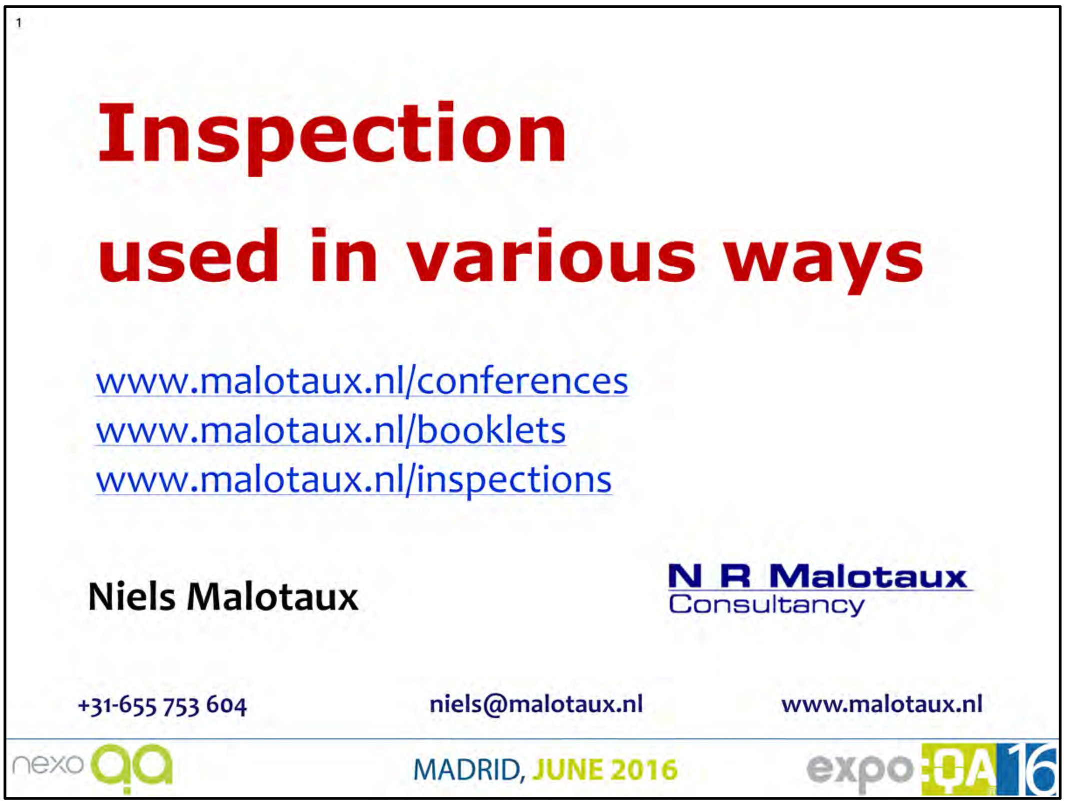

1

Inspection used in various ways

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

Niels Malotaux

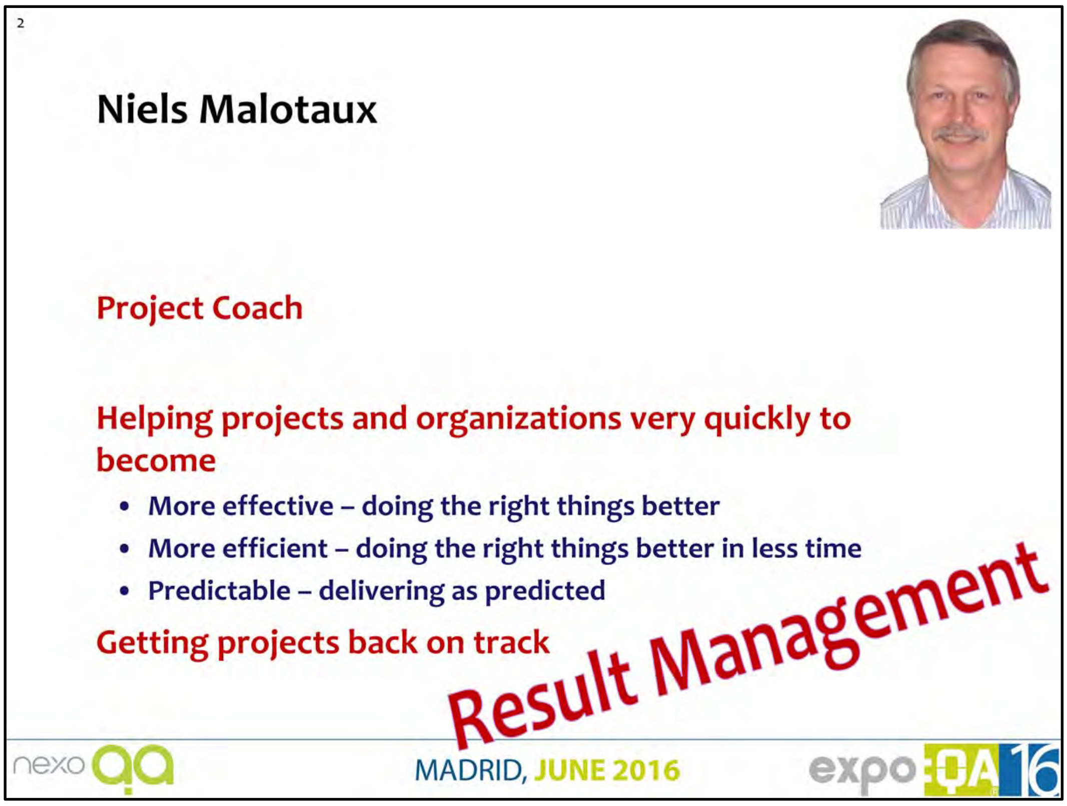
N R Malotaux Consultancy

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

MADRID, JUNE 2016

I would like to talk about how we used Review and Inspection techniques to achieve less expected, but very interesting outcomes.

I will show some cases as illustration.



2

Niels Malotaux

Project 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

Result Management

nexo QA MADRID, JUNE 2016 expo:QA 16

Niels Malotaux

Graduated **Electronics** at Delft University of Technology in **1974**

Army service

at the Dutch Laboratory for Electronic Developments for the Armed Forces, designing computer systems

Philips Electronics

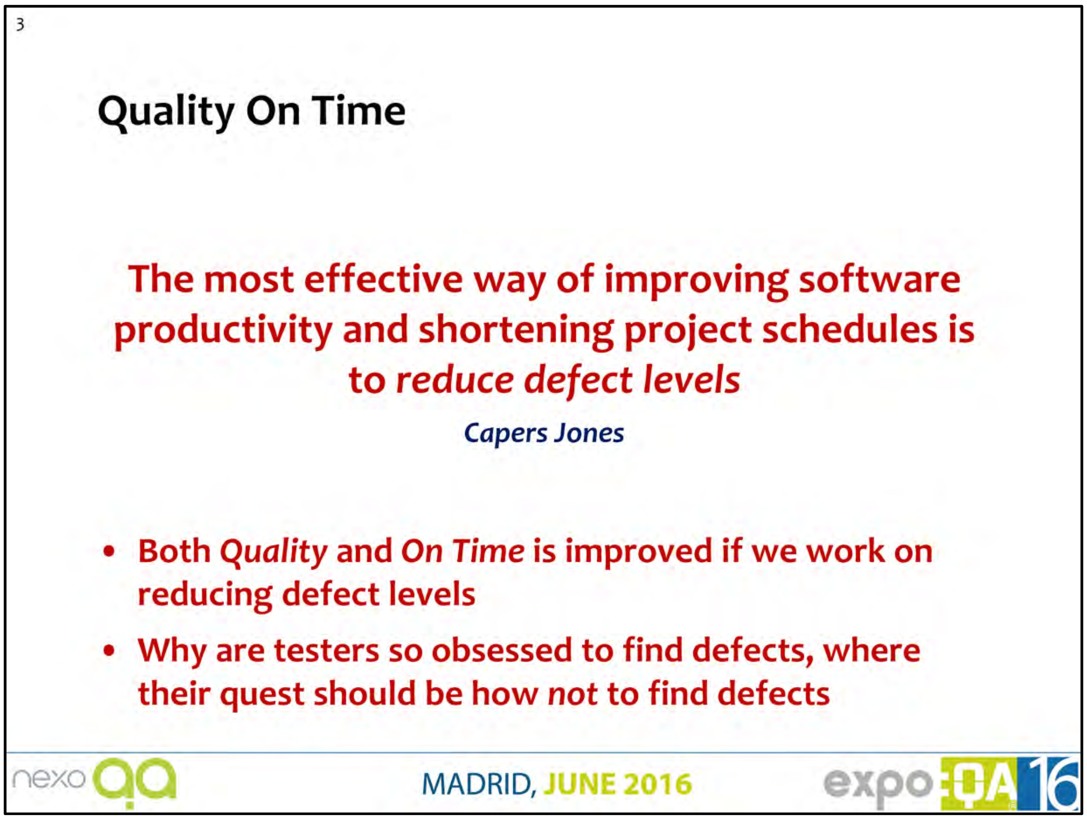
Application support for microcomputer systems design

Malotaux - Electronic Systems Design

Developing electronic systems for clients products

Now: **N R Malotaux – Consultancy**

Coaching projects/teams/organizations/individuals to deliver better results in less time


3

Quality On Time

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

Capers Jones

- **Both *Quality* and *On Time* is improved if we work on reducing defect levels**
- **Why are testers so obsessed to find defects, where their quest should be how *not* to find defects**

nexo QA MADRID, JUNE 2016 expo QA 16

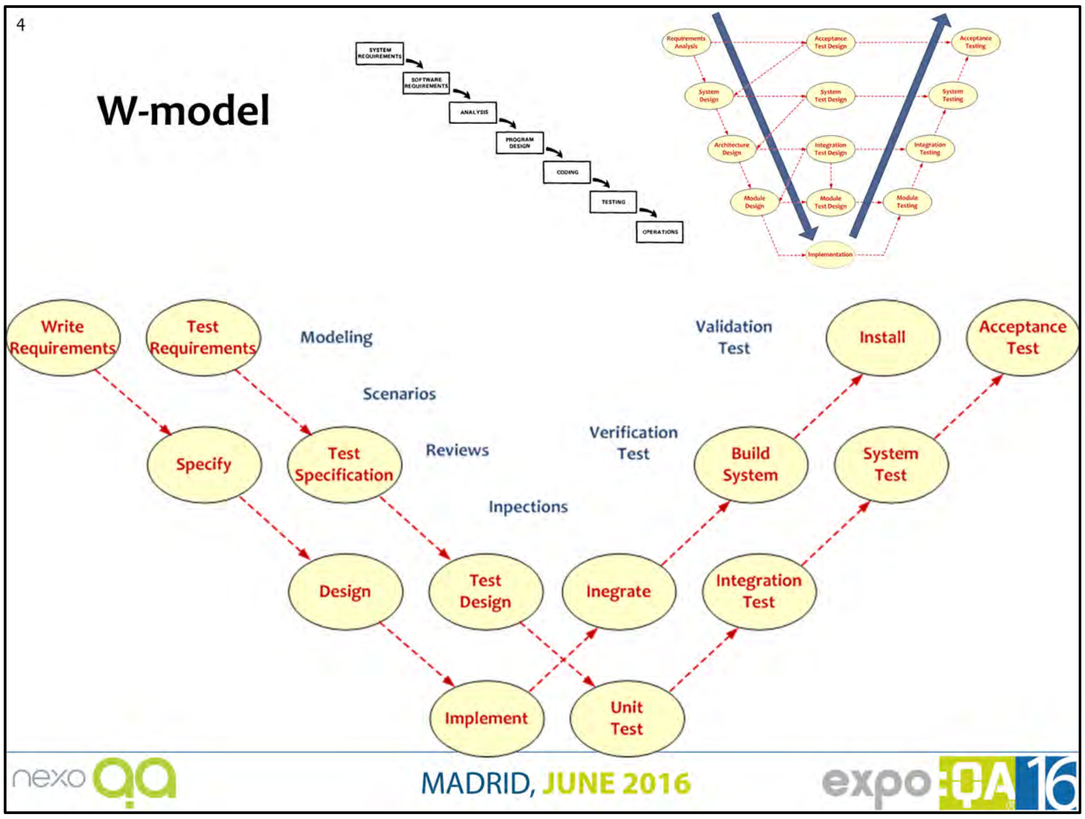
Capers Jones measures software productivity and many other software development metrics all his life.

The claim on this slide is not just an opinion. It's based on actually measuring the performance of thousands of software development projects, including Agile software development.

If Capers talks about 'software productivity', he doesn't just mean delivering software (as in the Agile Manifesto). He means delivering software that works at the required quality level.

So, how can we reduce defect levels?

Reviews and Inspections are known to be very useful techniques to quickly reduce the number of defects injected.


4

W-model

MADRID, JUNE 2016

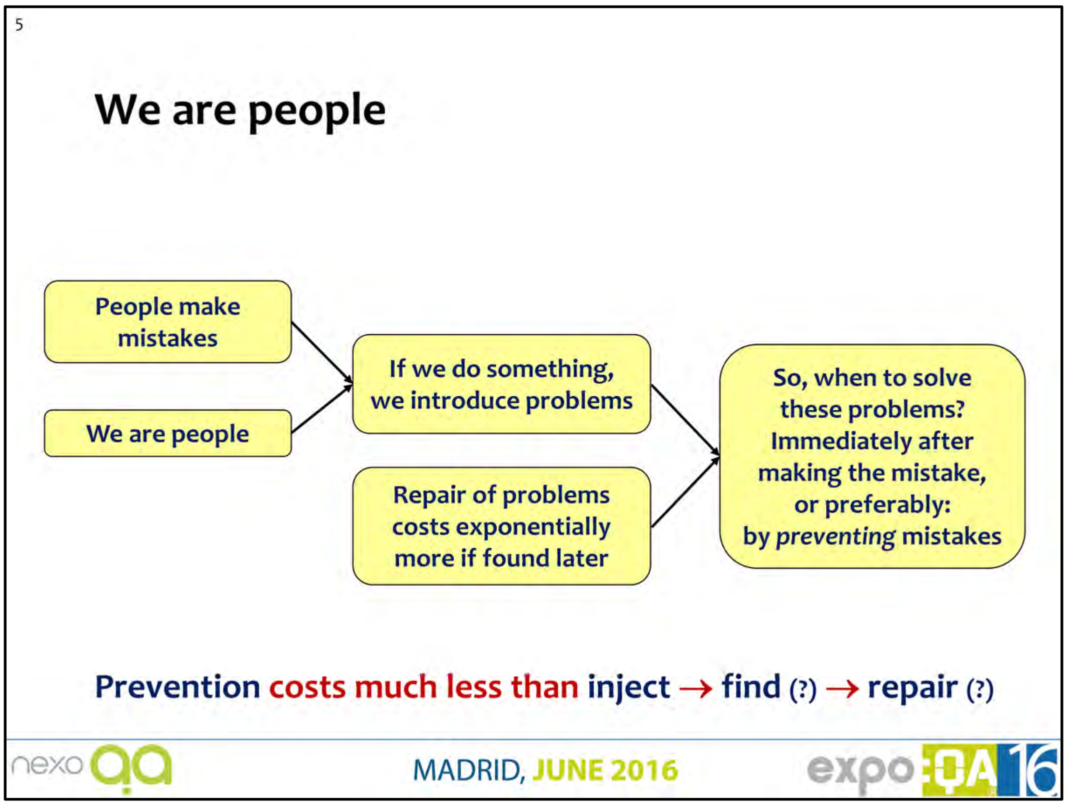
In the Agile world, the Waterfall and derived models are often seen as bad. All models are wrong, some are useful. If they're useful, we should use them.

