London
6 November 2009

Niels Malotaux

# Evolutionary Project Planning

## Getting and Keeping the Project On Time

N R Malotaux - Consultancy
The Netherlands
+31-30-2288868
niels@malotaux.nl
www.malotaux.nl

SkillsMatter - November 2009 - London

**Niels Malotaux**

# Evolutionary Project Planning

## Getting and Keeping the Project On Time

## Niels Malotaux

Niels Malotaux is an independent Project Coach specializing in optimizing project performance. He has over 35 years experience in designing electronic hardware and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading his own systems design company. Since 1998 he devotes his expertise to helping projects to deliver Quality On Time: delivering what the customer needs, when he needs it, to enable customer success. To this effect, Niels developed an approach for effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, India, Ireland, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less in the practice of real projects.

Niels puts development teams on the Quality On Time track and coaches them to stay there and deliver their quality software or systems on time, without overtime, without the need for excuses. Practical methods are developed, used, taught and continually optimized for:

- Evolutionary Project Management (Evo)
- Requirements Engineering and Management
- Reviews and Inspections.

Within a few weeks of turning a development project into an Evo project, the team has control and can tell the customer when the required features will all be done, or which features will be done at a certain date. Niels enjoys greatly the moments of enlightenment experienced by his clients when they find out that they can do it, that they are really in control, for the first time in their lives.


N R Malotaux
Consultancy

Niels Malotaux
project coach

*Result Management*

Bongerdlaan 53
3723 VB Bilthoven
The Netherlands
tel +31-30-228 88 68
fax +31-30-228 88 69
mob +31-6-5575 3604
niels@malotaux.nl
www.malotaux.nl

# Evolutionary Project Planning

## Getting and Keeping the Project On Time

Niels Malotaux

N R Malotaux
Consultancy

1

Niels Malotaux

Result Management

- Project Coach
  - Evolutionary Project Management (Evo)
  - Requirements Engineering
  - Reviews and Inspections

- Researching problems in projects
- Finding ways for fundamentally overcoming these problems
- Ploughing back into projects
- Tuning of the results (because theory isn't practice)

2

## Who's working in Projects ?

- Do projects regularly deliver Quality on Time?
- How do you know?
- Why not?
- Is this normal?
- Can we do something about it?

- What is the Right Result?
- What is the Right Time?

3

## Not every project is successful

- Apparently we're doing something wrong
- Otherwise projects would succeed and be on time

- 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*
  - They didn't

One of the problems is to determine what the project really is about

4

## 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

5

## The Importance of Time

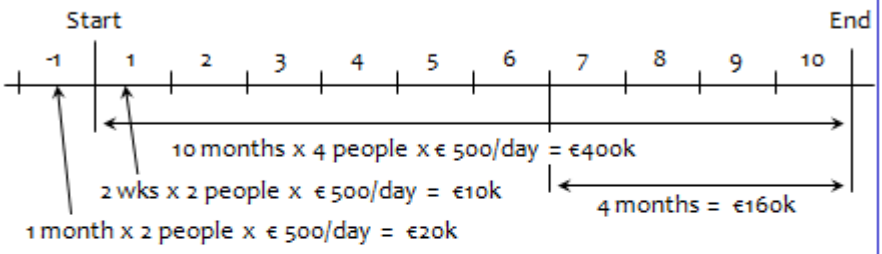
Business Case (why are we doing it)


Expected Return on Investment (ROI)

- – 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
- – Loss of doing nothing at all - diminishing benefit from legacy system
- + Benefit of doing - huge (otherwise other projects would be more rewarding)

6

## 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/mon)
- 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 !*

7

If our previous project was late,
our current project will also be late

unless we do things *differently* and *better*

If we don't learn from history,
we are doomed to repeat it

Projects don't have to be late
They deserve better

8

## Preflection, foresight, prevention

Insanity is doing the same things over and over again
and hoping the outcome to be different *(let alone better - nrm)*

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

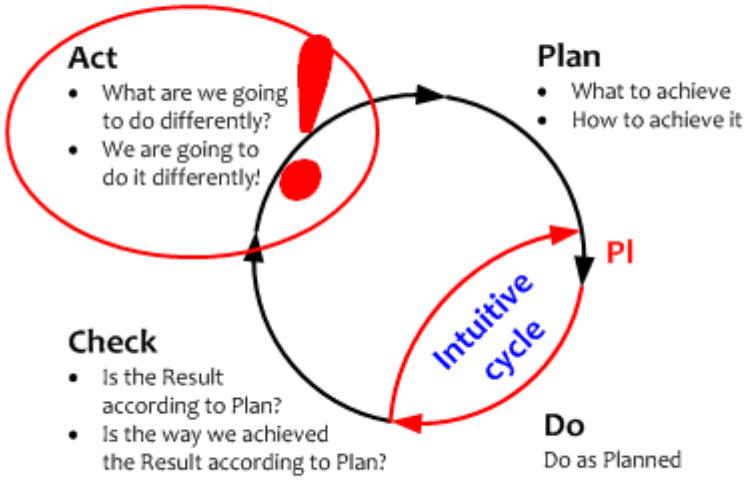
Only with *prevention* we can save precious time

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

9

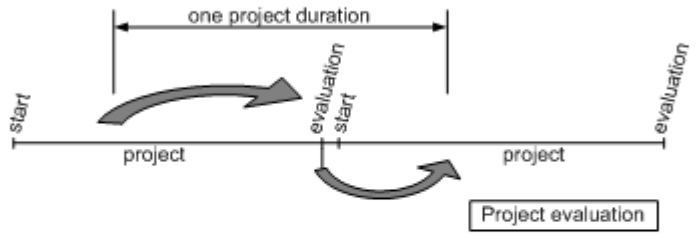
## The essential ingredient: the PDCA Cycle

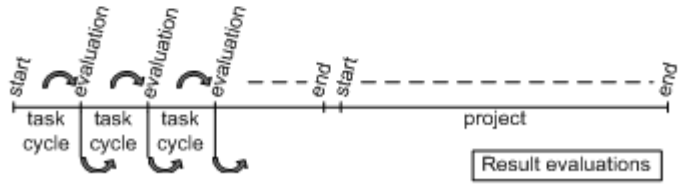
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)




10

## Project evaluations




11

## Knowledge

how to achieve the goal


If we

- Use very short Plan-Do-Check-Act cycles
- Constantly selecting the most important things to do

doing the right things

then we can

- Most quickly learn what the real requirements are
- Learn how to most effectively and efficiently realize these requirements

and we can

- Spot problems quicker, allowing more time to do something about them

doing the right things right

12

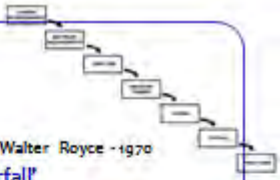
## Known for decades

- Benjamin Franklin (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- Henry Ford (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - Learning from waste, keeping things clean and safe, better treated people produce more
- Toyoda's (Sakichi, Kiichiro, Eiji) (1867-1930, 1894-1952, 1913-)
  - Jidoka: Zero-Defects, stop the production line (1926)
  - Just-in-time – flow – pull
- W. Edwards Deming (1900-1993)
  - Shewart cycle: Design-Produce-Sell-Study-Redesign (Japan – 1950)
  - Becoming totally focused on quality improvement (Japan – 1950) Management to take personal responsibility for quality of the product
  - Out of the Crisis (1986) - Reduce waste
- Joseph M. Juran (1904-2008)
  - Quality Control Handbook (1951, Japan – 1954)
  - Total Quality Management – TQM
  - Pareto Principe
- Philip Crosby (1926-2001)
  - Quality is Free (1980)
    - Zero-defects (1961)
- Taiichi Ohno (1912-1990)
  - (Implemented the) Toyota Production System (Beyond Large-Scale Production) (1988)
  - Absolute elimination of waste - Optimizing the TimeLine from order to cash
- Masaaki Imai (1930-)
  - Kaizen: The Key to Japan's Competitive Success (1986)
  - Gemba Kaizen: A Commonsense, Low-Cost Approach to Management (1997)

