Optimizing the Contribution of Testing to Project Success

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 **niels@malotaux.nl** **www.malotaux.nl/nrm/English**

The Goal

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

The Problem

- Still too many defects experienced by users

Apparently

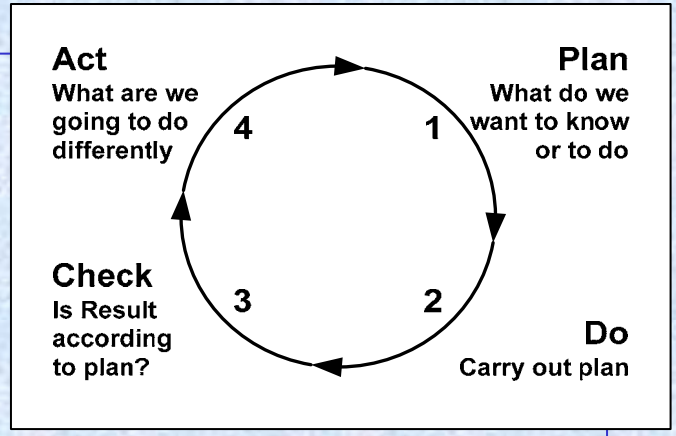
- Still too many defects generated by developers
- Still too many defects remain undiscovered

- There is a lot of knowledge how to reduce the generation and proliferation of defects

There is a large budget to do something about it:

- Some 50% of project time is consumed by all kinds of testing
- About 50% of developed software is never used
- About 50% of delivered software is never used

Knowledge
how to achieve the goal



- Using very short Plan-Do-Check-Act cycles
- Constantly selecting the most important things to do

then we can

doing the right things

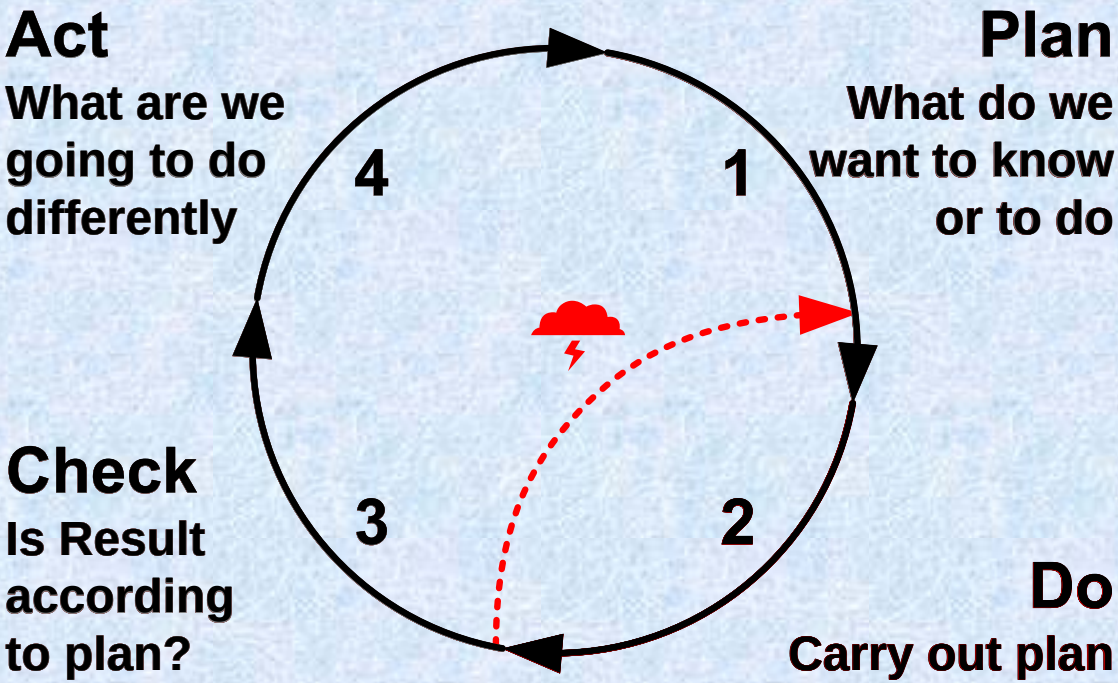
- Most quickly learn what the real requirements are
- Learn how to most effectively and efficiently realize these requirements

