Quality Comes Not From Testing

but from improving the development process

Niels Malotaux
Project Coach

N R Malotaux
Consultancy

questions or comments:
+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

Niels Malotaux

- **Project Coach**
  - Evolutionary Project Management (Evo)
  - Requirements Engineering
  - Reviews and Inspections
  - Dependability (Systems that simply work)

Result Management

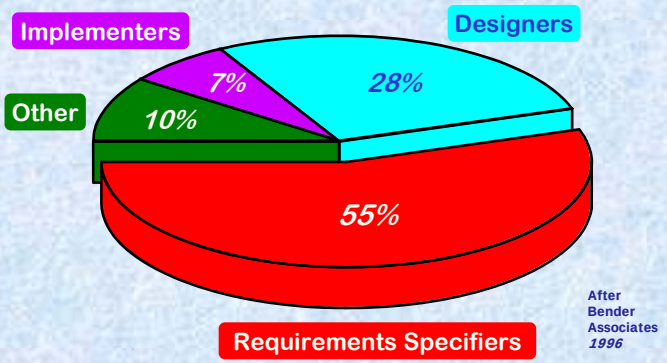
Helping projects and organizations to become predictable, effective and efficient very quickly

Who is the customer of Testing and QA ?

- **Deming:**
  - Quality comes not from testing, but from *improvement of the development process*
  - Testing does *not* improve quality, nor guarantee quality
  - It's too late
  - The quality, good or bad, is already in the product
  - You cannot test quality into a product
- **Who is the customer of Testing and QA ?**
- **Developers are the main customer**
- **What do we have to deliver to these customers ?**
  What are they *waiting for* ?

QA versus Testing !
Antagonism or Symbiosis ?

- **What is Quality Assurance ?**
  - Helping projects and management to prevent problems
- **What is Testing ?**
  - One of the quality measurement techniques of QA
  - Checking that what development did was right
  - Finding where development isn't perfect yet
  - Does Testing only apply to code ?
- **What else ?**
  - Modelling
  - Scenarios
  - Reviews
  - Inspections