**Basis: Eliminating Waste**
**Not doing what doesn't yield value**

13

## There is nothing new



- Managing the development of large software systems - Walter 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
- Incremental development - Harlan Mills - 1971
  - Continual Quality feedback by Statistical Process Control
  - Continual feedback by customer use
  - Accommodation of change - Always a working system
- Cleanroom software engineering - Harlan Mills - 1970's
  - Incremental Development - Short Iterations
  - Defect *prevention* rather than defect removal
  - Statistical testing
  - 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


14

## Evo


- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for *Product*, *Project* and *Process*, based on ROI and highest value
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- We know we are not perfect, but the customer shouldn't be affected
- Evo is about delivering Real Stuff to Real Stakeholders doing Real Things *"Nothing beats the Real Thing"*
- Projects seriously applying Evo, routinely conclude successfully on time, or earlier

15

## Evolutionary Project Management (Evo)

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

Right product

Zero Defects Attitude

## 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

Right time

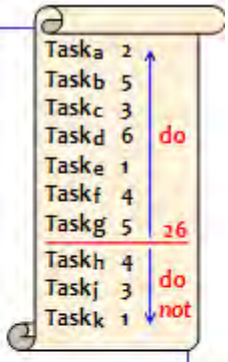
16

## 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

17

## Every week we plan



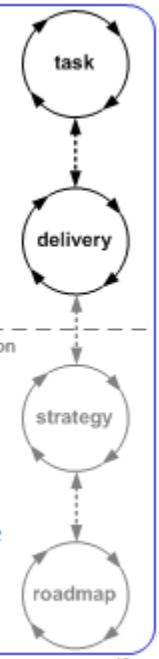

- How much time do we have available
- 2/3 of available time is net plannable time
- What is most important to do
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr per week)
- What can, and are we going to do
- What are we not going to do

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

18

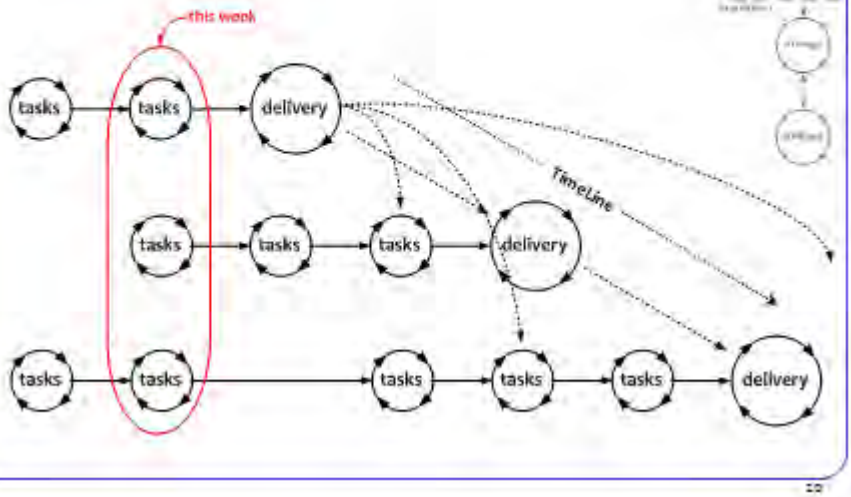
## DeliveryCycle

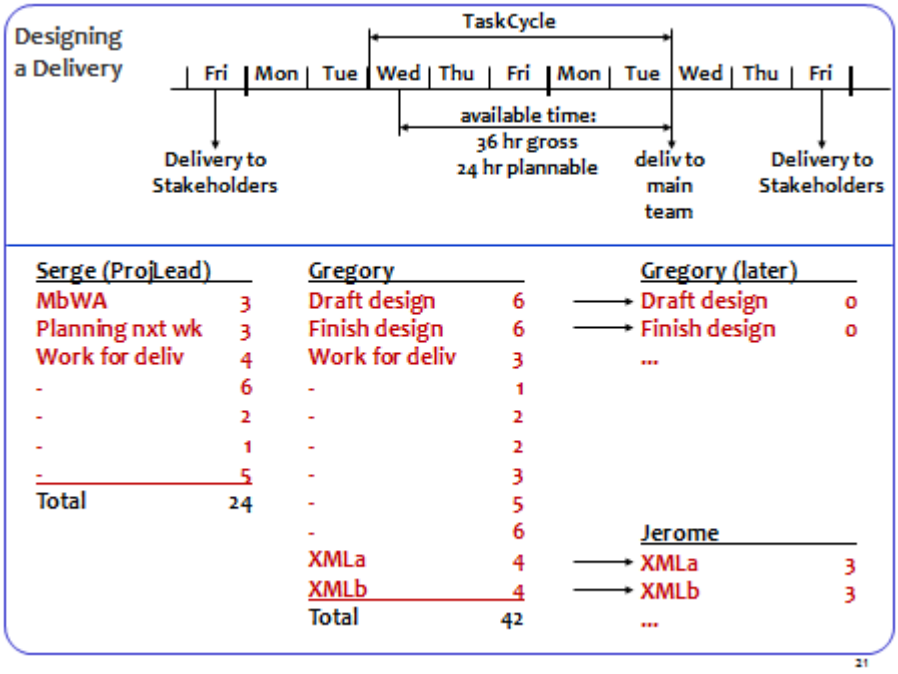
- Are we *delivering*
  the right things, in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
    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