and we can

doing the right things right

- Spot problems quicker, allowing more time to do something about them

The PDCA cycle



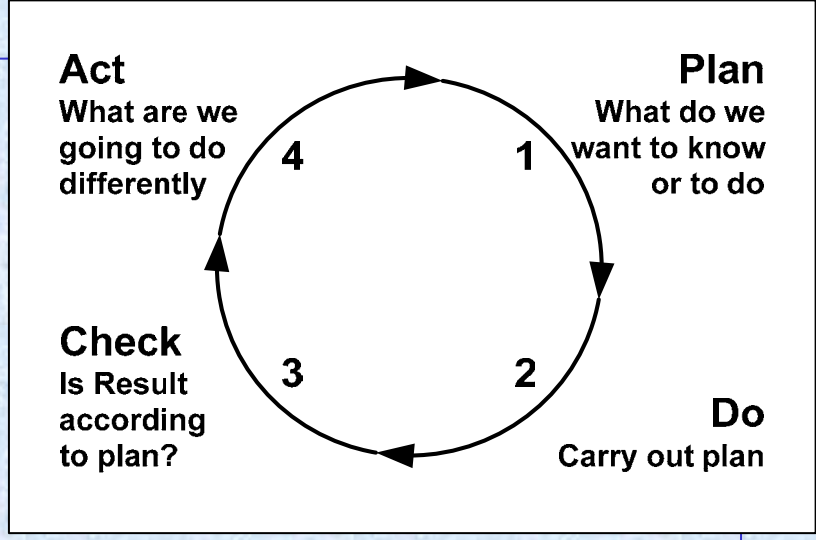
More knowledge

- **Preventing defects is better than trying to find them**

however

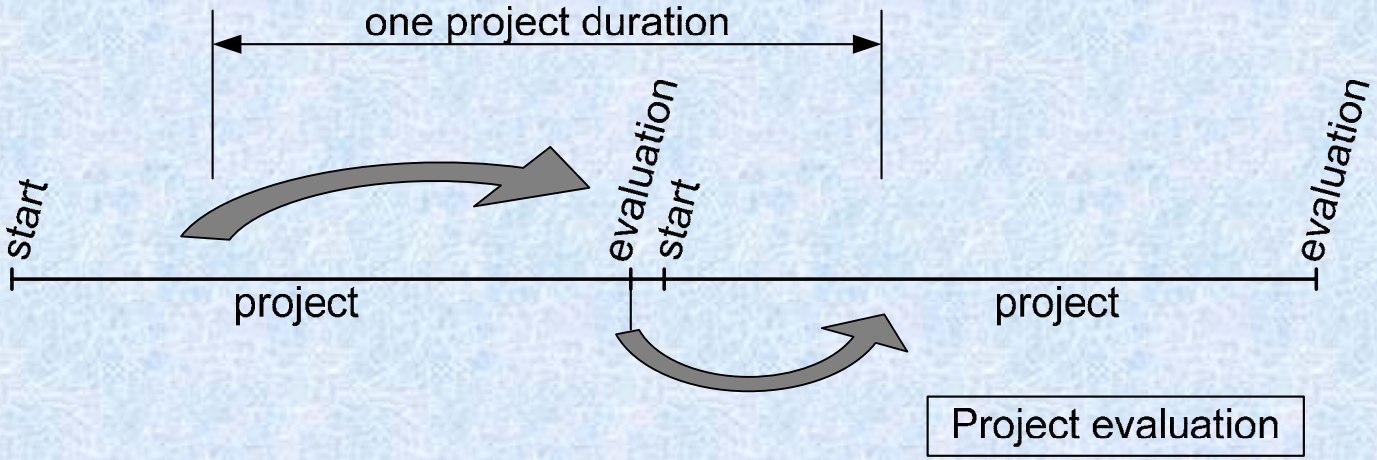
- **Prevention requires a specific attitude that generally does not come naturally**
- **It requires more than doing our best**

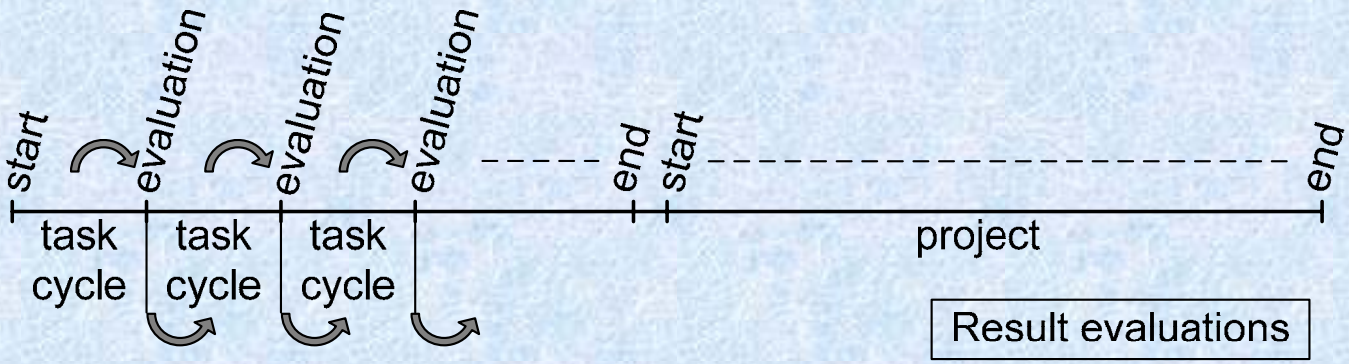
Evo



- Evo (short for Evolutionary...) uses this knowledge to the full
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- Applying the PDCA-cycle
  actively, deliberately, rapidly and frequently,
  for *Product*, *Project* and *Process*, based on ROI
- A desire to Learning how to be better
- Projects seriously applying Evo, routinely conclude successfully on time, or earlier, *by design*
- Proactively anticipating problems before they occur, working to prevent them

Project evaluations





All we have to do ...

- A defect is the cause of a problem experienced by any of the stakeholders while relying on our results
- Making the customer more successful implies no defects
- All we have to do is delivering results without defects
- Do we?

- Is being late a defect?

The process of defect injection

Conventional software development:

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

Can't we do better?

Bugs are so important, are they really?

- "Software without bugs is impossible"
- Bugs are counted
- We try to predict the number of bugs we will find
- It is suspect if we don't find the expected number
- Bugs are normal
- What would we do if there were no bugs any more?

As long as we keep putting bugs in the center of the testing focus, there will be bugs

Let's move

Let's move from

- Fixation to Fix

to

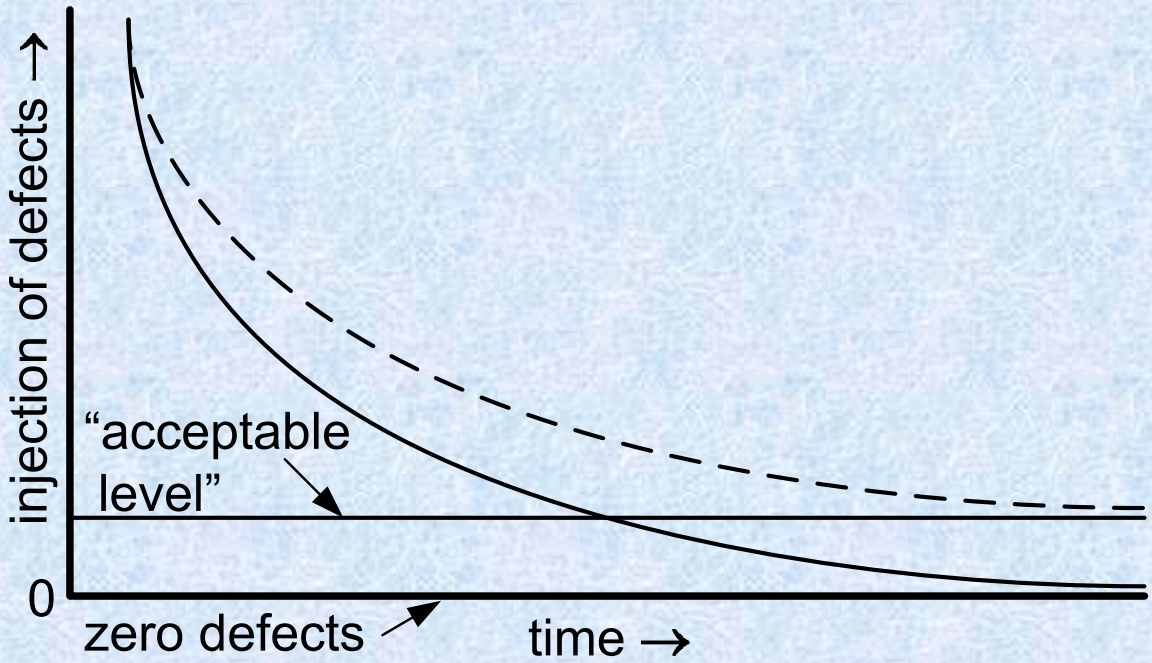
- Attention to Prevention

- If we don't deal with the root, we will keep making the same mistakes over and over

- Without feedback, we won't even know
- With quick feedback, we can put the repetition to a halt

Is defect free software possible?

- **Zero Defects is an asymptote**



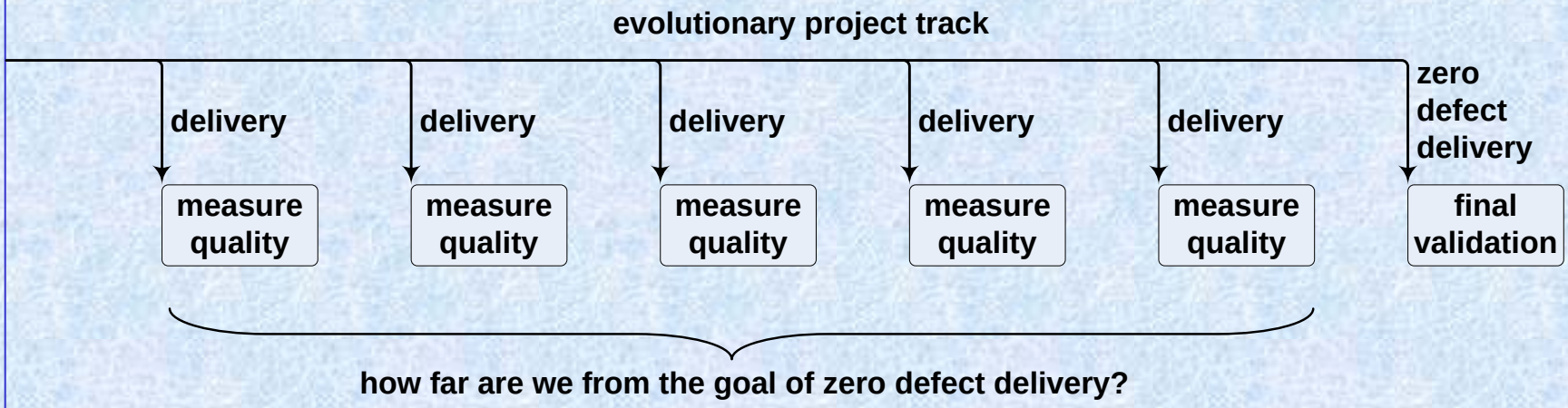
- **When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately**

Attitude

- As long as we think defect free software is impossible, we will keep producing defects

- From now on, we don't want to make mistakes any more
- We feel the failure (if we don't feel failure, we don't learn)
- If we deliver a result, we are sure it is OK and we are surprised when there proves to be a defect after all
- We do what we can to improve (continuous PDCA)

Current Evo Testing



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

Further Improvement

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

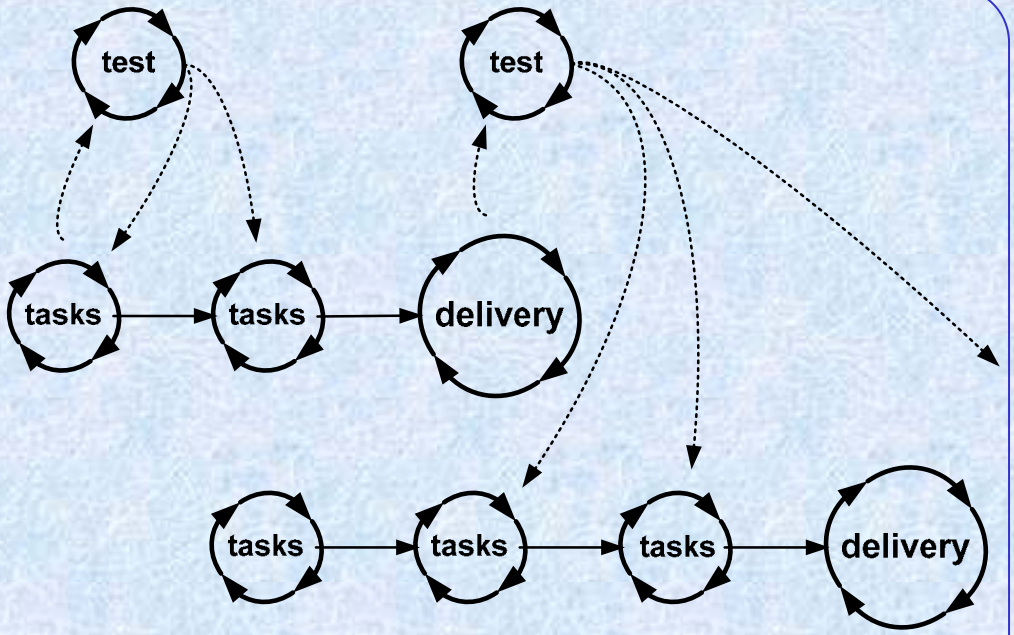
Cycles in Evo

- TaskCycle
- DeliveryCycle
- TimeLine

During these Cycles we are constantly optimizing

- The product
- The project
- The process

Evo cycles for Testing



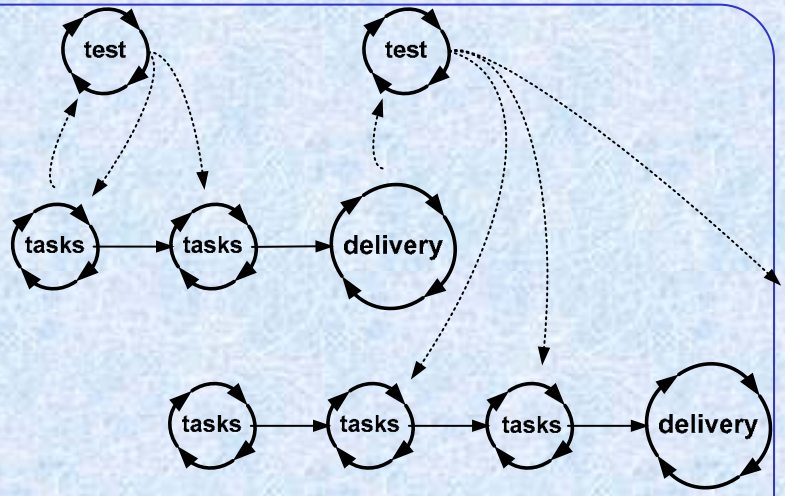
- Testers organize their work in weekly TaskCycles
- DeliveryCycle is the Test-Feedback cycle
- Testers use their own TimeLine, synchronized with the developers TimeLine
- Testers conclude their work in sync with developers
- Testers check work in progress *even before* it is finished

Metrics

Don't *improve* non-value-adding activities - better *eliminate* them

- **Estimation - planning - tracking**
  - **If estimation is a TimeBox, tracking is a "zero activity"**
- ***Defects per kLoC* or *Defects per Page***
  Stop counting defects, it conveys a bad message. Decrease numbers *by design.*
- ***Incoming defects* per month (by test, by user)**
  Don't count. Do something. Users shouldn't experience problems.
- ***Defect detection effectiveness* or *Inspection yield***
  - **Yield is 30% ~ 80%; testers are human after all**
  - **Zero defects at user means zero defects before final test**
  - **Whether that is difficult is beside the point**

More metrics



- *Cost to find and fix a defect*
  - The less defects the higher the cost per defect
  - This was a bad metric anyway
- *Closed defects per month*
  - Closing depends on prioritizing process, through Candidate Tasks List
- *Age of open customer found defects*
  - Purpose of many metrics seems to be *policing*: not trusting people to take appropriate action
  - In Evo we take appropriate action
- *Remaining defects*
  - Still useful as measure of Prevention success

When are we done with testing?

- **Conventional:**
  - Number of bugs found per day less than n
  - Defect backlog decreased to zero
  - Prediction by curve fitting based on early found defect numbers
  - Using historical data
  - Other?
- **Evo:**
  - The project is ready at the agreed date, or earlier
  - That includes testing

Useful Evo metric

- **Size of the smile on the customers face**
- **In many cases, the Evo attitude and techniques replace the need for metrics**
- **I did not say always**

Dijkstra (1972)

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

However:

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

- **Conventional testing is pursuing the very effective way to show the presence of bugs**
- **The challenge is, however, to solve the hopeless inadequacy of showing their absence**
- **And working towards their absence**

Links

- **http://www.gilb.com**
  Tom Gilb's website: Evo guru
- **http://www.malotaux.nl/nrm/English**
  Niels' activities: Evo evangelist
- **http://www.malotaux.nl/nrm/Evo**
  Evo pages
- **http://www.malotaux.nl/nrm/pdf/MxEvo.pdf**
  Evolutionary Project Management Methods
  (issues and 2001 experience)
- **http://www.malotaux.nl/nrm/pdf/Booklet2.pdf**
  How Quality is Assured by Evolutionary Methods
  (more recent practical implementation experience)
- **http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf**
  Optimizing the Contribution of Testing to Project Success
- **http://www.malotaux.nl/nrm/Evo/ETAF.htm**
  Download the Evo Task Administrator (ETA) tool
  (expects MSAccess2000~2003)

Optimizing the Contribution of Testing to Project Success

Questions?

Niels Malotaux

N R Malotaux
Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl/nrm/English