

TECHNOLOGISCH INSTITUUT

**Antwerpen
21 September 2010**

Niels Malotaux

**Help !
We have a QA Problem !**

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

Niels Malotaux

Help! We have a QA Problem!

Niels Malotaux

Niels Malotaux is an independent Project Coach and expert in optimizing project performance. He has 35 years experience in designing electronic and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading his own systems design company. Since 1998 he devotes his expertise to helping projects to deliver Quality On Time: being predictable while 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 well over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, India, Ireland, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less in the practice of real projects. He is a frequent speaker at conferences and published several booklets around the topic of the presentation (see www.malotaux.nl/Booklets).

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
<i>Result Management</i>	

Help !

We have a QA Problem !

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68

niels@malotaux.nl

www.malotaux.nl

1

Niels Malotaux

Result Management

- **Project Coach**

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



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

1

Help! We have a QA Problem!

We have a QA problem !



- **Large stockpile of modules to test** (hardware, firmware, software)
- **You shall do Full Regression Tests**
- **Full Regression Tests take about 15 days each**
- **Too few testers** (“Should we hire more testers?”)
- **Senior Tester paralyzed**
- **Can we do something about this?**

3

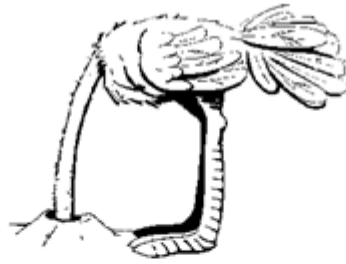
The essential ingredient: the PDCA Cycle

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



4

Help! We have a QA Problem!



Instead of complaining about a problem ...

(Stuck in the Check-phase)

Let's do something about it !

(Moving to the Act-phase)

5

Objectifying and quantifying the problem is a first step to the solution

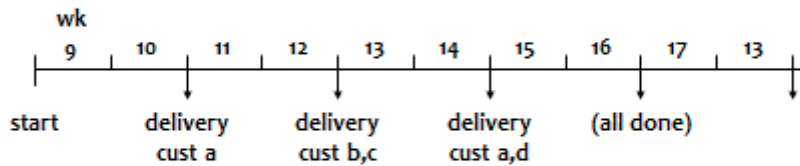


Line	Activity	Estim	Alter native	Junior tester	Devel opers	Customer	Will be done (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	1	1			Sev	
9	Package 8.2	1	1			?	
10	Package 8.3	1	1			Chrt	24 Feb
11	Package 8.4	1	1			Chrt	
12	Package 8.5	1.1	1.1			Yet	28 Feb
13	Package 8.6	3	3			Yet	24 Mar
14	Package 8.7	0.1	0.1			Cli	After 8.5 OK
15	Package 8.8	18	18			Ast	
	totals	106	47	32	36		

6

Help! We have a QA Problem!

TimeLine



Selecting the priority order of customers to be served

- “We’ll have a solution at that date ... Will you be ready for it ?”
Another customer could be more eagerly waiting
- Most promising customers

7

Result

- Tester empowered
- Done in 9 weeks
- So called “Full Regression Testing” was redesigned
- Customers systematically happy and amazed
- Kept up with development ever since
- Increased revenue

Recently:

- Tester promoted to product manager
- Still coaching successors how to plan

8

Help! We have a QA Problem!

Who is the customer of Testing and QA?

- **Deming:**
 - Quality comes not from testing, but from improvement of the development process. Testing does not improve quality, nor guarantee quality. It's too late. The quality, good or bad, is already in the product. You cannot test quality into a product.
- **Developers are the customer**
- **Testers help developers to become perfect**
- **Testing is a project to run alongside and synchronized to the development project**
- **Therefore, it must be organised like any other project**

9

Universal Project Goal

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
- and to be more successful than he was without it

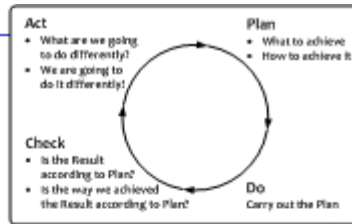
Constrained by (win - win)

- what the customer can afford
- what we mutually beneficially and satisfactorily can deliver
- in a reasonable period of time

10

Help! We have a QA Problem!