19

## Tasks feed Deliveries





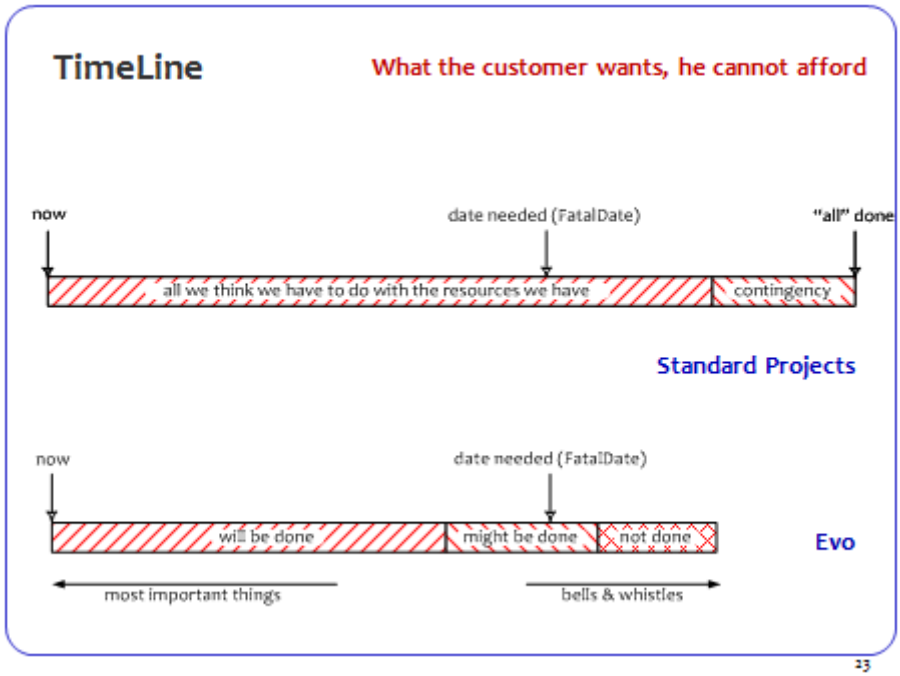

## Agile, but will we be on time?

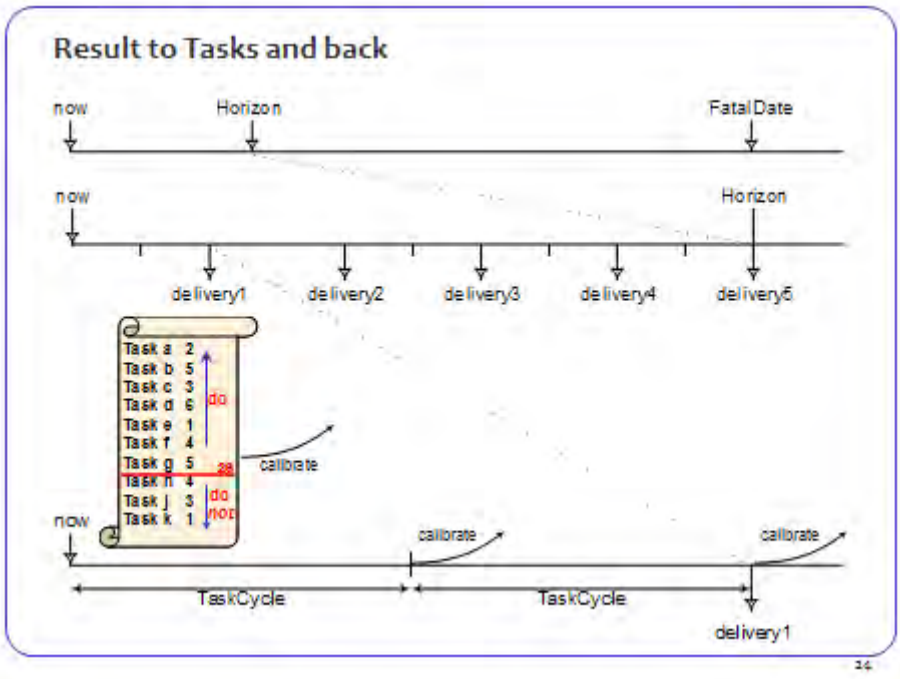
- Organizing the work in very short cycles
- Making sure we are doing the right things
- Doing the right things right
- Continuously optimizing (also what not to do)
- So, we already work more efficiently

but ...

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

22


TimeLine
What the customer wants, he cannot afford
now
date needed (FatalDate)
"all" done
all we think we have to do with the resources we have
contingency
Standard Projects
now
date needed (FatalDate)
will be done
might be done
not done
Evo
most important things
bells & whistles
23


Result to Tasks and back
now
Horizon
FatalDate
now
Horizon
delivery1
delivery2
delivery3
delivery4
delivery5
Task a 2
Task b 5
Task c 3
Task d 6
Task e 1
Task f 4
Task g 5
Task h 4
Task j 3
Task k 1
do
do not
calibrate
now
calibrate
calibrate
TaskCycle
TaskCycle
delivery1
24

## Calibration

| Activity | Estimate | Real |
|---|---|---|
| Act1 | Ae1 | Ar1 |
| Act2 | Ae2 | Ar2 |
| Act3 | Ae3 | Ar3 |
| Act4 | Ae4 | Ar4 |
| Act5 | Ae5 | Ar5 |
| Act6 | Ae6 | Ar6 |
| Act7 | Ae7 | Ar7 |
| Act8 | Ae8 | Ar8 |
| Act9 | Ae9 | Ar9 |
| Act10 | Ae10 | Ar10 |
| Act11 | Ae11 | |
| Act12 | Ae12 | |
| Act13 | Ae13 | |
| Act14 | Ae14 | |
| Act15 | Ae15 | |
| Act16 | Ae16 | |
| Act17 | Ae17 | |
| Act18 | Ae18 | |
| Act19 | Ae19 | |
| Act20 | Ae20 | |
| Act21 | Ae21 | |
| Act... | Ae... | |

Ratio ΣAr/ ΣAe till now

now

Predicted future Value Still To Earn

then

then2

**Calibration Factor**

$$\frac{\sum_{now-1}^{now-n} Ar}{\sum_{now-1}^{now-n} Ae}$$

**Value Still To Earn**

$$Calibration\ Factor * \sum_{now}^{then} Ae$$

15

## Predicting *what* will be done *when*

| Line | Activity | Estim | Spent | Still to spend | Ratio real/es | 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 | 3.5 | 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 |
| | | | | | | | | |

16

## Product/Portfolio/Resource Management

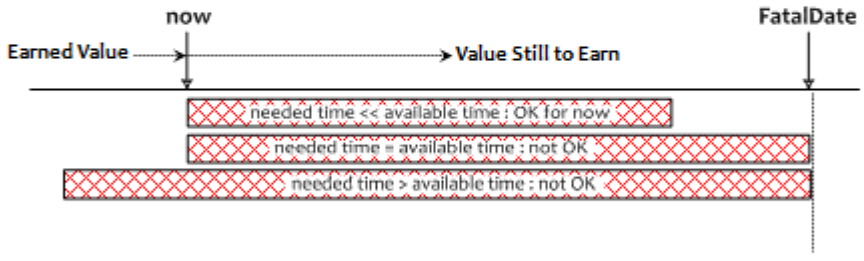
- Current Program/Portfolio/Resource Management is based on hope
- More a game than management

- With TimeLine we can provide PPR Management with sufficiently reliable data
- To start managing

27

## Most projects are late

- Apparently not appropriately managed
- Developers decisions have large impact on the time




- What do we do if we can see we won't make it on time ?
- When the match is over, scoring a goal is useless

28

## Deceptive options

- **Hoping for the best** (fatalistic)
- **Going for it** (macho)
- **Working Overtime** (fooling ourself)
- **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

29

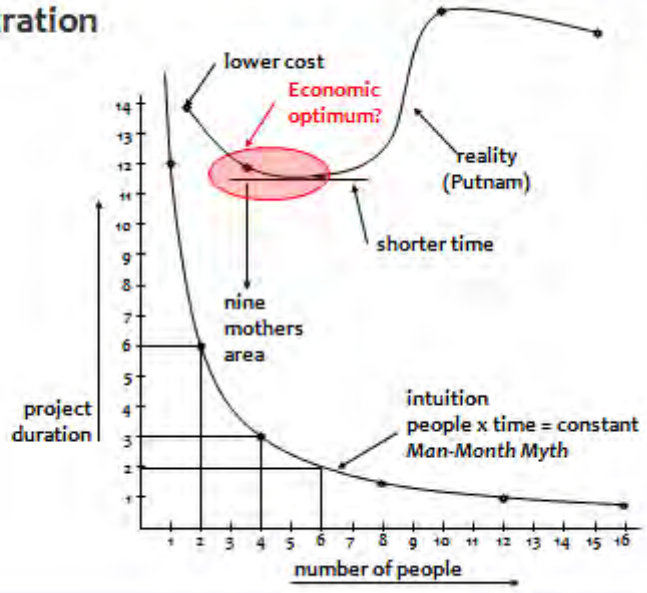
## Adding people to a late project ...

