Predictable Projects

Delivering
the Right Things at the Right Time

No excuses needed

How can Testers and QA help ?

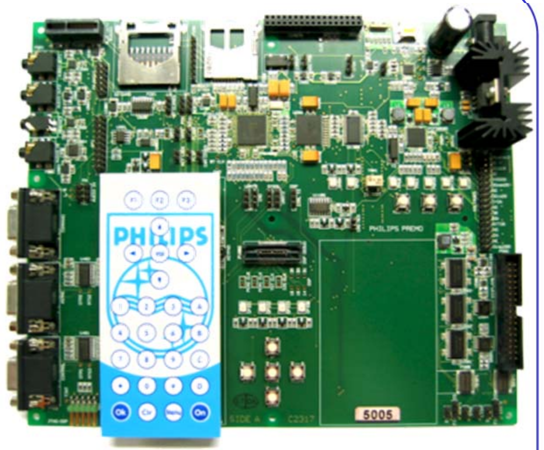
Niels Malotaux

N R Malotaux
Consultancy

niels@malotaux.nl

www.malotaux.nl

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 on track

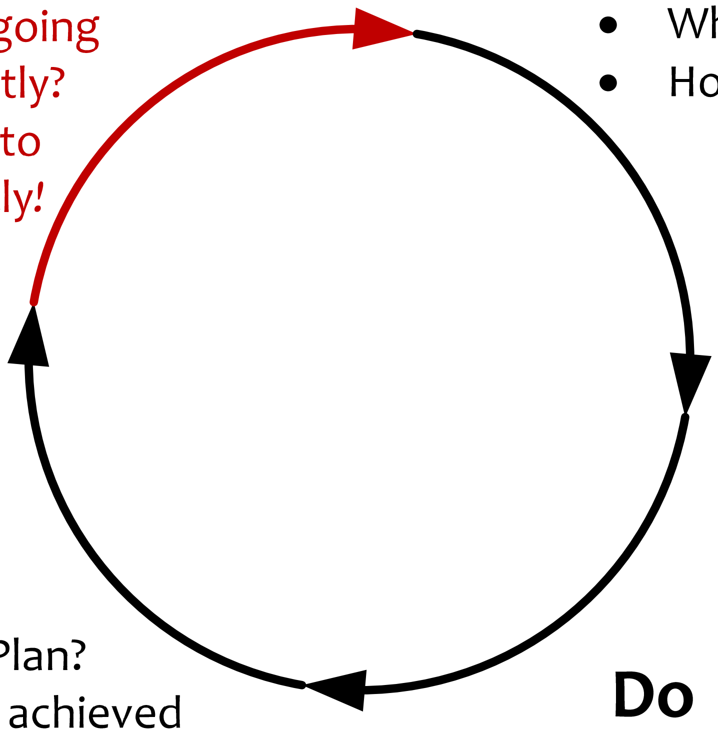
Result Management

We have a QA problem !

- Large stockpile of modules to test (hardware, firmware, software)
- Senior Tester paralyzed
- You shall do Full Regression Tests
  - Full Regression Tests take 10 to 15 days each
- Too few testers ("Should we hire more testers ?")
- Can we do something about this?
- Now do you know ?

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

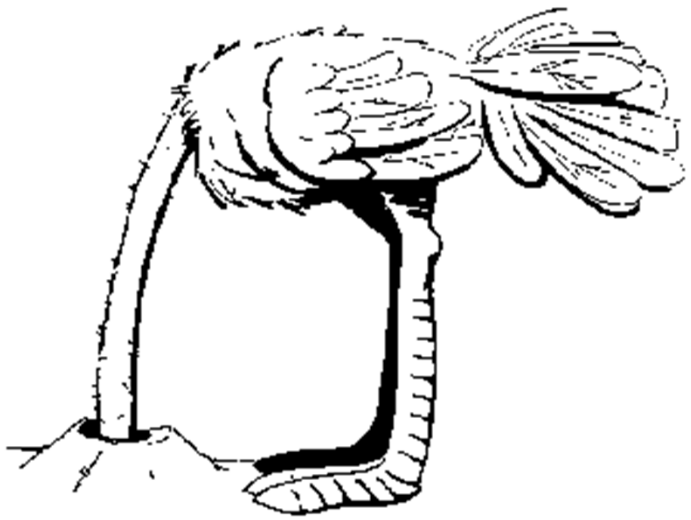
- What to achieve
- How to achieve it

Check

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

Do

Carry out the Plan



Instead of complaining about a problem …

(Stuck in the Check-phase)

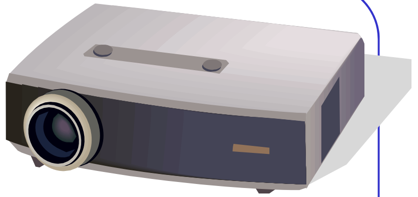
Let's do something about it *!*

(Moving to the Act-phase)

No dilution of quality

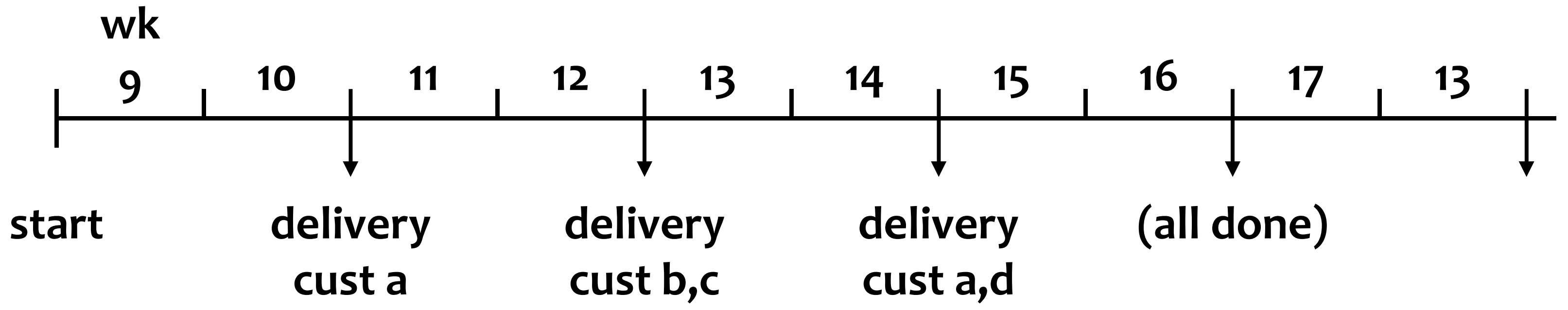
- No dilution of the responsibility for quality
- Responsibility *whether* the products can be delivered
- Sure that the customer won’t get a problem
- A defect being the cause of a problem at the customer

Objectifying and quantifying the problem is a first step to the solution



Line	Activity	Estim	Alter native	Junior tester	Devel opers	Customer	Will be done (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	1	1			Sev	
9	Package 8.2	1	1			?	
10	Package 8.3	1	1			Chrt	24 Feb
11	Package 8.4	1	1			Chrt	
12	Package 8.5	1.1	1.1			Yet	28 Feb
13	Package 8.6	3	3			Yet	24 Mar
14	Package 8.7	0.1	0.1			Cli	After 8.5 OK
15	Package 8.8	18	18			Ast	
	totals	106	47	32	36		