Evo

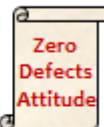


- **Evo (short for Evolutionary...)** uses PDCA consistently
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for **Product, Project and Process**, based on ROI and highest value
- Combining Planning, Requirements- and Risk-Management into **Result Management**
- We know we are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

11

- **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
- **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

Evolutionary Project Management (Evo)

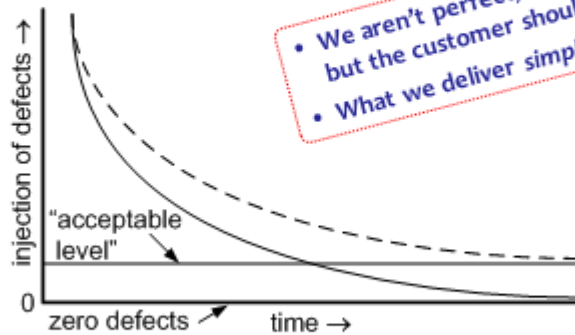


Evo Project Planning

12

Zero Defects ?

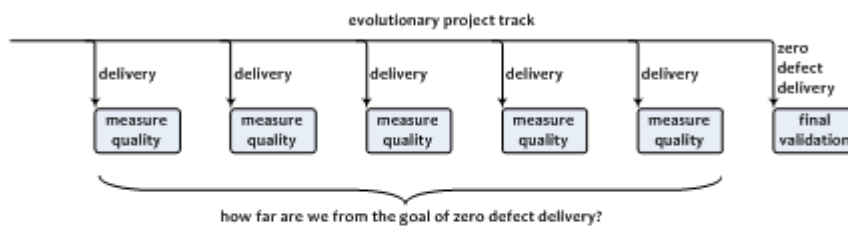
- Zero Defects is an asymptote



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

13

Conventional Evo Testing



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

14

Help! We have a QA Problem!

The aim of Testing

- **Being done as soon as the development is done**
- **Well, almost**
- **Excuses, excuses, excuses**
 - **The developers are always late**
(Evo developers live up to their promises)
 - **The developers don't take us seriously**
(Evo developers ask testers for help)
 - **The developers don't inject enough defects**
(now testing becomes a real challenge)
- **Helping development to be successful**



15

Developers are constantly optimizing

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

16

Help! We have a QA Problem!

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

17

Further Improvement

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

18

Help! We have a QA Problem!

Evo Planning: Weekly TaskCycle

- Optimizing the efficiency of what we do
- 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



Every week we plan

- 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 per week)
- What can, and are we going to do
- What are we *not* going to do

Taska	2	↑ do
Taskb	5	
Taskc	3	
Taskd	6	
Taske	1	
Taskf	4	
Taskg	5	26
Taskh	4	↓ do not
Taskj	3	
Taskk	1	

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

20

Help! We have a QA Problem!

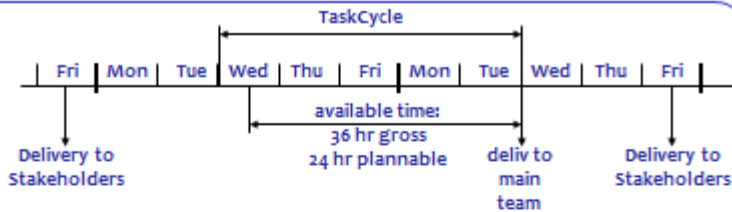
DeliveryCycle

- **Optimizing the efficiency of what we deliver**
- **Are we delivering the right things, in the right order to the right level of detail for now**
- **Optimizing requirements and checking assumptions**
 1. What will generate the optimum feedback
 2. We deliver only to eagerly waiting stakeholders
 3. 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**



21

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

Gregory (later)

Draft design	0
Finish design	0
...	
Repair deliv	0
...	