**makes it later**

(Brooks' Law, 1975)

30

## Project-duration




31

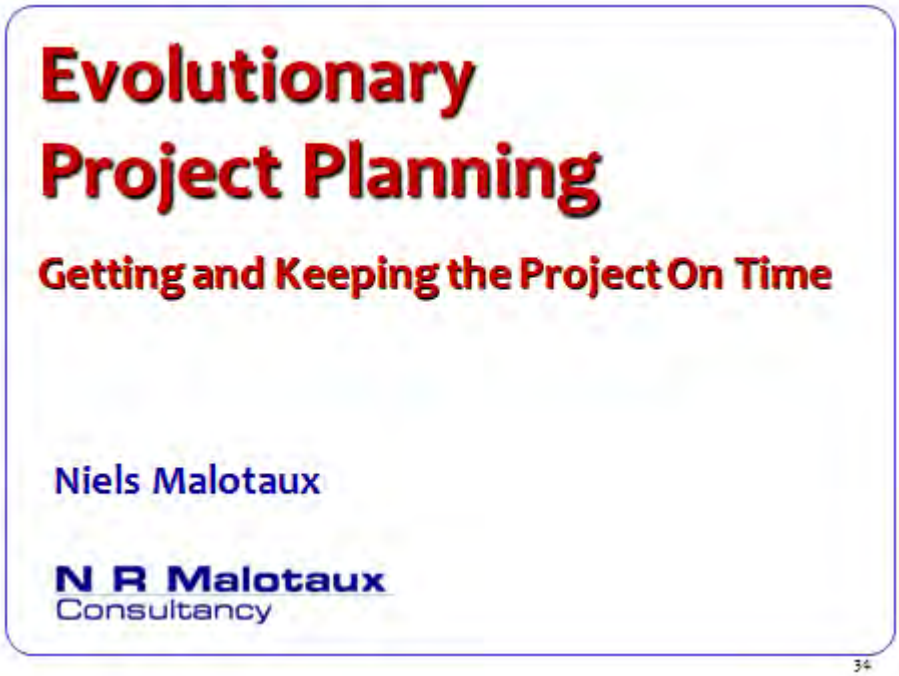
## Saving time

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, in stead of the solution we always used
  - The project
    - Doing the same in less time in stead 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 - doing things in the right order, at the right time
- TimeBoxing - much more efficient than FeatureBoxing

32


Evolutionary Project Management (Evo)
Plan-Do-Check-Act
The powerful ingredient for success
Business Case
Why we are going to improve what
Requirements Engineering
What we are going to improve and what not
How much we will improve: quantification
Architecture and Design
Selecting the optimum compromise for the conflicting requirements
Early Review & Inspection
Measuring quality while doing, learning to prevent doing the wrong things
Zero Defects Attitude
Right product
Evo Project Planning
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
Right time
33


Evolutionary
Project Planning
Getting and Keeping the Project On Time
Niels Malotaux
N R Malotaux
Consultancy
34