These models are still applicable to every Sprint. After all, we have to determine what value we should deliver by the end of the Sprint (requirement), how we can and are going to realize it (design), how we implement it (coding), integrate it, test it, and deliver.

For QA the challenge is not to test only the code and find bugs, but to help development to prevent producing problems in the first place. By reviewing the requirement, the design, and the implementation, so that the final test can conclude that it simply works as it is supposed to work.

Inspection is a special form of review.

The V-model is actually a folded Waterfall, where the most expensive issues (requirements issues) are found at the end. The W-model shows that we should find any remaining issues as soon as they are created, rather than when they cause trouble later. The requirement, the design, the code: they're all different manifestations of the same product. However, only the code can be run to check that it does what it is supposed to do. That's what we usually call 'testing'. Before we have code, there are other techniques to check that the product is right, like Modelling, Scenarios, Reviewing, Inspecting. In these areas QA can prove its value as well.


5

We are people

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 these problems? Immediately after making the mistake, or preferably: by *preventing* mistakes

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

Conventional wisdom says that "people make mistakes".

We are people, so if we are preparing some result, we're making mistakes, causing problems.

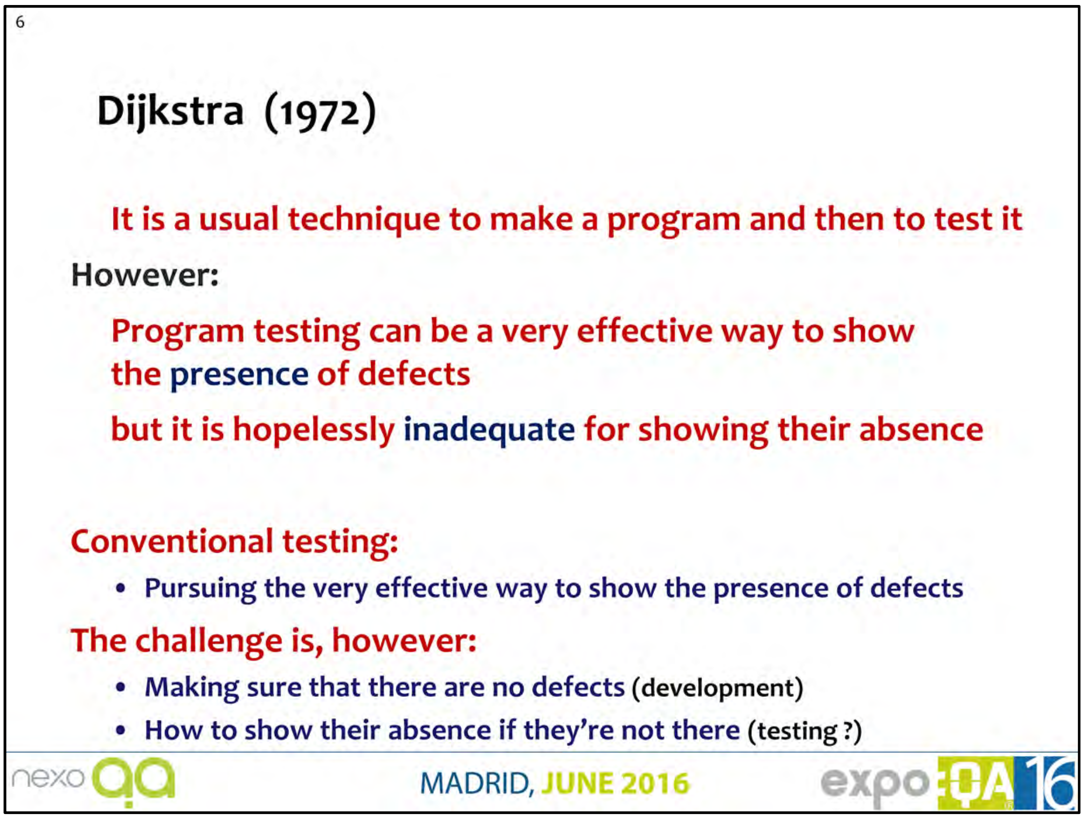
We also have learnt that the longer a problems stays in our result, the costlier it is to find and fix it, if we can and will fix it at all.

Combining these two issues, when should we find and solve the problems caused by the mistakes we, as human beings, are making during our work?

Answer: Immediately after injection of the defect, instead of much later at a 'final test', or even worse, after we caused a hassle to the users.

We can do even better: preferably we should *prevent* issues to be created in the first place: by not making the mistakes at all.

Prevention costs much less than first injecting a defect, then hoping to find it, and then hoping that we can repair it properly, while we know that not all defects are found (testers are also human, aren't they?) and while we know that not all defects found will be repaired properly (usually done under stress), or even cause other defects to emerge or to be introduced.


6

Dijkstra (1972)

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

However:

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

but it is hopelessly inadequate for showing their absence

Conventional testing:

- Pursuing the very effective way to show the presence of defects

The challenge is, however:

- Making sure that there are no defects (development)
- How to show their absence if they're not there (testing ?)

nexo QA MADRID, JUNE 2016 expo QA 16

This is a quote from Edsger Dijkstra, who called himself the first programmer of the Netherlands.

As testing is 'highly inadequate for showing the absence of defects', it seems that testers decided to go for showing the presence of defects, as this is what they can do by testing very effectively. Why so much emphasis on defects while what we should pursue is *no defects*, as that is win-win: what the users are entitled to expect, and Capers Jones has measured to be less costly for us to produce.

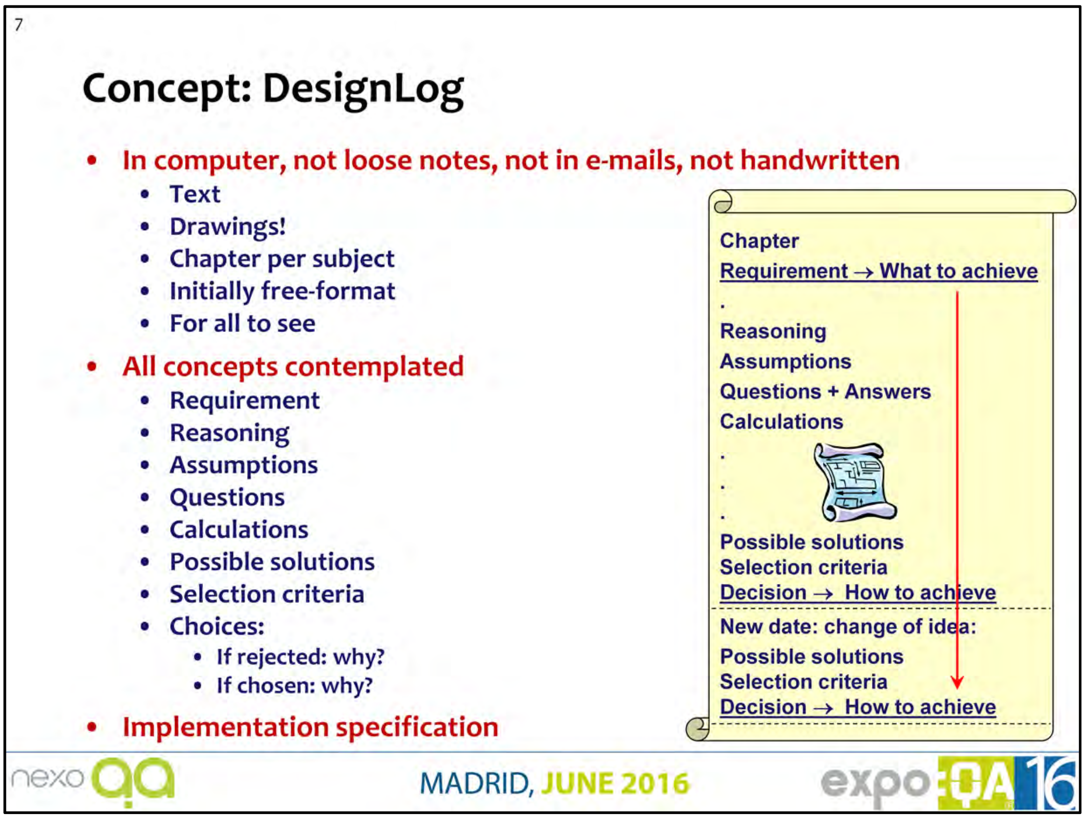
But what should the testers do if they only know how to show the presence of defects, once we have educated the developers not to produce defects anymore? Note: this is not as difficult as some people will try to convince you of!

That's the real challenge: how can we, together with development, make sure there are no defects. Isn't that what we should be after?

I assume that we don't *want* to deliver defects to the users of our software, do we?

Inspections catch issues much earlier, better and much less costly than testing.

Inspections also find different issues, issues that testing would hardly find at all.


7

Concept: DesignLog

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

Chapter
Requirement → What to achieve
.
Reasoning
Assumptions
Questions + Answers
Calculations
.
.
.
Possible solutions
Selection criteria
Decision → How to achieve
New date: change of idea:
Possible solutions
Selection criteria
Decision → How to achieve

nexoQA MADRID, JUNE 2016 expoQA16

For recording our design considerations and decisions, we use the concept of the 'DesignLog'.

When I started my career at Philips Electronics in 1976 (at the same time Philips started to sell its first microprocessor), we got a notebook to note our thoughts, experiments and findings chronologically. It was difficult, however, to retrieve an idea I had several weeks before, because it was buried in many pages of hardly readable handwriting.

Nowadays we can use a word processor, add pictures, organize by subject rather than chronologically, and search through the text. We log our thoughts in chapters, which start with what we have to achieve (requirement), end with how we think we will achieve it (implementation specification), with in between the reasoning, assumptions, questions and answers, possible solutions, decision criteria and the selected solution (design).

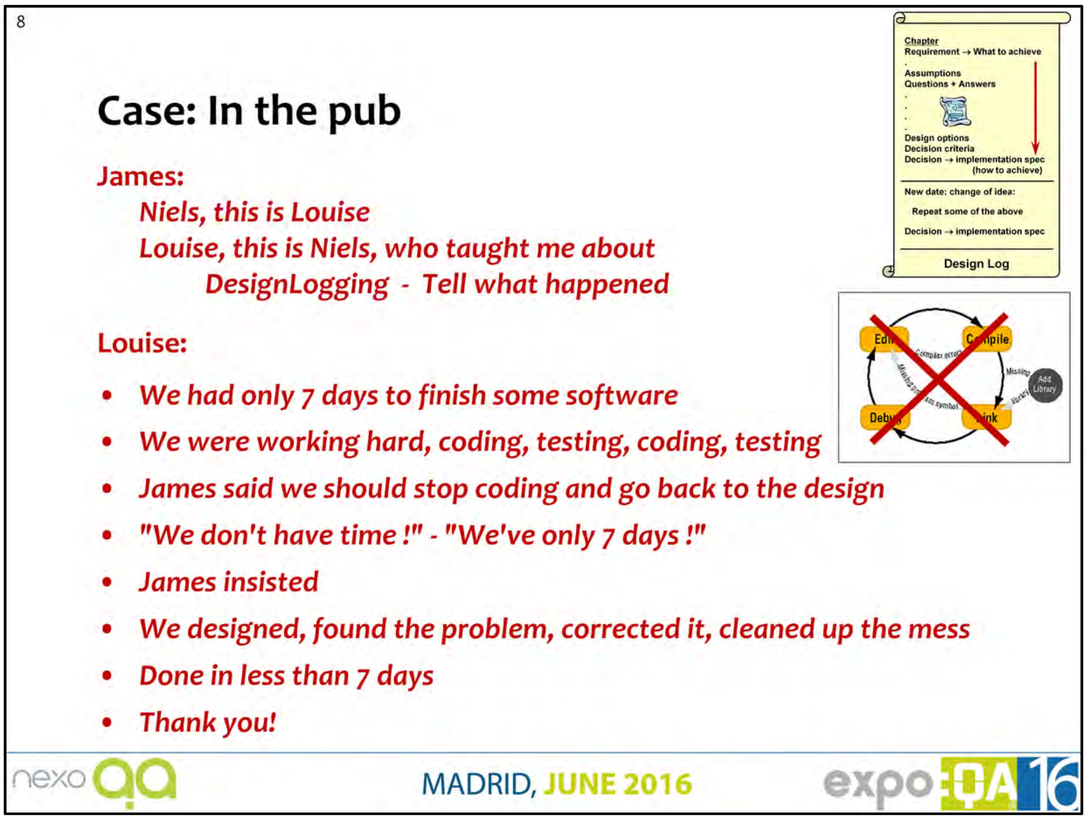
If I see design documentation, this often only shows what people decided to do, rather than also recording why and how they arrived at this decision.

The DesignLog should be reviewed to find possible issues before we start the implementation. Because the choices and design are well documented, and minimum time is lost in the maintenance of the software. Maintenance often being the largest portion of the cost of deployment!

People ask me: "How much detail do I need to put in the DesignLog?"

My answer is: "You'll have to find out yourself. One of the requirements for the quality of the DesignLog is: 'If someone has to change something in the software one year later, he should be up and running within a day.' "

When QA asks development to review the DesignLog, if there is one they can review and also use this information to define and optimize the test-approach. If there is none, this is a good time to introduce the concept.



