How to move towards Zero Defects

“acceptable level”

zero defects

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

- Independent Project and Organizational Coach
- Expert in helping projects and organizations to quickly become
  - More effective - doing the right things better
  - More efficient - doing the right things better in less time
  - More predictable - delivering as needed
- Getting projects back on track
- Embedded Systems architect (electronics/firmware)
- Project types
  electronic products, firmware, software, space, road, rail, telecom, industrial control, parking system
Do we deliver Zero Defect software?

- How many defects are acceptable?
- Apparently 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?
What is a defect?

A defect is the cause of a problem experienced by any of the stakeholders while relying on our results.
Is Zero Defects possible?

- 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

We aren’t perfect, but the customer shouldn't find out

Zero Defects = no hassle

Hassle: problem, unnecessary difficulty
Prevention: Root Cause Analysis

- Is Root Cause Analysis routinely performed - every time?
- What is the difference between Cause and Root Cause?

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

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

- Without proper Root Cause Analysis, we’re **doomed to repeat the same errors**
Does Testing ‘assure’ quality?

- Some testers call themselves QA: ‘Quality Assurance’
- Can testers assure quality?  
  - yes  
  - no
- 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
- So, how to create quality?
Some Examples

We’re not perfect, but the customer shouldn’t find out
Design techniques

- Requirement
- Review
- Design
- Review
- Code
- Review

- Integration 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?"

Cleanroom
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!
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
  - One that was design and code reviewed had no issues after deployment
  - Other one, was the source of quite some defects
- This success proved instrumental in buy-in for DesignLogs which are now embedded in the development process
There are many ways to represent a design

- Only few are useful
- Don't waste reviewer's time
Useful design?
Choose the appropriate design

47 pages documentation condensed into one page
What is better than reviewing code?

- If you review software, what do you review?
- What is better than reviewing code?
  - May I review the design first?
State diagram PC watchdog process

- **Init**: Initial state, no immediate action.
- **PC**: Check if PC is active.
- **Active**: PC is active, continue.
- **Dead**: PC is dead, enterPause.
- **Init Dead**: PC is dead, enterPause.
- **PC Wait**: Wait for PC to be active.
- **Reset**: Reset PC.
- **Power**: Power management state.
- **Initial Delay done**: Initial delay completed.
- **Reset pulse done**: Reset pulse completed.
- **Power pulse done**: Power pulse completed.
- **Not PC Received**: PC not received.
- **Restart Delay done**: Restart delay completed.
- **Restarts not done**: Restarts not completed.
- **24 hr Delay done**: 24-hour delay completed.
- **PC Received**: PC received.
- **Set Restart Delay**: Set restart delay.
- **Set Initial Delay**: Set initial delay.
- **Set Resets**: Set resets.
- **Set Powers**: Set powers.
- **Set Restarts**: Set restarts.
- **Restart Delay done**: Restart delay completed.
- **Restarts done**: Restarts completed.
- **Powers done**: Powers completed.
- **Resets done**: Resets completed.
- **Resets not done**: Resets not completed.

**Notes**:
- StandAlone or NoPCWatch
- = transient state

**Graph**:
- Diagram showing state transitions and conditions.

**References**:
- Malotaux - BCS - Zero Defects 2022
State diagram PC watchdog process

PCWait
Init
Reset
Active2
Init Dead
PC
Reset
PC
Power
InitialDelay done
Init
Power
State diagram PC watchdog process

Reset pulse done
Power pulse done
Not PCReceived and RestartDelay done and Restarts not done
24hrDelay done
PCReceived
Powers done
Not PCReceived and
RestartDelay done
and Restarts done
PCReceived
PCReceived
PCInit1
PCInit2
PCInit3
PCInit4
Set Restarts
Init
Restart2
Set Resets

PC Init State

Malotaux - BCS - Zero Defects 2022
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!
• 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?
The ‘Demo’

Concurrent database record update

No questions – no issues!
Delivery Strategy Suggestions

• 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).

• Would you dare to deliver no-cure-no-pay?
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
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
Do we deliver Zero Defect products?

- How many defects do you think 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”?
Approaching Zero Defects is Absolutely Possible

If in doubt, let's talk about it

Niels Malotaux

Copyright © 2022 by Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/?id=zerodefects
Inquiries: niels@malotaux.eu
www.malotaux.eu/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, except for the step-by-step TimeLine procedure

6 Human Behaviour in Projects (APCOSE 2008)
   Human Behavioural 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)
   Use of TimeLine technique: How we solved a 6 month backlog in 9 weeks

9 Predictable Projects (2012) - How to deliver the Right Results at the Right Time

RS Measurable Value with Agile (Ryan Shriver - 2009)
   Use of Evo Requirements and Prioritizing principles

www.malotaux.eu/inspections
   Inspection pages