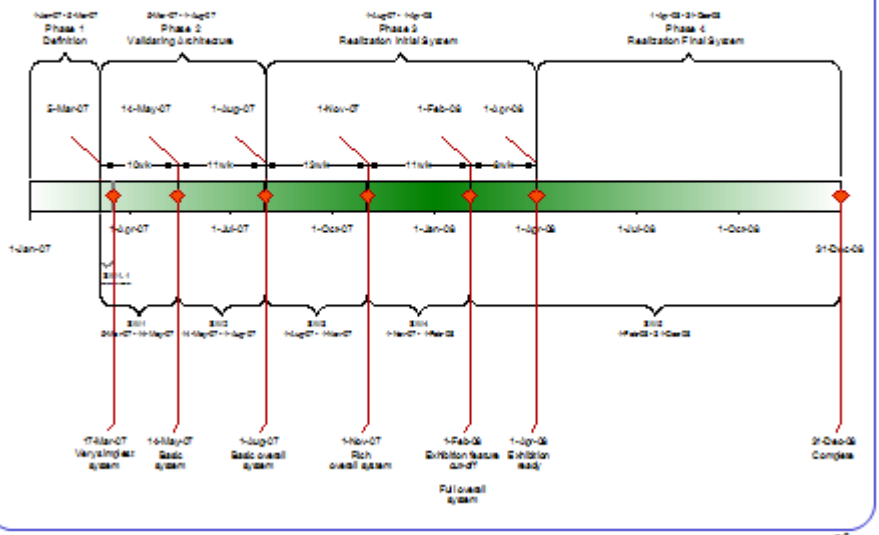
# Some examples
## from practice

Niels Malotaux

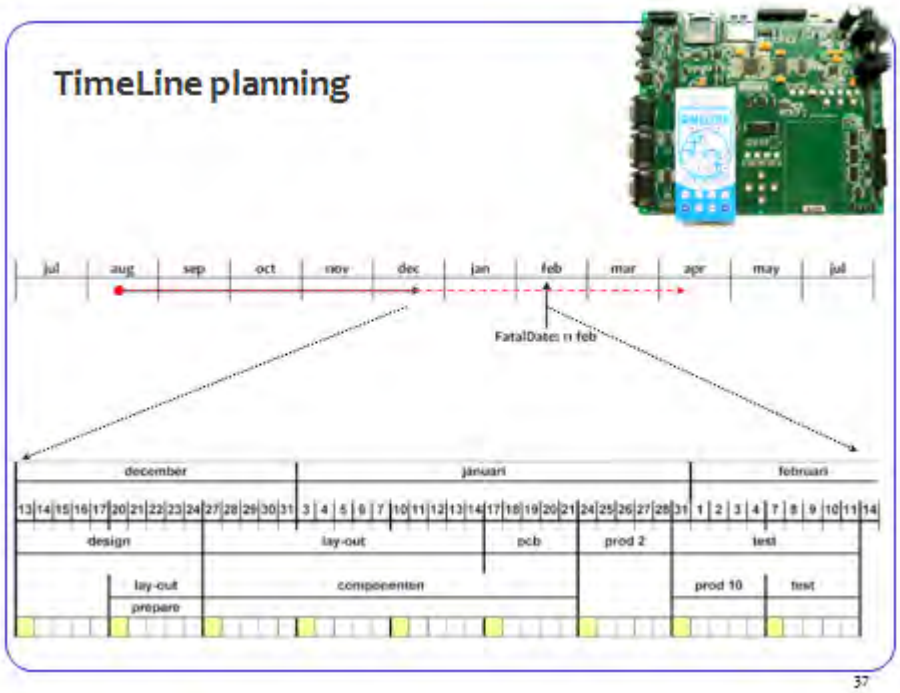
N R Malotaux
Consultancy

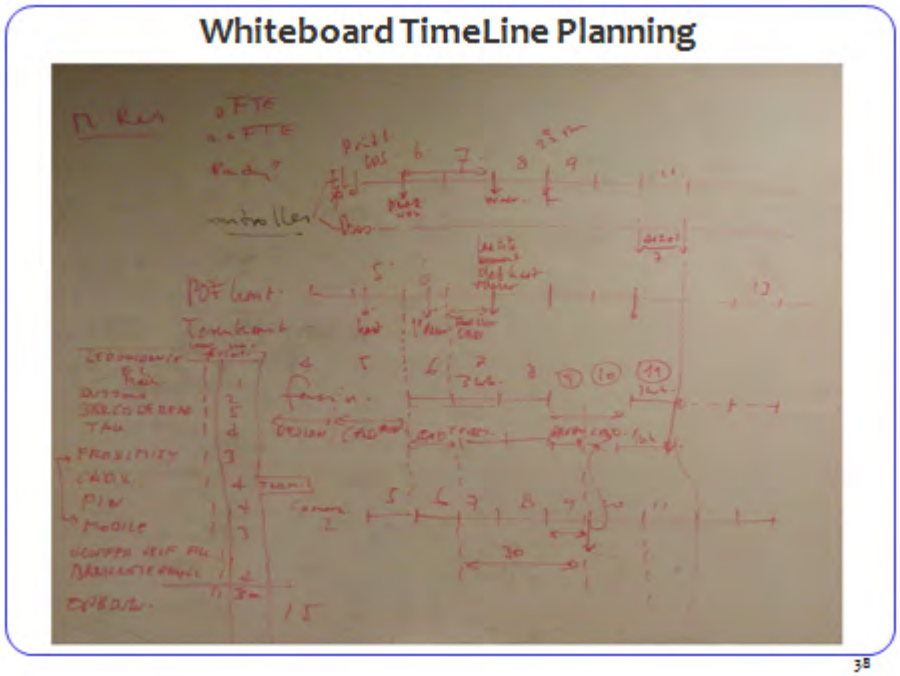
35

## TimeLine example

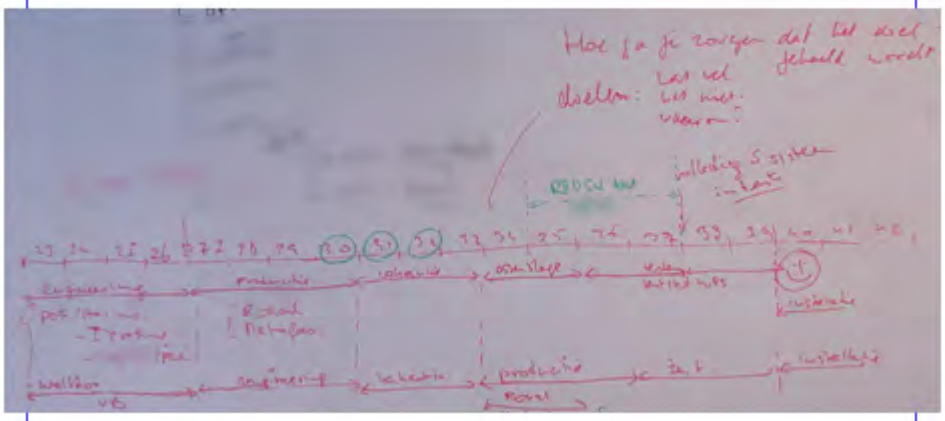



36

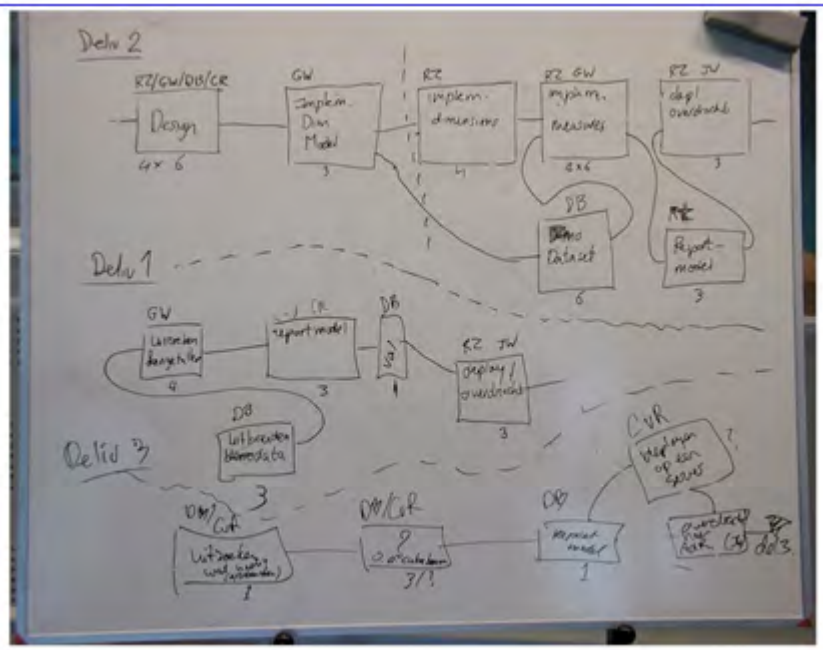

TimeLine planning
jul
aug
sep
oct
nov
dec
jan
feb
mar
apr
may
jul
FatalDate: 11 feb
december
januari
februari
13 14 15 16 17 20 21 22 23 24 27 28 29 30 31 3 4 5 6 7 10 11 12 13 14 17 18 19 20 21 24 25 26 27 28 31 1 2 3 4 7 8 9 10 11 14
design
lay-out
pcb
prod 2
test
lay-out
prepare
componenten
prod 10
test
37


Whiteboard TimeLine Planning
38

## Whiteboard TimeLine Planning



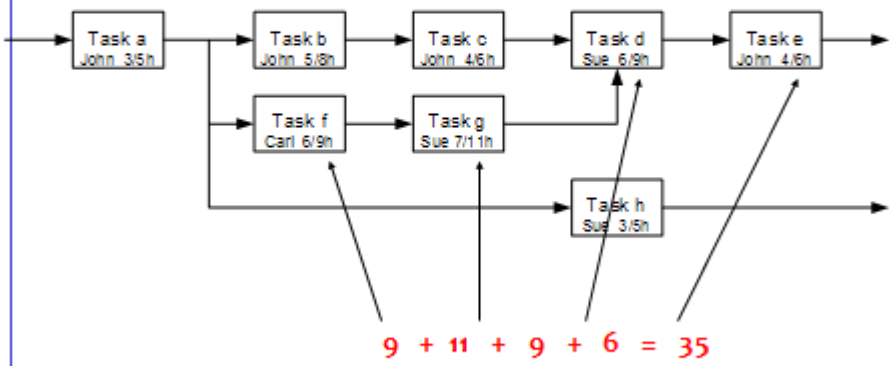
39




40

## PERT (Project Evaluation Review Technique)

used for *Designing* a Delivery




9 + 11 + 9 + 6 = 35

41

## 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?

42

## In stead of complaining about a problem ...

(Stuck in the Check-phase)

## Let's do something about it !

(Moving to the Act-phase)

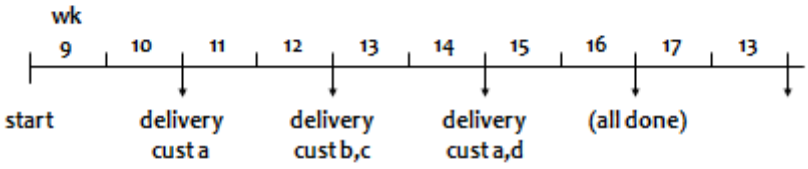
43

## 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 | | |

44

## 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

45

## 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:

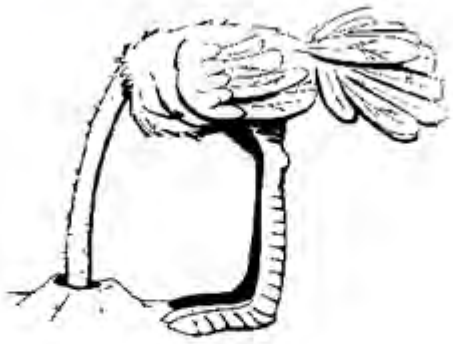
- Tester promoted to product manager
- Still coaching successors how to plan

46

## Links

- www.gilb.com
  Tom Gilb's website: Evo guru
- www.malotaux.nl
  Niels' activities: Evo evangelist
- www.malotaux.nl/nrm/Evo
  Evo pages
- www.malotaux.nl/nrm/Insp
  Inspection pages
- www.malotaux.nl/Booklets
  1. Evolutionary Project Management Methods (2001)
  2. How Quality is Assured by Evolutionary Methods (2004)
  3. Optimizing the Contribution of Testing to Project Success (2005)
  3a. Optimizing Quality Assurance for Better Results (2005)
  4. Controlling Project Risk by Design (2006)
  5. TimeLine: How to Get and Keep Control over Longer Periods of Time (2007)
  6. Human Behavior in Projects (2008)
  7. How to Achieve the Most Important Requirement (2008)
  8. Help! We have a QA Problem!
- www.malotaux.nl/nrm/Evo/ETAF.htm
  Download the Evo Task Administrator (ETA) tool (expects MSAccess 2000~2003)

47



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

48

# Defining the Result

Niels Malotaux

N R Malotaux
Consultancy

49

## Stakeholders are 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!*
- Some Stakeholders are victims of the project
  - They have no reason for the project to succeed, on the contrary
- Project risks,
  happening in almost every project
- No excuse to fail!