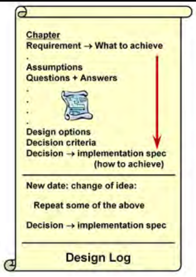
Case: In the pub

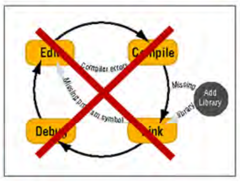
James:

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

Louise:

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





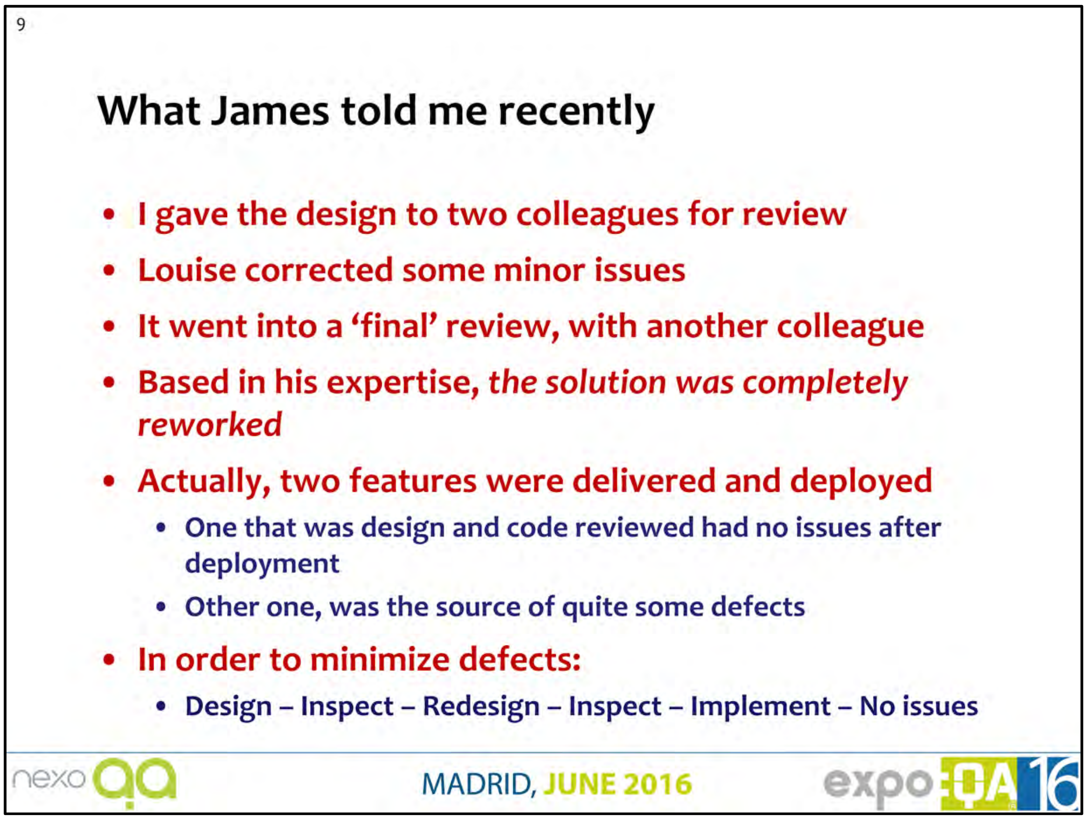
This case happened just a few months ago, see the text on the slide.

It's always nice to experience that the techniques that worked for me and for many others in the past, still work today. Many old techniques never get out of date.

We see, however, that it's not so easy to convince people to do something that seems counter-intuitive: going back to the design rather than grinding on in code and leaving a lot of dangerous scars in the process.

Delivering quality often needs counter-intuitive measures.

On the next slide we'll see how a review even caused the whole design to be changed!


9

What James told me recently

- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a 'final' review, with another colleague
- Based in his expertise, *the solution was completely reworked*
- Actually, two features were delivered and deployed
    - One that was design and code reviewed had no issues after deployment
    - Other one, was the source of quite some defects
- In order to minimize defects:
    - Design – Inspect – Redesign – Inspect – Implement – No issues

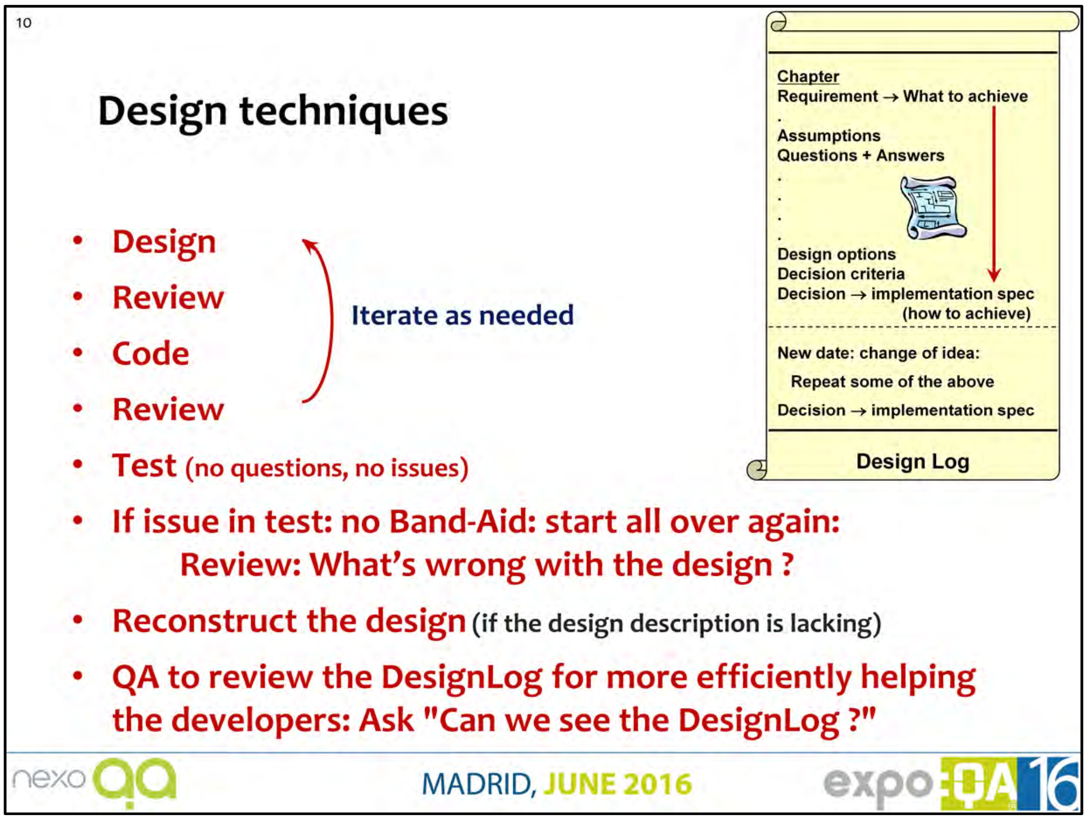
nexo QA MADRID, JUNE 2016 expo QA 16

James explained me later more interesting details of this case:

- There were two features required for a release, one of which was on the critical path and placed the delivery at risk
- I saw an "opportunity" for a Design - "*prevention, rather than fixing*" and also an opportunity to encourage documentation
- Because Louise struggled a bit with the design (not many people in software have been educated in how to design), we Timeboxed the initial draft
- I emailed it to two colleagues to review: "please review, assuming you will code-review the implementation, and based on this DesignLog you know what the implementation you will have to code-review will be"
- Louise emailed me in a panic that if she knew it would be reviewed, she would have written something different. I said - "no, do nothing yet - review first and then update with your new understanding and feedback. Reason: the next draft will be better."
- For the next review, another colleague who was not previously available was invited. At the meeting where I expected the DesignLog to be approved with minor modifications and to get estimates for the work involved, the Design was *totally reworked*
- We agreed the new Design was better than the original ideas
- Actually, two features were delivered and deployed
    - The one that was designed, reviewed, coded, and code reviewed had no issues after deployment
    - Another one, which was done in the 'traditional' way, was the source of quite a few defects
- In summary, this success has proved instrumental in buy-in for DesignLogs which are now embedded in the development process

Using Inspection (Review) in various ways:

- The review caused them not to implement a bad solution
- If they would have implemented the original solution, they probably wouldn't have found out until much later
- The whole process allowed them to deliver well before the deadline rather than after.


10

Design techniques

- 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 ?"

Chapter
Requirement → What to achieve
.
Assumptions
Questions + Answers
.
.
.
.
Design options
Decision criteria
Decision → implementation spec
(how to achieve)

New date: change of idea:
Repeat some of the above
Decision → implementation spec

Design Log

nexoQA MADRID, JUNE 2016 expoQA16

In the previous case we saw the power of design – review – redesign – review – code – review – resulting in test and user not finding any issues.

For recording our design considerations and decisions, we use the concept of the 'DesignLog'.

When I started my career at Philips Electronics in 1976 (at the same time Philips started to sell its first microprocessor), we got a notebook to note our thoughts, experiments and findings chronologically. It was difficult, however, to retrieve an idea I had several weeks before, because it was buried in many pages of hardly readable handwriting.

Nowadays we can use a word processor, add pictures, organize by subject rather than chronologically, and search through the text. We log our thoughts in chapters, which start with what we have to achieve (requirement), end with how we think we will achieve it (implementation specification), with in between the reasoning, assumptions, questions and answers, possible solutions, decision criteria and the selected solution (design).

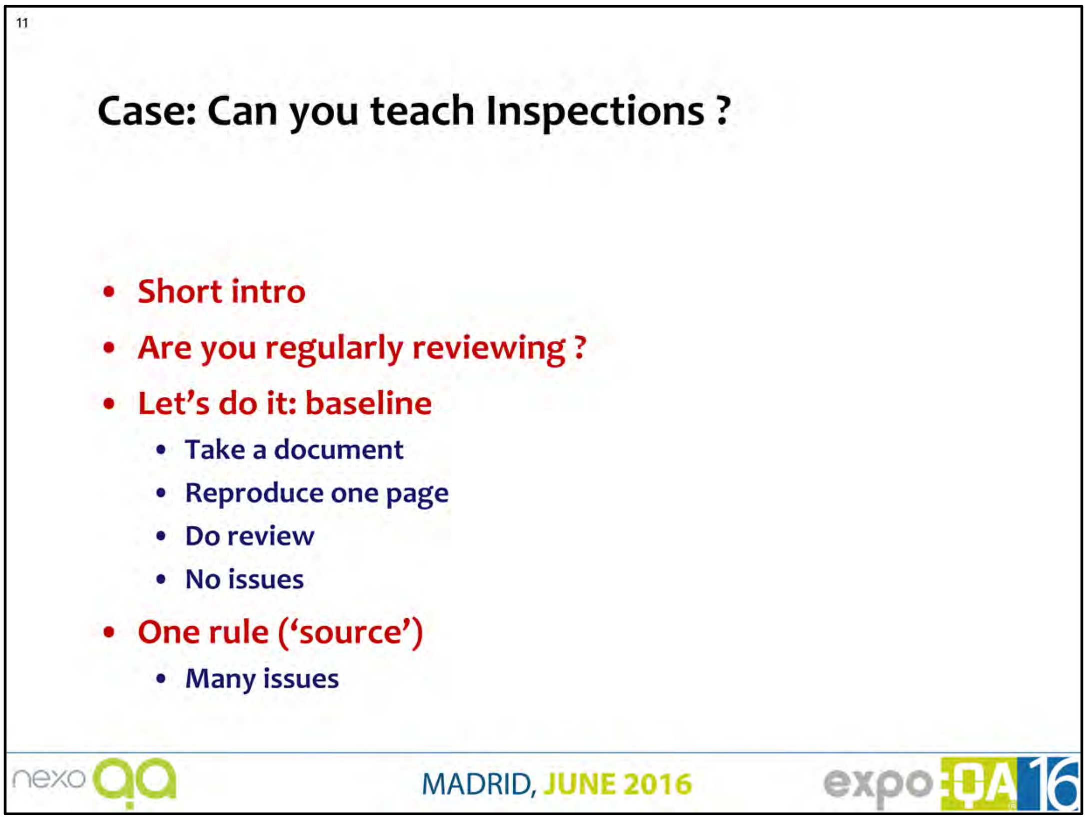
If I see design documentation, this often only shows what people decided to do, rather than also recording why and how they arrived at this decision.

The DesignLog should be reviewed to find possible issues before we start the implementation. Because the choices and design are well documented, and minimum time is lost in the maintenance of the software. Maintenance often being the largest portion of the cost of deployment!

People ask me: "How much detail do I need to put in the DesignLog?"

My answer is: "You'll have to find out yourself. One of the requirements for the quality of the DesignLog is: 'If someone has to change something in the software one year later, he should be up and running within a day.' "

When QA asks development to review the DesignLog, if there is one they can review and also use this information to define and optimize the test-approach. If there is none, this is a good time to introduce the concept.


11

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

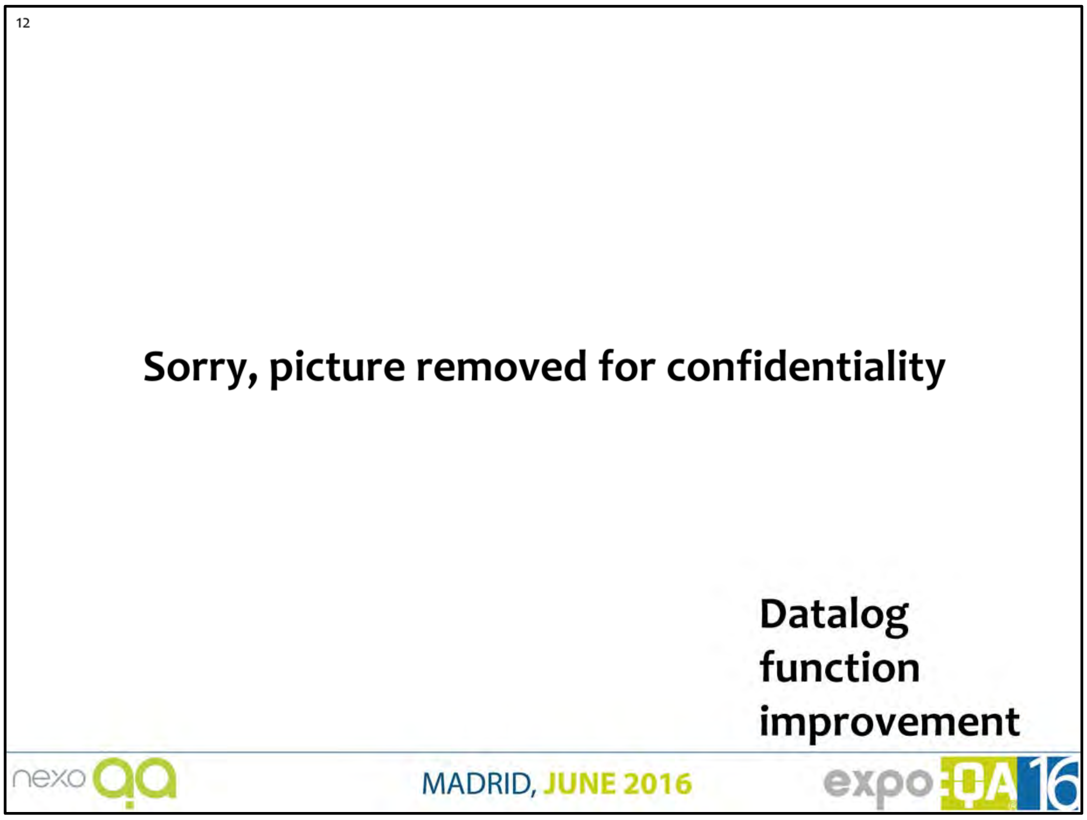
nexo QA MADRID, JUNE 2016 expo QA 16

In another case I was asked to teach Document Inspections to a group of developers. I gave a short introduction and then we did a baseline review. After all, most developers do reviews, don't they?

We selected a design document for some firmware datalog functionality in a controller, took one page from the document, made 20 copies of that page for everyone to review. They started reviewing and after some 10 minutes everyone seemed ready.
I asked about the issues found. Hardly any. Perhaps a typo or two.

Then I introduced a 'rule' (In Inspections, we use 'rules', which are the 'laws' for making a useful document): "If we don't know the requirement of this design, how do we know that the design does what it should do and does *not* what it should *not* do?" I asked them to review once more.

After a short while, everyone's paper was full of remarks: This I cannot judge, that I cannot judge, because I don't know what was required and why this is the best solution. With the review they had found out *themselves* that there was a lack of knowledge what 'design' actually means. If I would have told them, they wouldn't have accepted it. Now they showed it to themselves. This was an interesting alternative use of the Inspection technique!

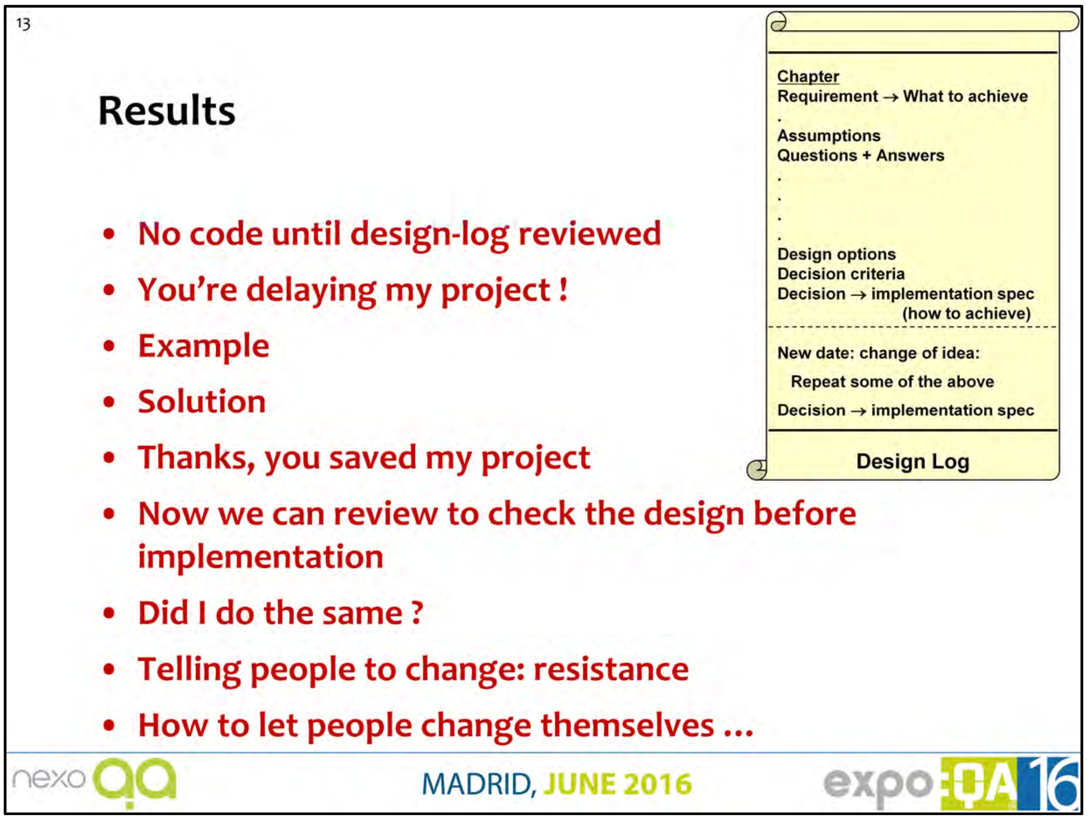

12

Sorry, picture removed for confidentiality

Datalog function improvement

MADRID, JUNE 2016

Explanation of the datalog environment



13

Results

- No code until design-log 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 …

Chapter
Requirement → What to achieve
.
Assumptions
Questions + Answers
.
.
.
.
Design options
Decision criteria
Decision → implementation spec
(how to achieve)

New date: change of idea:
Repeat some of the above
Decision → implementation spec

Design Log

nexo QA MADRID, JUNE 2016 expo QA 16

We suggested the designer to make a DesignLog. Not to write even one line of code until the DesignLog was reviewed and found OK.

It took some time until the author understood how to make the DesignLog, but it led to an interesting conclusion: He decided not to implement the functionality in the controller firmware, because of some intricacies which could much better be solved in the PC software at the other side of the network, when analysing the datalog data.

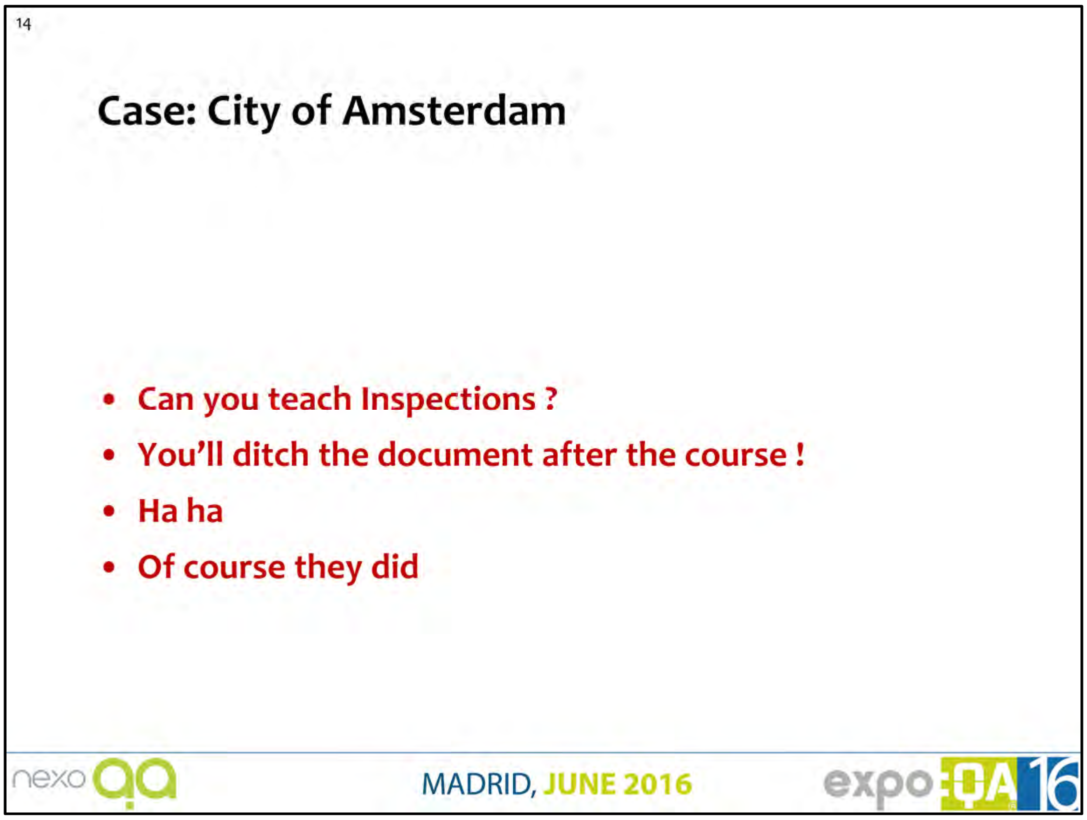
Imagine what would have happened if he had started coding already. Getting deeper and deeper into trouble, not wanting to stop, because having spent already so much time on coding.

First he had complained that I was delaying his project because he wasn't allowed to start coding. Later he said: Thank you, you saved my project!

The design and the Inspection caused an unanticipated decision.

Using Inspection in an interesting way:

- If I would have suggested that the document wasn't right, he wouldn't have believed it.
- Now they *showed themselves* that there was a problem with design. That's much more powerful!
- Instead of proceeding further with Inspections, we first started working on what design was about.


14

Case: City of Amsterdam

- Can you teach Inspections ?
- You'll ditch the document after the course !
- Ha ha
- Of course they did

nexo QA MADRID, JUNE 2016 expo QA 16

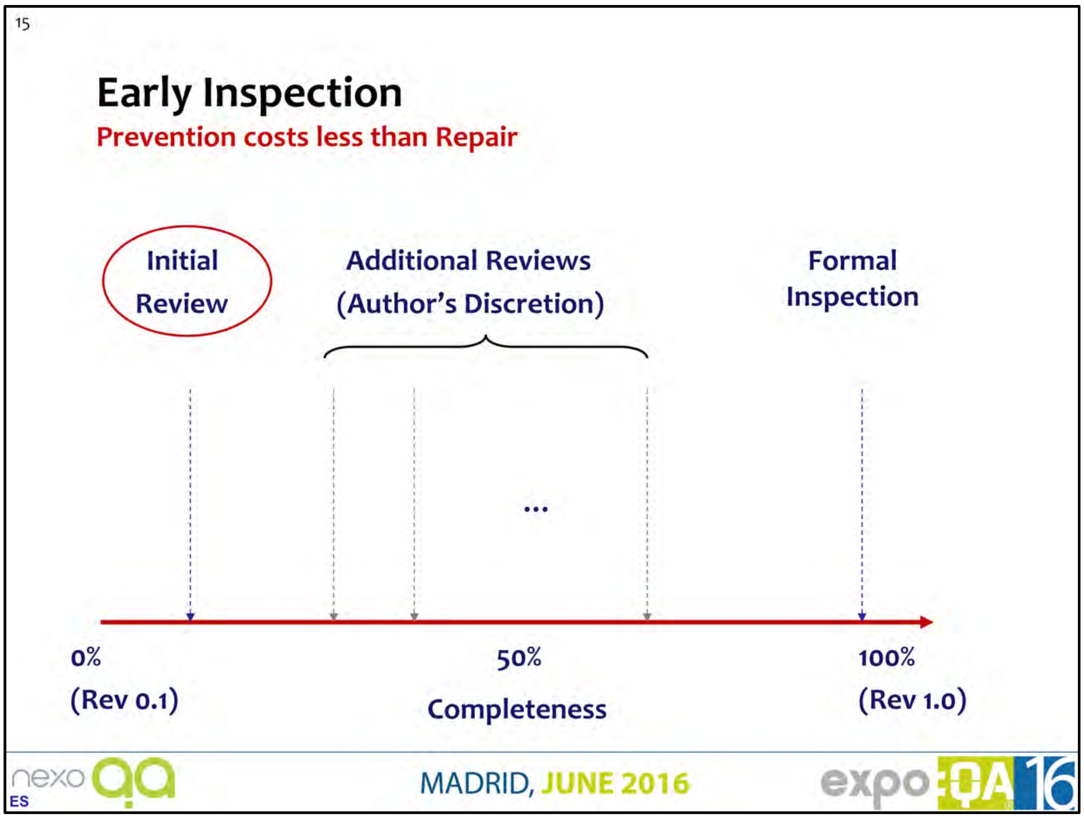
A team of the City of Amsterdam wrote a RfP (Request for Proposal) for some software development. Apparently they had some feeling that the RfP might have some issues, so they asked me to do an Inspection training with this document as training material.

When they called me for the training, I already warned them that after the training they probably would throw the document in the waste-bin, as that's my experience with first Inspections. They only laughed about this, not really believing me.

You can guess what happened at the end of the day: they ditched the document as useless. Within a few days, they even redefined the whole project, because of what they learnt by inspecting the document.

Using Inspection in various ways:

- If I would have suggested that the document wasn't right, they wouldn't have believed it.
- Now they showed themselves that it wasn't right. That's much more powerful!
- By doing the inspection, they learn so much about what their project should be about, that they redefined the project completely.
- Guess what would have happened if they would have sent out the RfP to prospective suppliers, who love those unclear documents, because it will allow them to sell many more hours than initially agreed. And apparently the City would have got a great solution for the wrong problem.


15

Early Inspection

Prevention costs less than Repair

nexo qa ES — MADRID, JUNE 2016 — expo QA 16

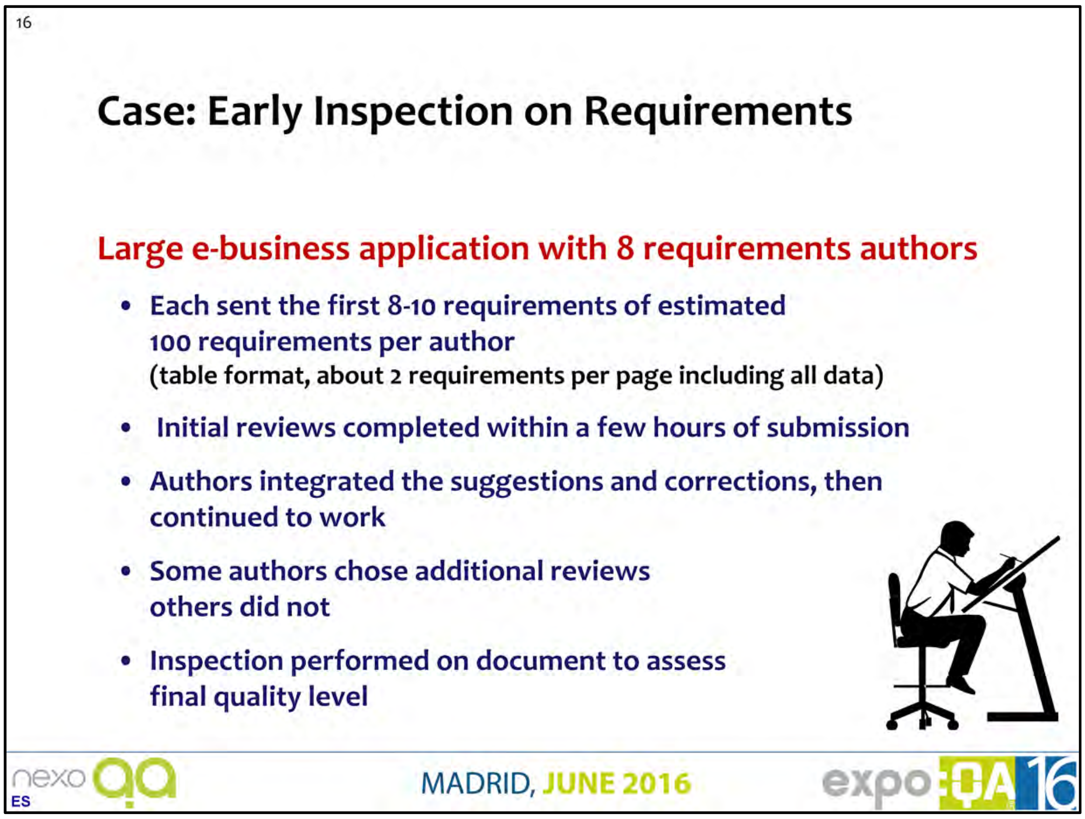
Early Inspection:

Why would we allow an author to complete the whole waterfall of a list of requirements, a design, a code module (for this Sprint), knowing that he is in the process of injecting a certain amount of defects?

How about after the author produced some 10%, reviewing this part, ploughing back the findings of the review immediately, so that the author can correct the issues in the first 10% and then prevent injecting similar defects in the remainder of his work?

I have a few case-studies (cannot mention the name of the large company), showing a huge Return on Investment of the Early Inspection technique. We always routinely calculate the RoI, to show that the time invested was worth spending. After all, we're always aware that we should spend our time on productive things, so we have to check that we are.

Because of time limitations, I'll show just one example (next slide).

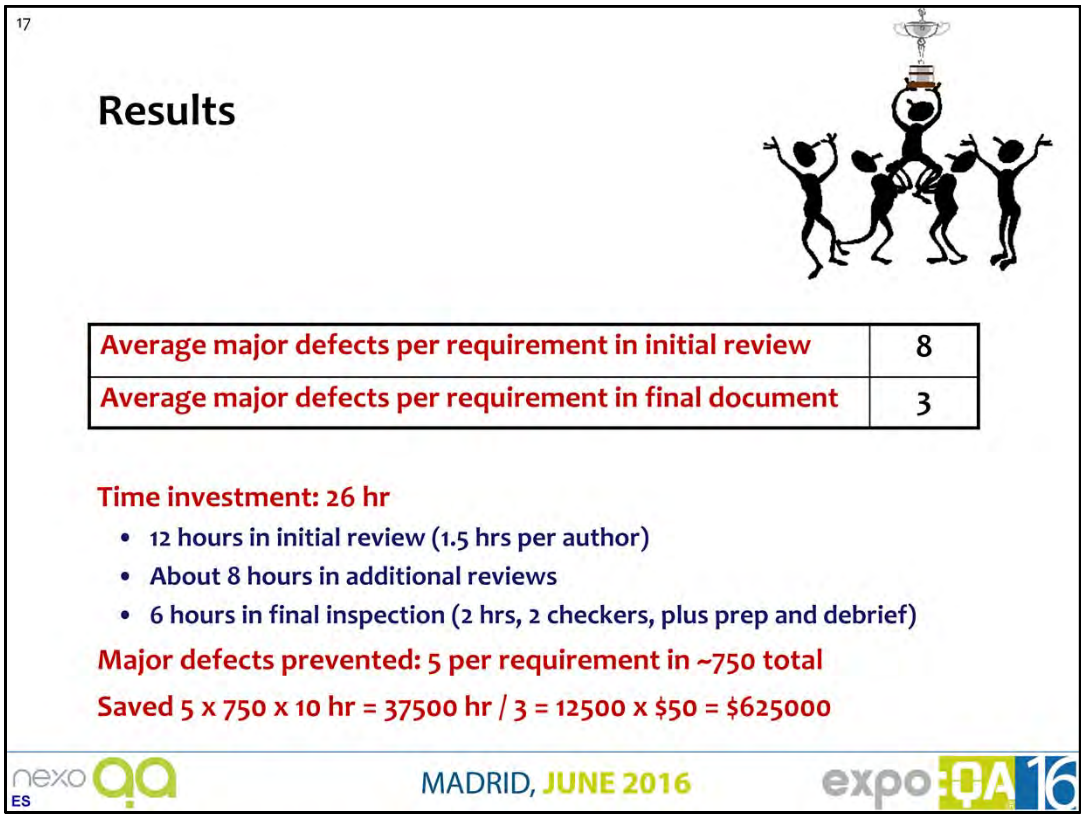


Case: Early Inspection on Requirements

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

Case study – what we did.



17

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

nexo QA ES MADRID, JUNE 2016 expo QA 16

Case study – results.

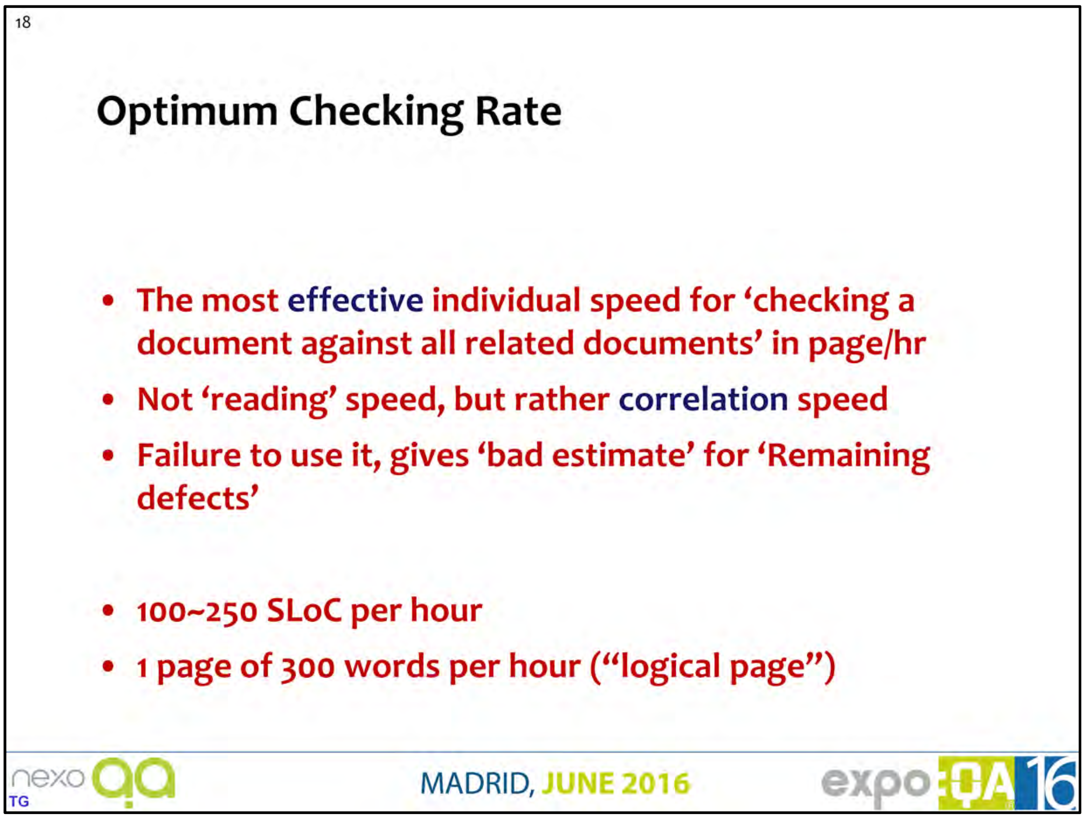
Major defects per requirement initially: 8

Major defects per requirement in final document: 3

Time saved in this case: some 37.500 hr (at least 1200 person-days).

Cost saved in this case: some $625.000.

I'll explain how we do these calculations.


18

Optimum Checking Rate

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

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

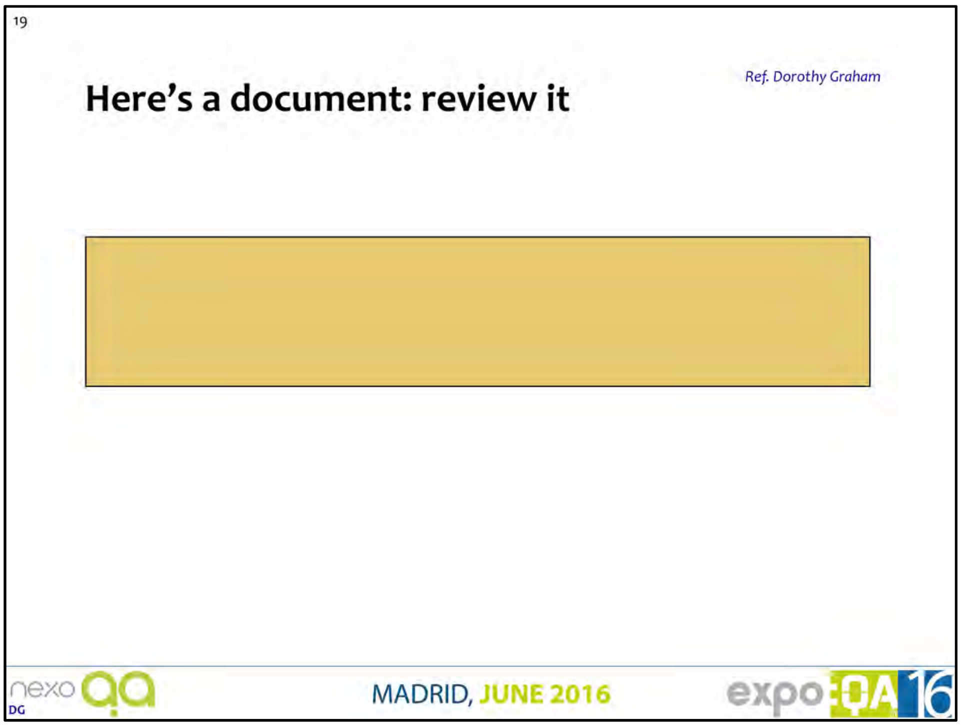
nexo QA TG
MADRID, JUNE 2016
expo QA 16

When I attended an Inspection Tutorial by Dorothy Graham, I had this 'Aha moment' about using sampling with Inspections.

When people first get the message that they should spend *one hour* per page of a document, they don't believe it (I didn't at first).

People who are used to doing reviews, but never had proper training, usually hardly find any issues. After some training they start understanding how to spot issues in a way they never contemplated before, and now start finding loads of issues on each page. You won't believe this until you experienced it yourself. I didn't believe it, until I tried it.

Actually, I found that when people start doing Inspections seriously, within some 20 min they already find so many issues on any page, that spending more time doesn't make sense.


19

Ref. Dorothy Graham

Here's a document: review it

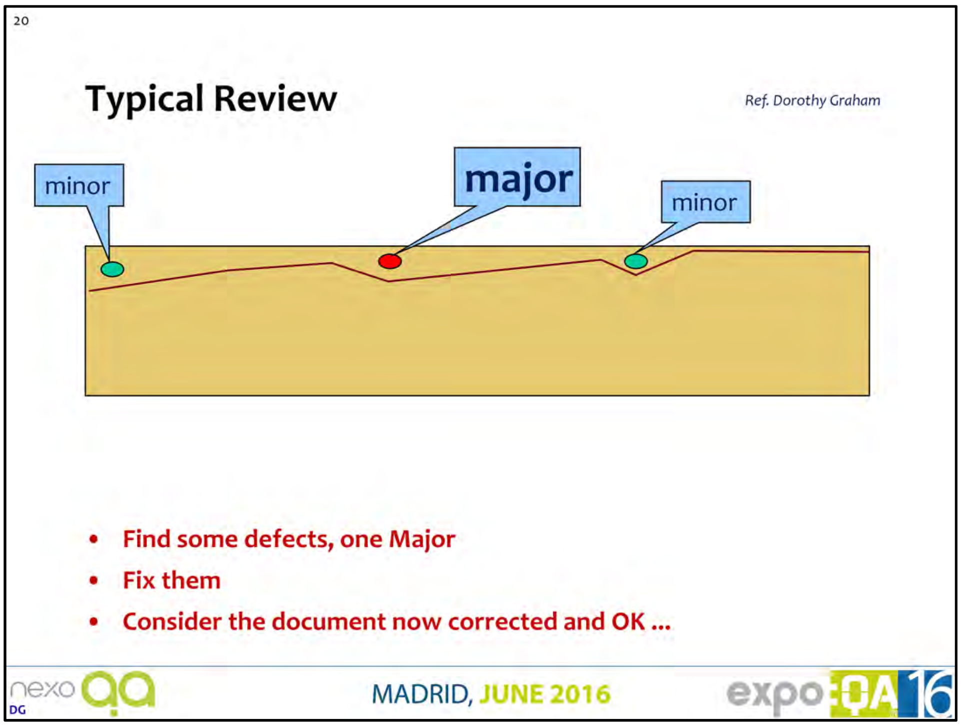
nexo QA DG   MADRID, JUNE 2016   expo QA 16

This and the next two slides are originally from Dorothy Graham. I had used these slides already many times in my own Inspection workshops.

When I had the opportunity to actually attend an Inspection Tutorial with Dorothy, I suddenly got an interesting 'Aha-moment'.

Normally, people are given a document for review: "Review this document".

In the datalog case we already found out what may happen then.


20

Typical Review

Ref. Dorothy Graham

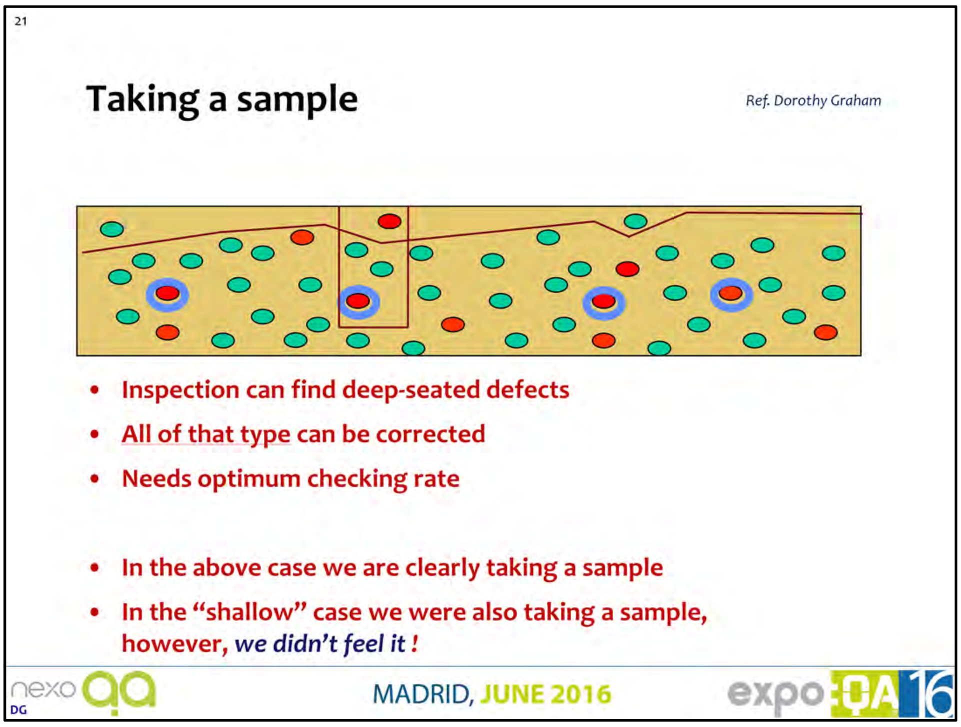
minor
major
minor

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

nexo QA DG MADRID, JUNE 2016 expo:QA16

They tend to read the whole document (authors usually want us to 'review everything', in order to 'find all issues'), using a few minutes per page 'otherwise it will take too long'.

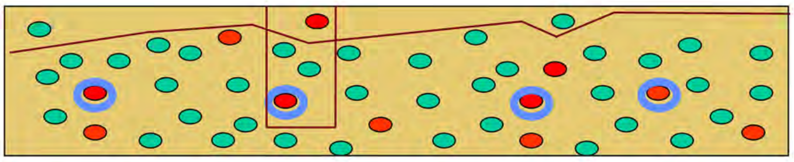
They find some issues, but by not following the 'optimum checking rate', they find only issues directly at the surface. They do not find the 'deeper seated' issues, which take more time for our mind to discover.



21

Taking a sample

Ref. Dorothy Graham



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

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

nexo QA DG MADRID, JUNE 2016 expoQA16

Because it takes some time before our mind detects the 'deeper seated' issues, the advice is to take a sample of only a few pages, and use the 'optimum checking rate' for these pages. That's not just reading the page many times, but also checking the correlation and consistency with other parts of the document as well as with other documents: How can we judge the correctness of a design, if we don't check what the design should accomplish (the requirement). Or: how can we judge the correctness of a code module, if we don't know the design the code is supposed to implement.

Most authors initially don't like us taking 'only' a sample. They want us to 'check the whole document'.

When for the first time hearing Dorothy herself explaining these principles, I suddenly got this 'Aha-moment':

When taking a 'vertical sample', it's clear that we are studying only a part of the document. If we are going through the whole document, as shown in the previous slide, we are, however, *also taking a sample*, namely a 'horizontal sample' (shallow sample), but we're *not aware of the fact that we are also taking a sample*.

This was also the first time Dorothy realized this.

22

Inspection used in various ways

Niels Malotaux

N R Malotaux Consultancy

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

nexo QA MADRID, JUNE 2016 expo QA 16

Most people who review code or any other document have not been trained in how to Review or Inspect, hence they miss most of the real issues that should be found, and hence these reviews cost time without much return on investment. This causes one of the most important outcomes of reviews and Inspections to be missed, namely the quick learning of how to prevent introducing issues in the first place.

Most reviews are just mechanically performed, reporting some minor issues, and then people continue with their other work.

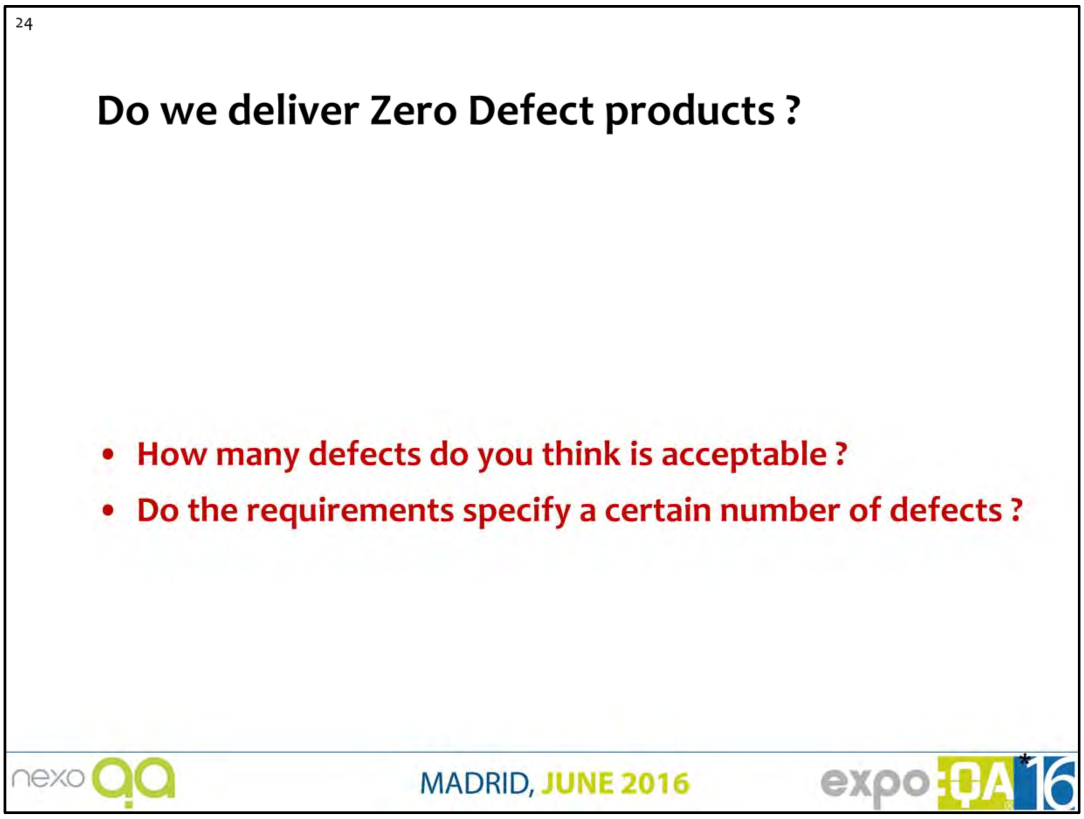
Here I have shown several cases of using Reviews and Inspections with various unexpected and interesting outcomes:

- The review caused a redesign
- The review caused not to implement at all
- The review caused the document to be discarded as completely useless for its purpose

And we've seen the use of early inspections to feed prevention even faster.

I hope that this will provide you with more flexible insights of applying Reviews and Inspections to improve the quality of the software you and your teams produce, which will lead to better Quality On Time: delivering better results, spending less time.

Some extra
(no time to present)



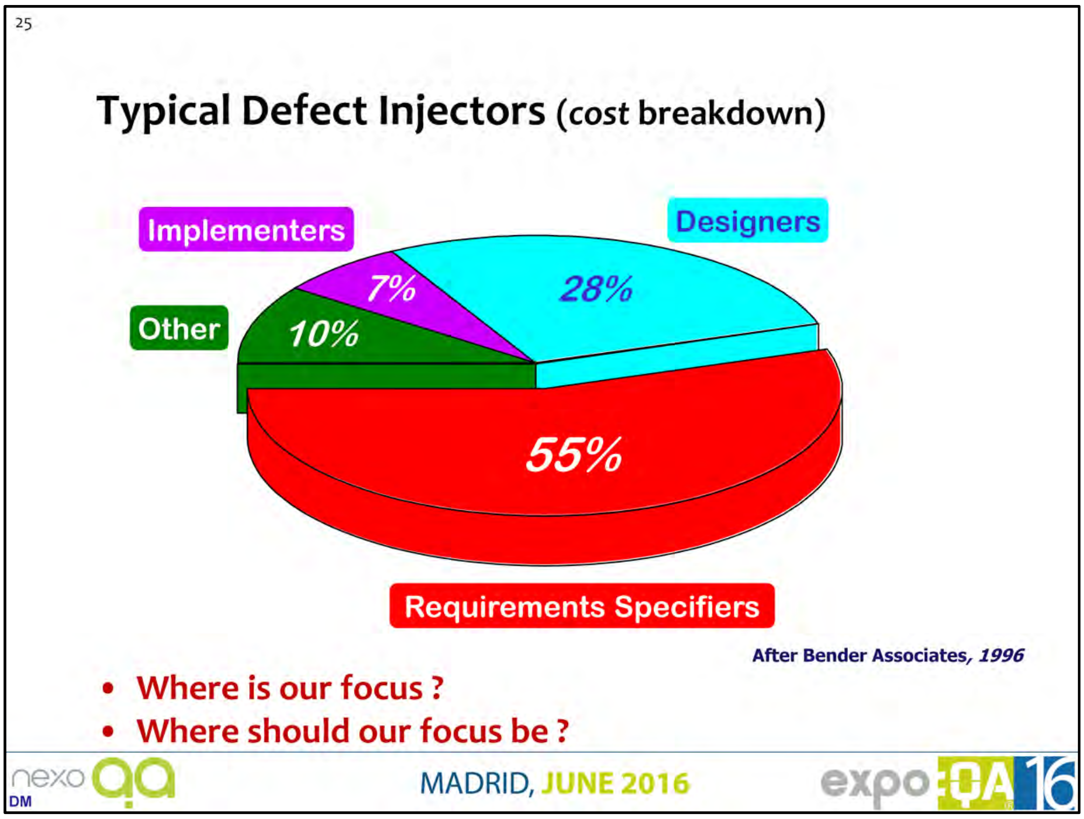
24

Do we deliver Zero Defect products ?

- How many defects do you think is acceptable ?
- Do the requirements specify a certain number of defects ?

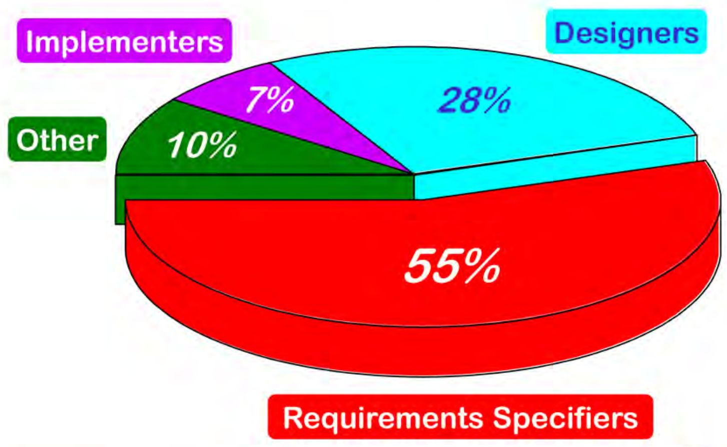
nexo QA MADRID, JUNE 2016 expo QA 16

Have a look at a video about Zero Defects, similar as presented at ExpoQA15: http://tinyurl.com/nn5gf7c



25

Typical Defect Injectors (*cost* breakdown)

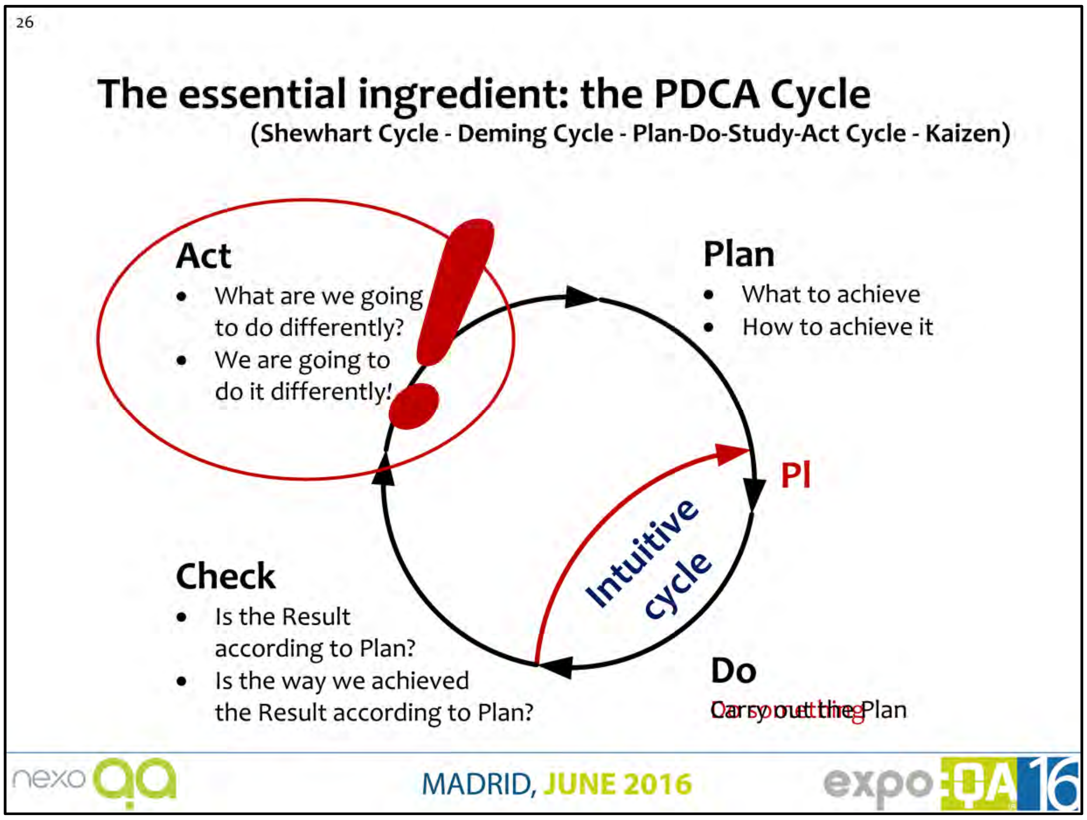


After Bender Associates, *1996*

- Where is our focus ?
- Where should our focus be ?

nexo QA DM MADRID, JUNE 2016 expo QA 16

If most of the cost of defects is in Requirements and Design, why is there so much emphasis on testing the code? Inspections by trained Inspectors help to find issues before the code is made. Saves a lot of effort and time.


26

The essential ingredient: the PDCA Cycle

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

Act
- What are we going to do differently?
- We are going to do it differently!

Plan
- What to achieve
- How to achieve it

Check
- Is the Result according to Plan?
- Is the way we achieved the Result according to Plan?

Do
Carry out the Plan

nexo qa MADRID, JUNE 2016 expo QA 16

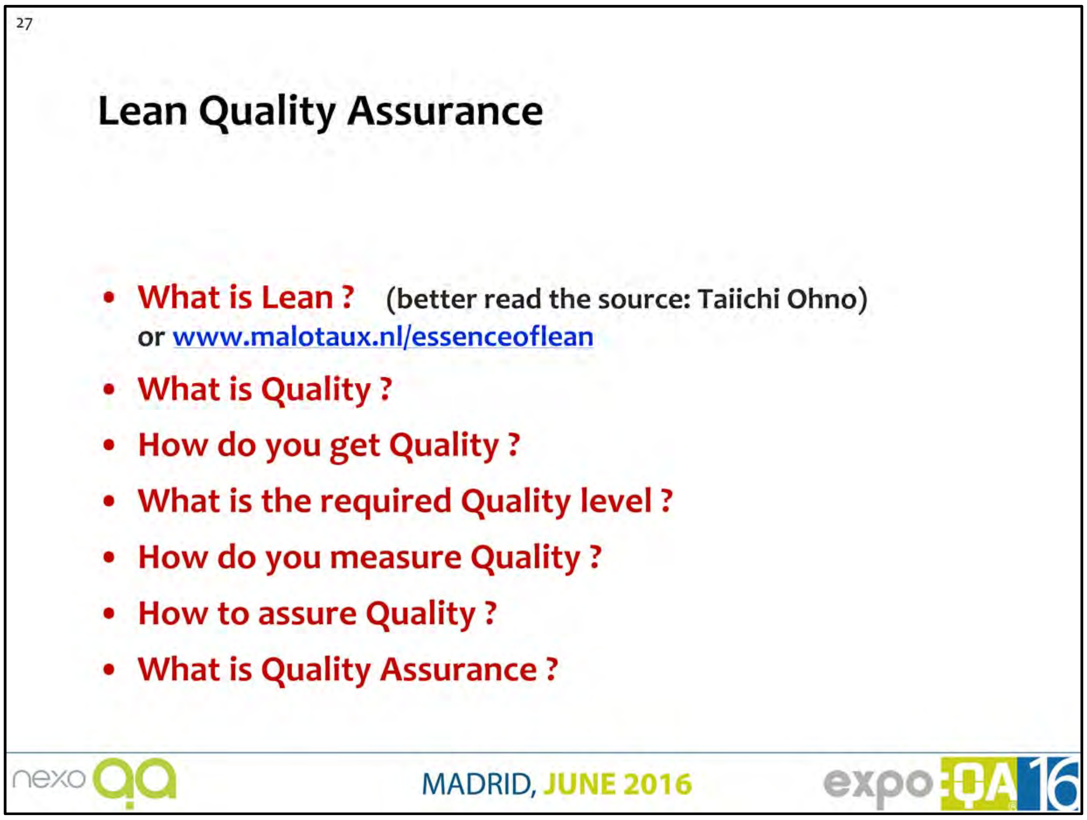
The essential technique for continuous improvement is the Deming or Plan-Do-Check-Act cycle. We Do all the time, Planning we do more or less, usually less and for Check and Act we don't have time.

Many people think they know the Deming cycle, but let's see how it really starts working for us.

The intuitive cycle, how we normally work, is the Pl-Do-cycle. I can't call it Plan, so I call it only Pl. "What was the next thing we are supposed to do?" and we are already doing it. If intuition would be perfect, everything would be perfect. Not everything we do is perfect, so apparently our intuition sometimes points us into the wrong direction.

So, let's first Plan what Result we want to achieve and how we think we can most efficiently achieve that (Planning is twofold: the product and the project). Then we Do according to the Plan. This is the first pitfall: the Plan must be doable and we must follow the Plan. Let's assume we did that, then in the Check phase we can Check (Deming also called it Study phase) whether the Result was according to Plan. If it was according to the Plan, we can think: "Can we do it even better the next time?". If it wasn't according to Plan, we can think: "How can we do it better the next time?". Then comes the Act phase: "What are we going to do differently the next time, because if we don't do anything differently, the result will be the same. If we want to improve we have to decide to do something differently, then Plan and Do accordingly and then Check whether the change actually was an improvement. If yes, can we do it better the next time. If not, can we do it better the next time. In the Act phase we introduce a "mutation" in our way of working, hence we call it the "Evolutionary" approach.

This way, we are continuously improving on the Result (the product), the way we realize the Result (the project) and even how we organize all of this (the process). Actually we can stop now, because using the PDCA technique, you can start from scratch and very quickly find out how to continuously do things better. Because we have been doing this already for a long time, we can save you time and give you a flying start.


27

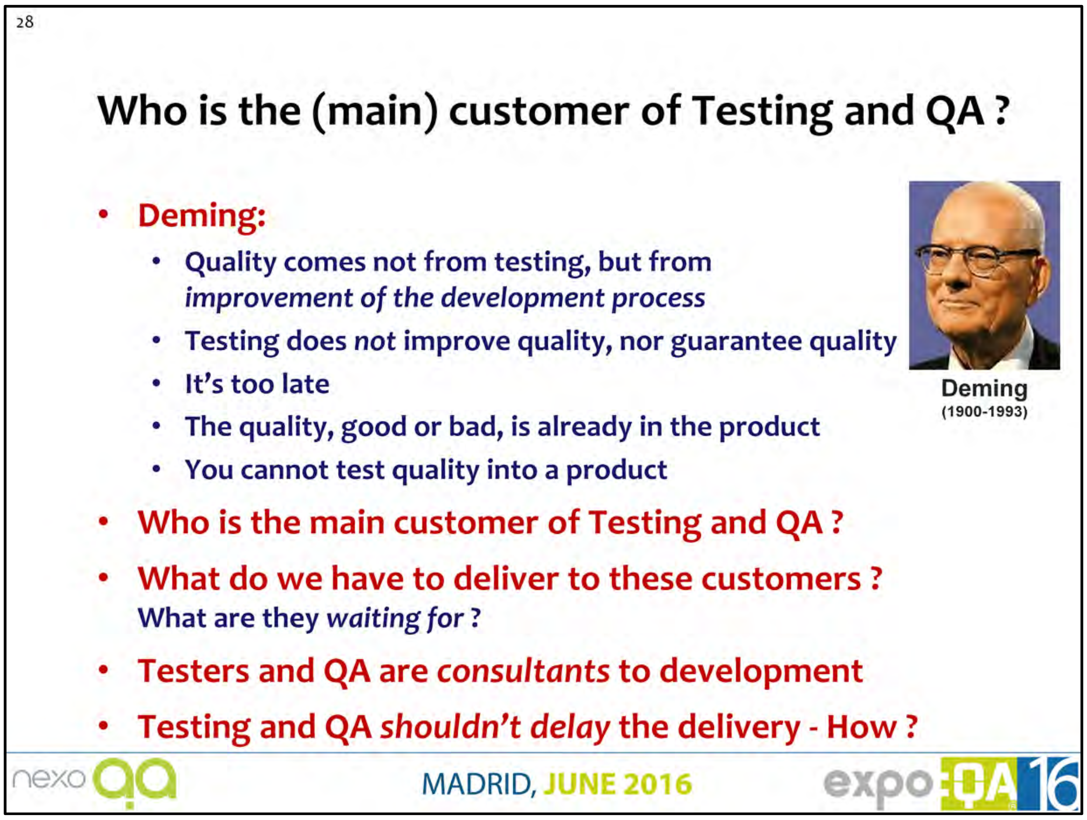
Lean Quality Assurance

- What is Lean ? (better read the source: Taiichi Ohno) or www.malotaux.nl/essenceoflean
- What is Quality ?
- How do you get Quality ?
- What is the required Quality level ?
- How do you measure Quality ?
- How to assure Quality ?
- What is Quality Assurance ?

nexo QA
MADRID, JUNE 2016
expo QA 16

I originally prepared this presentation (‘*Inspection used in various ways*’) as a guest lecture in Tom Gilb’s ‘*Lean Quality Assurance*’ course at BCS (British Computer Society). Therfore I had to shortly explain about Lean.

See www.malotaux.nl/essenceoflean



28

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

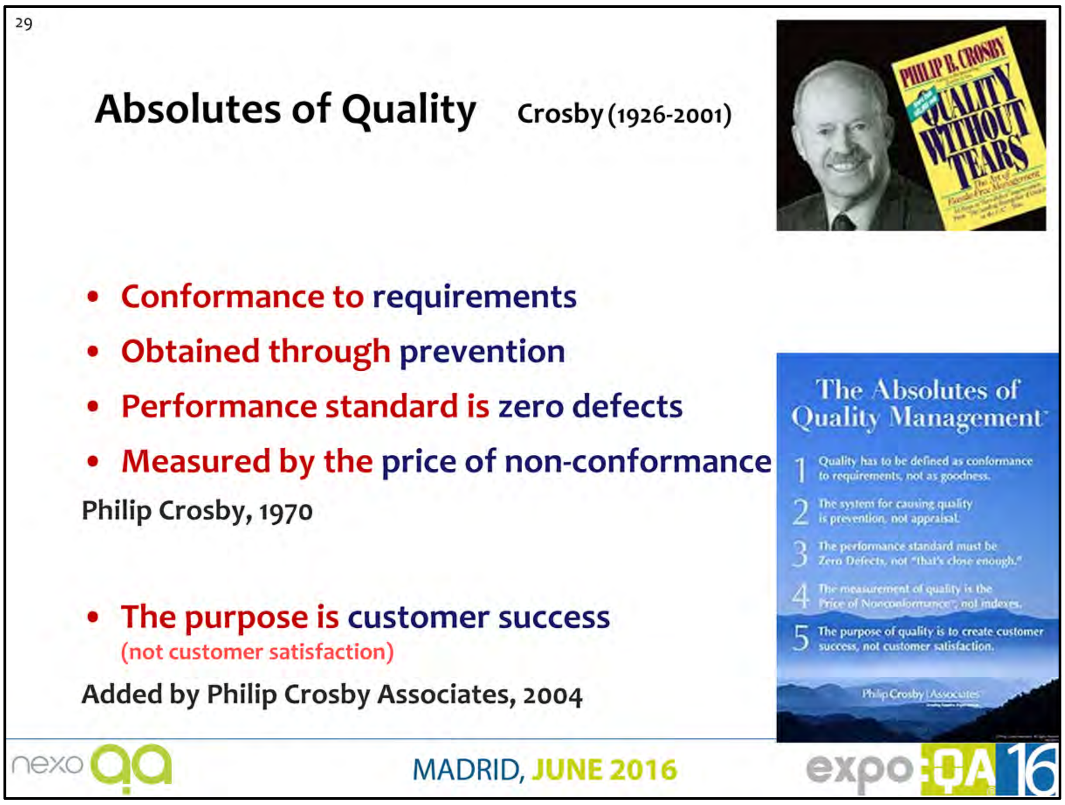
- **Deming:**
  - Quality comes not from testing, but from *improvement of the development process*
  - Testing does *not* improve quality, nor guarantee quality
  - It's too late
  - The quality, good or bad, is already in the product
  - You cannot test quality into a product
- **Who is the main customer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

Deming
(1900-1993)

nexo QA MADRID, JUNE 2016 expo QA 16

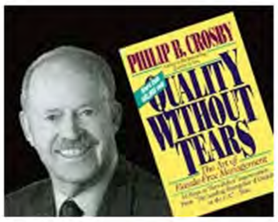
I experienced that to most testers this quote from Deming is quite a paradigm shift and usually comes as a shock. But usually it's a shock of recognition! It will change their attitude for the better forever.

Now let's see how we can optimize our contribution as consultants to development.



29

Absolutes of Quality Crosby (1926-2001)

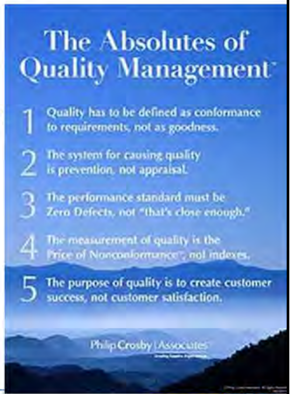


- Conformance to requirements
- Obtained through prevention
- Performance standard is zero defects
- Measured by the price of non-conformance

Philip Crosby, 1970

- The purpose is customer success
  (not customer satisfaction)

Added by Philip Crosby Associates, 2004



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.

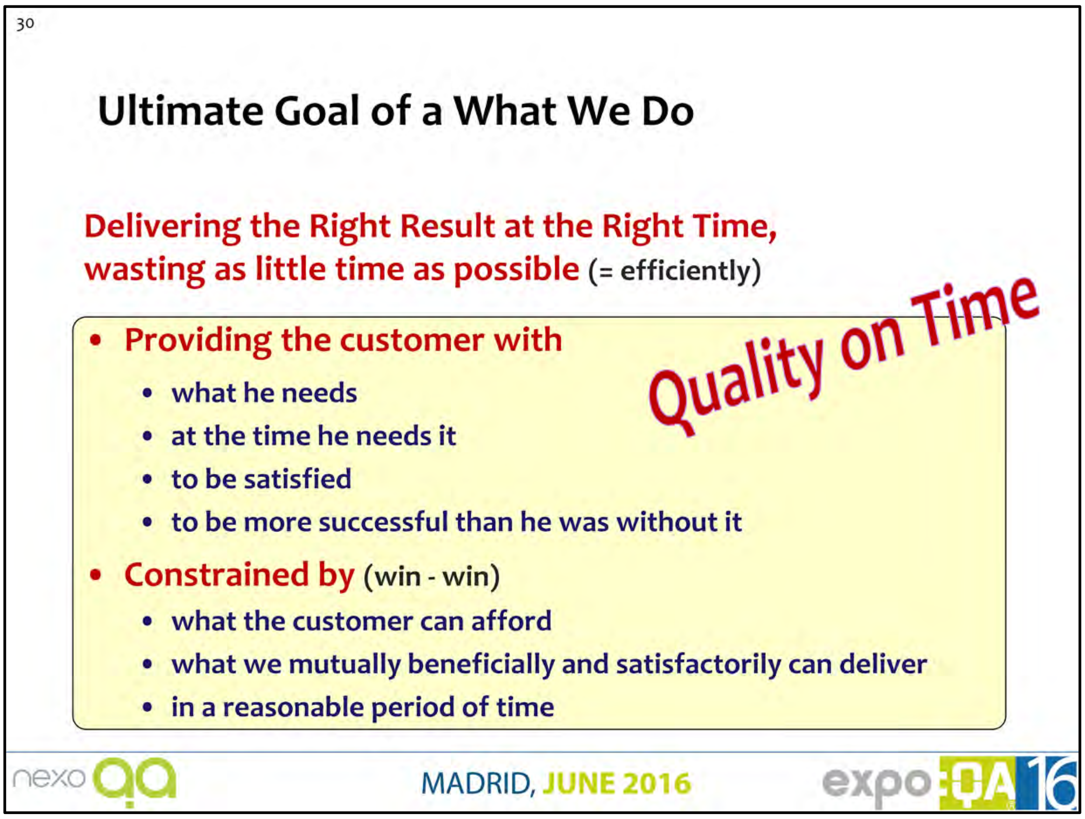
5 The purpose of quality is to create customer success, not customer satisfaction.