Zero 0
Defects
Attitude

Jerome

XMLa	4
XMLb	4
Total	42

XMLa	3
XMLb	3
...	

22

Help! We have a QA Problem!

Agile, but will we be on time ?

- Organizing the work in very short cycles
- Making sure we are doing the right things
- Doing the right things right
- Continuously optimizing (what not to do)
- So, we already work more efficiently

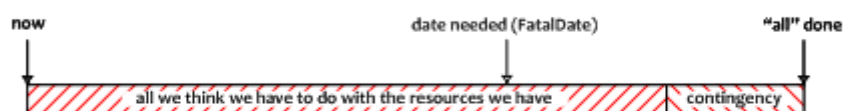
but ...

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

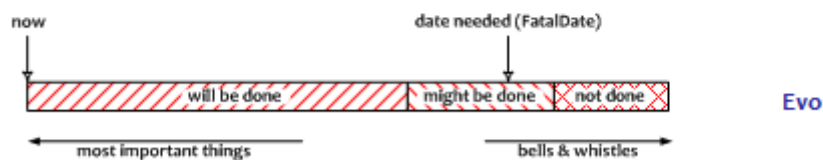
23

TimeLine

What the customer wants, he cannot afford



Standard Projects



Evo

24

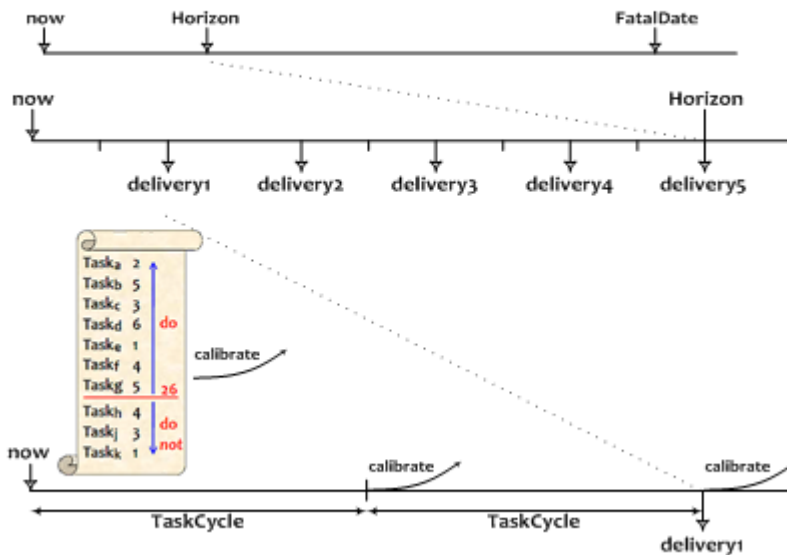
Help! We have a QA Problem!

If it easily fits ...



25

Result to Tasks and back



26

Niels Malotau

Help! We have a QA Problem!

Activity	Estimate	Real
Act1	Ae1	Ar1
Act2	Ae2	Ar2
Act3	Ae3	Ar3
Act4	Ae4	Ar4
Act5	Ae5	Ar5
Act6	Ae6	Ar6
Act7	Ae7	Ar7
Act8	Ae8	Ar8
Act9	Ae9	Ar9
Act10	Ae10	Ar10
Act11	Ae11	
Act12	Ae12	
Act13	Ae13	
Act14	Ae14	
Act15	Ae15	
Act16	Ae16	
Act17	Ae17	
Act18	Ae18	
Act19	Ae19	
Act20	Ae20	
Act21	Ae21	
...	...	
Act...	Ae...	

