Risk Management User Forum - Bremen

October 2005

Controlling Risk *by design*

Niels Malotaux

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

Risk Management User Forum - Bremen 2005

Niels Malotaux

Controlling Risk *by Design*

Niels Malotaux

Niels Malotaux is an independent consultant and project coach, teaching immediately applicable methods for delivering Quality On Time to projects and organizations. He has some 30 years experience in designing 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 teaching and coaching projects to deliver Quality On Time. Since 2001 he coached some 30 projects at 11 different organizations in the Netherlands, Belgium and USA. He is a frequent speaker at conferences.

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 Generation 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
Netherlands
tel +31-30-228 88 68
fax +31-30-228 88 69
mob +31-6-5575 3604
niels@malotaux.nl
www.malotaux.nl/nrm/English

Controlling Risk *by Design*

Niels Malotaux

N R Malotaux Consultancy

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

1

Risk?

- I hardly use the word "risk"
- A project is there to decrease the risks of failure to an acceptable minimum
- Known risks are controlled *by design* (by process) examples:
  - *Estimates are Time Boxes* *(tracking hardly needed)*
  - *Active Synchronization* *(dealing with the bad world around the project)*
  - *Interrupt process* *(dealing with perceived sudden requirements changes)*
- The techniques used are called "Evo"
- Evo projects are successfully concluded, typically in 30% shorter time

2

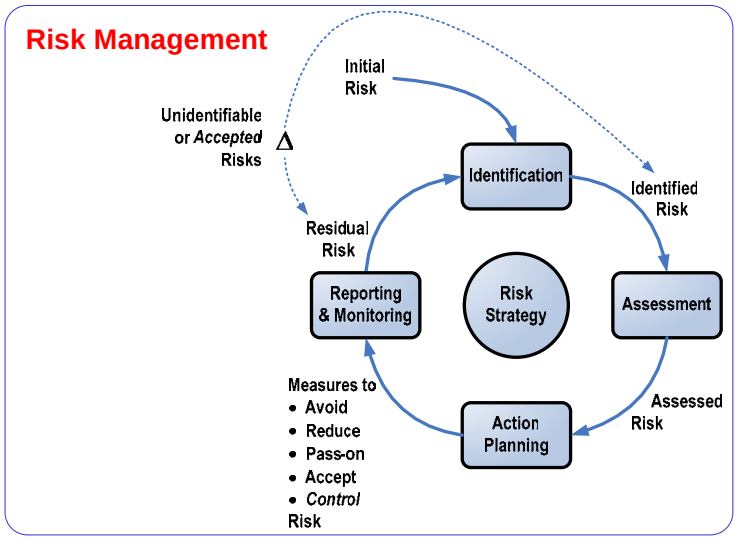
Risk Definition

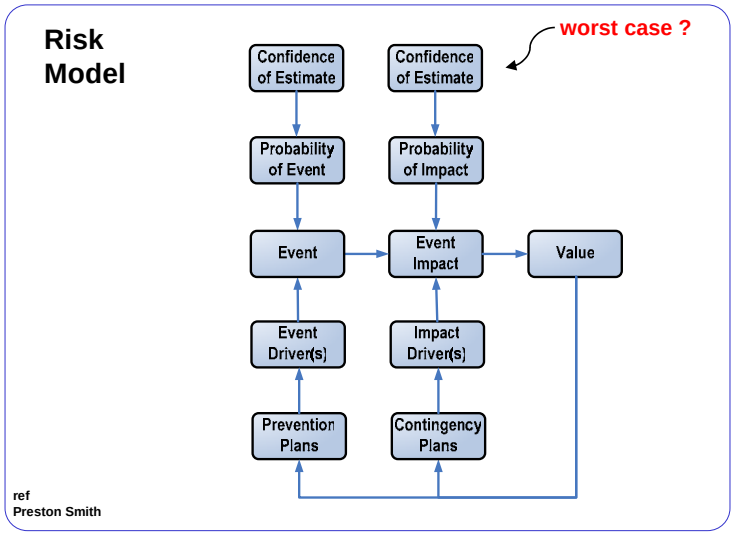
An uncertain event or condition that,
if it occurs,
has a positive or negative effect
on a project's objectives

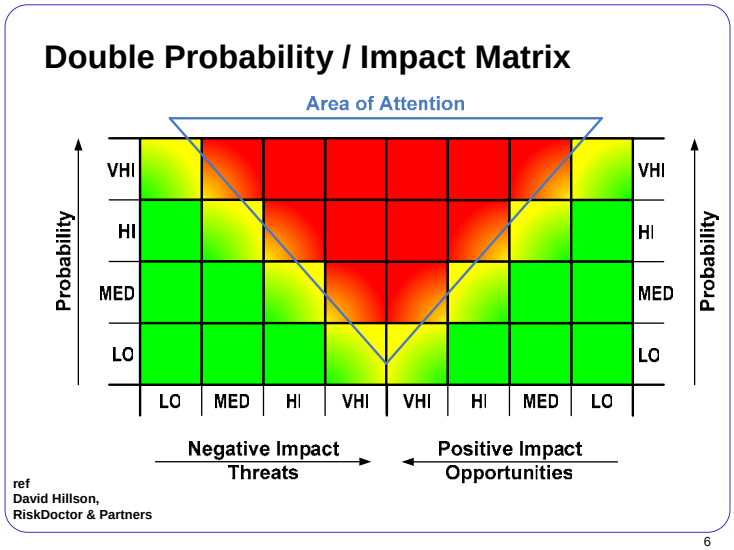
(PMBOK)

- 0% or 100% probability is not a risk
- Positive risk is also called: opportunity

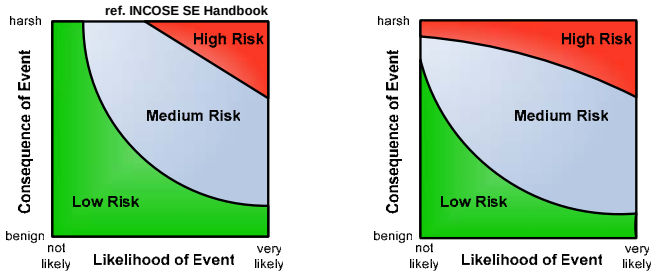
3


Risk Management
Initial Risk
Unidentifiable or Accepted Risks
Δ
Identification
Identified Risk
Residual Risk
Reporting & Monitoring
Risk Strategy
Assessment
Measures to
• Avoid
• Reduce
• Pass-on
• Accept
• Control
Risk
Action Planning
Assessed Risk
4


Risk Model
worst case ?
Confidence of Estimate
Confidence of Estimate
Probability of Event
Probability of Impact
Event
Event Impact
Value
Event Driver(s)
Impact Driver(s)
Prevention Plans
Contingency Plans
ref
Preston Smith
5


Double Probability / Impact Matrix
Area of Attention
Probability
VHI
HI
MED
LO
LO MED HI VHI VHI HI MED LO
Negative Impact
Threats
Positive Impact
Opportunities
ref
David Hillson,
RiskDoctor & Partners
6

Risk priorities?



Risk Priority = Likelihood x Consequence ??

7

Mathematical Risk Management can be risky

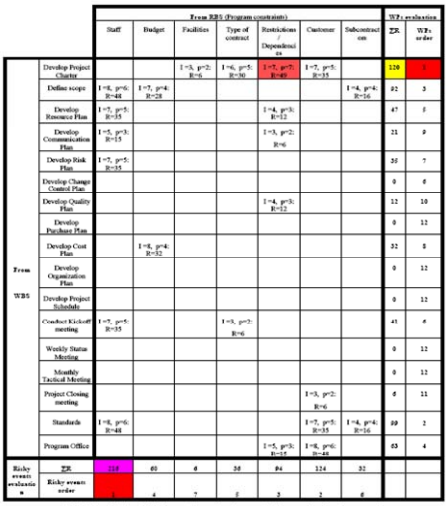


Exhibit 3 - Matrix RBM for a software development with a cardinal scale approach

ref
Carlo Rafele,
David Hillson,
Sabrina Grimaldi

8

Risk literature

- **Risk principles are quite simple**
- **Implemetation is vague**

9

The Goal of a project

- **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**
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

10

Too

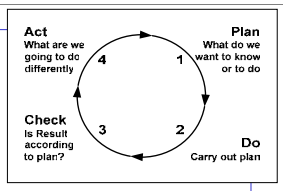
- **Too late** — **time**
- **Over budget** — **cost**
- **Incorrect** — **result**

} **Result**

- **Too early** — **started too early?**
- **Under budget** — **what to do with the savings?**
- **Too good** — **law of diminishing returns?**

11

Knowledge
how to achieve the goal



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

then we can

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

and we can

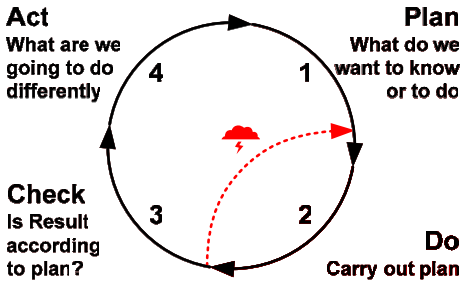
- **Spot problems quicker, allowing more time to do something about them**

doing the right things

doing the right things right

12

The PDCA cycle



13

Absolutes of Quality

- Conformance to requirements
- Obtained through prevention
- Performance standard is zero defects
- Measured by the price of non-conformance (PONC)

Philip Crosby, 1970

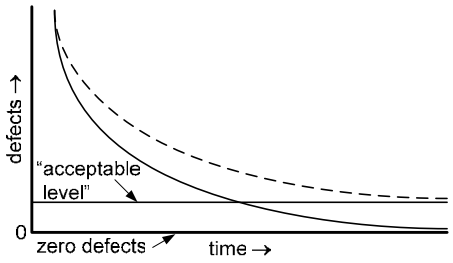
- The purpose is customer success
  (not customer satisfaction)

Added by Philip Crosby Associates, 2004

14

Is defect free software possible?

- Zero Defects is an asymptote



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

15

Attitude

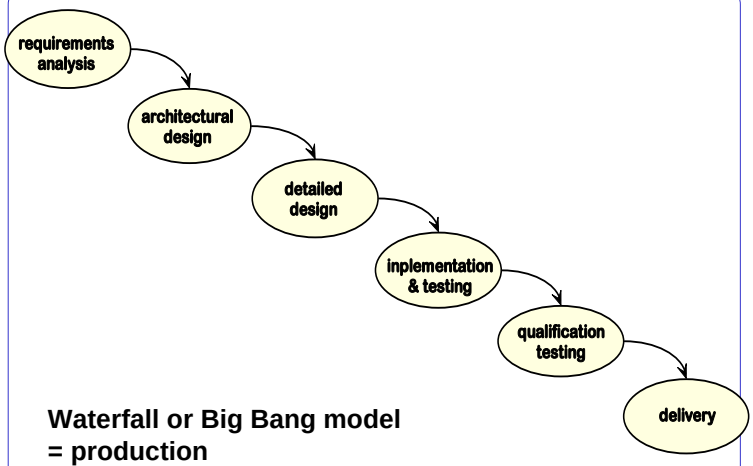
- As long as we think problem free results are impossible, we will keep producing problems
- From now on, we don't want to make mistakes any more
- We feel the failure (if we don't feel failure, we don't learn)
- If we deliver a result, we are sure it is OK and we are surprised when there proves to be a problem after all
- We do what we can to improve (continuous PDCA)

16

Murphy

- Whatever can go wrong, will go wrong
- This is not condoning defects
- This doesn't mean that we should accept fate
- It means that we should check all possibilities which can go wrong and make sure that they don't go wrong

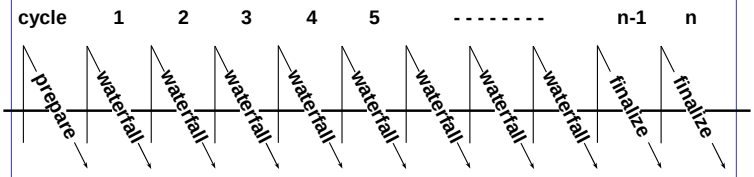
17



Waterfall or Big Bang model
= production
= fixed contract model (signed with blood)

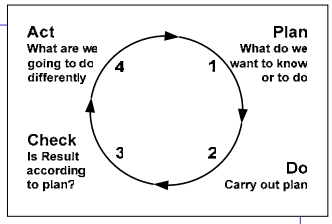
18

Using many waterfalls
of growing functionality



19

Evo



- Evo (short for Evolutionary...) uses this knowledge to the full
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for *Product*, *Project* and *Process*, based on ROI
- A desire to Learning how to be better
- Projects seriously applying Evo, routinely conclude successfully on time, or earlier, *by design*
- Proactively anticipating problems before they occur, working to prevent them

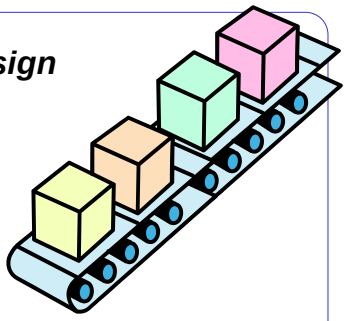
20

We are constantly optimizing

- The product
  how to arrive at the most effective product (goal !)
- The project
  how to arrive at the most effective product effectively and efficiently
- The process
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

21

Controlling Risk *by design*



- **Every project is unique**
  (otherwise it's production)

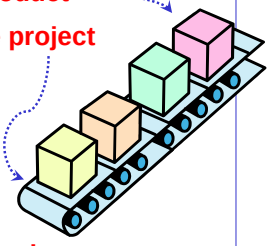
however

- **A lot is always the same:**
  - **Every project is done by people**
  - **There are many similarities (*known* risks)**
  - **So, a lot is predictable**
  - **Engineers control risks *by design* (= *engineering*)**

22

Known risks are hardly risks

- **Most of the real risks are in the product**
- **Most of the known risks are in the project**

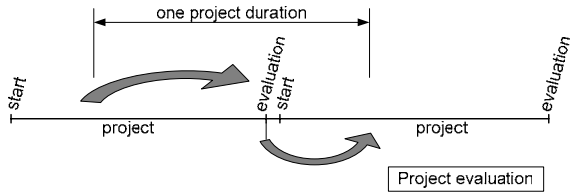


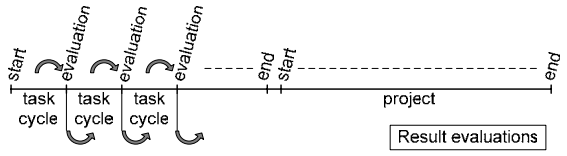
- **We don't only design the product,**
- **We also *design the project***

- **If we control 80% of the risks by design**
- **We have more time to handle the 20% *real* risks**

23

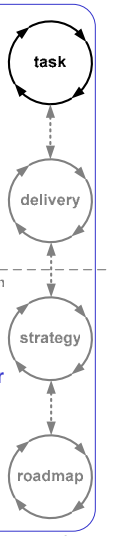
Project evaluations





24

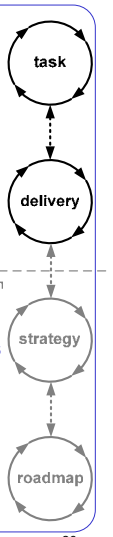
Cycles in Evo



- **Weekly Task Cycle** (organizing the work)
  - Are we *doing* the *right things*, in the *right order,* to the right *level of detail*
  - We optimize estimation, planning and tracking abilities to better predict the future
  - We select the highest priority tasks, we never do any lower priority tasks, we never do undefined tasks
  - There are only about 26 plannable hours in a week
  - In the remaining time: we do whatever else we have to do
  - Tasks are always done, 100% done

25

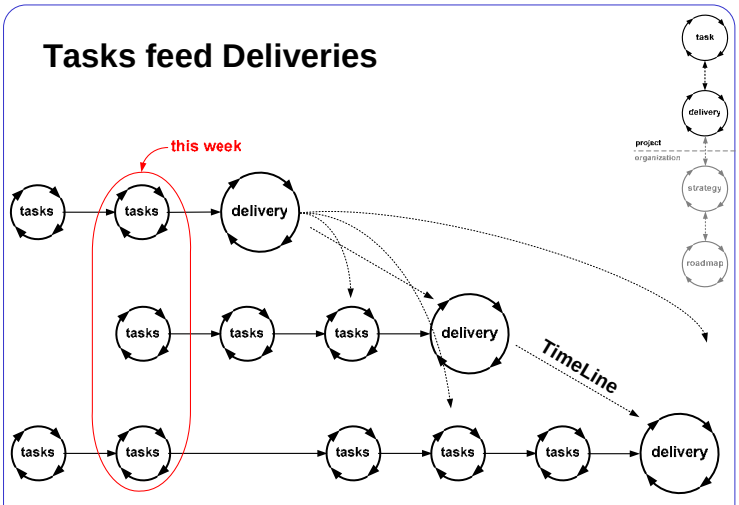
Cycles in Evo



- **Weekly Task Cycle** (organizing the work)
- **Delivery Cycle** (producing useful Results)
  - Are we *delivering* the right things, in the right order to the right level of detail
  - We optimize the requirements and check the assumptions
  - We seek to deliver the juiciest, most important stakeholder values that can be made in the least time
  - Or what will make Stakeholders more productive
  - Or at least what will generate the optimum feedback
  - A delivery takes never more than 2 weeks

26

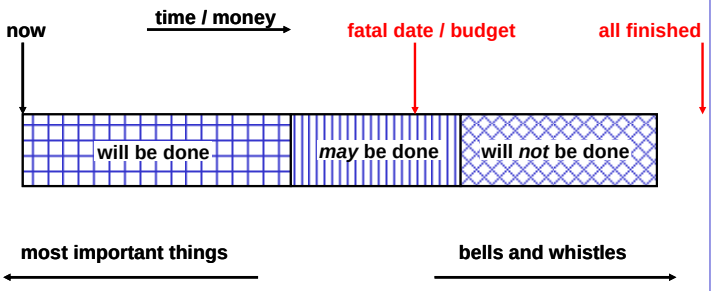
Tasks feed Deliveries



27

TimeLine

What the customer wants, he cannot afford



28

Priorities

Better 80% 100% done, than 100% 80% done

Let it be the most important 80%

29

Planguage (determining priorities)

RQ27: **Maximum Response Time**
Scale: **Seconds between <asking> for information and <appearance> of it.**
Meter: **Add a function to the software to measure the maximum response time value and the range of values per working day.**

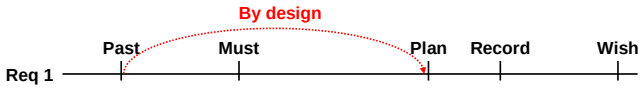
Benchmarks:
Past: **3 sec (our previous product)**
Current: **0.6 sec [competitor y, product x, 2001] ← Marketing Survey Jan 2002**
Record: **0.2 sec [competitor x, product y]**

Requirements:
Must: **1 sec [99%] ← project-contract**
Must: **1.5 sec [100%] ← project-contract**
Plan: **0.5 sec ← project-contract**
Wish: **0.2 sec [2005] ← customer's head of R&D, meeting 19 Feb 2003, see <document ...>**

Note: **Less than 0.2 sec is not noticed by the user, so there is no use in trying to be better than 0.2 sec**

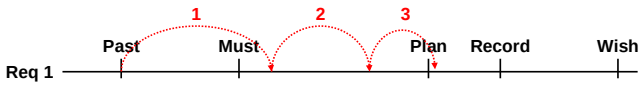
30

Design to Multidimensional Quality Requirements



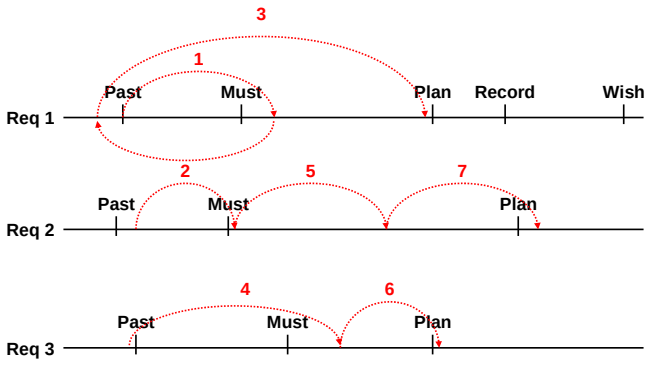
31

Design to Multidimensional Quality Requirements



32

Design to Multidimensional Quality Requirements



33

Impact Estimation

	On-line Support	*On-line Help*	*Picture Handbook*	*On-line Help + Access Index*
Learning 60 minutes <-> 10 minutes				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3 min.	±5 min.	±10 min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%?	±10%
Evidence	Project Ajax: 7 minutes	Other Systems	Guess	Other Systems + Guess
Source	Ajax Report, p.6	World Report, p.17	John B	World Report, p.17 + John B
Credibility	0.7	0.8	0.2	0.6
Development Cost	120 K	25 K	10 K	26 K
Performance to Cost Ratio	110/120 = 0.92	100/25 = 4.0	60/10 = 6.0	104/26 = 4.0
Credibility-adjusted Performance to Cost Ratio (to 1 decimal place)	0.92*0.7 = 0.6	4.0*0.8 = 3.2	6.0*0.2 = 1.2	4.0*0.6 = 2.4

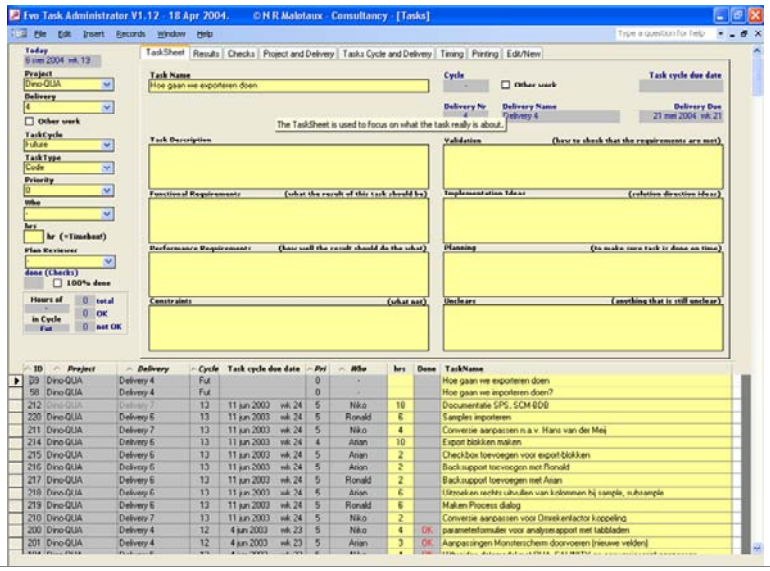
ref
Tom Gilb
Competitive Engineering

34

Types of Tasks

1. **Tasks done within estimated time (= timebox)**
2. **Analysis Tasks (*too short* timebox)**
   - What do you know now
   - What do you still not know
   - What do you still have to know
   - Which tasks can you define
3. **Mis-estimated tasks** (we're only human)
   - Feed the disappointment about the failure to your experience/intuition mechanism
   - What did you do
   - What did you not do
   - What do you still have to do
   - Which tasks can you define

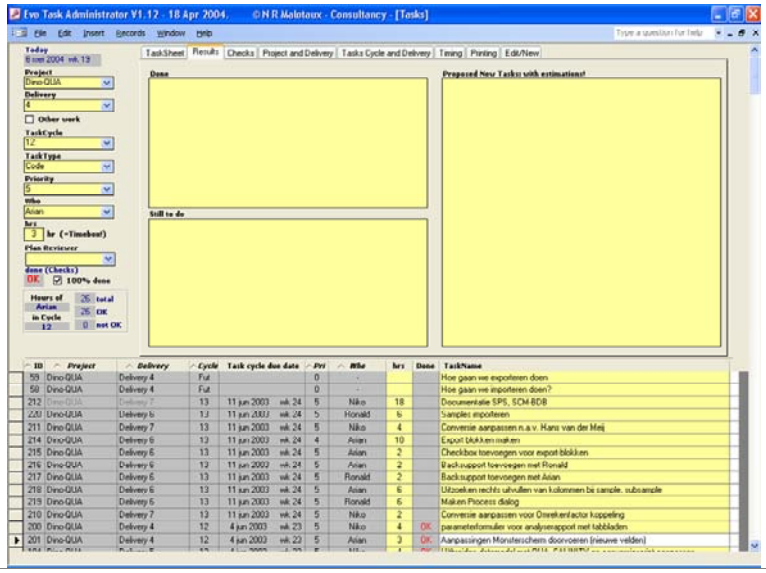
35



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

37



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

39

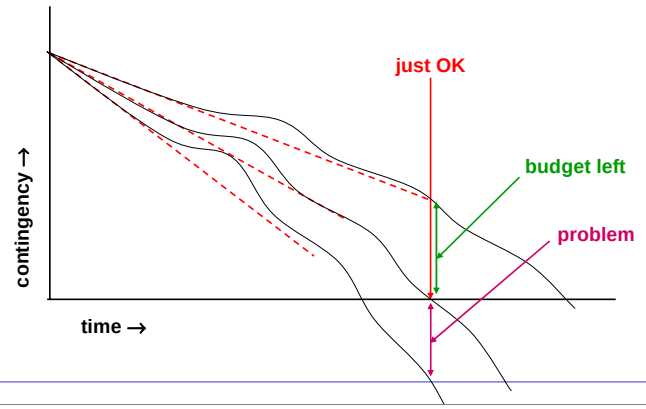
Interrupt Procedure "We shall work only on planned Tasks"

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.**

40

Contingency



41

TimeBox - taking Time seriously

- **A TimeBox is the maximum time available for an amount of work**
- **When the time is up, the work should be completely done: there is no more time !**
- **Because people tend to do more than necessary**
  (especially if the requirements of the work are unclear)
  - **Check halfway whether you're going to succeed on time**
  - **If not: what can you do less, without doing too little**
  - **Define the requirements of the work well**
  - **If the TimeBox is unrealistic: take the consequences (pdc*Act*) immediately**
    (if the work suddenly proves to need much more time, is it still worth the investment?)
- **If you really cannot succeed within the TimeBox (e.g. a Task):**
  - **Check what you did**
  - **Check what you didn't do**
  - **Check what still has to be done**
  - **Define new Tasks with estimations (TimeBoxes !)**
  - **Stop this Task to allow for finishing the other committed Tasks**
    (don't let other Tasks *randomly* be left undone)

42

Evo elements

- Task Cycle
- Delivery Cycle
- Time Line
- Requirements engineering, using Planguage
- Priorities
- Focus
- Active synchronisation
- Work is always done

- Integrated Planning, Requirements and Risk management

→ Result Management

43

Links

- **http://www.gilb.com**
  Tom Gilb's website: Evo guru
- **http://www.malotaux.nl/nrm/English**
  Niels' activities: Evo evangelist
- **http://www.malotaux.nl/nrm/Evo**
  Evo pages
- **http://www.malotaux.nl/nrm/pdf/MxEvo.pdf**
  Evolutionary Project Management Methods
  (issues and 2001 experience)
- **http://www.malotaux.nl/nrm/pdf/Booklet2.pdf**
  How Quality is Assured by Evolutionary Methods
  (more recent practical implementation experience)
- **http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf**
  Optimizing the Contribution of Testing to Project Success
- **http://www.malotaux.nl/nrm/Evo/ETAF.htm**
  Download the Evo Task Administrator (ETA) tool
  (expects MSAccess2000~2003)

44

Controlling Risk

by Design

Evo is Result Management

Niels Malotaux

N R Malotaux
Consultancy

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

45