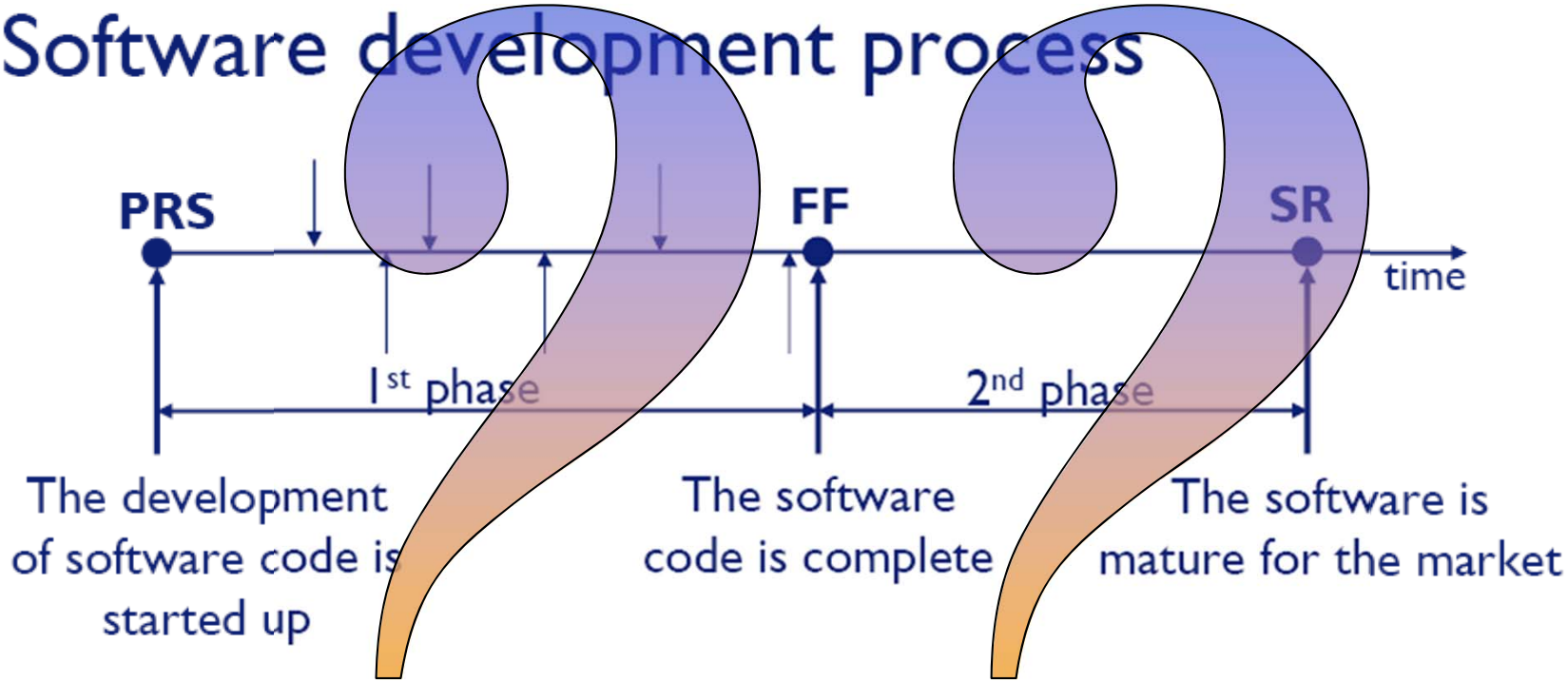


The process of defect injection

Conventional software development:

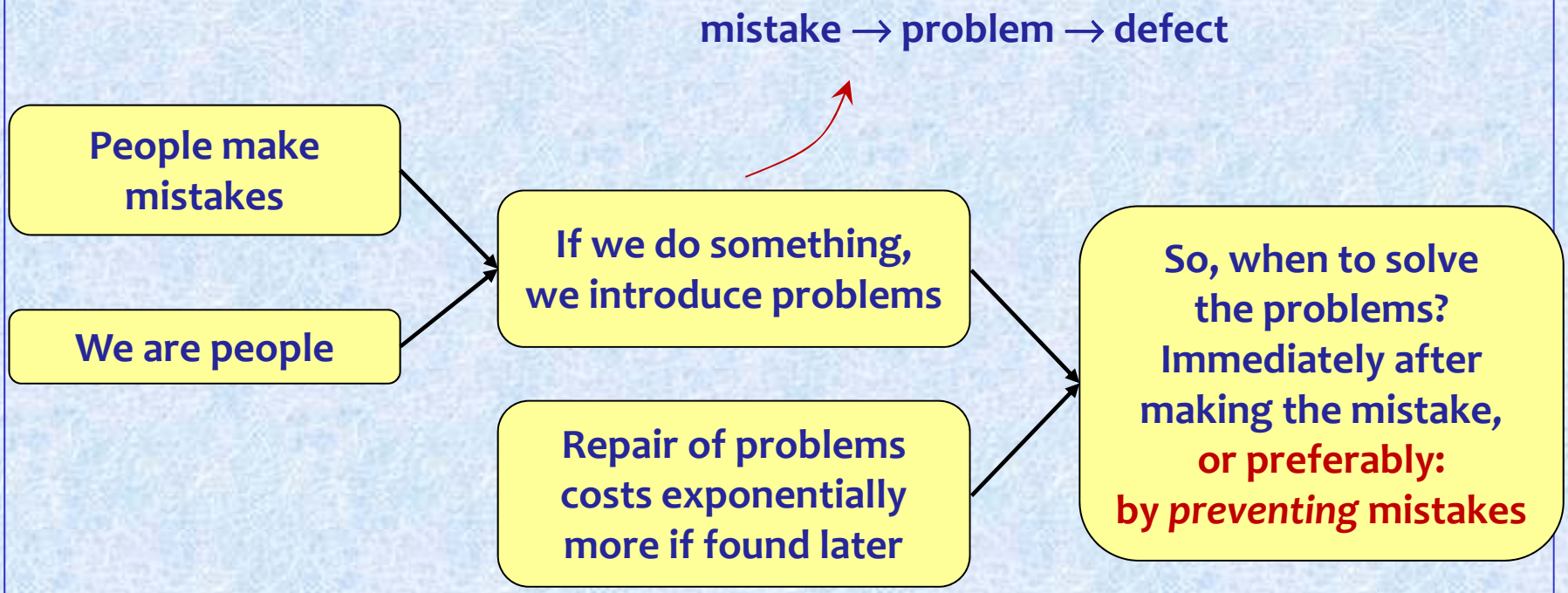
1. **Development phase: inject bugs**
2. **Debugging or Testing phase: find bugs and fix bugs**

