May 2006

Controlling Project 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

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 taught and coached some 40 projects at 12+ different organizations in the Netherlands, Belgium, Ireland, India and the USA. He is a frequent speaker at conferences and published three booklets on the subject.

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
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/nrm/English

Controlling Project Risk *by Design*

Niels Malotaux

N R Malotaux Consultancy

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

1

Risk Definition

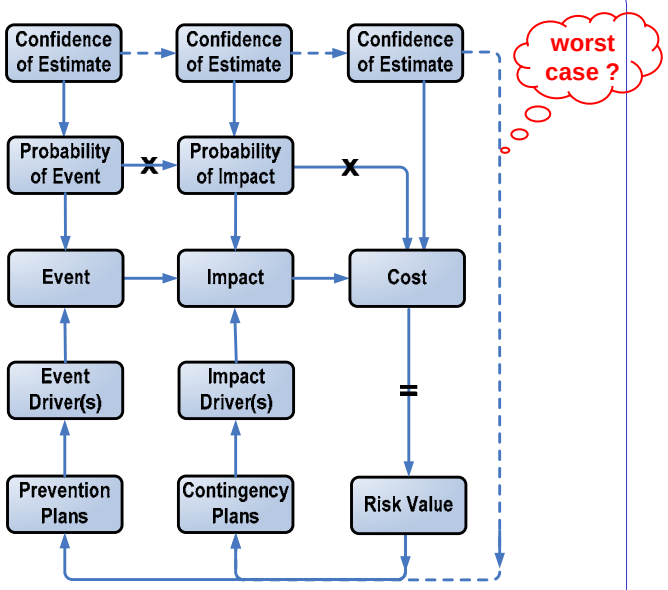
An uncertain event or condition that,
if it occurs,
has a ~~positive~~ or negative effect
on a project's objectives

(PMBOK)

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

2

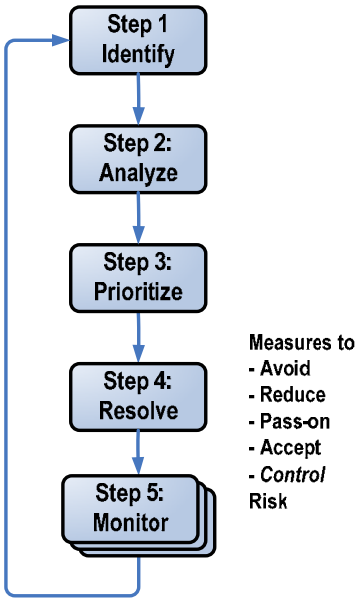
Risk Model



$V_R = P_e * P_i * C$

3

Risk Management



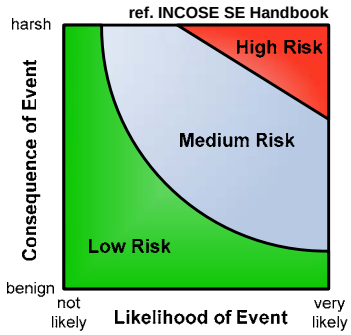
Measures to
- Avoid
- Reduce
- Pass-on
- Accept
- *Control*

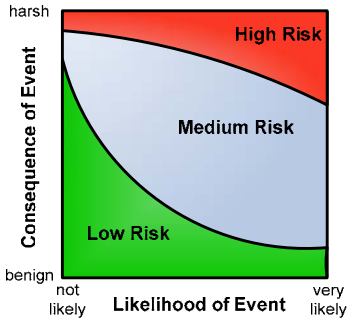
Risk

4

Risk

Prioritize Risk?





Risk Priority = Likelihood x Consequence ??

5

Mathematical Risk Management can be risky

ref
Carlo Rafele,
David Hillson,
Sabrina Grimaldi

		From RBS (Program constraints)							WPs evaluation	
		Staff	Budget	Facilities	Type of contract	Restrictions / Dependencies	Customer	Subcontractors	ER	WPs order
From WBS	Develop Project Charter			I=3, p=2: R=6	I=6, p=5: R=30	I=7, p=7: R=49	I=7, p=5: R=35		120	1
	Define scope	I=8, p=6: R=48	I=7, p=4: R=28					I=4, p=4: R=16	92	3
	Develop Resource Plan	I=7, p=5: R=35				I=4, p=3: R=12			47	5
	Develop Communication Plan	I=5, p=3: R=15				I=3, p=2: R=6			21	9
	Develop Risk Plan	I=7, p=5: R=35							35	7
	Develop Change Control Plan								0	6
	Develop Quality Plan					I=4, p=3: R=12			12	10
	Develop Purchase Plan								0	12
	Develop Cost Plan		I=8, p=4: R=32						32	8
	Develop Organization Plan								0	12
	Develop Project Schedule								0	12
	Conduct Kickoff meeting	I=7, p=5: R=35			I=3, p=2: R=6				41	6
	Weekly Status Meeting								0	12
	Monthly Tactical Meeting								0	12
	Project Closing meeting						I=3, p=2: R=6		6	11
	Standards	I=8, p=6: R=48					I=7, p=5: R=35	I=4, p=4: R=16	99	2
	Program Office					I=5, p=3: R=15	I=8, p=6: R=48		63	4
Risky events evaluation	ER	219	60	6	30	94	124	32		
	Risky events order	1	4	7	5	3	2	6		

