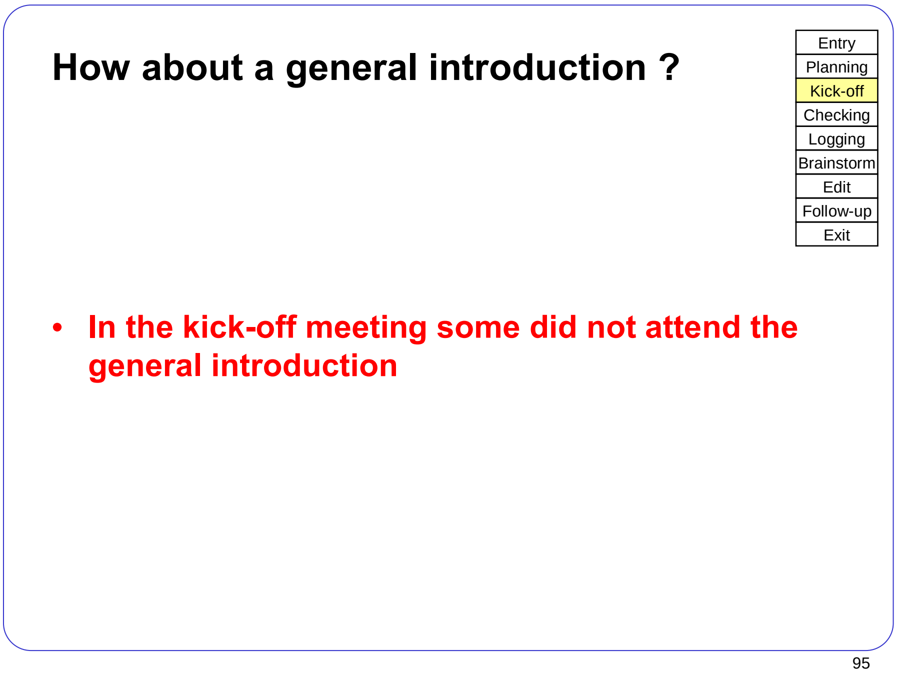
Kickoff meeting

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

- Instruction on Inspection
  - Mini Inspection Tutorial
  - Why, How, What
  - Until *all* participants know it all
- Explain tasks to checkers (masterplan)
- General explanation about documents

- 0 ~ 2 hr

94



How about a general introduction ?

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

- In the kick-off meeting some did not attend the general introduction

95

Individual checking

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

CC1 Try to identify a maximum number of potential issues on behalf of your team, and to help the author

CC2 Your job is to help 'make the author a hero'

CC3 If you get a ridiculously high number of issues, consult with the leader

CC5 Don't be shy of noting any kind of issue you think you have found (you can later decide whether or not to report it)

CC6 You do not have to write a perfectly presented log. It is better to concentrate on finding more issues, but you may write any notes you like, any way you like. They are normally your private notes

CC7 If you have trouble finding issues, consult with the leader or another team member

CC8 If you have any time difficulty, consult with your Inspection leader

CC10 Focus on major (and super-major) issues, do not spend a lot of time and effort finding and noting minor issues

CC11 Classify as you go as S (super), M (major), m (minor), ? (question of intent), P (process improvement)

CC12 Fill in the section called Data Collection at the bottom of your master plan, with your personal checking data, so you can swiftly report your data at the beginning of the Logging Meeting.

See Procedure for Checker during Checking: IN.PR.CC

96

TG

The Logging Meeting

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

- The purpose of the Logging meeting is to record for the Editor:
  - the highest possible number of unique issues in the time available
  - with sufficient clarity that the Editor can understand what the problem is
- *Discussing issues is not the purpose*
- *Fixing issues is not the purpose*
- Discovering additional issues *is* part of the purpose …
  (mark with *N* for New)

97

TG

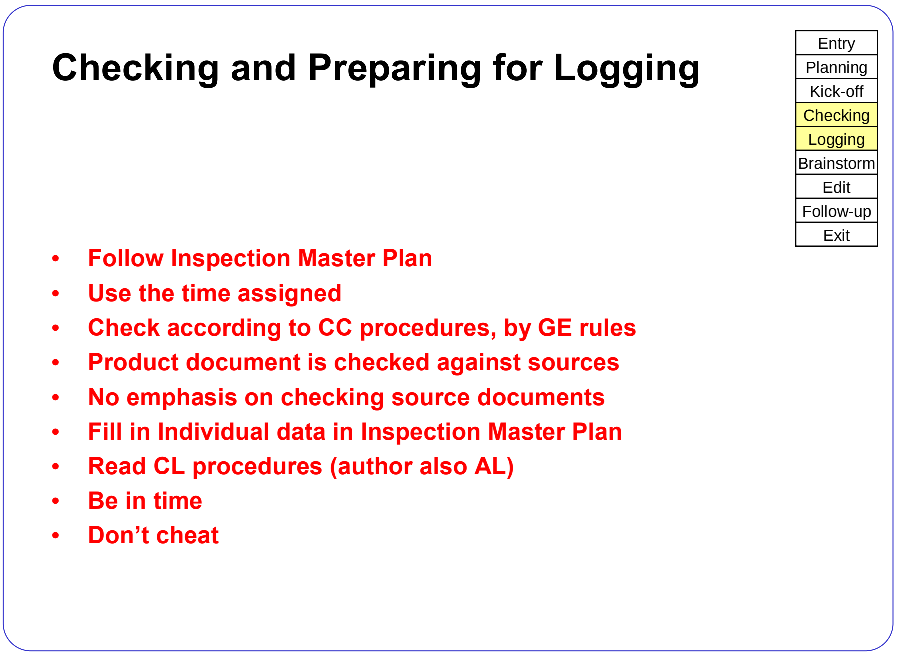
Logging meeting

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

- Logging
  - no discussion allowed
  - no suggestions, no solutions
  - mostly majors
  - any issue is a *violation of a rule*
  - Logging Rate: 0.5 ~ 2 issues per minute logged
- What did you think of the Inspection process
- How should the document have looked like
- 2 hours maximum

**Inspectors are consultants
helping the author to be a hero**

Use these colums during individual checking (print these columns up to row 40 or 50)									for logging meeting		used during edit	
Inspection Issue Log							Item types:					
							S, M, m, Q, P, N					
	InspectionID	1		Date	23 Oct 2007							
Item No	Doc ref	Doc page	Scan/ Check	Location on page	Type of item	Checklist or rule tag	Description If long explanation, remind: "say it in 7 w ords max!"	Number of occurr	time ref	who	Editor note	done
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30												
31												
32												
33												
34												

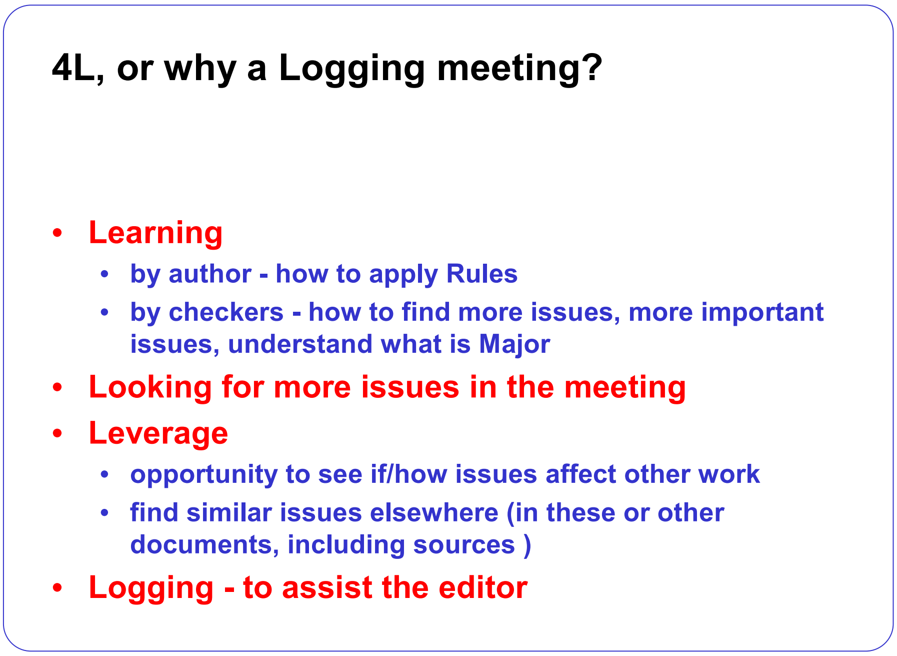


Checking and Preparing for Logging

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

- Follow Inspection Master Plan
- Use the time assigned
- Check according to CC procedures, by GE rules
- Product document is checked against sources
- No emphasis on checking source documents
- Fill in Individual data in Inspection Master Plan
- Read CL procedures (author also AL)
- Be in time
- Don't cheat

100



4L, or why a Logging meeting?

- Learning
  - by author - how to apply Rules
  - by checkers - how to find more issues, more important issues, understand what is Major
- Looking for more issues in the meeting
- Leverage
  - opportunity to see if/how issues affect other work
  - find similar issues elsewhere (in these or other documents, including sources)
- Logging - to assist the editor

101

DG

Process Brainstorming

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

- Purpose:
  - To get ‘grass roots’ insights as to root causes and cures for defects
- Method:
  - Use up to 30 minutes
  - Log opinions about up-to 10 Major defects (3 min. each)
  - Don’t go deeper!
  - Leave deeper analysis to Process Improvement Teams

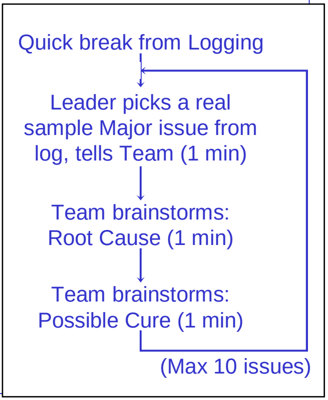
102

TG

The Brainstorming Process

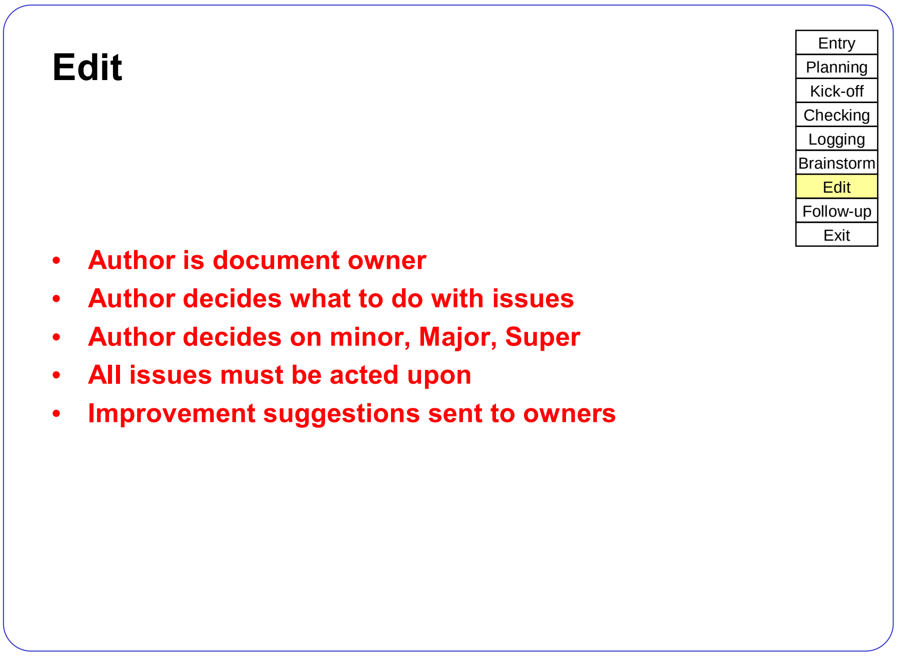
- Entry
- Planning
- Kick-off
- Checking
- Logging
- Brainstorm
- Edit
- Follow-up
- Exit

- Team Stays together after ‘Logging’
  - Same room
  - Same people
  - A break first (select issues)
  - Same documents
  - Up to half an hour
- Shift mentality!
  - Not the project
  - The process, our organization
  - How we feel it can be improved for us
  - So we are not ‘forced’ to make similar mistakes again


Quick break from Logging
Leader picks a real sample Major issue from log, tells Team (1 min)
Team brainstorms: Root Cause (1 min)
Team brainstorms: Possible Cure (1 min)
(Max 10 issues)

103

TG

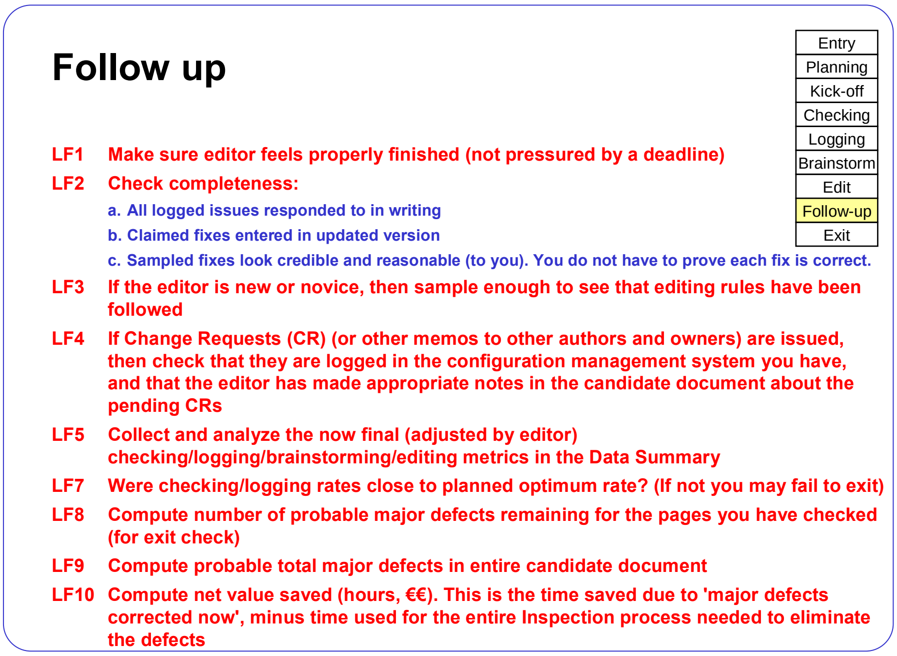


Edit

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

- Author is document owner
- Author decides what to do with issues
- Author decides on minor, Major, Super
- All issues must be acted upon
- Improvement suggestions sent to owners

104



Follow up

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

LF1 Make sure editor feels properly finished (not pressured by a deadline)

LF2 Check completeness:

a. All logged issues responded to in writing

b. Claimed fixes entered in updated version

c. Sampled fixes look credible and reasonable (to you). You do not have to prove each fix is correct.

LF3 If the editor is new or novice, then sample enough to see that editing rules have been followed

LF4 If Change Requests (CR) (or other memos to other authors and owners) are issued, then check that they are logged in the configuration management system you have, and that the editor has made appropriate notes in the candidate document about the pending CRs

LF5 Collect and analyze the now final (adjusted by editor) checking/logging/brainstorming/editing metrics in the Data Summary

LF7 Were checking/logging rates close to planned optimum rate? (If not you may fail to exit)

LF8 Compute number of probable major defects remaining for the pages you have checked (for exit check)

LF9 Compute probable total major defects in entire candidate document

LF10 Compute net value saved (hours, €€). This is the time saved due to 'major defects corrected now', minus time used for the entire Inspection process needed to eliminate the defects

105

Generic Inspection Exit Criteria

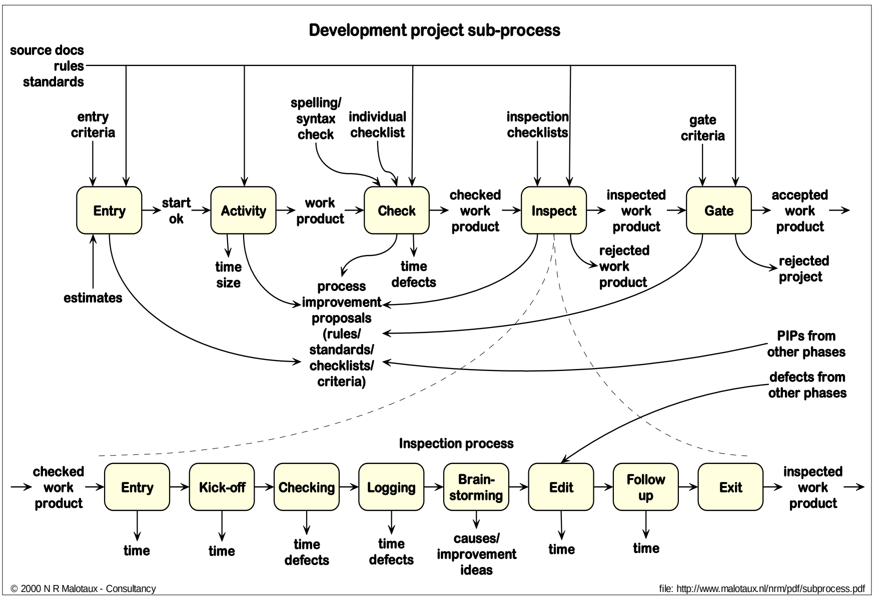
Entry
Planning
Kick-off
Checking
Logging
Brainstorm
Edit
Follow-up
Exit

GXC1	(edit)	All editing completed
GXC2	(CRsent)	All change requests sent to owners of other documents
GXC3	(summary)	Data summary completed and in database
GXC4	(remaining)	Not more than 0.25 (2 for beginners) major defects per page remaining
GXC5	(veto)	Author or Inspector can veto exit
GXC6	(release)	Can we release this document for further use?

Not zero defects, but economically defensible quality,
not worth looking further at this stage

106

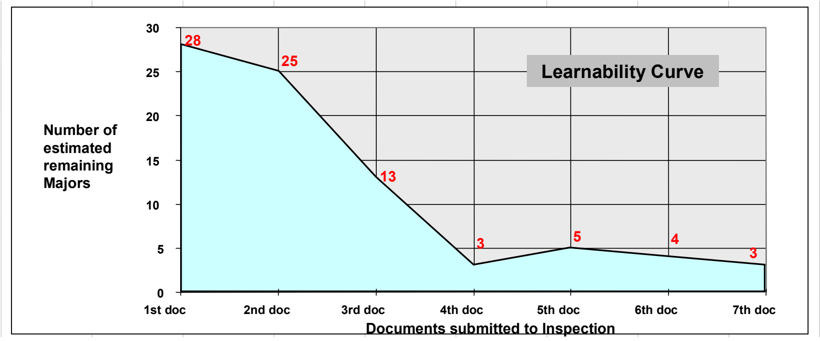
TG



107

Individual learning Curve

- The speed by an individual learning to follow the Rules,
- Measured by reduced Major Defects found in Inspections
  - Faster, earlier and more dramatic than "process improvement"
  - Never mentioned in literature as a measurable



108

TG

Inspection basics

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

HB1 The Inspection leader is trained and certified

HB2 The leader is responsible for managing the process

HB3 First objective is to identify and correct major defects

HB4 Second, but most important, objective is to identify and remove the source of defects

HB5 Fundamental measure of success is the quality-to-cost ratio of the total design life-cycle

HB6 Short term measures include majors found per work-hour (efficiency) and percentage of defects found and treated compared with total defects (effectiveness)

HB7 Productivity measure is the net hours saved due to defects found and removed earlier than they otherwise would be

(see One-page Inspection handbook)

109

TG

Undetected defects

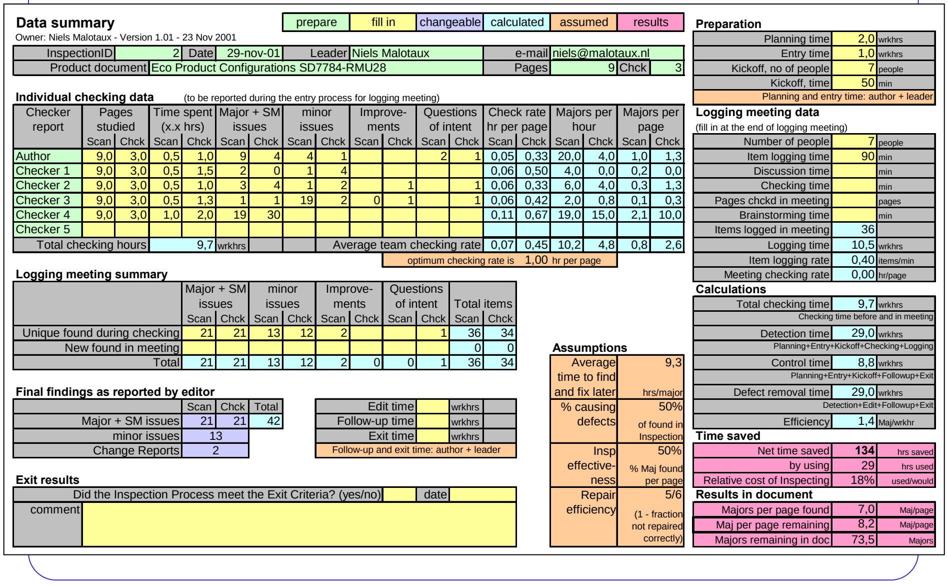
Entry
Planning
Kick-off
Checking
Logging
Brainstorm
Edit
Follow-up
Exit

Defects present but not yet detected by Inspection

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

(Lindner 1992)

Inspection statistics



Data summary

prepare	fill in	changeable	calculated	assumed	results

Owner: Niels Malotaux - Version 1.01 - 23 Nov 2001

InspectionID	2	Date	29-nov-01	Leader	Niels Malotaux	e-mail	niels@malotaux.nl
Product document	Eco Product Configurations SD7784-RMU28					Pages	9 Chck 3

Individual checking data (to be reported during the entry process for logging meeting)

Checker report	Pages studied		Time spent (x.x hrs)		Major + SM issues		minor issues		Improve-ments		Questions of intent		Check rate hr per page		Majors per hour		Majors per page	
	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck
Author	9,0	3,0	0,5	1,0	9	4	4	1			2	1	0,05	0,33	20,0	4,0	1,0	1,3
Checker 1	9,0	3,0	0,5	1,5	2	0	1	4					0,06	0,50	4,0	0,0	0,2	0,0
Checker 2	9,0	3,0	0,5	1,0	3	4	1	2		1		1	0,06	0,33	6,0	4,0	0,3	1,3
Checker 3	9,0	3,0	0,5	1,3	1	1	19	2	0	1		1	0,06	0,42	2,0	0,8	0,1	0,3
Checker 4	9,0	3,0	1,0	2,0	19	30							0,11	0,67	19,0	15,0	2,1	10,0
Checker 5																		
Total checking hours			9,7		wrkhrs			Average team checking rate					0,07	0,45	10,2	4,8	0,8	2,6

optimum checking rate is 1,00 hr per page

Logging meeting summary

	Major + SM issues		minor issues		Improve-ments		Questions of intent		Total items	
	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck	Scan	Chck
Unique found during checking	21	21	13	12	2			1	36	34
New found in meeting									0	0
Total	21	21	13	12	2	0	0	1	36	34

Final findings as reported by editor

	Scan	Chck	Total
Major + SM issues	21	21	42
minor issues	13		
Change Reports	2		

Edit time		wrkhrs
Follow-up time		wrkhrs
Exit time		wrkhrs

Follow-up and exit time: author + leader

Exit results

Did the Inspection Process meet the Exit Criteria? (yes/no)		date	
comment			

Assumptions

Average time to find and fix later	9,3 hrs/major
% causing defects	50% of found in Inspection
Insp effective-ness	50% % Maj found per page
Repair efficiency	5/6 (1 - fraction not repaired correctly)

Preparation

Planning time	2,0	wrkhrs
Entry time	1,0	wrkhrs
Kickoff, no of people	7	people
Kickoff, time	50	min

Planning and entry time: author + leader

Logging meeting data
(fill in at the end of logging meeting)

Number of people	7	people
Item logging time	90	min
Discussion time		min
Checking time		min
Pages chckd in meeting		pages
Brainstorming time		min
Items logged in meeting	36	
Logging time	10,5	wrkhrs
Item logging rate	0,40	items/min
Meeting checking rate	0,00	hr/page

