Evolutionary Planning

Producing even more in less time

www.malotaux.nl/conferences

Niels Malotaux

N R Malotaux Consultancy

+31 655 753 604

niels@malotaux.nl

www.malotaux.nl

Niels Malotaux

- Team and 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
- Helping teams to shine

Result Management

Who is doing what ?

- Developer ?
- Tester ?
- Architect ?
- Product Owner ?
- Scrum Master ?
- Team Member ?
- Customer ?
- Manager ?
- Consultant ?
- Coach ?

Who's responsible?

Everyone in the team !

Did you prepare ?

- The Goal of your current work or project (What and why are you working on it?)
- The Definition of Success (How will you know you're successfully done?)
- The most important stakeholder of your current work or project
  (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what will you do about it? - Failure is not an option)
- What you think you should and can do the coming week to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- Any issues you expect with the above or otherwise with your work or project
  During the conference we can discuss more (e.g. at AskTheExpert and Coaching sessions)

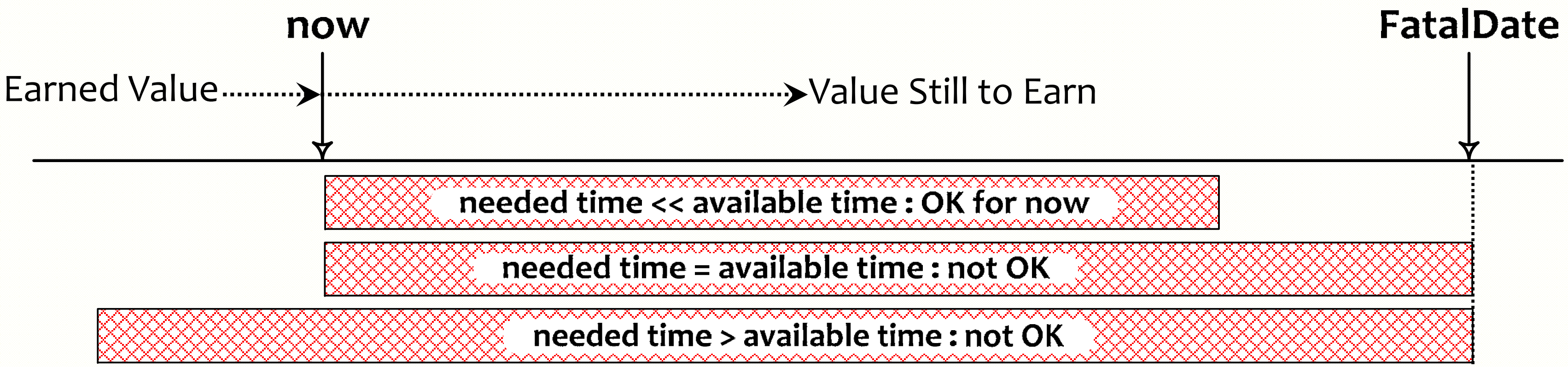
Universal Goal

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

Any Deadlines ?



- Value Still to Earn

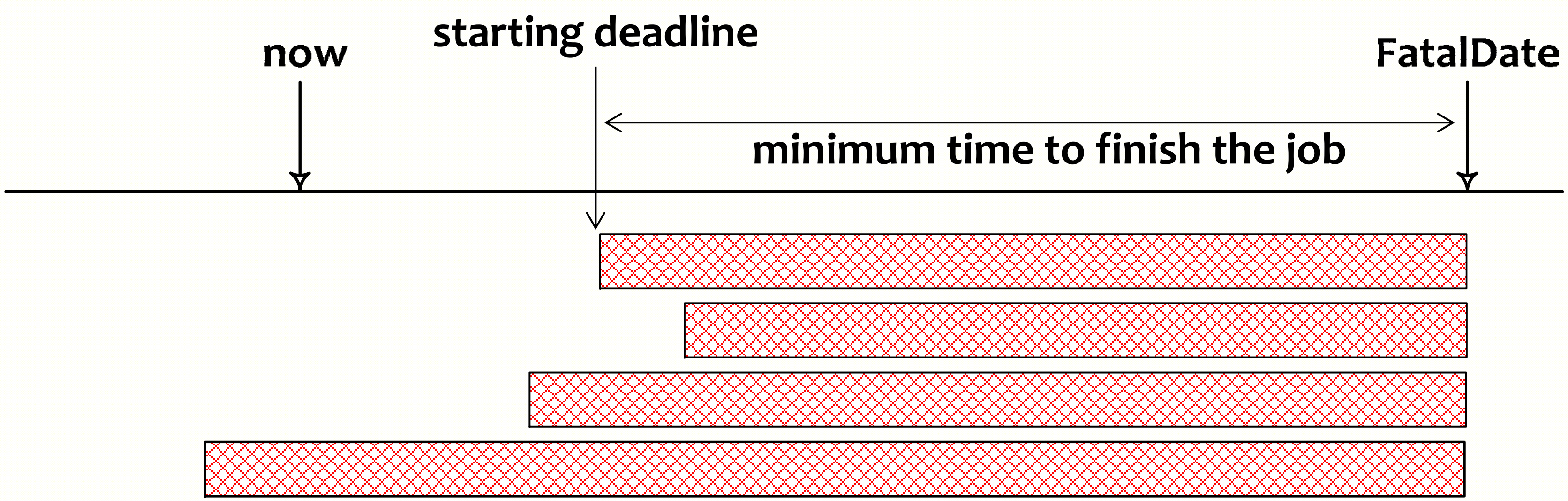
versus

- Time Still Available

If the match is over, you cannot score a goal

Even more important: *Starting Deadlines*

- Starting deadline
  - Last day to start to make the finish deadline
  - Every day we start later, we will end later



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

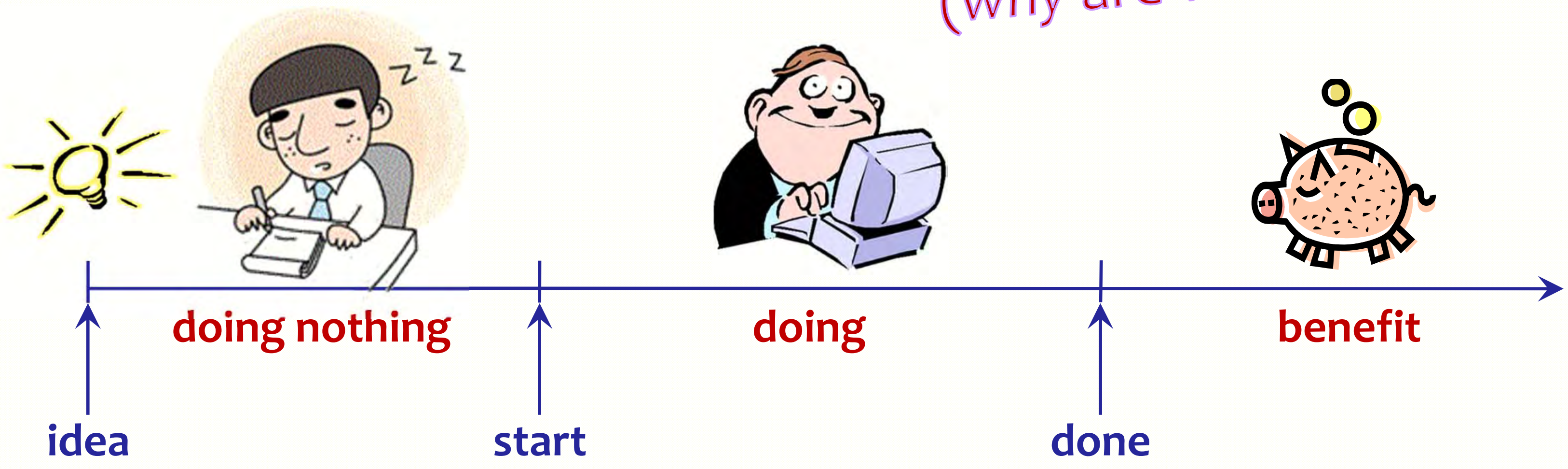
- What is the cost of the project per day ?
- Do you know how much you cost per day?
  Note: that's not what you get !
- If you don't know the benefit, assume 10 times the cost of the project
- 0th order estimations are good enough

- Do you know the benefit of your project ?
- Do you know the penalty for delay ?

The Importance of Time

Business Case

(why are we doing it)



This is why project time is usually more important than project budget

Return on Investment (ROI)

+ Benefit of doing - huge (otherwise we should do an other project)
– Cost of doing - project cost, usually minor compared with other costs
– Cost of being late - lost benefit
– Cost of doing nothing yet - every day we start later, we finish later

Delivery time is a Requirement

- Delivery Time is a Requirement,
  like all other Requirements
- How come most projects are late ???
  - Can Agile be late ?
- Apparently all other Requirements
  are more important than Delivery Time
- Are they really ?
- How about your current project ?

Did anyone tell you to go faster ?

- Produce more ! → bad quality → produce less
- Produce quality ! → produce more

Quick delivery of a solution that doesn't work means *no delivery*

The problem is: it's counter-intuitive

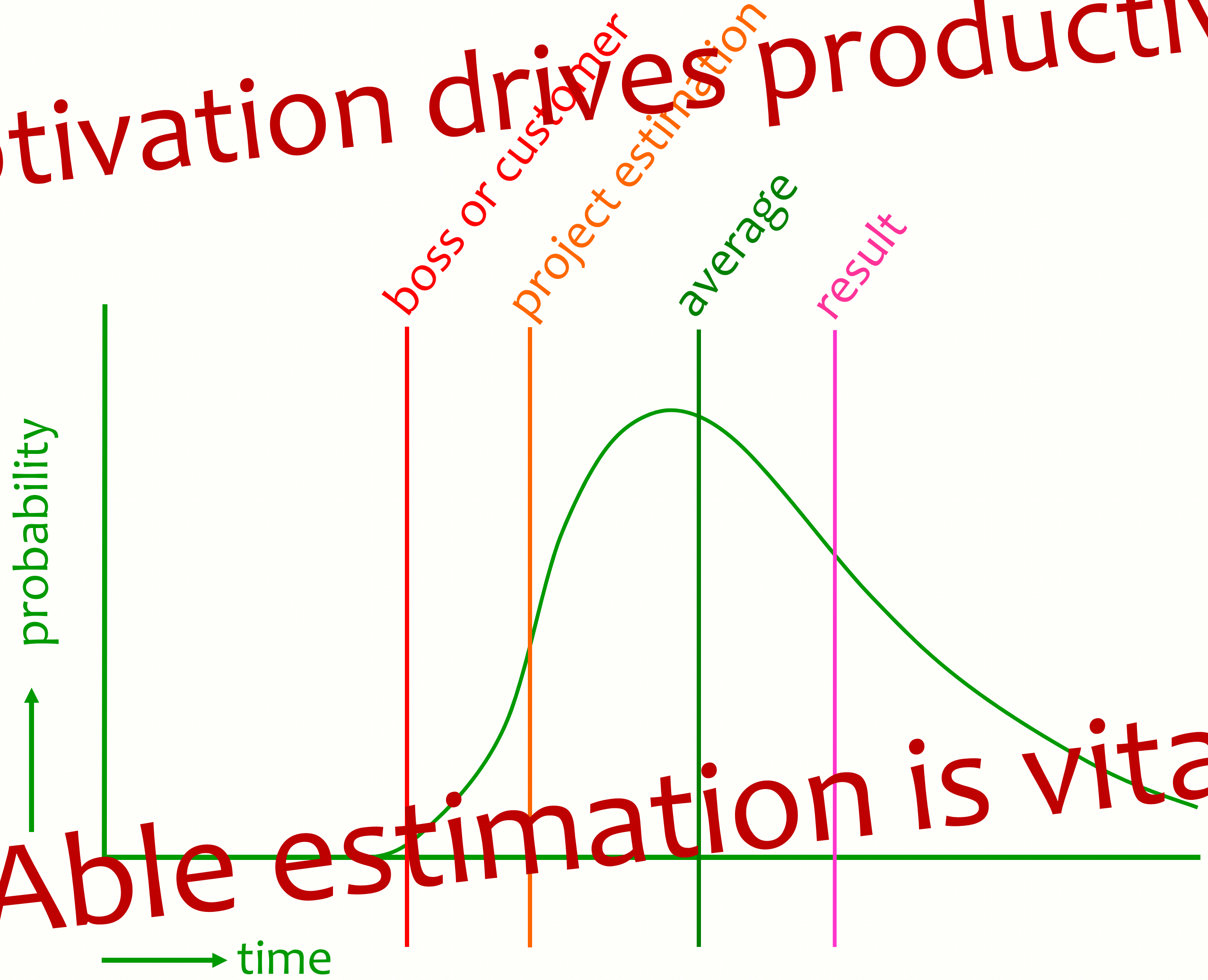
The challenge

Failure is not an option

- Getting and keeping the project under control
- Never to be late
- If we are late, we *failed*
- No excuses
- Not stealing from our customer's (boss') purse
- The only justifiable cost is the cost of doing the right things at the right time
- The rest is *waste*
- Who would enjoy producing waste ?

Lead time

Motivation drives productivity



Able estimation is vital

Estimation Exercise

Are you an optimistic or a realistic estimator?

Let's find out !

Project:

Multiplying two numbers of 4 figures

Example

0000	
0000	x
00000000	

How many seconds would you need to complete this Project?

Just your mind, pen, and paper

No electronics ! – otherwise you would spoil the exercise for yourself

Is this what you did?

Defect rate

- Before test ?
- After test ?

Alternative Design (*how* to solve the requirement)

Another alternative design

There are usually more,
and possibly better solutions
than the obvious one

What was the real requirement ?

Assumptions, assumptions ...

Better assume that many assumptions are wrong

Check !

Elements in the exercise

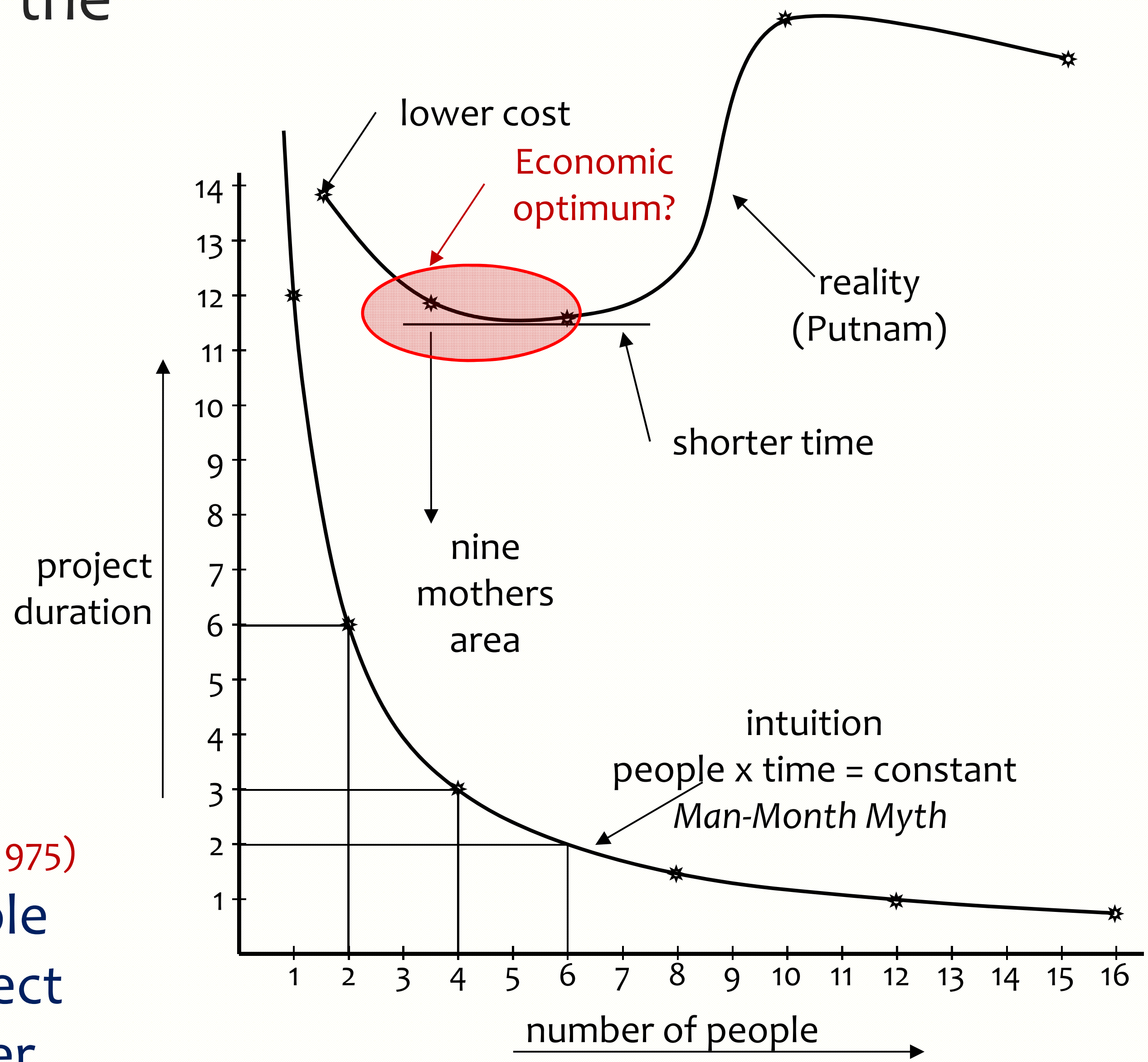
- Estimation, optimistic / realistic
- Interrupts
- Test, test strategy
- Defect-rate
- Design, design strategy
- Requirements
- Real Requirements
- Assumptions

How can we be On Time ?

Deceptive and difficult options to be on time

- Deceptive 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
- Difficult (but sometimes necessary) option
  - Adding people

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
  - Can we do less, without doing too little - doing exactly what is necessary
  - *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

Do you use Retrospectives ?

Do we really learn from what happened ?

Insanity is doing the same things over and over again
and hoping the outcome to be different *(let alone better - Niels)*
Albert Einstein 1879-1955, Benjamin Franklin 1706-1790, it seems Franklin was first

Only if we *change* our way of working,
the result may be *different*

- Hindsight is easy, but reactive
- Foresight is less easy, but proactive
- Reflection is for hindsight and learning
- Preflection is for foresight and prevention

Only with *prevention* we can save precious time

This is used in the Deming or Plan-Do-Check-Act cycle

The essential ingredient: the PDCA Cycle

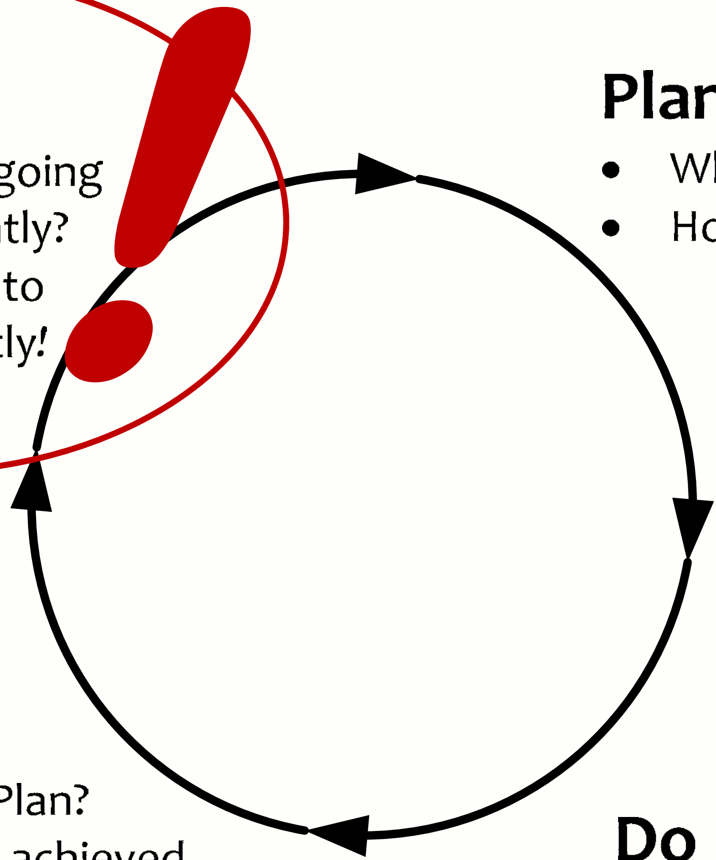
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)

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

Evolutionary Project Management elements (Evo) – Tom Gilb

- 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

Zero Defects Attitude

Check as early as possible

Right Result

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

Quality On Time

Efficiency of what we do

Right Time

Effectiveness of what we do

What will happen and **what will we do about it ?**

Do you have examples of requirements ?

Is this a Requirement ?

(Richard Smith)

(I call it: ‘nice input’, to be taken seriously)

design

need

“Create a new ‘Price Sentinel’ component that can detect if the bank’s published customer quotations go off-market, and then to immediately cancel all current quotations.”

how “immediately ?”

need

How “off” to warrant detection ?

“As a <role>, I want <goal/desire>, so that <benefit>”

Ref http://rsbatechnology.co.uk

So that … - using 5 Whys

Why do you need a "Price Sentinel" ?

1. To prevent publishing off-market tradable prices
2. To prevent trading loss
   (having to buy at a higher price than the bank offered to the customer)
3. To demonstrate to senior management that e-trading business can safely (no unexpected loss) manage customer trading
4. To ensure that senior management will agree to expand e-trading business in the future, based on current business performance to other customer segments and business areas
5. To meet business medium / long-term financial targets

Ref http://rsbatechnology.co.uk

First try

New ‘Price Sentinel’ component:

- detect if the bank’s customer quotations go off-market
- then immediately cancel all current quotations

- Off-market
  - ?? – Our margin less than 0.1% ?? – Will have to investigate
- Cancelling all current quotations
  - Scale: seconds after <detection>
  - Current: 600 sec (10 min)
  - Goal: 1 sec

Prioritize solutions by Impact Estimation

	Kill button	Price Sentinel
Cancel 600 → 1 sec	10.5 sec (note) 98%	1 sec 100%
Cost	1 day	30 day (6 sprint)
Note: 10 sec human recognition time, 0.5 sec cancel time		

Requirements have Rules

Some examples:

Rule 1: All quality requirements must be expressed *quantitatively*

Rule 2: No design (solutions) in the requirements

Rule 3: Unambiguous

Rule 4: Clear to test

Can you build this ?

- The system should be extremely user-friendly
- The system must work exactly as the predecessor
- The system must be better than before
- It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality
- It shall be reasonably easy to recover the system from failures, e.g. without taking down the power

Requirements with Planguage

ref Tom Gilb

Definition:

Specific

Measurable

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):

Attainable

Past:	2 min [minimum, 2015], 8 min [average, 2015], 83 min [max, 2015]
Current:	< 4 min [competitor y, Jan 2015] ← <who said this?>, <Survey Dec 2015>
Record:	57 sec [competitor x, Jan 2012]
Wish:	< 2 min [2018Q3, new system available] ← CEO, 19 Jan 2016, <document ...>

Requirements:

Time

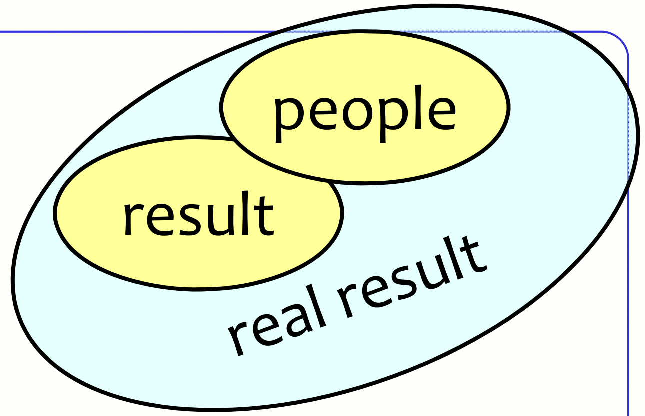
Realizable

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

Tom Gilb quote

- The fact that we can set numeric objectives, and track them, is powerful; *but in fact it is not the main point*
- The main purpose of quantification is to force us to *think deeply*, and *debate exactly*, what we mean
- So that others, later, *cannot fail* to understand us

Stakeholders are (not only) people



- Every project has some 30±20 Stakeholders
- Stakeholders have a stake 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 !*
- Project risks, happening in almost every project
- No excuse to fail !

No Stakeholder?

- 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?

Did anyone prepare ?

- The Goal of your current work or project (What and why are you working on it?)
- The Definition of Success (How will you know you're successfully done?)
- The most important stakeholder of your current work or project
  (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what will you do about it? - Failure is not an option)
- What you think you should and can do the coming week to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- Any issues you expect with the above or otherwise with your work or project
  During the conference we can discuss more (e.g. at AskTheExpert and Coaching sessions)

Human Behavior

Human Behavior

- Systems are conceived, designed, implemented, maintained, used, and tolerated *(or not)* by people
- People react quite predictably
- However, often differently from what we intuitively think
- Most projects
  - ignore human behavior,
  - incorrectly assume behavior,
  - or decide how people should behave *(ha ha)*
- To succeed in projects, we must study and adapt to real behavior rather than assumed behavior
- *Even if we don't agree with that behavior*

Discipline

- Control of wrong inclinations
- Even if we know how it should be done …
  (if nobody is watching …)
- Discipline is very difficult
- Romans 7:19
  - The good that I want to do, I do not …

→ Helping each other (watching over the shoulder)

→ Rapid success (do it 3 weeks for me…)

→ Making mistakes (provides short window of opportunity)

→ Openness (management must learn how to cope)

Intuition

- Makes us react on every situation
- Intuition is fed by experience
- It is free, we always carry it with us
- We cannot even turn it off
- Sometimes intuition shows us the wrong direction
- In many cases the head knows, the heart not (yet)
- Coaching is about redirecting intuition

Communication

- Traffic accident: witnesses tell *their* truth
- Same words, different concepts
- Human brains contain rather fuzzy concepts
- Try to explain to a colleague
- Writing it down is explaining it to paper
- If it’s written it can be discussed and changed
- Vocal communication evaporates immediately
- E-mail communication evaporates in a few days

Perception

- Quick, acute, and intuitive cognition (www.M-W.com)
- What people say and what they do is not always the same
- The head knows, but the heart decides
- Hidden emotions are often the drivers of behavior
- Customers who said they wanted lots of different ice cream flavors from which to choose,
  still tended to buy those that were fundamentally vanilla

- So, trying to find out what the real value to the customer is, can show many paradoxes
- Better not simply believe what they say: check!

Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures
- It's always 'them' – How about 'us' ?

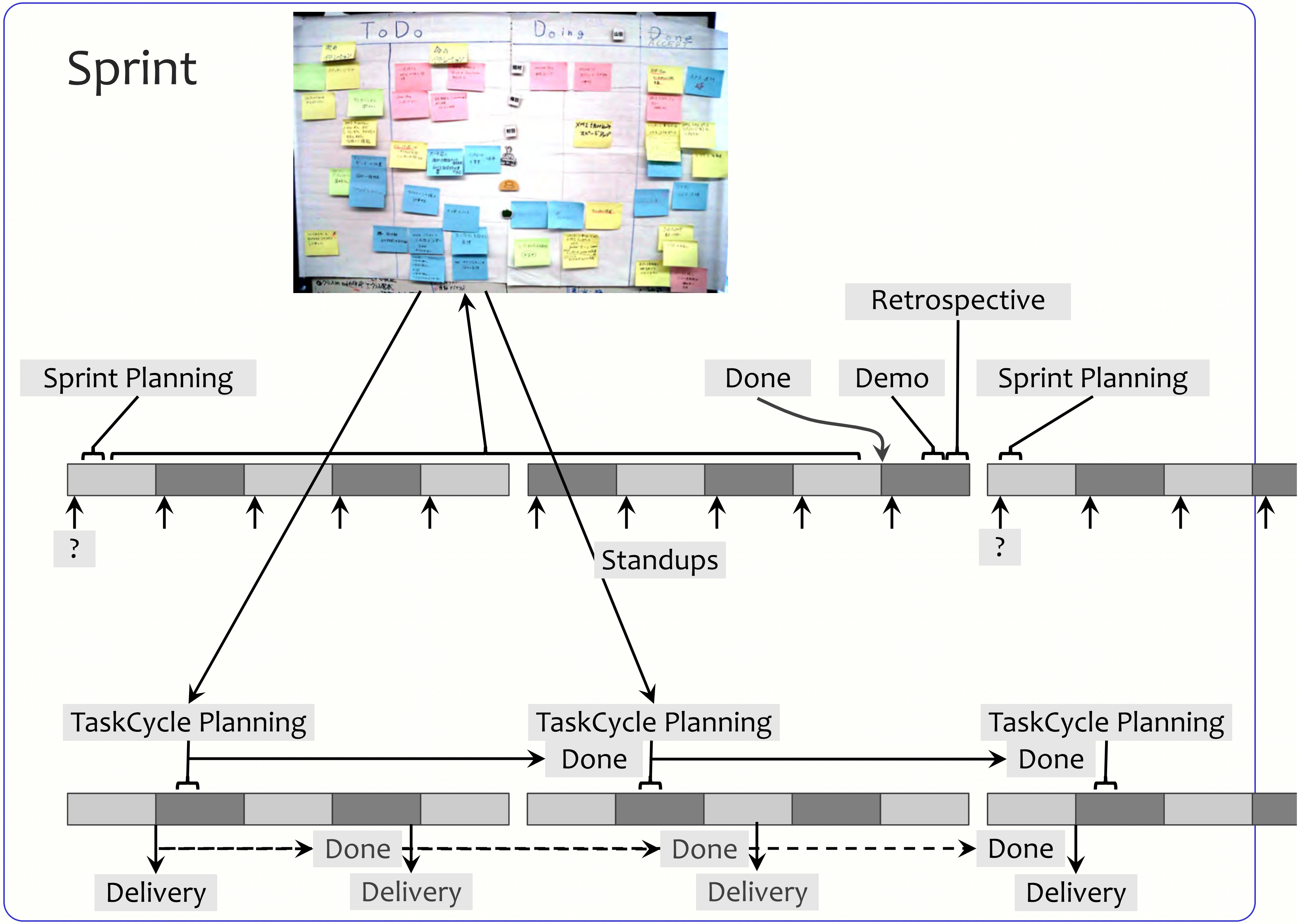
- At a Fatal Day, any excuse is in vain: we failed
- Even if we "really couldn't do anything about it"
- Failure is a very hard word. That's why we are using it !
- No pain, no gain
- We never say: "*You* failed" - Use: "*We* failed"
  - After all, *we* didn't help the person not to fail

Evolutionary Planning
prevention is better than cure

To-do lists

- Are you using to-do lists? → EXERCISE
  - List the things you have to do the coming week
    in order to achieve what you're supposed to achieve

Sprint



ToDo
Doing
Done
Retrospective
Sprint Planning
Done
Demo
Sprint Planning
?
Standups
?
TaskCycle Planning
TaskCycle Planning
Done
TaskCycle Planning
Done
Done
Done
Done
Delivery
Delivery
Delivery
Delivery

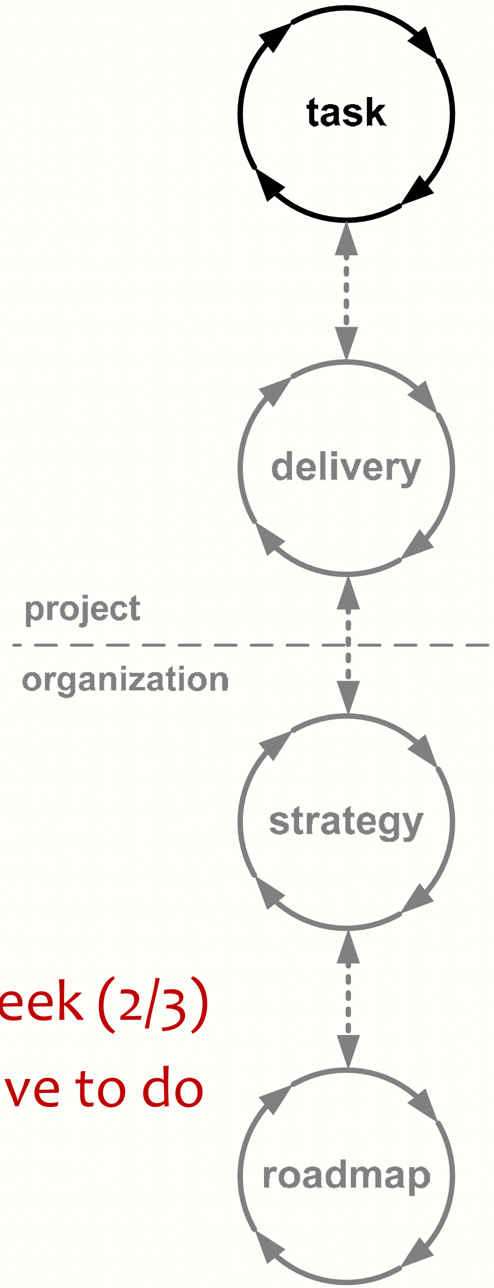
To-do lists

- Are you using to-do lists?
  - List the things you have to do the coming week
  - Did you add effort estimates?
  - Did you check how much time you have available the coming week ?
  - Does what you have to do fit in the available time ?
  - Did you check what you can do and what you cannot do?
  - Did you take the consequence?

- Evo:
  - Because we are short of time, we better use the limited available time as best as possible
  - We don’t try to do better than *possible*
  - To make sure we do the best possible, we *choose* what to do in the limited available time. We don’t just let it happen randomly

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



Effort and Lead Time

- Days estimation $\rightarrow$ lead time (calendar time)
- Hours estimation $\rightarrow$ effort

- Effort variations and lead time variations have different causes
- Treat them differently and keep them separate
  - Effort: complexity
  - Lead Time: time-management
    - (effort / lead-time ratio)

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

$Task_a$	2	
$Task_b$	5	
$Task_c$	3	
$Task_d$	6	do
$Task_e$	1	
$Task_f$	4	
$Task_g$	5	26
$Task_h$	4	
$Task_j$	3	do
$Task_k$	1	*not*

2/3 is default start value

this value works well in development projects

Did you prepare ?

- The Goal of your current work or project (What and why are you working on it?)
- The Definition of Success (How will you know you're successfully done?)
- The most important stakeholder of your current work or project
  (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what will you do about it? - Failure is not an option)
- What you think you should and can do the coming week to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work or project
  During the conference we can discuss more (e.g. at AskTheExpert and Coaching sessions)

Exercise

Task$_a$	2	
Task$_b$	5	
Task$_c$	3	
Task$_d$	6	do
Task$_e$	1	
Task$_f$	4	
Task$_g$	5	26
Task$_h$	4	
Task$_j$	3	do
Task$_k$	1	*not*

- 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

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 (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

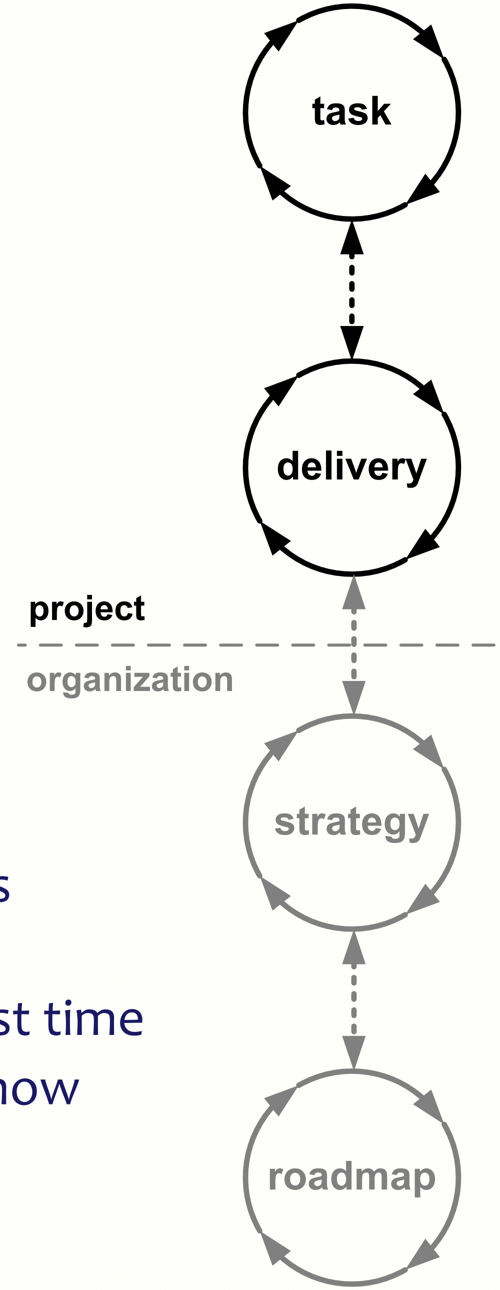
cycle	who	task description	estim	real	done	issues
3	John	*Net time available: 26*				
		aaaaaaaaa	3	3	yes	
		bbbbbbbb [Paul]	1			
		cccccccccc	5	13	yes	
		dddddddd	2			
		eeeeeeee	3	2		
		fffffffffff	2	1		
		ggggggggg	6	7	yes	
		hhhhhhhh	4			
			26	26		
4	John	*Net time available: 26*				
		jjjjjjjjjjjjjj	3			for proj x
		kkkkkkkkk	1			for proj x
		mmmmm	5			for proj x
		nnnnnnnn	2			for proj x
		pppppppp	3			for proj y
		qqqqqqqq	12			for proj y
		rrrrrrrrrrrr	6			for proj y
		ssssssssss	4			for proj y
		tttttttttttt	4			for proj y
			40			

TaskCycle Analysis
(retrospective)

learning

TaskCycle Planning
(presepective)

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

Delivery Strategy Suggestions (Requirements)

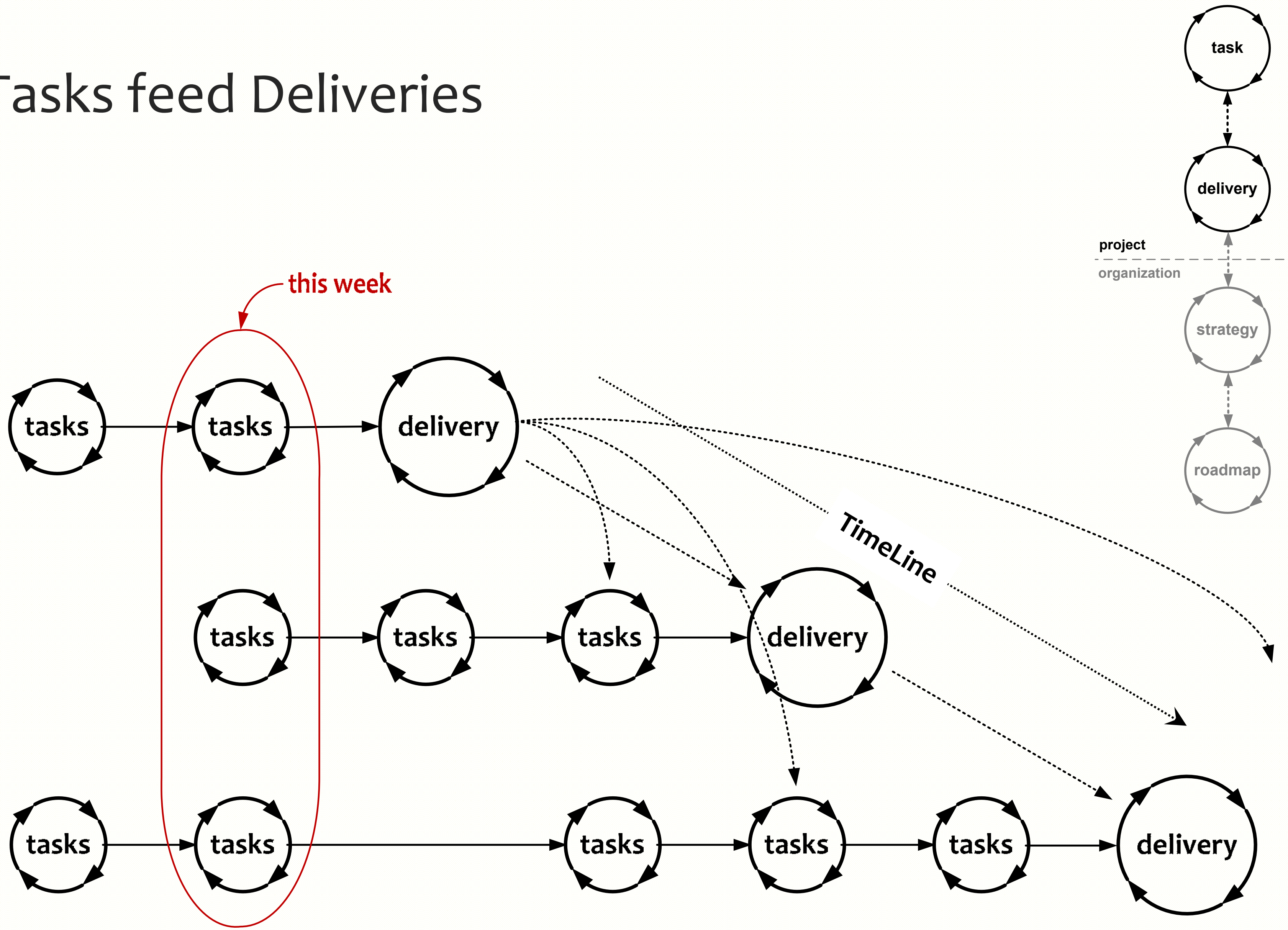
- What we deliver will be used by the appropriate users immediately, within one week not making them less efficient than before
- If a delivery isn't used immediately, we analyse and close the gap so that it will start being used (otherwise we don't get feedback)
- The proof of the pudding is when it's eaten and found tasty, by them, not by us
- The users determine success and whether they want to pay (we don't have to tell them this, but it should be our attitude)

Do you demo at the end of a Sprint ?

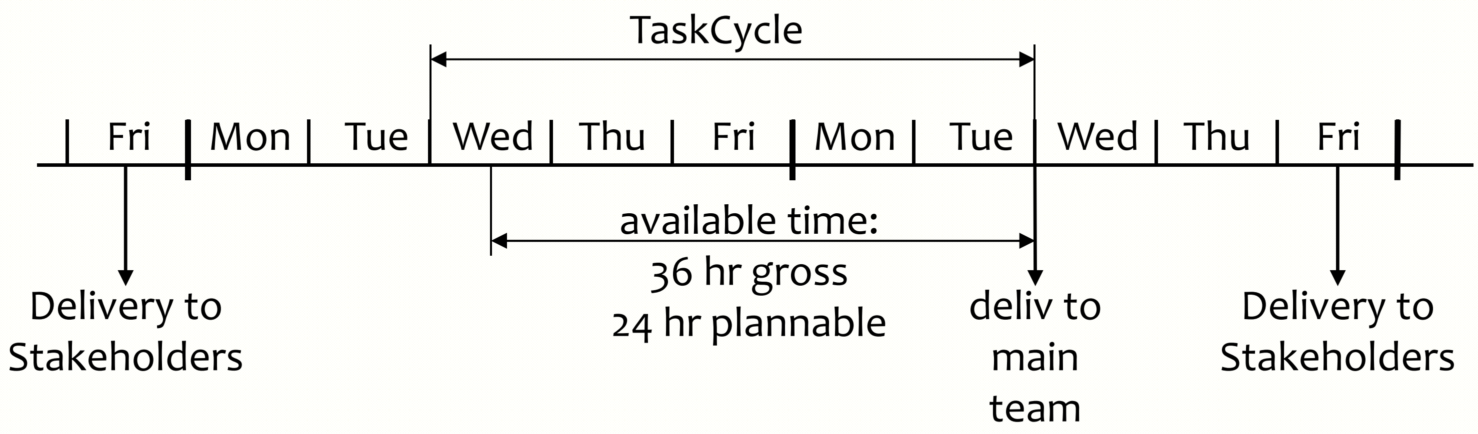
- Give the delivery to the stakeholders
- Keep your hands handcuffed on your back
- Keep your mouth shut
- and o-b-s-e-r-v-e what happens
- Seeing what the stakeholders actually do provides so much better feedback
- Then we can 'talk business' with the stakeholders

- Is this what you do ?
- Success criterion: "No Questions, No Issues"

Tasks feed Deliveries



Designing a Delivery



Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	42

Gregory (later)	
Draft design	0
Finish design	0
...	

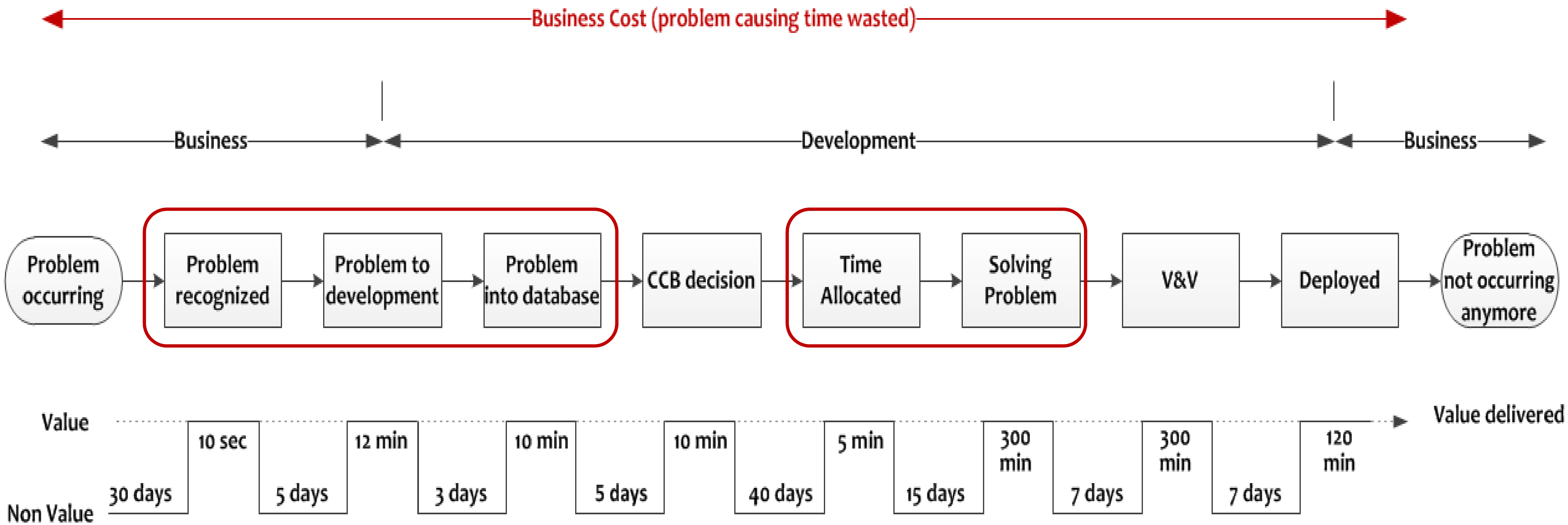
Jerome	
XMLa	3
XMLb	3
...	

Task $_a$	2	
Task $_b$	5	
Task $_c$	3	
Task $_d$	6	do
Task $_e$	1	
Task $_f$	4	
Task $_g$	5	26
Task $_h$	4	
Task $_j$	3	do
Task $_k$	1	*not*

TaskCycle Exercise

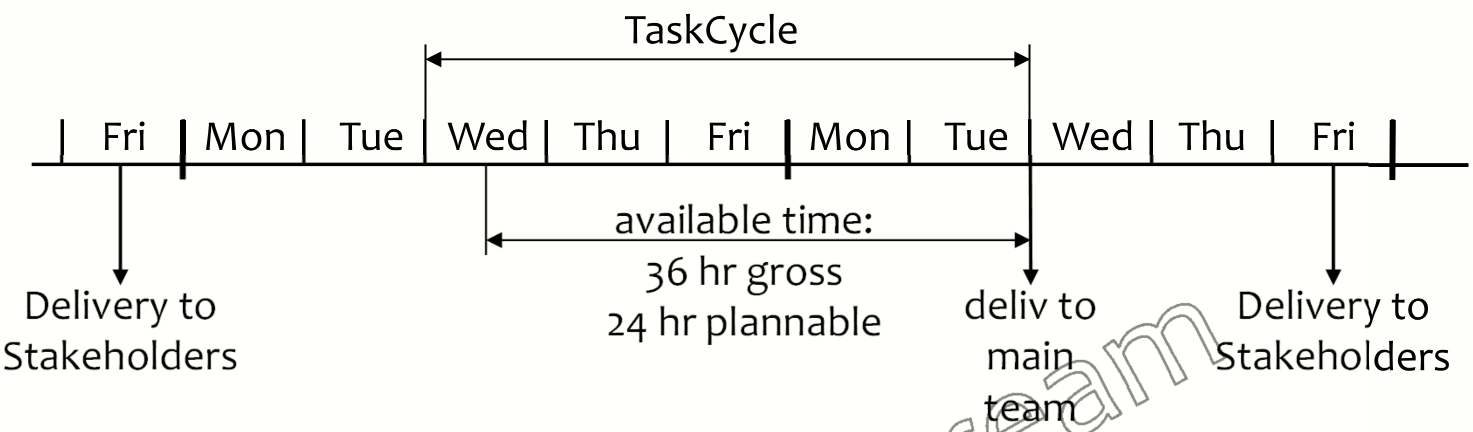
- How much time do you have available
- 2/3 of available time is net plannable time
- What is most important to do (update your list)
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr)
- What can you do, and what are you going to do
- What are you *not* going to do
- Why ?

Value stream mapping



- Total Business Cost 114 days, Cost of Non Value: 112 days
- Occurrence: 2 x per day, delay per occurrence: 10 min
- Number of business people affected: 100
- Business Cost of Non Value: 2 x 10 min x 112 days x 100 people x 400 €/day = 187 k€
- Net Cost of Value: 1.6 days → ~3 people x 1.6 days x 800 €/day = 5 k€

Designing a Delivery



Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

Gregory	
Draft design	0
Finish design	0
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	1
XMLb	1
Total	24

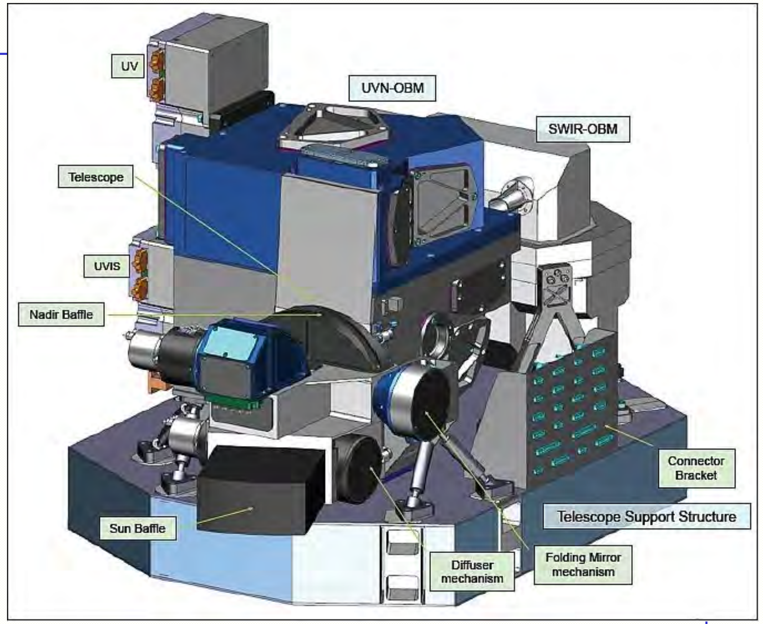
Gregory (later)	
Draft design	0
Finish design	0
...	

Jerome	
XMLa	3
XMLb	3
...	

Why is this important?

- Half (±30%) of what people do in projects later proves not having been necessary
- During the TaskCycle planning we can very efficiently see
  - What our colleagues think they're going to do
  - Make sure they're 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
- We'll see two cases where the architect led the project to success in record time

Earth Observation Satellite



- Very experienced Systems Engineers
- They use quantified requirements routinely
- They don't know exactly where they'll end up
- 10 year pure waterfall project (imposed by ESA)
- Only problem: They missed all deadlines
- 9 weeks later: They haven't missed any deadline since
- Recently: delivered 1 day early (instead of 1 year late)
- Savings: some 40 man-year
- How did they do that ?

Requirements weren't the problem

- Requirements for tropospheric O3
  - Ground-pixel size : 20 × 20 km2 (threshold); 5 × 5 km2 (target)
  - Uncertainty in column : altitude-dependent
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)
- Requirements for stratospheric O3
  - Ground-pixel size : 40 × 40 km2 (threshold); 20 × 20 km2 (target)
  - Uncertainty in column : altitude-dependent
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)
- Requirements for total O3
  - Ground-pixel size : 10 × 10 km2 (threshold); 5 × 5 km2 (target)
  - Uncertainty in column : 2%
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)

Awful schedule pressure !

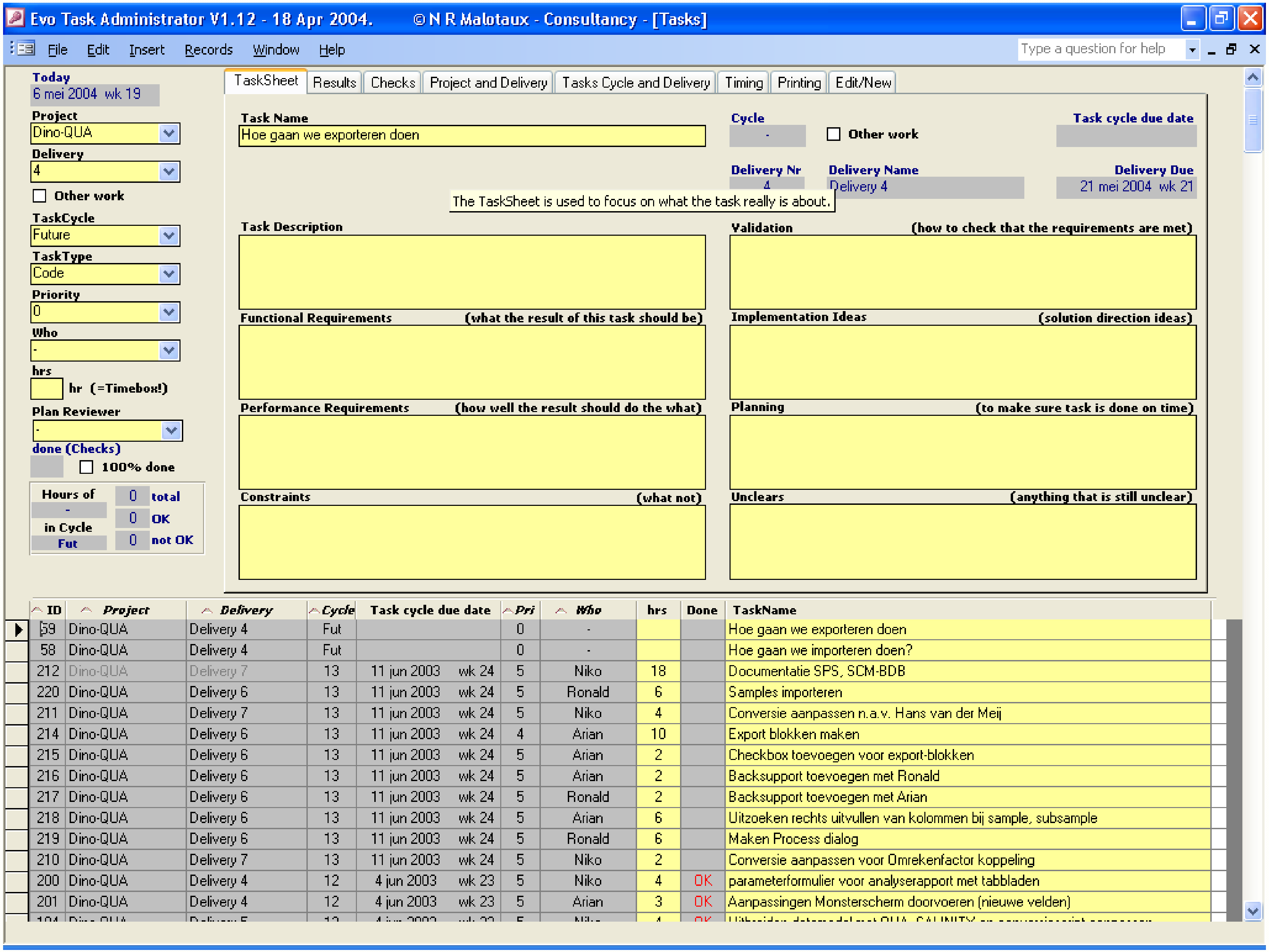
- Meeting with sub-contractors in three weeks
- Many documents to review
- Impossible deadline

- How many documents to review ?
- How much time per document ?

	per doc	hr
4 heavy	15	60
3 easy	2	6
	total	66
other work		33
	total	99

available	2 x 26	52

- Some suggestions …
- Result: well reviewed, great meeting, everyone satisfied



ID	Project	Delivery	Cycle	Task cycle due date		Pri	Who	hrs	Done	TaskName
59	Dino-QUA	Delivery 4	Fut			0	-			Hoe gaan we exporteren doen
58	Dino-QUA	Delivery 4	Fut			0	-			Hoe gaan we importeren doen?
212	Dino-QUA	Delivery 7	13	11 jun 2003	wk 24	5	Niko	18		Documentatie SPS, SCM-BDB
220	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Ronald	6		Samples importeren
211	Dino-QUA	Delivery 7	13	11 jun 2003	wk 24	5	Niko	4		Conversie aanpassen n.a.v. Hans van der Meij
214	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	4	Arian	10		Export blokken maken
215	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	2		Checkbox toevoegen voor export-blokken
216	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	2		Backsupport toevoegen met Ronald
217	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Ronald	2		Backsupport toevoegen met Arian
218	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	6		Uitzoeken rechts uitvullen van kolommen bij sample, subsample
219	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Ronald	6		Maken Process dialog
210	Dino-QUA	Delivery 7	13	11 jun 2003	wk 24	5	Niko	2		Conversie aanpassen voor Omrekenfactor koppeling
200	Dino-QUA	Delivery 4	12	4 jun 2003	wk 23	5	Niko	4	OK	parameterformulier voor analyserapport met tabbladen
201	Dino-QUA	Delivery 4	12	4 jun 2003	wk 23	5	Arian	3	OK	Aanpassingen Monsterscherm doorvoeren (nieuwe velden)

Developing a new oscilloscope

- 4 teams of 10 people, 8 more people in Bangalore
- Introduced first in one team
- Other teams followed once convinced
- One team lagged because fear of 'micro-management'

- Even if we would drop all you suggested, the 1-on-1's will be kept, because so powerful:
  - We used to do something and afterwards found out it wasn't what it should be
  - Now we find out before, allowing us to do it more right the first time

Results

- Schedule accuracy for this platform development was 50% better than the program average (as measured by program schedule overrun) over the last 5 years
- This product was the fastest time-to-market with the highest quality at introduction of any platform in our group in more than 10 years
- The team also won a prestigious Team Award as part of the company's Technical Excellence recognition program

www.malotaux.nl/doc.php?id=19 chapter 4.7.1, page 70

Example

- Polish software project
  - Deadline in 6 weeks
  - ‘Mission Impossible’
- After reorganizing
  - Delivered in 5 weeks to happy customer
  - No overtime !
- Magic question:
  - What do you have to deliver by the end of the week, and
  - What do you all have to do to achieve that ?

  - Many issues surface immediately !
  - To be solved before causing more problems

Quality on Time

- Evo development gradually delivers function and performance, while eating up resources
- Not just what to deliver, but also how we are going to deliver it and whether this is the right way to deliver it
- EvoPlanning prevents a lot of bad implementations before they are implemented, saving a lot of time

Now we are already much more efficient

- Organizing the work in very short cycles
- Making sure we are doing the right things
- Doing the right things right
- Continuously optimizing (what not to do)
- So, we already work more efficiently

but …

- How do we make sure the whole project is done on time ?

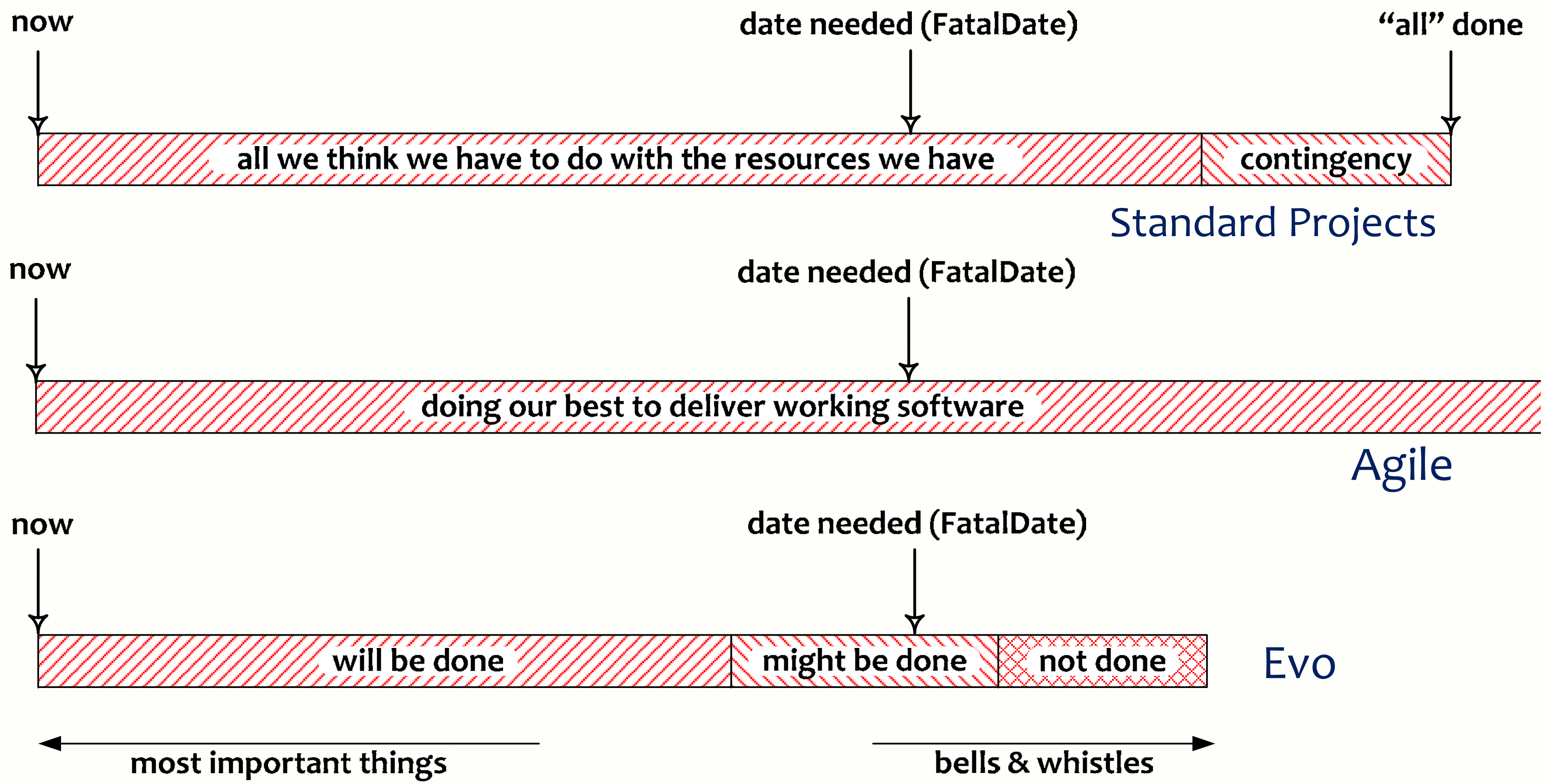
TimeLine

How to make sure we get

the Right Results at the Right Time

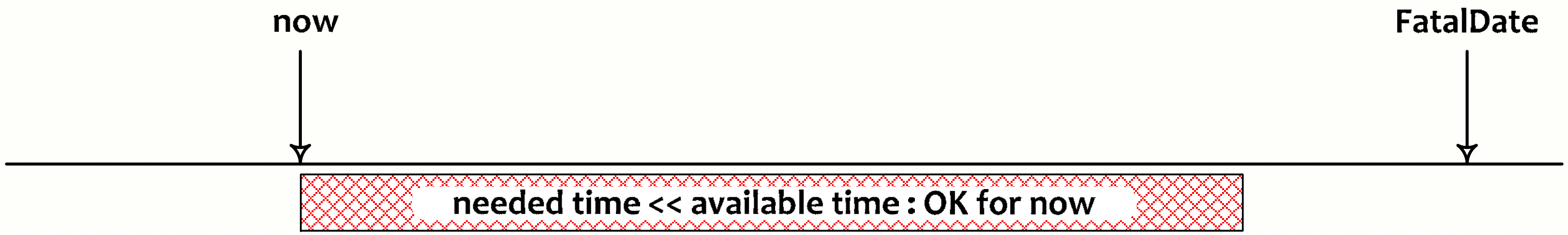
TimeLine

What the customer wants, he cannot afford

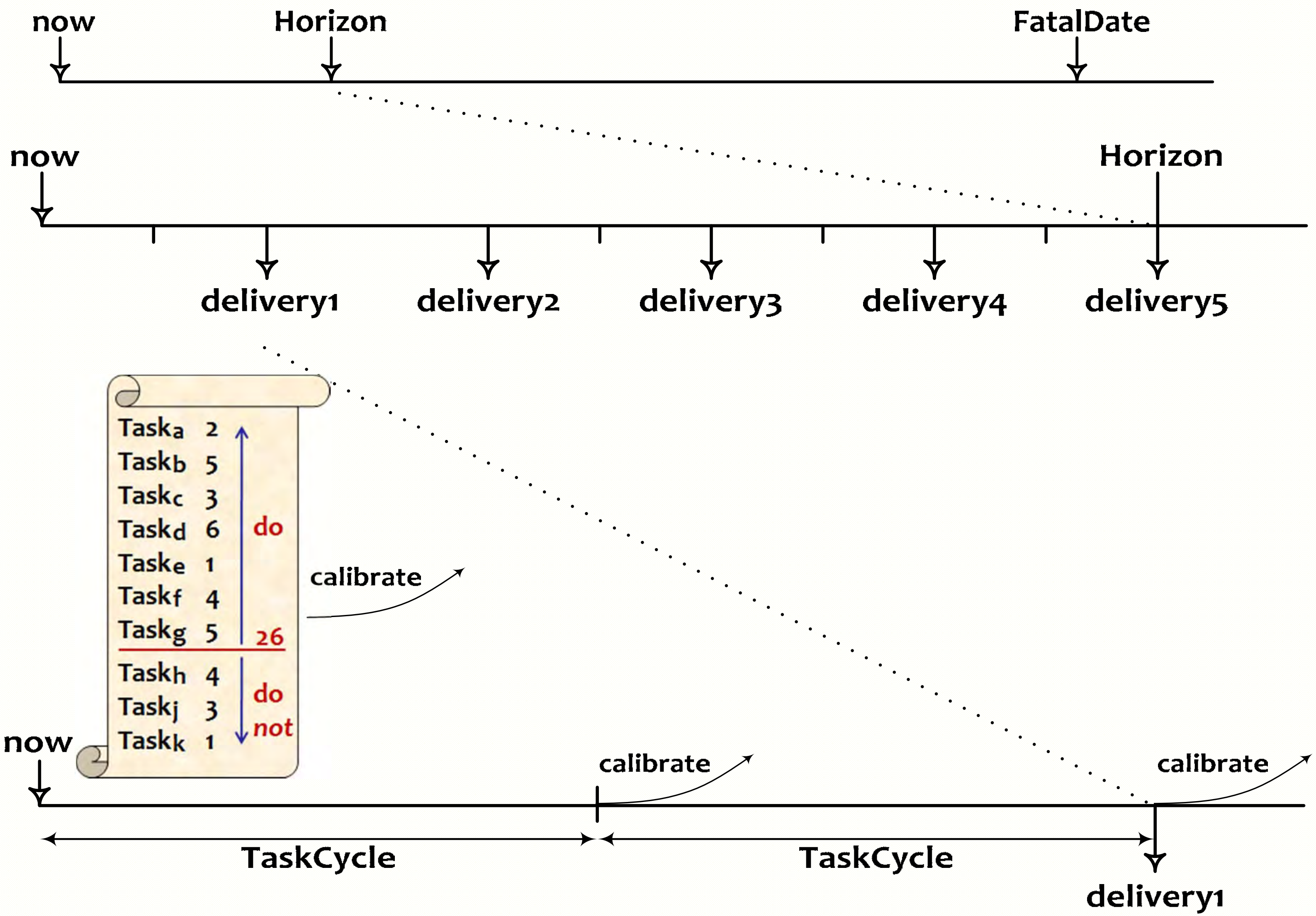


- Better 80% 100% done, than 100% 80% done
- Let it be the most important 80%

If it easily fits ...



Result to Tasks and back

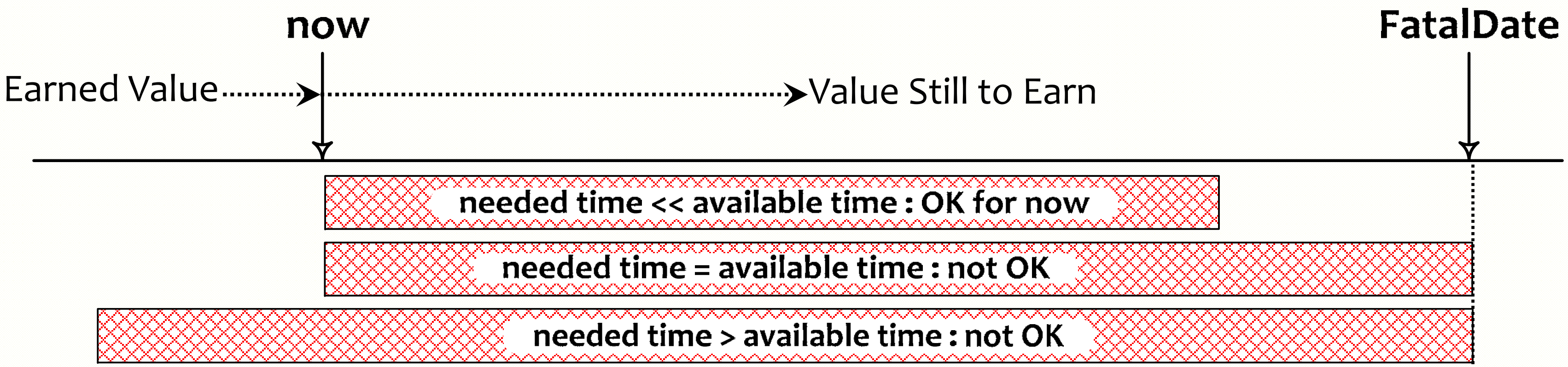


Sorry, picture removed for confidentiality

Sorry, picture removed for confidentiality

Sorry, picture removed for confidentiality

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



- Value Still to Earn

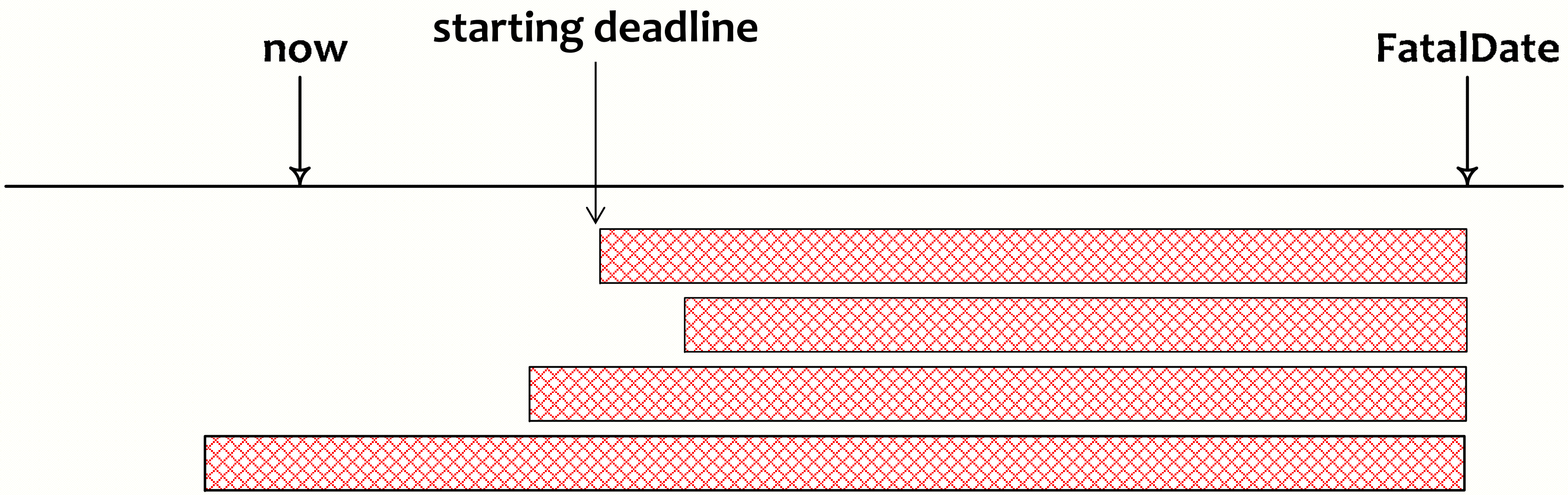
versus

- Time Still Available

If the match is over, you cannot score a goal

Starting Deadlines are even more important

- Starting deadline
  - Last day to start not to delay the finish deadline
  - Every day we start later, we will cause delay

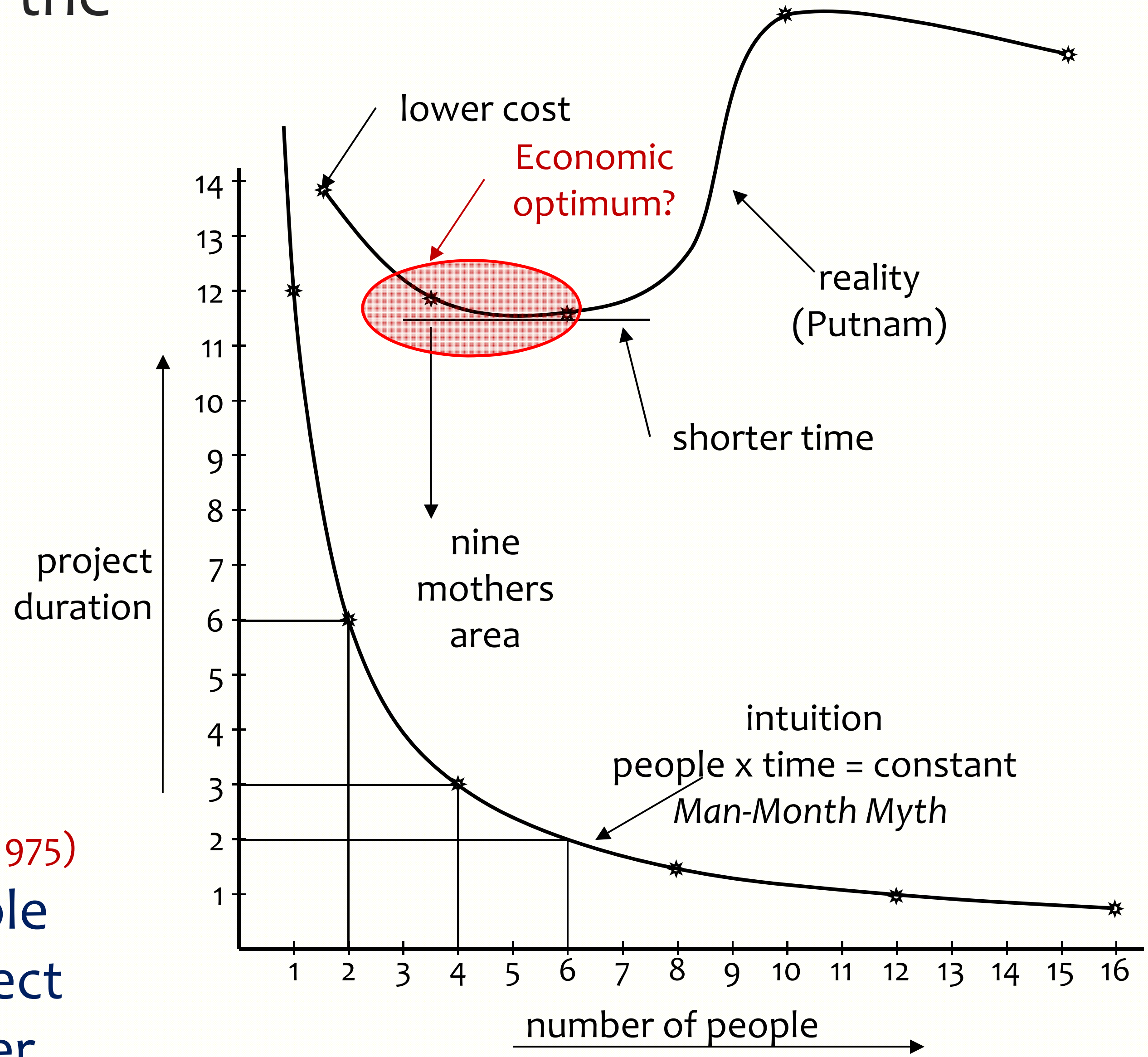


Deceptive options

- 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

Intuition often guides us into the wrong direction

The Myth of the Man-Month



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

Saving time

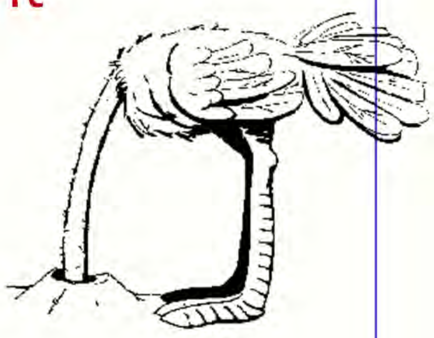
Continuous elimination of waste

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
  - Can we do less, without doing too little - doing exactly what is necessary
  - *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

TimeLine

- The TimeLine technique doesn't solve our problems
- It helps to expose the real status early and continuously
- Instead of accepting the undesired outcome,
  we *do something about it*
- The earlier we know, the more we can do about it
- We start saving time from the very beginning
- We can save a lot of time in any project,
  while producing a better outcome



If, and only if, we are serious about time !

www.malotaux.nl/booklets

More

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 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

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

www.malotaux.nl/inspections

Inspection pages

More

- Session: Inspection used in various ways (reserve)
- Session: How to create empowered teams (reserve)
- Session: How to move to Zero Defects (2015 video)
- Tuesday, Wednesday: AskTheExpert and Coaching sessions
- Thursday
- Friday
- Booklets – www.malotaux.nl/booklets
- Email – niels@malotaux.nl

Evolutionary Planning

Producing even more in less time

www.malotaux.nl/conferences

Niels Malotaux

N R Malotaux Consultancy

+31 655 753 604 niels@malotaux.nl www.malotaux.nl

Some extra

(we won't have time for)

Active Synchronization

Somewhere around you, there is the bad world.

If you are waiting for a result outside your control, there are three possible cases:

1. You are sure they'll deliver Quality On Time
2. You are not sure
3. You are sure they'll not deliver Quality On Time

- If you are not sure (case 2), better assume case 3
- From other Evo projects you should expect case 1
- Evo suppliers behave like case 1

In cases 2 and 3: Actively Synchronize: Go there !

1. Showing up increases your priority
2. You can resolve issues which otherwise would delay delivery
3. If they are really late, you'll know much earlier

Interrupts

- Boss comes in: “Can you paint the fence?”
- What do you do?

- In case of interrupt, use the interrupt procedure

Interrupt Procedure ”We shall work only on planned Tasks”

In case a new task suddenly appears in the middle of a Task Cycle (we call this an Interrupt) we follow this procedure:

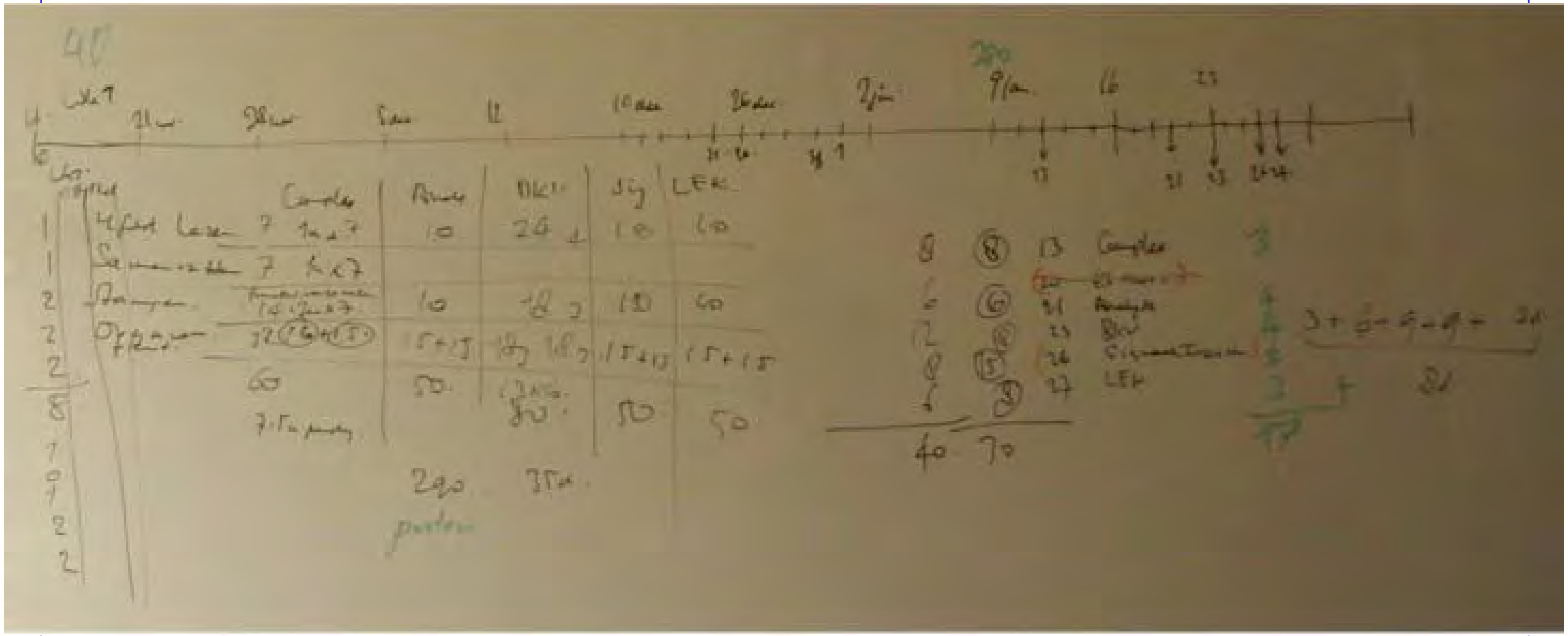
1. Define the expected Results of the new Task properly
2. Estimate the time needed to perform the new Task, to the level of detail really needed
3. Go to your task planning tool (many projects use the ETA tool)
4. Decide which of the planned Tasks is/are going to be sacrificed (up to the number of hours needed for the new Task)
5. Weigh the priorities of the new Task against the Task(s) to be sacrificed
6. Decide which is more important
7. If the new Task is more important: replan accordingly
8. I the new Task is not more important, then do not replan and *do not work* on the new Task. Of course the new Task may be added to the Candidate Task List
9. Now we are still working on planned Tasks.

TimeLine exercise example

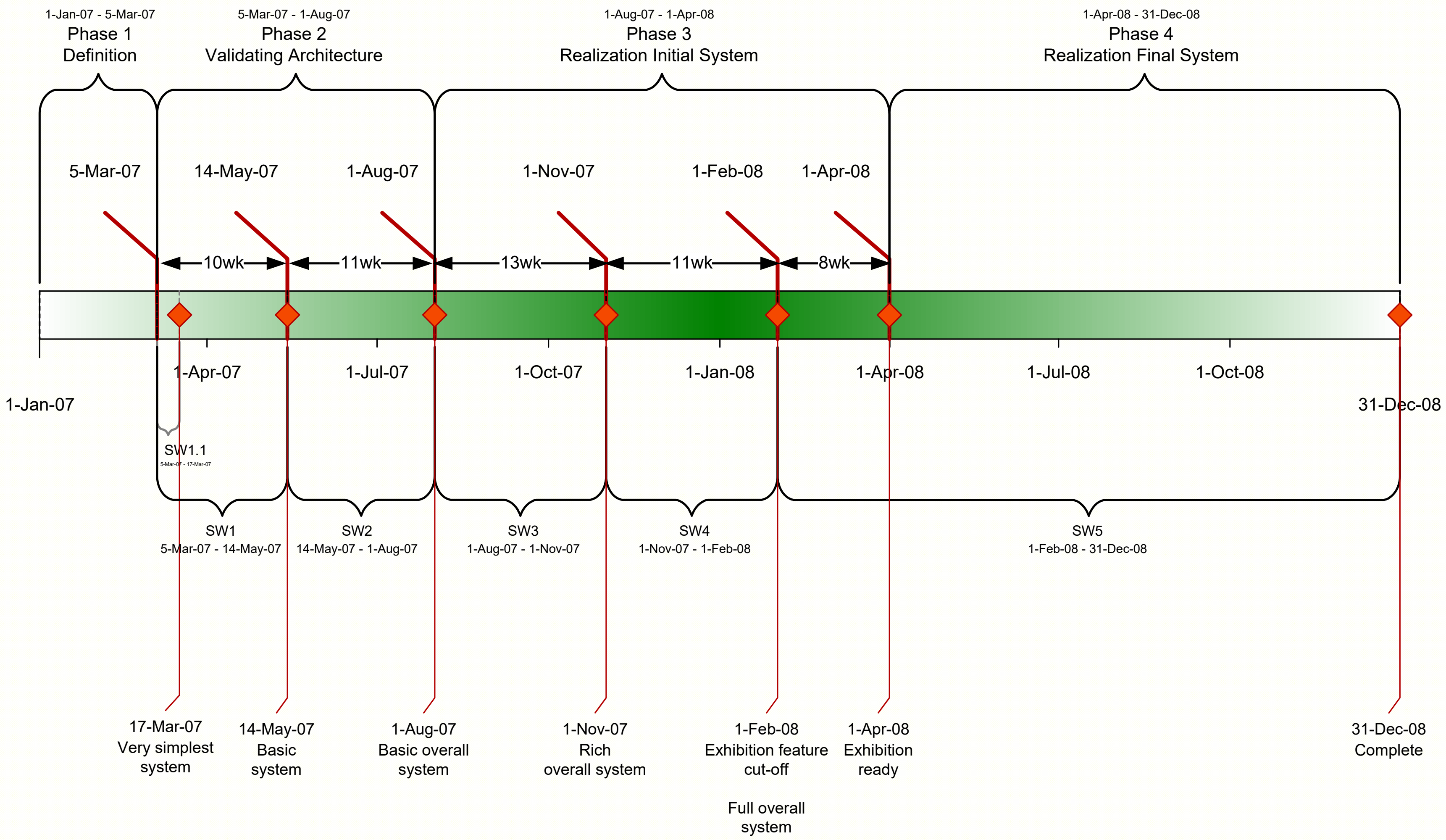
- Preparing for student exams

*

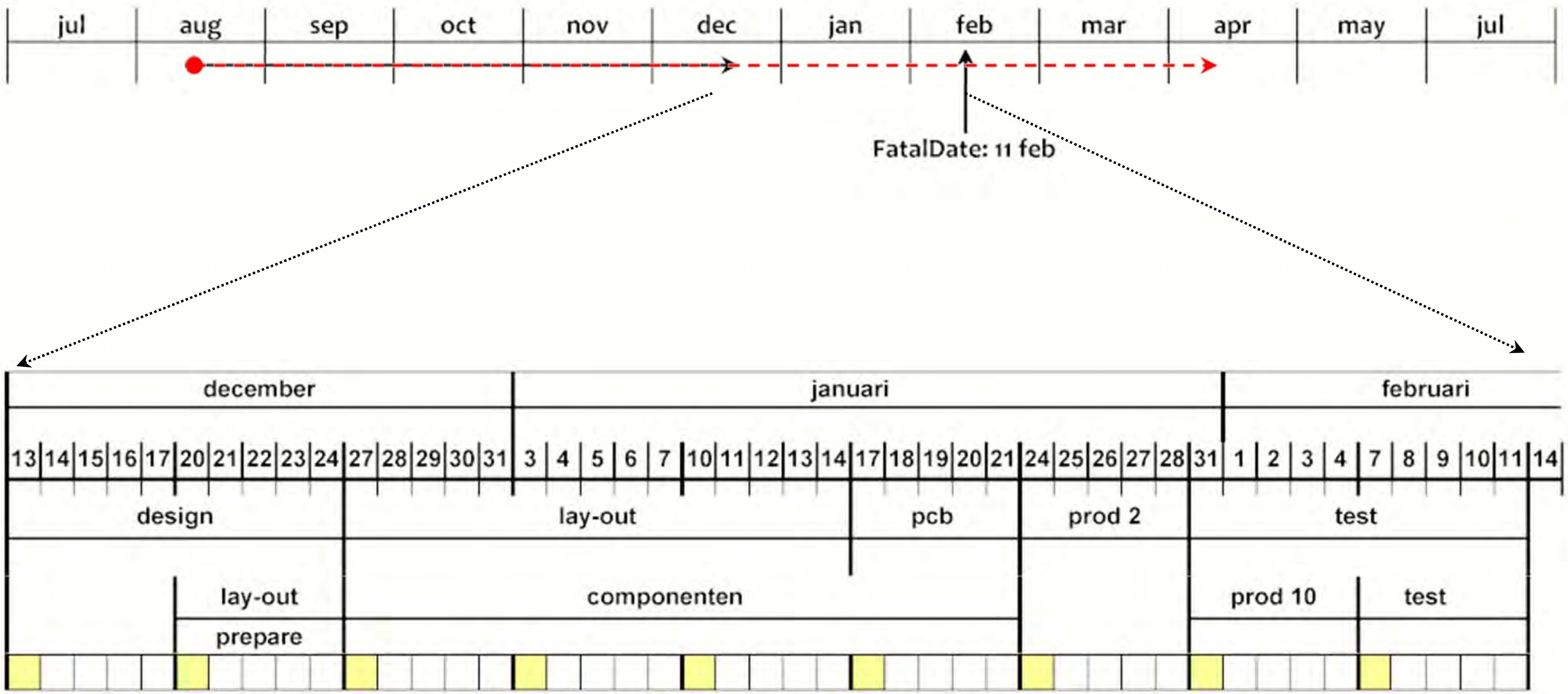
What we did



TimeLine example



TimeLine planning

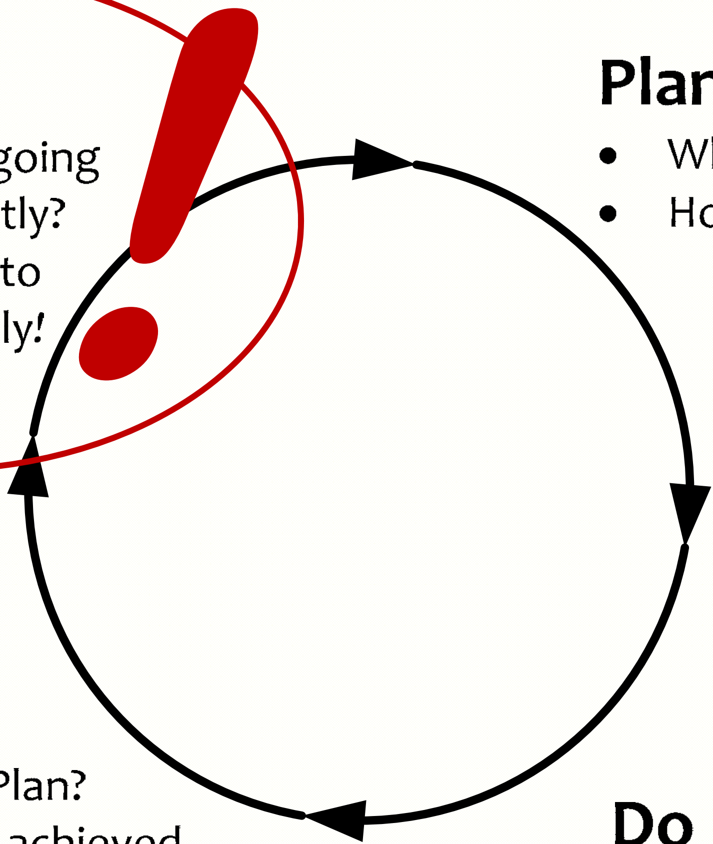


Help !
We have a QA problem !

- 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 we do something about this?

www.malotaux.nl/booklets - booklet#8

Do you think you can help us ?



Act

- What are we going to do differently?
- We are going to do it differently!

Plan

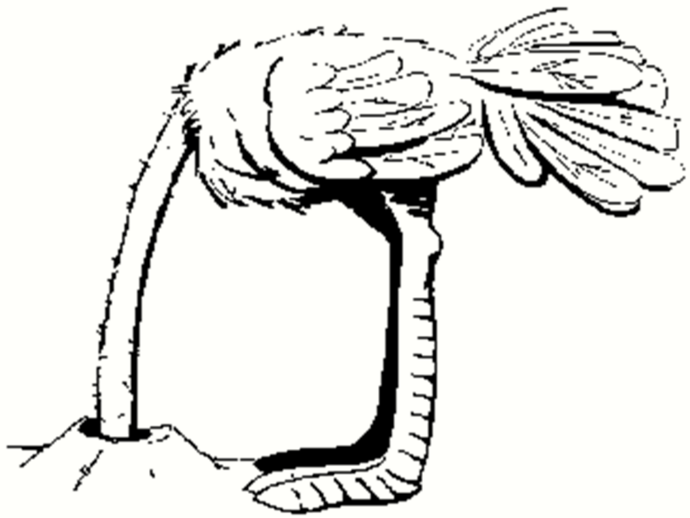
- What to achieve
- How to achieve it

Check

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

Do

Carry out the Plan



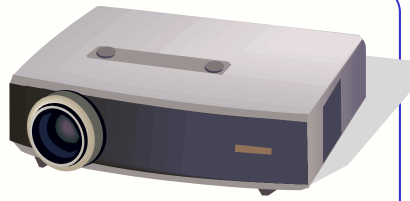
In stead of complaining about a problem …

(Stuck in the Check-phase)

Let's do something about it !

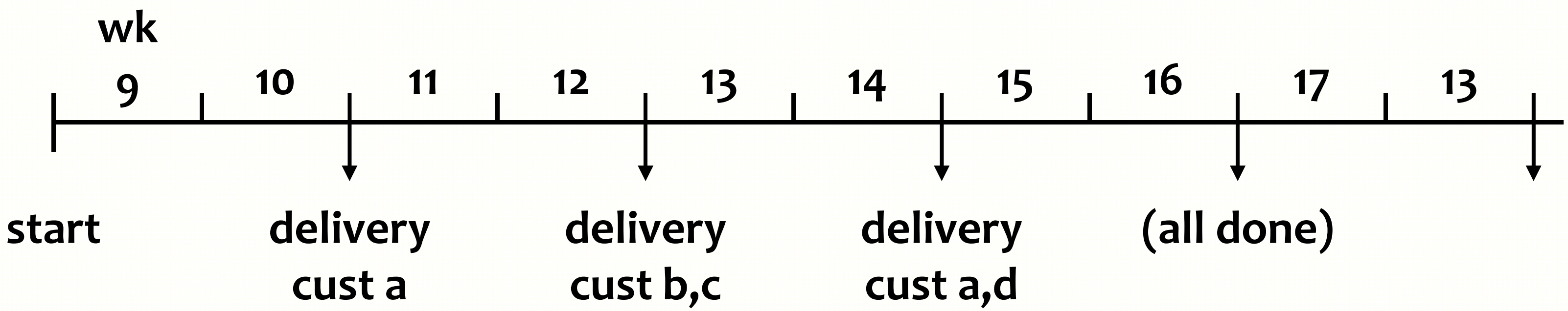
(Moving to the Act-phase)

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



Line	Activity	Estim	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

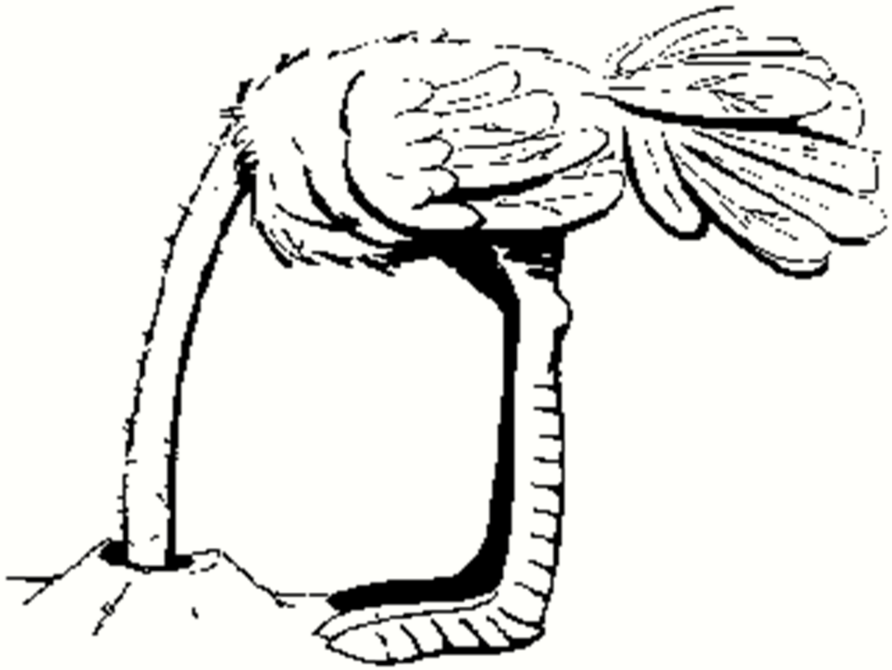
- "We'll have a solution at that date … Will you be ready for it ?"
  An other customer could be more eagerly waiting
- Most promising customers

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

Recently:

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

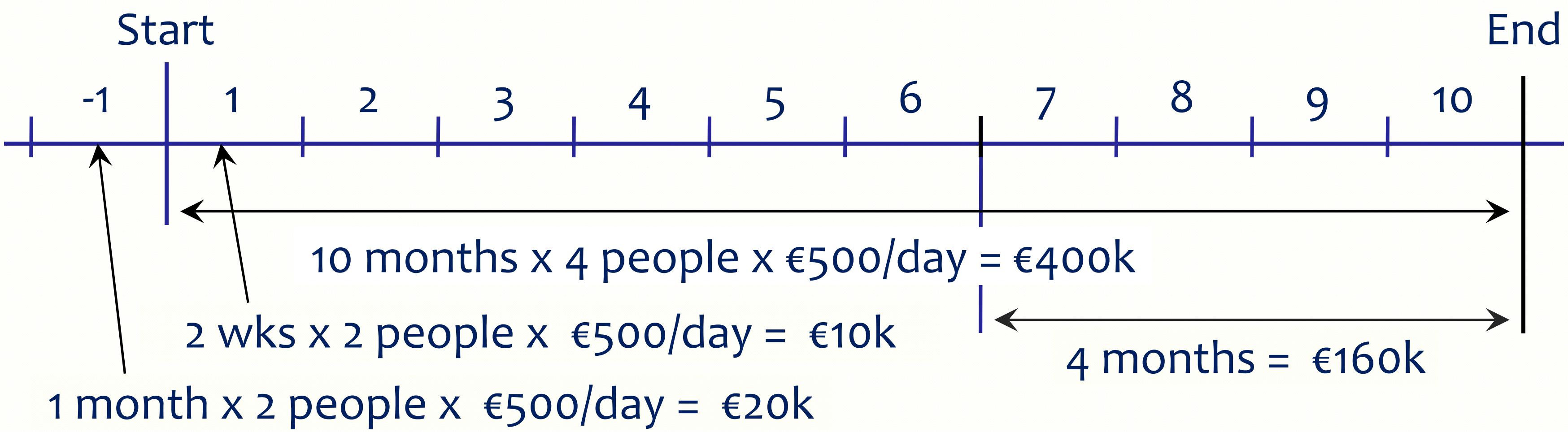


The problems in projects are not the real problem,
the real problem is that we don’t do something about it

Doing retrospectives does not solve the problem !

Prespectives save a lot of time

The Cost of Time



- We can save 4 months by investing €200k → "That's too much !"
- It's a *nicer* solution - Let's do 2 weeks more research on the benefits
- What are the expected revenues when all is done? → €16M/yr (1.3M/mnd)
- So 2 weeks extra doesn't cost €10k, but rather €16M/24 = €670k
- And saving 4 months brings €16M/3 = €5M extra

➔ Invest that €200k *NOW* and *don't waste time* **!**

Impact Estimation principle

How much % of what we want to achieve do we achieve by this solution

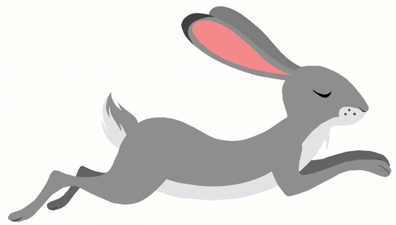
At what cost ?

Possible solutions to achieve it

Could we get all, within the budgets of time and cost ?

		Design Idea #1	Design Idea #2	Design Idea #3	Total Impact
What to achieve	Objectives	Impact on Objective	Impact on Objective	Impact on Objective	Sum of Impacts on Objectives
Cost to achieve it	Resources Time Money	Impact on Resources	Impact on Resources	Impact on Resources	Sum of Impact on Resources
Return on Investment	Benefits to Cost Ratio	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	

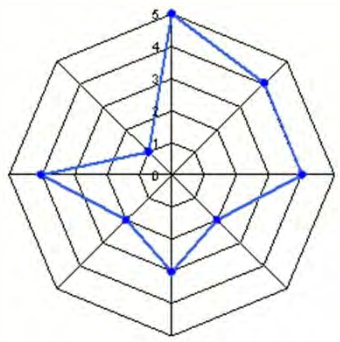
Agile or agile ?



What is Agile ?

- A philosophy (Agile Manifesto)

The Agile Manifesto (2001)



We are uncovering better ways of developing software by doing it and helping others do it

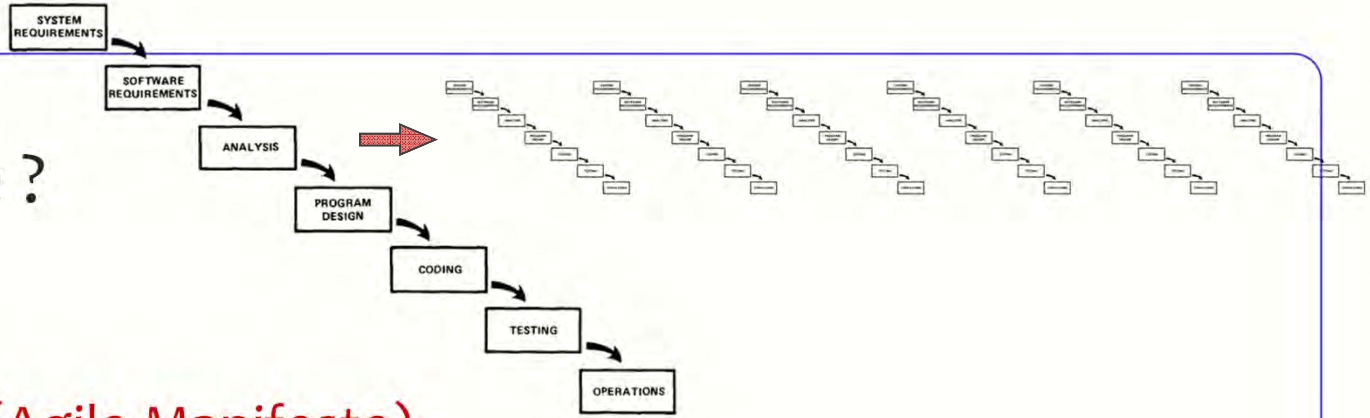
Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is *value in the items on the right*, we value the items on the left more

From the Principles behind the Agile Manifesto

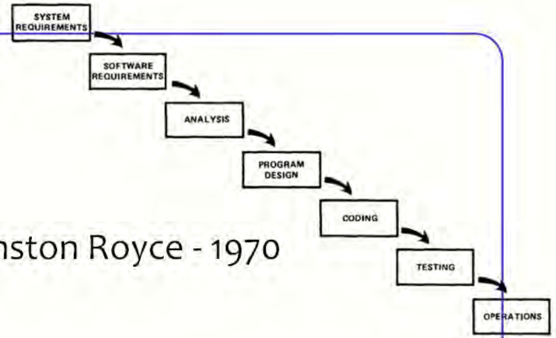
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
  Software is always part of a system
- We welcome changing requirements, even late in development
  If requirements have to change, let's *provoke* requirements change as quickly as possible
- We deliver working software frequently;
  Working software is the primary measure of progress
  What we deliver simply works.
  If the working software doesn't do what it should, is that a measure of progress?
- Business people and developers must work together daily
  Do they ? Should they ? Daily ?
- Simplicity - the art of maximizing the amount of work not done
  The art of not doing what is superfluous ! Why make it complex if we can keep it simple ?
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly
  Not just retrospectives, but even more importantly: prespectives



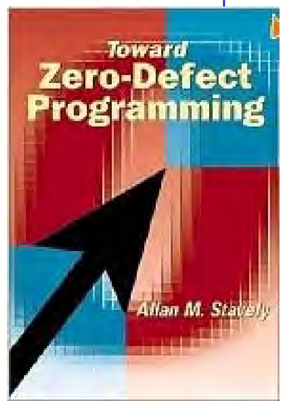
What is Agile ?

- A philosophy (Agile Manifesto)
- agile = ability to move quick, easy and adaptable
- Short iterations – not one Waterfall
- Delivering value (do we measure progress towards real value ?)
- Retrospectives (retrospectives on retrospectives: did it really work ?)
- Not a standard: You can make of it whatever you want
- XP - focus on software development techniques
- Scrum - very basic short term organization of development
- Are you agile if you religiously focus on a ‘method’ ?

