



Vilnius - 17 October 2018

Keynote

Niels Malotaux

Examples of how to move towards Zero Defects

N R Malotaux - Consultancy
tel +49-5632 922 5132
mob +31-655 753 604
niels@malotaux.nl
www.malotaux.nl

Niels Malotaux

Examples of how to move towards Zero Defects

Niels Malotaux

Niels Malotaux is an independent Project Coach and expert in optimizing project performance. He has some 40 years of experience in designing electronic and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading a systems design company. Since 1998 he has devoted his expertise to helping projects and organizations to deliver Quality On Time: delivering what the customer needs, when they need it, to enable customer success. To this effect, Niels developed an approach for effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, Review and Inspection techniques, as well as Reliable Embedded Systems Design and how to achieve Zero Defects for the customer. Since 2001, he has taught and coached well over 400 projects in 40+ organizations in the Netherlands, Belgium, China, Germany, Ireland, India, Israel, Japan, Poland, Romania, Serbia, South Africa, the UK and the US, which has led to a wealth of experience in which approaches work better and which work less well in practice.

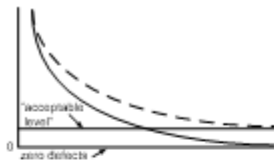
Niels puts development teams on the Quality On Time track and coaches them to stay there and deliver their quality systems on time, without overtime, without the need for excuses. Practical methods are developed, used, taught and continually optimized for:

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

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	Zum Anspel 5 59964 Dűdinghausen Germany tel +49-5632-922 5132 mob +31-655 753 604 niels@malotaux.nl www.malotaux.nl
Result Management	

Examples of how to move towards Zero Defects



Examples how to move towards Zero Defects

Niels Malotaux:
"In my experience the
'zero defects' attitude
results in 50% less defects
almost overnight."

Niels Malotaux

niels@malotaux.nl

www.malotaux.nl/conferences

A colleague found this remark in one of my tutorials: "In my experience the 'zero defects' attitude results in 50% less defects almost overnight" and asked me to explain this to a project team he was coaching. I thought that my experience with Zero Defects might be interesting for more people than just this team.

Niels Malotaux



- Independent Project and Organizational Coach
- Expert in helping optimizing performance
- 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 on track

Result Management

Malotaux - Zero Defects TestCon Vilnius 2018

2

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

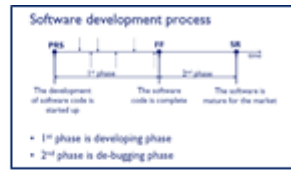
Philips Electronics – Application support for microcomputer systems design (1976-1980)

Malotaux - Electronic Systems Design - : developing electronic systems for clients products (1980-1998)

Now: **N R Malotaux - Consultancy**: coaching projects to deliver successfully and much faster (1998-)

Examples of how to move towards Zero Defects

Do we deliver Zero Defect software ?



- How many defects are acceptable ?
- Do the requirements specify a certain number of defects ?
- Do you check that the required number has been produced ?

In your projects

- How much time is spent putting defects in ?
- How much time is spent trying to find and fix them ?
- Do you sometimes get repeated issues ?
- How much time is spent on defect prevention ?

Better quality costs less

As many people think that even talking about ZD is useless, I'd like first to discuss some questions with the audience.

What is a defect ?

A defect is the cause of a problem
experienced by any of the stakeholders
while relying on our results

Short definition: A defect is the cause of a problem for the users

If we cause a problem by being late, it is a defect (by the above definition)

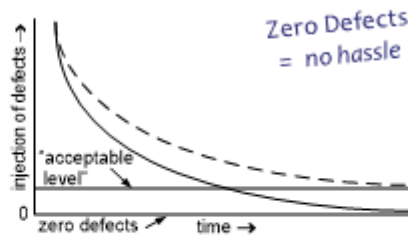
If the software isn't being used (over 50% of delivered software), the defects in that part of the software aren't defects according to this definition. The only defect is the fact that that part of the software was made in the first place.

This urges us to determine what software we are going to make that eventually won't be actually used, so that we can refrain from making it, saving a lot of time. Whether that's easy or not is beside the point.