TimeLine



Selecting the priority order of customers to be served

- Most important customers (the company's future is at stake)
- "We'll have a solution at that date … Will you be ready for it ?"
  Another customer could be more eagerly waiting
- Delivering in a continuous flow

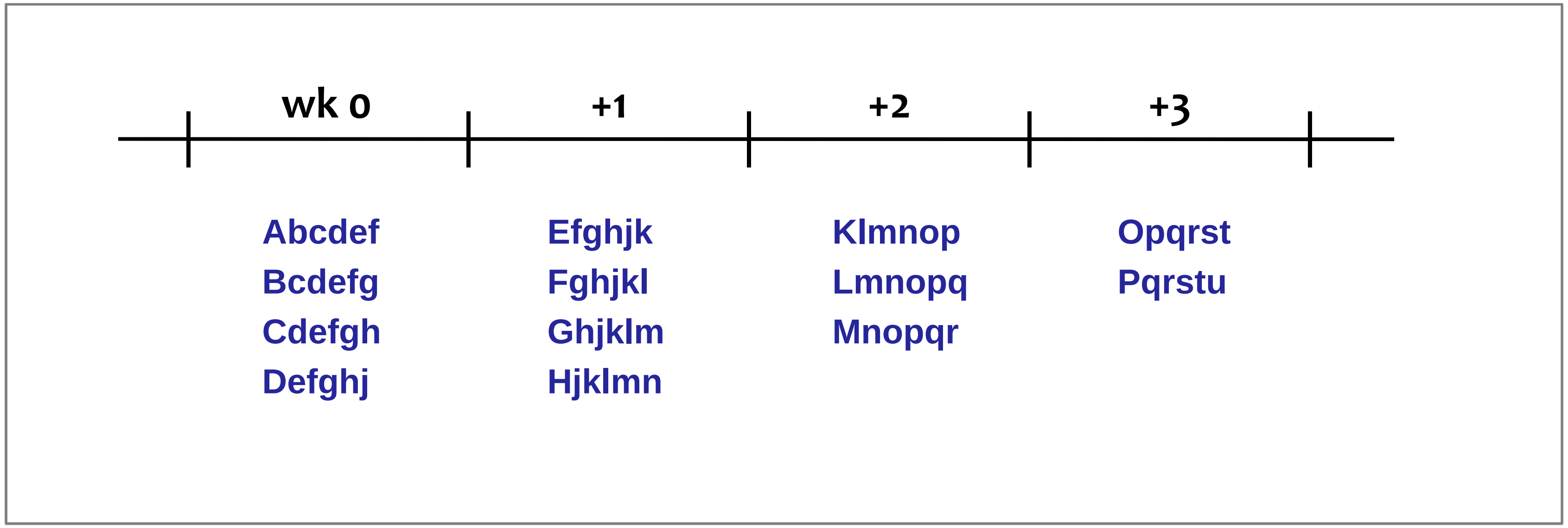
Result

- Tester empowered
- Done in 9 weeks
- So called "Full Regression Testing" was redesigned
- Customers systematically happy and amazed
- Kept up with development ever since
- Increased revenue

Later:

- Tester promoted to product manager
- Coaching successors how to stay ahead

Mobile Whiteboard



Selecting the priority order of customers to be served

- Most important customers
- "We'll have a solution at that date … Will you be ready for it ?"
  Another customer could be more eagerly waiting
- Delivering in a continuous flow

Is testing only about testing ?

- Is ‘doing your best to test’ enough ?
- Done when it’s ready ?

- What are the requirements for QA and Testing ?
- Suggestions ?

What are the requirements for QA and Testing ?

- **QA:**

- **Testing:**

Top Level Requirement

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

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

Assuming we want a quality product … (delivering *value*)

- **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 QA and Testing ?**
- **What do we have to deliver to this customer to make him more successful than before ?**
- **What do you think ?**

Top Level Requirement

Quality on Time

Delivering the Right Result at the Right Time,
wasting as little time as possible (= efficiently)

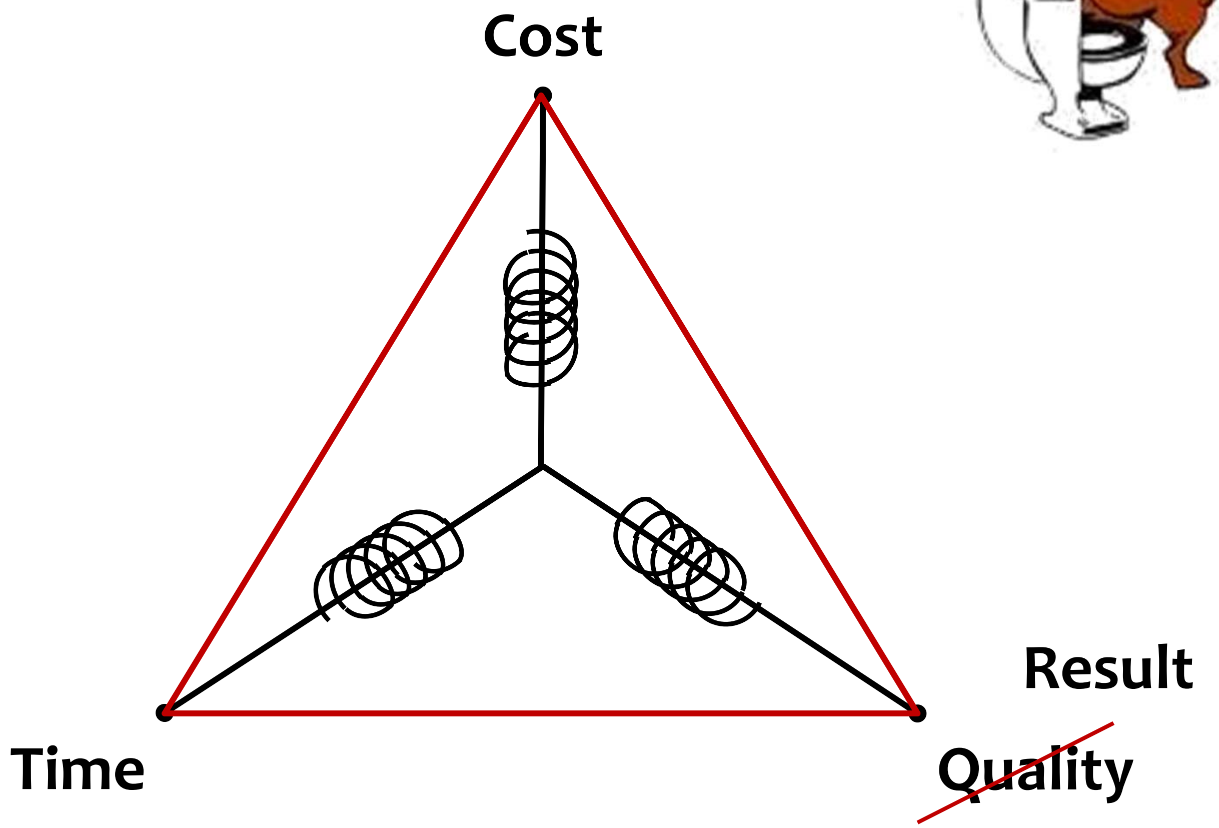
- Providing the customer with
  - what he needs
  - at the time he needs it
  - to be satisfied
  - to be more successful than he was without it
- Constrained by (win - win)
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

Can of course also be used for QA and Testing

Is Testing about finding defects ?