50

## 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

51

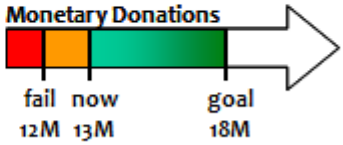
## Requirements Case

- Organisation collecting online giving for charities
- CEO: "Improve website to increase online giving for our 'customers' (charities)"
- Increasing market share for online giving
- Budget: 1M€ - 10 months
- Show results fast

Ref Ryan Shriver
ACCU Overload Feb 2008

52

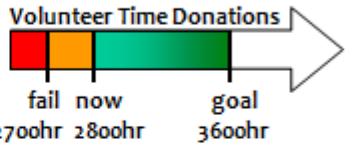
## Objective: Monetary Donations




| | |
|---|---|
| Name | Monetary Donations |
| Scale | Euro's donated to non-profits through our website |
| Meter | Monthly Donations Report |
| Fail | 12M |
| Now | 13M [2008] ← Annual Report 2008 |
| Goal | 18M [2009] |

Ref Ryan Shriver
ACCU Overload Feb 2008
53

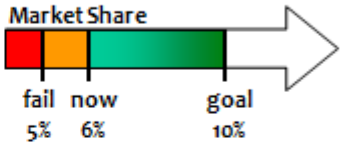
## Objective: Volunteer Time (Natura) Donations




| | |
|---|---|
| Name | Volunteer Time Donations |
| Scale | Hours donated to non-profits through our website |
| Meter | Monthly Donations Report |
| Fail | 2700 hr |
| Now | 2800 hr [2008] ← Annual Report 2008 |
| Goal | 3600 hr [2009] |

Ref Ryan Shriver
ACCU Overload Feb 2008
54

## Goal: Market Share




| | |
|---|---|
| Name | Market Share |
| Scale | Market Share %% online giving |
| Meter | Quarterly Industry Report |
| Fail | 5% |
| Now | 6% [Q1-2009] ← Quarterly Industry Report |
| Goal | 10% [Q1-2010] |

Ref Ryan Shriver
ACCU Overload Feb 2008

55

## Impact Estimation principle

| | Design Idea #1 | Design Idea #2 | Design Idea #3 | Total Impact |
|---|---|---|---|---|
| Objectives | Impact on Objective | Impact on Objective | Impact on Objective | Sum of Impacts on Objectives |
| Resources Time Money | Impact on Resources | Impact on Resources | Impact on Resources | Sum of Impact on Resources |
| Benefits to Cost Ratio | $\frac{\text{Benefits}}{\text{Cost}}$ | $\frac{\text{Benefits}}{\text{Cost}}$ | $\frac{\text{Benefits}}{\text{Cost}}$ | |

56

## Impact Estimation

| Impact Estimation | Monthly Donations | Facebook integration | Image & video uploads | Total effect for requirement |
|---|---|---|---|---|
| €€ donations 13M€ → 18M€ | 80% ±30% | 30% ±30% | 50% ±20% | 160% ±80% |
| Time donations 2800hr→3600hr | 10% ±10% | 50% ±20% | 80% ±20% | 140% ±50% |
| Market share 6% → 10% | 30% ±20% | 30% ±20% | 20% ±10% | 80% ±50% |
| Total effect per solution | 120% ±60% | 110% ±70% | 150% ±50% | 380% ±180% |
| Cost - money % of 1M€ | 30% ±10% | 20% ±10% | 50% ±20% | 100% ±40% |
| Cost - time % of 10 months | 40% ±20% | 20% ±10% | 50% ±20% | 110% ±50% |
| Total effect / money budget | 120/30 = 4 1.5 … 9 | 110/20 = 5.5 1.3 … 18 | 150/50 = 3 1.4 … 6.7 | |
| Total effect / time budget | 120/40 = 3 1 … 9 | 120/20 = 6 1.3 … 18 | 120/50 = 2.4 1.4 … 6.7 | |

Het Ryan Shinver - ACCU Overload Feb 2009

57

# Evolutionary Project Planning

## Getting and Keeping the Project On Time

Niels Malotaux

N R Malotaux Consultancy

58

# Supplementary Slides

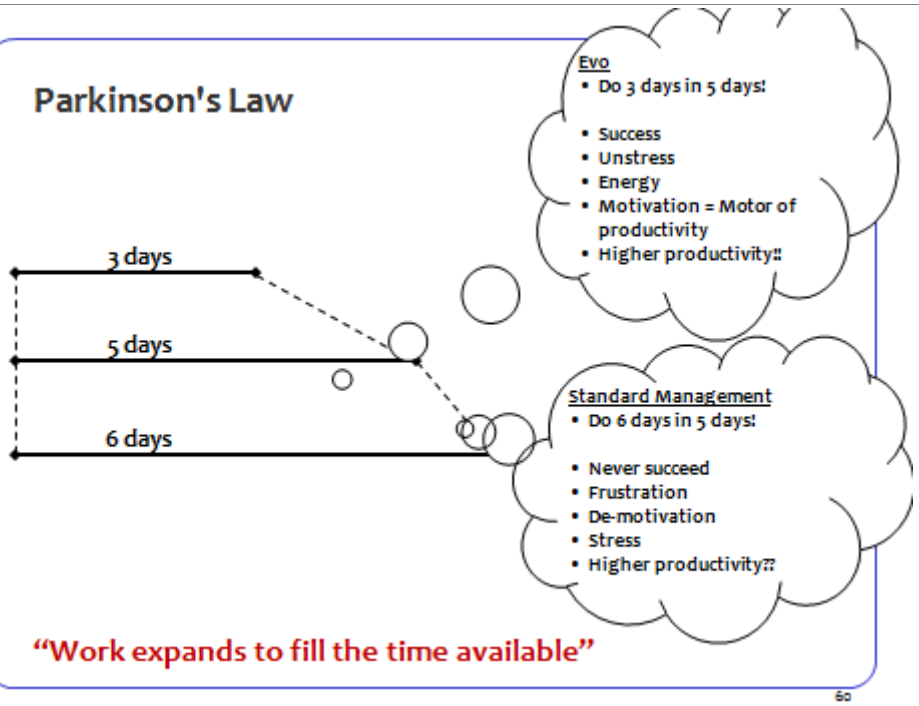
(no special order)

Niels Malotaux

N R Malotaux Consultancy

59

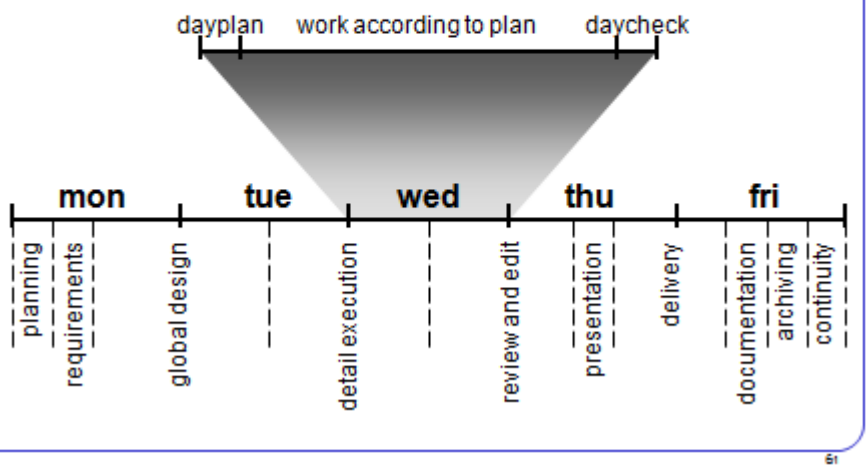
## Parkinson's Law




"Work expands to fill the time available"

60

## 5 day project model



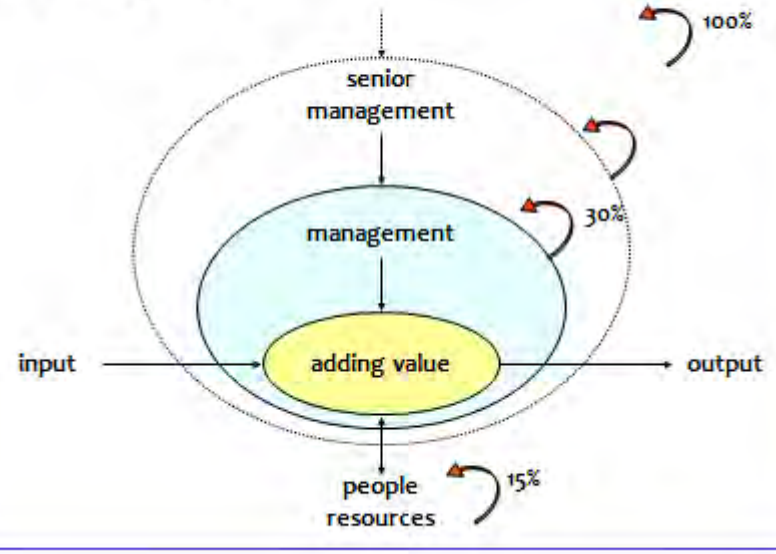

61

## Available TimeBoxes

| activity | ~% | hrs |
|---|---|---|
| Planning | 5 | 2 |
| Requirements | 5 | 2 |
| Global design | 20 | 8 |
| Detail execution | 20 | 8 |
| Review and edit | 20 | 8 |
| Presentation | 5 | 2 |
| Delivery | 10 | 4 |
| Documentation | 5 | 2 |
| Archiving | 5 | 2 |
| Continuity | 5 | 2 |
| total | 100 | 40 |

62

## The managers task




53

## Interrupts

- Boss comes in: "Can you paint my fence?"
- What do you do?

- In case of interrupt, use interrupt procedure

64

## 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.

65

## Estimation techniques used

- Just-enough estimation (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- Changing from optimistic to realistic predictions
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- $0^{th}$ order estimations (ball-park figures)
  - For decision-making in Business Case and Design
- Simple Delphi
  - For estimating longer periods of time in TimeLine
  - For duration of several (15 or more) elements of work
- Simpler Delphi
  - Same, but for quicker insight
  - Recently added by practice
- Calibration
  - Coarse metrics provide accurate predictions
- *Doing something about it* (if we don't like what we see)
  - Taking the consequence
  - Saving time

66

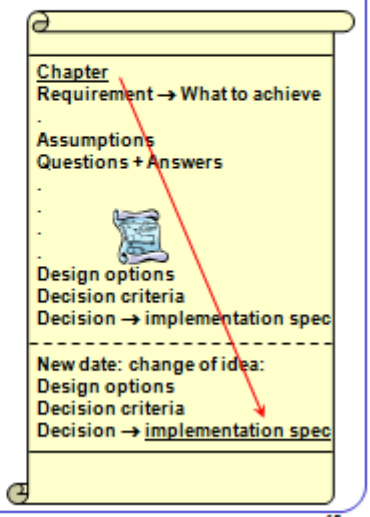
## Simple Delphi estimation

1. Make a list of things we think we have to do in just enough detail
2. Distribute the list among people who will do the work, or who should be knowledgeable about the work
3. Ask them to add what we apparently forgot, and to estimate how much time the elements of work would cost, "as far as you can judge"
4. In a meeting the estimates are compared
5. If estimates differ significantly between estimators, *do not take the average*, but discuss about the *contents* of the work, *not about the estimate* (some may forget to include things that have to be done, some others may think that more has to be done than necessary)
6. After discussion, people estimate individually again and the estimates are compared again
7. Repeat until sufficient consensus (usually not more than once or twice)
8. Add up all the estimates to end up with an estimate for the whole project

67

## DesignLog (project level)

- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + reasoning:
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices
- Implementation

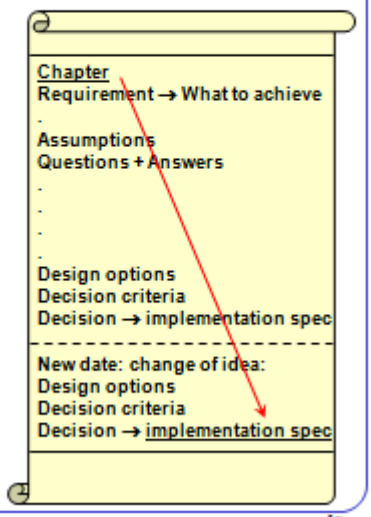



68

## ProcessLog

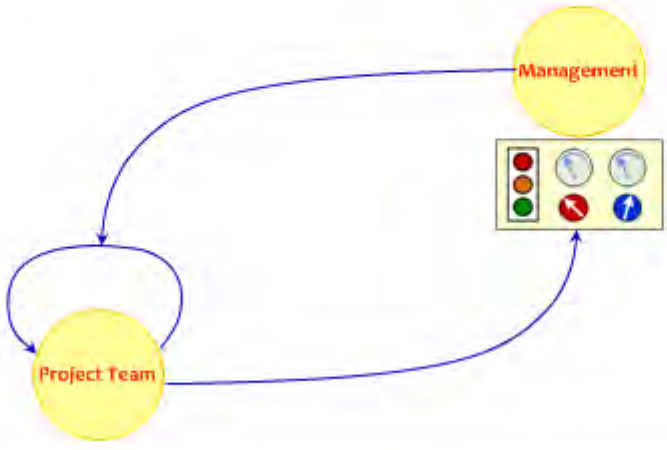
(department / organization level)

- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Graphics (drawings)
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Known techniques
  - Choices + reasoning :
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices
- Implementation