Calibration

Calibration Factor

$$\frac{\sum_{now - n}^{now - 1} Ar}{\sum_{now - 1}^{now - n} Ae}$$

Value Still To Earn

$$\text{Calibration Factor} * \sum_{now}^{then} Ae$$

ratio $\Sigma Ar / \Sigma Ae$ in the past

← now

predicted Value Still To Earn in the future

← then

← then2

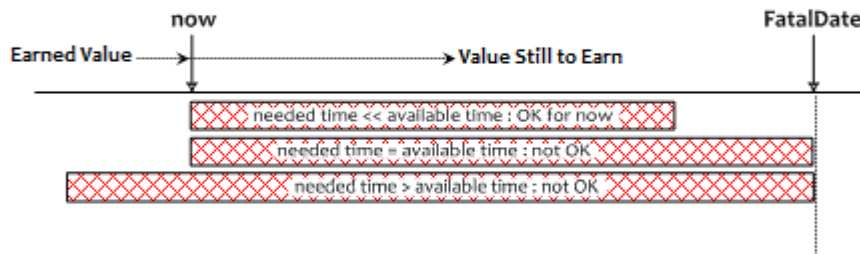
27

Predicting what will be done when

Line	Activity	Estim	Spent	Still to spend	Ratio real/es	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009
7	Activity 7	1				1.4	1.4	10 Apr 2009
8	Activity 8	3				1.4	4.2	16 Apr 2009
↓	↓							
16	Activity 16	4				1.4	5.6	2 Jun 2009
17	Activity 17	5				1.4	7.0	11 Jun 2009
18	Activity 18	7				1.4	9.8	25 Jun 2009

28

What do we do if we see we won't make it on time?



If it doesn't fit ... count backwards

When the match is over, you can't score a goal any more

29

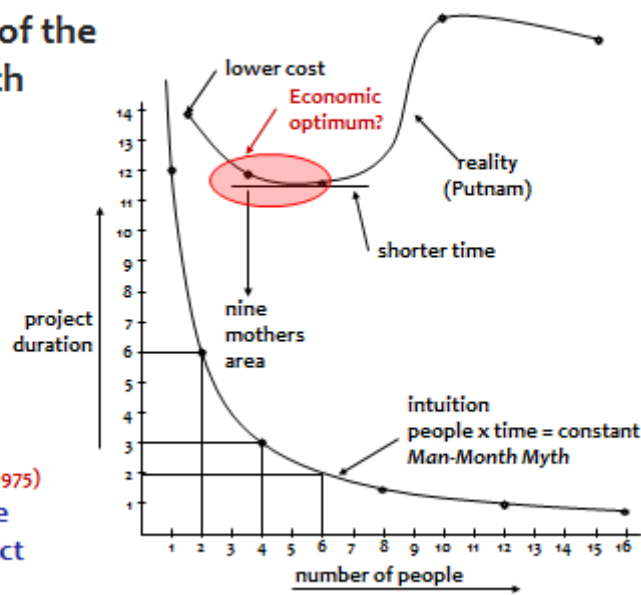
Deceptive options

- **Hoping for the best** (fatalistic)
- **Going for it** (macho)
- **Working Overtime** (fooling ourselves)
- **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

30

Help! We have a QA Problem!

The Myth of the Man-Month



31

Saving time



We don't have enough time, but we can save time without negatively affecting the Result !

- **Efficiency in what (why, for whom) we do** - doing the right things
 - Not doing what later proves to be superfluous
- **Efficiency in how we do it** - doing things differently
 - **The product**
Using proper and most efficient solution, instead of the solution we always used
 - **The project**
Doing the same in less time, instead of immediately doing it the way we always did
 - **Continuous improvement and prevention processes**
Constantly learning doing things better, overcoming bad tendencies
- **Efficiency in when we do it** - right time, right order
- **TimeBoxing** - much more efficient than FeatureBoxing

32

Help! We have a QA Problem!

- www.malotaux.nl/Booklets

- 1 **Evolutionary Project Management Methods (2001)**
Issues to solve, and first experience with the Evo Planning approach
- 2 **How Quality is Assured by Evolutionary Methods (2004)**
After a lot more experience: rather mature Evo Planning process
- 3 **Optimizing the Contribution of Testing to Project Success (2005)**
How Testing fits in
- 3a **Optimizing Quality Assurance for Better Results (2005)**
Same as Booklet 3, but for non-software projects
- 4 **Controlling Project Risk by Design (2006)**
How the Evo approach solves Risk by Design (by process)
- 5 **TimeLine: How to Get and Keep Control over Longer Periods of Time (2007)**
~Replaced by Booklet 7
- 6 **Human Behavior in Projects (2008)**
Human Behavioral aspects of Projects
- 7 **How to Achieve the Most Important Requirement (2008)**
Planning of longer periods of time, what to do if you don't have enough time
- 8 **Help! We have a QA Problem! (2009)**
The story of this presentation

