INCOSE International Symposium 2009 Singapore

Tutorial
21 July 2009

Niels Malotaux

Evolutionary Project Planning

How Systems Engineers can Contribute to Getting and Keeping the Project On Time

N R Malotaux - Consultancy
The Netherlands
+31-30-2288868
+31-30-2288869
niels@malotaux.nl
www.malotaux.nl

Copyright © 2009 by Niels Malotaux. Published and used by INCOSE with permission.

Niels Malotaux

Evolutionary Project Planning

How Systems Engineers can Contribute to 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

How Systems Engineers can Contribute to Getting and Keeping the Project On Time

Niels Malotaux

N R Malotaux Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

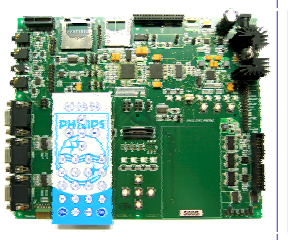
1

Niels Malotaux

Result Management

Project Coach

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



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

2

The Right Result at the Right Time

Quality on Time

- 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

What is the most important Requirement ?

4

Time as a Requirement

- Delivery Time is a Requirement, like all other Requirements
- How come most projects are late ???
- Apparently all other Requirements are more important than Delivery Time
- Are the really?

5

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* at the departure airport, and get their luggage back *as quickly as possible* in *acceptable condition* at the destination airport
- They didn't

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

6

Fallacy of *all* requirements

- We're done when *all* requirements are implemented
- Isn't delivery time a requirement ?
- Requirements are always contradictory
- Perception of the requirements
- Who's requirements
- Do we really know the real requirements?
- Are customers able to define requirements ?
  - Customers specify things they do not need
  - And forget things the do need
- What we *think* we have to do should fit the available time
- Use the Business Case

7

Will your current project be on time ?

- Was your previous project successful and on time ?
- Will your current project be successful and on time ?
- How do you know ?

8

**If your previous project was late,
your current project will probably be late**

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

9

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

10

But we're Systems Engineers !

- 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?

11

Systems Engineers

- Other Engineers
  - Silo thinking
  - Sub-optimizing
  - Gold plating (hobbies)
  - Little attention to interfaces

- Systems Engineers
  - Multi-dimensional thinking
  - Optimizing design decisions over all dimensions
  - All disciplines
  - Whole life-cycle (cradle to cradle)
  - Balancing requirements
  - Including delivery time

12

What *is* On Time ?

- Yesterday?
- Before the next exhibition?
- Managers dream?
- Time to market?
- Time to profit?

Compromise between what is *needed*
and what is *possible*
just like any other requirement

13

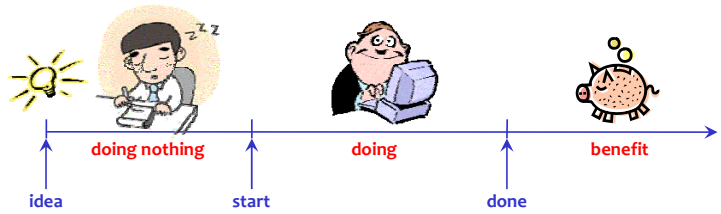
Time to market

- 5000 products per year ≈ 20 products per day
- € 5000 per product
- Profit € 500 per product
- Profit € 10.000 per day

Every day you start later, you'll be done a day later
and miss € 10.000

14

Project ROI

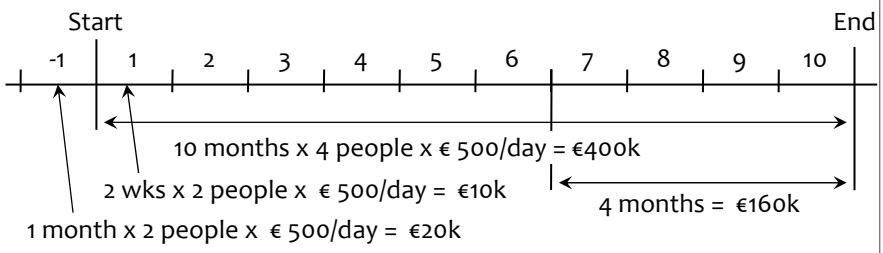


Expected Return on Investment (ROI)

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

15

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* !

16

What could *we* have done to save time?

17

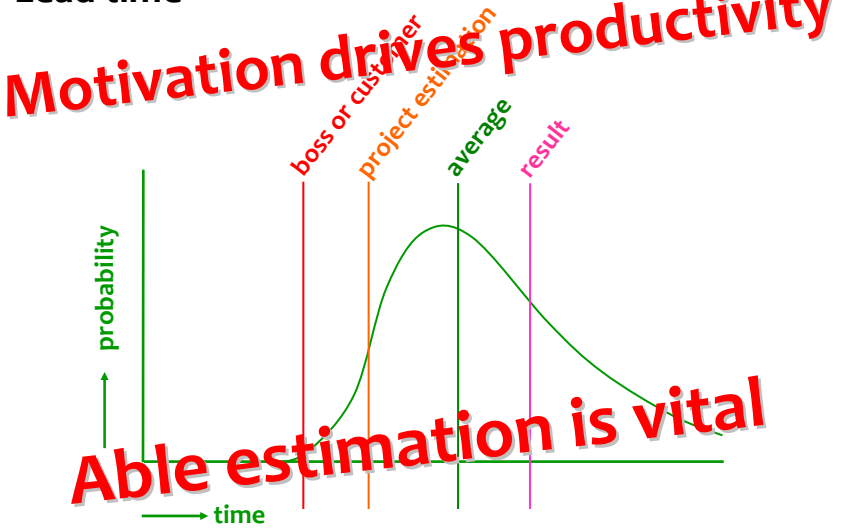
How about this challenge ?

- Getting and keeping the project under control
- Never to be late
- We don't want to fail, we're not going to let it happen
- No excuses needed
- Not stealing from our customer's (boss') purse
- If it's impossible, how quickly will we know ?
- The only justifiable cost is the cost of developing the right things at the right time

18

Lead time

Motivation drives productivity



Able estimation is vital

19

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?

20

Is this what you did?

21

Defect rate

- Before test ?
- After test ?

22

Alternative Design (*how* to solve the requirement)

23

Another alternative design

24

What was the real requirement?

Assumptions, asssumptions ...

Better assume that many assumptions are wrong.

Check !

25

Elements in the exercise

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

26

No excuse

Ignorance
Lack of Education





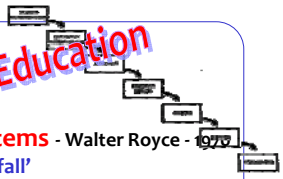



- **Henry Ford**
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - Learning from waste
- **Toyoda's (Sakichi, Kiichiro, Eiji)**
  - Jidoka: Zero-Defects, stop the production line (1926)
  - Just-in-time – flow – pull
- **W. Edwards Deming**
  - 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**
  - Quality Control Handbook (1951, Japan – 1954)
  - Total Quality Management – TQM
  - Pareto Principe
- **Philip Crosby**
  - Quality is Free (1980)
    - Zero-defects (1961)
- **Masaaki Imai**
  - Kaizen: The Key to Japan's Competitive Success (1986)
  - Gemba Kaizen: A Commonsense, Low-Cost Approach to Management (1997)
- **Taiichi Ohno**
  - (Implemented the) Toyota Production System (Beyond Lange-Scale Production) (1988)
  - Absolute elimination of waste – Optimizing the TimeLine from order to cash
- **Shigeo Shingo**
  - Industrial Engineering (1976, 1985 …)

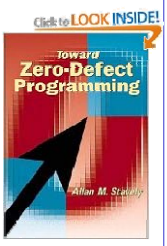
27

There is nothing new

Ignorance
Lack of Education



- **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 - 1976 (and 1988, 2005)
  - Principles of Software Engineering Management - 1988
  - Incremental - Iterative - Learning and consequent adaptation
  - Fast and Frequent Plan-Do-Study-Act, especially Study and Act and then Plan
  - Quantifying Requirements - Real Requirements
  - Defect *prevention* rather than defect removal

28

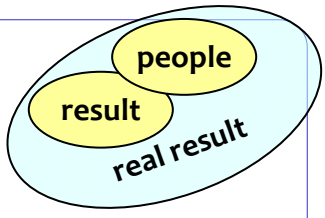
Human Behaviour

- 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 (project) process approaches (as well as developers) *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*

29

Is Human Behavior a risk?



- Human behavior is a risk for the success of the system
  - When human behavior is incorrectly modeled in the system
  - Not because human users are wrong
- Things that can go wrong
  - Customers not knowing well to describe what they really need
  - Users not understanding how to use or operate the system
  - Users using the system in unexpected ways
  - Incorrect modeling of human transfer functions within the system: ignorance of designers of systems engineers
- Actually, the humans aren't acting unpredictably
  - Because it happens again and again

30

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)

31

Intuition

- Makes us react on every situation
- Intuition is fed by experience
- It is free, we always carry it with us
- We cannot even switch it off
- Sometimes intuition is simply wrong
- Coaching is about redirecting intuition

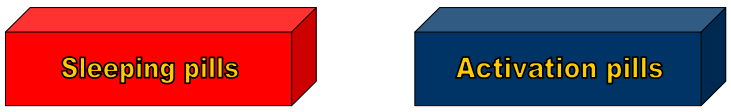
32

Culture

- Latin: Cultus - adoration, worship
- Culture: Ingrained customs
  - Things we learn by mimicking what we experience around us
  - Language
  - Social behaviour
  - Faith
  - Folklore
  - Doing what we're *used to*
  - We don't really realise why we do it, we just do it
  - Experience → intuition → culture
- Once we see other cultures, we can see that our own culture isn't obvious at all; neither is theirs
- Still we judge others through our own cultural (conditioned) spectacles, whether we like it or not

33

Is intuition wrong, or is the design wrong?



34

Communication

- Talking as near as possible along each other

To each other | Along each other

- Don't *assume* we understand: *check !*

35

Communication

- Information exchanged between systems
- 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

36

Perception

- What we intuitively, sub-consciously observe and notice
- What people say and what they do is not always equal
- 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
- If you ask a Stakeholder what's important, believing what he says is dangerous
- Check, double-check !
- Prevent the *perfect solution for the wrong problem syndrome*

37

Logical thinking is not always better

- Intuitive decision is often good
- Logical thinking feeds the sub-consciousness
- Sub-consciousness needs some time

Real Options

- Option to make or abandon a decision
- The later you make the decision, the more information you can have about it
- Options have value until expiration
- On expiration the value has disappeared
- Just in Time delivery

38

Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures

- At the Fatal Day, any excuse is in vain: we failed
- Even if we "couldn't do anything about it"

39

We failed because of politics

- Good politics:
  - People decide differently based on different values
- Bad politics: hidden agenda's
  - Say this, mean that - often even unintentionally
  - Politics thrive by vagueness
  - Facts can make bad politics loose ground

- If you accepted the responsibility for the project, failure because of "politics" is just an excuse
- What did you really do about it ?

40

Culture

- It failed because of the existing culture
  (no good excuse !)
- Culture is the result of how people work together
- Culture can't be changed
- Culture *can* change
- By doing things differently

41

The boss is always right

- Is he or she ?
- Afraid for losing 'face' ?
- How about losing face invisibly ?

42

People like change

- People are not against change
- People (subconsciously) don't like *uncertainty*
- Any Project changes something and thus introduces uncertainty
- How to make sure that the uncertainty remains as shortly as possible
- Before people abandon

43

Murphy's Law

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

Murphy's Law for Engineers:

- 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*

44

Preflection, foresight, prevention

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

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**

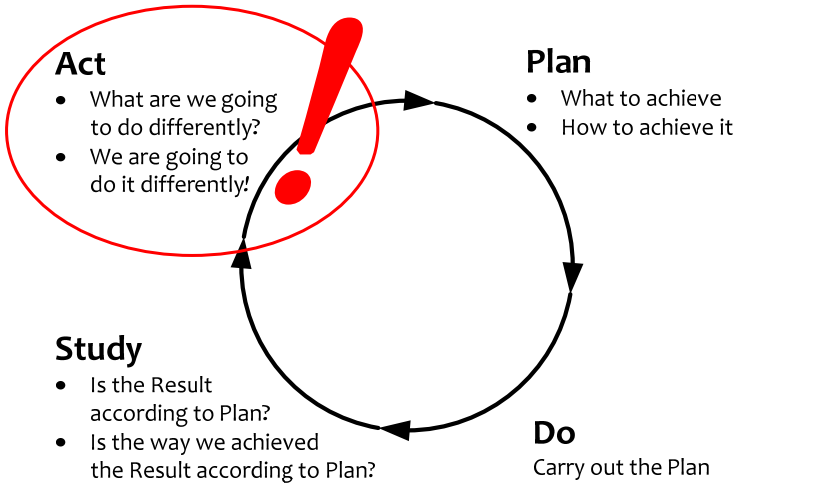
Only with *prevention* we can save precious time

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

45

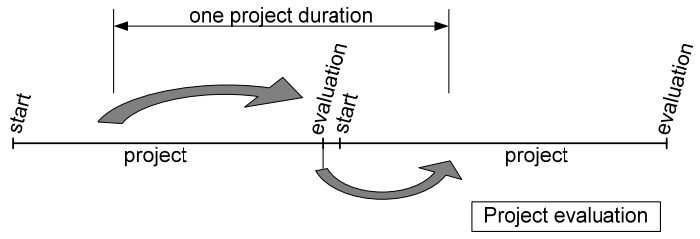
The essential ingredient: the PDSA Cycle

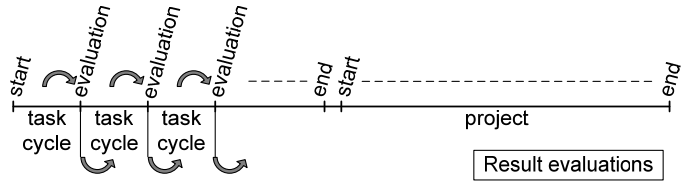
(Shewhart Cycle - Deming Cycle - Plan-Do-Check-Act Cycle - Kaizen)



46

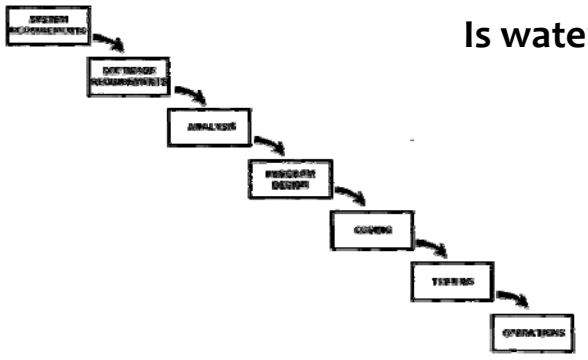
Project evaluations

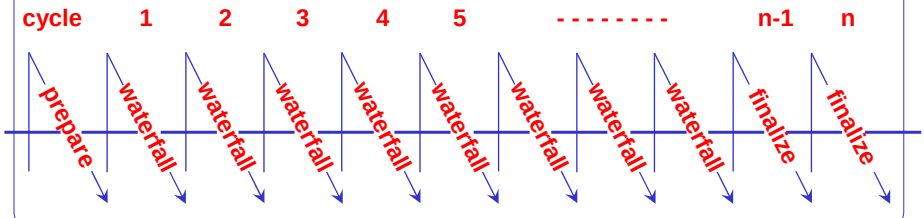




47

Is waterfall wrong ?

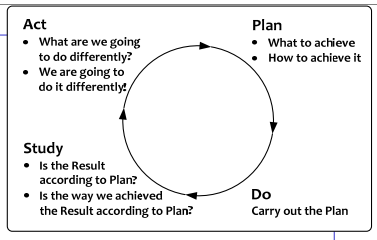




48

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

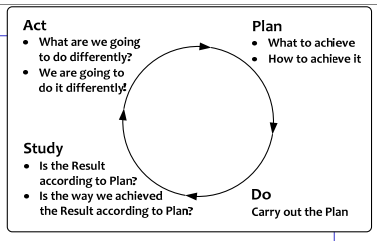
and we can

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

doing the right things right

49

Evo



- Evo (short for Evolutionary...) uses PDSA consistently
- Applying the PDSA-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 find out
- 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

50

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

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

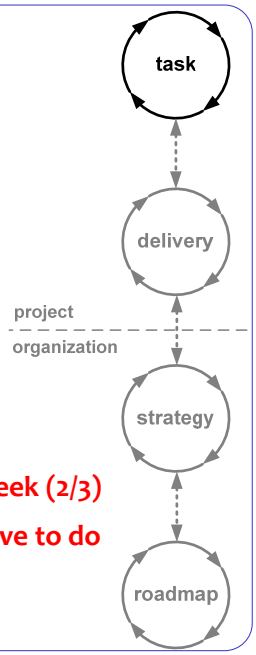
51

Universal Project Goal

- 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

52

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

53

Effort and Lead Time

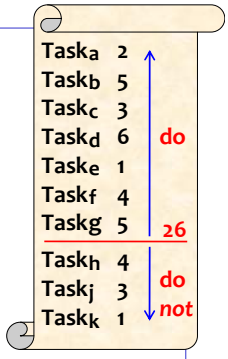
- Days estimation → lead time (calendar time)
- Hours estimation → effort

- 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)

54

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



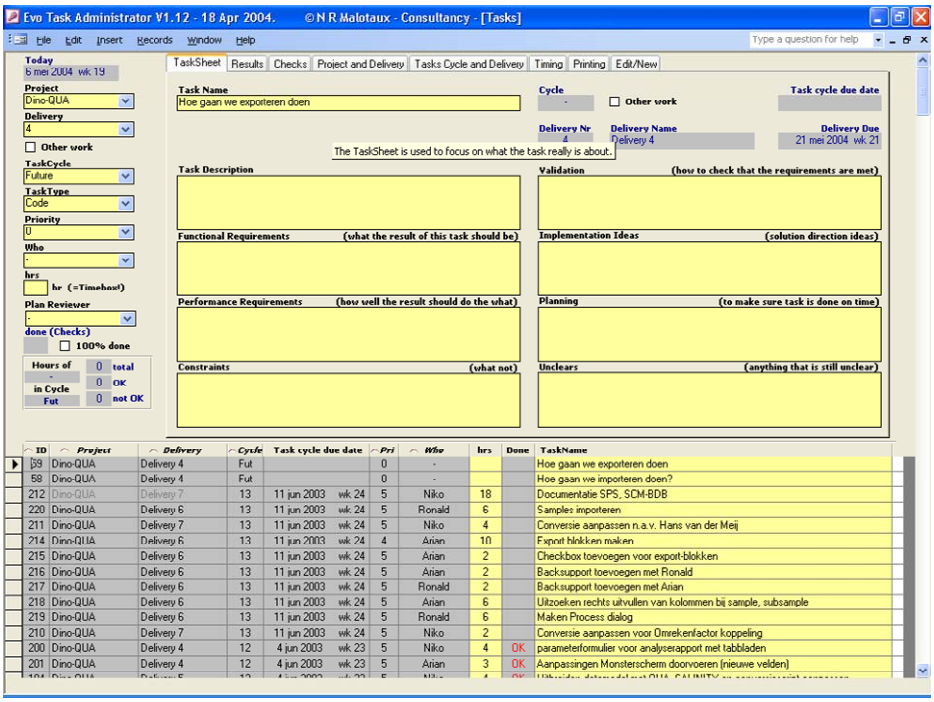
2/3 is default start value
this value works well in development projects

55

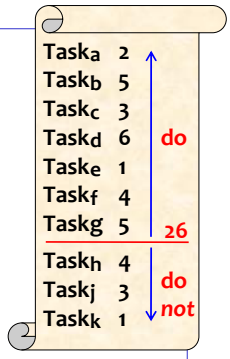
Weekly 3-Step Procedure

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

56



TaskCycle Exercise

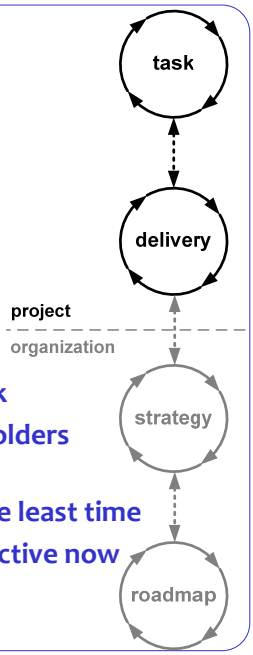


- How much time do you have available
- 2/3 of available time is net plannable time
- What is most important to do (make list)
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr)
- What can you do, and what are you going to do
- What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

58

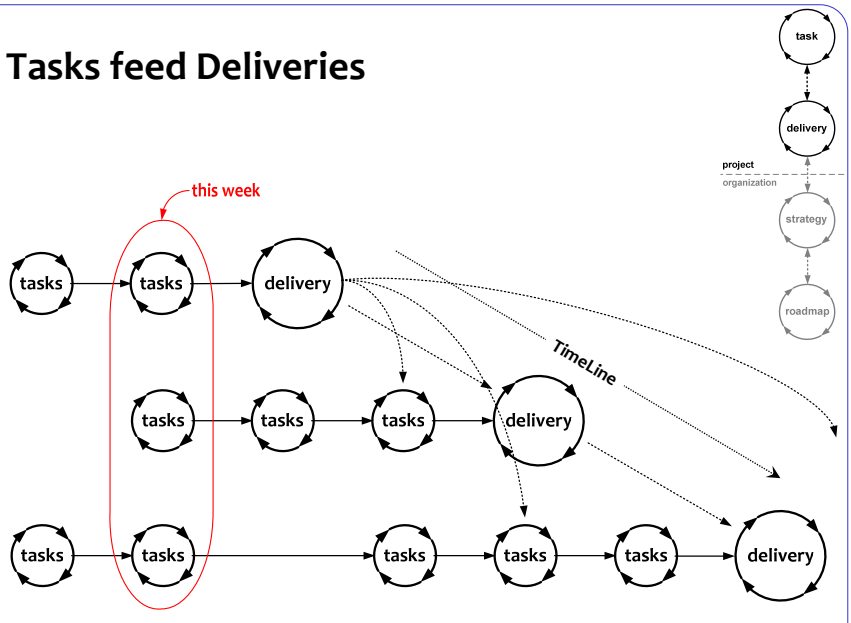
DeliveryCycle

- Are we delivering the *right things*, in the *right order* to the right *level of detail for now*
- Optimizing requirements and checking assumptions
  a. What will generate the optimum feedback
  b. We deliver only to *eagerly waiting* stakeholders
  c. 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



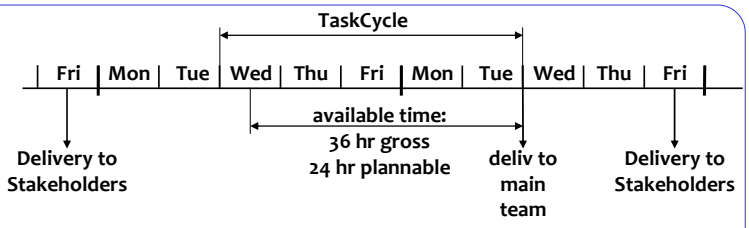
59

Tasks feed Deliveries



60

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
...	

61

Interrupts

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

- In case of interrupt, use interrupt procedure

62

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.

63

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

64

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 (what *not* to do)
- So, we already work more efficiently

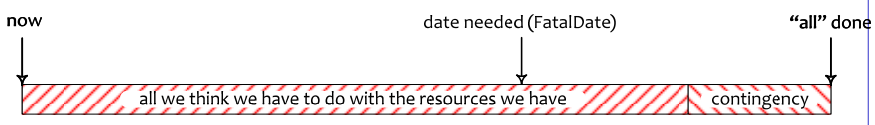
but ...

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

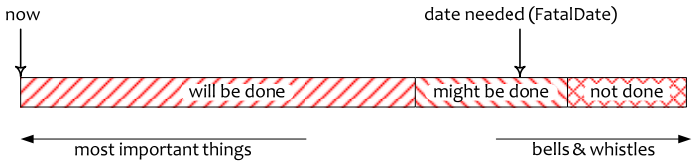
65

TimeLine

What the customer wants, he cannot afford



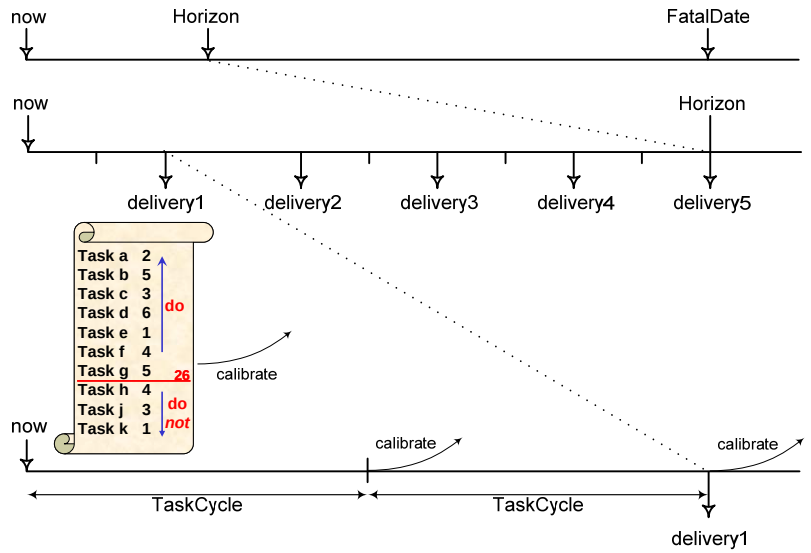
Standard Projects



Evo

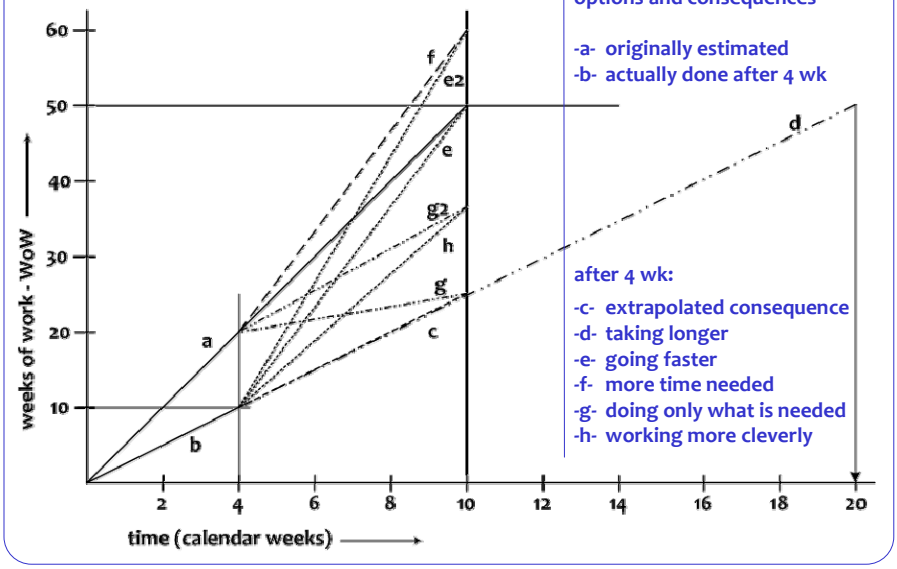
66

Result to Tasks and back



67

Accepting Fate?



Calibrating estimations and options and consequences

-a- originally estimated
-b- actually done after 4 wk

after 4 wk:
-c- extrapolated consequence
-d- taking longer
-e- going faster
-f- more time needed
-g- doing only what is needed
-h- working more cleverly

68

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 in the past

now

predicted Value Still To Earn in the future

then

then2

Calibration Factor

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

Value Still To Earn

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

69

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

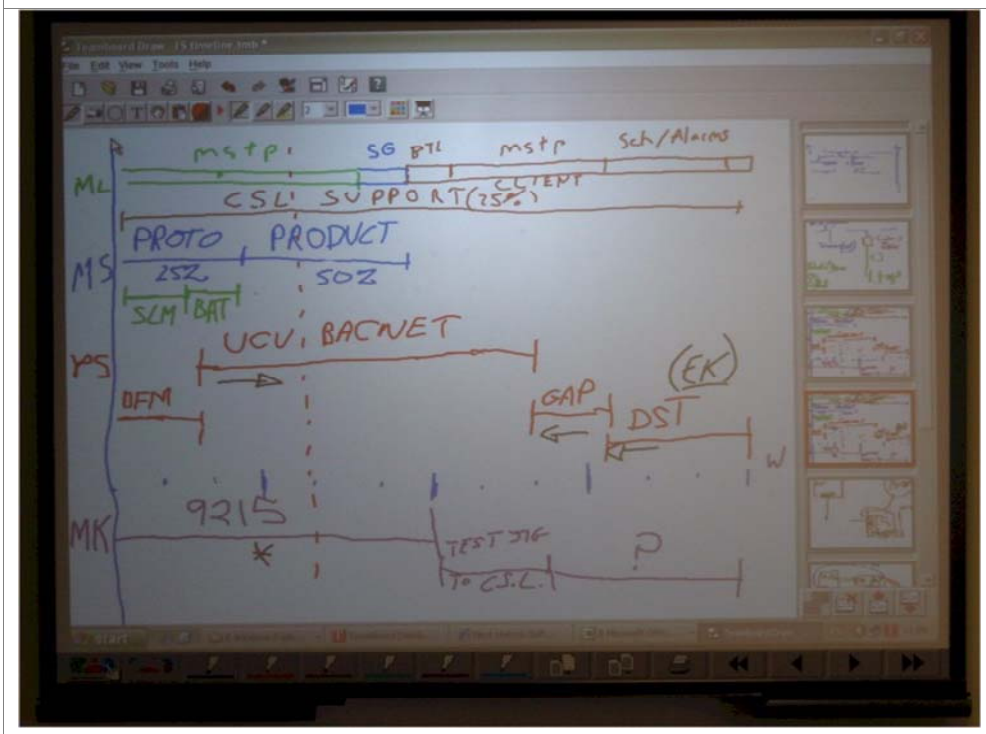
70

Product/Portfolio/Resource Management

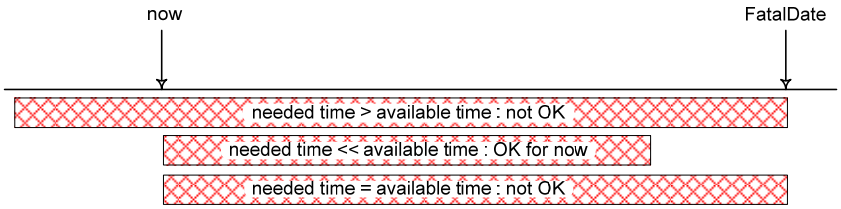
- Current Program/Portfolio/Resource Management is based on hope
- More a game than management

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

71



If it doesn't fit ... count backwards



73

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

74

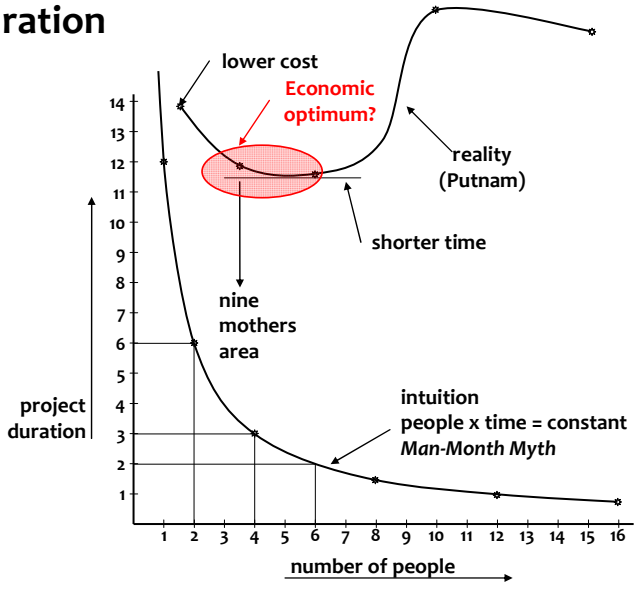
Adding people to a late project ...

makes it later

(Brooks' Law, 1975)

75

Project-duration



76

Saving time

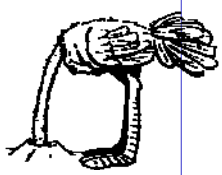
We don't have enough time, but we can save time *without negatively affecting the Result !*

- 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

77

TimeLine

- The TimeLine technique doesn't *solve* our problems
- They help 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 *!*

78

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
- **0th 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

79

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

80

Simple and Simpler Delphi

1. List things to do
2. Distribute the list
3. Add and estimate
4. List estimates
5. Discuss if differences
6. Estimate again
7. Repeat until consensus
8. Add up all the estimates

1. List things to do
2. Distribute the list
3. Add and estimate
4. List estimates: min and max
5. Discuss if differences
6. Agree on value between min and max
7. Add up all the estimates

Even with coarse estimates per element of work,
the sum averages out the variations and can be quite predictive

81

0^{th}- order approximations

- In the Business Case we often use 0^{th}- order estimations
- Order of magnitude
- Better than $0 < guess < \infty$ (any number is better than no number)
- 0^{th} order is better than *no clue*
- 1^{st} order is often less accurate than 0^{th} order
- Using two different ways of estimation for crosscheck
- Errors may average if we estimate several pieces

82

Optimizing Estimation

- Immediately consuming the metrics for learning
- Change from optimistic to realistic estimation in 3 weeks
- Only if we are Serious about Time (Sense of Urgency)
- Using the metrics for calibration of predictions
- Estimation method: Intuition + optimizing intuition
- The person doing the task estimates
- Others should never challenge the estimation
- Estimates are non-negotiable !
- We can and should negotiate about the *contents*

83

Culture

- Latin: Cultus - adoration, worship
- Culture: Ingrained customs
  - Things we learn by mimicking what we experience around us
  - Language
  - Social behaviour
  - Faith
  - Folklore
  - Doing what we're *used to*
  - We don't really realise why we do it, we just do it
  - Experience → intuition → culture
- Once we see other cultures, we can see that our own culture isn't obvious at all; neither is theirs
- Still we judge others through our own cultural (conditioned) spectacles, whether we like it or not

84

Things I heard

- Group is important
  - Project team is a group
- Face
  - We are not perfect, but the customer should never find out
- Cannot say "No"
  - How do you then indicate "no"?
- Is that clear? - Yes
  - If you don't understand:
    - Is the teacher unclear ?
    - Am I stupid ?
- Authority
  - Boss is always right
  - Teacher is always right

85

Things I heard (2)

- Group is responsible
  - Personal responsibility ?
- Survival
  - Win or Win - win ?
- Harmony
  - With whom ?
  - To what extent ?

86

Cultural differences ?

Dutch	You ?
open, direct, explicit, blunt	
informal	
arrogant	
preaching	
assertive	
can say no	
egalitarian, not showing wealth	
consensus	
rules, tasks	
win-win	
little power distance	
authority must be earned	
little brand value	
not spending more than necessary	

87

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

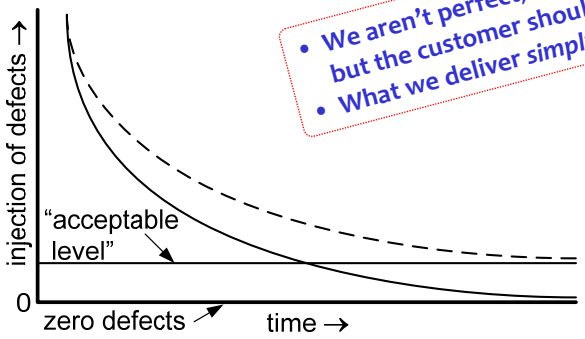
Evo Project Planning

- **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

88

Zero Defects ?

- **Zero Defects** is an asymptote

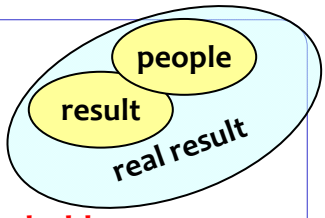


- We aren't perfect, but the customer should't find out
- What we deliver simply works

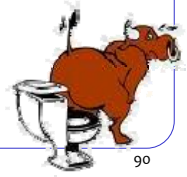
- When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately

89

Stakeholders are people

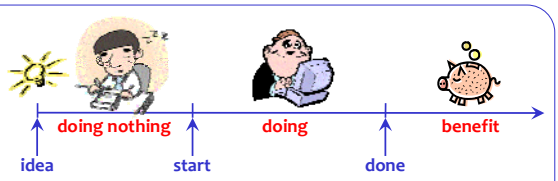


- Every project has some 30±20 Stakeholders
- Stakeholders have a stake in the project
- The concerns of Stakeholders are often contradictory
  - Apart from the Customer *they don't pay*
  - So they *have no reason to compromise !*
- Some Stakeholders are victims of the project
  - They have no reason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



90

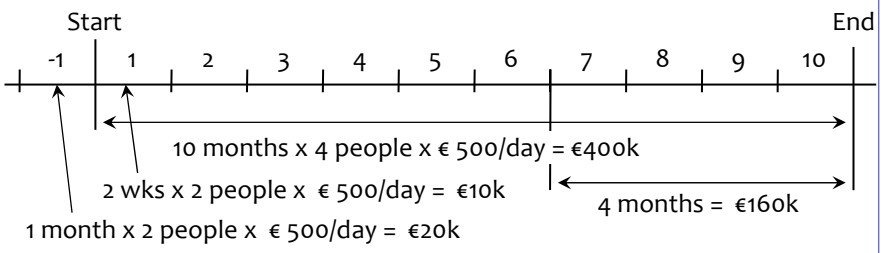
Business Case



- *Why* to improve
- Drives the decision making processes
- To continually align the Projects progress to the dynamic business objectives
- Stakeholders
- Total LifeCycle - cradle to cradle
- Expected Return on Investment (ROI)
  - \+ Benefit of doing - huge (otherwise other projects would be more rewarding)
  - – Cost of doing - project cost, usually minor compared with other costs
  - – Cost of doing nothing - every day we start later, we finish later
  - – Cost of being late - lost benefit
  - – Loss of doing nothing at all - losing benefit from obsolete result

91

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* !

92

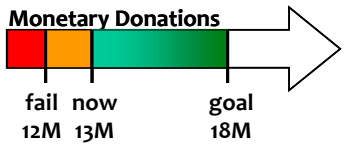
Requirements Case

- 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 2009

93

Objective: Monetary Donations



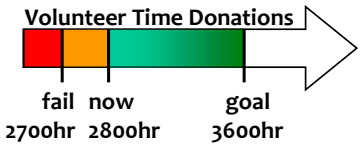
Name Monetary Donations
Scale Euro's donated to non-profits through our website
Meter Monthly Donations Report

Fail 12M
Now 13M [2008] ← Annual Report 2008
Goal 18M [2009]

Ref Ryan Shriver
ACCU Overload Feb 2009

94

Objective: Volunteer Time (Natura) Donations



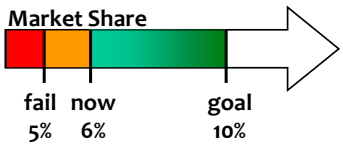
Name Volunteer Time Donations
Scale Hours donated to non-profits through our website
Meter Monthly Donations Report

Fail 2700 hr
Now 2800 hr [2008] ← Annual Report 2008
Goal 3600 hr [2009]

Ref Ryan Shriver
ACCU Overload Feb 2009

95

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 2009

96

Impact Estimation	Monthly Donations	Facebook integration	Image & video uploads	Total effect for requirement
Market share 6% → 10%	30% ±20%	30% ±20%	20% ±10%	80% ±50%
€€ donations 13M€ → 18M€	80% ±30%	30% ±30%	50% ±20%	160% ±80%
Time donations 2800hr→3600hr	10% ±10%	50% ±20%	80% ±20%	140% ±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 on budget	70% ±30%	40% ±20%	100% ±40%	
Total effect / total cost	120/70 = 1.7 0.6 … 4.5	110/40 = 2.8 0.7 … 9	150/100 = 1.5 0.9 … 2.9	
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	

Ref Ryan Shriver - ACCU Overload Feb 2009

97

Design and requirements

- Finding the best compromise between the conflicting requirements
- All requirements are equal, but some are more equal than the others
- Some aren't really requirements
- Some elements will never be used
- Some requirements are incorrect
- A lot of real requirements are unexplored

98

Design is always a compromise

- Design is the process of collecting and selecting options how to implement the requirements
- The Requirements are *always* conflicting

example:

- Performance 
- Budget (time, money) 

99

Design Process

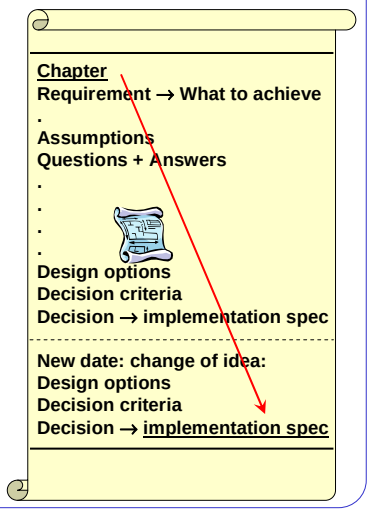
- Collect obvious design(s)
- Search for *one* non-obvious design
- Compare the relative ROI of the designs
- Select the best compromise
- Describe the selected design
- Use a DesignLog to document the decision process

- Books:
  - Ralph L. Keeyney: Value Focused Thinking
  - Gerd Gigerenzer: Simple Heuristics That Make Us Smart

100

DesignLog (project level)

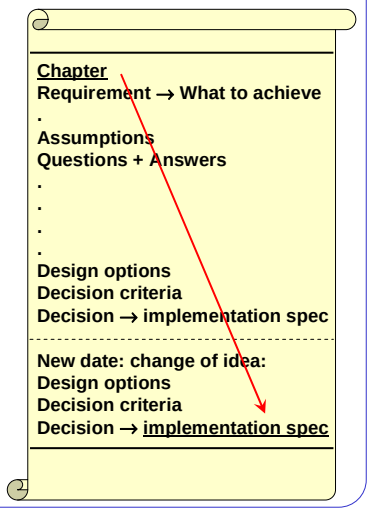
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirements
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + argumentation:
 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



101

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
  - Related requirement
  - Assumptions
  - Questions
  - Known techniques
  - Choices + argumentation:
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices
- Implementation



102

Some Examples from Practice

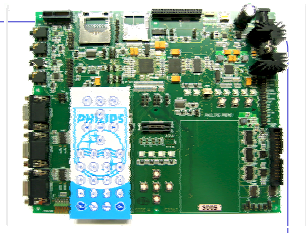
Niels Malotaux

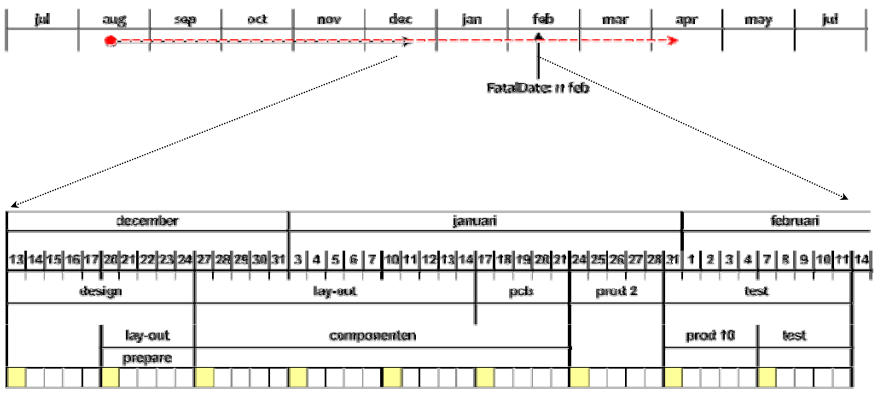
N R Malotaux Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

103

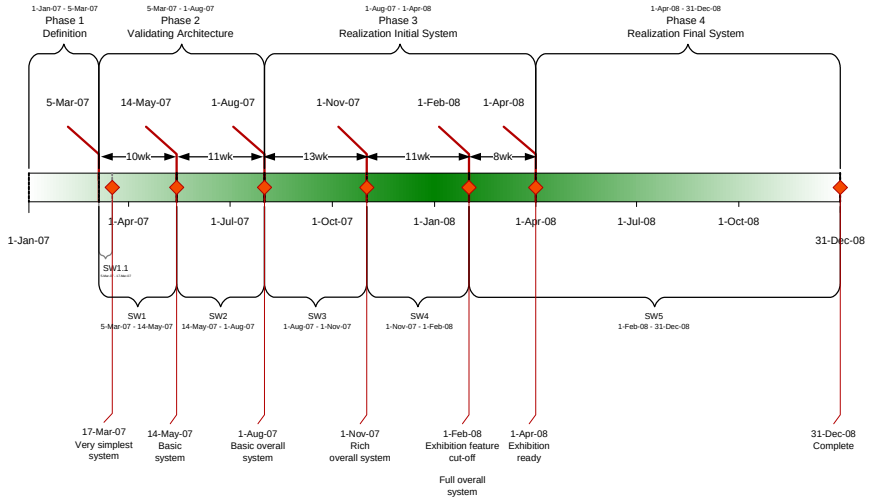
TimeLine planning





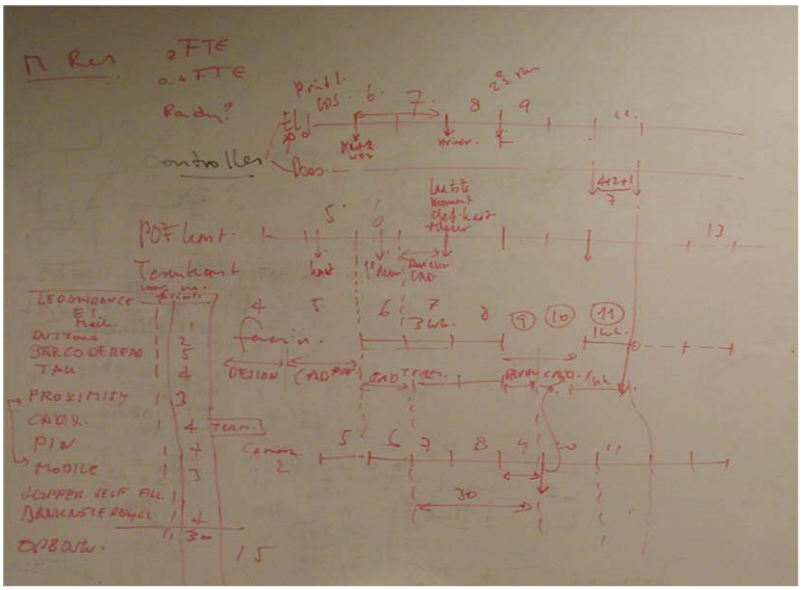
104

TimeLine example



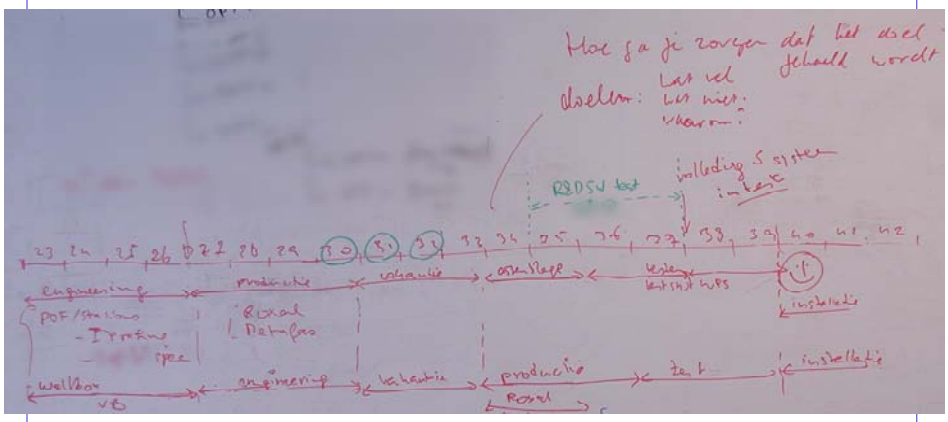
105

Whiteboard TimeLine Planning



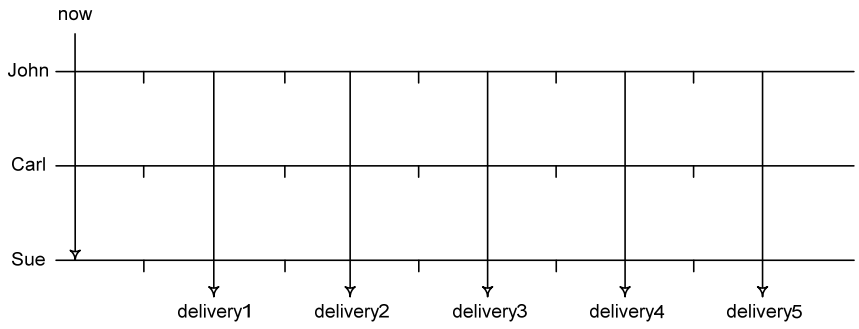
106

Whiteboard TimeLine Planning

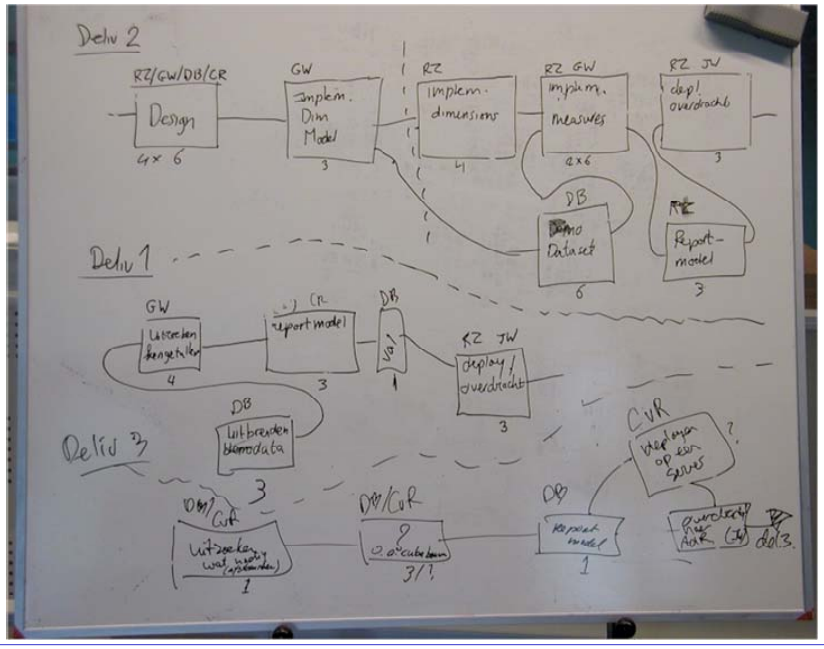


107

Making individual TimeLines



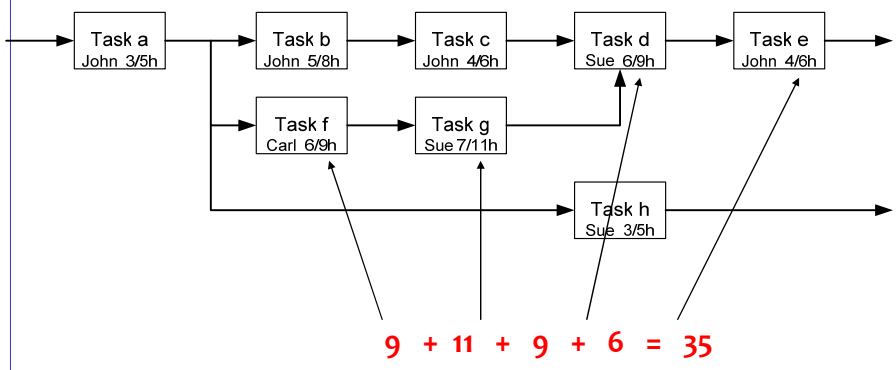
108



109

PERT (Project Evaluation Review Technique)

used for *Designing* a Delivery



110

We have a QA problem !

- Large stockpile of modules to be tested
- Estimate: will cost half year of testing
- You shall do Full Regression Tests
- Full Regression Tests take about 15 days each
- QA is bottleneck
- Can we do something about this?

111

What did we do

- Projector
- Seeing the extent of the problem
- No dilution of quality

112

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

Line	Activity	Estim	Alter native	Junior tester	Devel oper	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		

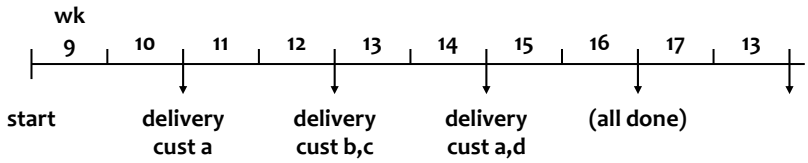
113

What did we do

- Projector
- Seeing the extent of the problem
- No dilution of quality
- Developers stop developing
- Minimizing Testers bottleneck

114

TimeLine



115

Result

- Tester empowered
- "Year of work" done in 9 weeks
- Customers systematically happy
- Kept up with development ever since
- Increased revenue

Recently:

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

116

Summary

- Using Plan-Do-Check-Act
- TaskCycles to organize the work
- DeliveryCycles to make Stakeholders happy
- TimeLine to predict what will happen
- And to take the consequence !

- Doing something about it

117

TimeLine Summary

- Cutting the work into about 20 chunks
- Estimating
- Adding up
- Usually doesn't fit in the available time
- Find strategies to solve the dilemma
- Find 'optimum' strategy
- Predict what will happen when
- Repeat every week, keeping predictions up-to-date

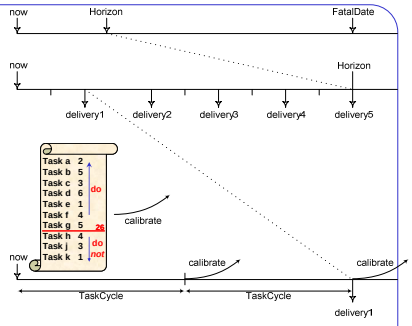
118

Who is the customer of Testing and QA ?

- **Deming:**
  - *Quality comes not from testing, but from improvement of the development process. Testing does not improve quality, nor guarantee quality. It's too late. The quality, good or bad, is already in the product. You cannot test quality into a product.*
- **Developers are the customer**
- **Testing is a project to run along and synchronized to the development project**
- **Testers to help developers to learn to become perfect**

119

TimeLine exercise for *your* Project



- **Try to describe the TimeLine for your project**
- **What do you have to do the coming 10 weeks**
- **Can you define the first few deliveries**
  - *What* to do, for *whom*, and *why*
- **Is your TaskCycle plan still adequate ?**

120

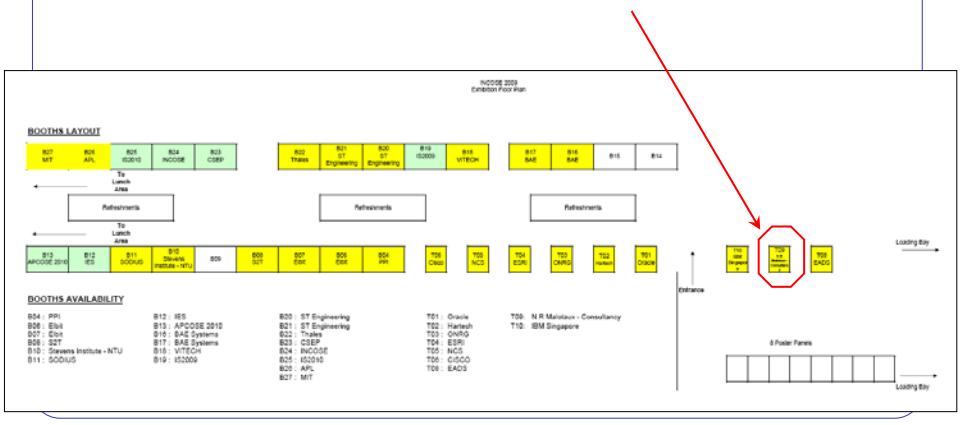
Links

- **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)
- **www.malotaux.nl/nrm/Evo/ETAF.htm**
  Download the Evo Task Administrator (ETA) tool (expects MSAccess 2000~2003)

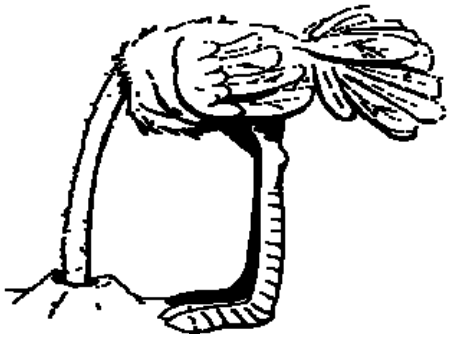
121

Visit T09

Booklets available



122



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

123

Evolutionary Project Planning

How Systems Engineers can Contribute to Getting and Keeping the Project On Time

Niels Malotaux

N R Malotaux Consultancy

+31-30-228 88 68 niels@malotaux.nl www.malotaux.nl

124