

# Help !



# We have a QA Problem !

Niels Malotaux

[niels@malotaux.eu](mailto:niels@malotaux.eu)

[www.malotaux.eu/conferences](http://www.malotaux.eu/conferences)

[www.malotaux.eu/booklets](http://www.malotaux.eu/booklets) - booklet#8



# Niels Malotaux

- Independent Team, Project, 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
- Project rescue
- Sometimes actually developing a product, eating my own dogfood



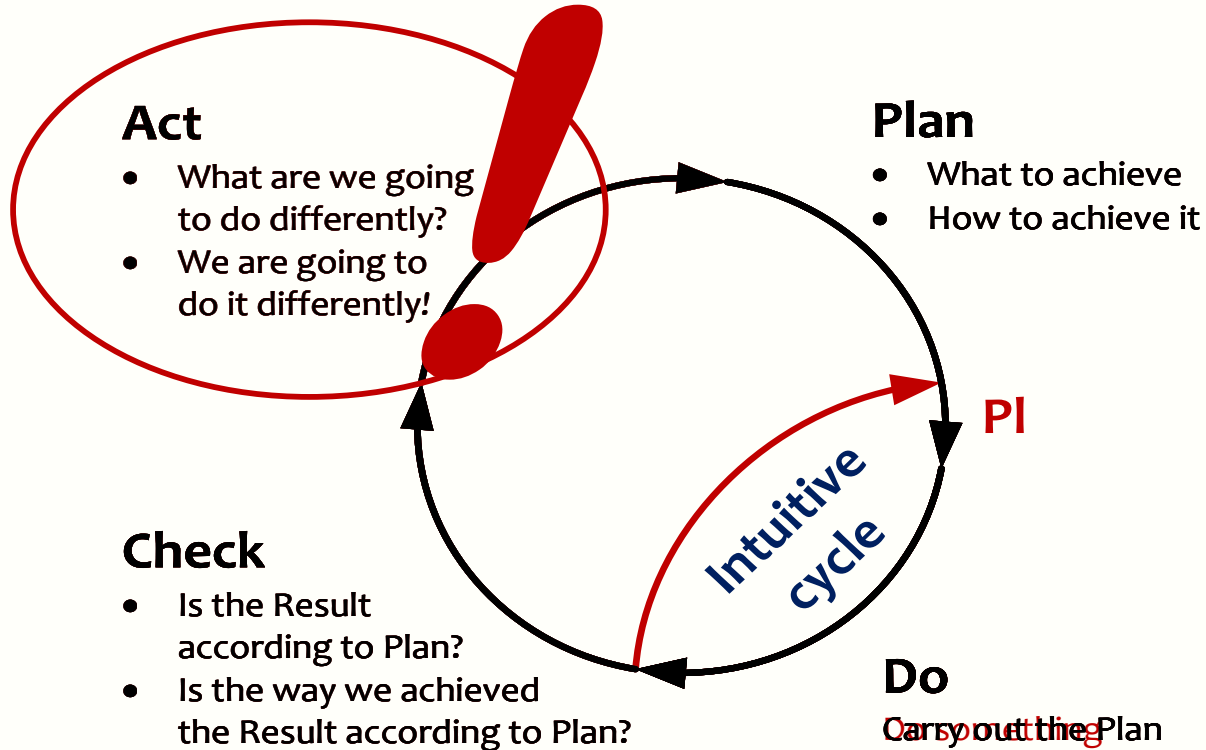
## We have a QA problem !

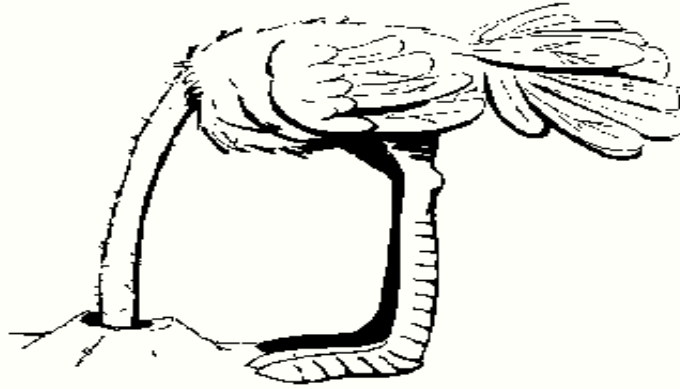
- Large stockpile of modules to test (hardware, firmware, software)
- You shall do Full Regression Tests
- Full Regression Tests take about 15 days each
- Too few testers (“Should we hire more testers ?”)
- Senior Tester paralyzed
- Can you help us out ?



# The essential ingredient: the PDCA Cycle

(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)





Instead of complaining about a problem ...

(Stuck in the Check-phase)

Let's do something about it !

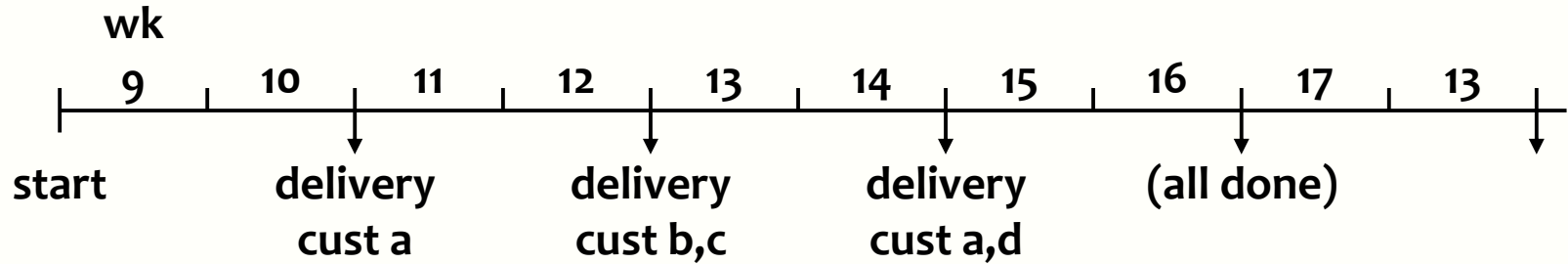
(Moving to the Act-phase)

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



Line	Activity	Estim	Alternative	Junior tester	Developers	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	26	1			Sev	
9	Package 8.2	1	1			?	
10	Package 8.3	1	1			Chrt	
11	Package 8.4	1	1			Chrt	
12	Package 8.5	1.1	1.1			Yet	
13	Package 8.6	3	3			Yet	
14	Package 8.7	0.1	0.1			Cli	
15	Package 8.8	18	18			Ast	
	<b>totals</b>	<b>106</b>	<b>47</b>	<b>32</b>	<b>36</b>		

# TimeLine



## Selecting the priority order of customers to be served

- “We’ll have a solution at that date ... Will you be ready for it ?”  
Another customer could be more eagerly waiting
- Most promising customers

# Can we make an important customer happy the next day ?

Line	Activity	Estim	Alternative	Junior tester	Developers	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	
11	Package 8.4	1	1			Chrt	24 Feb
12	Package 8.5	1.1	1.1			Yet	20 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	
	<b>totals</b>	<b>106</b>	<b>47</b>	<b>32</b>	<b>36</b>		



# 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

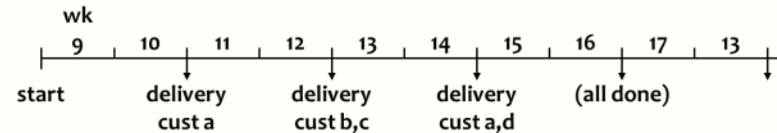
## Later:

- Tester promoted to product manager
- Still coaching successors how to plan

# TimeLine principles

- Cutting the work into chunks
- Estimating (usually takes very little time)
- Adding up (this averages the uncertainties !)
- Usually doesn't fit in the available time
- Find strategies to solve the dilemma
- Select 'best' strategy
- Predict what will happen when
- Learn and repeat every week, keeping predictions up-to-date

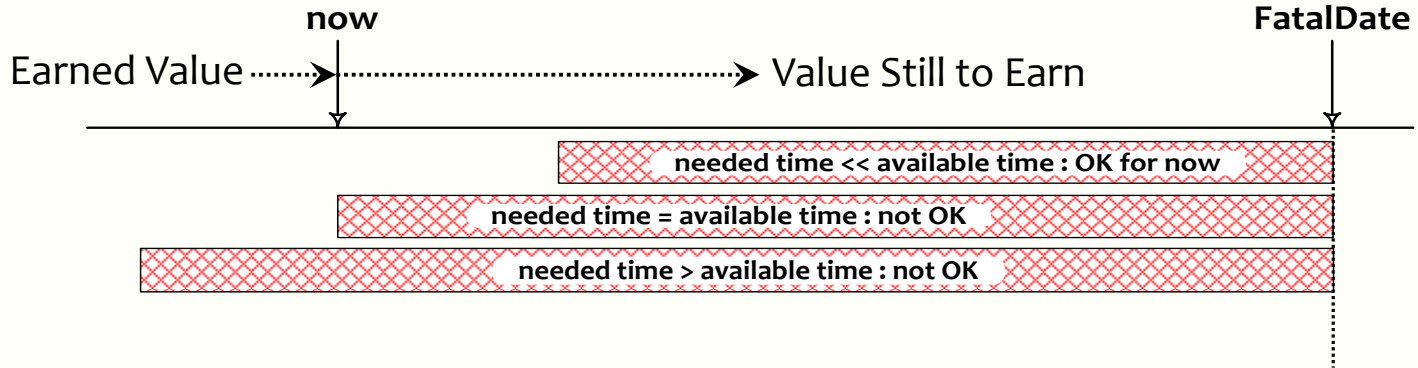
Line	Activity	Estim	Alter native	Junior tester	Devel opers	Customer	Will be done (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	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: Predicting *what* will be done *when*

Line	Activity	Estim	Spent	Still to spend	Ratio real/est	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	21/15 = 1.4 1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009
7	Activity 7	1				1.4	1.4	10 Apr 2009
8	Activity 8	3				1.4	4.2	16 Apr 2009
↓	↓							
16	Activity 16	4				1.4	5.6	2 Jun 2009
17	Activity 17	5				1.4	7.0	11 Jun 2009
18	Activity 18	7				1.4	9.8	25 Jun 2009

# What do we do if we see we won't make it on time ?



- Value Still to Earn ←versus→ Time Still Available
- If it doesn't fit ... count backwards
- If the match is over, you cannot score a goal

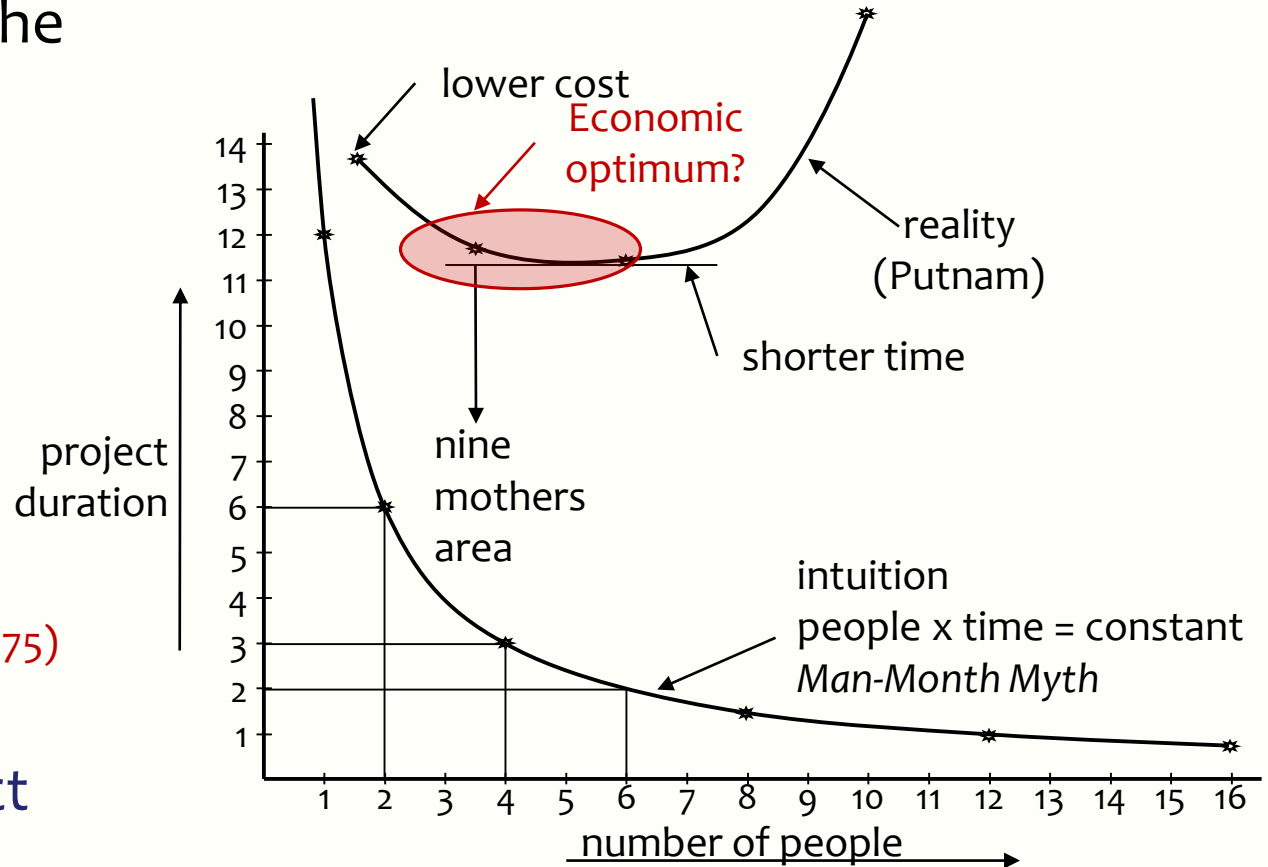


# Deceptive options

- **Hoping for the best** (fatalistic)
- **Going for it** (macho)
- **Working Overtime** (fooling ourselves)
- **Moving the deadline**
  - Parkinson's Law
    - Work expands to fill the time for its completion
  - Student Syndrome
    - Starting as late as possible, only when the pressure of the FatalDate is really felt

# The Myth of the Man-Month

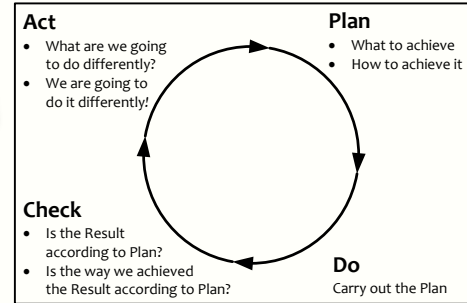
**Brooks' Law (1975)**  
Adding people  
to a late project  
*makes it later*





## Saving time

Continuous  
elimination of waste



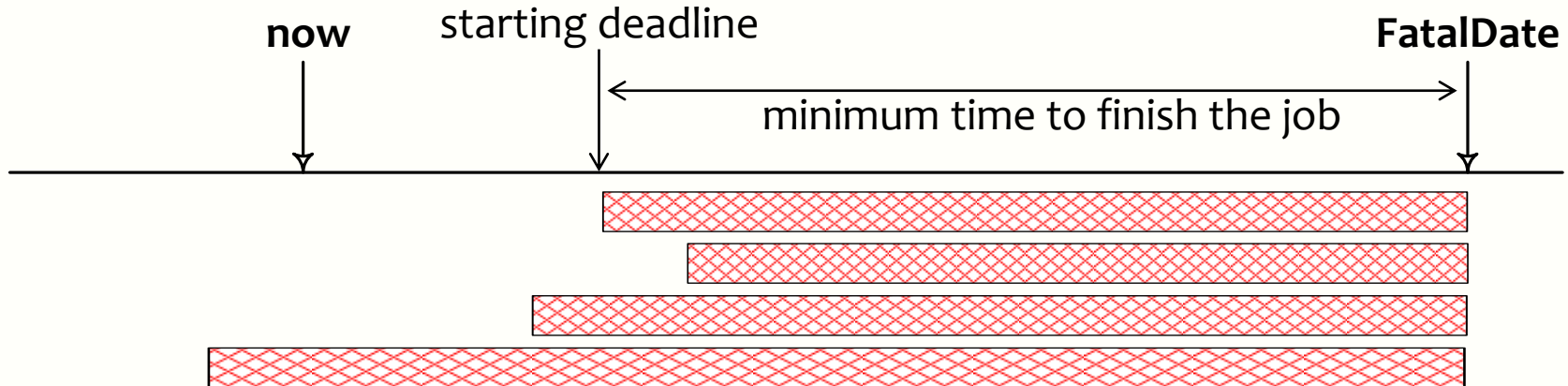
We don't have enough time, but we can save time  
*without negatively affecting the Result !*

- **Efficiency in *what (why, for whom) we do*** - doing the right things
  - Not doing what later proves to be superfluous
- **Efficiency in *how we do it*** - doing things differently
  - The product
    - Using proper and most efficient solution, instead of the solution we always used
  - The project
    - Doing the same in less time, instead of immediately doing it the way we always did
  - Continuous improvement and prevention processes
    - Constantly learning doing things better and overcoming bad tendencies
- **Efficiency in *when we do it*** - right time, in the right order
- **TimeBoxing** - much more efficient than FeatureBoxing

# Even more important: *Starting Deadlines*

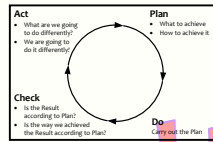
## Starting deadline

- Last day we can start to deliver by the end deadline
- Every day we start later, we will end later





# Evolutionary Project Management elements (Evo) – Tom Gilb



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

## Business Case

Why

- Why we are going to improve what
- Requirements Engineering
  - What we are going to improve and what not
  - How much we will improve: *quantification*

What  
How much  
Are we done

## 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

How  
Check as early  
as possible

Zero  
Defects  
Attitude

## Weekly TaskCycle

- Short term planning
- Optimizing estimation
- Promising what we can achieve
- Living up to our promises

Efficiency  
of what we do

## Bi-weekly DeliveryCycle

- Optimizing the requirements and checking the assumptions
- Soliciting feedback by delivering Real Results to *eagerly waiting Stakeholders*

Effectiveness  
of what we do

## 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?*

## Evo Project Planning - Niels

# Help !

## We have a QA Problem !

### Problem Solved

Niels Malotau

[niels@malotau.eu](mailto:niels@malotau.eu)

[www.malotau.eu/conferences](http://www.malotau.eu/conferences)

[www.malotau.eu/booklets](http://www.malotau.eu/booklets) - booklet#8

---

## [www.malotaux.eu/booklets](http://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 Evolutionary Planning, or 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 - 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/insp](http://www.malotaux.eu/insp)

Inspection pages