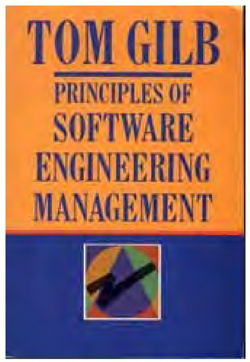
The past was already ahead

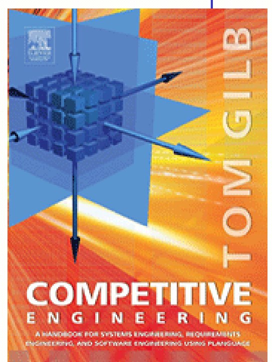


- Managing the development of large software systems - Winston Royce - 1970
  - Famous ‘Waterfall document’: figure 2 showed a ‘waterfall’
  - Text and other figures showed that Waterfall doesn’t work
  - Anyone promoting Waterfall doesn’t know or didn’t learn from history
- Cleanroom software engineering - 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





XP – eXtreme Programming

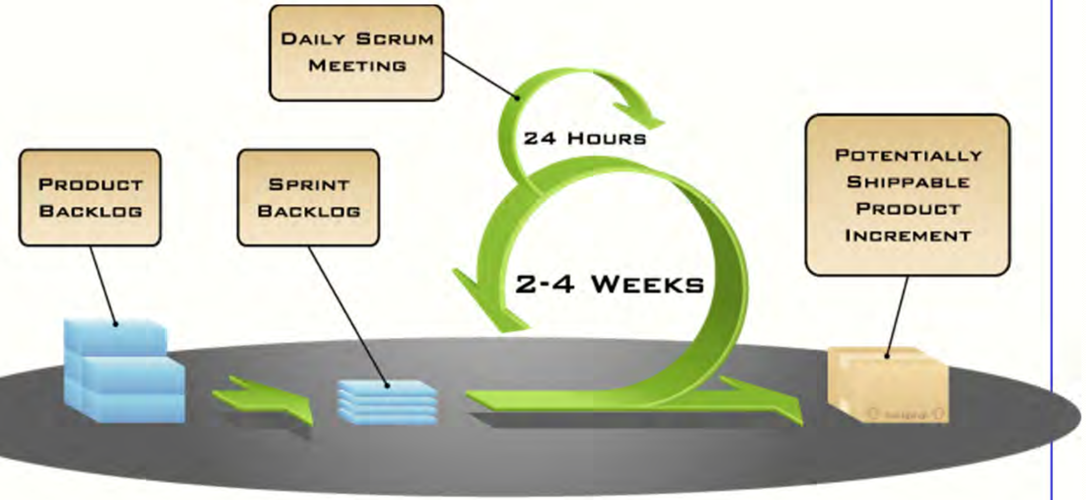
- Planning Game
- Metaphor
- Simple Design
- Testing (TDD)
- Refactoring
- Coding standards
- Small releases
- Pair programming
- Collective Ownership
- Continuous integration
- 40-hour week
- On-site customer

Original project was not successful
as soon as the writer of the book left the project

Scrum

80% of Scrum projects
are ScrumBut

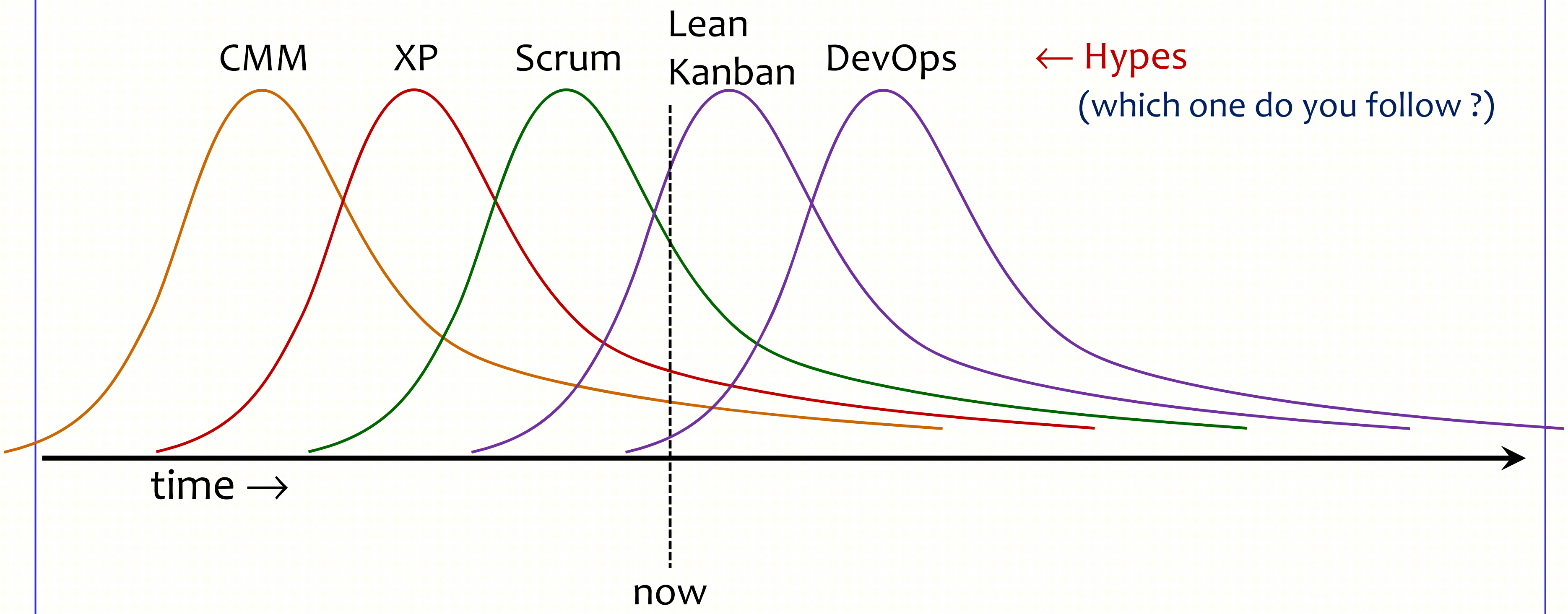
- Sprint
  - 1 – 4 weeks
  - Sprint Planning meeting
  - Sprint Review meeting
  - Sprint Retrospective
- Artefacts
  - Product backlog
  - Sprint backlog
  - Sprint burn down chart



- Roles
  - Scrum Master (facilitates, coaches on rules)
  - Team – multifunctional (design, develop, test, etc)
  - Product Owner – voice of customer
- Daily Scrum - Stand-up meeting
  a. What have you done since yesterday
  b. What are you planning today
  c. Impediments limiting achieving your goals ?

a lot of ritual

It’s not the method



If the previous method didn’t work, the next won’t work either

What's usually missing in Agile ?

Ref Tom Gilb

Stakeholder Focus

- Real projects have dozens of stakeholders
  - Not just a customer in the room, not just a user with a use case or story

Results Focus

- It is not about *programming*, it is about making *systems* work, for *real* people

Systems Focus

- It is not about coding, but rather:
  reuse, data, hardware, training, motivation, sub-contracting, outsourcing, help lines, user documentation, user interfaces, security, etc.
- So, a *systems engineering* scope is necessary to deliver results
- Systems Engineering needs *quantified performance and quality objectives*

Planning

Ref Niels Malotaux

- Retrospectives within the Sprint
- Retrospectives of retrospectives
- Planning what *not* to do → *preflection*
- Overall planning and prediction: when will what be done

If we add something …

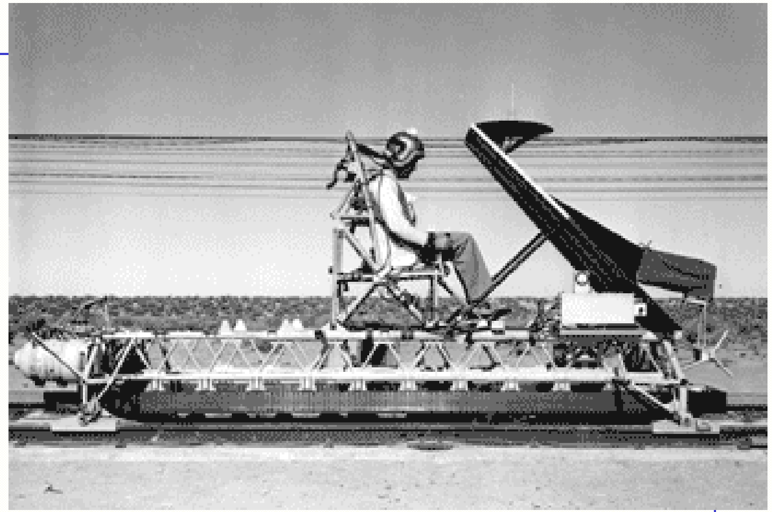
If we add something, something else will not be done

Rather than letting it happen randomly

We better decide what will happen

Murphy's Law

See www.malotaux.nl/murphy



- Whatever can go wrong, will go wrong
- Should we accept fate ??

Murphy's Law for Professionals:

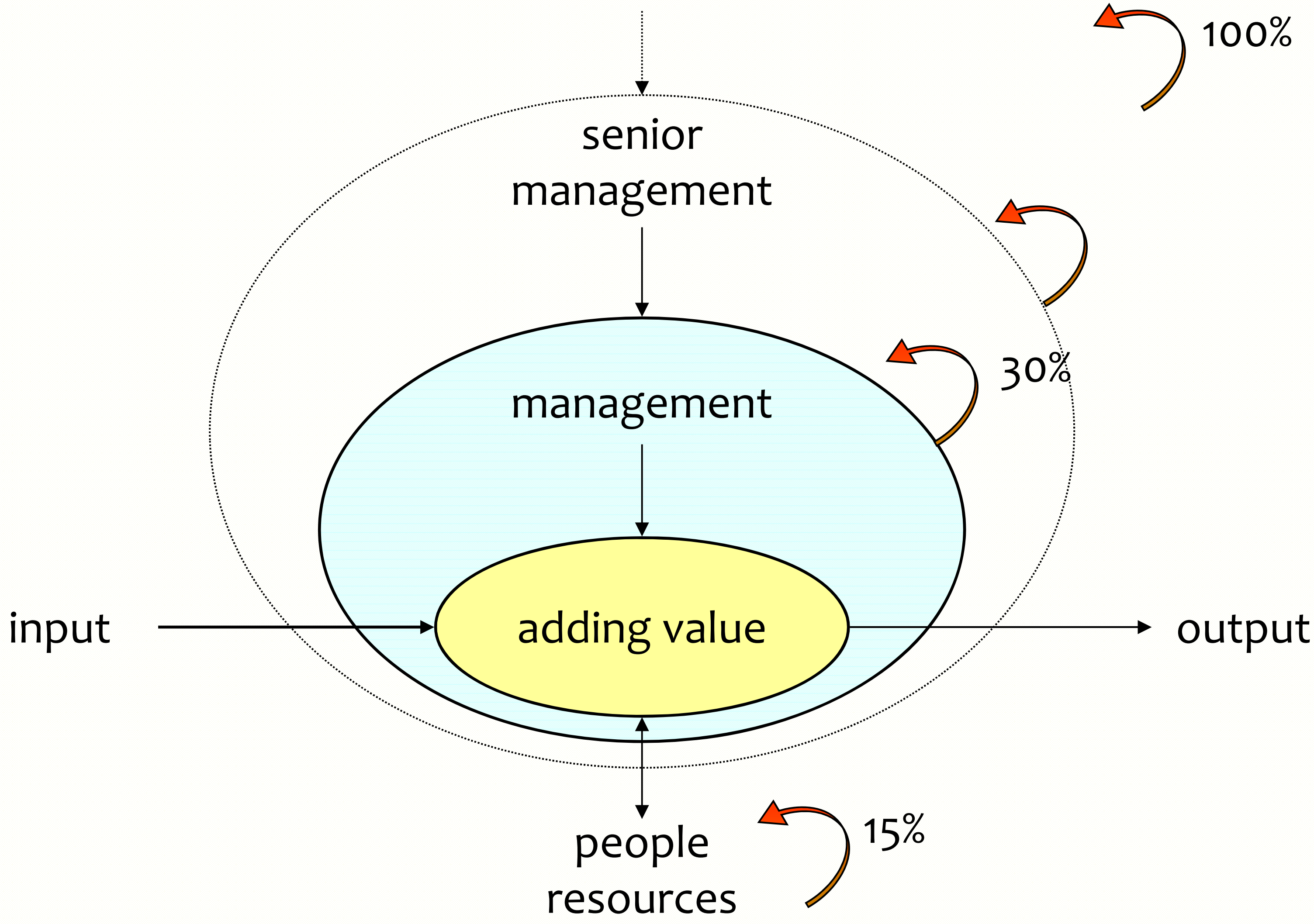
Whatever can go wrong, will go wrong …

Therefore:

We should actively check all possibilities that can go wrong and *make sure that they cannot happen*

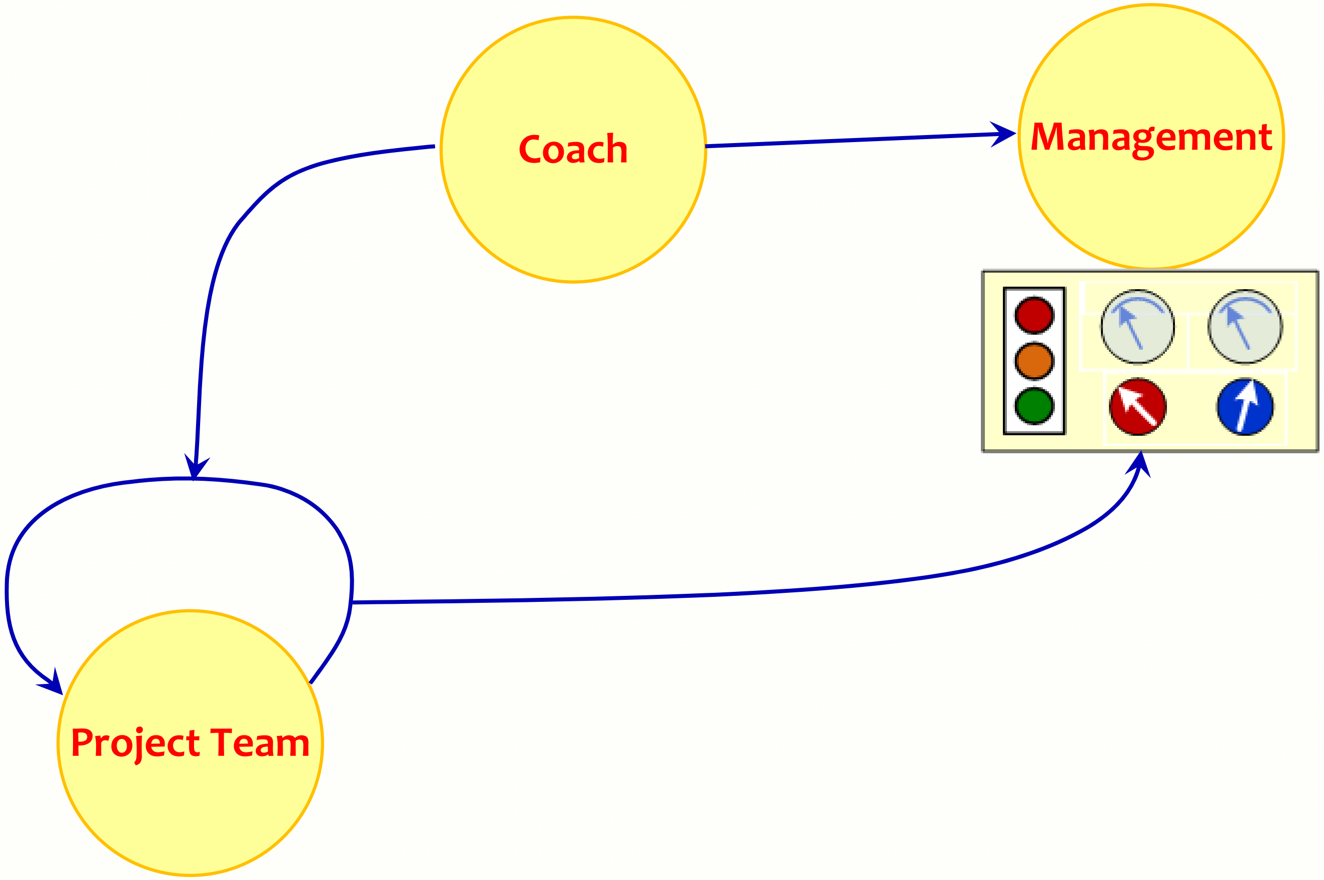
Management Issues

Simple model of Management



See www.malotaux.nl/managementmodel

Local Loop Principle



See www.malotaux.nl/localloop