Examples of how to move towards Zero Defects

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

Malotaux - Zero Defects TestCon Vilnius 2018

5

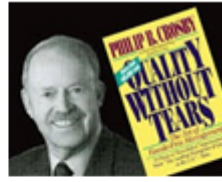
When I actively started using the Zero Defects (ZD) concept in software projects, defects made decreased by at least 50% almost immediately. It took about 2 weeks before the developers understood that I was dead serious about it. Then the testers came to me saying: "Niels, something weird is going on: 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.

Examples of how to move towards Zero Defects

Crosby (1926-2001) - Absolutes of Quality



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

Philip Crosby, 1970

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.

Examples of how to move towards Zero Defects

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

Malotaux - Zero Defects - TestCon Vilnius 2018

7

This is to me the top-level requirement for any project or any work we do.

- The customer is the entity that orders and pays. The customer, however, in many cases doesn't use the result of our project himself. He gets the benefit through the users of the result.
- What the customer says he wants is usually not what he really needs
- The time he needs it may be earlier or later than he says
- If the customer isn't satisfied, he doesn't want to pay
- If the customer isn't successful with what we deliver, he cannot pay
- If he's not more successful, why would he pay?
- What the customer wants, he cannot afford. If we try to satisfy all customer's wishes, we'll probably fail from the beginning. We can do great things, given unlimited time and money. But neither the customer nor we have unlimited time and money. Therefore: The requirements are what the Stakeholders require, but for a project: the requirements are what the project is planning to satisfy.
- The customer is king, but we aren't slaves. Both sides should benefit and be happy with the result.
- We will get the best result in the shortest possible time, but not shorter than possible. The impossible takes too much time.

Examples of how to move towards Zero Defects

Prevention: Root Cause Analysis

- Is Root Cause Analysis routinely performed – every time ?
- 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 Root Cause Analysis ,
we're doomed to repeat the same errors

Malotaux - Zero Defects TestCon Vilnius 2018

8

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(?). No wonder that projects take more time than they hoped for.

Examples of how to move towards Zero Defects

We're QA: What has this to do with us ?

- What is the goal of QA in a software development project ?
- Who is our customer ?

Many people equate QA with Testing. Testing, however, is just one of the quality measuring instruments of QA and hence only a small part of QA. So, let's shortly discuss what QA actually is and who the customer of QA is.

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



Deming
(1900-1993)

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.

Some Examples

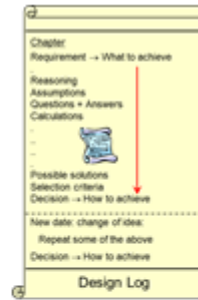
We're not perfect,
but the customer shouldn't find out

Let's discuss some examples of techniques that helped me and others to move towards Zero Defect deliveries.

Examples of how to move towards Zero Defects

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)
 - What happens if you ask "Can I see the DesignLog ?"



Malotaux - Zero Defects TestCon Vilnius 2018

There are many techniques known to approach ZD faster. One of them is what I call 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 usually only shows what people decided to implement, 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, in the maintenance phase (often a the largest portion of the cost of deployment of software!) minimum time is lost. One of the requirements for the DesignLog is: "If someone has to change something in the software one year later, he should be up and running within one or at most two days."

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

Examples of how to move towards Zero Defects

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

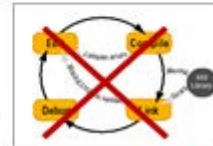
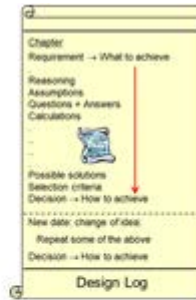
"We don't have time!" - "We've only 7 days!"

James insisted

We designed, found the problem, corrected it, cleaned up the mess

Done in less than 7 days

Thank you!



Malotaux - Zero Defects TestCon Vilnius 2018

13

This happened some time ago. 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.

Examples of how to move towards Zero Defects

What James told me afterwards

- 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
 - The one that was design and code reviewed had no issues after deployment
 - The other one was the source of quite some defects
- In summary, this success has proved instrumental in buy-in for DesignLogs which are now embedded in the development process

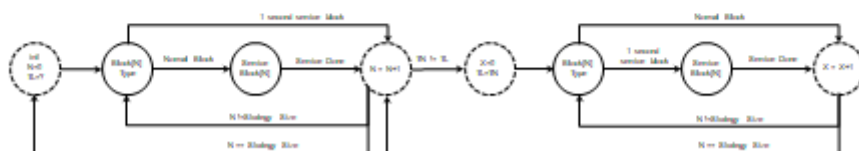
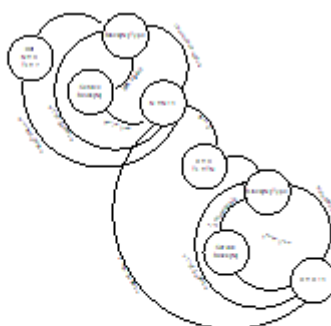
James explained me later more interesting details of this case:

- 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.

