Improving the effectiveness of Reviews and Inspections

www.malotaux.nl/conferences
www.malotaux.nl/booklets
www.malotaux.nl/inspections

Niels Malotaux

N R Malotaux Consultancy

+31-655 753 604 niels@malotaux.nl www.malotaux.nl

Niels Malotaux

Project and Organizational Coach

Helping projects and organizations very quickly to become

- More effective – doing the right things better
- More efficient – doing the right things better in less time
- Predictable – delivering as predicted

Getting projects back on track

Helping with Architecture/Design/Review of electronics/firmware/software

Result Management

Ultimate Goal of a What We Do

Quality on Time

Delivering the Right Result at the Right Time,
wasting as little time as possible (= efficiently)

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

Evolutionary Project Management (Evo)

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

Right product

Evo Project Planning

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

Efficiency of what we do

Effectiveness of what we do

Right time

What will happen and what will we do about it ?

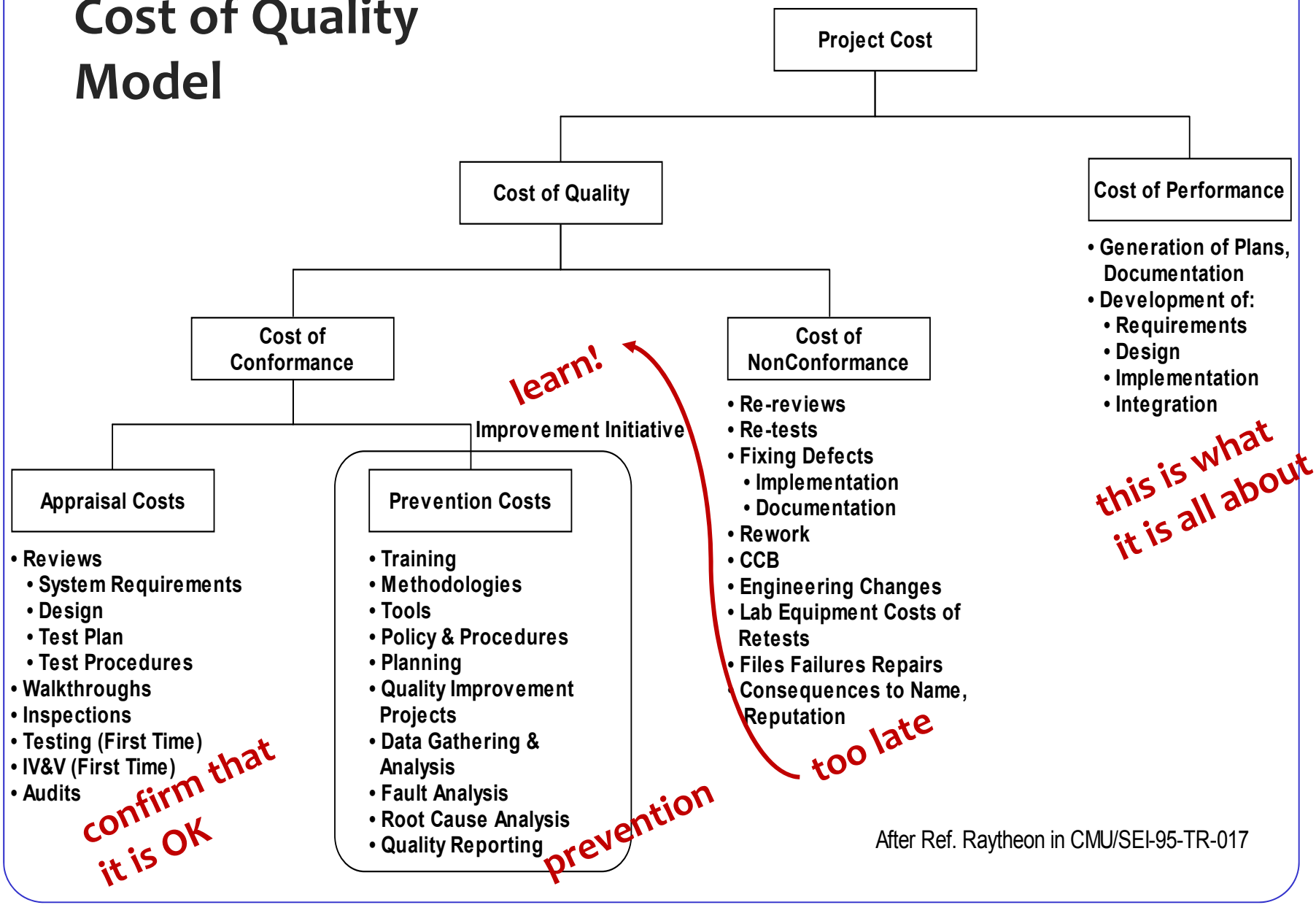
Quality On Time?

- Do your projects normally produce Quality ?
- Do your projects deliver the Right Results On Time ?
  - Yes, also testing projects !
- What is
  - Right Results ?
  - On Time ?

Does quality cost more ?

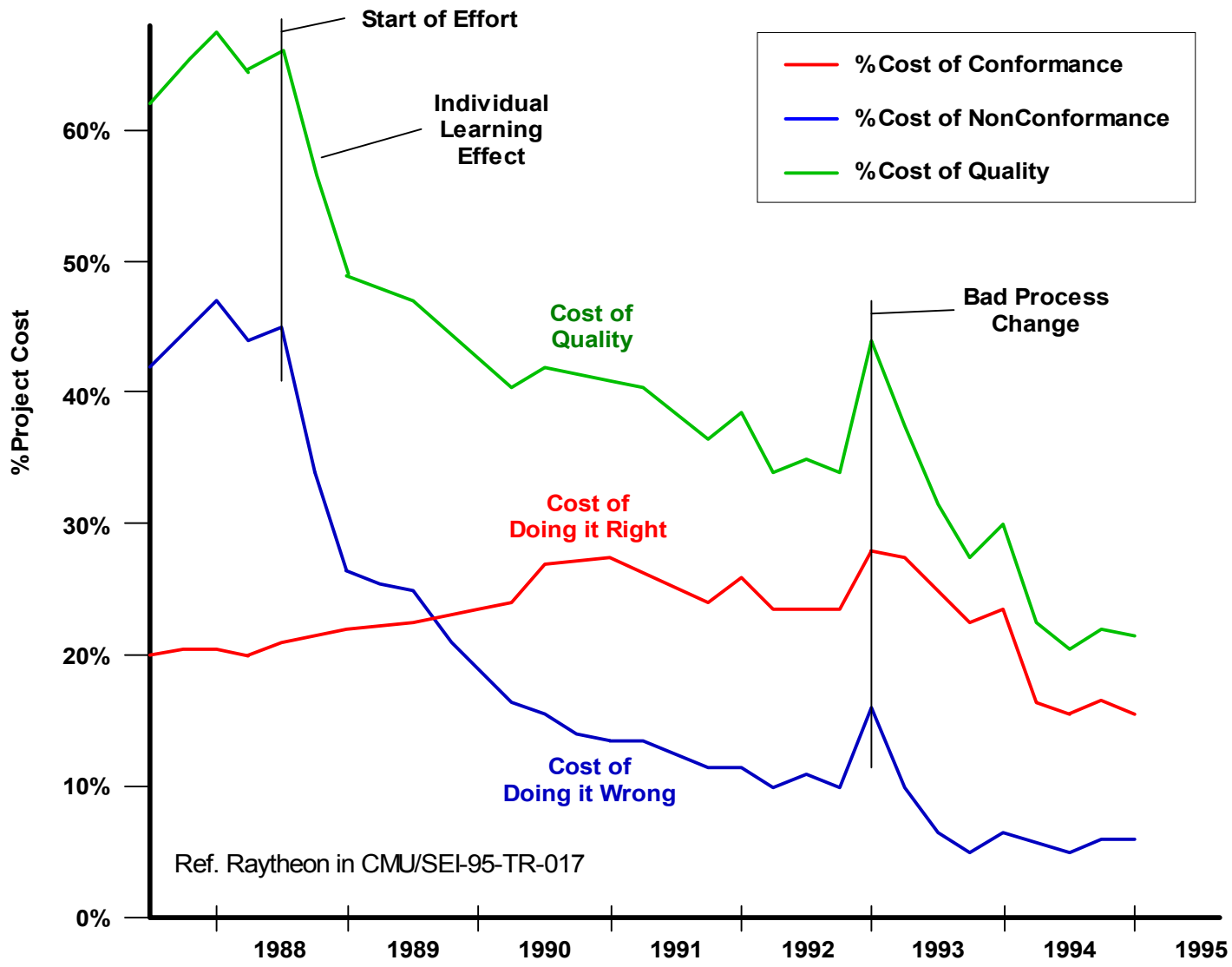
- **The cost is not in the quality**
- **The cost is in the non-quality**

Cost of Quality Model

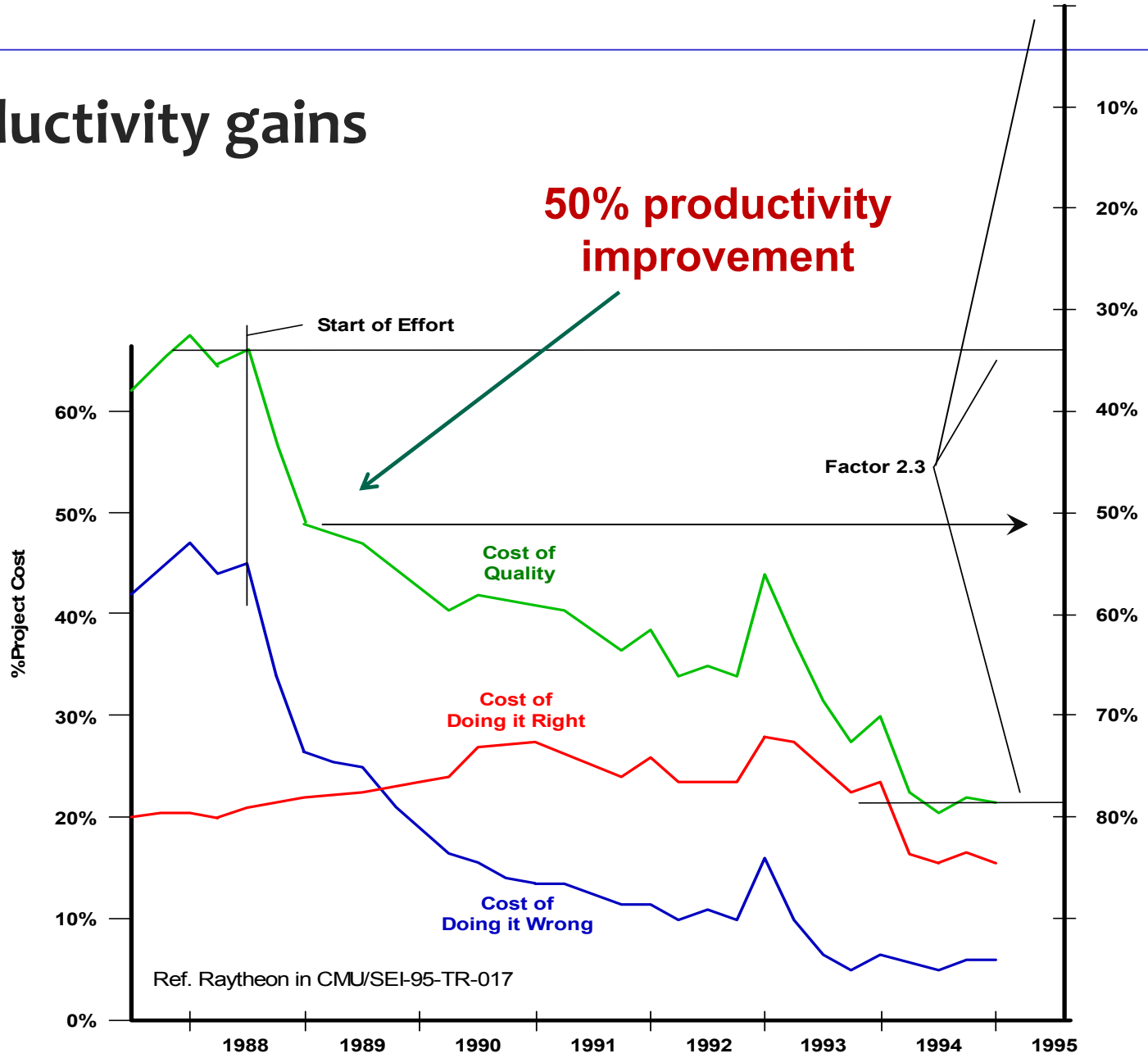


After Ref. Raytheon in CMU/SEI-95-TR-017

Cost of Quality

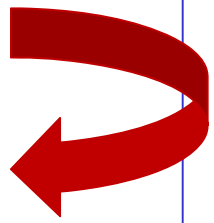


Productivity gains

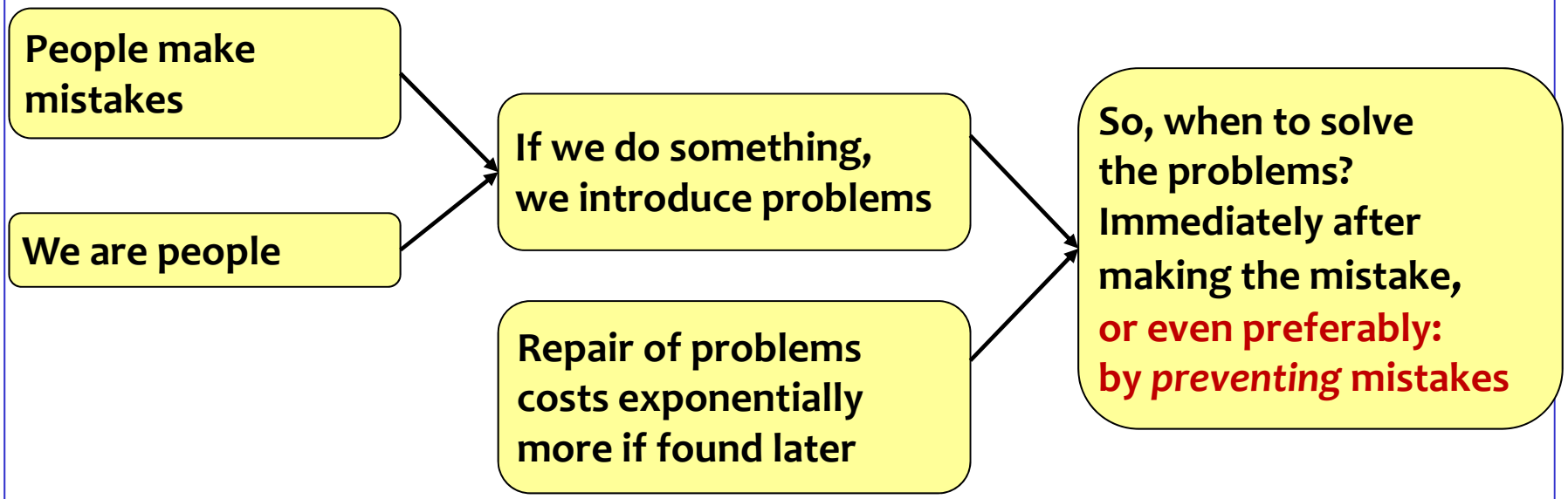


How to cut the waste ?