Philip Crosby | Associates

nexo QA MADRID, JUNE 2016 expo QA 16

Philip Crosby defined the four 'Absolutes of Quality'. When I started as a coach in a company recently, I gave his book "Quality without tears" to the CEO for homework: "Next week I'll check that you read it!". He did and it immediately had an impact on his behaviour. He delayed a major release to first get rid of the hassles that we were going to deliver to the costumers. He also calculated the 'Price of Non-Conformance' (PONC), to be at least a quarter of a million Euro in the past year.

Phil Crosby's organization later added a 5th Absolute: Customer Success. I agree completely. But I don't agree with them adding "... not customer satisfaction". After all, the customer should be successful, but satisfied as well, as we'll see on the next slide.



Ultimate Goal of a What We Do

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

Quality on Time

Inspections, if conducted properly by trained people, can help us to provide the customer even faster and better what he needs, when he needs it, to be satisfied, and more successful than without our product.

See www.malotaux.nl/goalofaproject

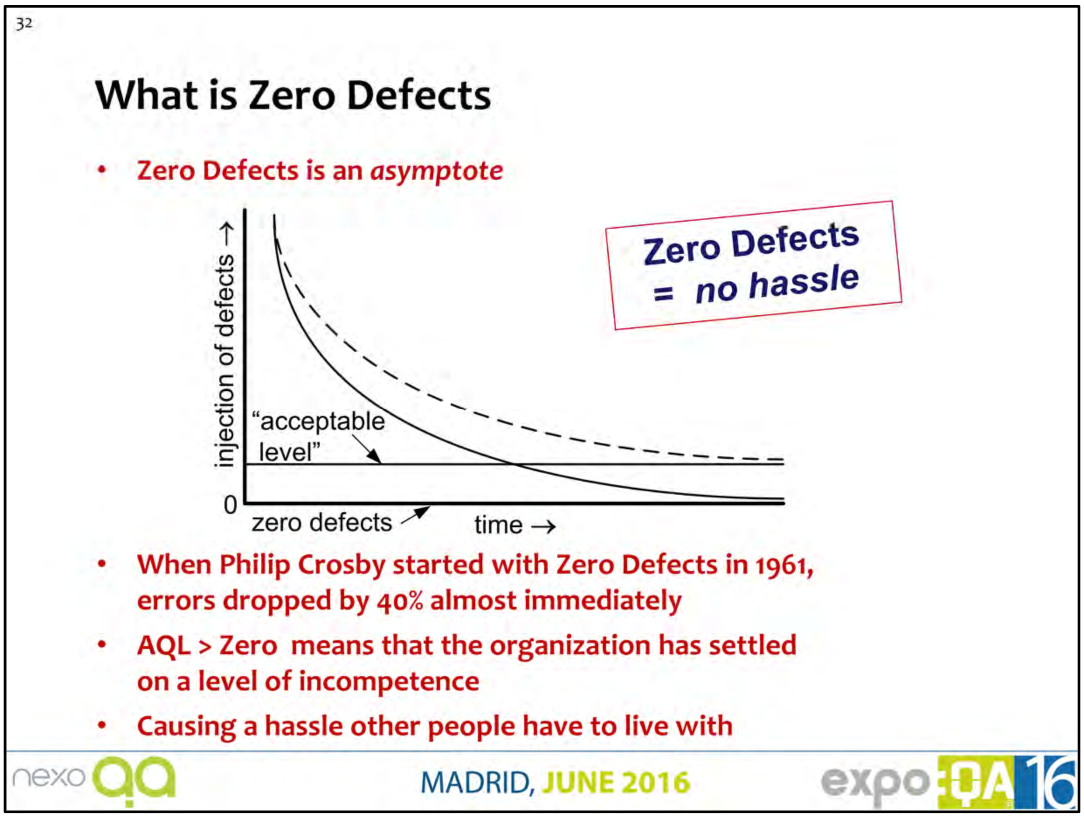
Conformance to requirements

- We meet the agreed requirements

or

- Have the requirements changed to what we and the customer really need
- We create requirements with care and we meet them with care
- Does you management take quality seriously ?

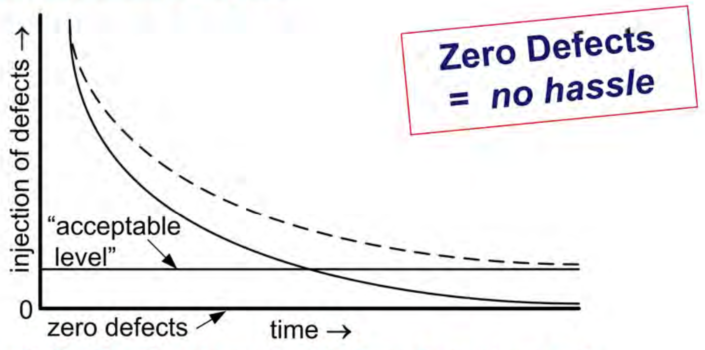
Phil Crosby



32

What is Zero Defects

- Zero Defects is an *asymptote*



- 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

MADRID, JUNE 2016

When I actively started using the Zero Defects (ZD) concept in software projects, defects made decreased by at least 50% almost immediately. It took about 2 weeks before the developers understood that I was dead serious about it. Then the testers came to me saying: "Niels, something weird is going on: we don't find issues anymore! It simply works!" I said: "Isn't that exactly what we want to see? Now testing is becoming a real challenge, namely proving that there are no errors."

So, even if you don't believe that this can be true, if two people (Crosby and me) did it and showed a huge decrease of errors *made*, only by adopting the attitude, isn't it at least worth a try, especially if you realize that about half of most projects is spent on finding and fixing defects. That's a huge budget. Any savings on that is probably well worth trying.

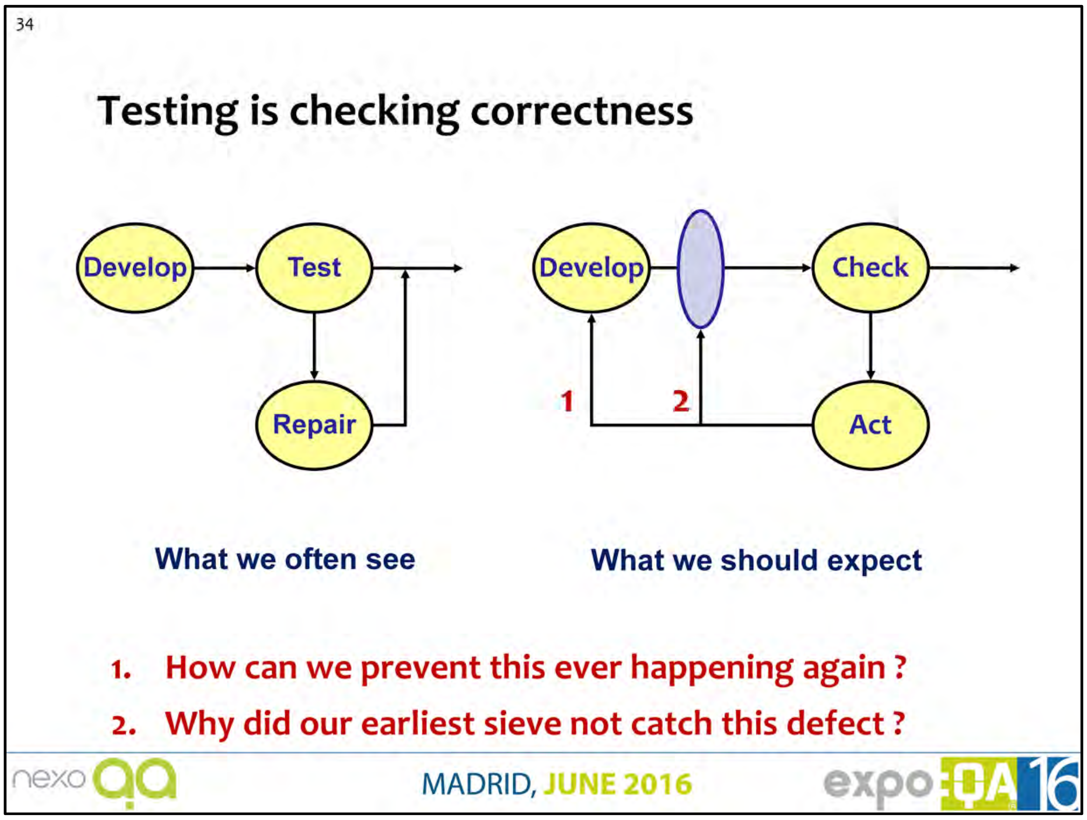
"No Hassle" proved to be easier to use than ZD: Don't cause a hassle. No hassle to yourself, to your peers, to your organization, to your customers.

.

Philip Crosby

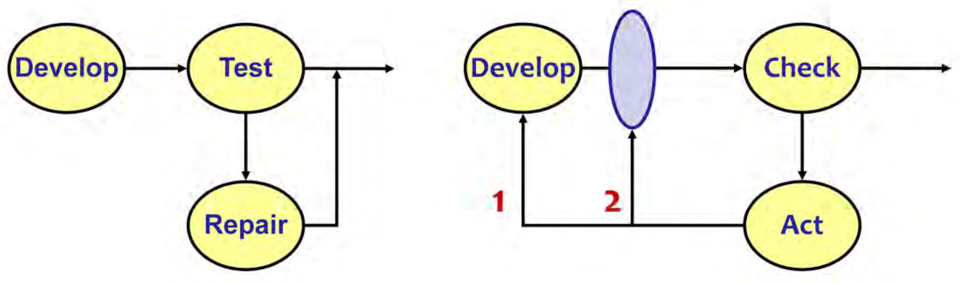
[Quality is Still Free]

- Conventional wisdom says that error is inevitable
- As long as the performance standard requires it, then this self-fulfilling prophecy will come true
- Most people will say: People are humans and humans make mistakes
- And people do make mistakes, particularly those who do not become upset when they happen
- Do people have a built-in defect ratio ?
- Mistakes are caused by two factors: lack of knowledge and lack of attention
- Lack of attention is an attitude problem



34

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 ?

nexo QA MADRID, JUNE 2016 expo QA 16

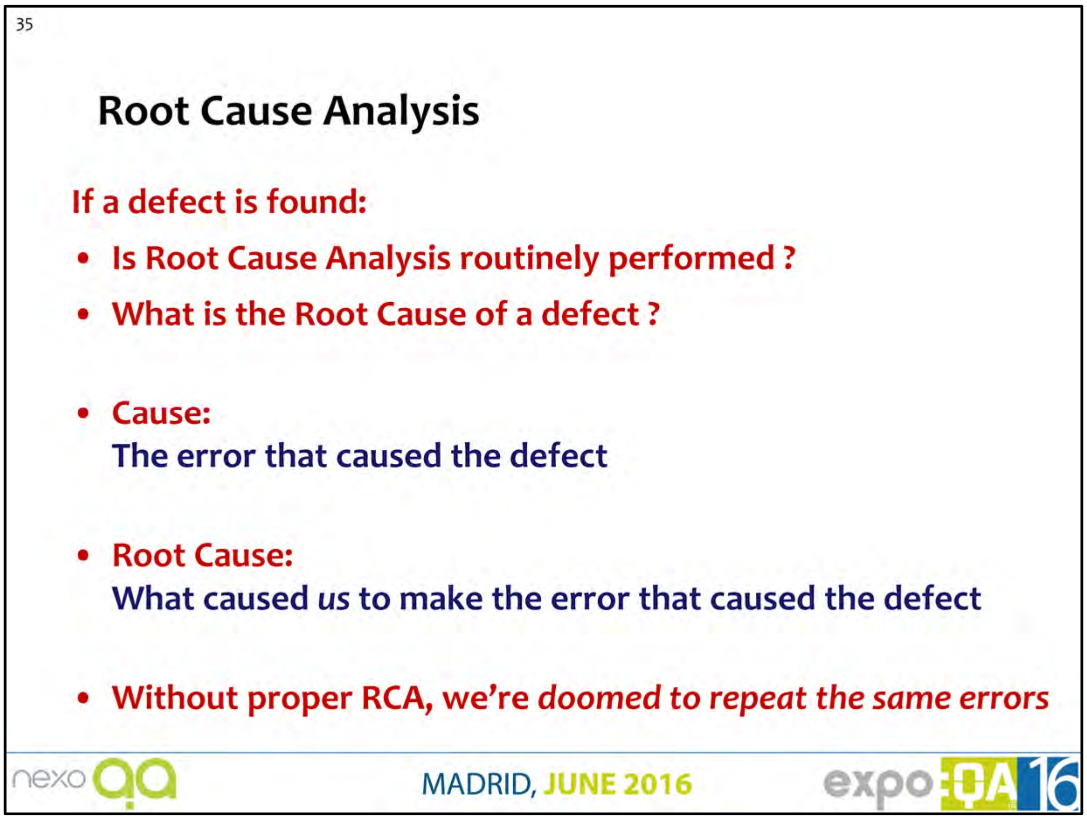
Testing should preferably just check correctness: the product does what it should do and does not what it shouldn't do.

- In practice, however, this is what we often see: repair of the defects testing happened to find.
  Note that it is absolutely impossible to check all paths through the software. And even if we could do that, we couldn't do it will all combinations of data. And, as testers are humans too, they will not find all issues, and find some issues that aren't issues.
  Phil Crosby once said:

  The error that isn't made, cannot be missed…

- What I suggest we should expect from the testing process is in the first place:
  - Feeding the prevention process
  - Improving the sieve that didn't catch the issue immediately after injection, in case prevention isn't perfect yet.

- Repair of issues found is a secondary effect. If the are 100 defects, of which the test-process finds 50%, and the developers repair 80% of those correctly, there are still 60 defects left. So what's the point?



35

Root Cause Analysis

If a defect is found:

- **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***

nexo QA — MADRID, JUNE 2016 — expo QA 16

Years ago I suggested to add a box for the 'Root Cause' and for the 'Root Cause Suggested Solution' in a bug-tracking system. When I later checked how people were using this, I found that in the Root Cause box they documented the cause of the bug and in the Root Cause Suggested Solution box the suggestion how to repair the bug.

Apparently, they didn't see the difference between 'Cause' and 'Root Cause':

- The Cause of a defect is the error that caused the defect
- The Root Cause is what caused ***us*** to make the error that caused the defect

In another project I asked the project manager what they do with the results of the code reviews. "People repair the bugs" he said. I asked: "Don't you do Root Cause Analysis, in order to learn how to prevent this type of error from now on?" The response was: "On every issue we found??? We have no time for that!"

Apparently they have no time to learn to prevent, and rather spend a lot of time to find and fix(?), over and over gain. No wonder that projects take more time than they hoped for.