Examples of how to move towards Zero Defects

There are many ways to represent a design

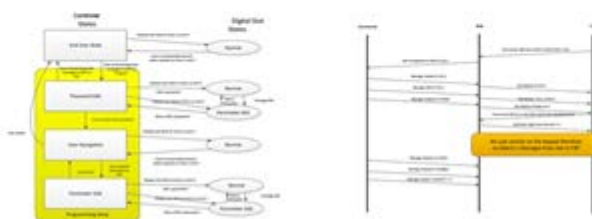
- Only few are useful
- Don't waste reviewer's time



There are many ways to represent a design. Only few are useful.



Useful design ?



A 'Systems Engineer' made a 'requirements document' for communication between a user interface device (made in Switzerland) and a climate controller (made in another country) for hotel rooms. The user interface also had to be used by the installers to tune the climate controller equipment. He made a 47 page document with a lot of cut-and-paste, which, of course, caused a lot of errors. After a few pages, nobody kept reading, both in Switzerland and the other country. When I read the interface documents from both sides, it was clear to me that when these devices would be connected to each other, the system wouldn't work. So I condensed the design into one picture, which I checked with the engineers at both sides.

Examples of how to move towards Zero Defects

Choose the appropriate design

47 pages documentation condensed into one page

Malotaux - Zero Defects TestCon Vilnius 2018 17

If I see documentation at all, it is usually just text. Sometimes a lot of text. One of my mantra's is: "Where are the pictures?"

This and the next slide are an example of some design I made recently (anonymised). You don't have to check the text and what it actually does. It's just to show some examples of concisely documenting functionality in a way most people, with a bit of understanding, can follow immediately.

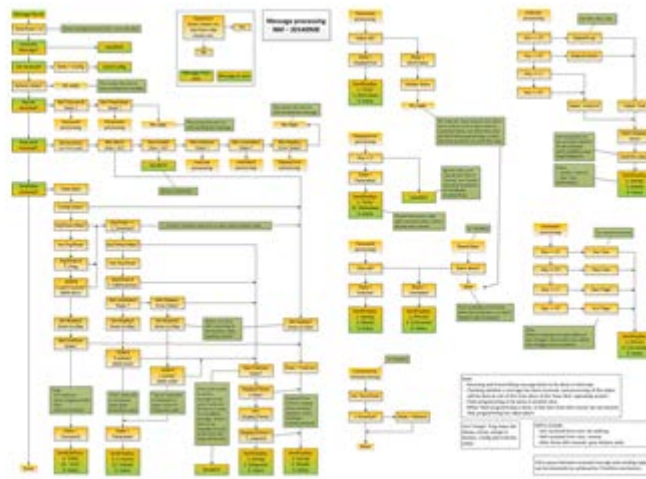
This slide shows a design of the communication between some controller and a remote user interface. It was documented in a 47 page document by a 'Systems Engineer'. 47 pages of interface description is almost impossible to oversee by humans, hence it contained a lot of inconsistencies and the people who had to implement it actually ignored it.

Once I made this one page overview, we could discuss, ease out the inconsistencies, make decisions, agree, and everyone knew exactly what to do. Conclusion: just documenting isn't enough. We have to learn how to document for usefulness.

QA can ask a developer to explain how the interface should work. If the developer only shows code to review, we know we have a problem. If the QA person doesn't understand the explanation, the explanation apparently isn't clear enough, which is a big risk for the quality of the result. If it's only text, it won't work either.

Examples of how to move towards Zero Defects

How
could it
look like ?



Malotaux - Zero Defects TestCon Vilnius 2018

18

This is how I implemented the communication design based on my discussions with the suppliers of the remote user interface. The design was made to be reviewed and then it could readily be implemented based on this design. If I see how much I moved and reshuffled before I was content that this was right, I cannot imagine how this could be done properly in code without having this design. Like in the Cleanroom Approach to Software Development I designed down to a level of some 3 lines of code per design element. Sorry, I have no time now to go into detail, but the Cleanroom Approach routinely delivered an order of magnitude less defects in shorter time. Making changes in the code is not allowed before we have updated the design. The code should always be derived from the current design. Reviews of code should always check that the code does what the design says

These were just examples. The challenge is every time again to find the right representation that is easiest to comprehend

Of course the projects the audience is working in usually do these things properly. But I still see too often that the 'design' is only in the mind of the developer who writes the code, or just a rough sketch, with devastating effects in software quality and delivery time.

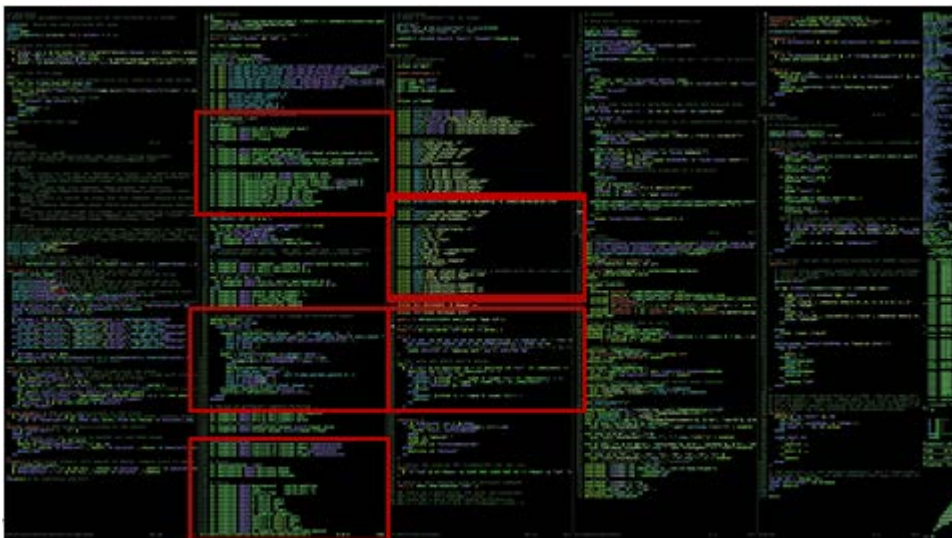
If as a QA person you encounter these effects, think what you should do about it.

Examples of how to move towards Zero Defects

What is better than reviewing code ?



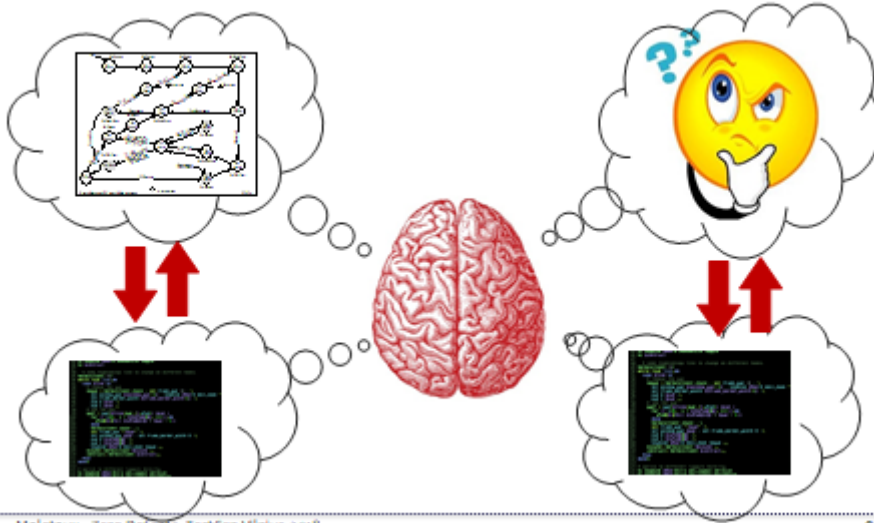
- Do you ever review software ?
- What do you review ?
- What is better than reviewing code ?
 - May I review the design first ?



Can you really see structure in code?

If I observe software people, I usually see them staring to code on their screen, trying to keep the model of what is happening in their mind.

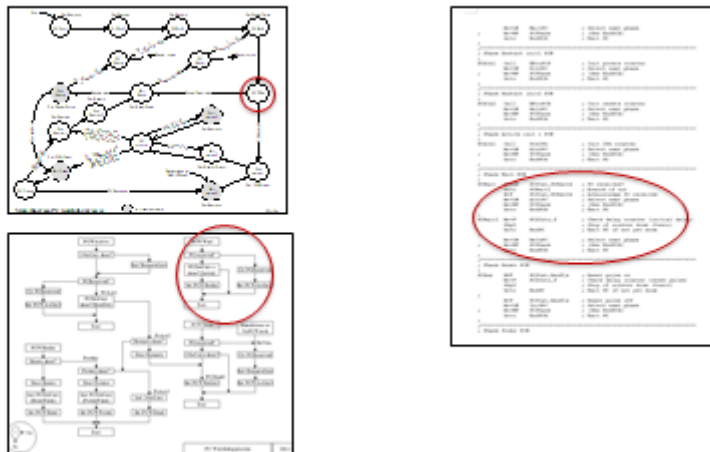
Examples of how to move towards Zero Defects



Malotaux - Zero Defects TestCon Vilnius 2018

21

If we take the model out of our brain on paper, we have our whole brain at our disposal to think what to improve.



Malotaux - Zero Defects TestCon Vilnius 2018

22

Simple example of correlation between a state diagram, the derived flows, and the resulting code.

Can you see what's happening in the code? I can't.

Can you see what's happening in the flow diagram? A bit better.

Can you see coherence in the state diagram? I can. And I easily can reason with this with anyone, even the CEO.

It took me quite some time to compose the state diagram. But then the flows and the code were done quite straight-forward and had no errors.

The product worked for years without any problem.

It also adheres the requirement: "After a year I should be able to add or change anything in this system without introducing any issues, being up and running within a day".

Examples of how to move towards Zero Defects

Case: Scrum Sprint Planning

- What is the measure of success for the coming sprint ?
- “What a strange question !
We're Agile, so we deliver working software. Don't you know ?”
- Note: Users are not waiting for software:
they just need *improved performance* of what they're doing
- How about a requirement for 'Demo': No Questions – No Issues
- How's that possible !!!
- They actually succeeded!

I came in an project of some 70 people, with 3 Scrum teams of some 12 people each. We know 12 is too many, but that's another story.

At a Sprint Planning meeting I asked one of the teams: "What would be the measure of success for this Sprint?"

They looked at me: "What a strange question. We're Agile, so we deliver working software. Don't you know?"

I asked: "Shouldn't we have a measure of success, to know that we really did a good job?" and suggested:

"No questions, No Issues". That's easy to measure: one question or one issue and we know we failed. No question and no issue and we know we were successful.

Their first reaction was: That's impossible! Surely there will be some questions when we deliver and there are always some issues.

I suggested: "You find out how to do it. It's just a simple requirement: "No questions, No Issues".

Interestingly, they immediately started thinking how they could deliver according to this requirement.

For example, someone thought: "Ah. Perhaps halfway the Sprint we ask someone to check it out and to see whether he would have any questions?" I said: "You're on the right track. Just find out how to do it. The requirement is simple."

Actually, I didn't expect them to be successful in this first Sprint, perhaps after a few. Surprisingly, they were successful. I'll tell how.

Examples of how to move towards Zero Defects

Demo ??

- Give the delivery to the stakeholders
- Zip your mouth
- Keep your hands handcuffed on your back
- and o-b-s-e-r-v-e what happens
- Seeing what the stakeholders actually do provides real feedback
- Then we can 'talk business' with the stakeholders

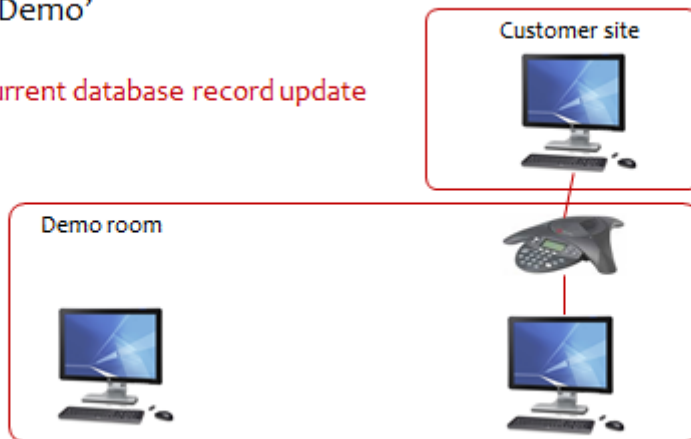
- Is this what you do ?



Scrum reviews usually only demo.
How about using the advice as shown here.
Is this what you do?
If not, why not?

The 'Demo'

Concurrent database record update



Examples of how to move towards Zero Defects

Delivery Strategy Suggestions (Requirements)

- What we deliver will be used by the appropriate users immediately, within one week not making them less efficient than before
- If a delivery isn't used immediately, we analyse and close the gap so that it will start being used (otherwise we don't get feedback)
- The proof of the pudding is when it's eaten and found tasty, by them, not by us
- The users determine success and whether they want to pay (we don't have to tell them, but it should be our attitude)

Malotaux - Zero Defects TestCon Vilnius 2018

26

This is my suggestion I gave the architect and the project manager to consider as Delivery Strategy. It took several months before they dared to use this advice.

Case: How much legwork is being done in your project ?

- Requirements/specifications were trashed out with product management
- Technical analysis was done and
- Detail design for the first delivery



At the first delivery:

- James: *How is the delivery? (quality versus expectation)*
- Adrian: *It's exactly as expected, which is absolutely unprecedented for a first delivery*
The initial legwork has really paid off

Malotaux - Zero Defects TestCon Vilnius 2018

27

This case was an organization with extraordinary bright people. In many projects we have to explain things over and over again, but in this project people needed only half a word to understand and do things better. James (their new QA person) told me this story. He asked them to prepare well, design properly, and then do the coding. The result: *"It's exactly as expected, which is absolutely unprecedented for a first delivery."* He suggested it, they did it, and it worked. It's great for a QA person to work in such a fertile environment!

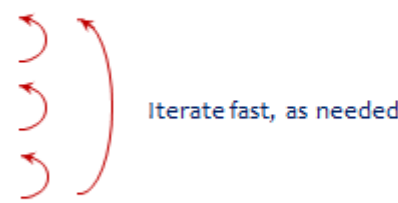
Examples of how to move towards Zero Defects

Some techniques shown

- Design
- Drawings
- DesignLog
- Review
- No Questions – No Issues

A Zero Defects attitude makes an immediate difference

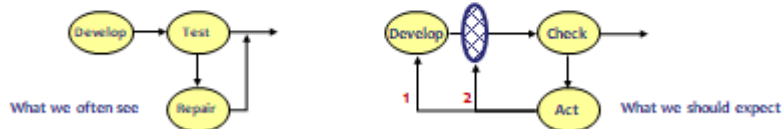
Basic approach

- Design the requirement
 - Review
 - Design implementation
 - Review
 - Implement (code ?)
 - Review
 - Test doesn't find issues (because they're not there)
- 
- Iterate fast, as needed

Examples of how to move towards Zero Defects

What's in it for QA ?

- Did we see much testing in the previous ?
- Testing shouldn't find anything (because there should be no issues)
- Did you ever find similar issues as you found before?
 - First time: Developers 'fault'
 - Second time: Testers 'fault'



- QA to help developers to produce less and less defects

Do we deliver Zero Defect software ?

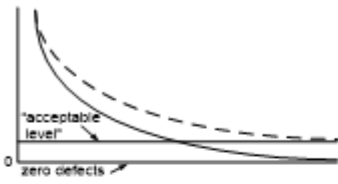
Better quality costs less

- How many defects are acceptable ?
- Do the requirements specify a certain number of defects ?
- Do you check that the required number has been produced ?

In your projects

- How much time is spent putting defects in ?
- How much time is spent trying to find and fix them ?
- Do you sometimes get repeated issues ?
- How much time is spent on defect prevention ?
- Could you use "No Questions – No Issues" ?

Examples of how to move towards Zero Defects



Approaching Zero Defects is Absolutely Possible

If in doubt, let's talk about it

Niels Malotaux

niels@malotaux.nl

www.malotaux.nl/conferences

If in doubt, let's discuss.
To you it may be theory.
To me it's reality.