Software development process

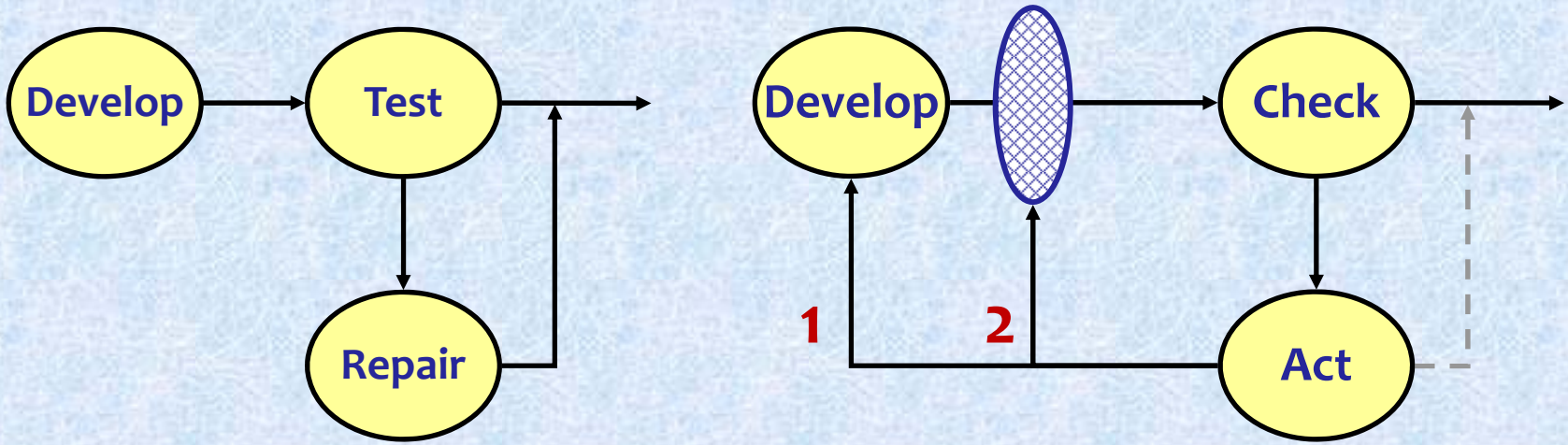


- 1st phase is developing phase
- 2nd phase is de-bugging phase

Inevitable consequence



Testing is checking correctness



What we often see

What we should expect

1. Prevention
2. Catch immediately after injection

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

All we have to do ...

- A defect is only a defect if it causes a problem
- 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 Problem

- Users experience defects

Apparently

- Developers inject defects
- Testers don't find them all

However,

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

And there is a large budget to do something about it:

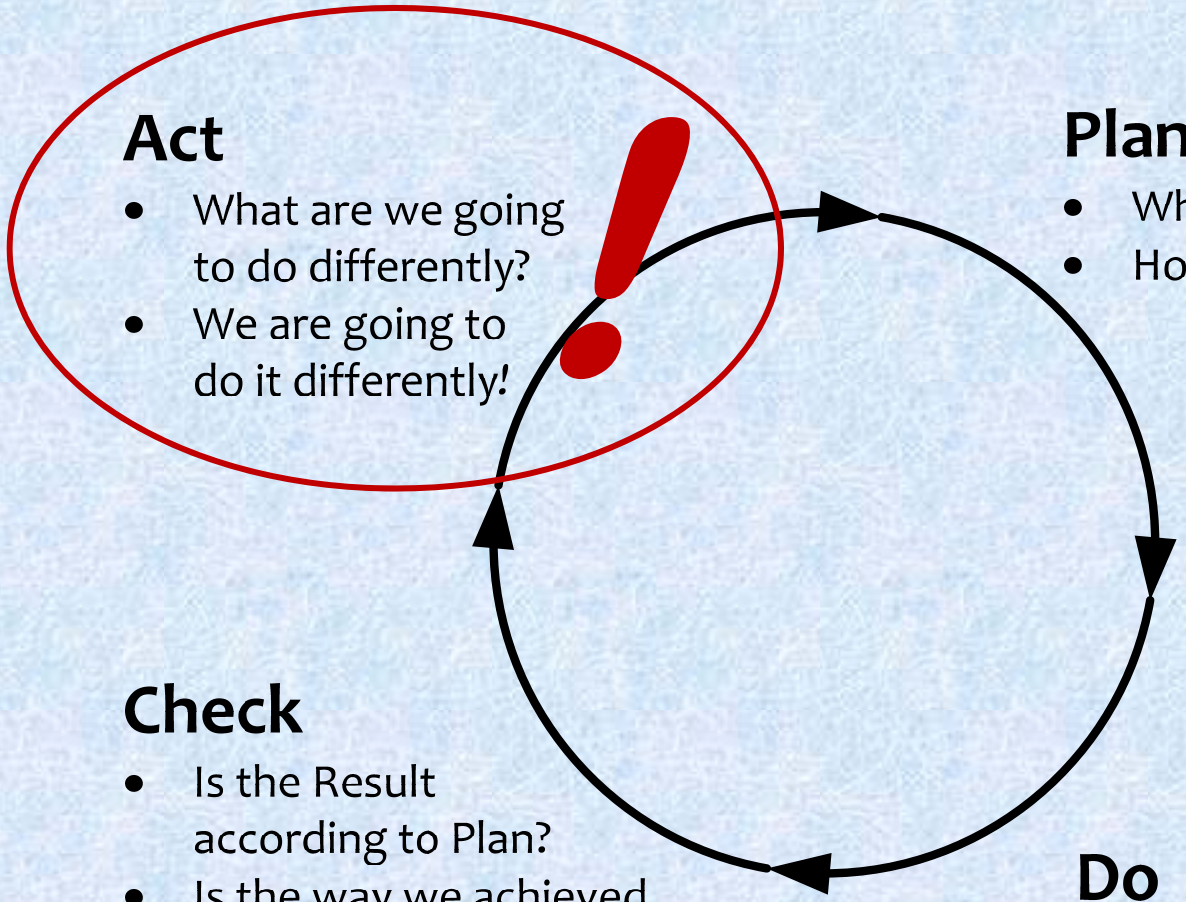
- Some 50% of project time is consumed by all kinds of testing and repairing
- About 50% of developed software is never put into use
- Over 50% of software put into use is never used

What techniques do testers have
to do their job properly ?

- **What was their job ?** (just checking)
- **What techniques do they need to do a proper job ?**
  - Focus on what's necessary to reach the goal
  - Even if that's not what you've been told before
  - Don't believe anything I say, check it out yourself
  - Continuous improvement using Plan-Do-Check-Act on
    - What you do (the product)
    - How you do it (the project)
    - How you organize all of this (the processes)