69

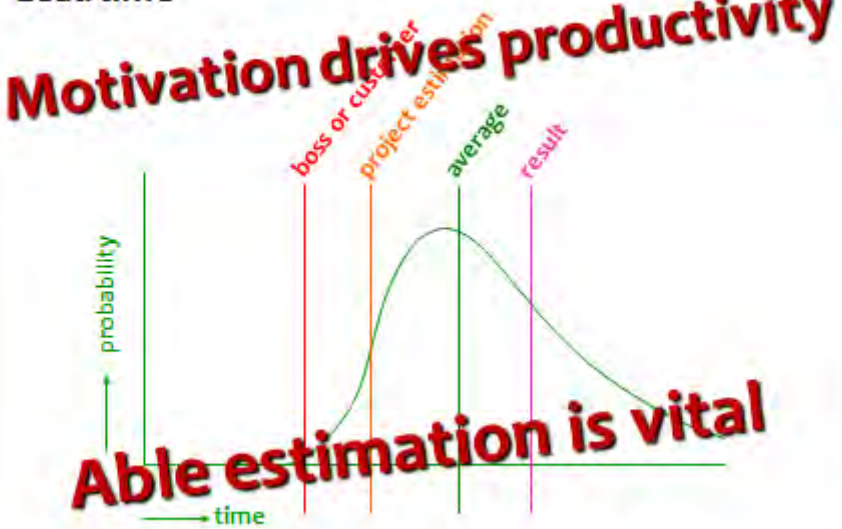
## Local Loop Principle




70

Lead time

Motivation drives productivity




Able estimation is vital

71

Estimation Exercise

Are you an optimistic or a realistic estimator?

Let's find out!

Project:
Multiplying two numbers of 4 figures

How many seconds would you need to complete this Project?

72

## Elements in the exercise

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

73

## Benjamin Franklin's Daily Leadership Standard Work

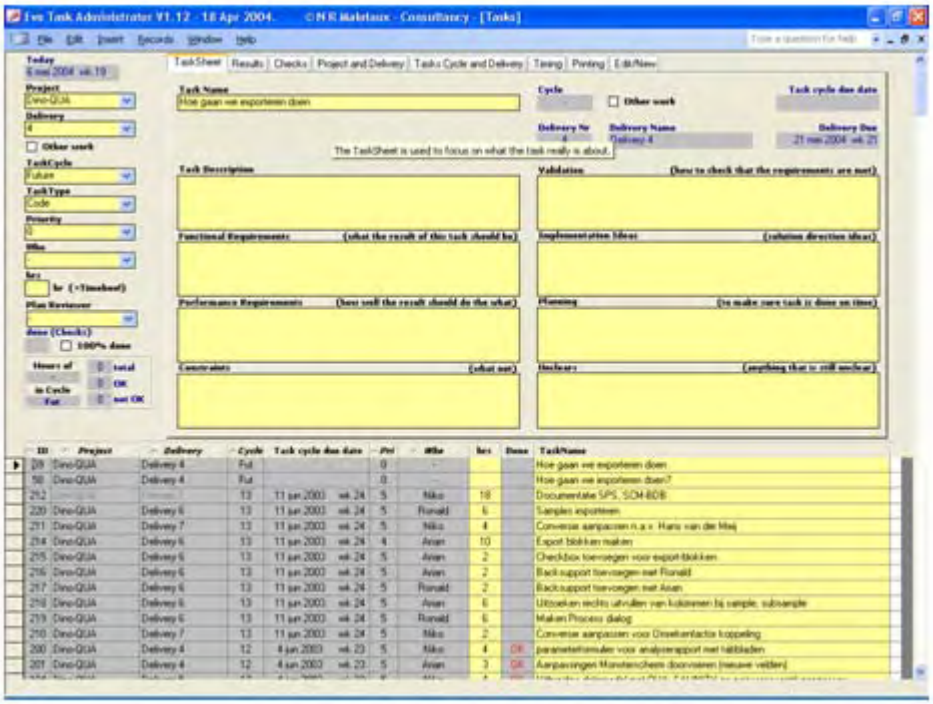
| | |
|---|---|
| 5h | Rise |
| 6h | Ask the question: What good shall I do this day?<br>And which virtue am I focusing on today? |
| 7h | Prosecute the present study and eat |
| 8h | Work |
| 12h | Read, review business progress, eat |
| 13h | |
| 14h | Work |
| 18h | Put things in their places. Ask the question: What good have I done today? How did I perform according to the daily virtue? |
| 19h | Supper |
| 20h | Music or diversion or conversation |
| 21h | Examination of the day |
| 22h | Sleep |

74

## Weekly 3-Step Procedure

1. Individual preparation
   - Conclude current tasks
   - What to do next
   - Estimations
   - How much time available
2. Modulation with / coaching by Project Management
   - Status
   - Priority check
   - Feasibility
   - Commitment and decision
3. Synchronization with group (team meeting)
   - Formal confirmation
   - Concurrency
   - Learning
   - Helping
   - Socializing

75



## Causes of Delay

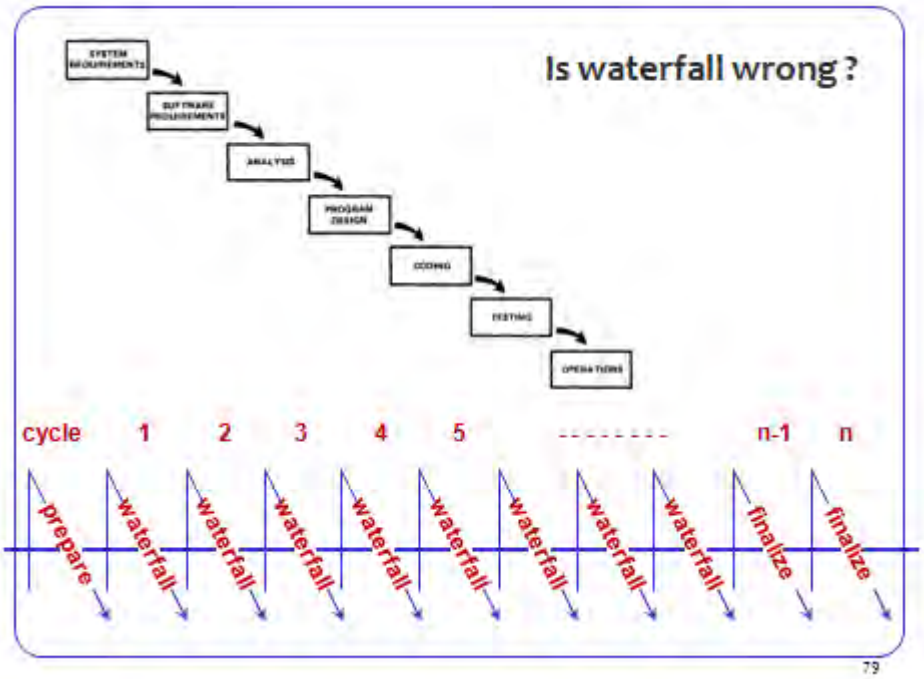
- Some typical causes of delay are:
  - Developing the wrong things
  - Unclear requirements
  - Misunderstandings
  - No feedback from stakeholders
  - No adequate planning
  - No adequate communication
  - Doing unnecessary things
  - Doing things less cleverly
  - Waiting (before and during the project)
  - Changing requirements
  - Doing things over
  - Indecisiveness
  - Suppliers
  - Quality of suppliers results
  - No Sense of Urgency
  - Hobbying
  - Political ploys
  - Boss is always right (culture)
- A lot of delay is avoidable and therefore unjustifiable
- Excuses, excuses: "external factors" being the cause of delays

77

## But I'm not a Project Manager!

- What caused the project being late?
- Could we have prevented the project being late?
- Was delivery time important?
- Was delivery time a requirement?
- Were *all* other requirements really more important?

78


Is waterfall wrong ?
cycle 1 2 3 4 5 - - - - - - - - n-1 n
prepare
waterfall
finalize
79