- How about helping the developers delivering the Right Results at the Right Time ?
  - How many defects are specified in the requirements ?
  - A defect is not a Right Result
  - If there are defects, the Right Time may be missed
- So, what would testing be about ?
  - What is the Right Result ?
  - What is the Right Time ?
- Is that clear in your current project ?

The 'Iron Triangle' fallacy



The *right* quality costs *less*

Why do we want to find defects ? 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 ?)

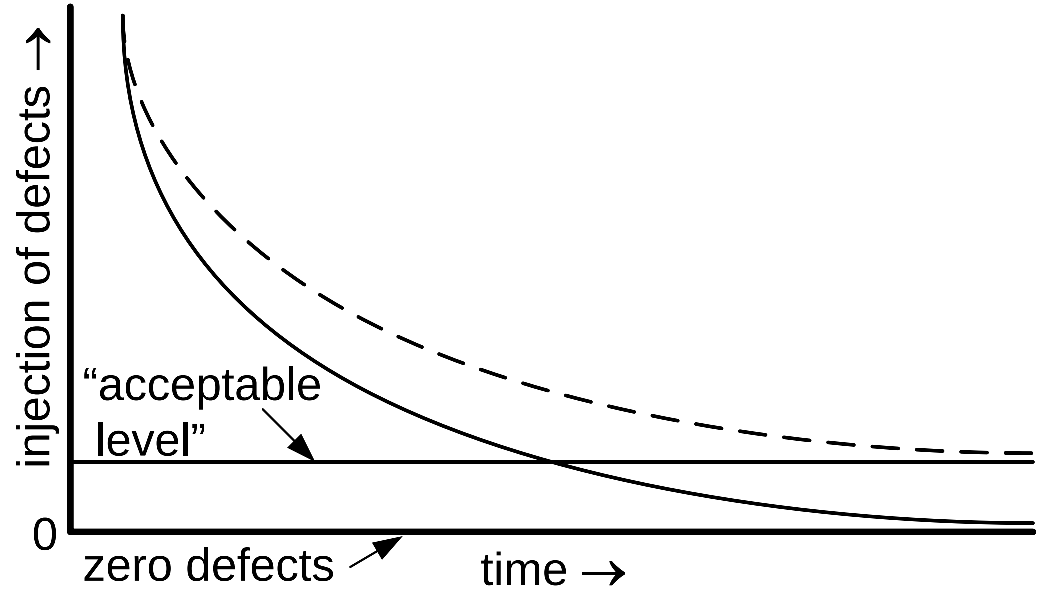
Do we deliver Zero Defect products ?

- **How many defects do you think is acceptable ?**

*

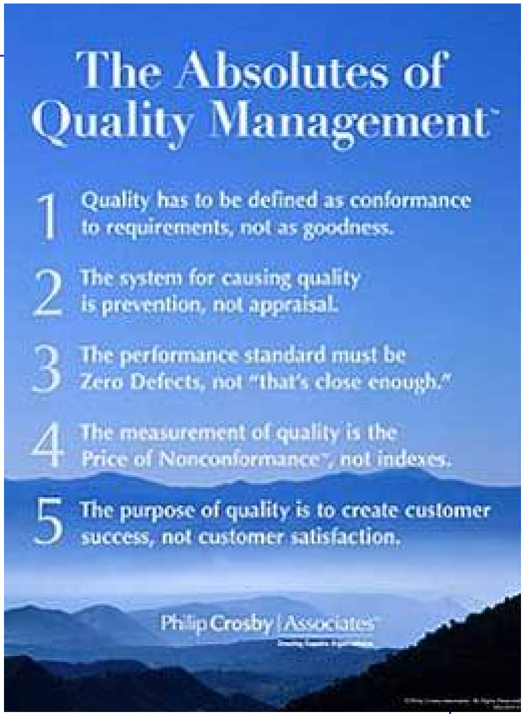
Zero Defects

- 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

Crosby: Absolutes of Quality



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™

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

Philip Crosby, 1970

- The purpose is customer success (not customer satisfaction)

Added by Philip Crosby Associates, 2004

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

Conformance to requirements

- We meet the agreed requirements

or (if that doesn't make sense)

- Have the requirements changed to what we and the customer really need
- We create requirements with care and we meet them with care