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

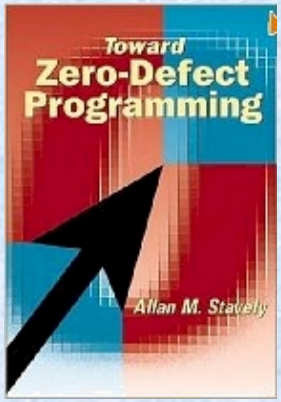
Do

Carry out the Plan

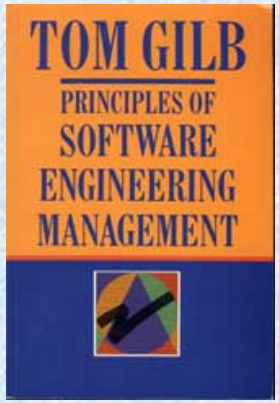
Check

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

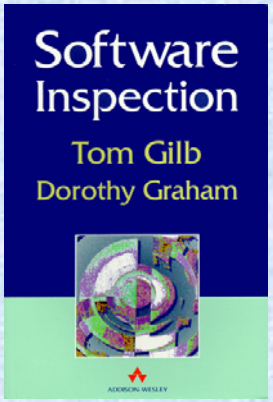
‘Old’, forgotten or ignored methods



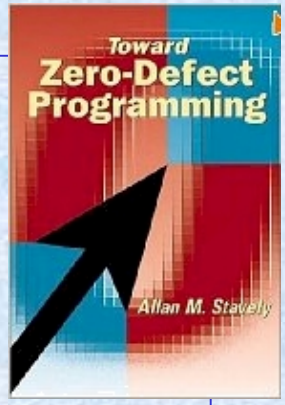
- **Cleanroom software engineering** - Harlan Mills - 1970’s
  - Incremental Development - Short Iterations
  - Defect prevention rather than defect removal
  - Inspections to feed prevention
  - No unit tests needed
  - Statistical testing
  - If final tests fail: no repair - start over again
  - 10-times less defects at lower cost
  - Quality is cheaper



- **Evolutionary Delivery - Evo** - Tom Gilb - 1974, 1976, 1988, 2005
  - Incremental + Iterative + Learning and consequent adaptation
  - Fast and Frequent Plan-Do-Check-Act
  - Quantifying Requirements - Real Requirements
  - Defect prevention rather than defect removal
- **Early Inspections**
  - Not waiting until the whole waterfall of a document (or code-module) has been polluted with defects



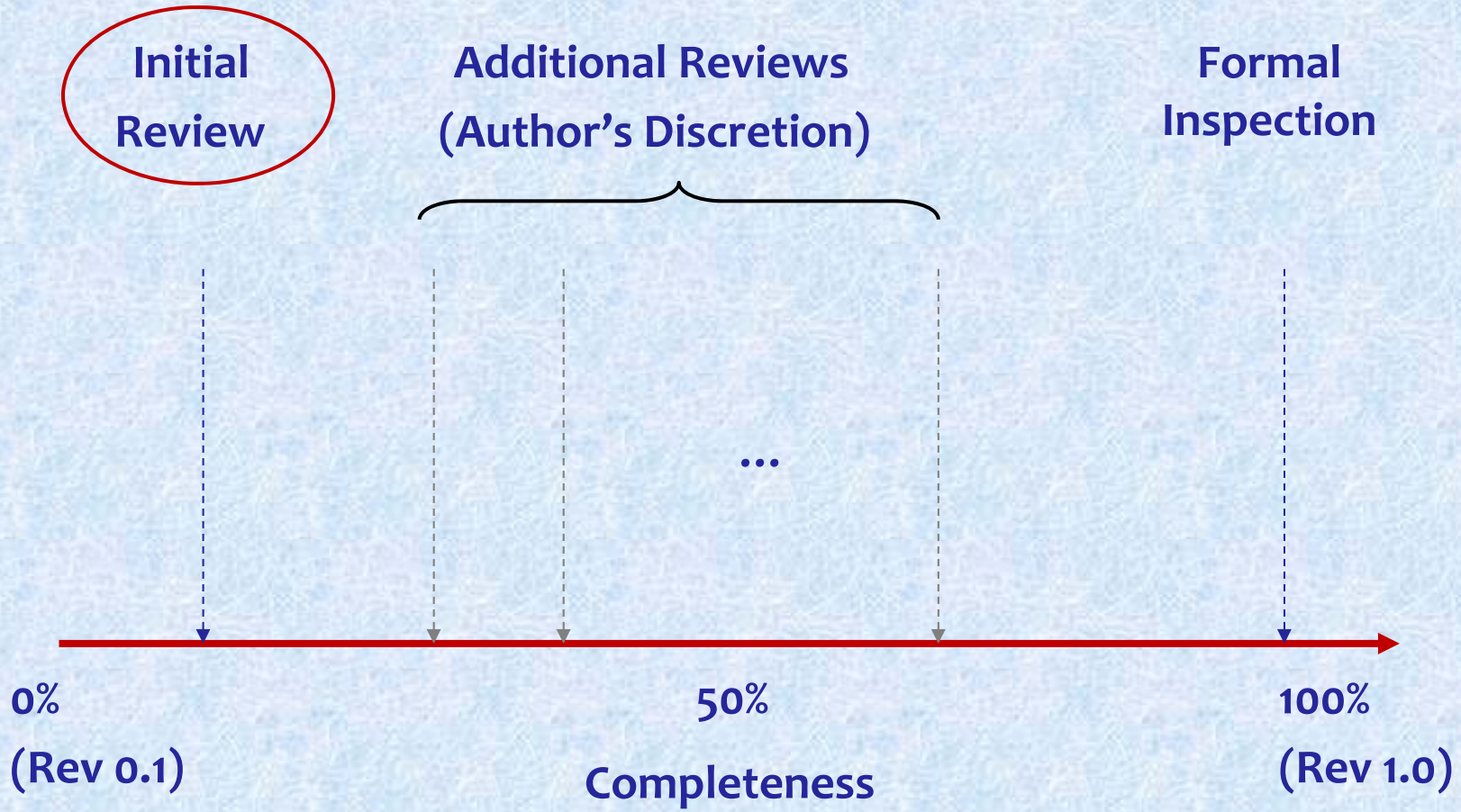
Testing in Cleanroom (1970's)