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

6

Risk literature

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

7

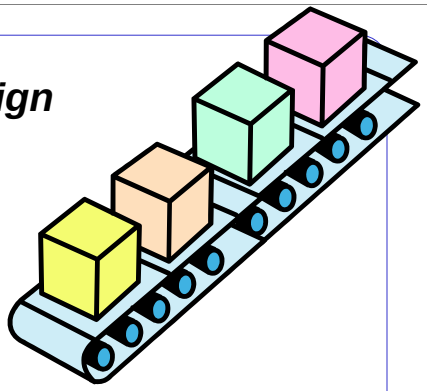
Murphy

- **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**

8

Risk

Controlling Risk *by design*



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

however

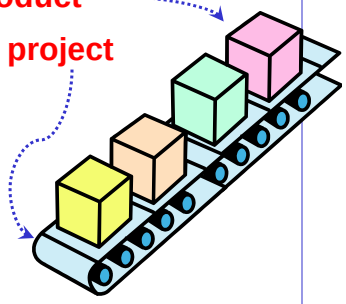
- **A lot is always the same:**
  - **Every project is done by people**
  - **No project is very much unique**
  - **There are many similarities (*known* risks)**
  - **So, a lot is predictable**
  - **We know the Requirements change (don't know *which*)**
  - **Engineers control risks *by design* (= *engineering*)**

9

Many *known* risks are hardly risks

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

$$V_{Risk} = P_{event} * P_{impact} * C \qquad P_{event} = 1 \qquad P_{impact} \to 0$$



- **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**

10

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**

11

Defect and Risk

If a Defect is:

a cause of a problem experienced by a stakeholder of the system, ultimately by the customer

- **Being late is a defect**
- **Being over budget is a defect**
- **Not satisfying the Goal is a defect**

Risk is:

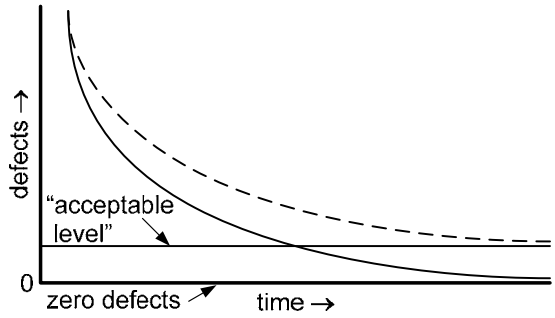
an event that *may* cause a defect

12

Risk

Is *defect free* possible?

- **Zero Defects is an asymptote**



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

13

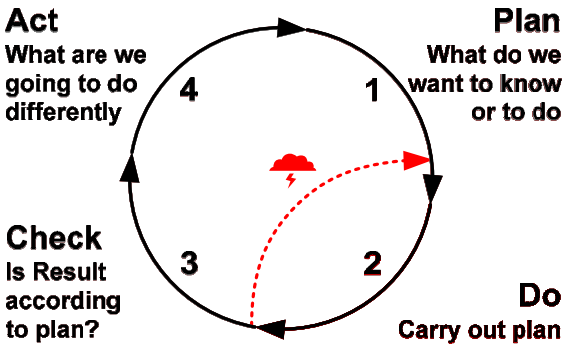
Attitude

- **As long as we think defect free software is impossible, we will keep producing defects**

- **From now on, we don't want to cause problems 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'll be highly surprised when there proves to be a problem after all**
- **We do what we can to improve (continuous PDCA)**

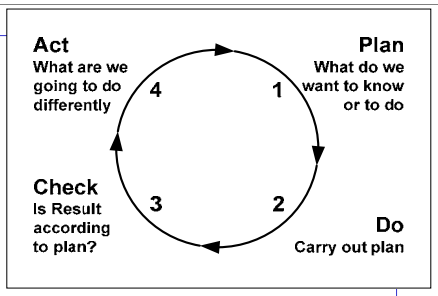
14

The PDCA cycle



15

Evo

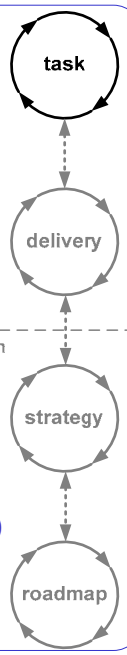


- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle
  actively, deliberately, rapidly and frequently,
  for *Product*, *Project* and *Process*, based on ROI
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- Controlling risk *by design*
- Projects seriously applying Evo, routinely conclude successfully on time, or earlier

16

Risk

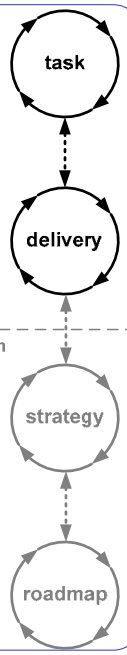
Cycles in Evo: TaskCycle



- **Weekly TaskCycle**
  - Are we ***doing*** the ***right things***, in the ***right order,*** to the right ***level of detail***
  - 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

Cycles in Evo: DeliveryCycle



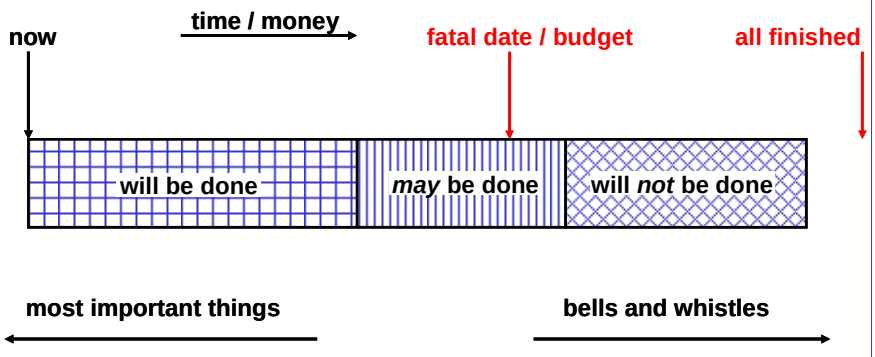
- **Weekly Task Cycle**
- **Value Delivery Cycle**
  - Are we ***delivering*** the right things, in the right order to the right level of detail
  - Optimizing requirements and checking assumptions
  - What will generate the optimum feedback
  - We only deliver to eagerly waiting stakeholders
  - Delivering the juiciest, most important stakeholder values that can be made in the least time
  - Not more than 2 weeks

18

Booklets:
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf - http://www.malotaux.nl/nrm/pdf/Booklet2.pdf
http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf - http://www.malotaux.nl/nrm/pdf/EvoRisk.pdf

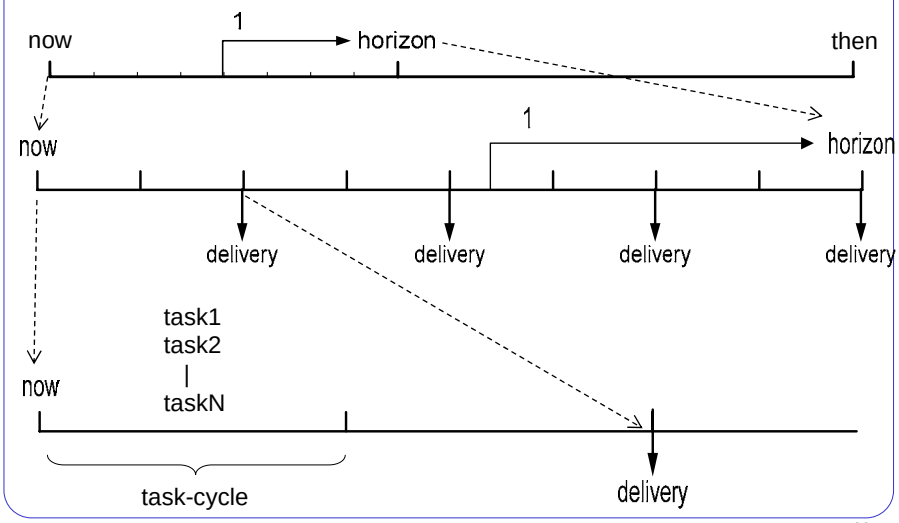
TimeLine

What the customer wants, he cannot afford



19

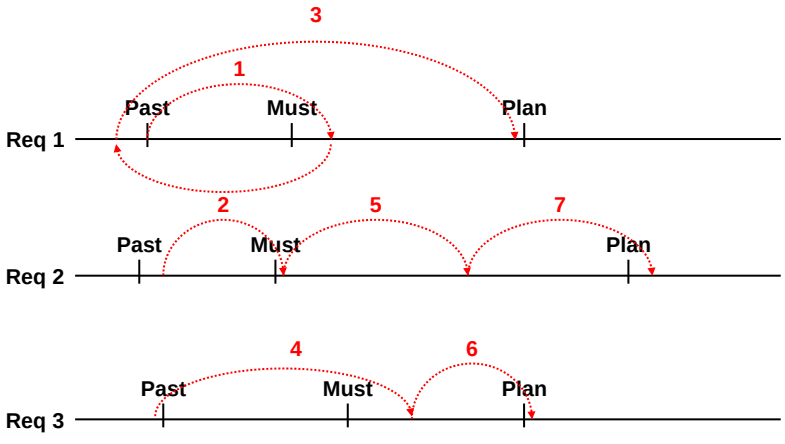
TimeLine



20

Risk

Design to Multidimensional Quality Requirements



21

Personnel Shortfalls

Boehm 1991

- **There are a certain number of people in the organization**
- **If we don't get the people we think we need, they are working on more profitable activities**
- **Using TimeLine, we inform management about the consequences**

- **This is not risk - it's choice**

22

Booklets:
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf - http://www.malotaux.nl/nrm/pdf/Booklet2.pdf
http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf - http://www.malotaux.nl/nrm/pdf/EvoRisk.pdf

Unrealistic schedules and budgets

Boehm 1991

- How can we speak about realistic schedules if the requirements will change anyway?
- If the time/cost budgets are insufficient to get a profit, we shouldn't start or continue
- If management insist on unrealistic schedules (*Check*), they may need education (*Act*), or their aim is to fail
- People can quickly learn to change from optimistic to realistic estimators and thus live up to their promises
- We continuously update the TimeLine to predict what we will get, what not and what we may get
  - Not using "Earned Value"
  - But rather "Value still to earn"

23

Developing the wrong product

Boehm 1991

- Why do we have Requirements?
- *We* don't know the real requirements
- *They* don't know the real requirements
- First develop the problem, then the solution
- Without feedback we probably are developing the wrong product
- Rapid feedback is used to optimize the requirements and check the assumptions

24

Developing the wrong user interface

Boehm 1991

- The goal is making the customer satisfied and more successful than he already was
- If the users don't become more productive we fail
- We don't want to fail
- So we quickly find out what the right user interface should be

25

Gold plating

Boehm 1991

- We do as little as possible at every step
- We specify Must and Plan values
- When we reach the Plan value, we are done
- People tend to do more than necessary, especially if it is not clear what should be done
- So we define what should be done and *what not*

26

Continuing stream of Requirements changes

Boehm 1991

- **Requirements do change because**
  - **We learn**
  - **They learn**
  - **The market changes**
- **If we would deliver according to obsoleted requirements, we don't create customer success**
- **We *know* that requirements will change, so we have to find out quickly which will change:**
- **We *provoke* requirements change as quickly as possible**

27

Problems with externally furnished components

Boehm 1991

- **If our FatalDate has come, we have no excuse**
- **We use Active Synchronization to stay on top**

28

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**

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

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**

29

Real time performance shortfalls

Boehm 1991

- **This is why we have Performance Requirements**
- **Then we use engineering techniques to make sure the system is according to the requirements**

30

Managers ignorance

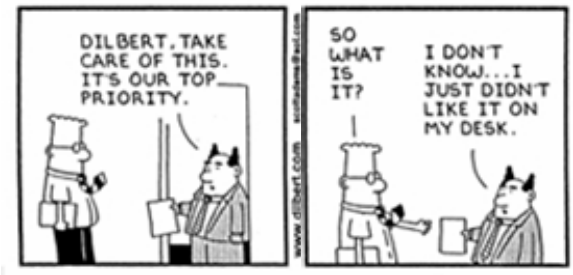
- **The product has to generate income**
- **If management impede the workers to produce the product in the most optimal way ...**
- **Management usually is not stupid**
- **But if you don't supply the right facts ...**

- **The boss *may* mess up the Result, *if* he's the owner of the company**
- **All the others have the option to leave**

31

Interrupts

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



- **In case of interrupt, use interrupt procedure**

32

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. **If 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.**

33

Worst risk

The worst risk is that you forgot an important issue

- **It’s within your control, but you didn‘t see it before it happened**
- **It’s beyond your control, but you saw it too late and/or you didn’t react appropriately**
- **The trick is to be ahead of any problem, before it occurs**
- **Don’t ostrich: actively take your head out of the sand!**
- **If anybody complains, you’re too late**

If you control 80% of the risks by design, you have a lot more time to address the remaining 20%

34

Controlling Project Risk *by Design*

Niels Malotaux

N R Malotaux Consultancy

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

35

Risk