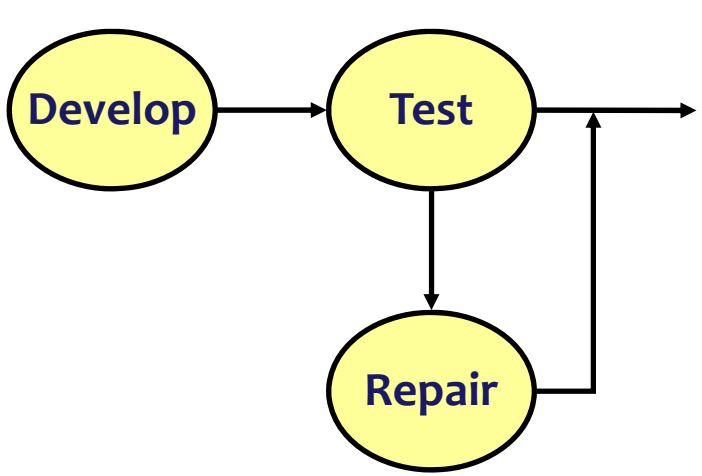
- **Hope ??** - Does nothing
- **Test ?** - To show how much waste we already produced
- **Debug ??** - Wasteful way of finding how much waste already produced
- **Review ?** - Helping preventing waste (doing the right things better)
- **Inspection ?** - Stopping generation and proliferation of waste
- **Prevention !!!** - Not producing waste



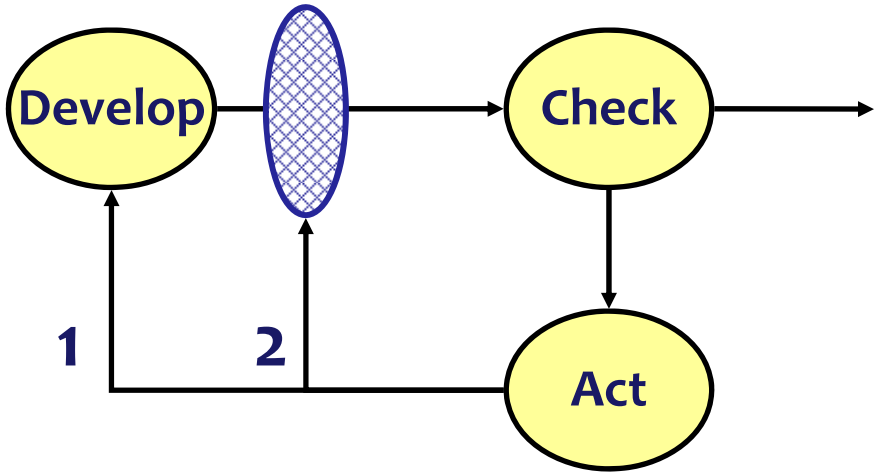
Inevitable consequence



Testing is checking correctness



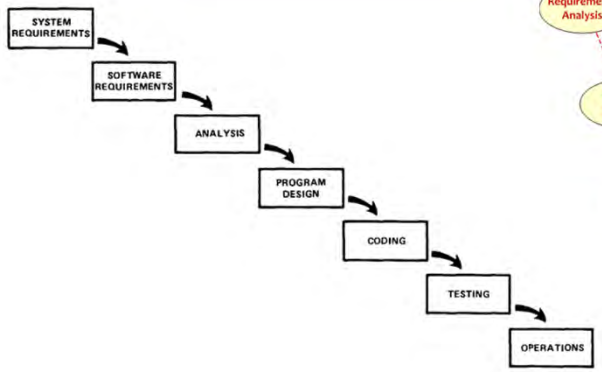
What we often see

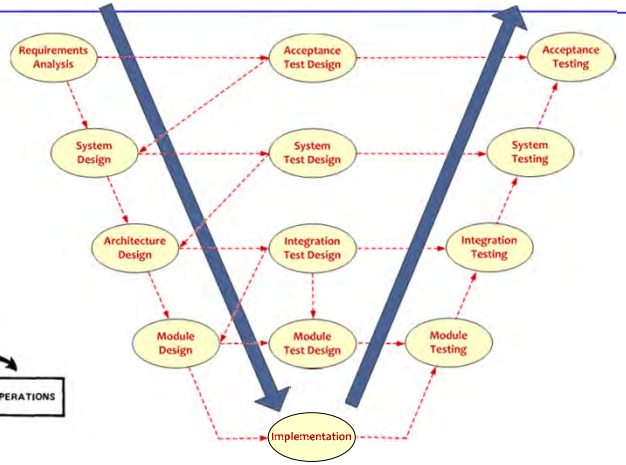


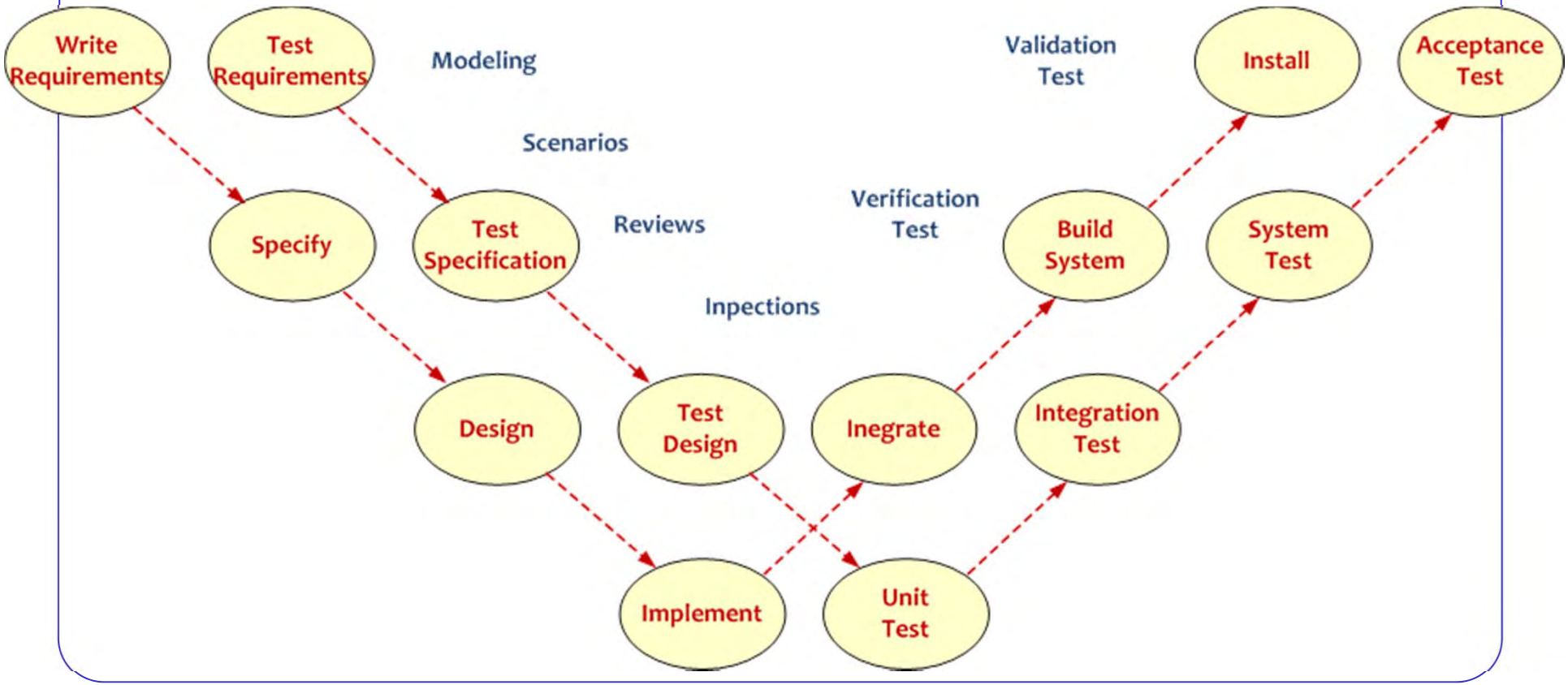
What we should expect

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

W-model


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Module Test Design
Module Testing
Implementation


Write Requirements
Test Requirements
Modeling
Validation Test
Install
Acceptance Test
Scenarios
Specify
Test Specification
Reviews
Verification Test
Build System
System Test
Inpections
Design
Test Design
Inegrate
Integration Test
Implement
Unit Test

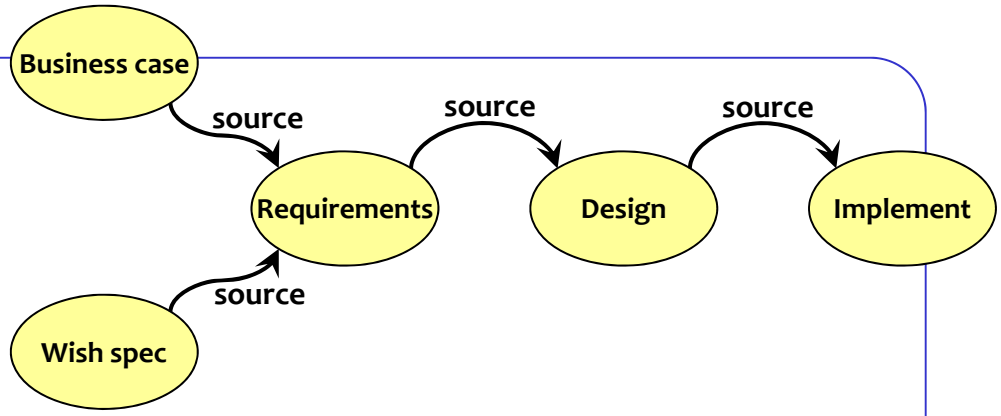
Can you find these by testing ?

- **Fuzzy requirements**
- **Functions that won't be used** (superfluous requirements)
  - Why to repair defects in the implementation of these requirements ?
  - The only defect is that it has been implemented
- **Nice things** (not checked for real need, not paid for)
  Shouldn't be there in the first place
- **Missing quality levels** (should have been in requirements)
  Checking the implementation of the requirements won't help
- **Missing constraints** (should have been in requirements)
  Product could be illegal (if that's the purpose, you'd better tell)

Why should we review ?

- All human activities are error-prone
- Bugs are “injected” at every stage of software development
- Relying on “testing” to find and remove them
  - Multiplies their cost up to 20-fold
  - Generally finds only 50%

It's not just code



- **Wish specification** — Thank you, nice input, to be taken seriously
- **Contract** — This is what I'll take you to court with
- **Business Case** — Why are we doing it
- **Requirements** — What the project agrees to satisfy
- **Design/ DesignLog** — Selecting the 'optimum' compromise and how we arrived at this decision
- **Specification** — This is how we are going to implement it
- **Implementation** — Code, schematics, plans, procedures, hardware, documentation, training

We're Agile !

- We don't need all these documents !
- We deliver working software !
- The next sprint we deliver more working software !
- We are efficient !
- We don't do all those bad things !
- We are superhuman !

- Are we really ?
- Do you really know what your customer needs ?
- Is trial and error the best way ?

Reviews & Inspections

Are all your documents always reviewed ?

- Do you have documents at all ?
- And ?

- If code is tested, how do you know it's correct ?
- If you would know a more economical way than (much of the) testing, what would you do ?
- Without proper education reviews are not very effective
- Inspections are a special way of review

What's the point ?

- Are your requirements clear ?
- What's the point in designing and implementing based on unclear requirements ?
- Working on a great solution for the wrong problem ?
- First develop the problem, then the requirements, then the design, only then the implementation
- What's your experience ?

Buying a second hand car

Checked at the bridge

What you think

What they mean

Can you review these requirements ?

- The system should be extremely user-friendly
- The system must work exactly as the predecessor
- The system must be better than before

- Do you know other examples ?

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

- It shall be reasonably easy to recover the system from failures, e.g. without taking down the power

Let's use some Rules

ref Tom Gilb

- **Unambiguous**
  Every word and phrase should be unambiguous to all potential intended readers
- **Clear to test**
  Every word and phrase should be clear enough to allow objective test
- **Quantified quality**
  All qualities (good things we want to improve) shall be expressed quantitatively
- **No design in requirements**
  Objectives shall not be expressed in terms of solutions

How many issues can you find ?

Unambiguous, Clear to Test, Quantified, No Design

- The system should be extremely user-friendly
- The system must work exactly as the predecessor
- The system must be better than before

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

- It shall be reasonably easy to recover the system from failures, e.g. without taking down the power

Unambiguous, Clear to Test, Quantified, No Design

(ref TG)

Sorry, removed for confidentiality

Can we develop based on Management Poetry ?

- Nice input, to be taken seriously
- We write back the requirements, don't we ?
- This is what we plan to do, if you let us continue

- Are we better at requirements ?
  - Unambiguous, Clear to Test, Quantified, No Design

Kennedy - May 1961

… before this decade is out, of landing a man on the moon and returning him safely to the earth

How many times F, f ?

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

(Deming)

Many types of Review to choose from

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

Techniques

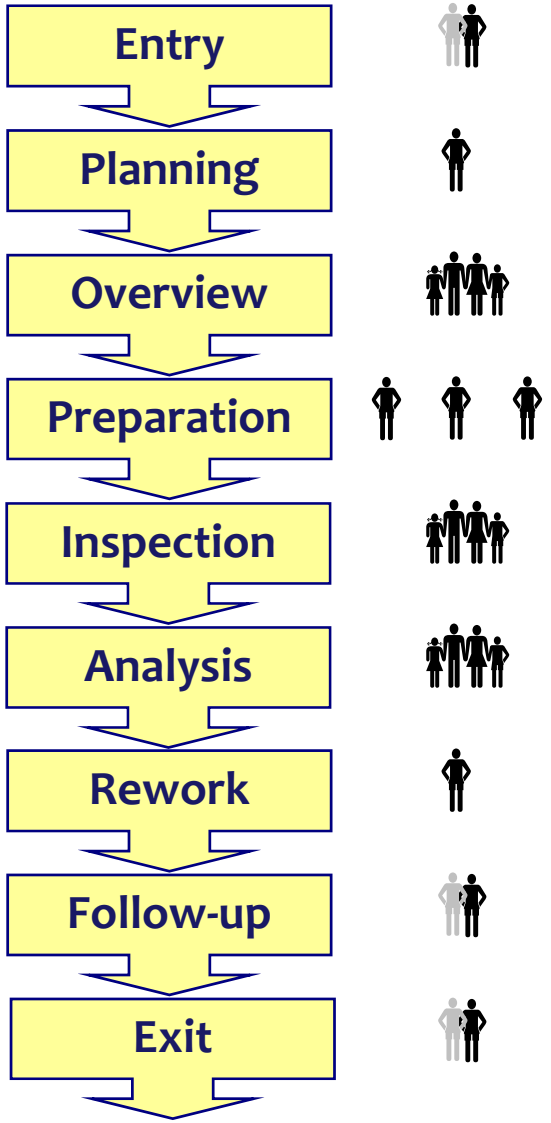
- Can you look at this ?
- Over the shoulder
- Pair Programming
- E-mail
- Tool
- On Screen
- Projector
- On Paper
- Formal process

Formal Reviews (vs Ad-Hoc)

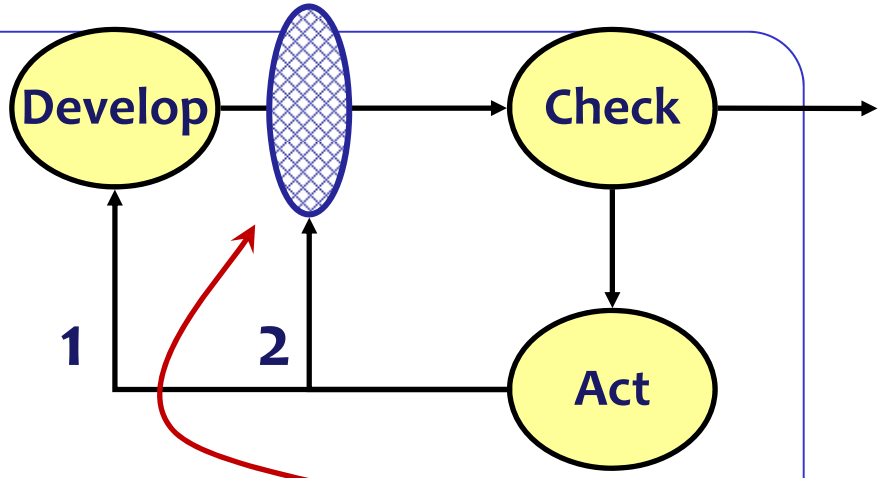
- **Defined, repeatable process**
- **Measures effectiveness**
- **Continuous improvement**
- **Rules/checklists**
- **Feeds prevention process**

Fagan Inspections

Inspection Process Steps

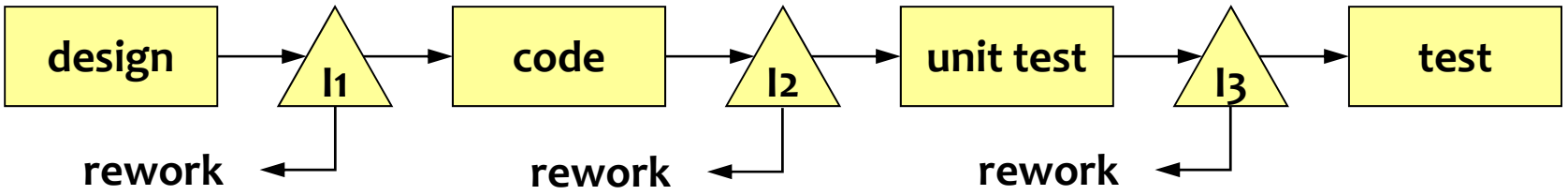


Purposes of Inspection



- Producing defect-free products with high productivity
- Reduce total defect rework
- Reduce the schedule impact of defects
- Find defects immediately after injection
- Provide the author with the quickest feedback on defects, how to recognize and avoid them in the future
- Without immediate feedback and learning, we will keep making the same mistakes

Fagan experiment



Productivity change by Inspections:

- No Inspection: 100% (baseline)
- I1 only: 112% (9/10 people can do the same)
- I1 and I2: 123% (8/10 people can do the same)
- 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

Fagan Defect-Free Process

- **Formal Process Definition**

  ensuring each member of the team is conversant in the objectives, function and entry and exit criteria of each process phase

- **Inspection Process**

  the seven-step process used to find defects

- **Continuous Process Improvement**

  removing systemic defects from the development process

Perseverance and results

- **I did not receive much support**

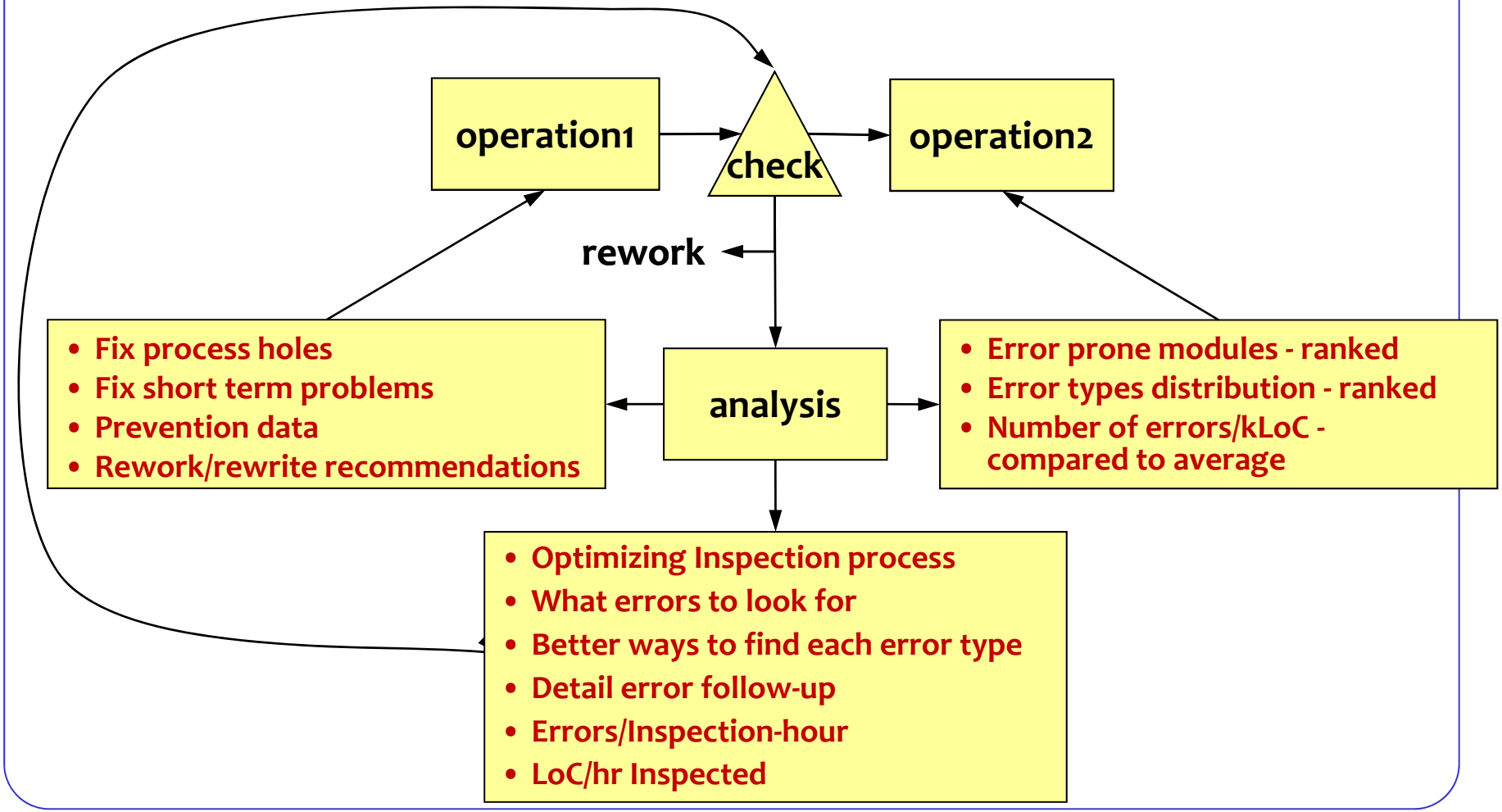
  – in fact, I was ridiculed, counselled and otherwise told to stop the nonsense and get on with the job of managing projects the way everyone else was doing it

- **Applied and executed as intended**

  it produces significant improvements to the software development process, including
  - **schedule and cost reduction**
  - **productivity improvements**
  - **fewer customer-reported defects**

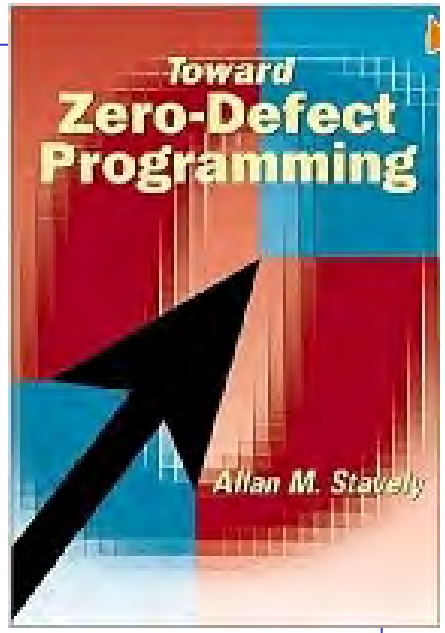
Prevention and knowledge building

(ref Fagan)



Cleanroom Inspections

Cleanroom Software Development



- Design (Mathematical proof)
- Verification (review of design by others)
- Implementation
- Verification (review of code 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

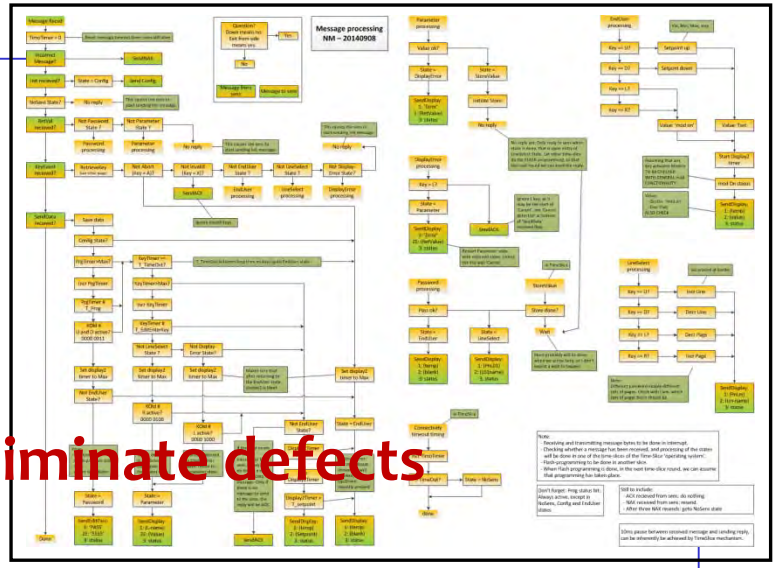
Cleanroom fundamentals

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

Cleanroom Principles

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

Cleanroom Inspections



- The purpose of Inspection is to eliminate defects
- Exit criterion for design:
  - One design statement materializes as 3 to 10 code statements
- Checklists of typical errors we make
  - Listed in order of frequency
- No Unit Test - Developer does not 'try' software !

- Testing:
  - Finding as many of the remaining defects as possible
  - Too many errors discovered
    - → previous steps are not being done properly
    - → redo previous steps (do not "repair")

Cleanroom: 'Slowest reviewer sets the pace'

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

It is important to resist taking shortcuts here

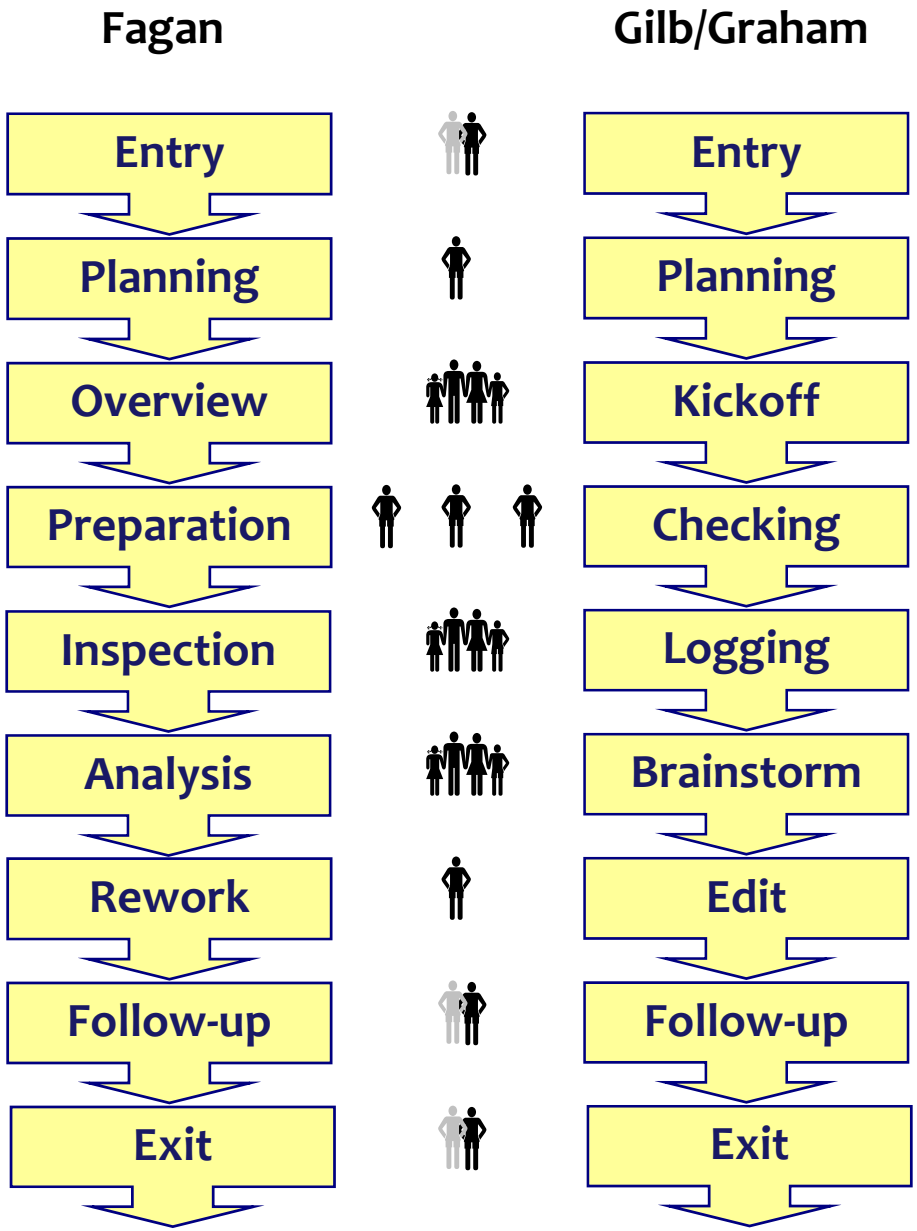
Cleanroom benefits

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

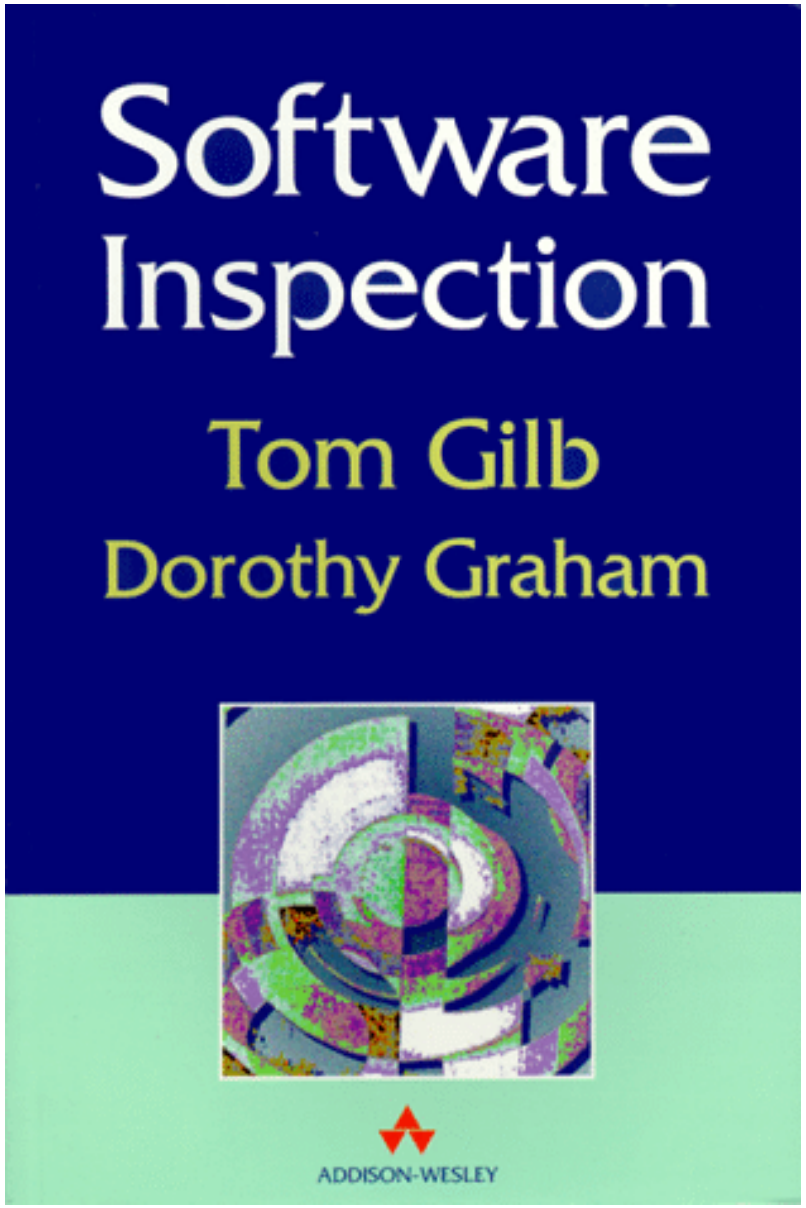
Quality costs less

Gilb / Graham Inspections

Inspection Process Steps



A ready to use recipe …



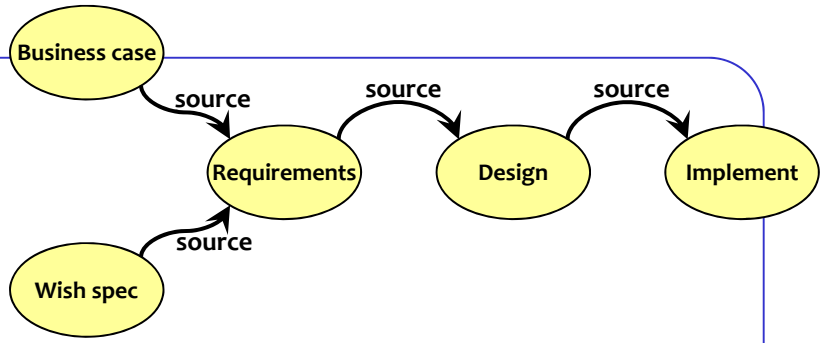
Do you recognize this ?

- 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

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

Rules



- **Rules are the law for documents**
- **Defect = Rule violation**
  not "I think this is wrong"

- **Rules:**
  - All quality requirements must be expressed quantitatively
  - The document should be consistent with itself and with source documents

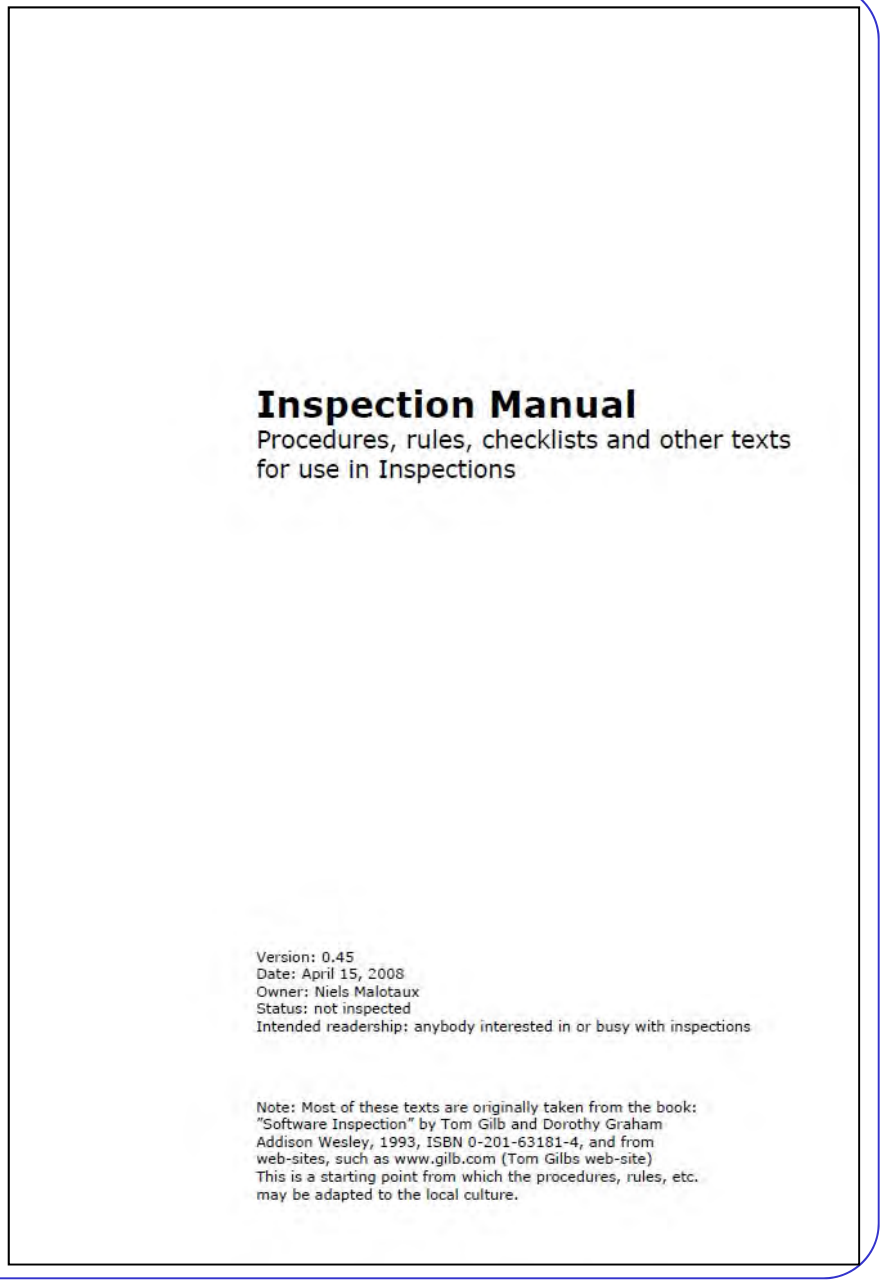
- **Typical requirements found:**
  The system should be extremely user-friendly

  The system must work exactly as the predecessor

  The system must be better than before

16 page
Inspection Manual

www.malotaux.nl/doc.php?id=61



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

Version: 0.45
Date: April 15, 2008
Owner: Niels Malotaux
Status: not inspected
Intended readership: anybody interested in or busy with inspections

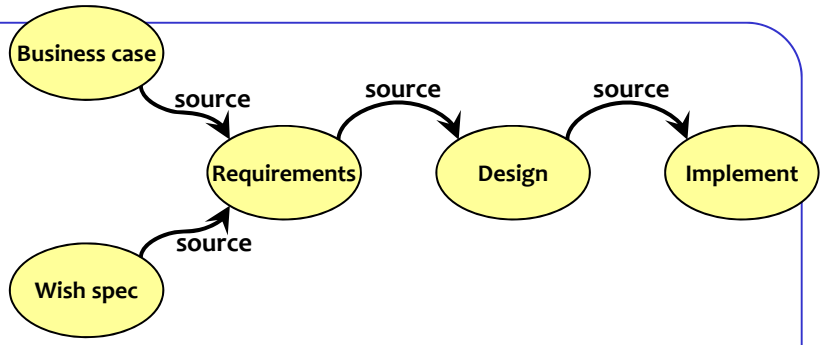
Note: Most of these texts are originally taken from the book:
"Software Inspection" by Tom Gilb and Dorothy Graham
Addison Wesley, 1993, ISBN 0-201-63181-4, and from
web-sites, such as www.gilb.com (Tom Gilbs web-site)
This is a starting point from which the procedures, rules, etc.
may be adapted to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

Generic Specification Rules

(see Inspection Manual)

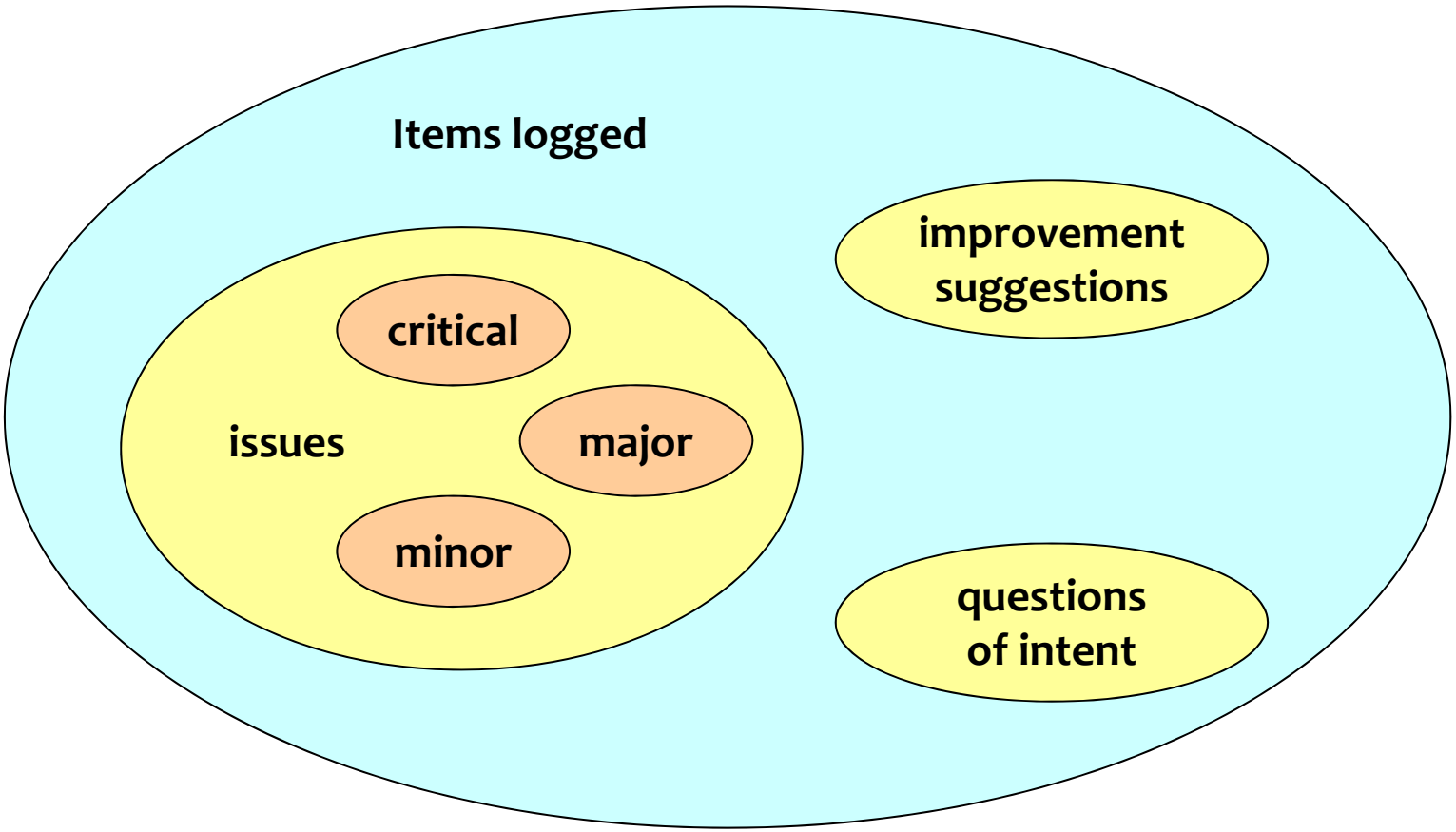
GE0	(def)	Generic engineering specification rules apply to all engineering documents as required best practices
GE1	(relevant)	All statements should be relevant to the subject
GE2	(complete)	There should not be any significant omissions
GE3	(consistent)	Statements should be consistent with other statements 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

Typical documents



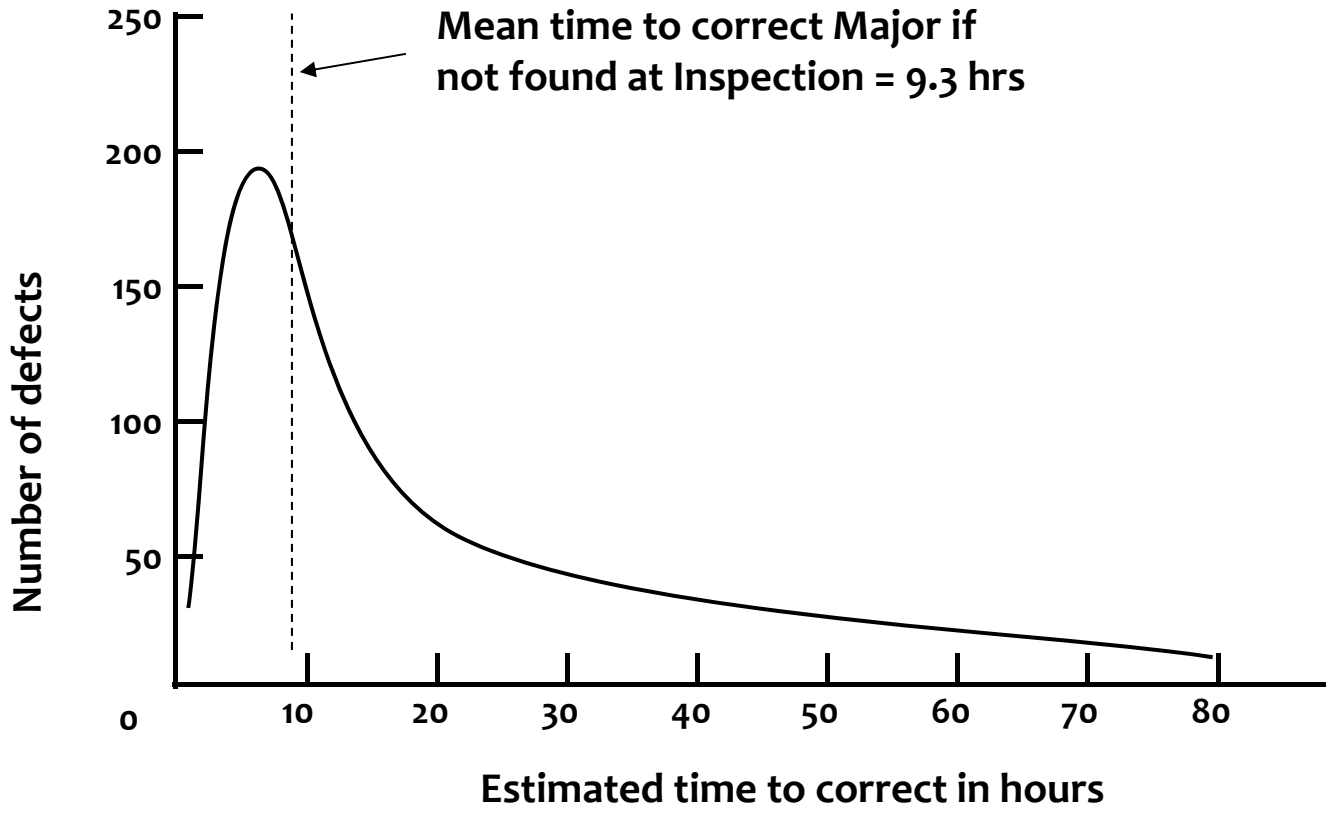
- **Wish specification** — Thank you, nice input
- **Contract** — This is what I'll take you to court with
- **Business Case** — Why are we doing it
- **Requirements** — What the project agrees to satisfy
- **Design/ DesignLog** — Selecting the 'optimum' compromise and how we arrived at this decision
- **Specification** — This is how we are going to implement it
- **Implementation** — Code, schematics, plans, procedures, hardware, documentation, training

What are we looking for?



Cost of Repair

ref SI, fig 14.6, p315



Optimum Checking Rate

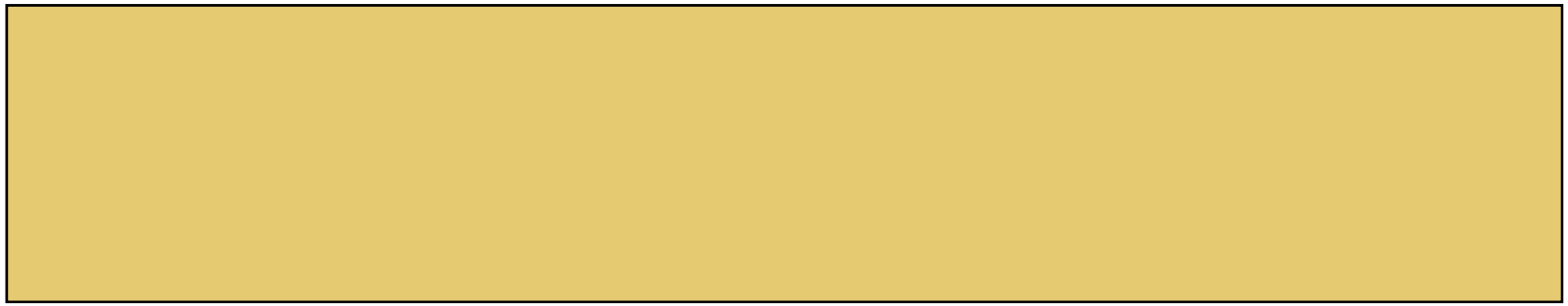
- 100~250 SLoC per hour
- 1 page of 300 words per hour ("logical page")

- How much time to spend per page
  - How much time do you spend per document ?
  - What is the size of a typical document ?

- Optimum Checking Rate:
  The most effective individual speed for 'checking a document against all related documents' in page/hr
- Not 'reading' speed, but rather correlation speed
- Failure to use it, gives 'bad estimate' for 'Remaining defects'
- More than we can afford. So … ? We must sample !

Optimum checking rate

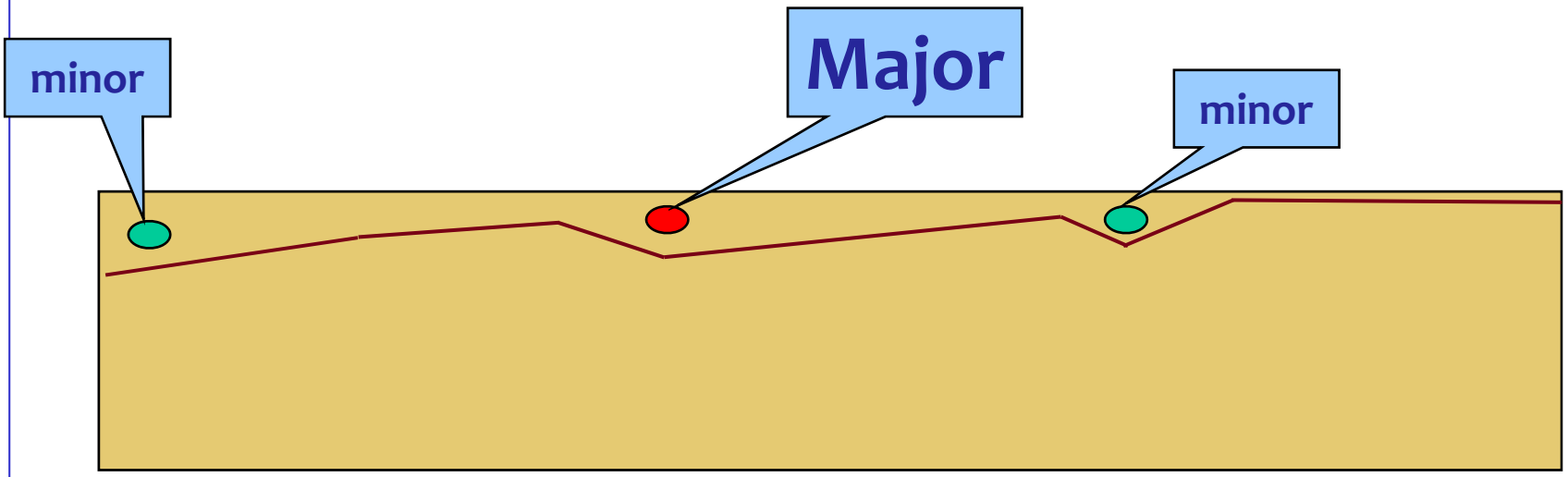
Ref. Dorothy Graham



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

Ref. Dorothy Graham

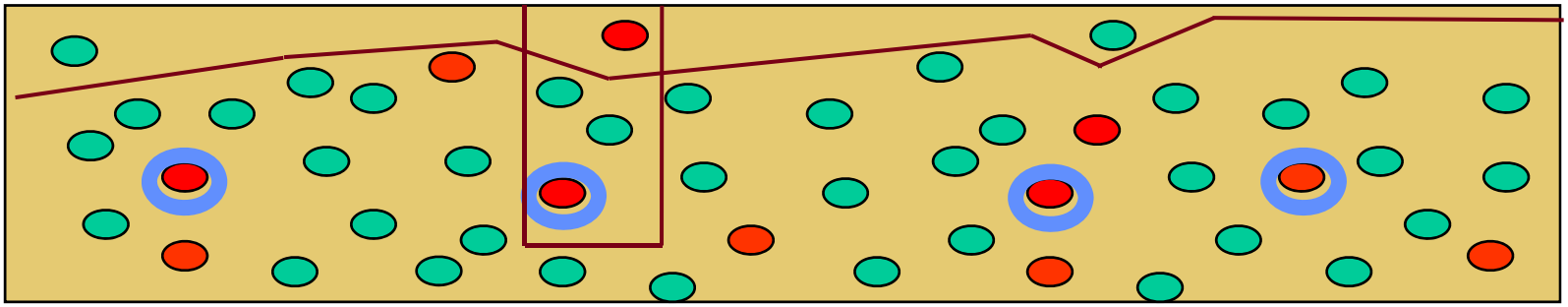
Review "Thoroughness"?



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

Inspection Thoroughness

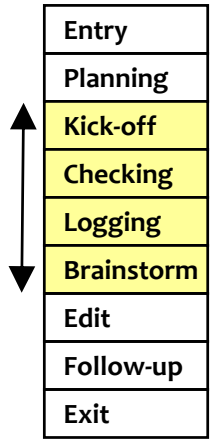
Ref. Dorothy Graham



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

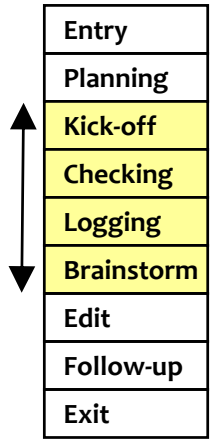
- **In the above case we are clearly taking a sample**
- **In the "shallow" case we were also taking a sample, however, we didn't realize it !**

6 hour initial Inspection process



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

4 hour mature Inspection process



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

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.

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

Early Inspections

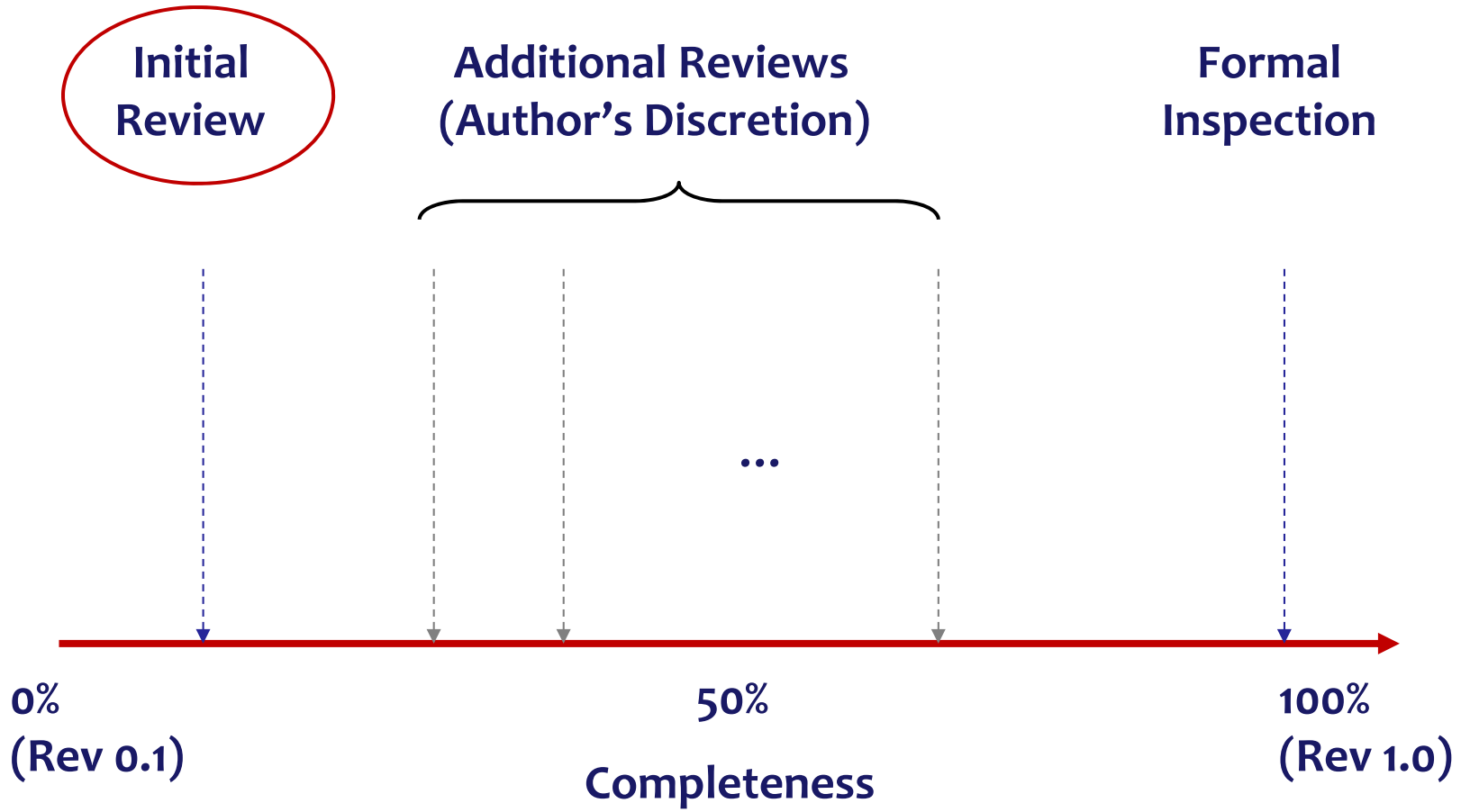
Extreme Inspections

Agile Inspections

Lean QA

Early Inspection

Prevention costs less than Repair



Case: Early Inspection on Requirements

Large e-business application with 8 requirements authors

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

ES

Results

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

Time investment: 26 hr

- 12 hours in initial review (1.5 hrs per author)
- About 8 hours in additional reviews
- 6 hours in final inspection (2 hrs, 2 checkers, plus prep and debrief)

Major defects prevented: 5 per requirement in ~750 total

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

Early Inspection

Prevention costs less than Repair

ES

Case: Test Cases

A tester's improvement writing successive test plans

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

Results

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

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

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

Early Detection vs. Prevention

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

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

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

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

Nice job! Early detection has big benefits - BUT…

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

Cost-effective defect prevention is the bottom line

Inspections

Used in Various Ways

Case: Can you teach Inspections ?

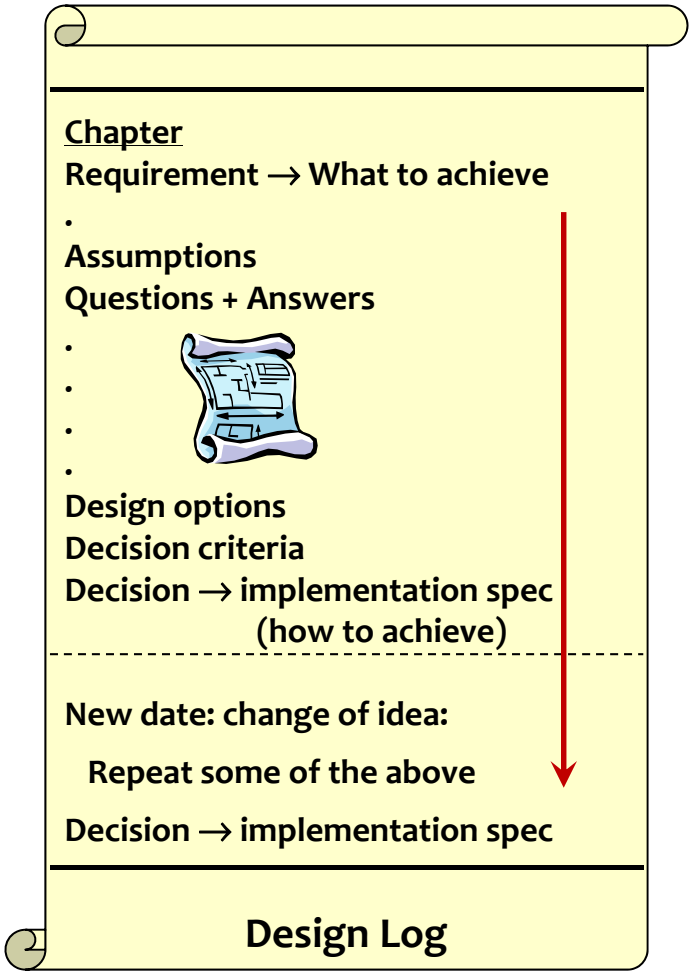
- Short intro
- Are you regularly reviewing ?
- Let's do it: baseline
  - Take a document
  - Reproduce one page
  - Do review
  - No issues
- One rule ('source')
  - Many issues

Sorry, picture removed for confidentiality

Datalog function improvement

DesignLog

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



Results

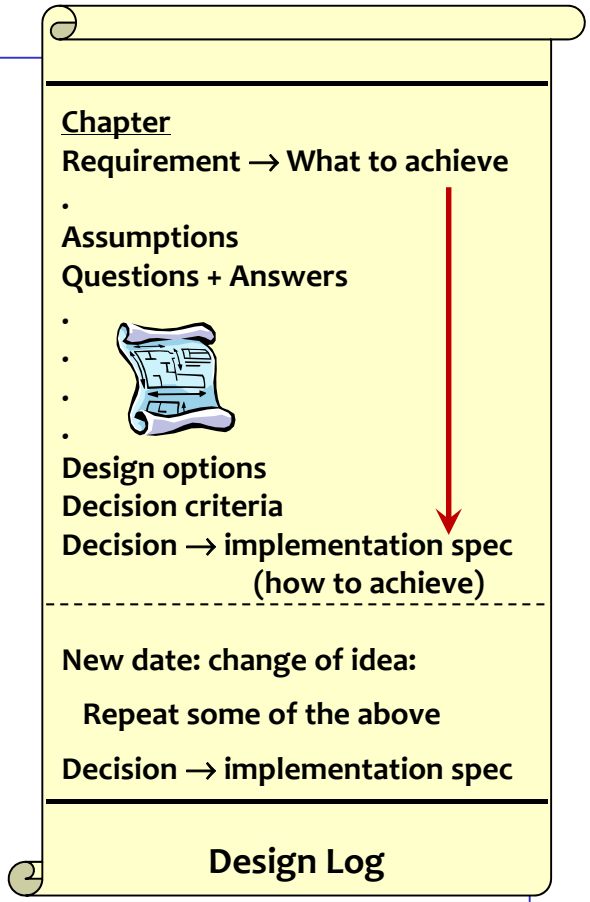
- No code until DesignLog reviewed
- You're delaying my project !
- Example
- Solution
- Thanks, you saved my project
- Now we can review to check the design before implementation
- Did I do the same ?
- Telling people to change: resistance
- How to let people change themselves …

Use a DesignLog

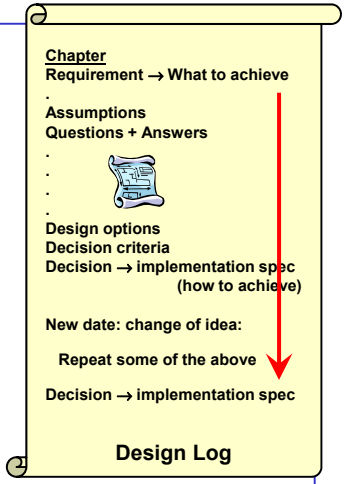
- Design
- Review
- Code
- Review

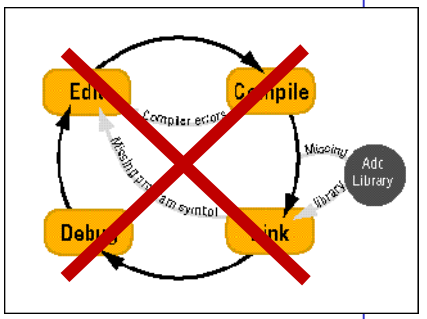
Iterate as needed

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



In the pub





James:

Niels, this is Susan

Susan, this is Niels, who taught me about DesignLogging

Tell what happened

Susan:

- *We had only 1.5 week to finish some software*
- *We were working hard, coding, testing, coding, testing*
- *James said we should stop coding and go back to the design*
- *"We don't have time!" - "We've only 7 days!"*
- *James insisted*
- *We designed, found the problem, corrected it, cleaned up the mess*
- *Done in less than 7 days*
- *Thank you!*

Case: City of Amsterdam

- Can you teach Inspections ?
- Using a tender document that was already 3 weeks late
  (please can you come tomorrow ?)
- You'll ditch the document after the course !
- Ha ha
- Of course they did
- The project was ditched a few weeks later
- Why ?
- Saved a lot of tax-payers money

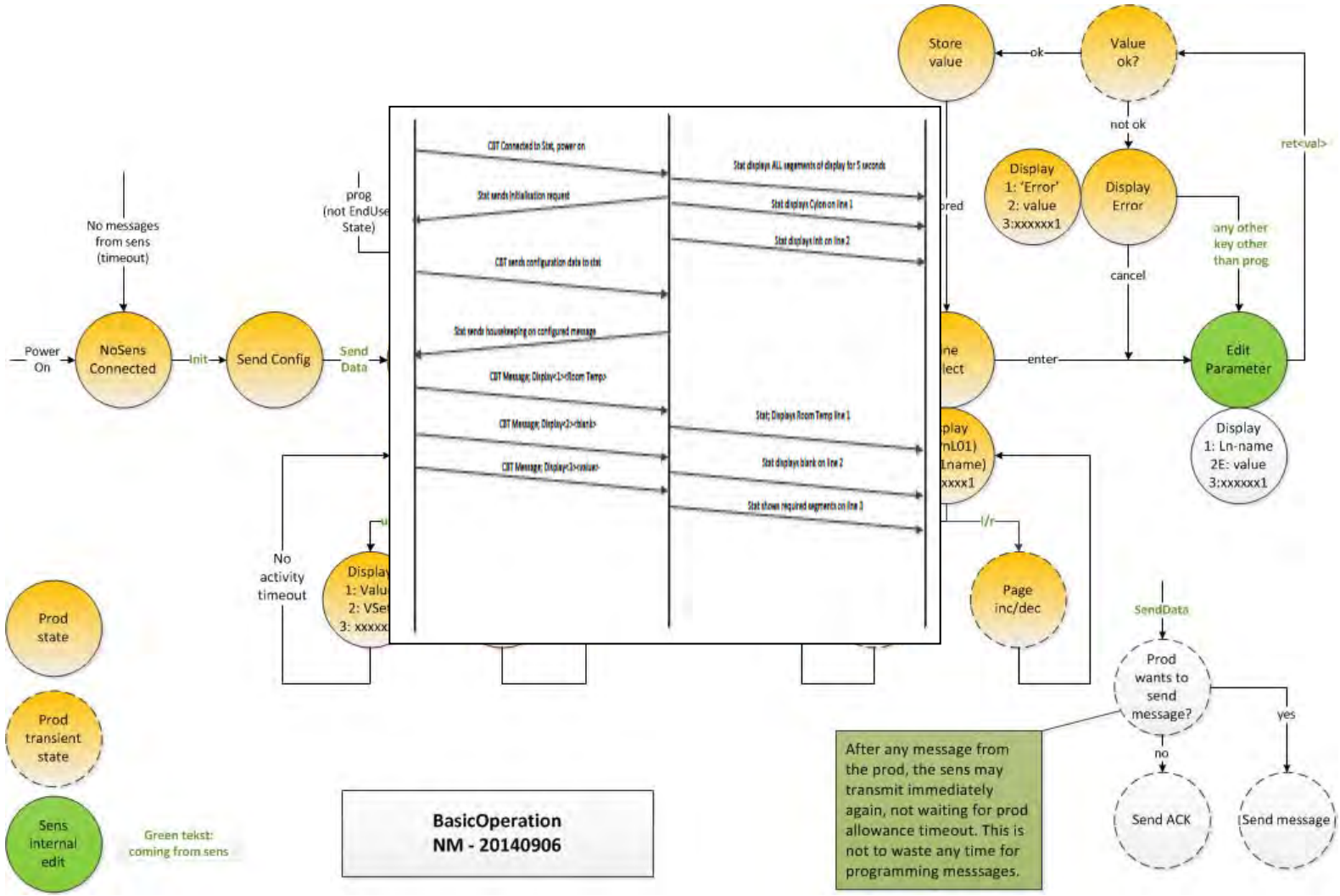
Make Documents Reviewable

If not, they're probably not very useful

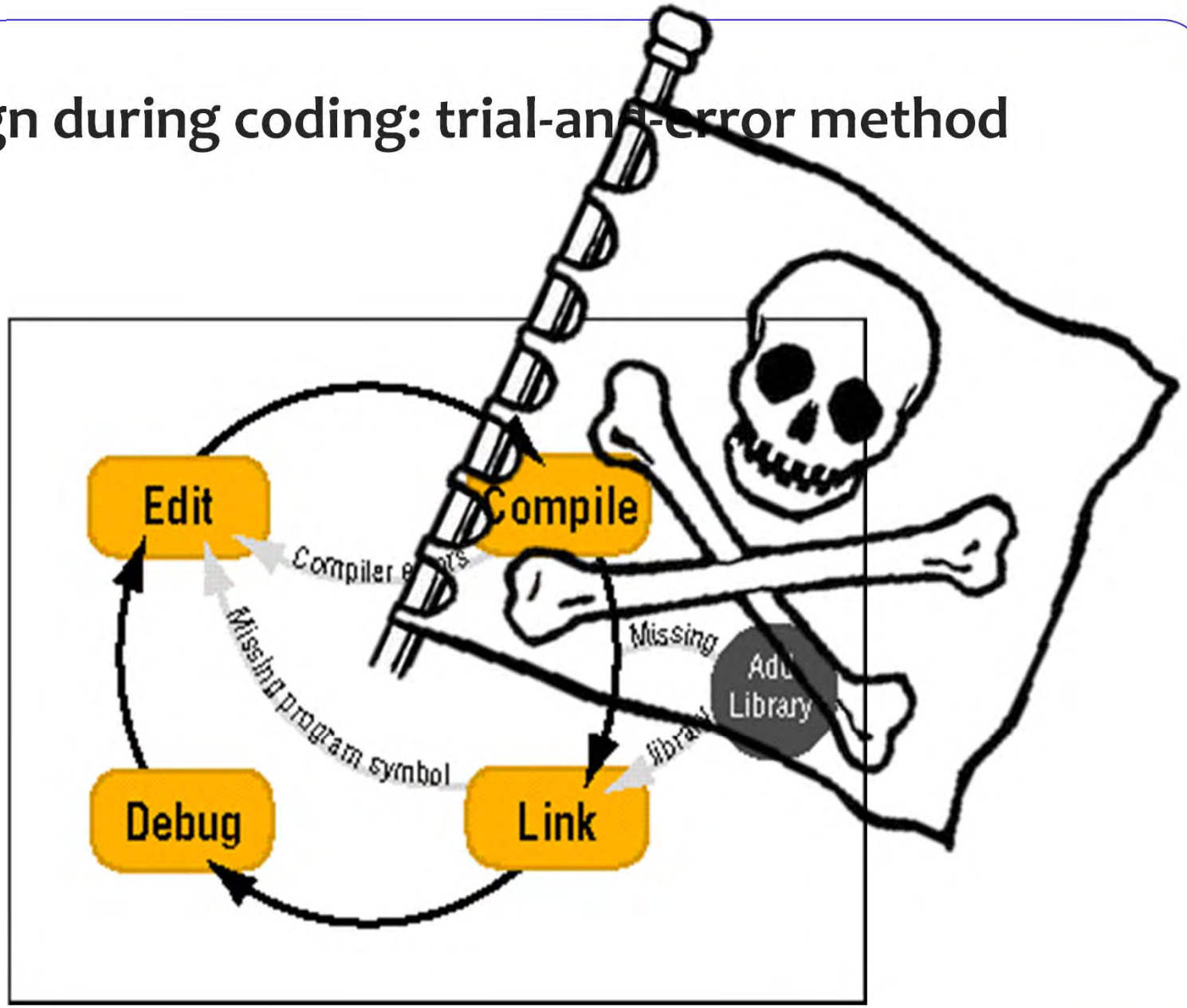
Unambiguous, Clear to Test, …

Design example

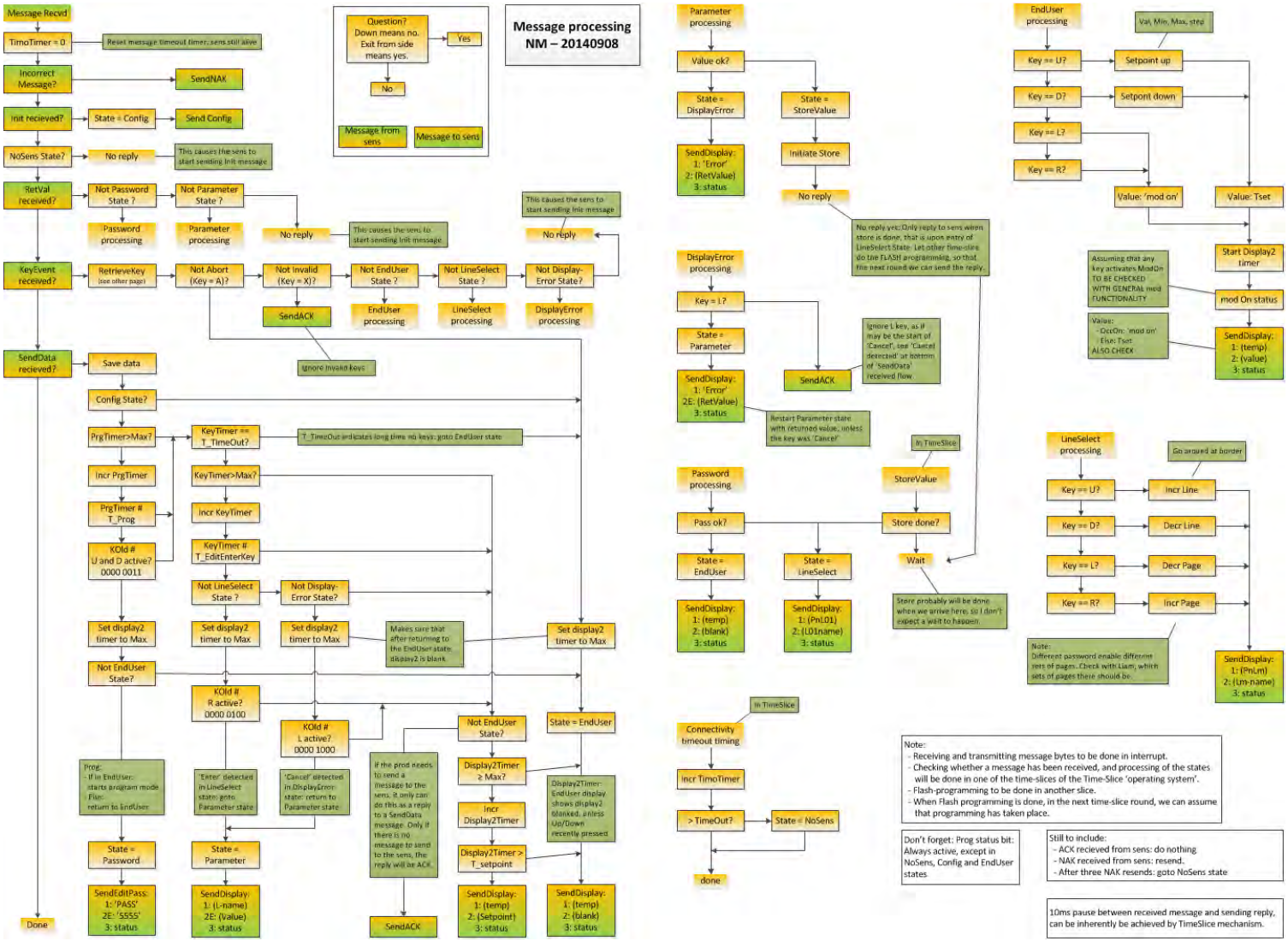
47 pages documentation condensed into one page



Design during coding: trial-and-error method



Design example



Improving the effectiveness of Reviews and Inspections

www.malotaux.nl/conferences
www.malotaux.nl/booklets
www.malotaux.nl/inspections

Niels Malotaux

N R Malotaux Consultancy

+31-655 753 604 niels@malotaux.nl www.malotaux.nl

Prevention costs a lot less

- **What will lead to *prevention* ?**
- **The error that does not exist cannot be missed** (Crosby)

How do we get prevention ?

- **By learning**
- **Learning about our tendencies of doing things wrong and not any more doing it wrong**

How do we learn

- **By quick confrontation**
- **Not by testing at the end**

There is much waste we can save

- **A defect is**
  the cause of a problem experienced by the users
  (hassle to a stakeholder)
- **All we have to do is delivering results without defects**
  - **Some 50% of project time is consumed by all kinds of testing and repairing**
  - **About 50% of developed software is never used**
  - **Over 50% of delivered software is never used**
- **What's your experience ?**
- **Is being late a problem ?**

Let's do more testing ! … ?

Dijkstra (1972):

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

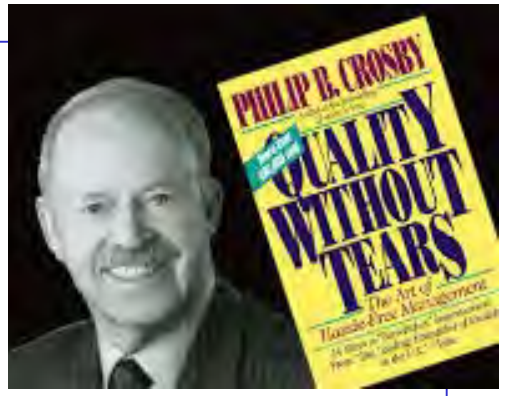
however:

Program testing can be a very effective way to show the presence of bugs but it is hopelessly inadequate for showing their absence

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

Absolutes of Quality

Crosby (1926-2001)



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

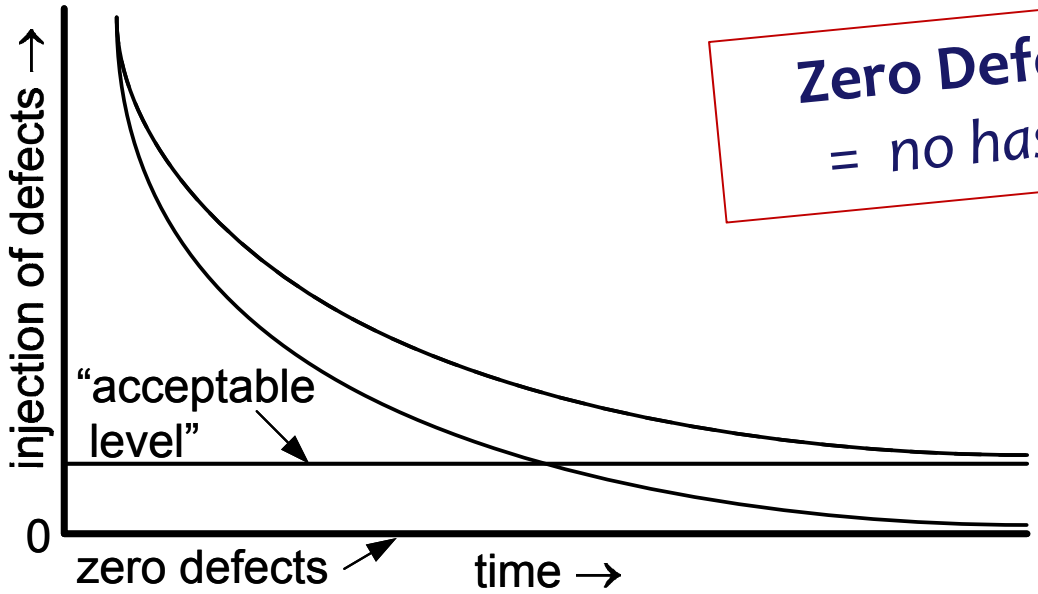
Philip Crosby, 1970

- The purpose is customer success
  (not customer satisfaction)

Added by Philip Crosby Associates, 2004

What is Zero Defects

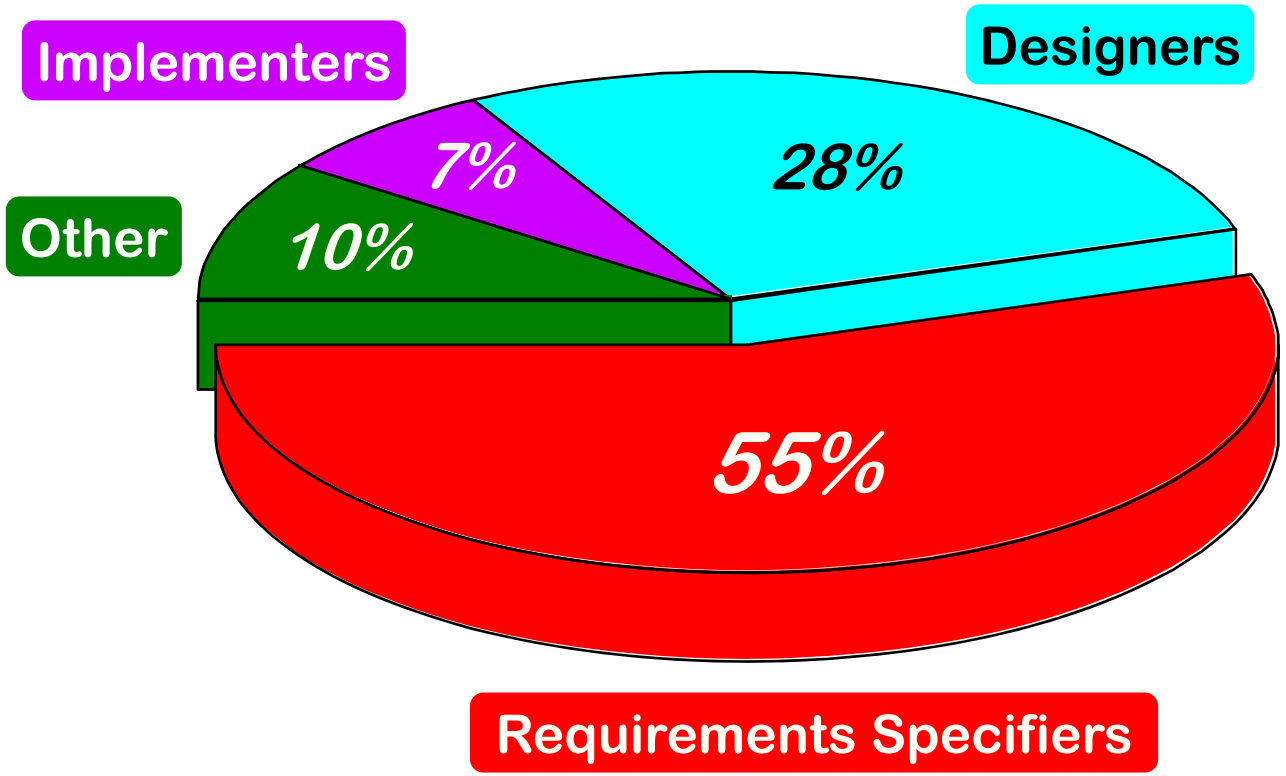
- **Zero Defects is an *asymptote***



Zero Defects
= no hassle

- **When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately**
- **AQL > Zero means that the organization has settled on a level of incompetence**
- **Causing a hassle other people have to live with**

Typical Defect Injectors (*cost* breakdown)



After Bender Associates, *1996*

Debugging ? ? ?

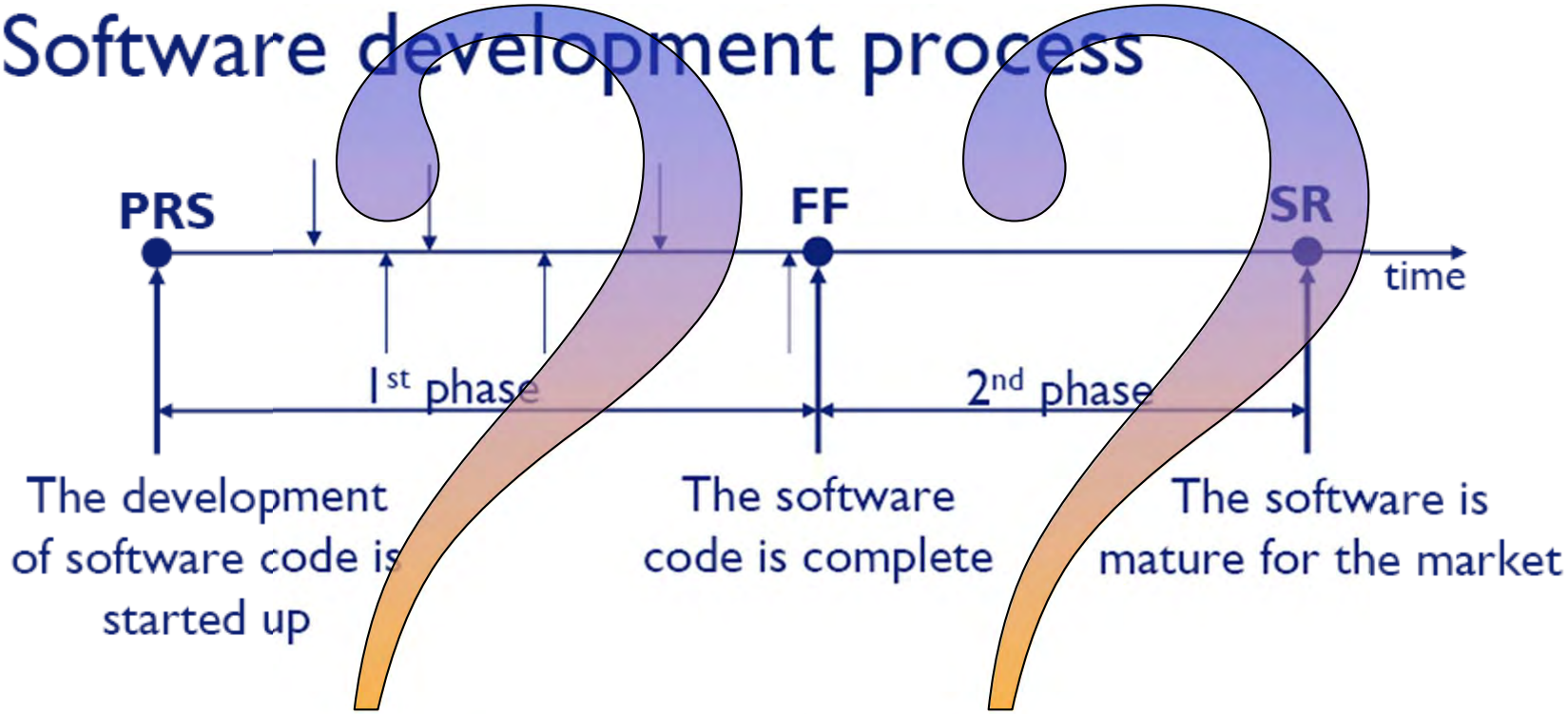
The process of defect injection

Conventional software development:

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

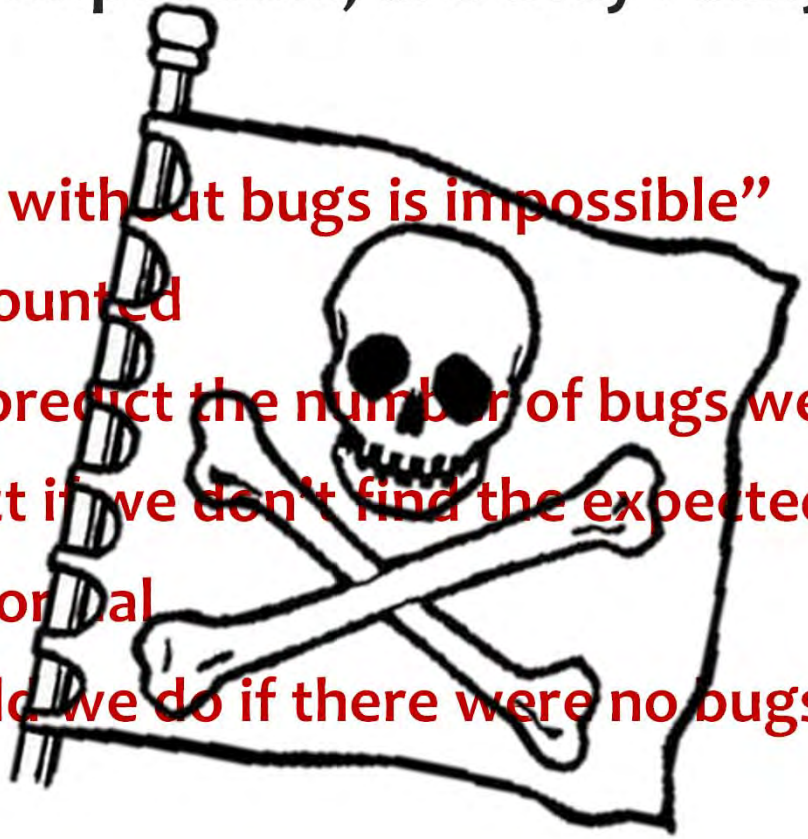
How about your environment ?

Software development process



- 1st phase is developing phase
- 2nd phase de-bugging phase

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 focusing on bugs, there will be bugs
- Testing is about finding *no bugs*

Defects found are symptoms of deeper lying problems

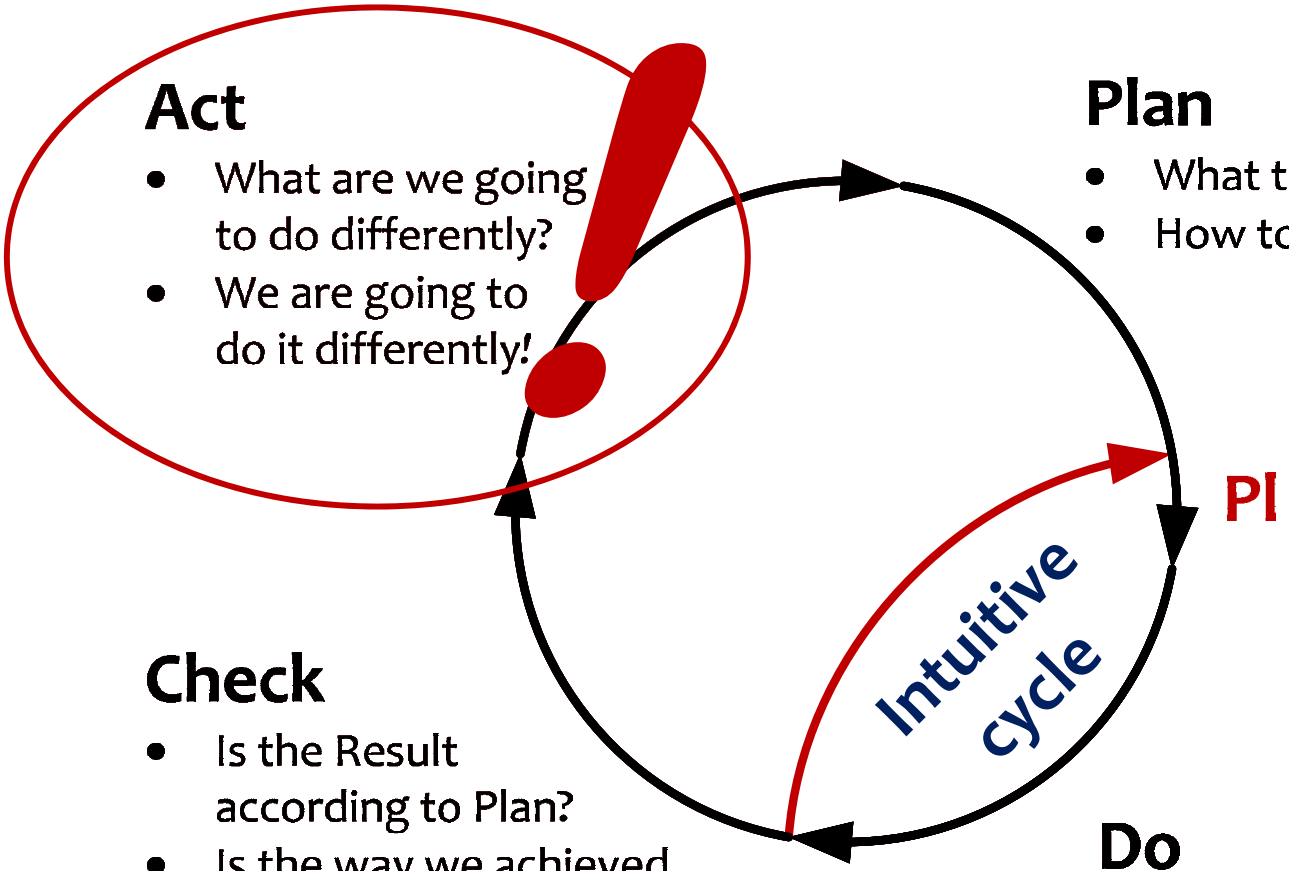
Repairing defects creates risks:

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

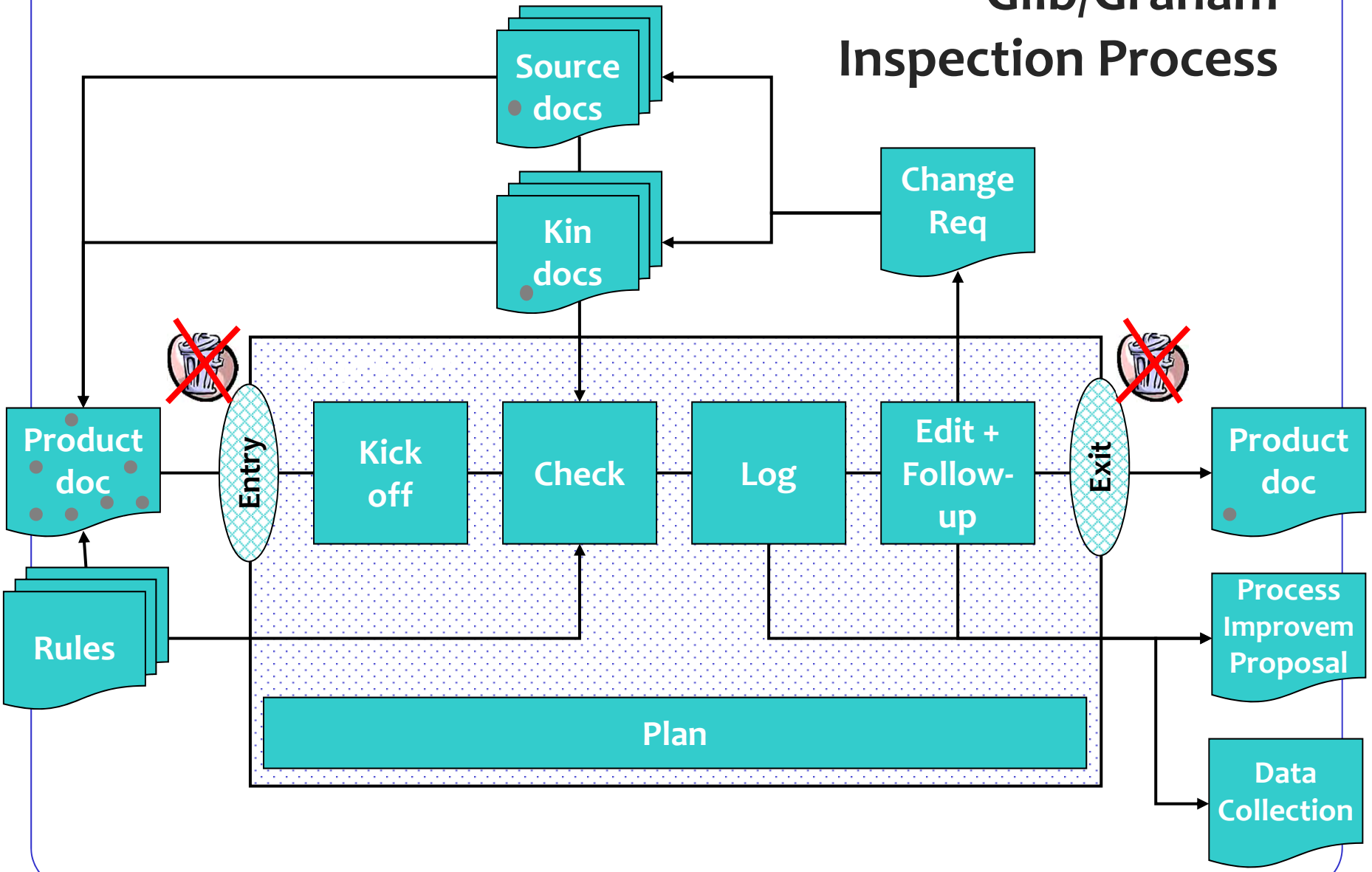
→ **Do Root Cause Analysis and make sure it never happens again**

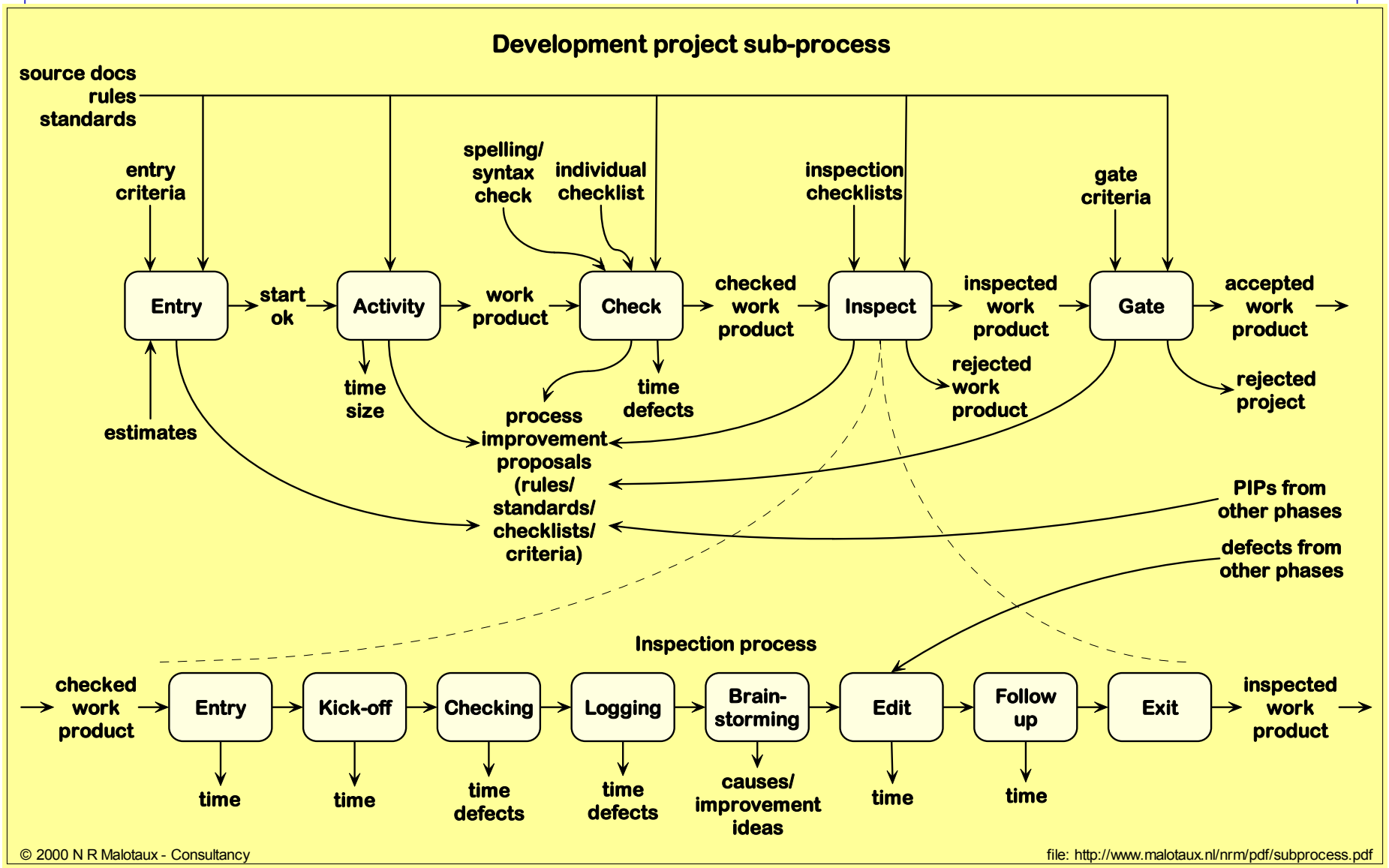
The essential ingredient: the PDCA Cycle

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



Gilb/Graham Inspection Process




Development project sub-process
source docs
rules
standards
entry
criteria
spelling/
syntax
check
individual
checklist
inspection
checklists
gate
criteria
Entry
start
ok
Activity
work
product
Check
checked
work
product
Inspect
inspected
work
product
Gate
accepted
work
product
time
size
time
defects
rejected
work
product
rejected
project
estimates
process
improvement
proposals
(rules/
standards/
checklists/
criteria)
PIPs from
other phases
defects from
other phases
Inspection process
checked
work
product
Entry
Kick-off
Checking
Logging
Brain-
storming
Edit
Follow
up
Exit
inspected
work
product
time
time
time
defects
time
defects
causes/
improvement
ideas
time
time
© 2000 N R Malotaux - Consultancy
file: http://www.malotaux.nl/nrm/pdf/subprocess.pdf

Testing vs Reviews & Inspections

- If you find an issue during Test, you still have to find the origin
- If you find an issue during Review or Inspection, you're on top of it

- If Testing means running the system
- And Review / Inspection means Verifying and/or Validation of a document

Root Cause Analysis

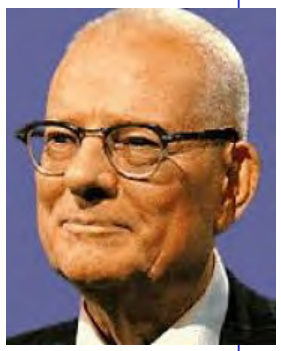
- **Is Root Cause Analysis routinely performed ?**
- **What is the Root Cause of a defect ?**

- **Cause:**
  The error that caused the defect

- **Root Cause:**
  What *caused us* to make the error that caused the defect

- **Without proper RCA, we're *doomed to repeat the same errors***

Who is the (main) customer of Testing and QA ?



Deming
(1900-1993)

- **Deming:**
  - Quality comes not from testing, but from *improvement of the development process*
  - Testing does *not* improve quality, nor guarantee quality
  - It's too late
  - The quality, good or bad, is already in the product
  - You cannot test quality into a product
- **Who is the main customer of Testing and QA ?**
- **What do we have to deliver to these customers ?**
  What are they *waiting for* ?
- **Testers and QA are *consultants* to development**
- **Testing and QA *shouldn't delay* the delivery - How ?**