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

Early Inspection

Prevention costs less than Repair



Case Study - Situation

- A tester's improvement writing successive test plans
  - Early Inspection used on a project to improve test plan quality
  - Test plan nearly "complete", so we simulated Early Inspection
  - First round: inspected 6 randomly-selected test cases
  - Author notes systematic defects in the results, reworks the document accordingly (~32 hrs)
  - Second round: inspected 6 more test cases; quality vastly improved
  - Test plan exits the process and goes into production
  - The author goes on to write another test plan on the next project

Case Study - Results

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

- Time investment: 2 hours in initial review, 36 hours total in final formal inspection, excluding rework
  (2 inspections, 4 hrs each, 4 checkers, plus preparation and debrief)
- Historically about 25% of all defects found by testing were closed as "functions as designed", with 2-4 hrs spent on each to find out
- This test plan yielded over 1100 software defects with only 1 defect (0.1 %) closed as "functions as designed"
- Time saved: 500 - 1000 hrs (25% x 1100 x 2-4 hrs)
- Total time spent: < 100 hrs

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

Case Study 2 - Situation

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

Case Study 2 - Results

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

- Time investment: 26 hr
  - 12 hours in initial review (1.5 hrs per author)
  - About 8 hours in additional reviews
  - 6 hours in final inspection
    (2 hrs, 2 checkers, plus prep and debrief)
- Major defects prevented:
  5 per requirement in ~750 total requirements
- (5 x 750 x 10 hr = 37500 hr) / 3 → 12500 x $50 = $625 000 saved

Early Detection vs. Prevention

An eight-work-year development, delivered in five increments over nine months for Sema Group (UK), found:

- 3512 defects through inspection
- 90 through testing
- and 35 (including enhancement requests) through product field use

After two evolutionary deliveries, unit testing of programs was *discontinued* because it was *no longer cost-effective*

Nice job! Early detection has big benefits - BUT…

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

Cost-effective defect prevention is the bottom line

Let's move

Let's move from
Fixation to Fix
to
Attention to Prevention

- If we don't immediately deal with the root, we will keep making the same mistakes over and over
- Toyota Production System: "Stop the Line"
- Without direct feedback, we won't even know
- Only with *quick feedback* we can put the repetition to a halt
- That's one reason why agile development *can* produce better quality

Evolutionary Project Management (Evo)

- **Plan-Do-Check-Act**
  - The powerful ingredient for success
- **Business Case**
  - *Why* we are going to improve *what*
- **Requirements Engineering**
  - *What* we are going to improve *and what not*
  - *How much* we will improve: quantification
- **Architecture and Design**
  - Selecting the optimum compromise for the conflicting requirements
- **Early Review & Inspection**
  - Measuring quality while doing, learning to prevent doing the wrong things