Calculations

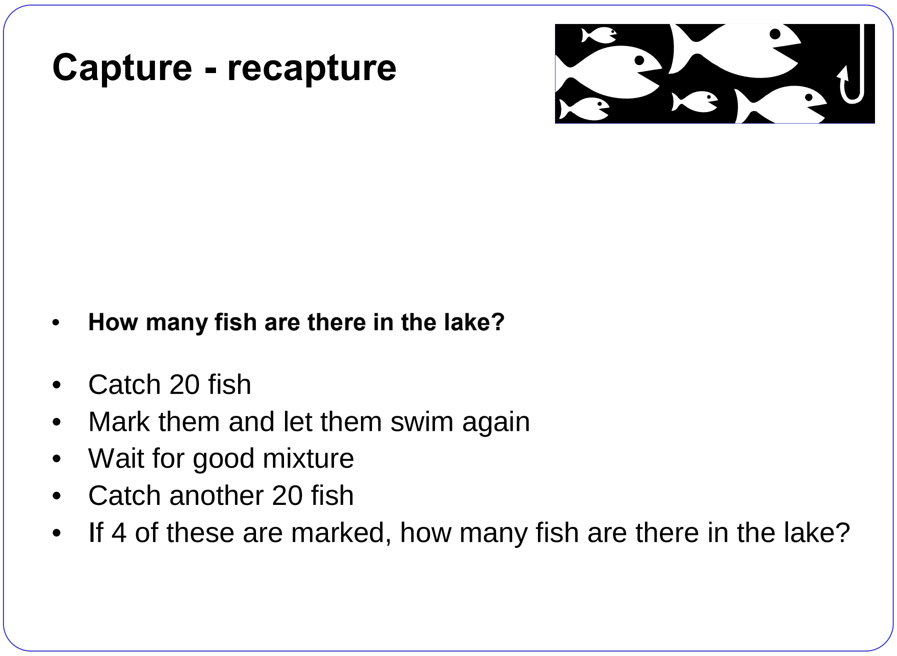
Total checking time	9,7	wrkhrs
Checking time before and in meeting		
Detection time	29,0	wrkhrs
Planning+Entry+Kickoff+Checking+Logging		
Control time	8,8	wrkhrs
Planning+Entry+Kickoff+Followup+Exit		
Defect removal time	29,0	wrkhrs
Detection+Edit+Followup+Exit		
Efficiency	1,4	Maj/wrkhr

Time saved

Net time saved	**134**	hrs saved
by using	29	hrs used
Relative cost of Inspecting	18%	used/would

Results in document

Majors per page found	7,0	Maj/page
Maj per page remaining	8,2	Maj/page
Majors remaining in doc	73,5	Majors

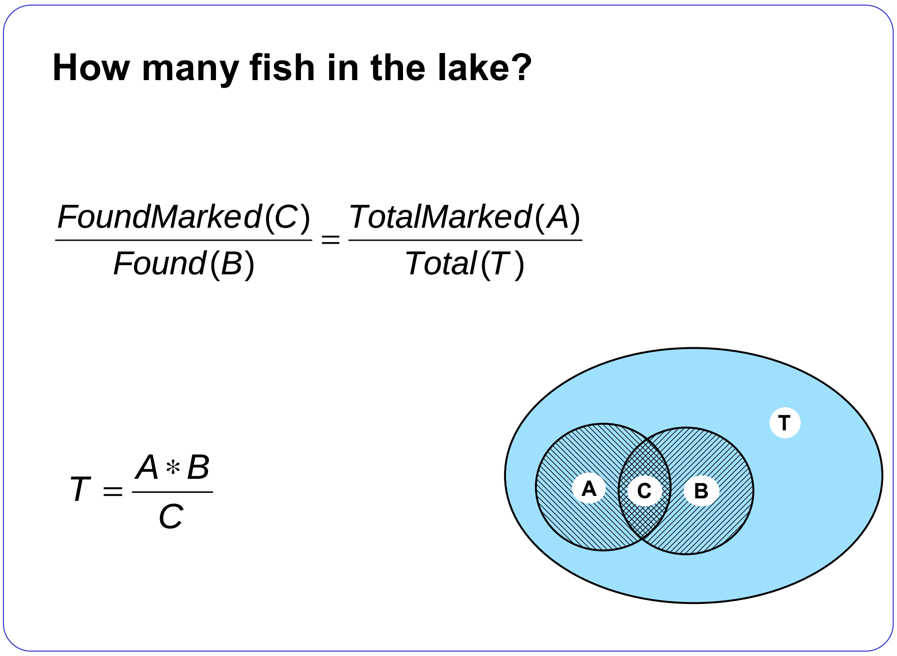


Capture - recapture

- **How many fish are there in the lake?**

- Catch 20 fish
- Mark them and let them swim again
- Wait for good mixture
- Catch another 20 fish
- If 4 of these are marked, how many fish are there in the lake?

112

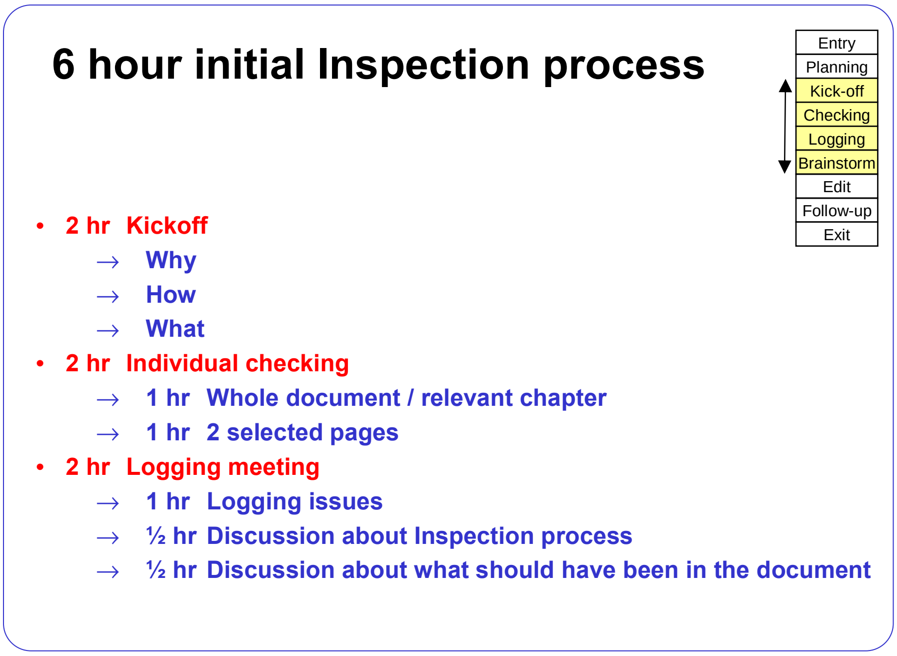


How many fish in the lake?

$$\frac{FoundMarked(C)}{Found(B)} = \frac{TotalMarked(A)}{Total(T)}$$

$$T = \frac{A * B}{C}$$

113

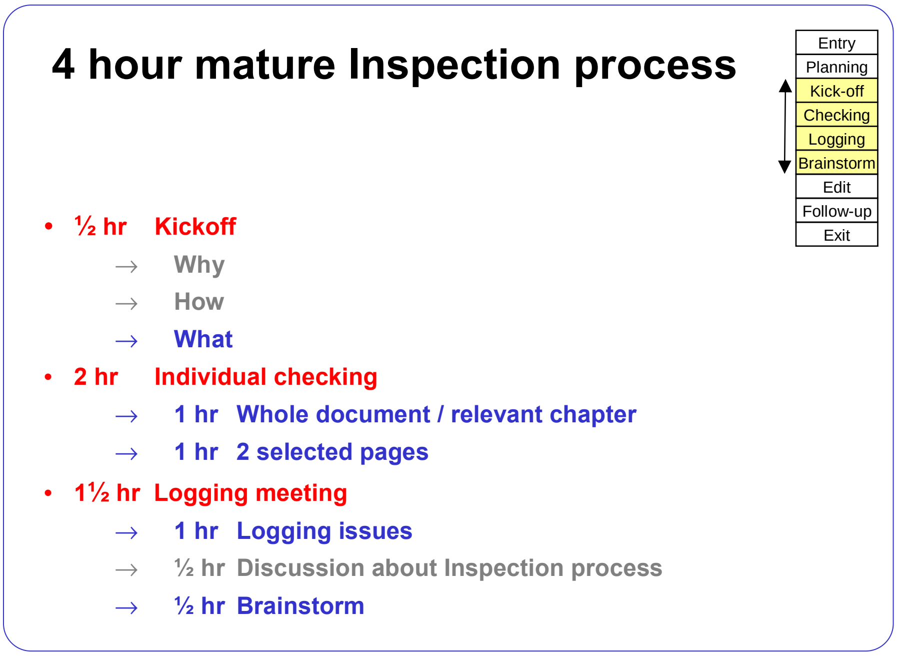


6 hour initial Inspection process

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

- 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

114

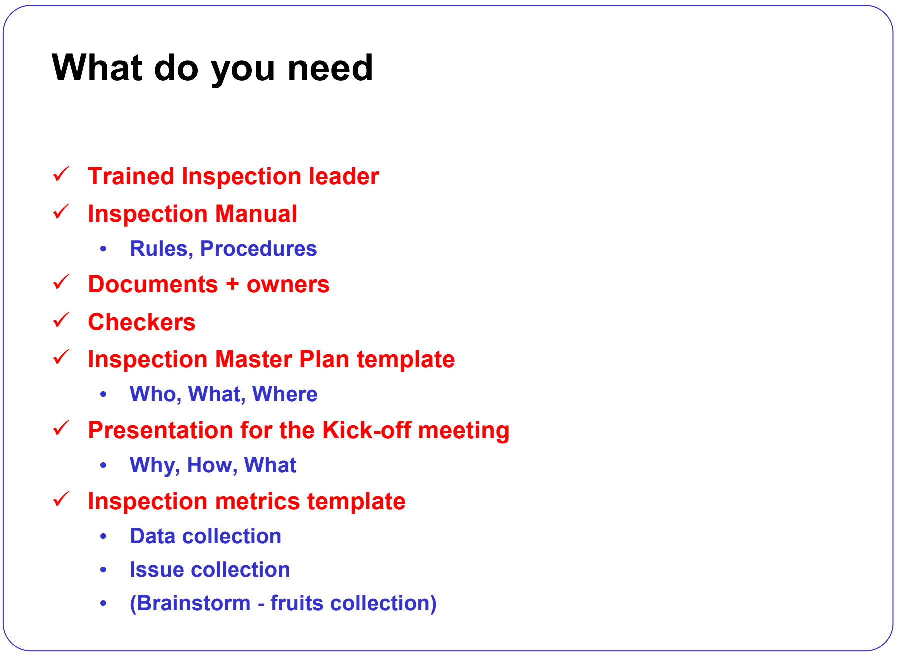


4 hour mature Inspection process

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

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

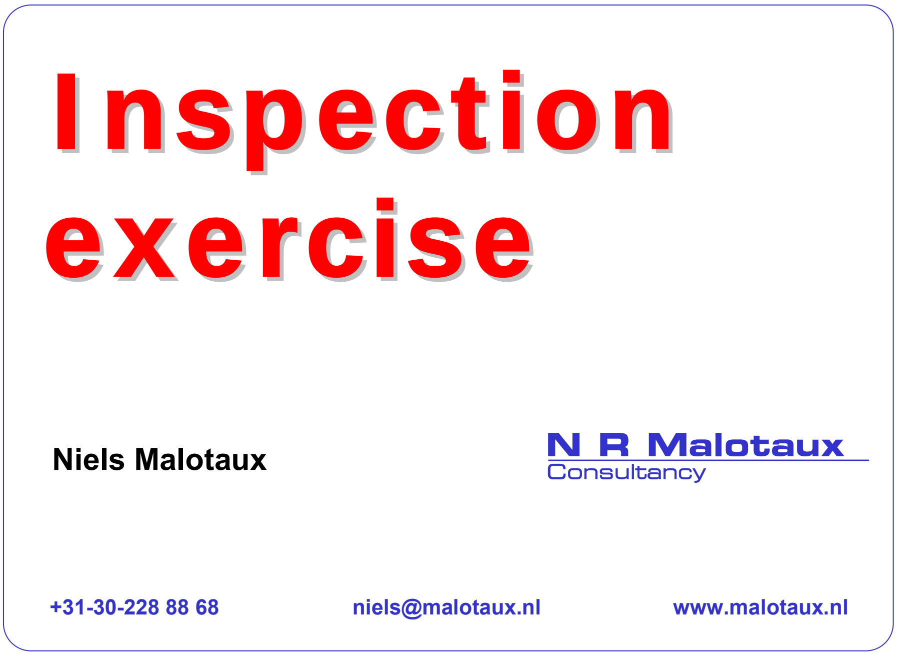
115



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)

116



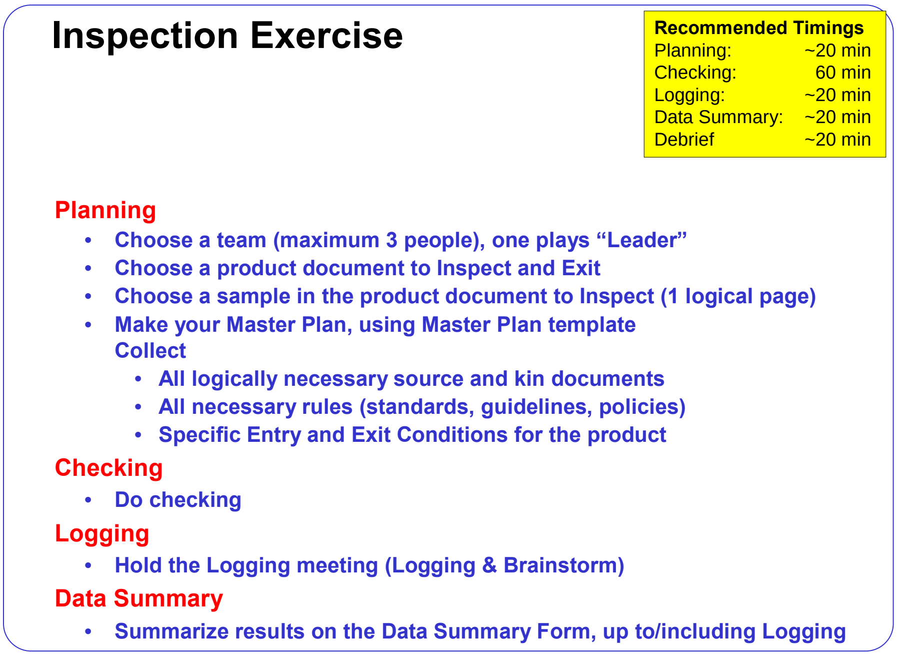
Inspection exercise

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

117



Inspection Exercise

Recommended Timings	
Planning:	~20 min
Checking:	60 min
Logging:	~20 min
Data Summary:	~20 min
Debrief	~20 min

Planning

- Choose a team (maximum 3 people), one plays “Leader”
- Choose a product document to Inspect and Exit
- Choose a sample in the product document to Inspect (1 logical page)
- Make your Master Plan, using Master Plan template
  Collect
  - All logically necessary source and kin documents
  - All necessary rules (standards, guidelines, policies)
  - Specific Entry and Exit Conditions for the product

Checking

- Do checking

Logging

- Hold the Logging meeting (Logging & Brainstorm)

Data Summary

- Summarize results on the Data Summary Form, up to/including Logging

118

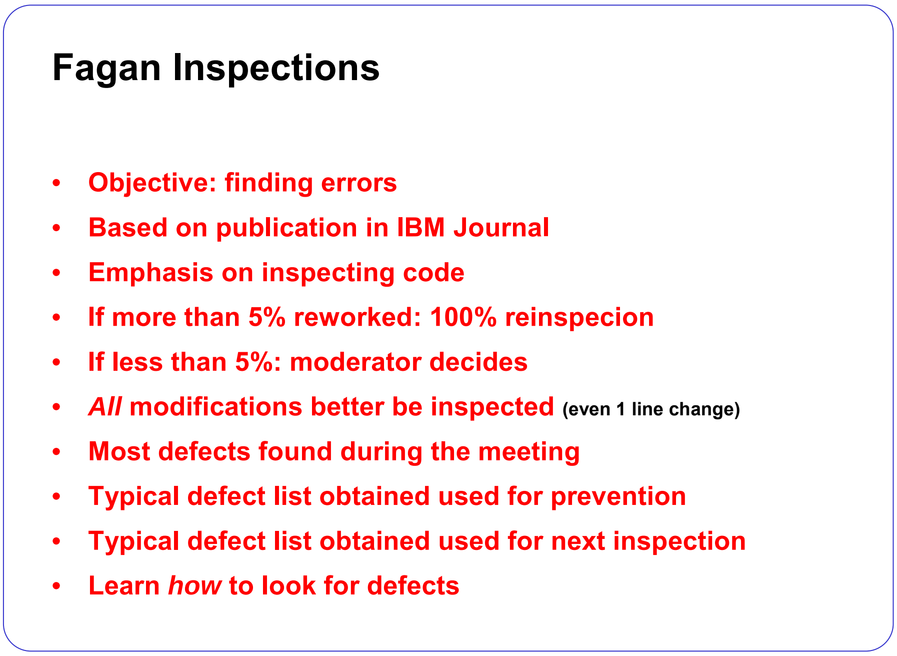
Fagan Inspections

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

119

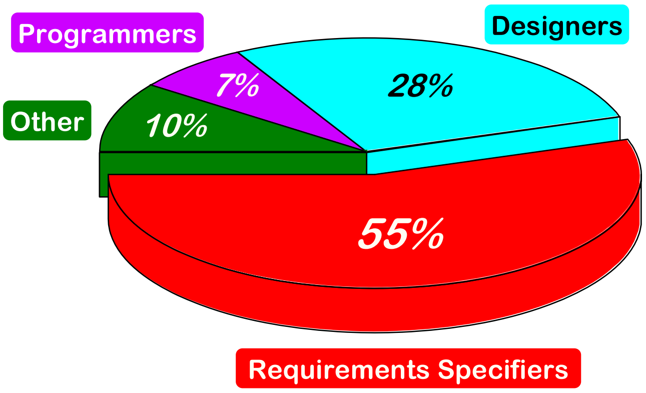


Fagan Inspections

- Objective: finding errors
- Based on publication in IBM Journal
- Emphasis on inspecting code
- If more than 5% reworked: 100% reinspecion
- 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

120

Typical Defect Injectors

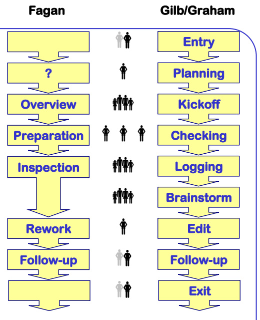


After Bender Associates, *1996*

121

DM

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 - analyze - process

- *What* to look for in Inspection
  Errors classified by type, ranked by frequency,
- *How* to look for presence of errors (education!)
- Analyze results for prevention

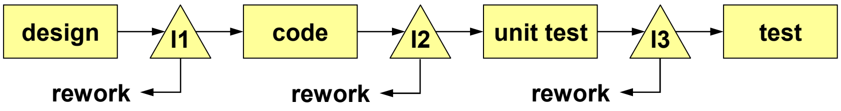
122

Fagan roles

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

123

Fagan experiment



Coding *productivity* change by Inspections:

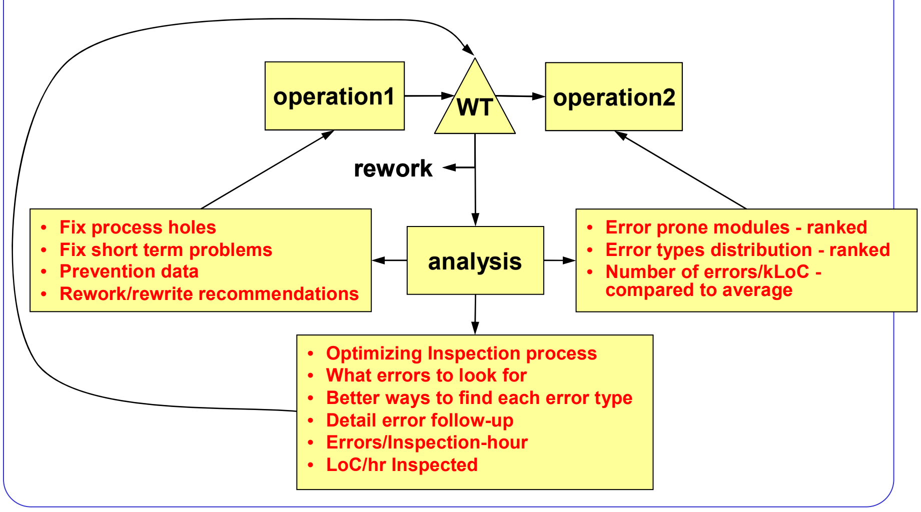
- No Inspection: 100% (baseline)
- I1 only: 112%
- I1 and I2: 123%
- I3 had negative ROI, it was discarded

M.E. Fagan: *Design and Code Inspections to reduce errors in program development*
IBM Systems Journal, Vol15, No3, 1976

124

Walkthroughs and Inspections

(ref Fagan)

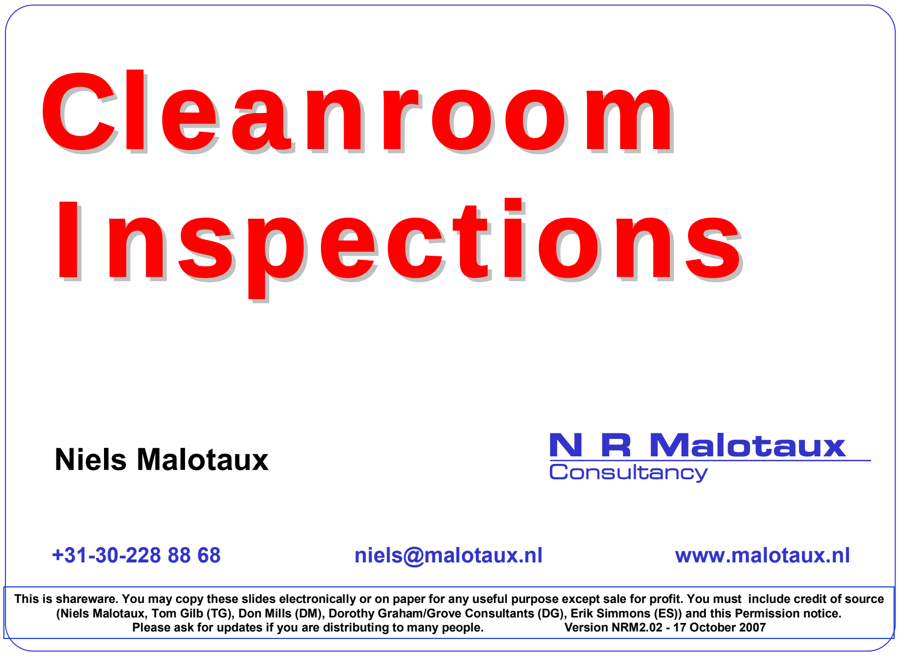


- Fix process holes
- Fix short term problems
- Prevention data
- Rework/rewrite recommendations

- 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

125



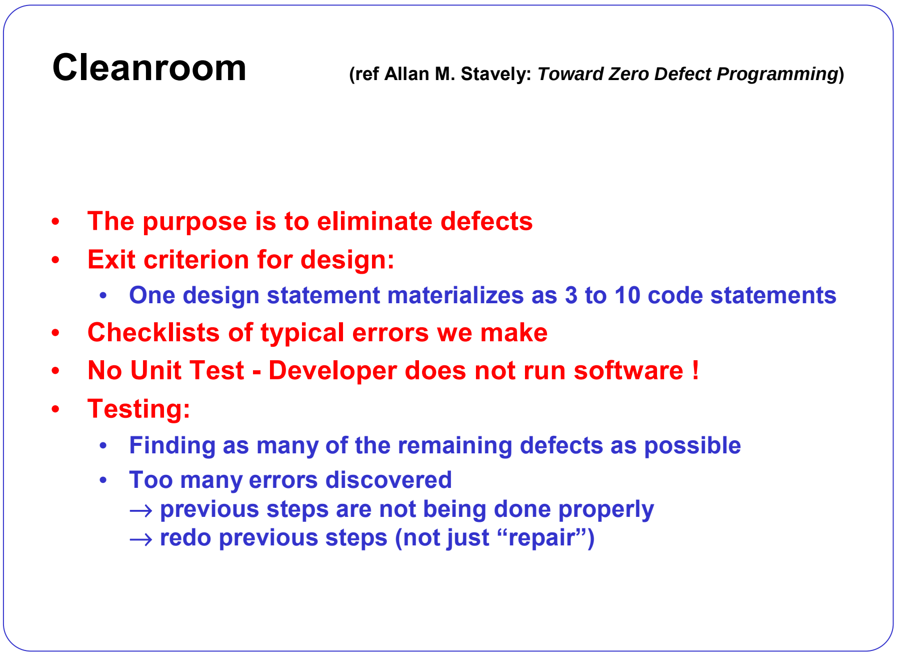
Cleanroom Inspections

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

126



Cleanroom

(ref Allan M. Stavely: *Toward Zero Defect Programming*)

- The purpose is to eliminate defects
- Exit criterion for design:
  - One design statement materializes as 3 to 10 code statements
- Checklists of typical errors we make
- No Unit Test - Developer does not run 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 (not just "repair")

127

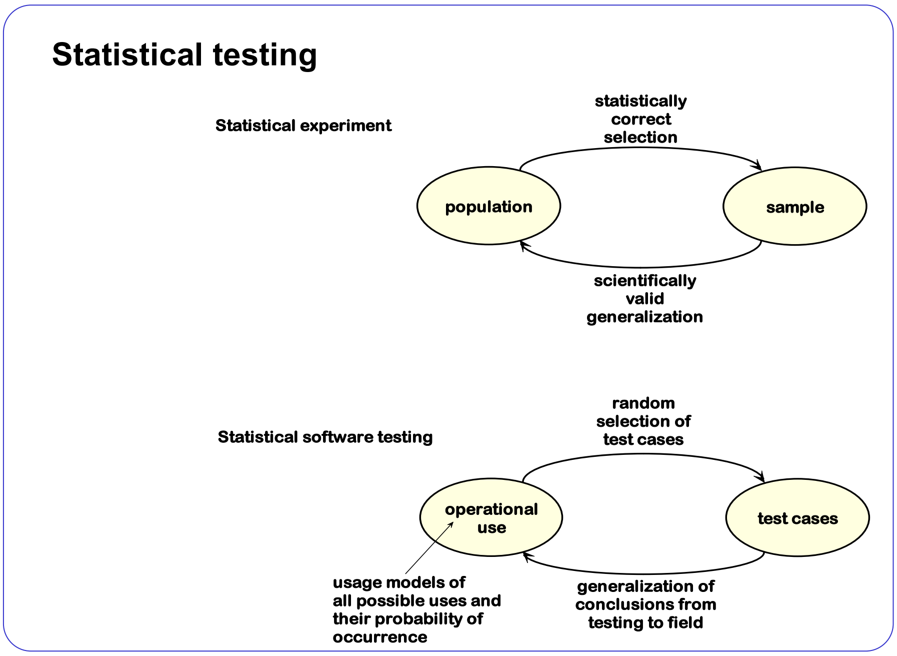
Cleanroom Software Development

- 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

Testing in Cleanroom

- Testing is an important part of the process, but it is done only after verification has been 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 should be less than 10 after verification, compilation and syntax checking
- Very good teams produce 2.3 bugs per kloc and reject code with 4 or 5 bugs 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



Statistical testing

Statistical experiment

statistically correct selection

population

sample

scientifically valid generalization

Statistical software testing

random selection of test cases

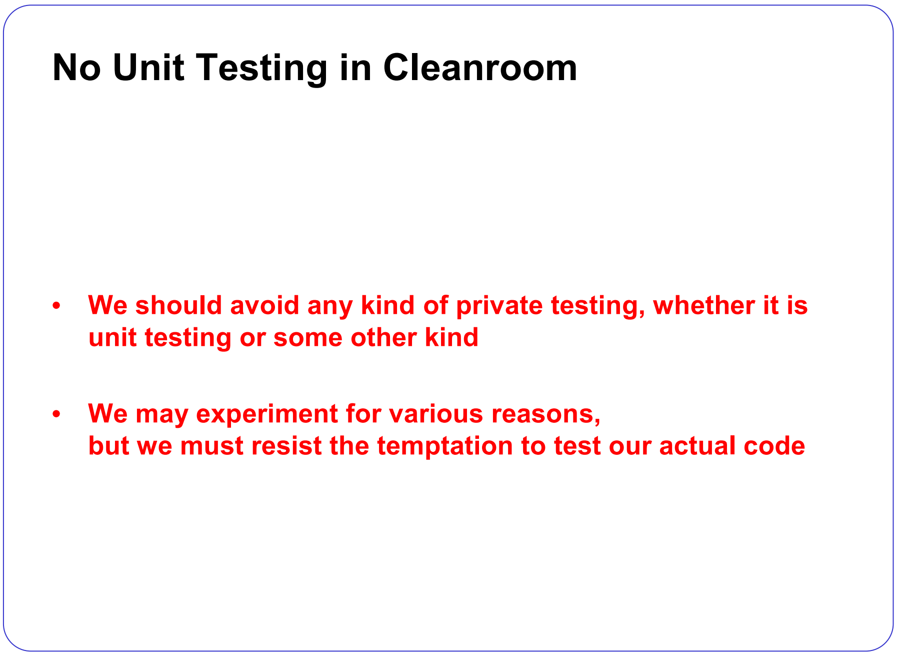
operational use

test cases

usage models of all possible uses and their probability of occurrence

generalization of conclusions from testing to field

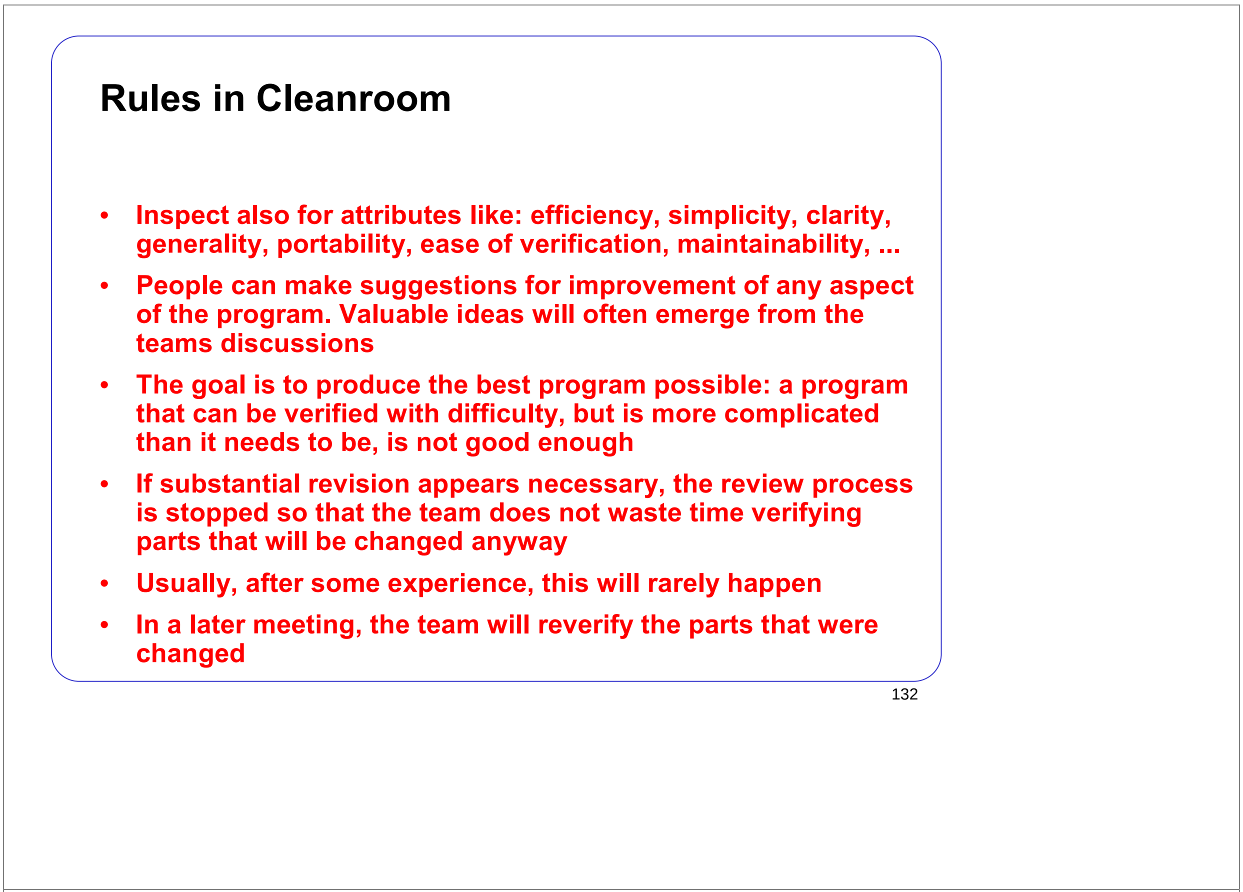
130



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

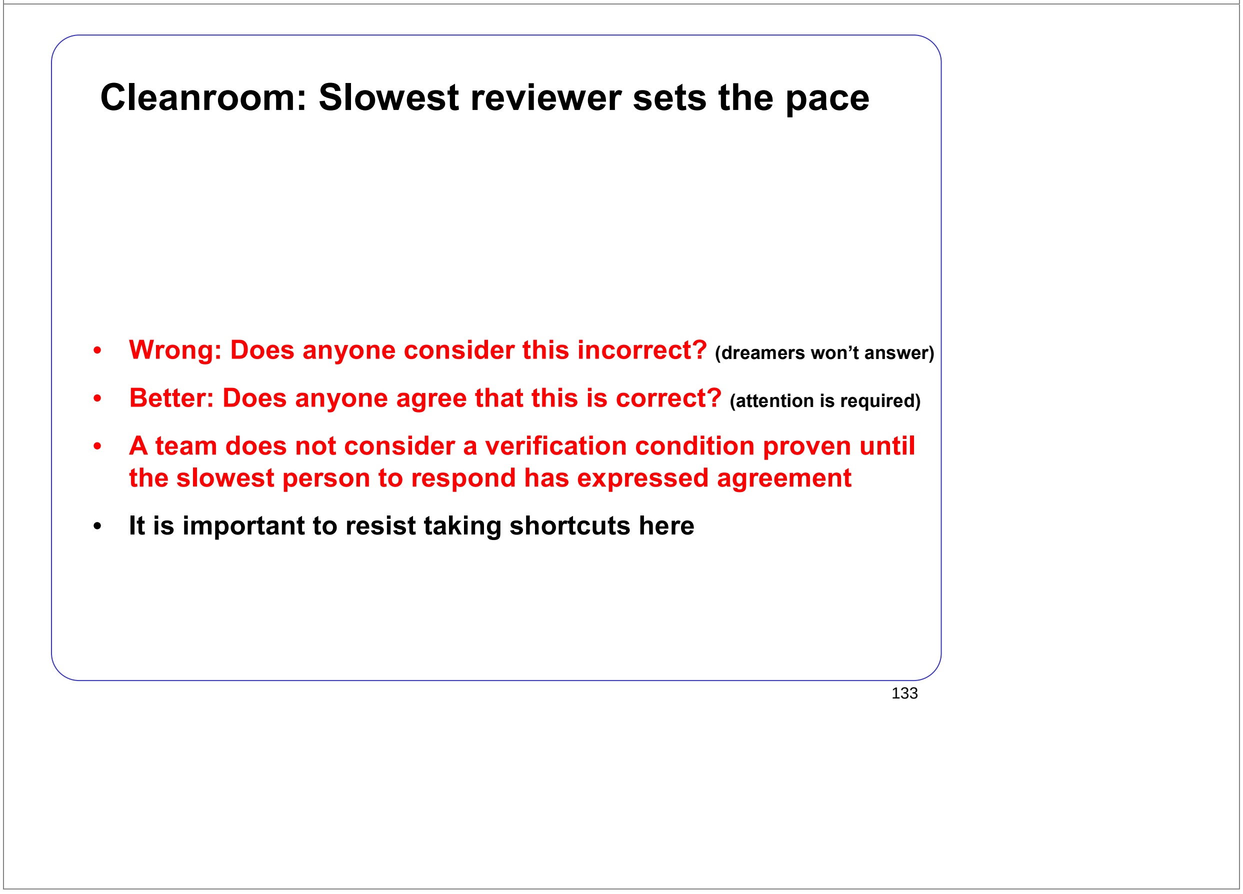
131



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

132



Cleanroom: Slowest reviewer sets the pace

- Wrong: Does anyone consider this incorrect? (dreamers won't answer)
- Better: Does anyone 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

133

SQC process

(Agile Inspection)

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

134

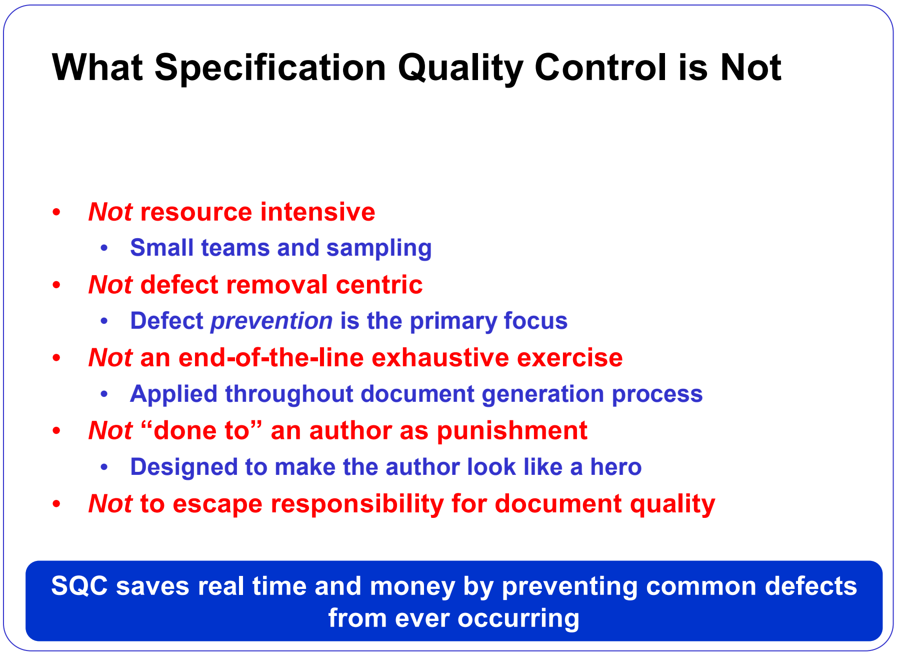
What is Specification Quality Control?

- For ensuring that specifications meet established quality goals according to objective, measured standards
- Specification Quality Control emphasizes:
  - Cost and TTM reduction
  - Defect prevention
  - Resource efficiency
  - Early learning
  - Author confidentiality
  - Quantified specification quality

Specification: Any representation (electronic or otherwise) of a requirement, constraint, design idea, plan, etc.

135

ES



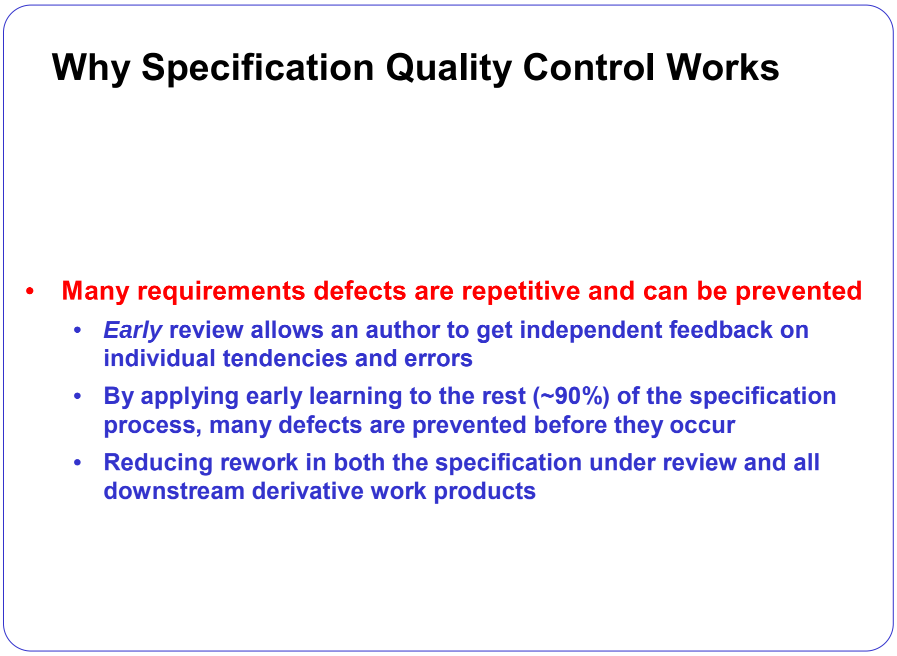
What Specification Quality Control is Not

- *Not* resource intensive
  - Small teams and sampling
- *Not* defect removal centric
  - Defect *prevention* is the primary focus
- *Not* an end-of-the-line exhaustive exercise
  - Applied throughout document generation process
- *Not* “done to” an author as punishment
  - Designed to make the author look like a hero
- *Not* to escape responsibility for document quality

SQC saves real time and money by preventing common defects from ever occurring

136

ES

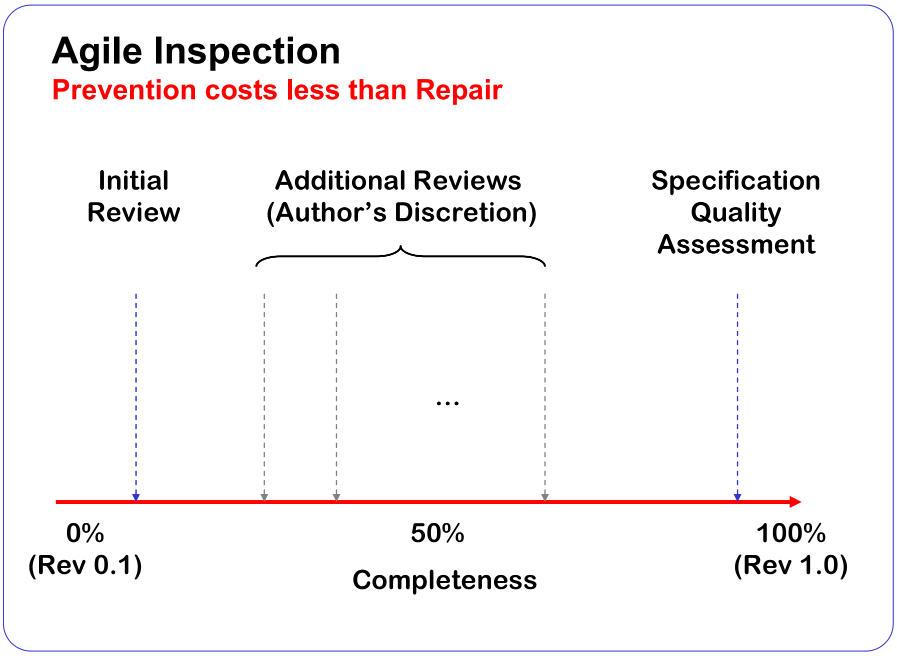


Why Specification Quality Control Works

- Many requirements 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 specification process, many defects are prevented before they occur
  - Reducing rework in both the specification under review and all downstream derivative work products

137

ES



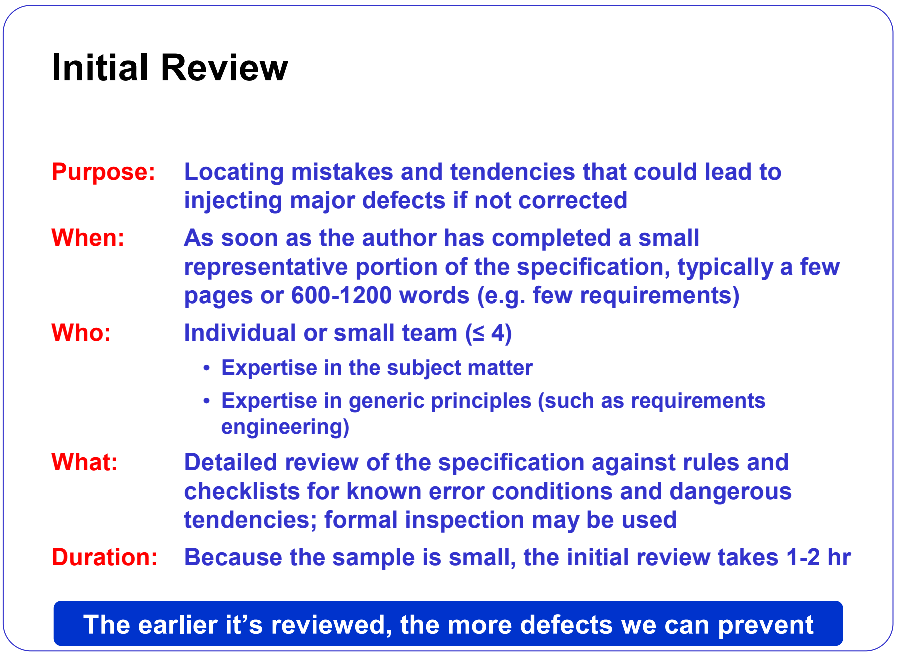
Agile Inspection

Prevention costs less than Repair

Initial Review
Additional Reviews (Author's Discretion)
Specification Quality Assessment
...
0% (Rev 0.1)
50%
Completeness
100% (Rev 1.0)

138

ES



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 (≤ 4)

- Expertise in the subject matter
- Expertise in generic principles (such as requirements engineering)

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 1-2 hr

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

139

ES

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

140

ES

Additional Reviews

Purpose: Providing authors with additional review cycles in order to address significant issues in their work

When: Any time after the initial review, upon author request

Who: Individual or small team (≤ 4)

- Expertise in the subject matter
- Expertise in generic principles

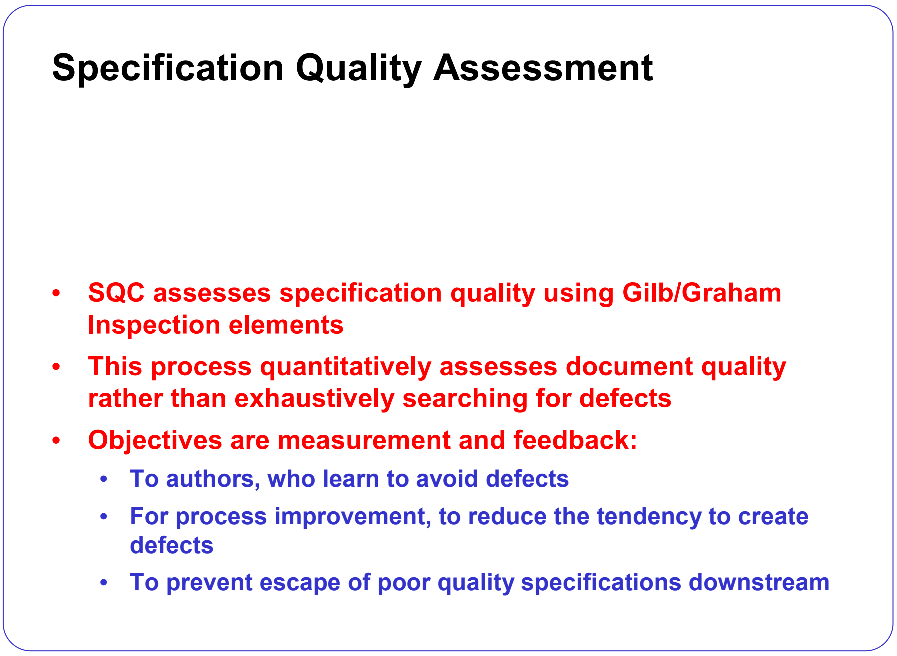
What: Detailed review of samples of the specification against specific defects or tendencies identified in prior reviews

Duration: Because the samples remain small, each review takes 1-2 hours

Follow the guidelines for the Initial Review

141

ES

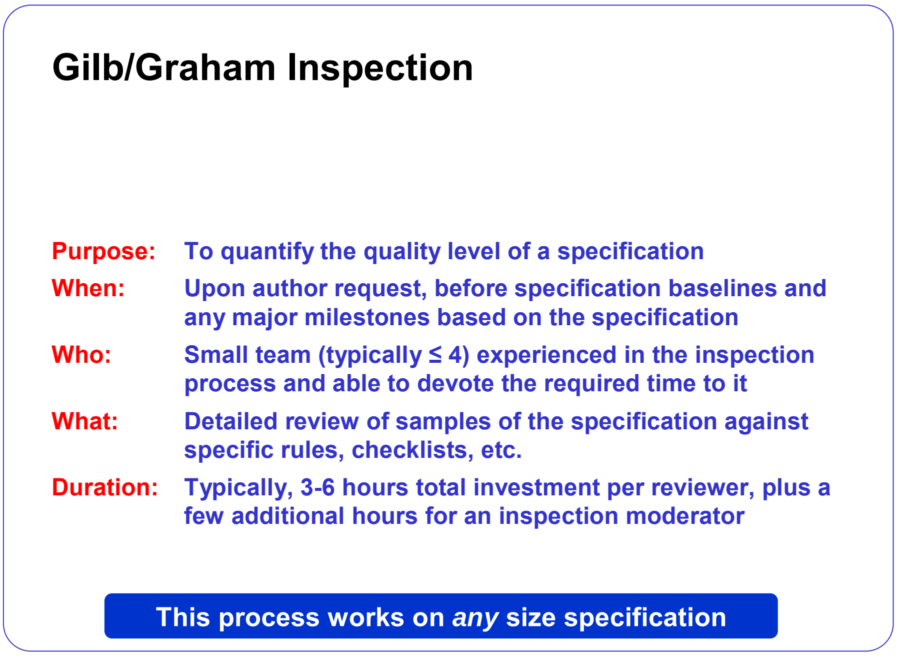


Specification Quality Assessment

- SQC assesses specification quality using Gilb/Graham Inspection elements
- This process quantitatively assesses document quality rather than exhaustively searching for defects
- Objectives are measurement and feedback:
  - To authors, who learn to avoid defects
  - For process improvement, to reduce the tendency to create defects
  - To prevent escape of poor quality specifications downstream

142

ES



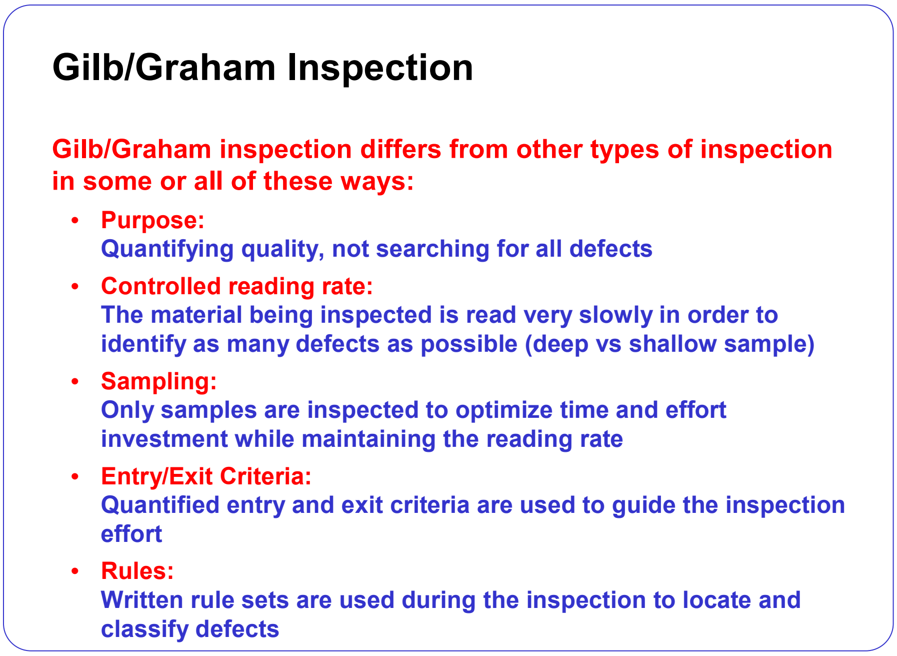
Gilb/Graham Inspection

Purpose:	To quantify the quality level of a specification
When:	Upon author request, before specification baselines and any major milestones based on the specification
Who:	Small team (typically ≤ 4) experienced in the inspection process and able to devote the required time to it
What:	Detailed review of samples of the specification against specific rules, checklists, etc.
Duration:	Typically, 3-6 hours total investment per reviewer, plus a few additional hours for an inspection moderator

This process works on *any* size specification

143

ES



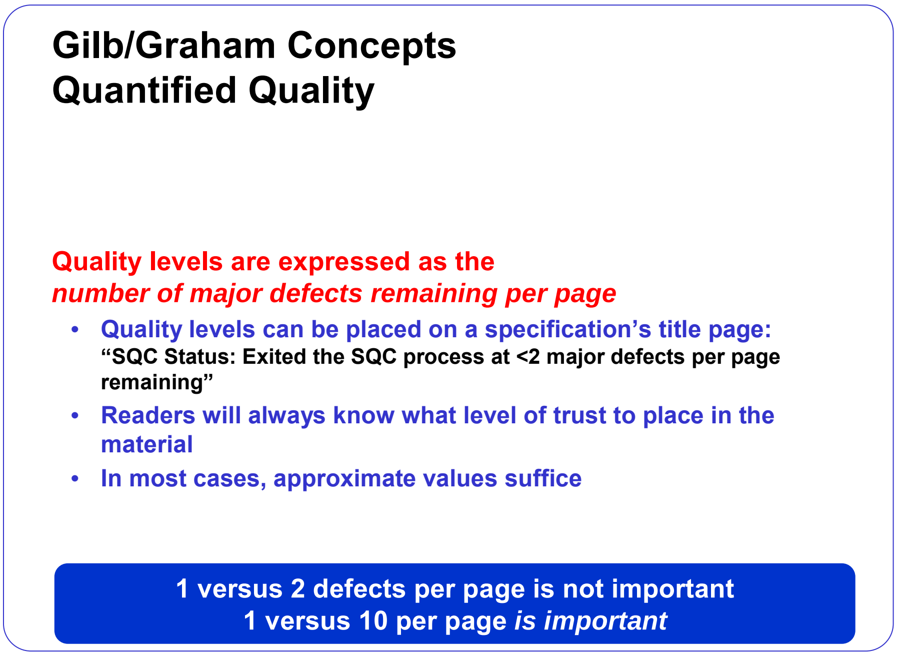
Gilb/Graham Inspection

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 slowly 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 are used to guide the inspection effort
- Rules:
  Written rule sets are used during the inspection to locate and classify defects

144

ES



Gilb/Graham Concepts
Quantified Quality

Quality levels are expressed as the
number of major defects remaining per page

- Quality levels can be placed on a specification's title page:
  "SQC Status: Exited the SQC process at <2 major defects per page remaining"
- Readers will always know what level of trust to place in the material
- In most cases, approximate values suffice

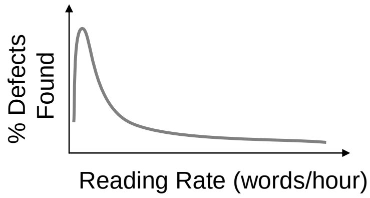
1 versus 2 defects per page is not important
1 versus 10 per page *is important*

145

ES

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!

146

ES

Gilb/Graham Concepts
Calculating Quality Levels

After inspection, the overall defect density remaining in the document will be:

- Any defects we missed in the sample, plus
- Defects we injected while correcting the defects we found, plus
- The defects on the pages we did not check
- All divided by the total (logical) page count

Rule of Thumb: For small samples and average teams, the overall density will be about twice the density of the sample

147

ES

Gilb/Graham Concepts
Entry and Exit Criteria

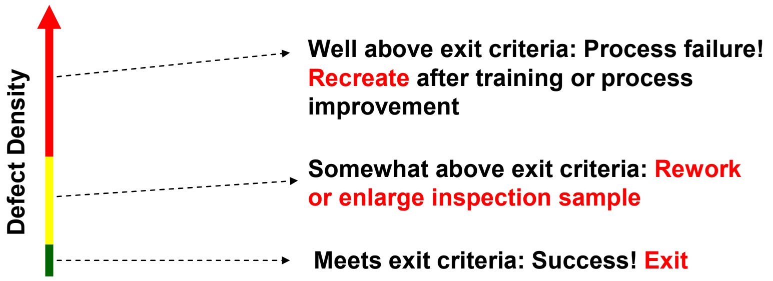
- Entry Criteria prevent valuable inspection resources from being wasted on specifications that are not ready
- Exit Criteria allow quantitative decision making on when the specification is good enough to use
  - Major defects cost far more to fix later than now (~10x)
  - How many defects is economical to leave in the specification?
  - Typical level is < 1 major per page, but you may have to accept somewhat higher to get started
  - Assume 50% detection via inspection (i.e., at least half the defects remain) unless you can prove better

148

ES

Gilb/Graham Concepts
Entry and Exit Criteria

Once the quality level of a specification is known, there are three possible paths forward:



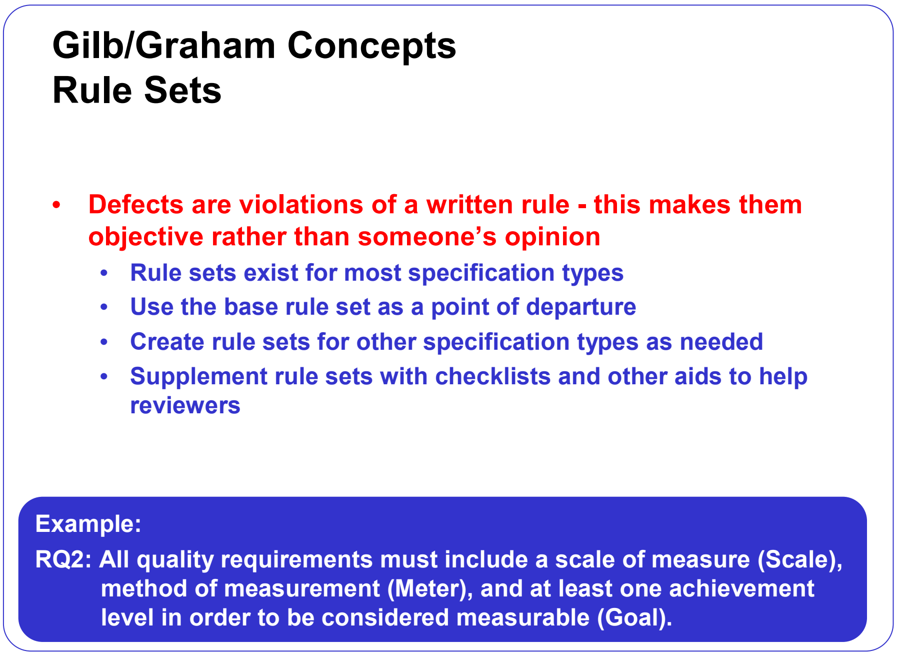
Well above exit criteria: Process failure! Recreate after training or process improvement

Somewhat above exit criteria: Rework or enlarge inspection sample

Meets exit criteria: Success! Exit

149

ES



Gilb/Graham Concepts
Rule Sets

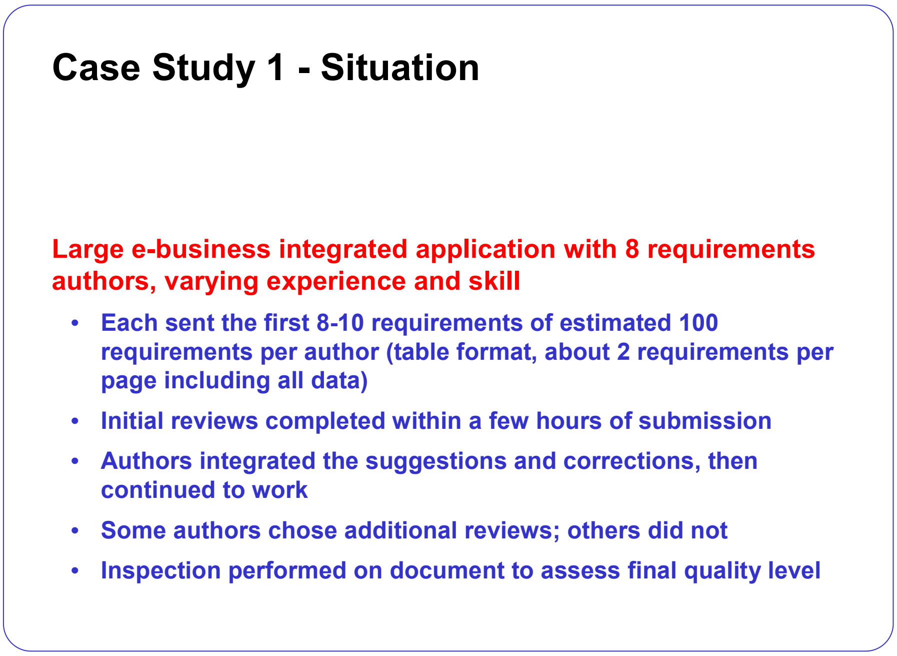
- **Defects are violations of a written rule - this makes them objective rather than someone’s opinion**
  - **Rule sets exist for most specification types**
  - **Use the base rule set as a point of departure**
  - **Create rule sets for other specification types as needed**
  - **Supplement rule sets with checklists and other aids to help reviewers**

Example:

RQ2: All quality requirements must include a scale of measure (Scale), method of measurement (Meter), and at least one achievement level in order to be considered measurable (Goal).

150

ES



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**

151

ES

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 requirements
- 5 x 750 x 10 hr = 37500 hr / 3 = 12500 x $50 = $625000

152

ES

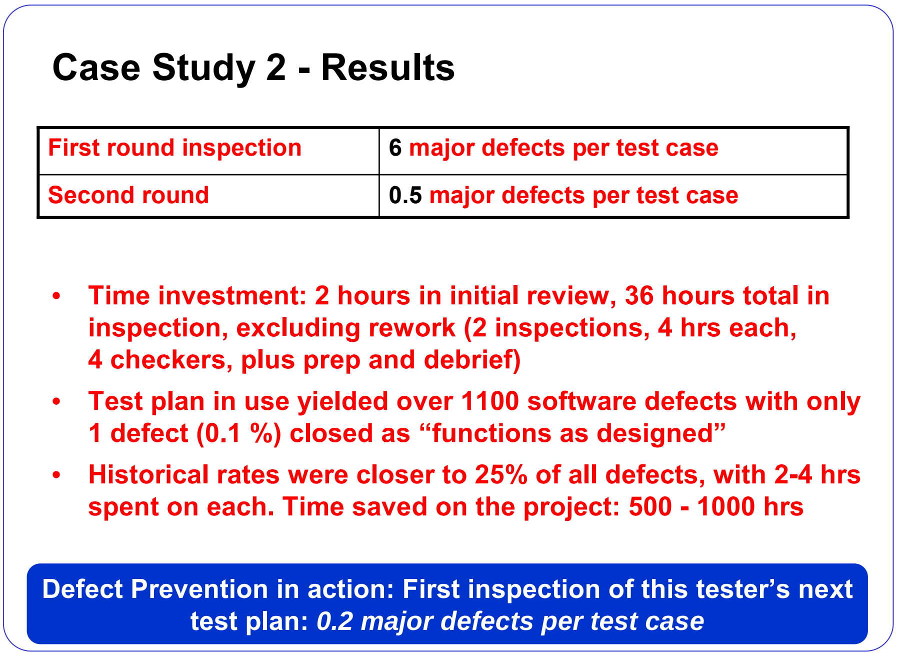
Case Study 2 - Situation

A tester's improvement writing successive test plans:

- SQC used on an existing project to improve test plan quality
- Test plan nearly "complete", so no initial review possible
- 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…

153

ES



Case Study 2 - Results

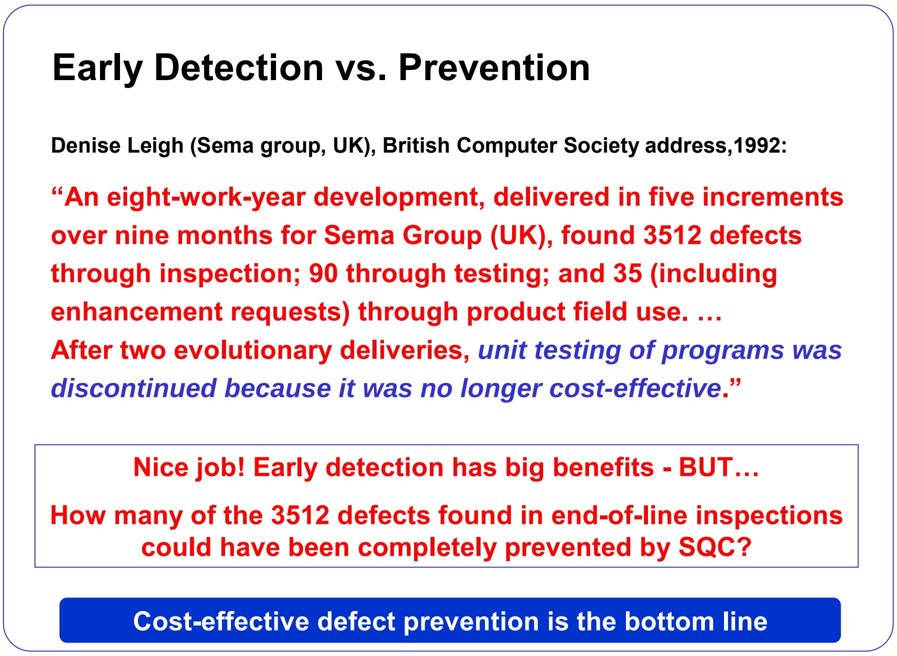
First round inspection	6 major defects per test case
Second round	0.5 major defects per test case

- Time investment: 2 hours in initial review, 36 hours total in inspection, excluding rework (2 inspections, 4 hrs each, 4 checkers, plus prep and debrief)
- Test plan in use yielded over 1100 software defects with only 1 defect (0.1 %) closed as “functions as designed”
- Historical rates were closer to 25% of all defects, with 2-4 hrs spent on each. Time saved on the project: 500 - 1000 hrs

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

154

ES



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*.”

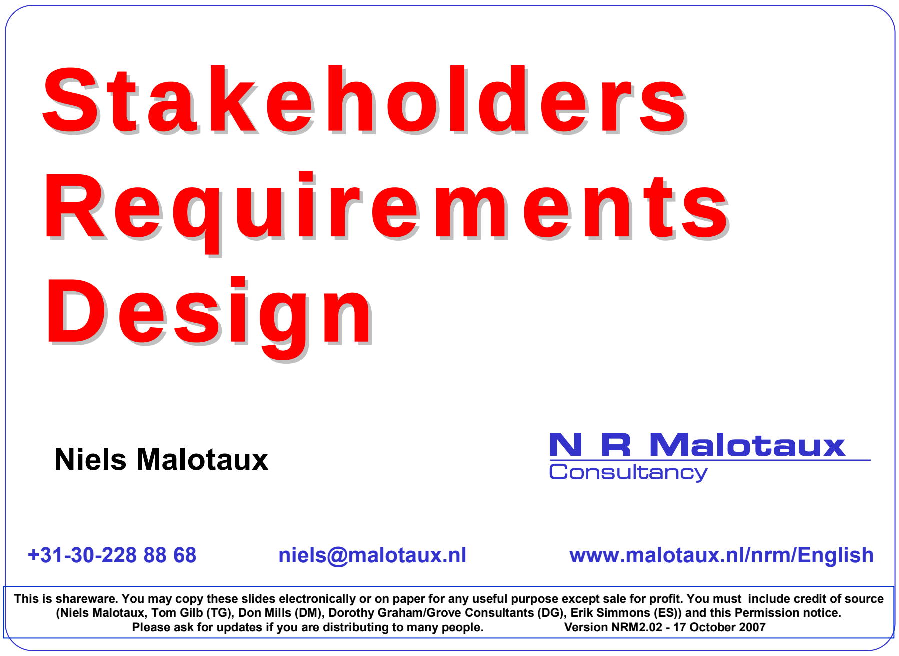
Nice job! Early detection has big benefits - BUT…

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

Cost-effective defect prevention is the bottom line

155

ES

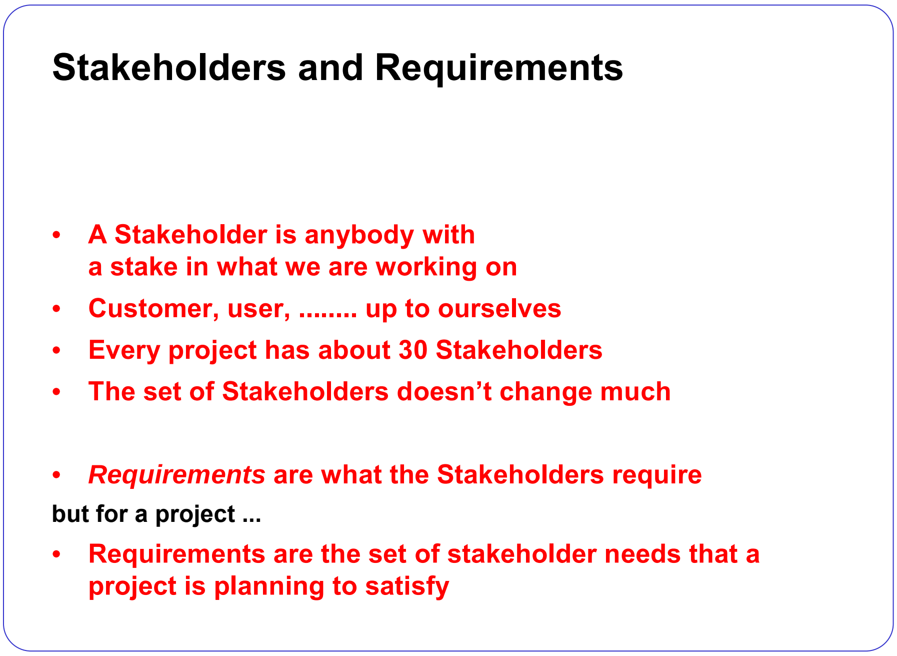


Stakeholders Requirements Design

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl/nrm/English



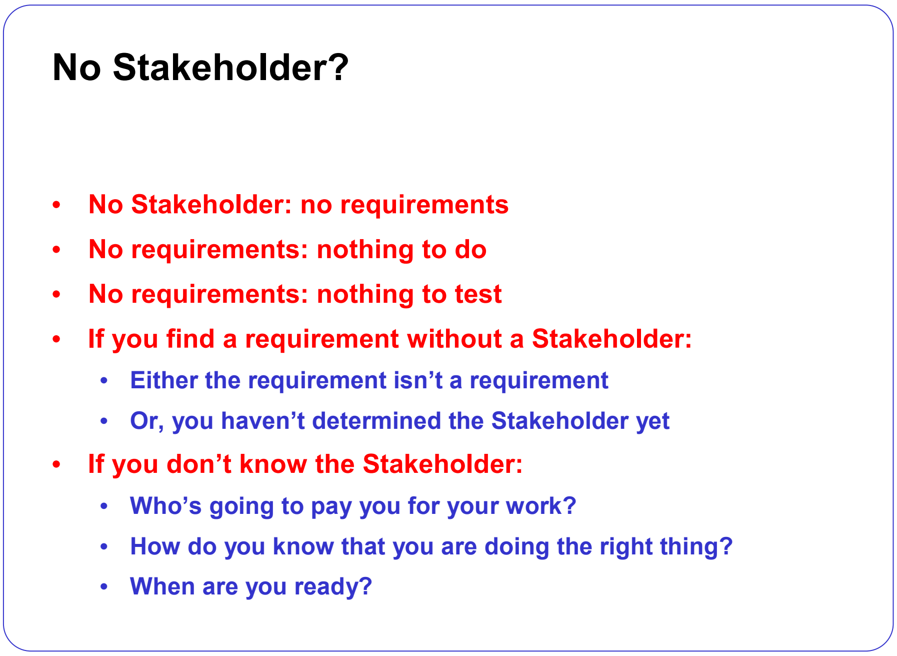
Stakeholders and Requirements

- A Stakeholder is anybody with a stake in what we are working on
- Customer, user, ... up to ourselves
- Every project has about 30 Stakeholders
- The set of Stakeholders doesn't change much

- *Requirements* are what the Stakeholders require

but for a project ...

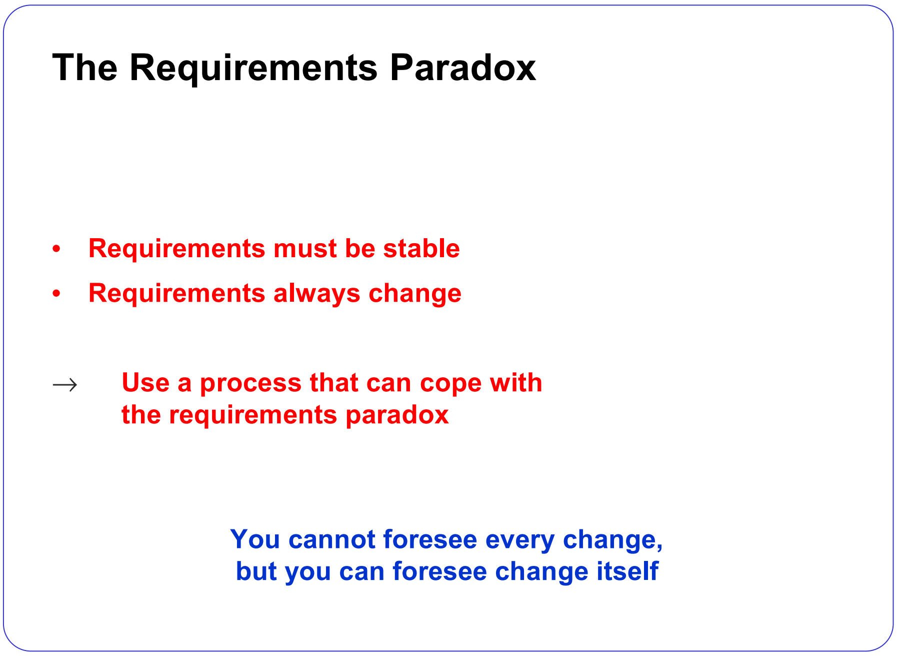
- Requirements are the set of stakeholder needs that a project is planning to satisfy



No Stakeholder?

- No Stakeholder: no requirements
- No requirements: nothing to do
- No requirements: nothing to test
- If you find a requirement without a Stakeholder:
  - Either the requirement isn't a requirement
  - Or, you haven't determined the Stakeholder yet
- If you don't know the Stakeholder:
  - Who's going to pay you for your work?
  - How do you know that you are doing the right thing?
  - When are you ready?

158



The Requirements Paradox

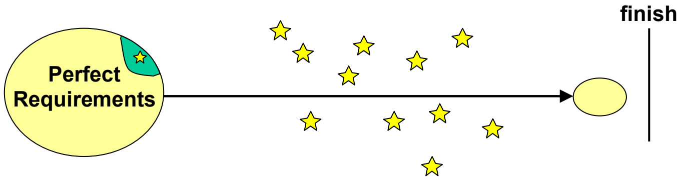
- Requirements must be stable
- Requirements always change

→ Use a process that can cope with
the requirements paradox

You cannot foresee every change,
but you can foresee change itself

159

The 2nd requirements paradox



- **We don’t want requirements to change, however,**
- **Because requirements change now is a *known risk:*
  We must *provoke* requirements change *as early as possible***

160

Basic Types of Requirements

- **Functional** ***binary***
  - **Determine the scope of the project:**
  - **What are we working on**
- **Quality/performance** ***scalar***
  - **To enhance the performance of the selected functions**
- **Constraints** ***binary / scalar***
  - **What should we not do, be aware of, be limited by**

161

Requirements with Planguage

ref Tom Gilb

Definition:

RQ27: Maximum Response Time

Scale: Seconds between <asking> for information and <appearance> of it.

Meter: Add a function to the software to measure the maximum response time value and the <range of values> per <working day>.

Benchmarks (Playing Field):

Past: 3 sec (our previous product)

Current: 0.6 sec [competitor y, product x, 2006] ← Marketing Survey Jan 2006

Record: 0.2 sec [competitor x, product y]

Wish: 0.2 sec [2008] ← customer's head of R&D, 19 Feb 2005, <document ...>

Note: Less than 0.2 sec is not noticed by the user,
so there is no use in trying to be better than 0.2 sec

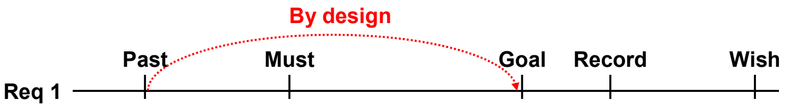
Requirements:

Must: 1 sec [99%] ← project-contract

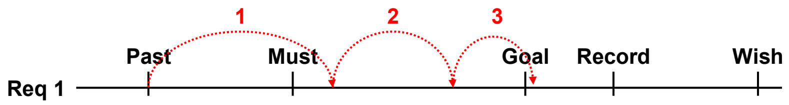
Must: 1.5 sec [100%] ← project-contract

Goal: 0.5 sec ← project-contract

Design to a Quality Requirement



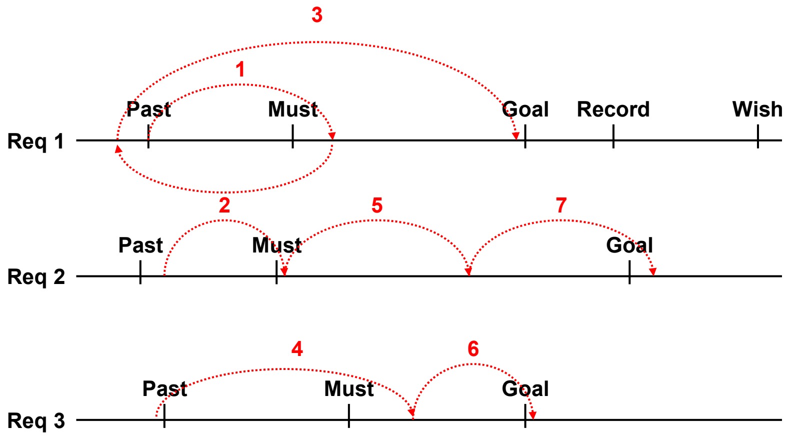
Design to a Quality Requirement one step at the time



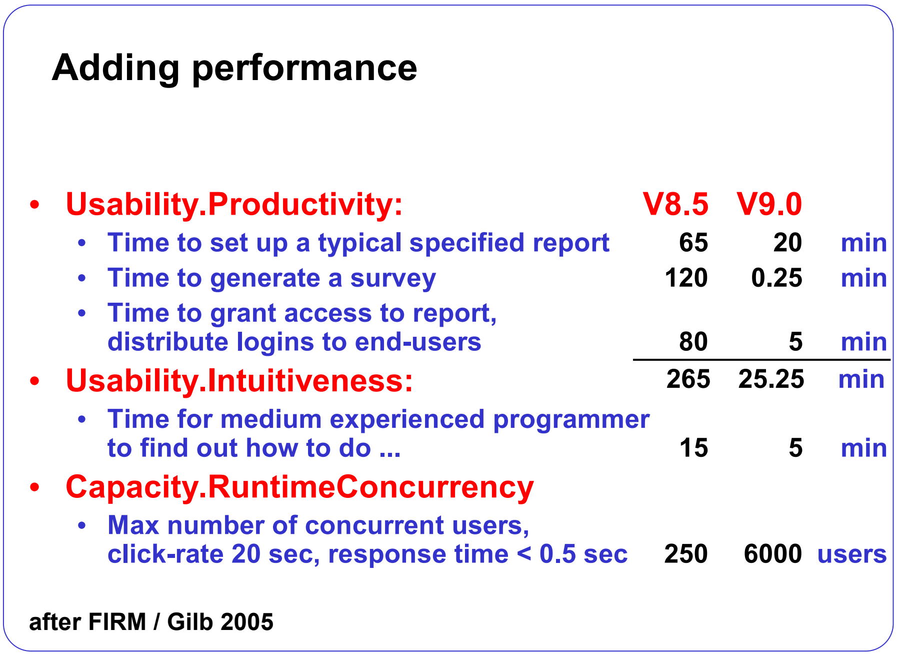
If the Quality requirement is composed of several elements, start with the most contributing factor for the least cost

164

Design to Multidimensional Quality Requirements



165

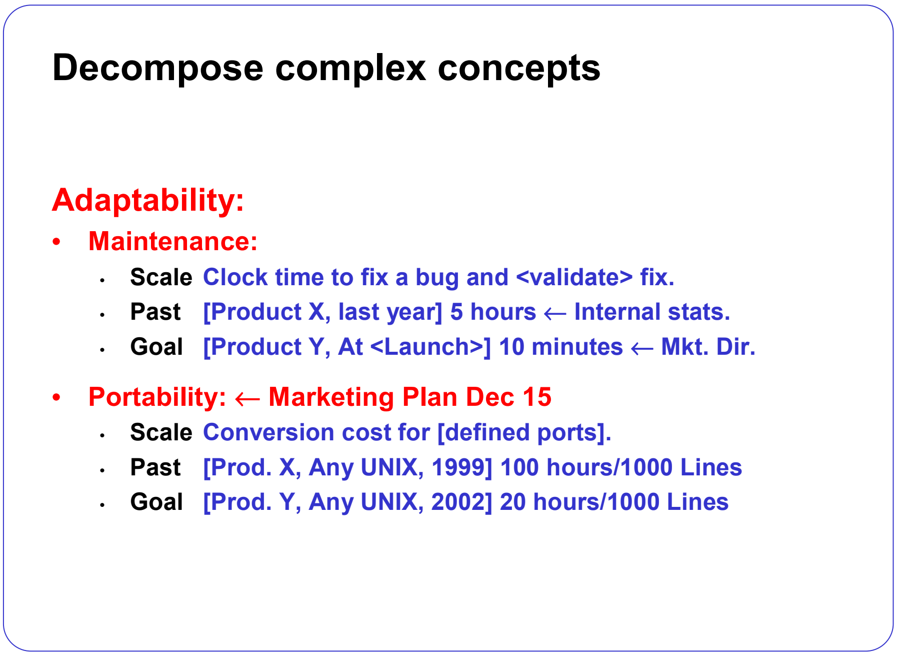


Adding performance

	V8.5	V9.0	
Usability.Productivity:			
- Time to set up a typical specified report	65	20	min
- Time to generate a survey	120	0.25	min
- Time to grant access to report, distribute logins to end-users	80	5	min
Usability.Intuitiveness:	265	25.25	min
- Time for medium experienced programmer to find out how to do ...	15	5	min
Capacity.RuntimeConcurrency			
- Max number of concurrent users, click-rate 20 sec, response time < 0.5 sec	250	6000	users

after FIRM / Gilb 2005

166



Decompose complex concepts

Adaptability:

- **Maintenance:**
  - Scale Clock time to fix a bug and <validate> fix.
  - Past [Product X, last year] 5 hours ← Internal stats.
  - Goal [Product Y, At <Launch>] 10 minutes ← Mkt. Dir.
- **Portability: ← Marketing Plan Dec 15**
  - Scale Conversion cost for [defined ports].
  - Past [Prod. X, Any UNIX, 1999] 100 hours/1000 Lines
  - Goal [Prod. Y, Any UNIX, 2002] 20 hours/1000 Lines

167

TG

Rules for Performance Requirements

1. They should be unambiguously clear to the intended reader
2. There should be a SCALE of measure to define the Quality/Cost concept
3. Complex concepts should be decomposed into a set of *measurable* elementary concepts
4. To define 'relative' terms like 'higher' there should be at least two points of reference on the defined SCALE
5. They shall specify exactly when a quality level is to be available
6. They shall not mix design ideas (how to do it) in the specification of objectives/requirements (what to do)
7. The process input or “source” (like contract, standard, marketing plan) of the requirement shall be given
8. Fuzzy unclear concepts shall be marked with <angle brackets> for improvement later (= before use!)

168

TG

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)

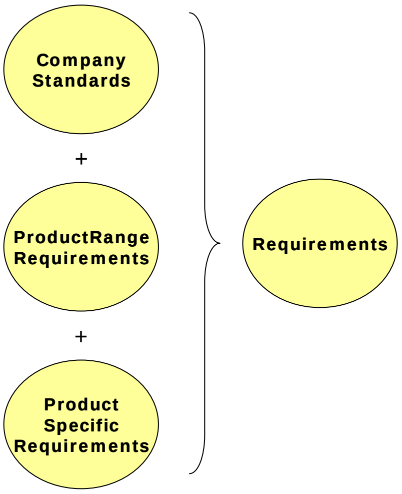
GE11 (risk) The author should clearly indicate any information which is uncertain or poses any risk to the project, using indications like:
{<vaguely defined>, ?, ??, 70% ± 20, suitable comments or notes}

GE12 (verifiable) All statements should be verifiable

GE13 (true) The statement is simply not true

169

Requirements should be at one place only



170

Requirements

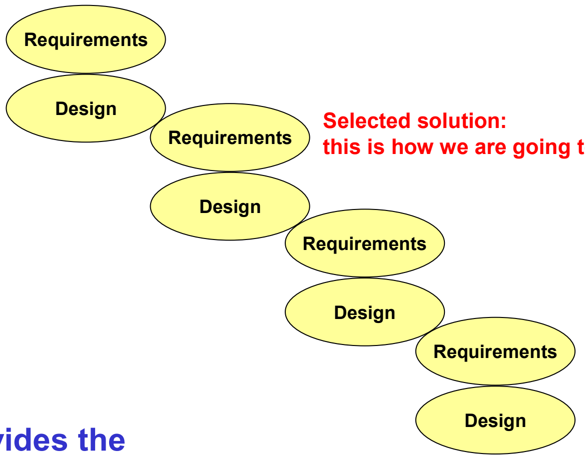
First develop the problem, only then the solution

- What Stakeholders need
- What the project is planning to satisfy
- No design (*how* it is to be done)

- Better spend 10 ~ 15% of the project time on Requirements in order to *save* time

171

No Design in the requirements, but ...



Needs:
what do we need

Options:
how can we do it

Selected solution:
this is how we are going to do it

Design provides the
Requirements for the next level

172

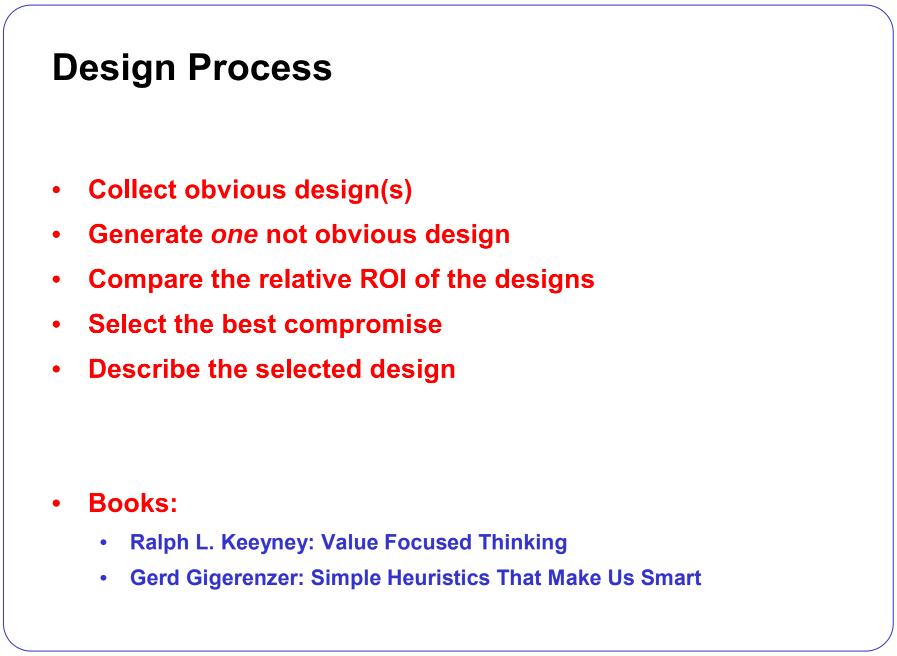
Design is always a compromise

- The Requirements are *always* conflicting

example:

- Performance ↑
- Budget (time, money) ↓

173

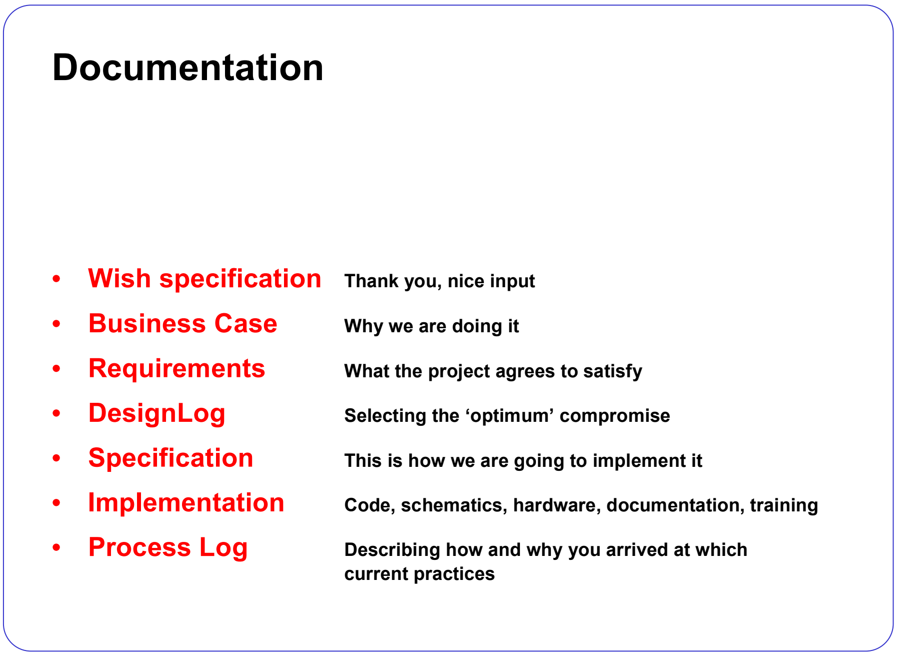


Design Process

- Collect obvious design(s)
- Generate *one* not obvious design
- Compare the relative ROI of the designs
- Select the best compromise
- Describe the selected design

- Books:
  - Ralph L. Keeyney: Value Focused Thinking
  - Gerd Gigerenzer: Simple Heuristics That Make Us Smart

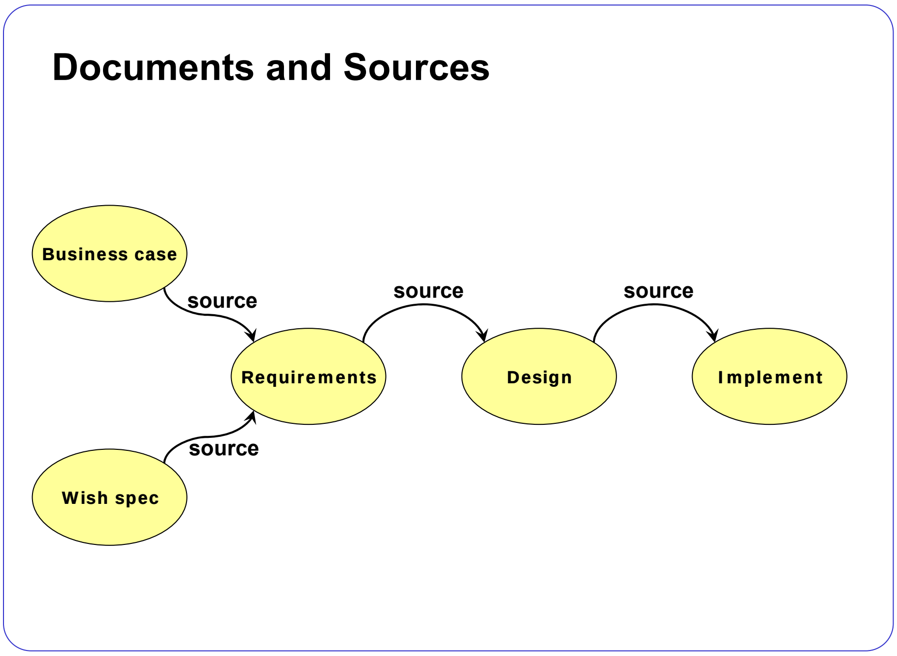
174



Documentation

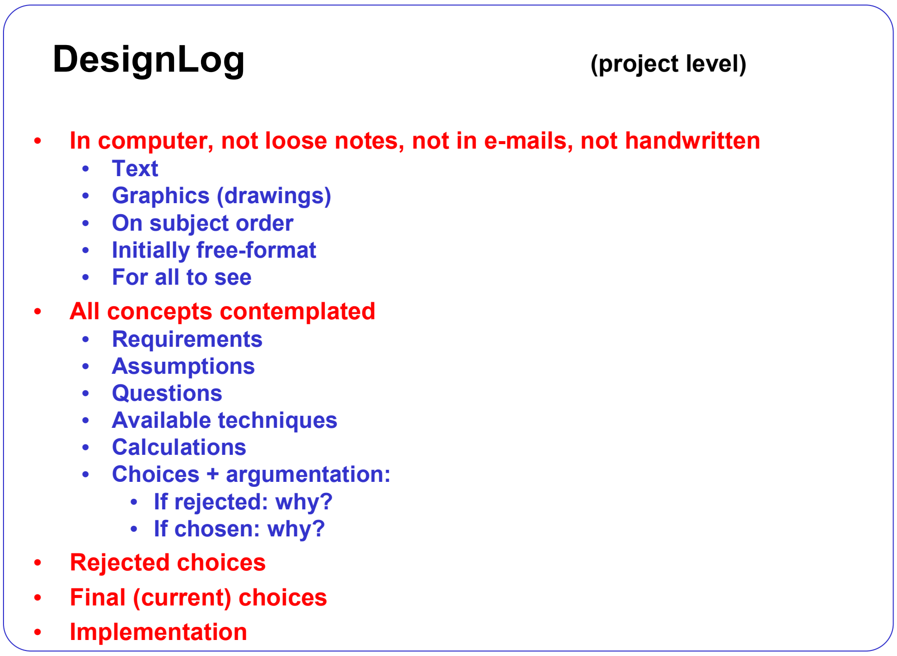
• Wish specification	Thank you, nice input
• Business Case	Why we are doing it
• Requirements	What the project agrees to satisfy
• DesignLog	Selecting the ‘optimum’ compromise
• Specification	This is how we are going to implement it
• Implementation	Code, schematics, hardware, documentation, training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

175



Documents and Sources

176



DesignLog (project level)

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

177

ProcessLog (department / organization level)

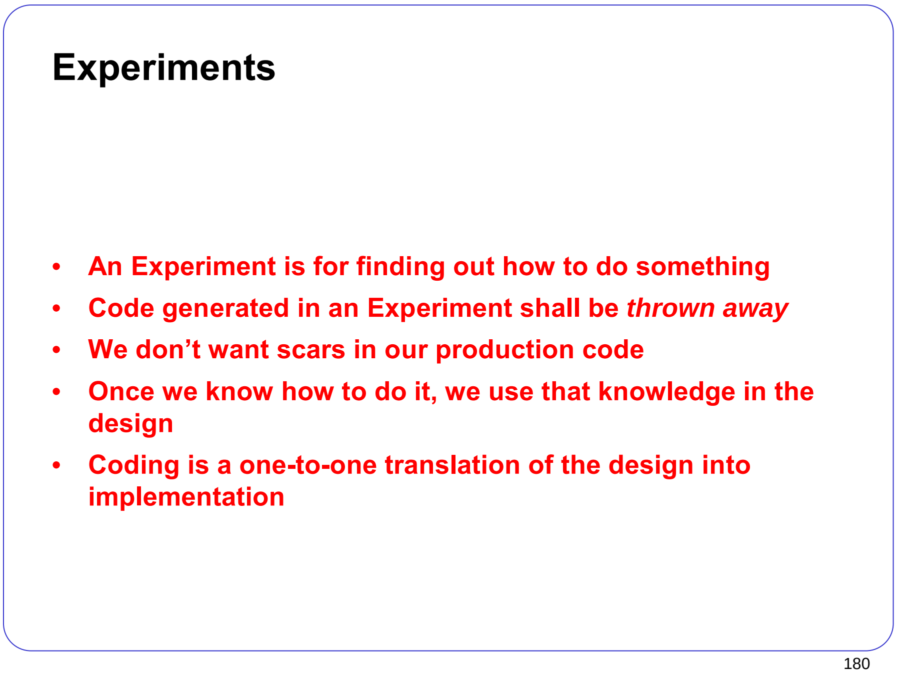
- E.g. to collect review process design and decisions
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Graphics (drawings)
  - On subject order
  - For all to see
- All concepts contemplated
  - Related requirement
  - Assumptions
  - Questions
  - Known techniques
  - Choices + argumentation:
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices

178

Rules

- Design time ≥ Coding time (Implementation time)
- Review time (to be tuned)
  - ≤ 3 pages per hour
  - ≤ 100 LoC per hour
- Review duration
  - ≤ 2 hrs

179



Experiments

- An Experiment is for finding out how to do something
- Code generated in an Experiment shall be *thrown away*
- We don't want scars in our production code
- Once we know how to do it, we use that knowledge in the design
- Coding is a one-to-one translation of the design into implementation

180

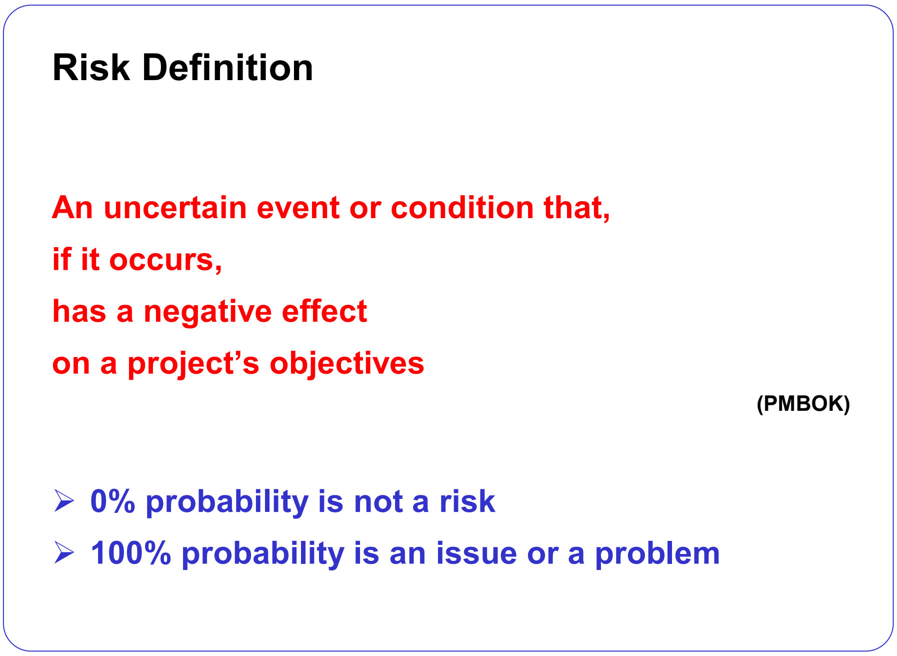
Risk

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

181



Risk Definition

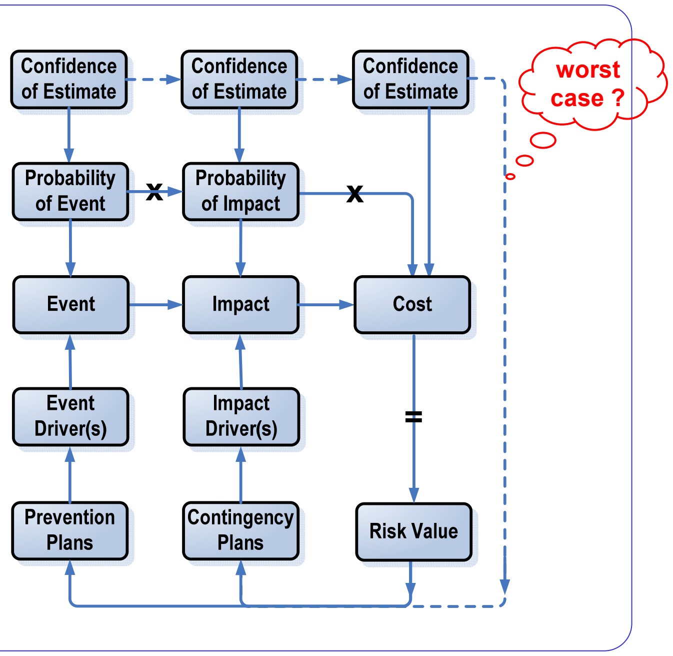
An uncertain event or condition that,
if it occurs,
has a negative effect
on a project's objectives

(PMBOK)

- 0% probability is not a risk
- 100% probability is an issue or a problem

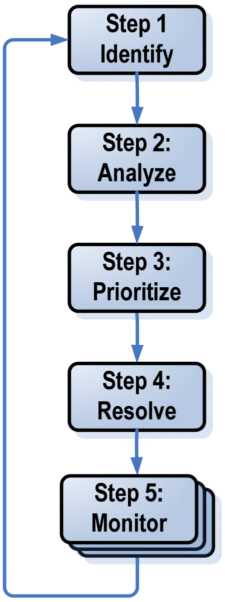
182

Risk Model



$$V_R = P_e * P_i * C$$

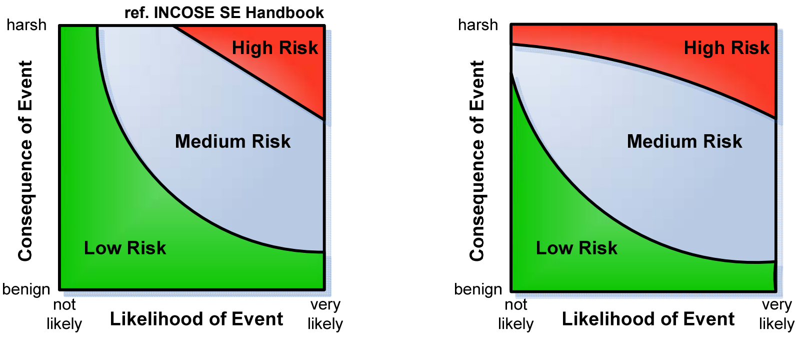
Risk Management



Measures to
- Avoid
- Reduce
- Pass-on
- Accept
- *Control*

Risk

Prioritize Risk?

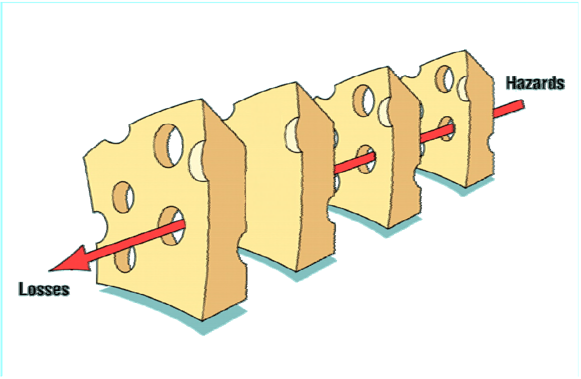


Risk Priority = Likelihood x Consequence ??

185

Swiss Cheese model

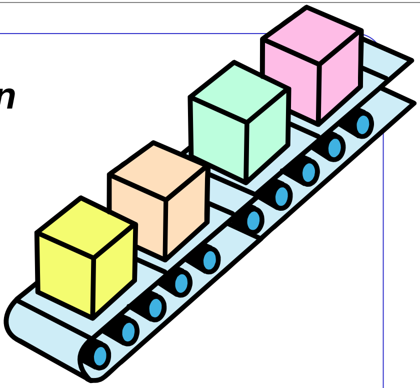
ref James Reason



Can we add some cheese from Holland?

186

Controlling Risk *by design*



- Every project is unique
  (otherwise it's production)

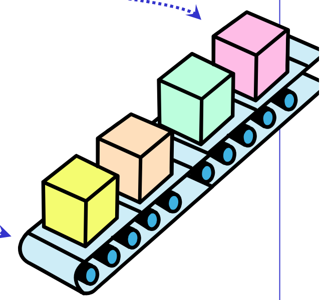
however

- A lot is always the same:
  - Every project is done by people
  - No project is very much unique
  - There are many similarities (*known* risks)
  - So, a lot is predictable
  - We know the Requirements change (don't know *which*)
  - Engineers control risks *by design* (= *engineering*)

187

Many *known* risks are hardly risks

- Most of the real risks are in the product
- Most of the known risks are in the project

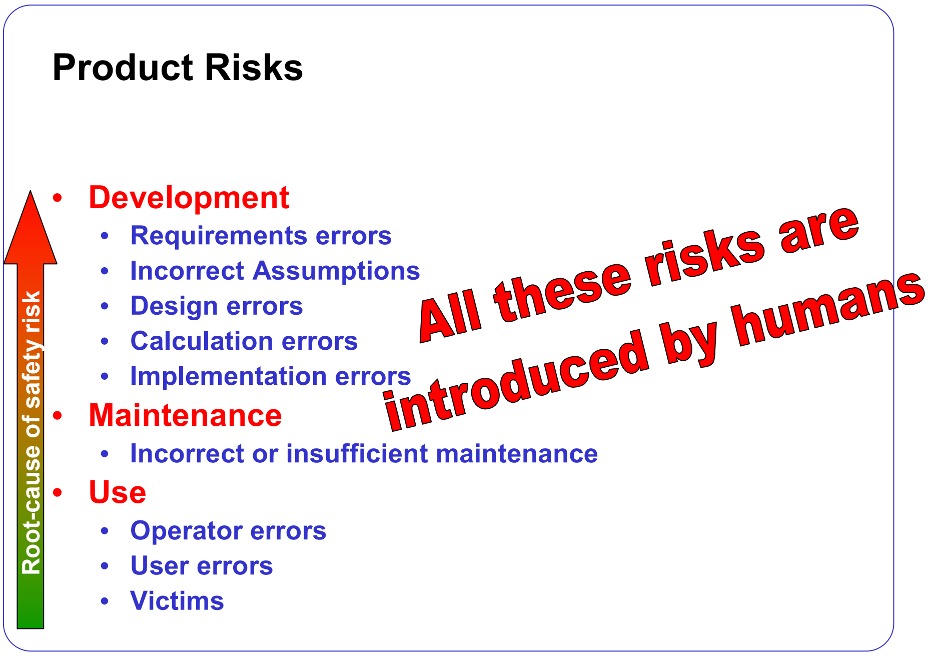


$$V_{Risk} = P_{event} * P_{impact} * C \qquad P_{event} = 1 \qquad P_{impact} \rightarrow 0$$

- We don't only design the product,
- We also *design the project*

- If we control 80% of the risks by design
- We have more time to handle the 20% *real* risks

188



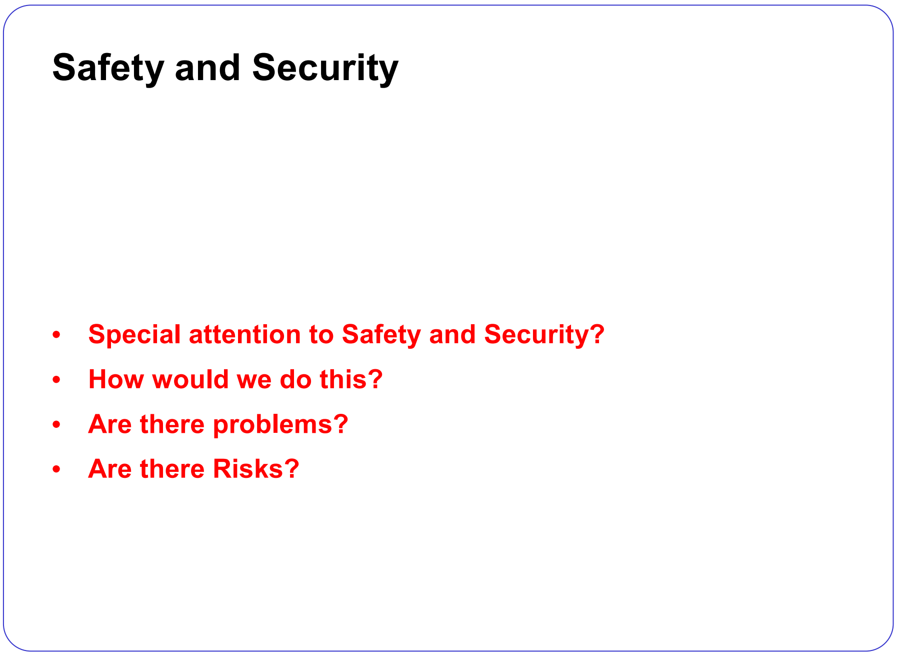
Product Risks

Root-cause of safety risk

- **Development**
  - **Requirements errors**
  - **Incorrect Assumptions**
  - **Design errors**
  - **Calculation errors**
  - **Implementation errors**
- **Maintenance**
  - **Incorrect or insufficient maintenance**
- **Use**
  - **Operator errors**
  - **User errors**
  - **Victims**

All these risks are introduced by humans

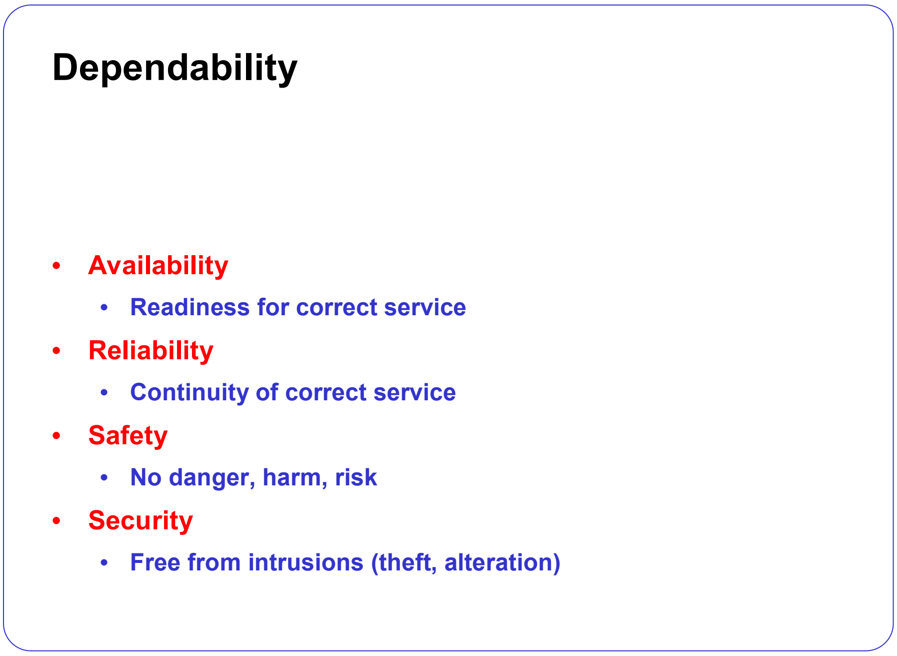
189



Safety and Security

- **Special attention to Safety and Security?**
- **How would we do this?**
- **Are there problems?**
- **Are there Risks?**

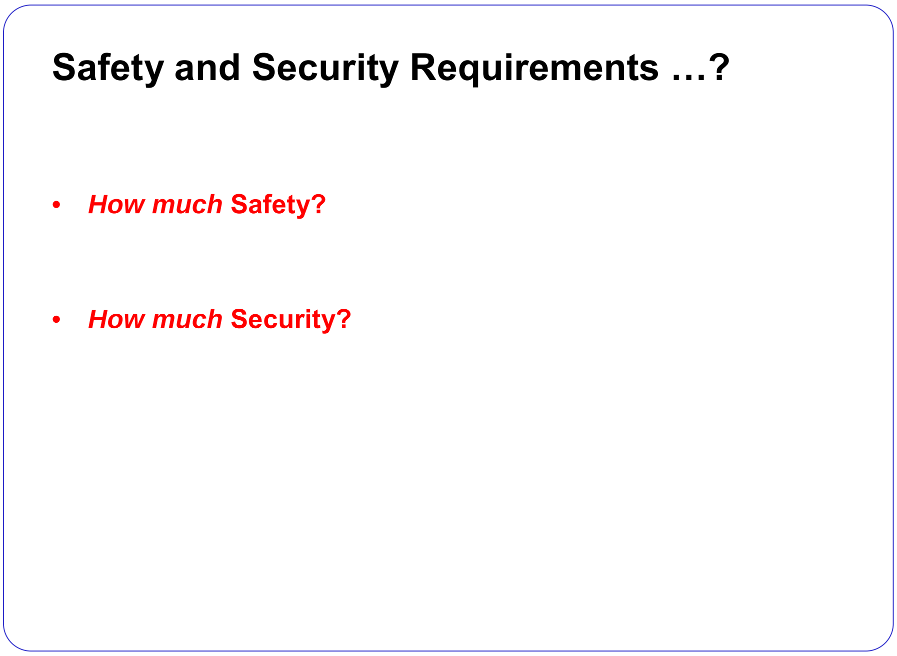
190



Dependability

- Availability
  - Readiness for correct service
- Reliability
  - Continuity of correct service
- Safety
  - No danger, harm, risk
- Security
  - Free from intrusions (theft, alteration)

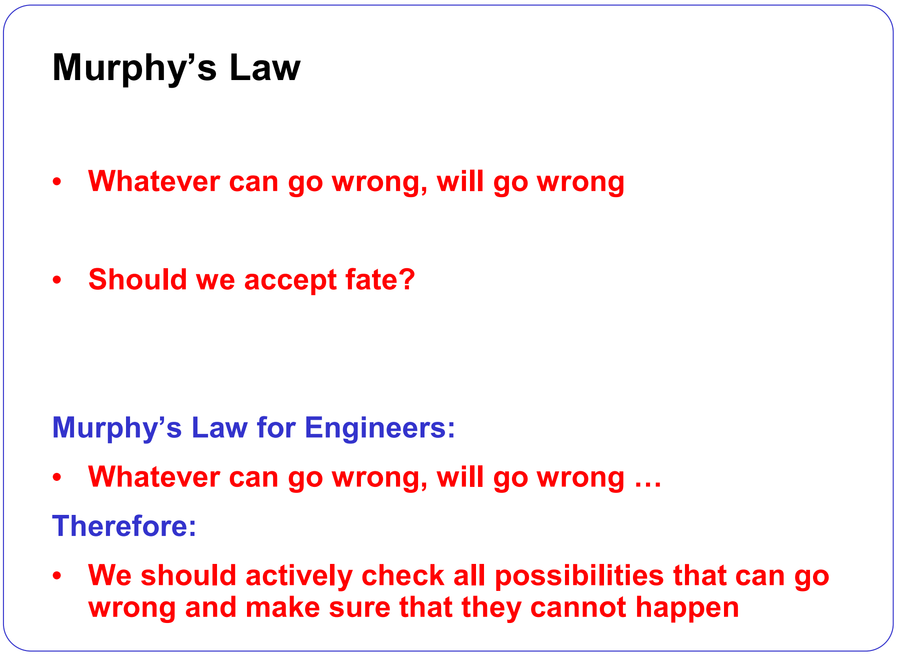
191



Safety and Security Requirements …?

- *How much* Safety?
- *How much* Security?

192



Murphy's Law

- Whatever can go wrong, will go wrong
- Should we accept fate?

Murphy's Law for Engineers:

- 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

193

Human Behaviour

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

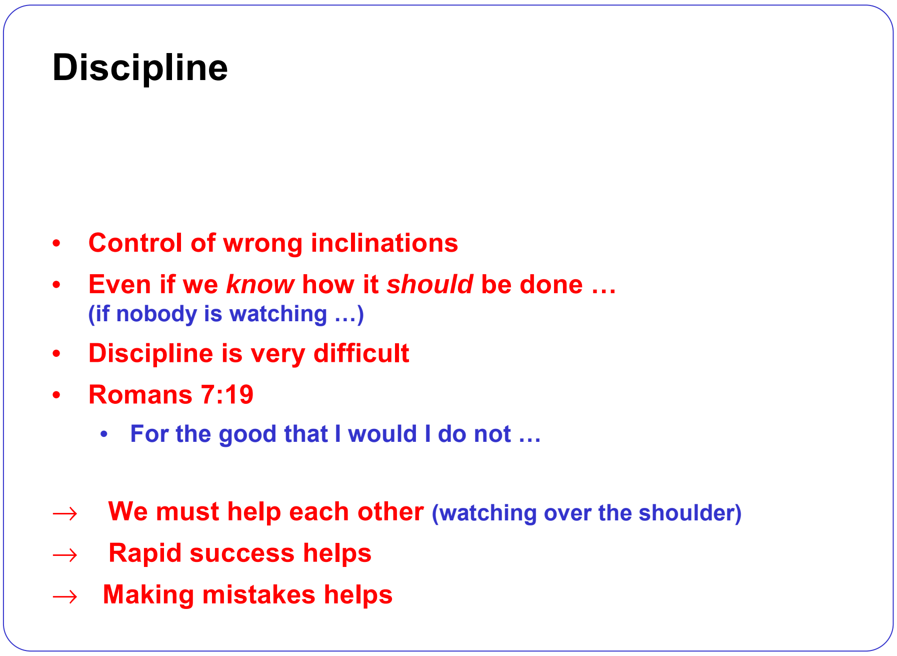
194

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 (as well as developers) *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

195

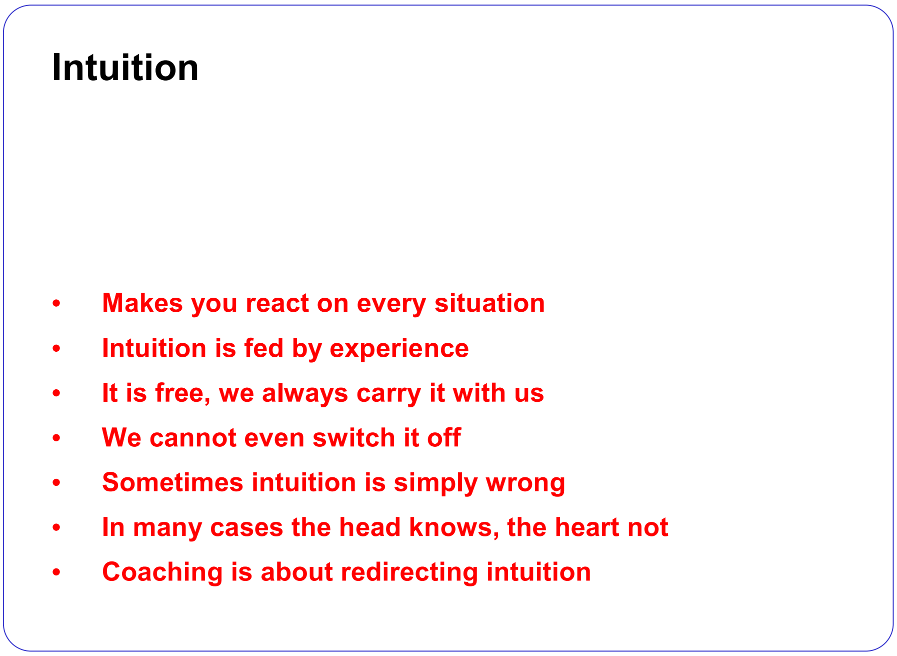


Discipline

- Control of wrong inclinations
- Even if we *know* how it *should* be done …
  (if nobody is watching …)
- Discipline is very difficult
- Romans 7:19
  - For the good that I would I do not …

→ We must help each other (watching over the shoulder)
→ Rapid success helps
→ Making mistakes helps

196



Intuition

- Makes you react on every situation
- Intuition is fed by experience
- It is free, we always carry it with us
- We cannot even switch it off
- Sometimes intuition is simply wrong
- In many cases the head knows, the heart not
- Coaching is about redirecting intuition

197

Is intuition wrong, or is the design wrong?

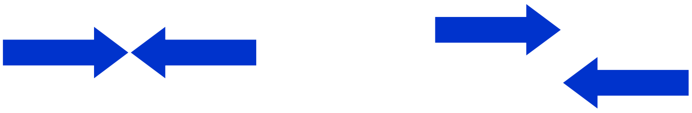
Sleeping pills

Activation pills

198

Communication

- Talking as near as possible along each other



To each other

Along each other

- So, don’t *assume* we understand: *check !*

199

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

200

Perception

- Quick, acute, and intuitive cognition (M-W)
- 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 flavors 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!

201

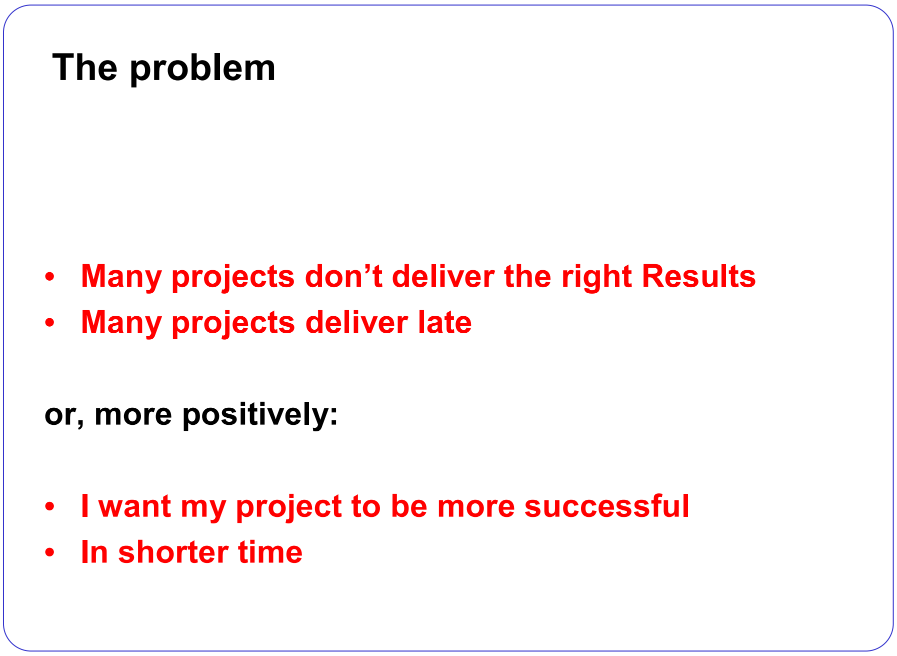
Organizing your Project: Evolutionary Project Management

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

202



The problem

- Many projects don’t deliver the right Results
- Many projects deliver late

or, more positively:

- I want my project to be more successful
- In shorter time

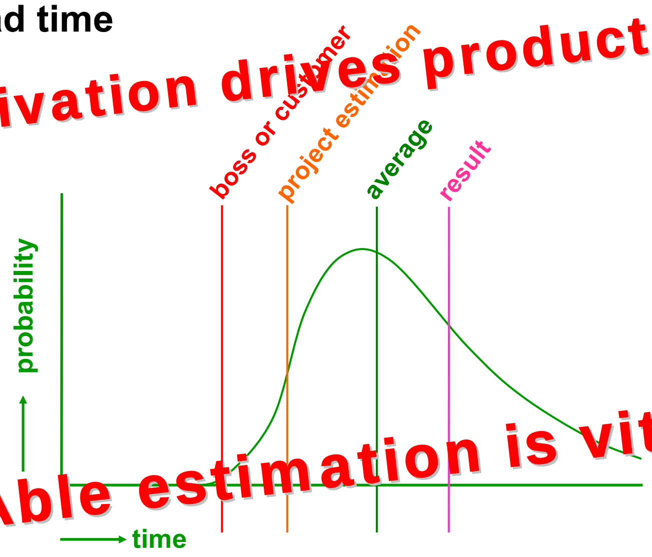
203

How about *your* projects?

- Are *you* any better than *they*?

Lead time

Motivation drives productivity



Able estimation is vital

Estimation Exercise

Are you an optimistic or a realistic estimator?

Let’s find out !

Project:
Multiplying two numbers of 4 figures

How many seconds would you need to complete this Project?

206

Is this what you did?

207

Defect rate

- Before test ?
- After test ?

208

Alternative Design (*how* to solve the requirement)

209

Another alternative design

9876 = 10000 - 124

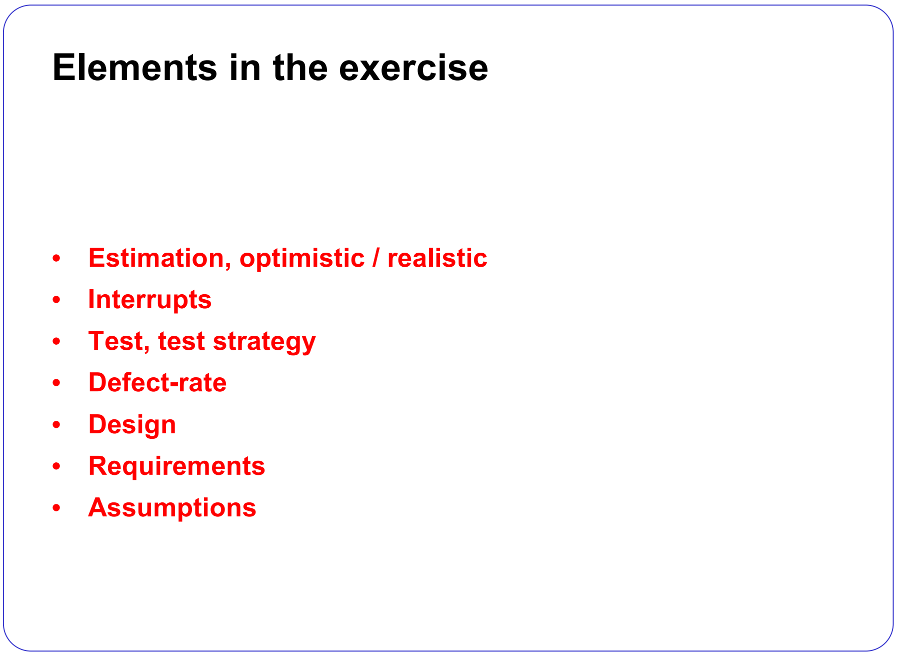
```
1 x 4567 x 100 =               456700
2 x 4567 x 10 =                 91340
2 x 2 x 4567 = 9134 x 2 =       18268   +
124 x 4567 =                   566308

10000 x 4567 =               45670000
124 x 4567 =                   566308   -
                             45103692
```

What was the real requirement?

Assumptions, asssumptions ...

Better assume that many assumptions are wrong.

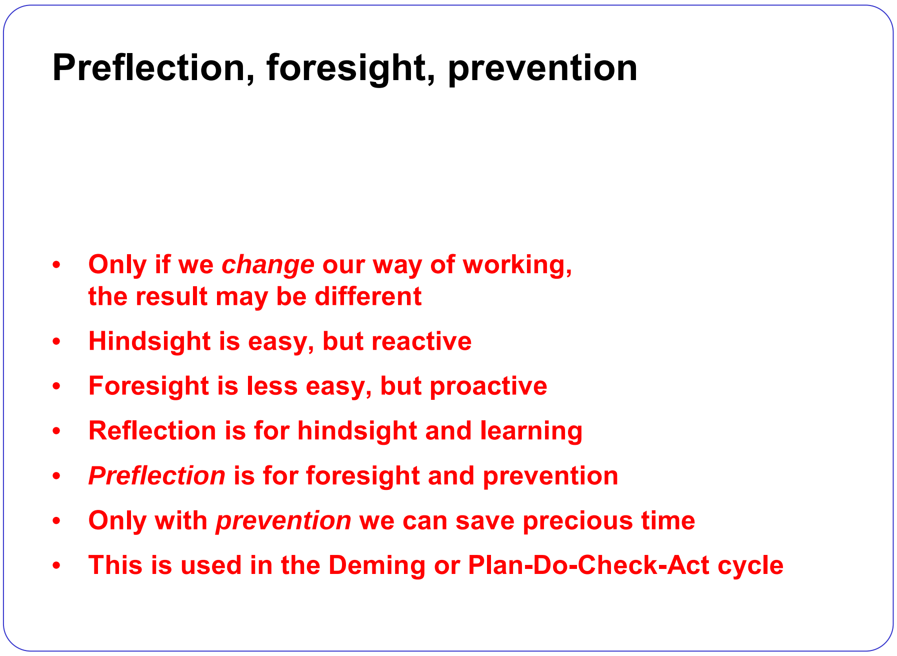
Check !



Elements in the exercise

- Estimation, optimistic / realistic
- Interrupts
- Test, test strategy
- Defect-rate
- Design
- Requirements
- Assumptions

212



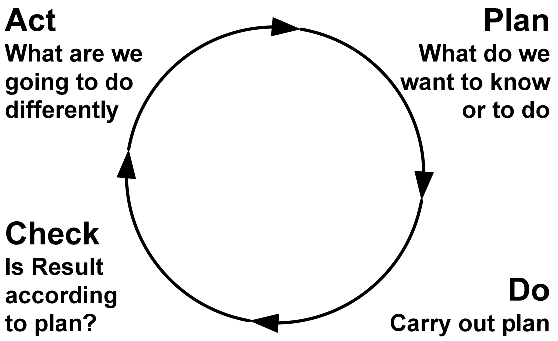
Preflection, foresight, prevention

- 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

213

The essential ingredient: the PDCA cycle

(Deming cycle)



Act
What are we going to do differently

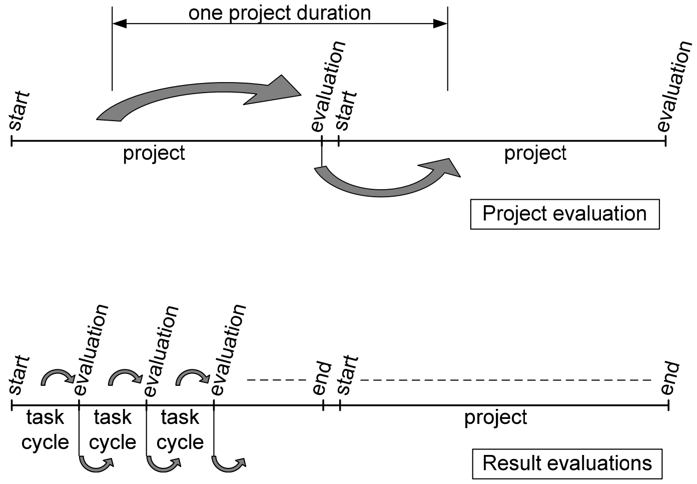
Plan
What do we want to know or to do

Check
Is Result according to plan?

Do
Carry out plan

214

Project evaluations

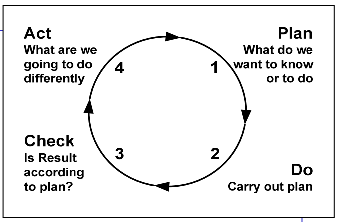


Project evaluation

Result evaluations

215

Knowledge
how to achieve the goal


Act
What are we going to do differently
Plan
What do we want to know or to do
4
1
Check
Is Result according to plan?
3
2
Do
Carry out plan

- Using very short Plan-Do-Check-Act cycles
- Constantly selecting the most important things to do

doing the right things

then we can

- Most quickly learn what the real requirements are
- Learn how to most effectively and efficiently realize these requirements

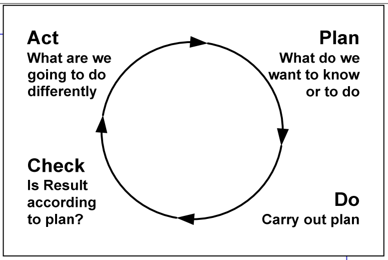
and we can

- Spot problems quicker, allowing more time to do something about them

doing the right things right

216

Evo


Act
What are we going to do differently
Plan
What do we want to know or to do
Check
Is Result according to plan?
Do
Carry out plan

- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for *Product*, *Project* and *Process*, based on ROI
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- We know we are not perfect, but the customer should never find out
- Evo is about delivering Real Stuff to Real Stakeholders doing Real Things

"Nothing beats the Real Thing"

217

Evo elements

- 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
- Agile Review & Inspection
  - Measuring the quality while we are doing, to prevent doing the wrong things

Zero Defects Attitude

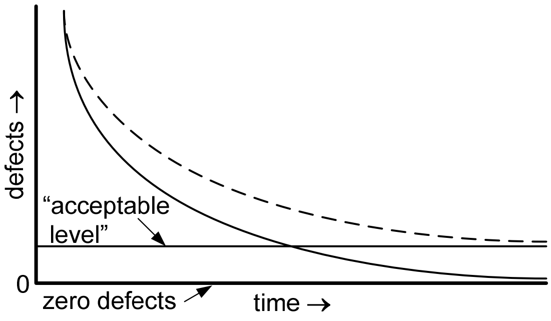
Evo planning

- Weekly TaskCycle
  - Short term planning
  - Optimizing estimation
  - Promising what you can achieve
  - Living up to your promises
- Bi-weekly DeliveryCycle
  - Optimizing the requirements and checking the assumptions
  - Soliciting feedback by delivering Real Results to appropriate and *eagerly waiting* Stakeholders
- TimeLine
  - Getting and keeping control of Time

218

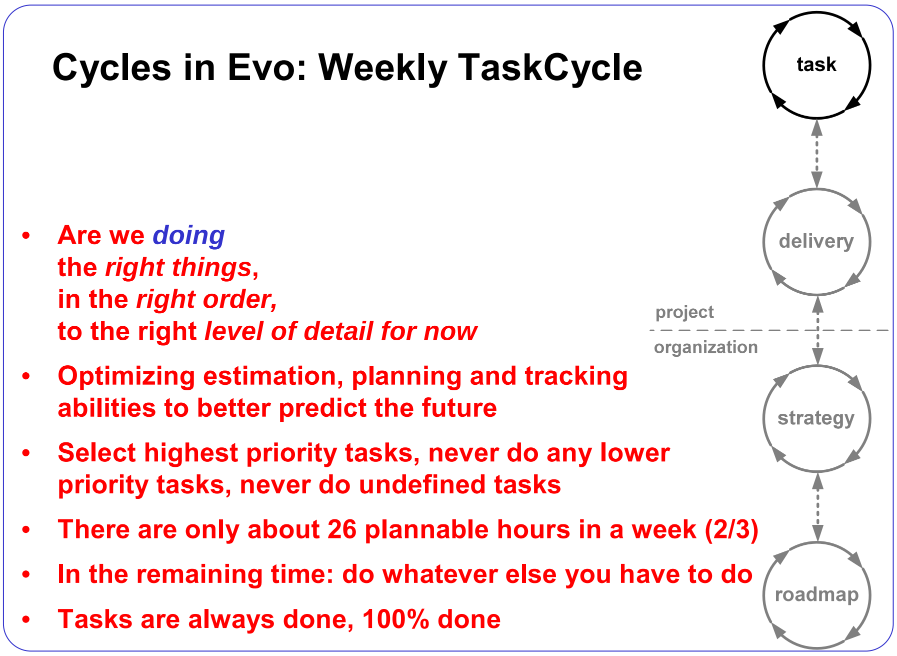
Is defect free software possible?

- ***Zero Defects*** is an asymptote



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

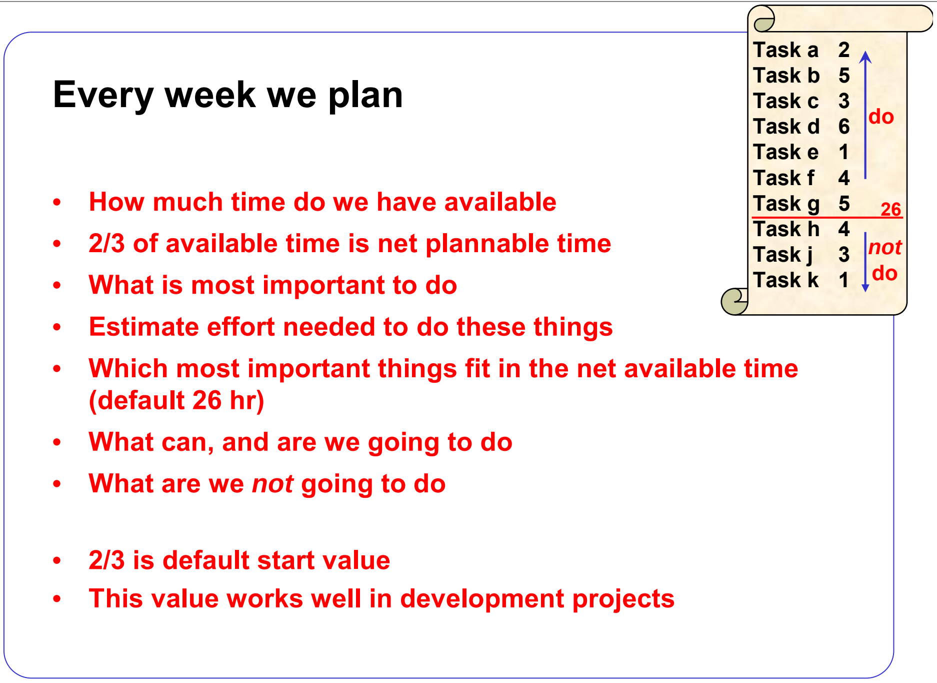
219



Cycles in Evo: Weekly TaskCycle

- Are we *doing*
  the *right things*,
  in the *right order,*
  to the right *level of detail for now*
- Optimizing estimation, planning and tracking abilities to better predict the future
- Select highest priority tasks, never do any lower priority tasks, never do undefined tasks
- There are only about 26 plannable hours in a week (2/3)
- In the remaining time: do whatever else you have to do
- Tasks are always done, 100% done

220



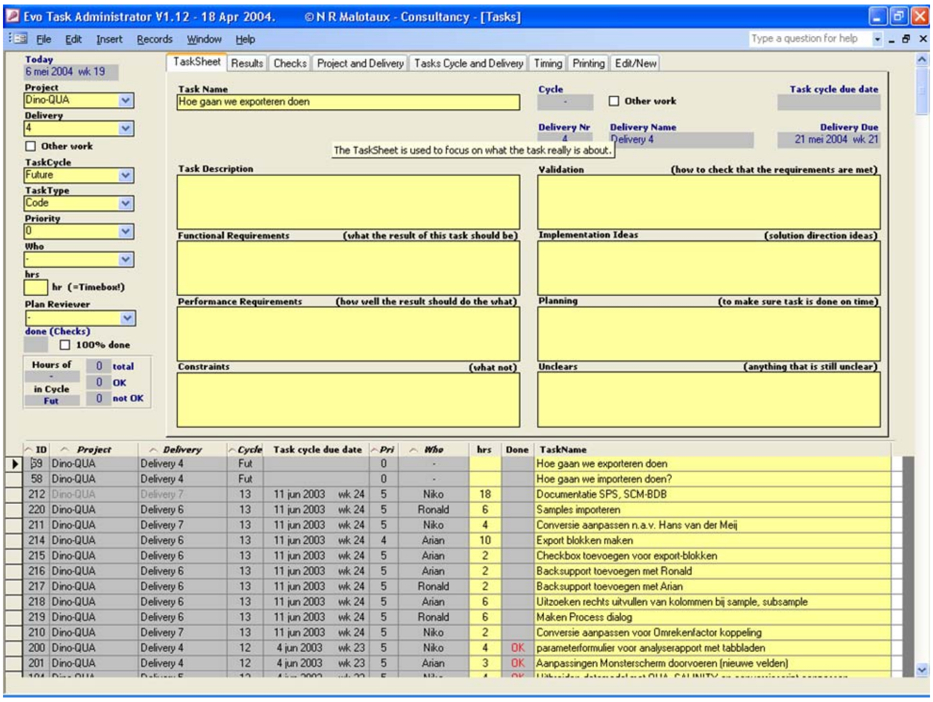
Every week we plan

Task a	2	do
Task b	5	
Task c	3	
Task d	6	
Task e	1	
Task f	4	
Task g	5	26
Task h	4	not do
Task j	3	
Task k	1	

- How much time do we have available
- 2/3 of available time is net plannable time
- What is most important to do
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr)
- What can, and are we going to do
- What are we *not* going to do

- 2/3 is default start value
- This value works well in development projects

221



Evo Task Administrator V1.12 - 18 Apr 2004. ©N R Malotaux - Consultancy - [Tasks]

Today: 6 mei 2004 wk 19
Project: Dino-QUA
Delivery: 4
☐ Other work
TaskCycle: Future
TaskType: Code
Priority: 0
Who: -
hrs: hr (=Timebox!)
Plan Reviewer: -
done (Checks)
☐ 100% done

Hours of - in Cycle Fut: 0 total, 0 OK, 0 not OK

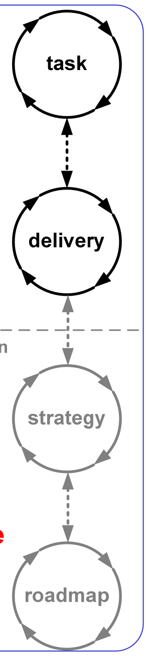
Task Name: Hoe gaan we exporteren doen
Cycle: -
☐ Other work
Task cycle due date:
Delivery Nr: 4
Delivery Name: Delivery 4
Delivery Due: 21 mei 2004 wk 21

The TaskSheet is used to focus on what the task really is about.

Task Description
Validation (how to check that the requirements are met)
Functional Requirements (what the result of this task should be)
Implementation Ideas (solution direction ideas)
Performance Requirements (how well the result should do the what)
Planning (to make sure task is done on time)
Constraints (what not)
Unclears (anything that is still unclear)

ID	Project	Delivery	Cycle	Task cycle due date	Pri	Who	hrs	Done	TaskName
59	Dino-QUA	Delivery 4	Fut		0	-			Hoe gaan we exporteren doen
58	Dino-QUA	Delivery 4	Fut		0	-			Hoe gaan we importeren doen?
212	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	18		Documentatie SPS, SCM-BDB
220	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Samples importeren
211	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	4		Conversie aanpassen n.a.v. Hans van der Meij
214	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	4	Arian	10		Export blokken maken
215	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	2		Checkbox toevoegen voor export-blokken
216	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	2		Backsupport toevoegen met Ronald
217	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	2		Backsupport toevoegen met Arian
218	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	6		Uitzoeken rechts uitvullen van kolommen bij sample, subsample
219	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Maken Process dialog
210	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	2		Conversie aanpassen voor Omrekenfactor koppeling
200	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Niko	4	OK	parameterformulier voor analyserapport met tabbladen
201	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Arian	3	OK	Aanpassingen Monsterscherm doorvoeren (nieuwe velden)

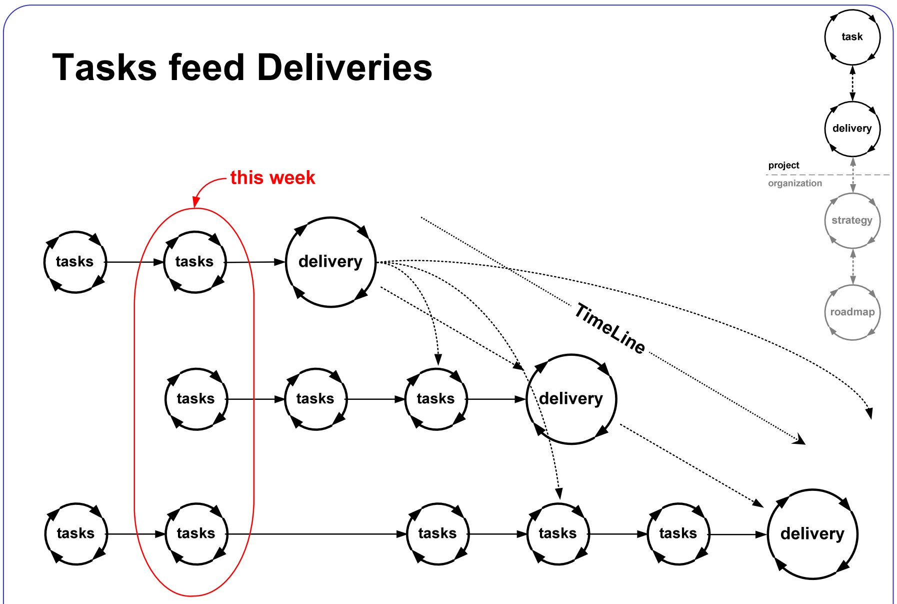
Cycles in Evo: DeliveryCycle



- Are we delivering the *right things*, in the *right order* to the right *level of detail for now*
- Optimizing requirements and checking assumptions
- What will generate the optimum feedback
- We deliver only to *eagerly waiting* stakeholders
- Delivering the juiciest, most important stakeholder values that can be made in the least time
- What will make Stakeholders more productive now
- Not more than 2 weeks

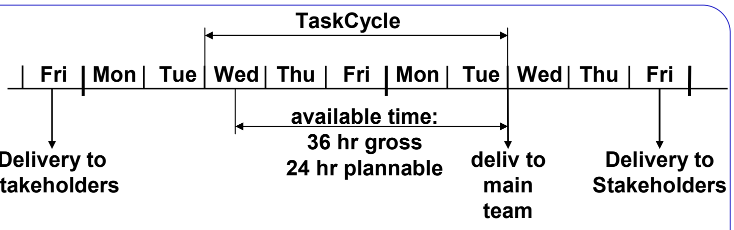
223

Tasks feed Deliveries



224

Designing a Delivery



Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

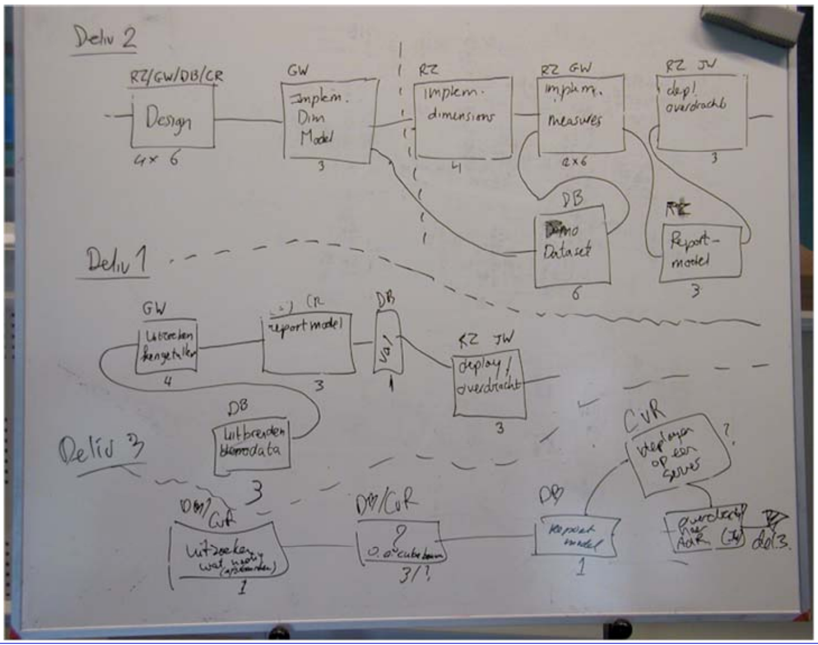
Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	42

Gregory (later)	
Draft design	0
Finish design	0
...	
Repair deliv	0
...	

Zero Defects Attitude

Jerome	
XMLa	3
XMLb	3
...	

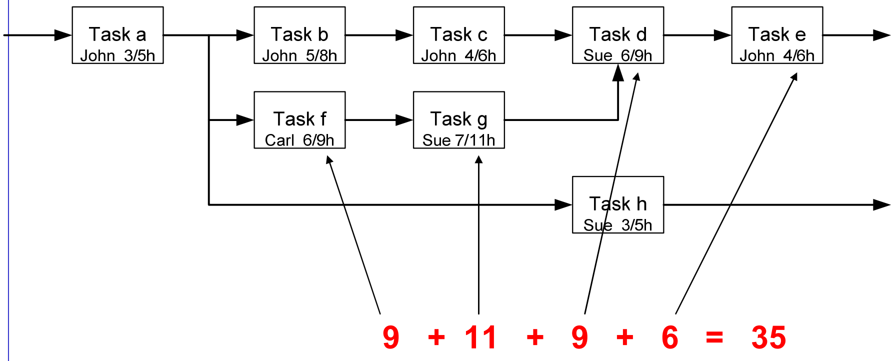
225



226

PERT (Project Evaluation Review Technique)

used for *Designing* a Delivery



Task a John 3/5h

Task b John 5/8h

Task c John 4/6h

Task d Sue 6/9h

Task e John 4/6h

Task f Carl 6/9h

Task g Sue 7/11h

Task h Sue 3/5h

9 + 11 + 9 + 6 = 35

227

Agile, but Still On Time

- Organizing the work in very short cycles
- To make sure we are doing the right things
- And that we are doing it the right way
- So, we already work more efficiently

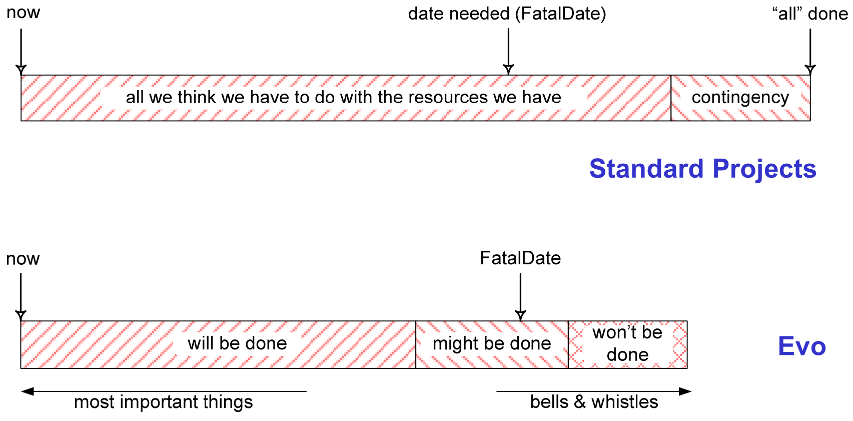
but ...

- How do we make sure the whole project is done on time?

228

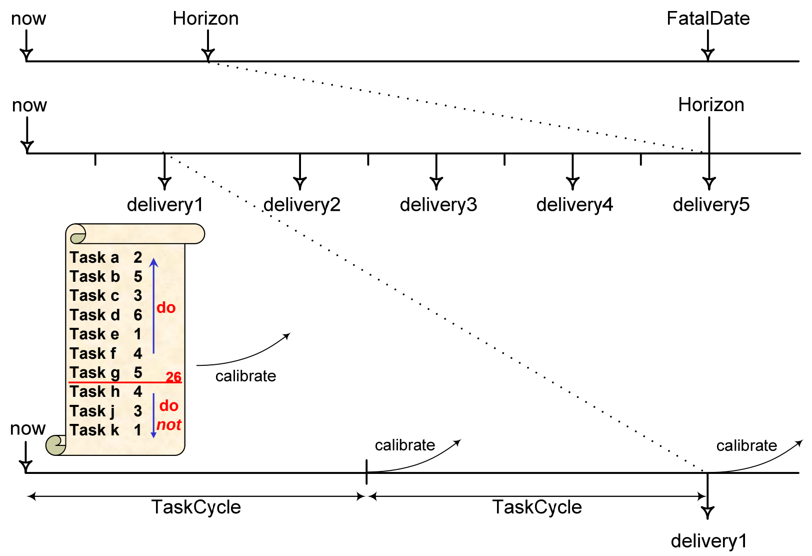
TimeLine

What the customer wants, he cannot afford



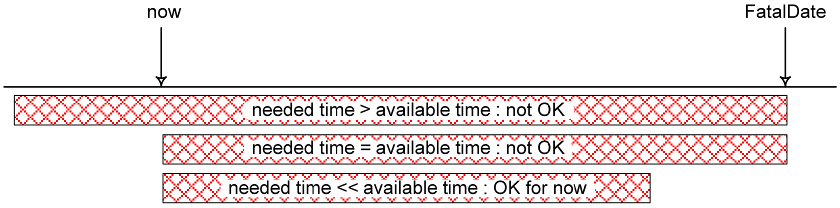
229

Result to Tasks and back



230

If it doesn't fit ...



needed time > available time : not OK

needed time = available time : not OK

needed time << available time : OK for now

231

Options in case things don’t fit in time

- If we ostrich till the end,
  things will be left undone *randomly*
- We use the early warning to *do something about it*:
  - Adding people
  - Hoping for the best
  - Going for it
  - Working Overtime
  - Adding time: Moving the deadline
  - Saving time

232

Deceptive options

- Hoping for the best (fatalistic)
- Going for it (macho)
- Overtime (fooling our self)
- Moving the deadline
  - Parkinson’s Law

    Work expands to fill the time for its completion
  - Student Syndrome

    Starting as late as possible, only when the pressure of the FatalDate is really felt

233

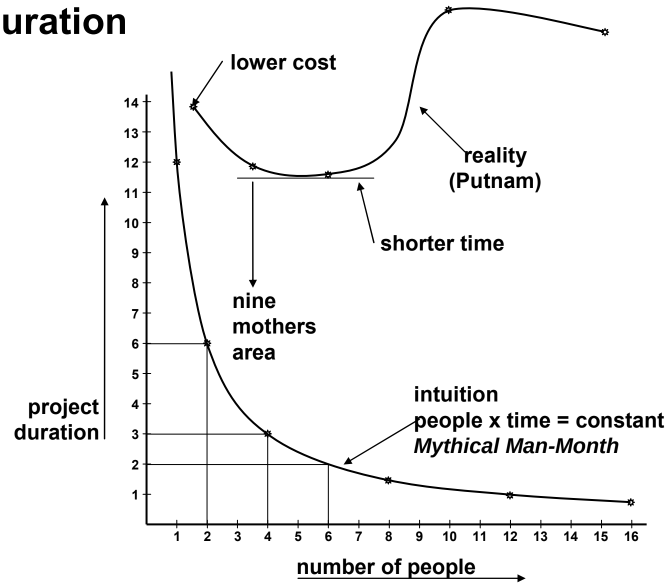
Adding people to a late project ...

makes it later

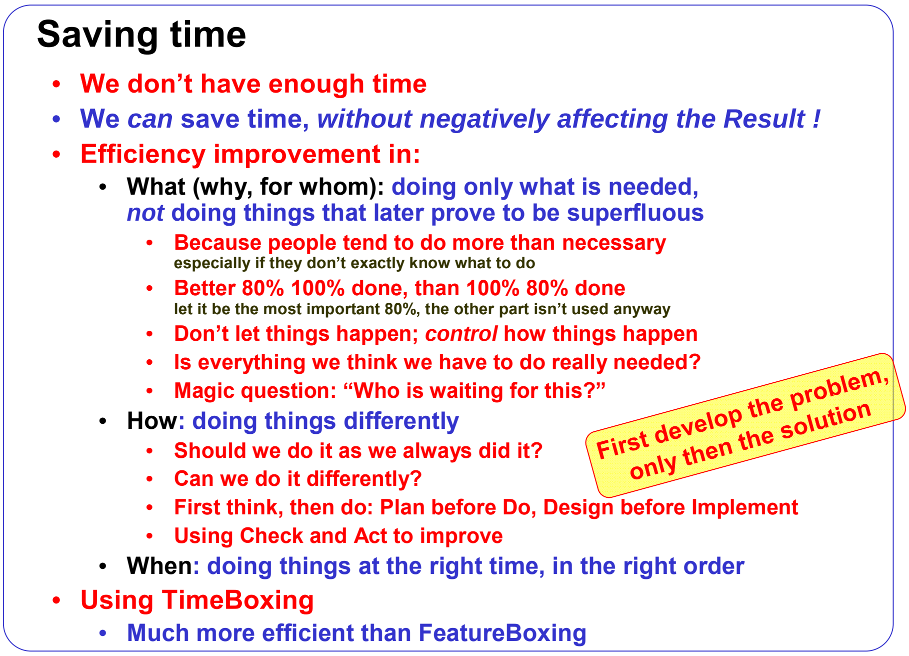
(Brooks' Law, 1975)

234

Project-duration



235

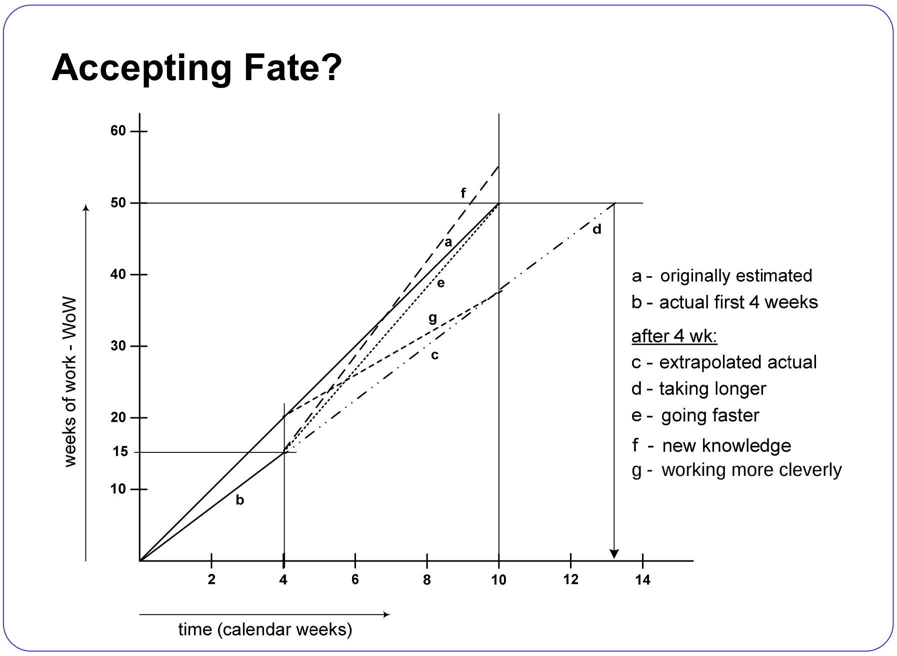


Saving time

- **We don't have enough time**
- **We *can* save time, *without negatively affecting the Result !***
- **Efficiency improvement in:**
  - **What (why, for whom): doing only what is needed, *not* doing things that later prove to be superfluous**
    - **Because people tend to do more than necessary**
      especially if they don't exactly know what to do
    - **Better 80% 100% done, than 100% 80% done**
      let it be the most important 80%, the other part isn't used anyway
    - **Don't let things happen; *control* how things happen**
    - **Is everything we think we have to do really needed?**
    - **Magic question: "Who is waiting for this?"**
  - **How: doing things differently**
    - **Should we do it as we always did it?**
    - **Can we do it differently?**
    - **First think, then do: Plan before Do, Design before Implement**
    - **Using Check and Act to improve**
  - **When: doing things at the right time, in the right order**
- **Using TimeBoxing**
  - **Much more efficient than FeatureBoxing**

First develop the problem, only then the solution

236



Accepting Fate?

a - originally estimated
b - actual first 4 weeks

after 4 wk:
c - extrapolated actual
d - taking longer
e - going faster
f - new knowledge
g - working more cleverly

237

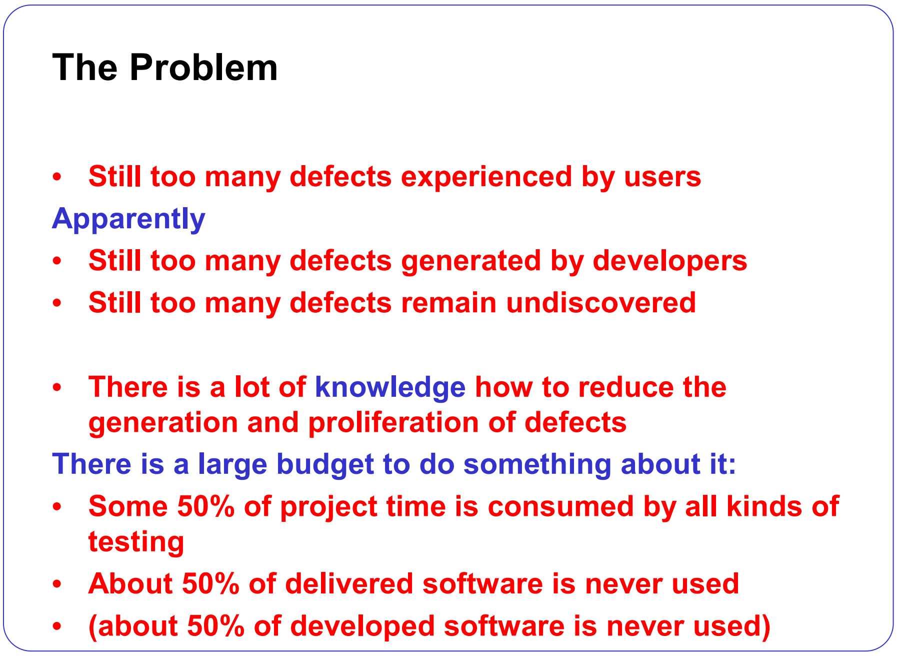
Testing with Evo

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

238



The Problem

- Still too many defects experienced by users

Apparently

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

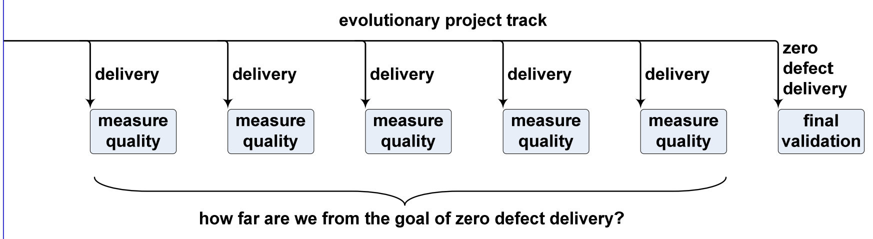
- There is a lot of knowledge how to reduce the generation and proliferation of defects

There is a large budget to do something about it:

- Some 50% of project time is consumed by all kinds of testing
- About 50% of delivered software is never used
- (about 50% of developed software is never used)

239

Current Evo Testing



evolutionary project track

delivery

measure quality

delivery

measure quality

delivery

measure quality

delivery

measure quality

delivery

measure quality

zero defect delivery

final validation

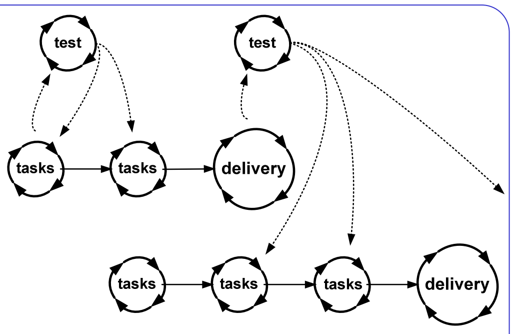
how far are we from the goal of zero defect delivery?

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

Further Improvement

- Testers focus on a clear goal
- Finding defects is not the goal
- Project Success is
- Tester's customer is "the developers"
- Testers select and use any method appropriate
- Testers check work in progress *even before* it is finished (see SQC)
- Testers solve the Review and Inspection organizing problem
- Testing is organized the Evo way, entangling intimately with the development process

Evo cycles for Testing



- 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 check work in progress *even before* it is finished

242

Developers *and* Testers are constantly optimizing

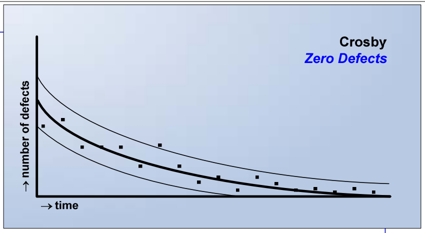
- The product
  how to arrive at the most effective product (Goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods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

243

Testers are part of the project

- Participating in the weekly 3-step process:
  1. Individual Preparation
  2. 1-to-1's with project management
     - Project Manager (project issues)
     - Architect (product issues)
     - Test Manager (testing issues)
  3. Team meeting: Synchronization and synergy with the team
- Testers see what developers are doing:
  - No ambiguity with what the developers are doing
  - To which degree requirements are implemented
- Testers can help optimizing Review processes

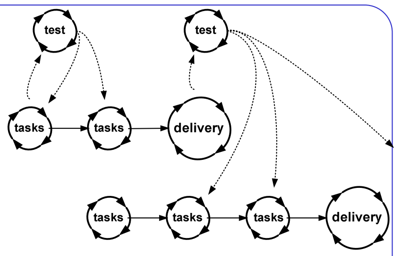
Metrics



Don't *improve* non-value-adding activities - better *eliminate* them

- Estimation - planning - tracking
  - If estimation is a TimeBox, tracking is a "zero activity"
- *Defects per kLoC* or *Defects per Page*
  Stop counting defects, it conveys a bad message. Decrease numbers *by design.*
- *Incoming defects* per month (by test, by user)
  Don't count. Do something. Users shouldn't experience problems.
- *Defect detection effectiveness* or *Inspection yield*
  - Yield is 30% ~ 80%; testers are human after all
  - Zero defects at user means zero defects before final test
  - Whether that is difficult is beside the point

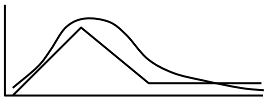
More metrics



- *Cost to find and fix a defect*
  - The less defects the higher the cost per defect
  - This was a bad metric anyway
- *Closed defects per month*
  - Closing depends on prioritizing process, through Candidate Tasks List
- *Age of open customer found defects*
  - Purpose of many metrics seems to be *policing*: not trusting people to take appropriate action
  - In Evo we take appropriate action
- *Remaining defects*
  - Still useful as measure of Prevention success

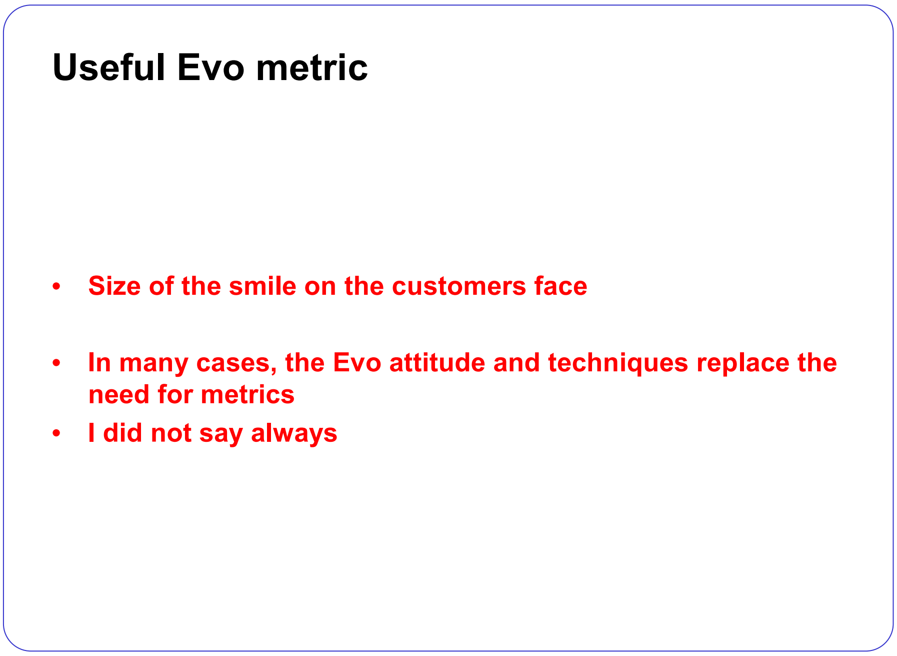
246

When are we done with testing?



- Conventional:
  - Number of bugs found per day less than n
  - Defect backlog decreased to zero
  - Prediction by curve fitting based on early found defect numbers
  - Using historical data
  - Other?
- Evo:
  - The project is ready at the agreed date, or earlier
  - That includes testing

247



Useful Evo metric

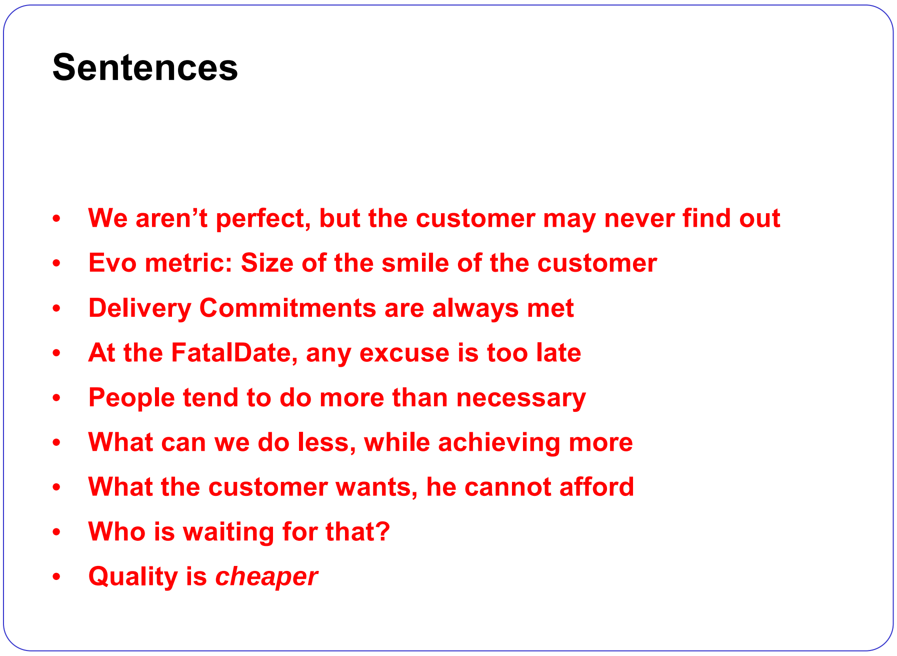
- Size of the smile on the customers face
- In many cases, the Evo attitude and techniques replace the need for metrics
- I did not say always

248

Links

- www.gilb.com
  Tom Gilb's website: Evo guru
- www.malotaux.nl
  Niels' activities: Evo evangelist
- www.malotaux.nl/nrm/Evo
  Evo pages
- www.malotaux.nl/nrm/Insp
  Inspection pages
- www.malotaux.nl/nrm/pdf/MxEvo.pdf
  Evolutionary Project Management Methods
  (issues and first - 2001 - experience)
- www.malotaux.nl/nrm/pdf/Booklet2.pdf
  How Quality is Assured by Evolutionary Methods
  (more recent - 2004 - practical implementation experience)
- www.malotaux.nl/nrm/pdf/EvoTesting.pdf
  Optimizing the Contribution of Testing to Project Success (2005)
- www.malotaux.nl/nrm/pdf/EvoRisk.pdf
  Controlling Project Risk *by Design* (2006)
- www.malotaux.nl/nrm/pdf/TimeLine.pdf
  TimeLine: How to get and keep control over longer periods of time (2007)
- www.malotaux.nl/nrm/Evo/ETAF.htm
  Download the Evo Task Administrator (ETA) tool
  (expects MSAccess 2000~2003)

249



Sentences

- We aren’t perfect, but the customer may never find out
- Evo metric: Size of the smile of the customer
- Delivery Commitments are always met
- At the FatalDate, any excuse is too late
- People tend to do more than necessary
- What can we do less, while achieving more
- What the customer wants, he cannot afford
- Who is waiting for that?
- Quality is *cheaper*

250

What now?

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

251

Nice story, but ...

- What are we going to do differently tomorrow?
- Who will be the Champion?
  - Owner of the process
  - Protecting and optimizing the process
  - Keeping the ProcessLog
- Which Review process to use
  - For Requirements
  - For Design
  - For Code
  - ...

252

Inspection Master Plan

Owner: Niels Malotaux – Version 1.01 – 23 Nov 2001

Inspection no.

Date requested:

who	name	init	tel	e-mail	role	scan	time	min / page	check	time	min / page	rule set
Leader					Leader							
Author					Author							
Checker					-							
Checker					-							
Checker					-							
Checker					-							
Checker					-							

doc	owner	init	tel	e-mail	docname	date	ver	Location	insp status	maj / page
Product										
Reference										
Source										
Source										
Source										
Source										

meeting	date	location	start	end
KickOff				
Logging				

Individual checker data collection To be filled in by each checker, ***before*** logging meeting	**Checker:**	
	scan	**check**
Time spent (X.X hrs)		
Pages studied		
Majors		
Super majors (project threat)		
Minors		
Process Improvements		
Questions		

Instructions

Inspection goals: Getting the product exited
Learning Inspections

Strategy to meet goal: Do Inspection, find as many issues as possible
Note: The brainstorm will initially be replaced by:
- 30 min. discussion about what you think of this inspection process
- 30 min. Just In Time Training on the subject of the document

Optimum checking rate: 60 min per page
At first Inspections we will use about 30 min per logical page

Exit condition: < 2 major defects remaining per page

Assignment for this Inspection:

Please check the sheets against all source document and rule set GE. See Inspection Manual. In this manual you can also find the procedure for checking (Procedure for Checker during Checking: CC). Read this procedure to know what to do during checking.

Inspection Master Plan

Owner: Niels Malotaux – Version 1.01 – 23 Nov 2001

Inspection no.

Date requested:

who	name	init	tel	e-mail	role	scan	time	min / page	check	time	min / page	rule set
Leader					Leader							
Author					Author							
Checker					-							
Checker					-							
Checker					-							
Checker					-							
Checker					-							

doc	owner	init	tel	e-mail	docname	date	ver	Location	insp status	maj / page
Product										
Reference										
Source										
Source										
Source										
Source										

meeting	date	location	start	end
KickOff				
Logging				

Individual checker data collection To be filled in by each checker, ***before*** logging meeting	**Checker:**	
	scan	**check**
Time spent (X.X hrs)		
Pages studied		
Majors		
Super majors (project threat)		
Minors		
Process Improvements		
Questions		

Instructions

Inspection goals: Getting the product exited
Learning Inspections

Strategy to meet goal: Do Inspection, find as many issues as possible
Note: The brainstorm will initially be replaced by:
- 30 min. discussion about what you think of this inspection process
- 30 min. Just In Time Training on the subject of the document

Optimum checking rate: 60 min per page
At first Inspections we will use about 30 min per logical page

Exit condition: < 2 major defects remaining per page

Assignment for this Inspection:

Please check the sheets against all source document and rule set GE. See Inspection Manual. In this manual you can also find the procedure for checking (Procedure for Checker during Checking: CC). Read this procedure to know what to do during checking.

Inspection Issue Log

InspectionID

Date

Item types:
S, M, m, Q, P, N

Item No	Doc ref	Doc page	Scan/Check	Location on page	Type of item	Checklist or rule tag	Description If long explanation, remind: "say it in 7 words max!"	Number of occurr	time ref	who	Editor note	done
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												

Inspection Issue Log

Item types:
S, M, m, Q, P, N

InspectionID

Date

Item No	Doc ref	Doc page	Scan/Check	Location on page	Type of item	Checklist or rule tag	Description If long explanation, remind: "say it in 7 words max!"	Number of occurr	time ref	who	Editor note	done
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												