Agile

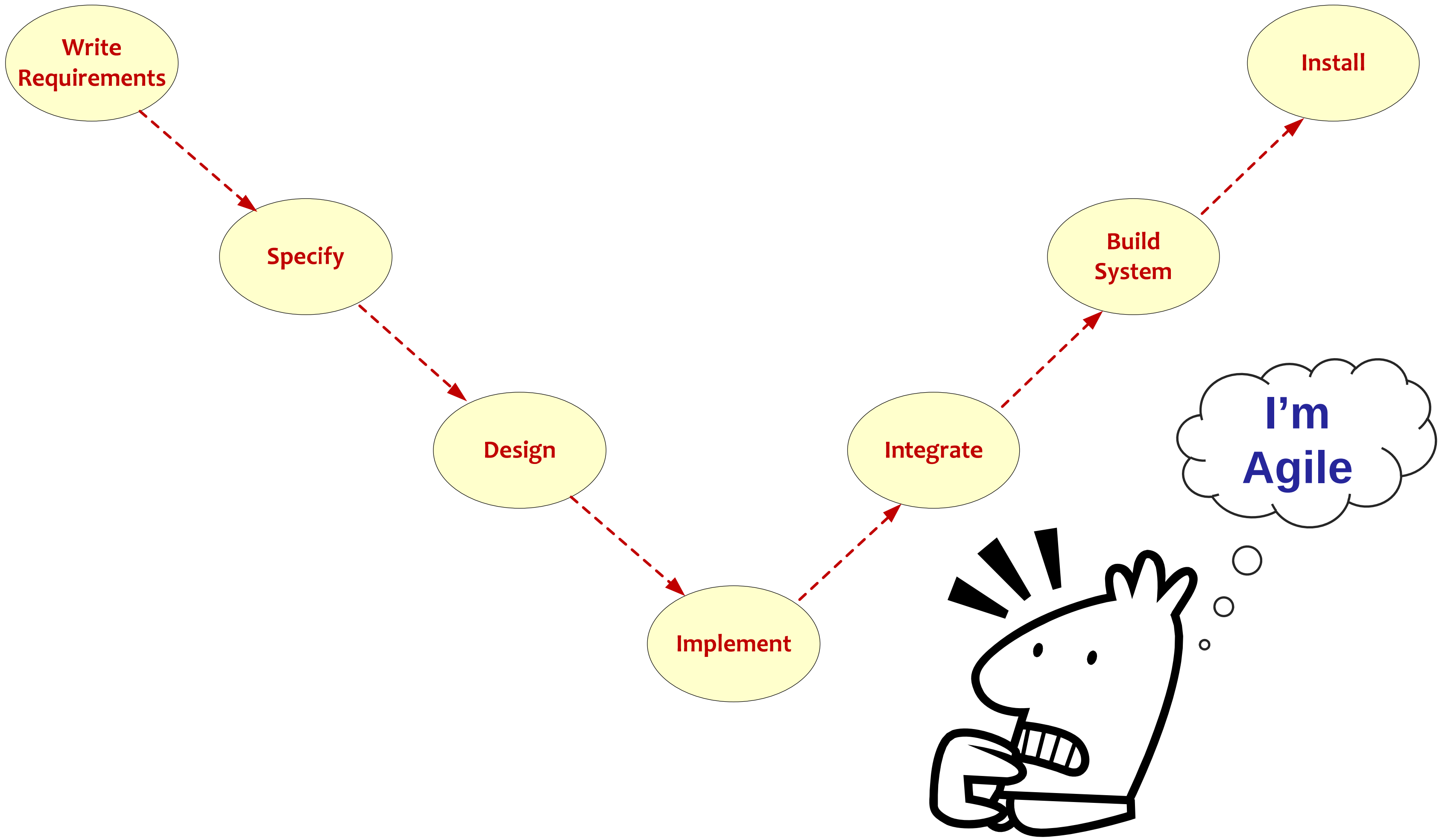
Phil Crosby

Is QA and is Testing Waste ?

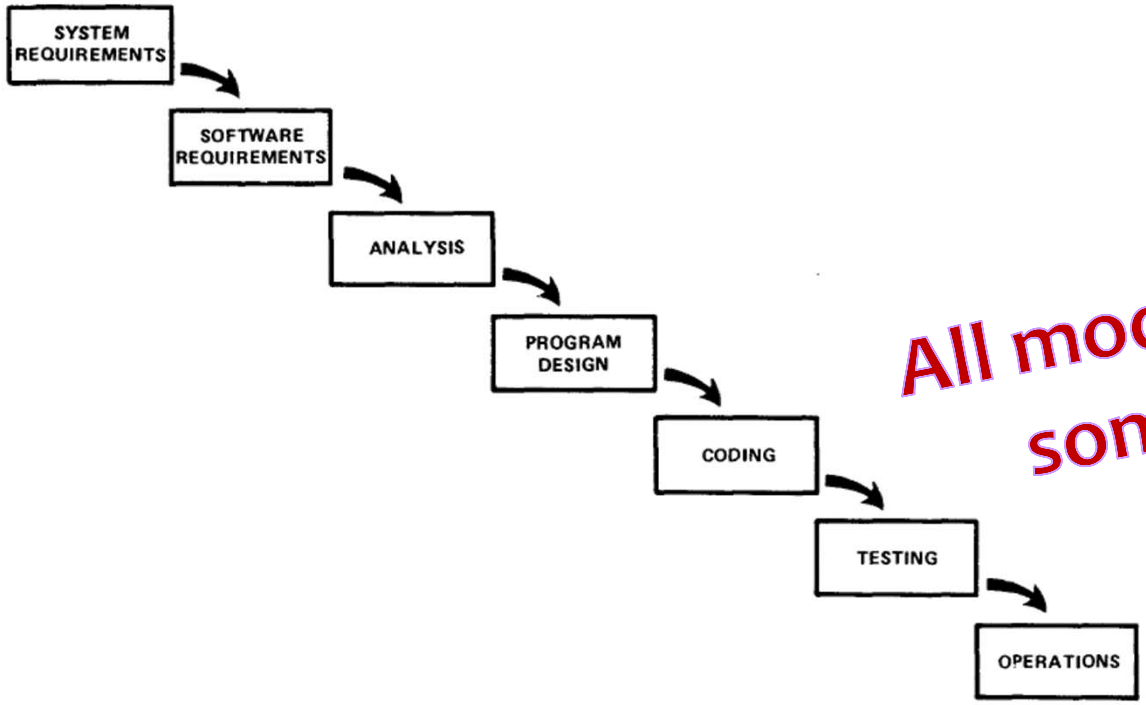
- If development would be perfect
  then what should we do as QA and Test ?

- Until development becomes perfect,
  what should we do as QA and Test ?

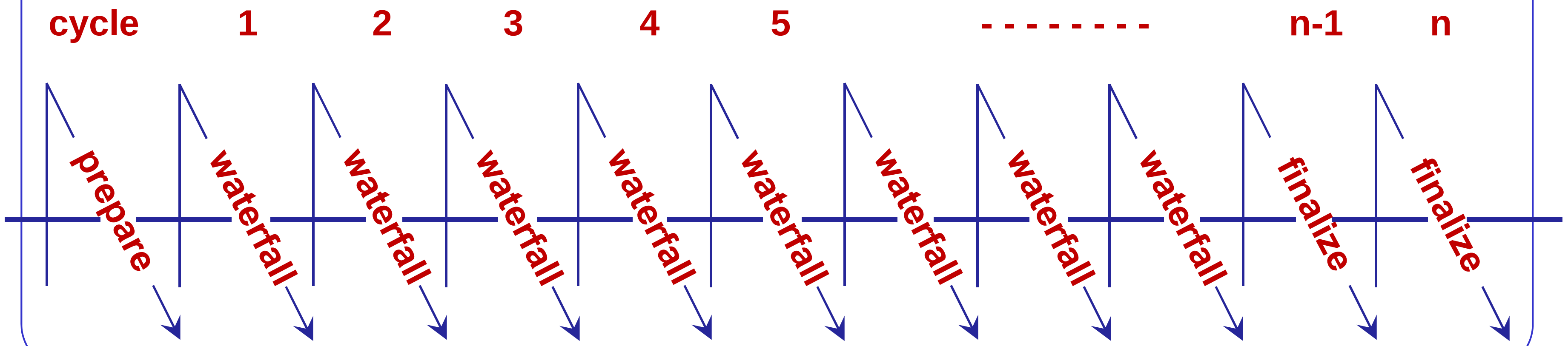
V-model


Write Requirements
Specify
Design
Implement
Integrate
Build System
Install
I'm Agile

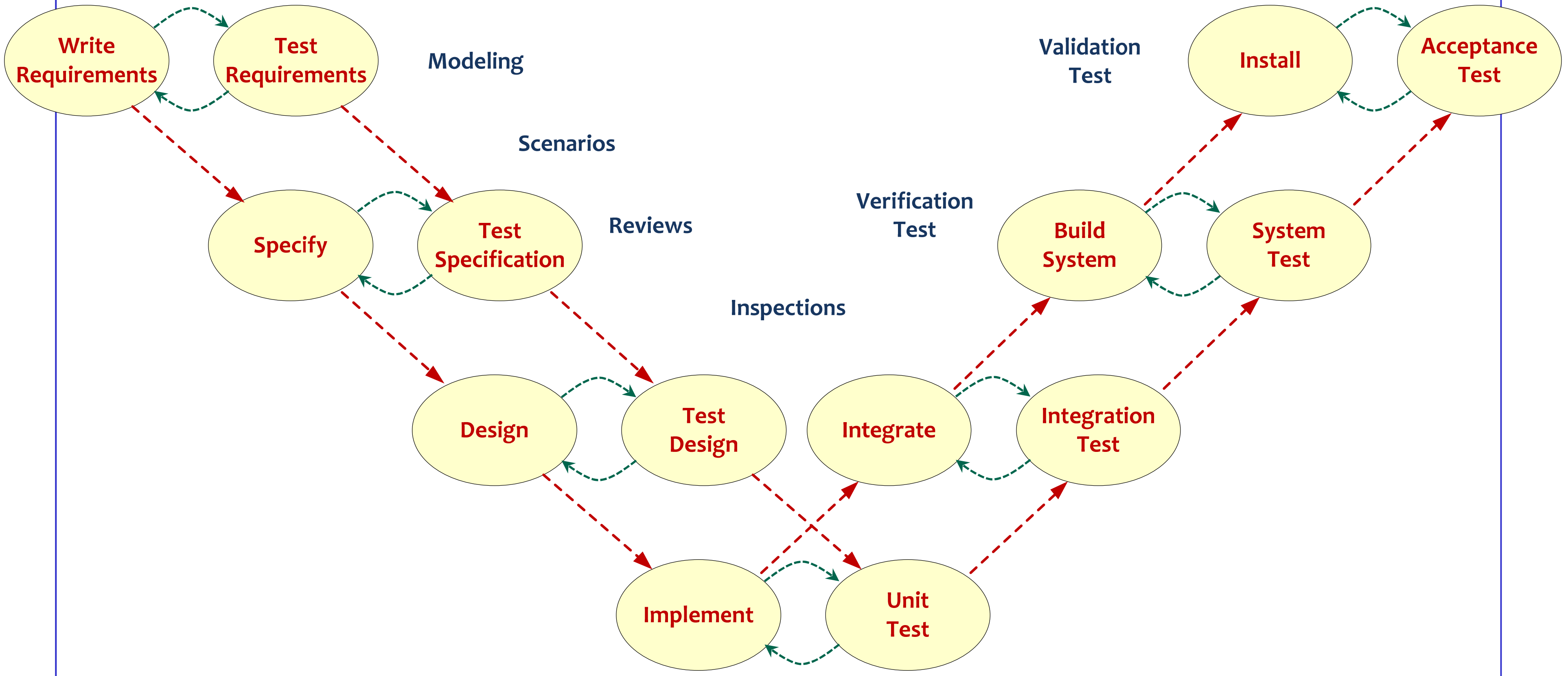
Waterfall ?



All models are wrong, some are useful



W-model

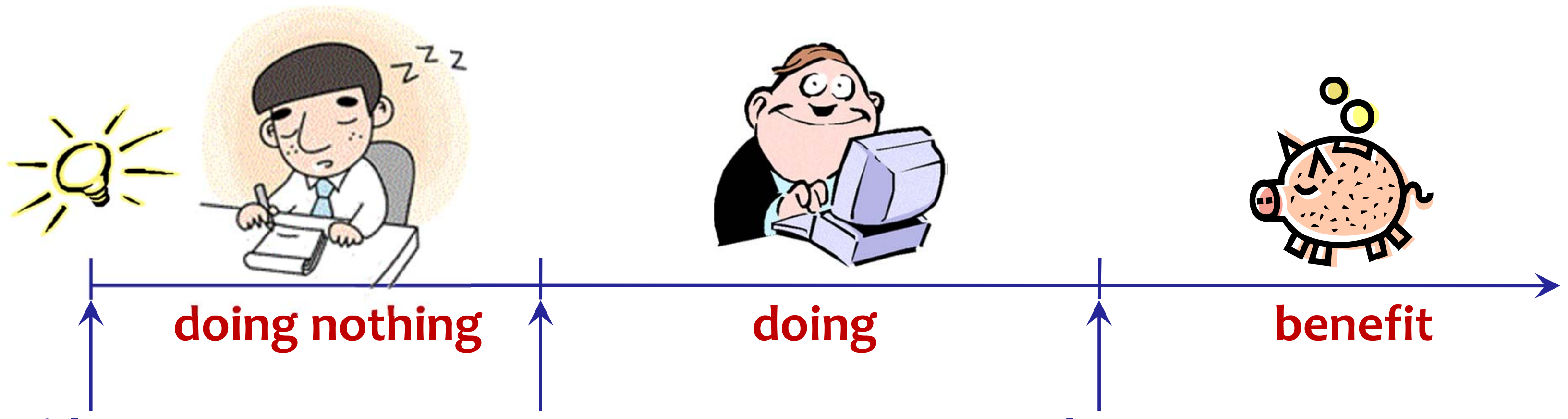

Write Requirements
Test Requirements
Modeling
Validation Test
Install
Acceptance Test
Scenarios
Specify
Test Specification
Reviews
Verification Test
Build System
System Test
Inspections
Design
Test Design
Integrate
Integration Test
Implement
Unit Test

Are many projects late ?

- Delivery Time is a Requirement, like all other Requirements
- Why are most projects late ???
- Apparently all other Requirements are more important than Delivery Time

- Are they really ?
- Does Testing have an impact on delivery ?

Why time is important



Return on Investment (ROI)

+ **Benefit of doing** - huge (otherwise other projects would be more rewarding)
– **Cost of doing** - project cost, usually minor compared with other costs
– **Cost of doing nothing** - every day we start later, we finish later
– **Cost of being late** - lost benefit

What is the cost of one day of (unnecessary) delay ?

- What is the cost of the project per day ?
- Do you know how much you cost per day?
  Note: that's not what you get !
- If you don't know the benefit,
  assume 10 times the cost of the project

- 0th order estimations are good enough

- Do we know the benefit of our project ?
- Do we know the penalty for delay ?

Testing and QA shouldn’t delay the delivery
(on the contrary)

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

No excuses needed !

- Excuses, excuses, excuses
  - The developers are always late
    (Developers can learn to live up to their promises)
  - The developers don’t take us seriously
    (Developers can ask testers for help)
  - The developers inject too many defects
    (What could you have done about it ?)
  - The developers don’t inject enough defects
    (Now testing becomes a real challenge !)

- Helping development to be successful and on time

If ‘*not delaying the delivery*’ is a requirement

- **QA / Testing not delaying the delivery**
  - **Is it possible ?**
  - **What do you think ?**
  - **Any suggestions how to make it work ?**

Things to consider

- **Plan-Do-Check-Act**
  - The powerful ingredient for succes
- **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

- **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

Efficiency of what we do

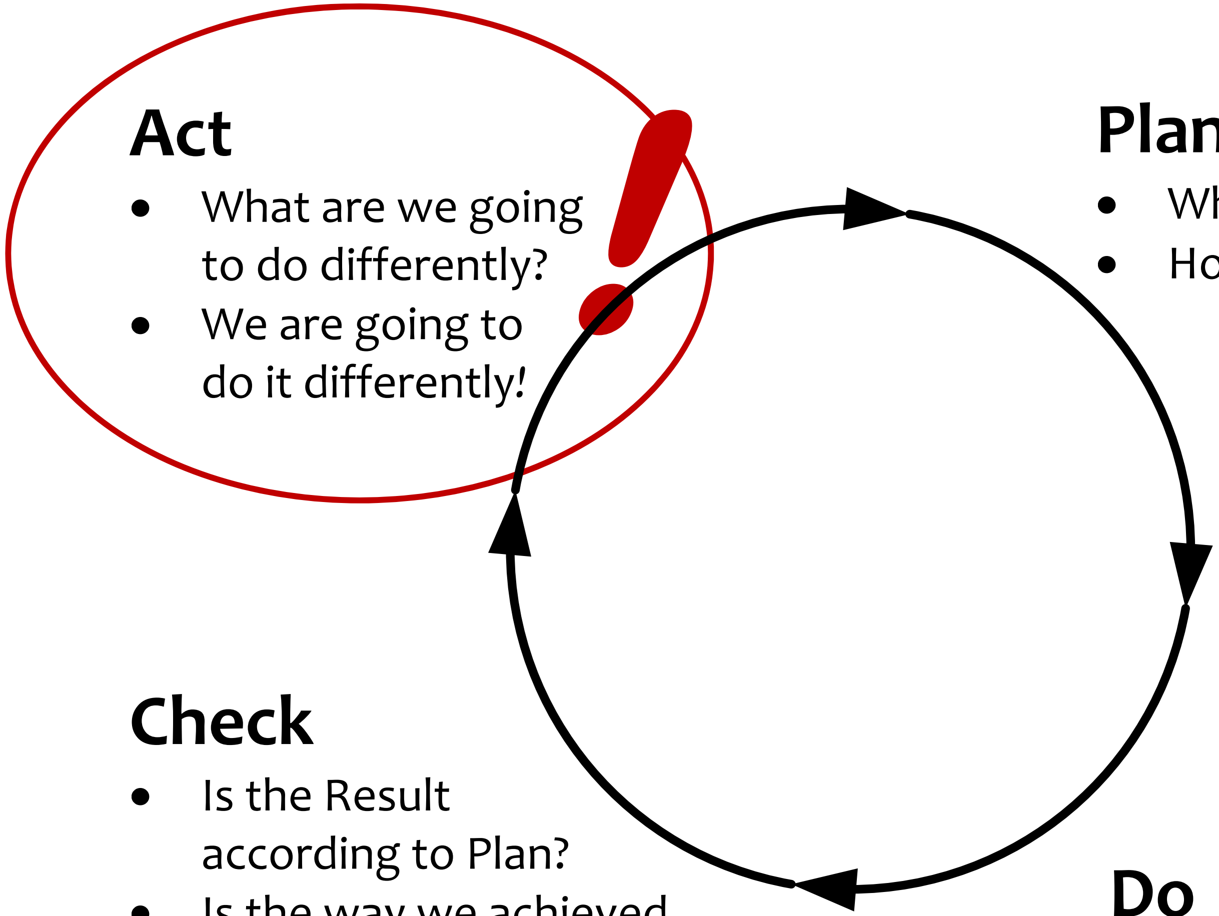
Right time

Effectiveness of what we do

What will happen and what will we do about it ?

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!

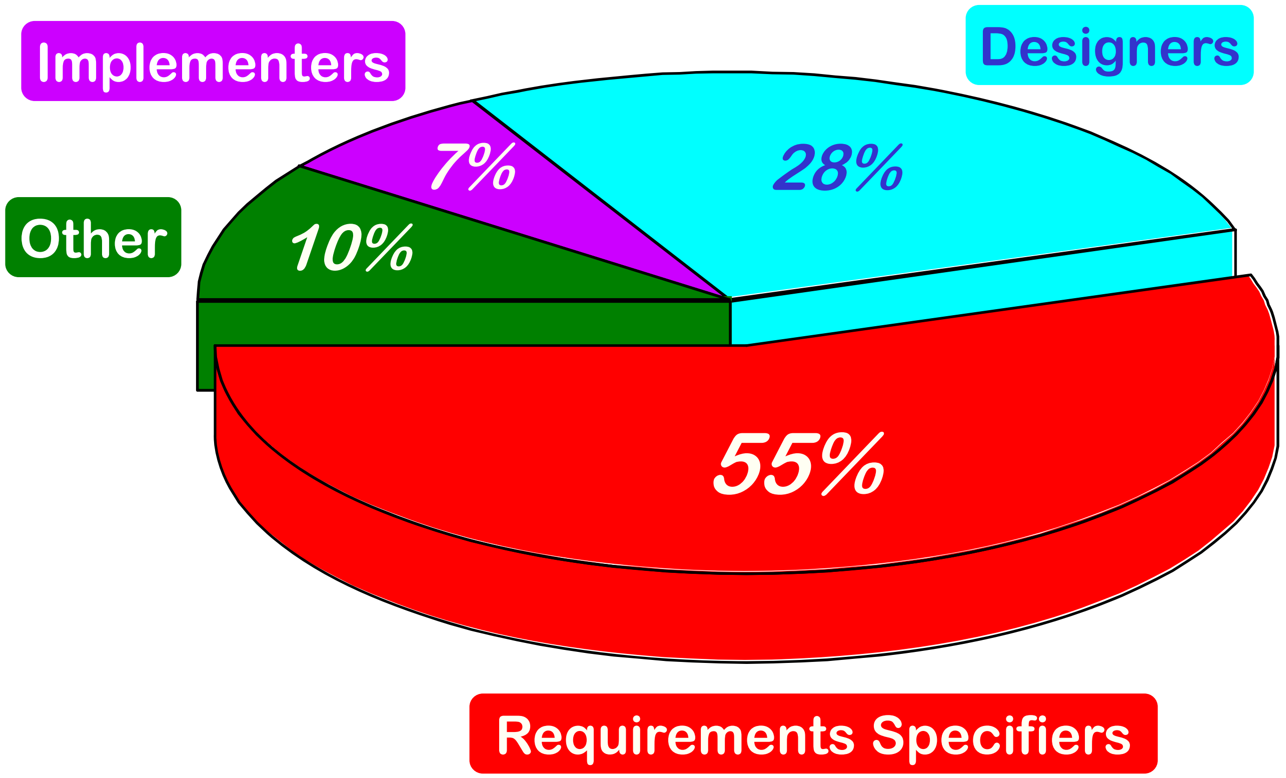
Plan
- What to achieve
- How to achieve it

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

Do

Carry out the Plan

Typical Defect Injectors (*cost* breakdown)



After Bender Associates, *1996*

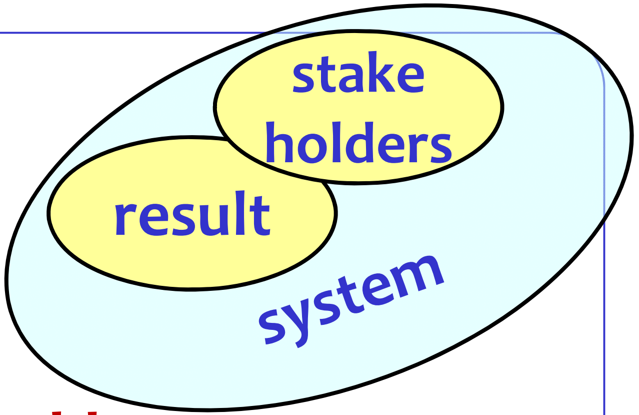
- **Where is our focus ?**
- **Where should our focus be ?**

Requirements for a game developer

- The game should be really enjoyed by the users
- The game should bring in more money

How could you have helped him ?

Stakeholders of a project



- **Every project has some 30±20 Stakeholders**
- **Stakeholders have a stake** (interest) **in the project**
- **The concerns of Stakeholders are often contradictory**
  - Apart from the Customer *they don't pay*
  - So they *have no reason to compromise !*
  - Finally, *we all pay*
- **Some Stakeholders are victims of the project**
  - They want the project to fail
- **Project risks, happening in almost every project**
- **No excuse to fail !**

If you don't know the Stakeholders

- **No Stakeholder: no requirements**
- **No requirements: nothing to do**
- **No requirements: *nothing to test***
- **If you find a requirement without a Stakeholder:**
  - **Either the requirement isn't a requirement**
  - **Or, you haven't determined the Stakeholder yet**
- **If you don't know the Stakeholder:**
  - **Who's going to pay you for your work?**
  - **How do you know that you are doing the right thing?**
  - **When are you ready?**
- **Magic Sentence: *"Who is waiting for it ?"***

What does a requirement look like ?

- Do you see requirements at all ?
- Can you test the requirements you usually see ?
- Are those requirements really relevant ?
- Are the tests you derive from those requirements really relevant ?
- What is the development project really about ?
- If you don’t know, how can you help your customer (development) ?
- What should requirements look like ?

What are *real* requirements ?

- Heathrow Terminal 5: Great success !
  - Normal people aren't interested in the technical details of a terminal
  - They only want to check-in their luggage as *easily* as possible

    and
  - Get their luggage back as *quickly* as possible in *acceptable* condition *at their destination*
  - They didn't
- One of the problems is to determine what the project (or our work in general) really is about
- What are the 'real' requirements ?
- Clear focus towards the real requirements saves time

Can you test this ?

- check-in their luggage as *easily* as possible
- get their luggage back as *quickly* as possible
- in *acceptable* condition

What should be your reflex ?

Improving something

- All functionality we produce *does already exist*
- The real reason for running our projects is creating *something better*
- Improvement of value, productivity, success, happiness for our customers through users

(remember the goal: to be *more* successful than he was without it …)

What is to be improved ?

- More productive
- More secure
- More dependable
- Better usable
- Better maintainable
- Better portable
- Better …

- Faster
- Larger
- More nice to see
- More nice to use
- More accurate
- More reliable
- More …

Did you get this measurably described ?

Improving on *existing* qualities

	V8.5	V9.0	
• **Usability.Productivity:**			
• Time to set up a typical specified report	65	20	min
• Time to generate a survey	120	0.25	min
• Time to grant access to report, distribute logins to end-users	80	5	min
• **Usability.Intuitiveness:**	265	25.25	min
• Time for medium experienced programmer to find out how to do …	15	5	min
• **Capacity.RuntimeConcurrency**			
• Max number of concurrent users, click-rate 20 sec, response time < 0.5 sec	250	6000	users

after FIRM / Gilb 2005

Somebody said the requirements should be *SMART*

S	Specific
M	Measurable
A	Attainable
R	Realisable
T	At the right Time

Requirements with Planguage

ref Tom Gilb

Specific

Measurable

Attainable

Time

Realizable

Definition:

RQ27:	Speed of Luggage Handling at Airport
Scale:	Time between <arrival of airplane> and first luggage on belt
Meter:	<measure arrival of airplane>, <measure arrival of first luggage on belt>, calculate difference

Benchmarks (Playing Field):

Past:	2 min [minimum, 2012], 8 min [average, 2012], 83 min [max, 2012]
Current:	< 4 min [competitor y, Jan 2013] ← <who said this?>, <Survey Dec 2012>
Record:	57 sec [competitor x, Jan 2010]
Wish:	< 2 min [2014Q3, new system available] ← CEO, 19 Jan 2013, <document ...>

Requirements:

Must:	< 10 min [99%, Q4] ← SLA
Must:	< 15 min [100%, Q4, Heathrow T4] ← SLA
Goal:	< 15 min [99%, Q2], < 10 min [99%, Q3], < 5 min [99%, Q4] ← marketing

Why care for ‘real’ requirements ?

- If you test what the developers developed,
  you may be testing a
  perfect solution to the wrong problem

- Is that the right thing to do ?

Things to consider

- **Plan-Do-Check-Act**
  - The powerful ingredient for succes
- **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

- **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

Efficiency of what we do

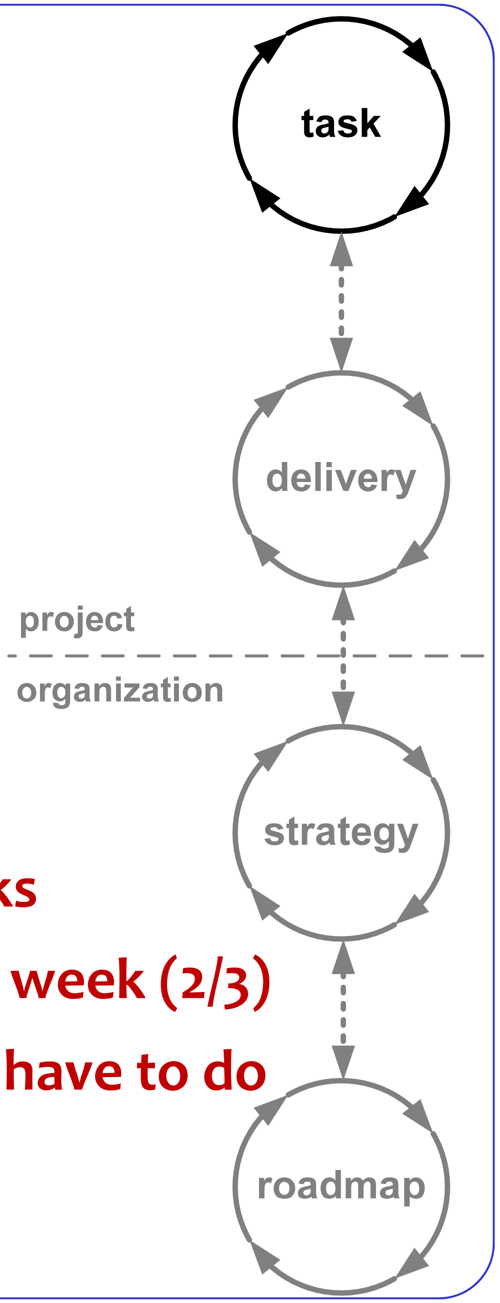
Right time

Effectiveness of what we do

What will happen and what will we do about it ?

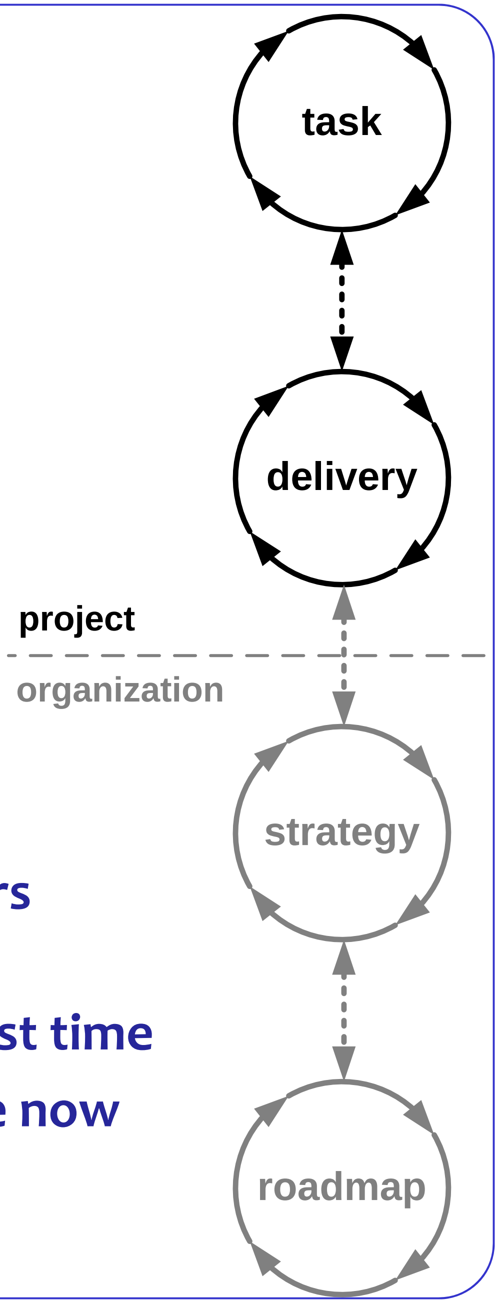
Evo Planning: Weekly TaskCycle

- Are we *doing* the right things,
  in the right order,
  to the right level of detail for now
- Optimizing estimation , planning and tracking abilities to better predict the future
- Select highest priority tasks, never do any lower priority tasks, never do undefined tasks
- There are only about 26 plannable hours in a week (2/3)
- In the remaining time: do whatever else you have to do
- Tasks are always done, 100% done

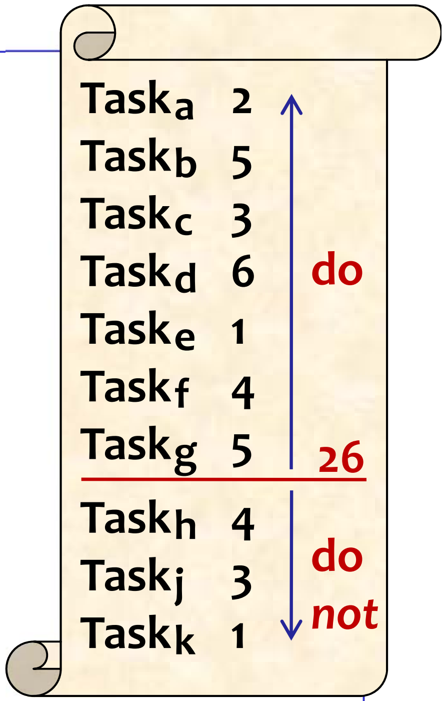


DeliveryCycle

- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
     stakeholder values that can be made in the least time
  - What will make Stakeholders more productive now
- Not more than 2 weeks



Every week we plan



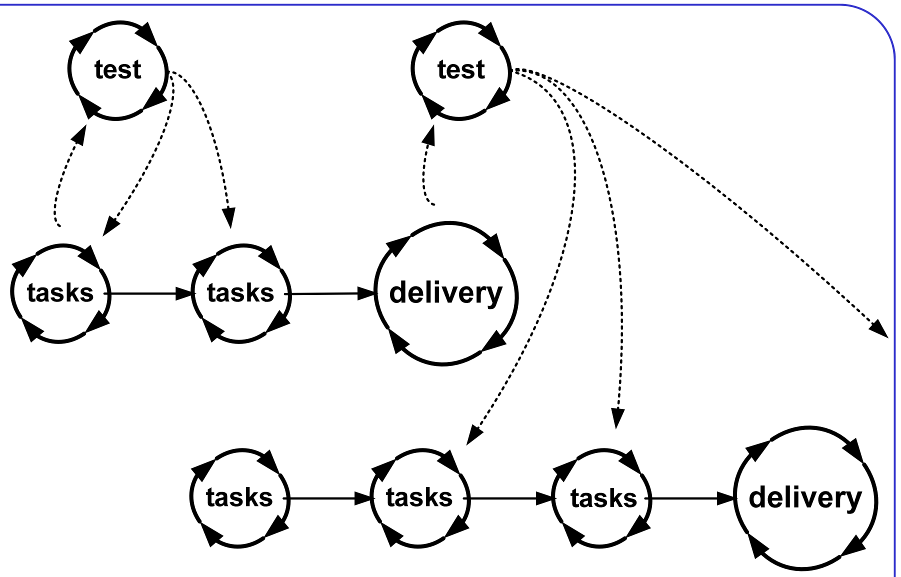
- How much time do we have available
- 2/3 of available time is net plannable time
- What is most important to do
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr per week)
- What can, and are we going to do
- What are we *not* going to do
- *Write it down ! Our fuzzy mind isn't good enough !*

2/3 is default start value
this value works well in development projects

Weekly 3-Step Procedure

- **Individual preparation**
  - Conclude current tasks
  - What to do next
  - Estimations
  - How much time available
- **Modulation with / coaching by Project Management (1-on-1)**
  - Status (all tasks done, completely done, not to think about it any more ?)
  - Priority check (are these really the most important things ?)
  - Feasibility (can it all be done by the end of the week ?)
  - Commitment and decision (it will all be done by the end of the week !)
- **Synchronization with group (team meeting)**
  - Formal confirmation (this is what we plan to do)
  - Concurrency (do we have to synchronize ?)
  - Learning
  - Helping
  - Socializing

Organizing it



- **Developers organize their work in weekly TaskCycles**
- **Testers organize their work in weekly TaskCycles**
- **Testers *know* what they are supposed to test**
  Because they *know* what the developers are doing and will deliver
- **Testers conclude their work in sync with developers**
- **Testers check work in progress *even before* it is finished**

The Real Benefit of TaskCycles

- We see issues before they cause trouble
- And deal with them before they cause trouble
- QA and Testing can see issues the developers don't see (yet)

- Retrospectives:
  - Seeing issues *after* they caused trouble
- Better use *prespectives*
  - That's what TaskCycle planning is for

Developers are constantly optimizing

- The product
  how to arrive at the most effective product (goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

Testers are constantly optimizing

- The product
  how to arrive at the most effective product (goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

Do we agree ?

- 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 *before* it is finished
- Testers solve the Review and Inspection organizing problem
- Testing is organized the Evo way, entangling intimately with the development process

More

www.malotaux.nl/?id=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/?id=inspections
Inspection pages

☑ **Take one !**
(if still there, otherwise download)

Predictable Projects

Delivering
the Right Things at the Right Time

No excuses needed

How can Testers and QA help ?

Niels Malotaux

N R Malotaux
Consultancy

niels@malotaux.nl

www.malotaux.nl

Please evaluate my presentation

Use the AgileTD Mobile App at
www.touchmyconference.com/ATD2013

I would appreciate your feedback at
niels@malotaux.nl

Thank you very much

Any issue with the requirements for you work in QA or Testing ?

- **QA:**
- **Testing:**

What could we do about it ?

•

Typical issues

- Testers squeezed between developers and deadline ?
- Testers delaying the delivery ?
- Testers (forced to) compromise their responsibility ?
- Customer finding issues ?
- …