Zero Defects Attitude

Right product

Evo Project Planning

Efficiency of what we do

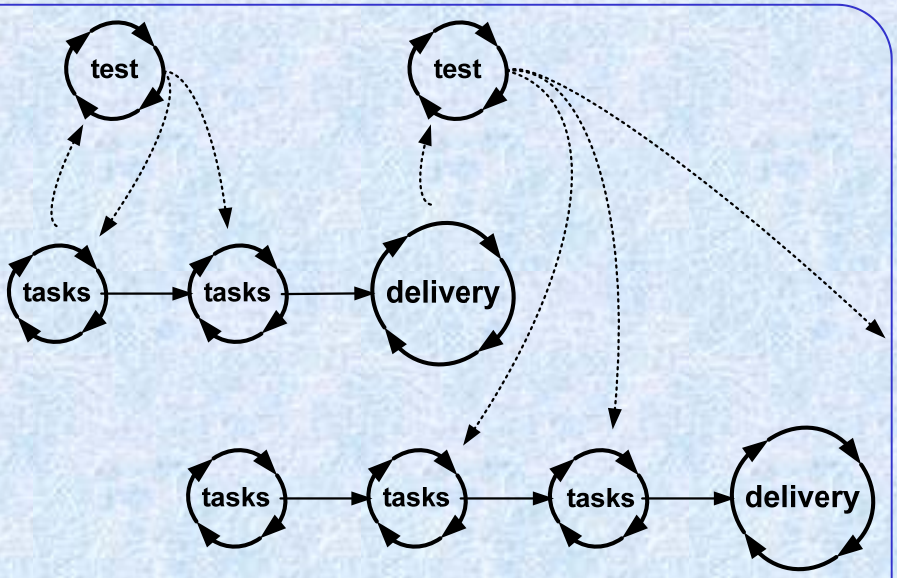
Effectiveness of what we do

Right time

- **Weekly TaskCycle**
  - Short term planning
  - Optimizing estimation
  - Promising what we can achieve
  - Living up to our promises
- **Bi-weekly DeliveryCycle**
  - Optimizing the requirements and checking the assumptions
  - Soliciting feedback by delivering Real Results to *eagerly waiting* Stakeholders
- **TimeLine**
  - Getting and keeping control of Time: Predicting the future
  - Feeding program/portfolio/resource management

What will happen and what will we do about it?

Suggestions



- Developers organize their work in weekly TaskCycles
- Testers also organize their work in weekly TaskCycles
- Testers *know* in advance what they are supposed to do when
- Testers conclude their work in sync with developers
- Testers check work in progress *even before* it is finished

The aim of Testing

- Being done as soon as the development is done
- Well, almost

- Excuses, excuses, excuses
  - The developers are always late
    (Evo developers live up to their promises)
  - The developers don't take us seriously
    (Evo developers ask testers for assistance)
  - The developers don't inject enough defects
    (now testing becomes a challenge)
  - Testers are the bearers of bad news
    (They should better find out what they're supposed to do)

No excuses needed !

- Helping development to be successful

More

www.malotaux.nl/booklets

1 **Evolutionary Project Management Methods (2001)**
Issues to solve, and first experience with the Evo Planning approach

2 **How Quality is Assured by Evolutionary Methods (2004)**
After a lot more experience: rather mature Evo Planning process

3 **Optimizing the Contribution of Testing to Project Success (2005)**
How Testing fits in

3a **Optimizing Quality Assurance for Better Results (2005)**
Same as Booklet 3, but for non-software projects

4 **Controlling Project Risk by Design (2006)**
How the Evo approach solves Risk by Design (by process)

5 **TimeLine: How to Get and Keep Control over Longer Periods of Time (2007)**
Replaced by Booklet 7, except for the step-by-step TimeLine procedure

6 **Human Behavior in Projects (APCOSE 2008)**
Human Behavioral aspects of Projects

7 **Evolutionary Planning, or How to Achieve the Most Important Requirement (2008)**
Planning of longer periods of time, what to do if you see 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

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

www.malotaux.nl/inspections

Inspection pages

Quality Comes Not From Testing

but from improving the development process

Niels Malotaux
Project Coach

N R Malotaux
Consultancy

questions or comments:
+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl