

Help !



We have a QA Problem !

Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/conferences

www.malotaux.eu/booklets - booklet#8



Niels Malotaux



- Independent Engineering and Team 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

Quality On Time
Delivering
the Right Result
at the Right Time



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 ?



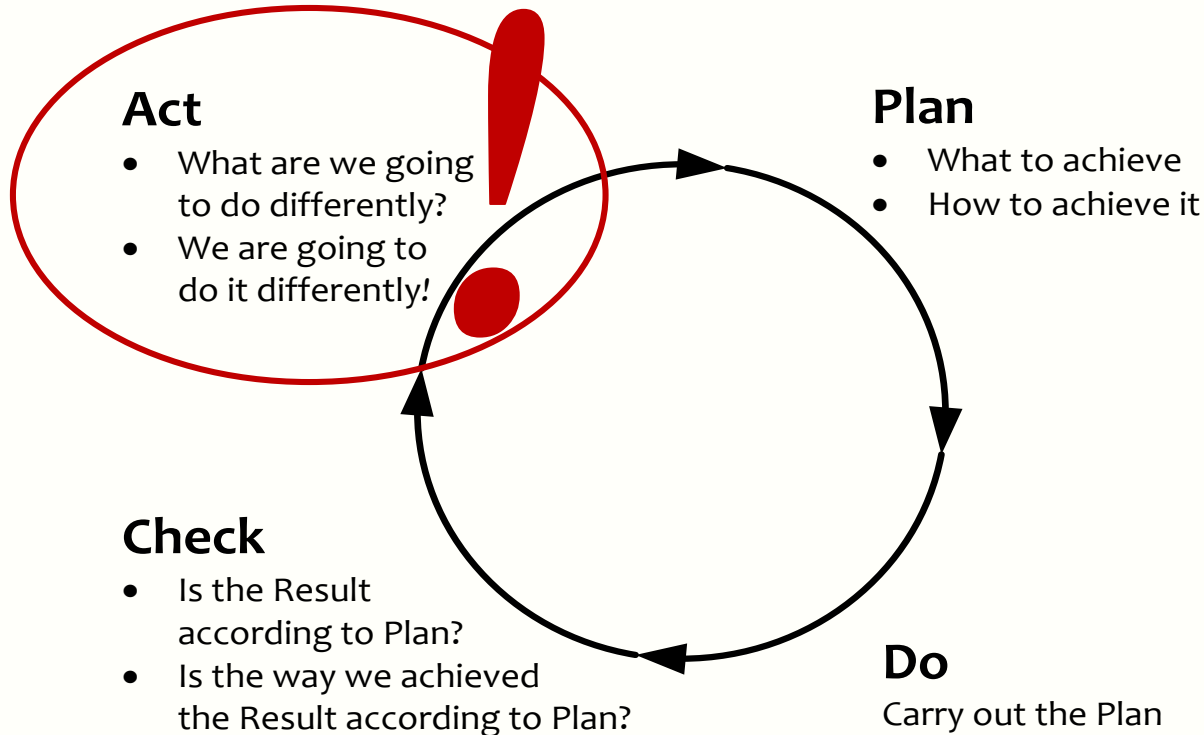
The essential ingredient: the PDCA Cycle

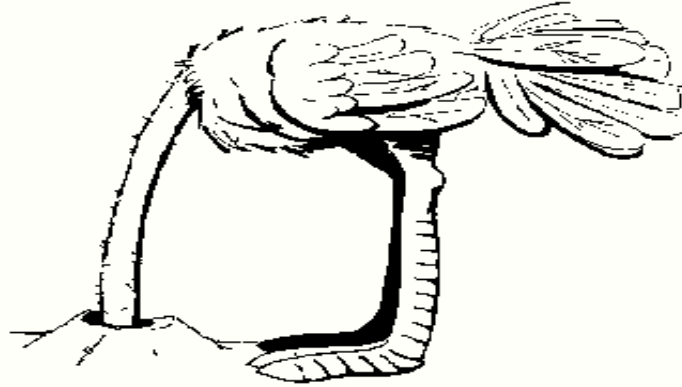
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)

www.malotaux.eu/?id=PDCA



Deming





Instead of complaining about a problem ...

(Stuck in the Check-phase)

Let's do something about it !

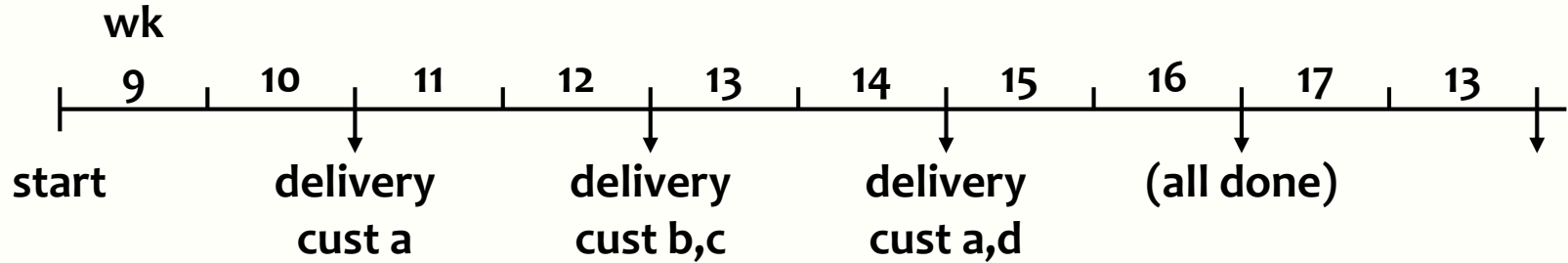
(Moving to the Act-phase)

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	
	totals	106	47	32	36		

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 ?

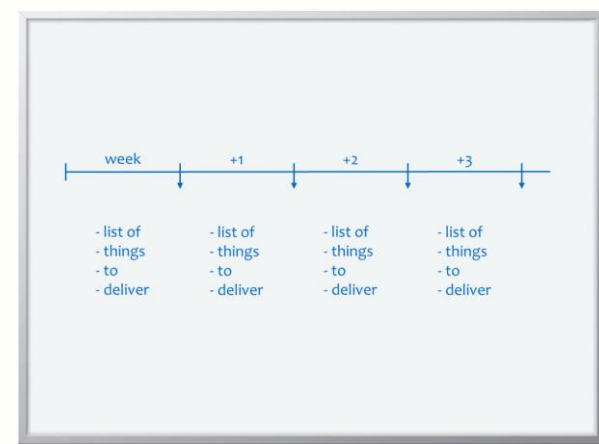
Line	Activity	Estim	Alternative	Junior tester	Developers	Customer	Will be done (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	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	
	totals	106	47	32	36		

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

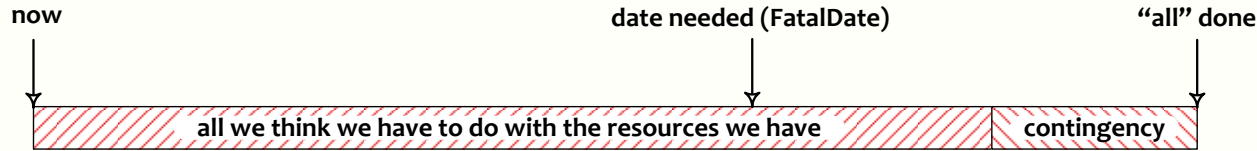
Later:

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

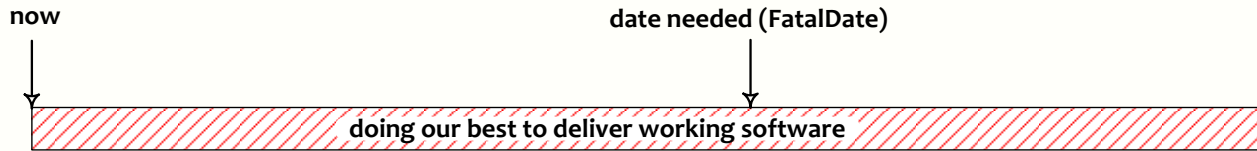


TimeLine

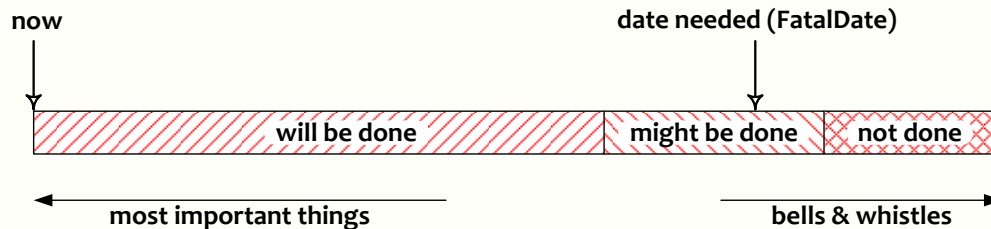
How do we know that we get and do *what* is needed, when it's needed ?



Standard Projects



Agile



Evo

- Better 80% 100% done, than 100% 80% done
- Let it be the most important 80%

www.malotaux.eu/?id=timeline

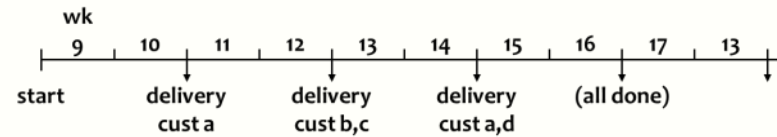
Fallacy of 'all' requirements

- “We’re done when all requirements are implemented”
- Is delivery time included ?
- Requirements are always *contradictory*
- Design is to find the optimum compromise between the conflicting requirements
- Do we really have focus on the real requirements ?
- Did the customers define real requirements ?
 - Usually even less trained in defining real requirements than we are
- What we think we have to do should fit the available time
- Instead of *letting* it happen, better decide *how* it will happen

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		



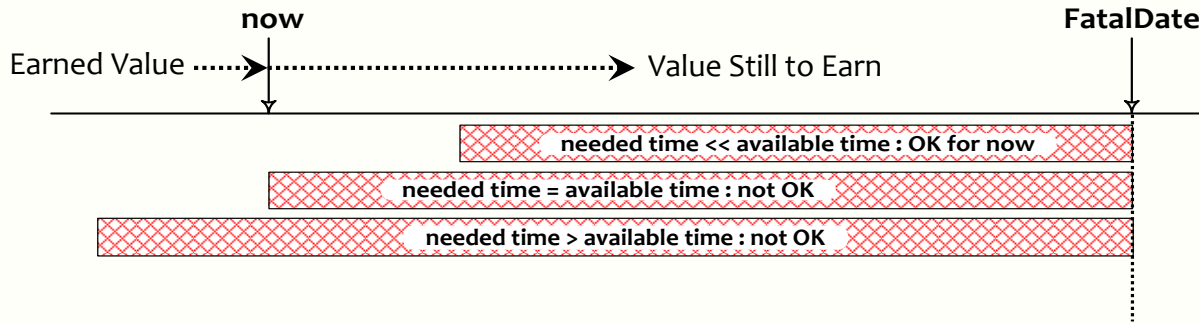
TimeLine: Predicting *what* may be done when

$$21/15 = 1.4$$

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	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 ?

What are we going to do about it ?



Failure is not an option

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



Deceptive options

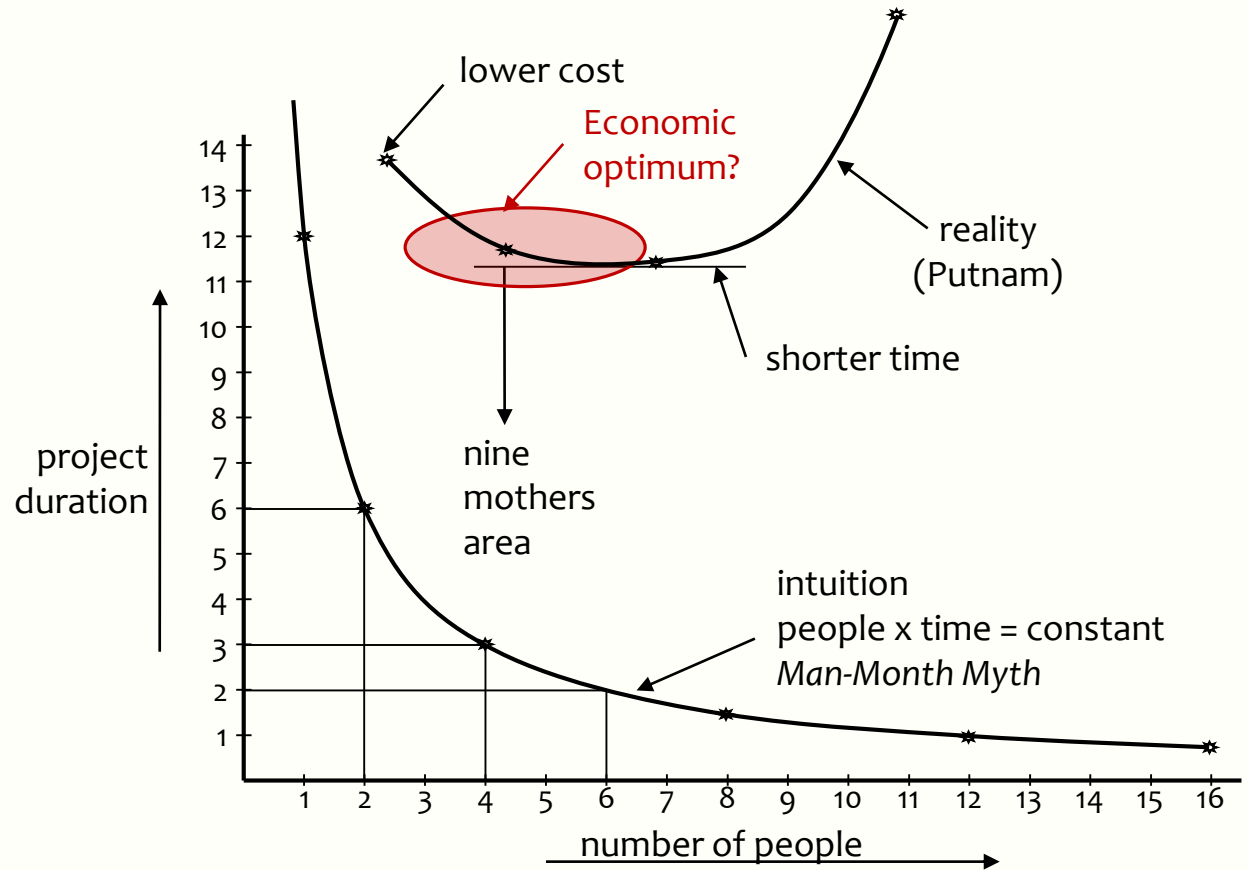
www.malotaux.eu/?id=options

- **Hoping for the best** (fatalistic)
- **Going for it** (macho)
- **Working Overtime** (fooling ourselves and the boss)
- **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

Adding people ?



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

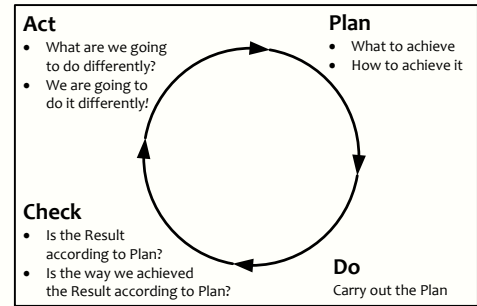




Saving time

Continuous
elimination of waste
(www.malotaux.eu/?id=essenceoflean)

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

(www.malotaux.eu/?id=evo)

- **Efficiency in how we do it** - doing things differently

- **The product**

- Using proper and most efficient solution, instead of the solution we always used

(www.malotaux.eu/?id=designlog)

- **The project**

- Doing the same in less time, instead of immediately doing it the way we always did

(www.malotaux.eu/?id=projectmanagement)

- **Continuous improvement and prevention processes**

- Constantly learning doing things better and overcoming bad tendencies

(www.malotaux.eu/?id=PDCA)

- **Efficiency in when we do it** - right time, in the right order

- **TimeBoxing** - much more efficient than FeatureBoxing

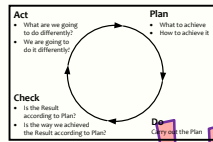
(www.malotaux.eu/?id=timeboxing)

Evolutionary Project

Management elements (Evo)

- Tom Gilb

www.malotaux.eu/?id=processes



- **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

Why

What

How much
Are we done

How

Check and learn
as early as possible

Zero
Defects
Attitude

Quality On Time

Evo Project Planning - Niels

Efficiency
of what we do

Effectiveness
of what we do

- **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?

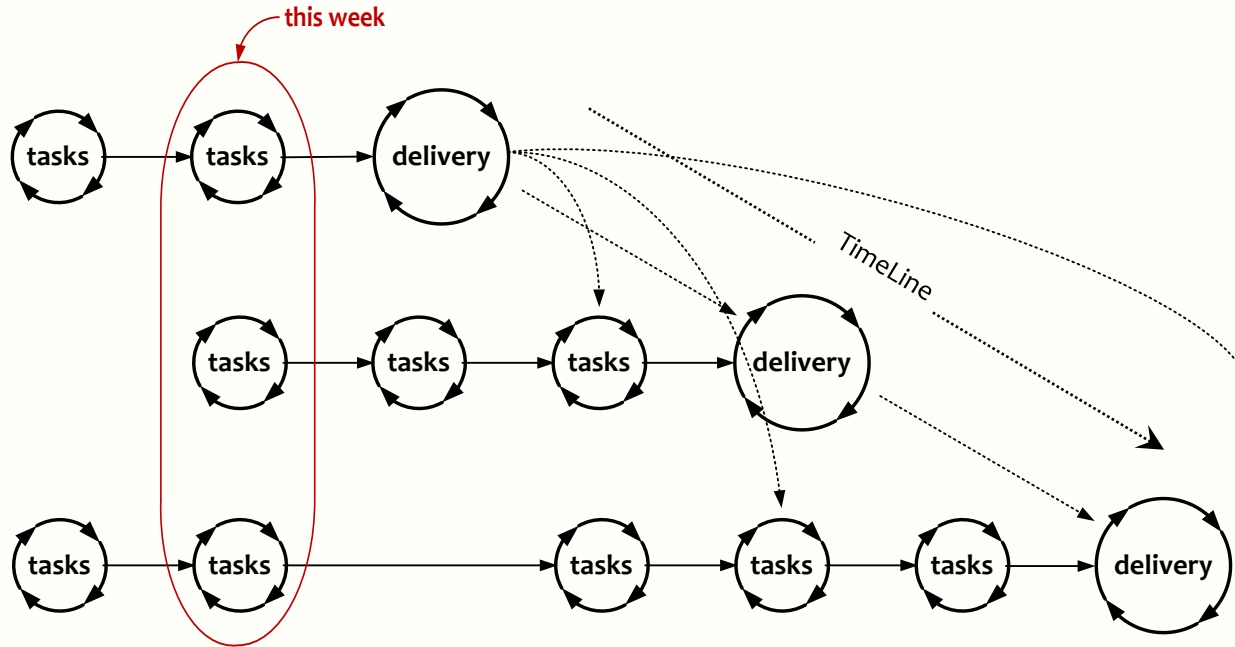
Weekly TaskCycle

- 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
 - What will generate the optimum feedback
 - We deliver only to eagerly waiting stakeholders
 - 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

Tasks feed Deliveries



Weekly planning

- **Individual preparation**
 - Conclude current tasks
 - What to do next
 - Estimations
 - How much time available
- **Modulation with / coaching by Coach / Team Lead / Peer(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** (will it be done by the end of the week ?)
 - **Commitment and decision**
- **Synchronization with group (team meeting)**
 - **Formal confirmation** (this is what we plan to do)
 - **Concurrency** (do we have to synchronize ?)
 - **Learning**
 - **Helping**
 - **Socializing**

Weekly Plan

- How much time do we have available
- $\frac{2}{3}$ 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

Task _a	2	
Task _b	5	
Task _c	3	
Task _d	6	do
Task _e	1	
Task _f	4	
Task _g	5	26
<hr/>		
Task _h	4	
Task _j	3	
Task _k	1	do not

$\frac{2}{3}$ is default start value
this value works well in development projects

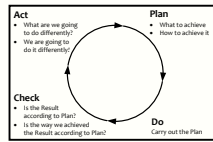
cycle	who	task description	estim	real	done	issues
3	John	<i>Net time available: 26</i>				
		aaaaaaaa	3	3	yes	
		bbbbbbbb [Paul]	1			
		cccccccc	5	13	yes	
		dddddddd	2			
		eeeeeeee	3	2		
		fffffffffff	2	1		
		ggggggggg	6	7	yes	
		hhhhhhhhh	4			
			26	26		
4	John	<i>Net time available: 26</i>				
		jjjjjjjjjjjj	3			for proj x
		kkkkkkkkkk	1			for proj x
		mmmmm	5			for proj x
		nnnnnnnn	2			for proj x
		pppppppp	3			for proj y
		qqqqqqqq	12			for proj y
		rrrrrrrrrr	6			for proj y
		sssssssss	4			for proj y
		ttttttttt	4			for proj y
			40			

TaskCycle Analysis
(retrospective)

learning

TaskCycle Planning
(presepective)

- **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



Why

What
How much
Are we done

How

Check and learn
as early as possible

Zero
Defects
Attitude

Evolutionary Project Management elements (Evo)

www.malotaux.eu/?id=processes – Tom Gilb

- **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

Why is this important ?

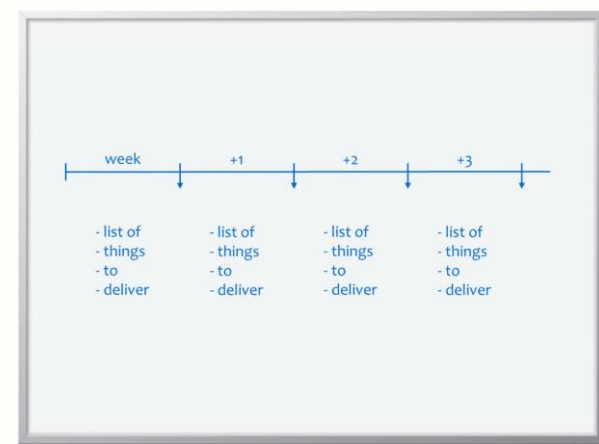
- TaskCycle Planning is not just planning the work for the coming week
- It exposes issues immediately
- Half of what people do in projects later proves not to have been necessary
- During the TaskCycle planning we can very efficiently see
 - What our colleagues think they're going to do
 - Make sure we're all going to work on the most important things
 - Not on unnecessary things
 - In line with the architecture and design
 - Leading most efficiently to the goal of the delivery
 - Everyone knows exactly what's going to happen, what not, and why

Would you like the same ?

- 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



Focus on the Result,

then think how to achieve that result successfully and efficiently

Delivering
Quality On Time
the Right Result
at the Right Time

Help !

Problem Solved

We have a QA Problem !

Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/conferences

www.malotaux.eu/booklets - booklet#8



- 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