- www.malotaux.nl/nrm/Insp
Inspection pages

More

33

Some extra

34

Testing in Cleanroom

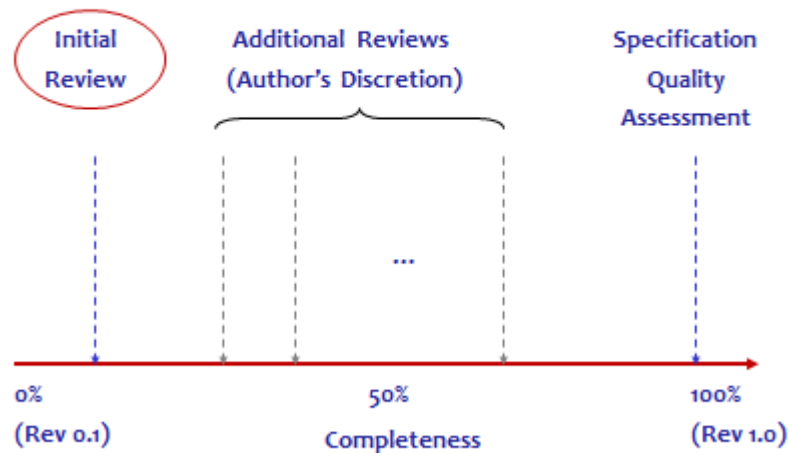
- Testing is an important part of the process, but it is done only after verification (by Inspection) is successfully completed
- Testing is done:
 - Primarily to measure quality
 - Secondarily to find defects that escaped detection during verification
- Number of defects per thousand lines of code 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

35

Help! We have a QA Problem!

Early Inspection

Prevention costs less than Repair



ES

36

Initial Review

- Purpose:** Locating mistakes and tendencies that could lead to injecting major defects if not corrected
- When:** As soon as the author has completed a small representative portion of the specification, typically a few pages or 600-1200 words (e.g. few requirements)
- Who:** Individual or small team (1 or 2)
- Expertise in the subject matter
 - Expertise in generic principles (such as requirements engineering, design, specific language)
- What:** Detailed review of the specification against rules and checklists for known error conditions and dangerous tendencies; formal inspection may be used
- Duration:** Because the sample is small, the initial review takes only 1-2 hr

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

ES

37

Help! We have a QA Problem!

Initial Review Checklist

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

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

ES

38

Case Study 1 - Situation

Large e-business integrated application with 8 requirements authors, varying experience and skill

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

ES

39

Help! We have a QA Problem!

Case Study 1 - Results

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

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

ES

40

Why Early Inspection Works

Many defects are repetitive and can be prevented

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

ES

41

Help! We have a QA Problem!

Case Study 2 - Situation

A tester's improvement writing successive test plans:

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

ES

42

Case Study 2 - Results

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

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

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

ES

43

Help! We have a QA Problem!

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...

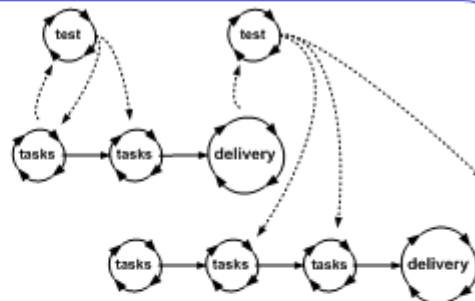
How many of the 3512 defects found in end-of-line inspections could have been completely prevented by Early Inspection?

Cost-effective defect prevention is the bottom line

E 3

44

How to start



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

45

Niels Malotaux

Help! We have a QA Problem!

Help !

We have a QA Problem !

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68

niels@malotaux.nl

www.malotaux.nl

46