SD Best Practices 2005 Boston

28 September 2005

Class

Routinely Assuring Project Success

You can do it too

Niels Malotaux

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

SD Best Practices - Boston 2005

Niels Malotaux

Routinely Assuring Project Success

you can do it too

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

Evolutionary Project Management
Requirements Engineering
Review & Inspection

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

Routinely Assuring Project Success

you can do it too

Niels Malotaux

N R Malotaux Consultancy

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

1

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

2

The problem

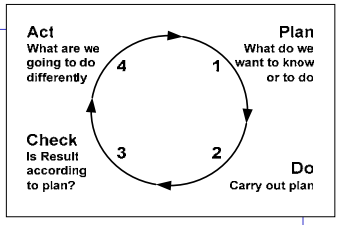
- **Many projects don't deliver the right Results**
- **Many projects deliver late**

or, more positively:

- **I want my project to be more successful**
- **In shorter time**

3

Knowledge
how to achieve the goal



- **Using 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**

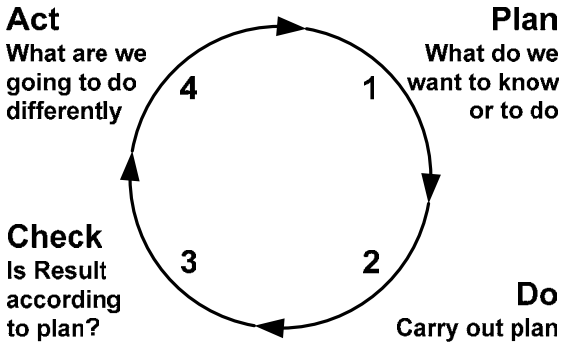
and we can

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

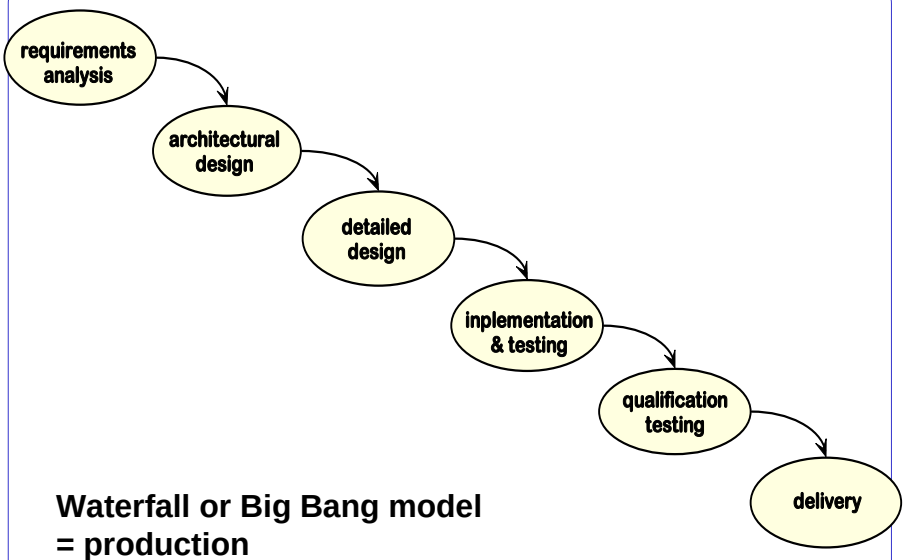
doing the right things right

4

The PDCA cycle



5

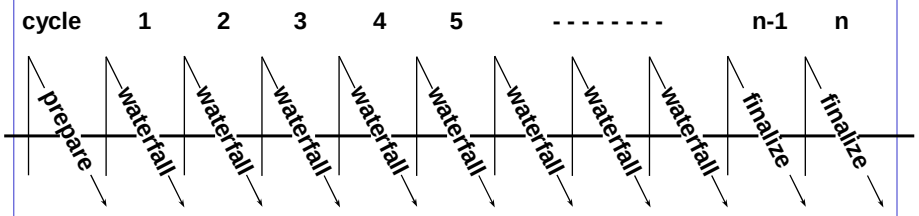


Waterfall or Big Bang model
= production
= fixed contract model (signed with blood)

6

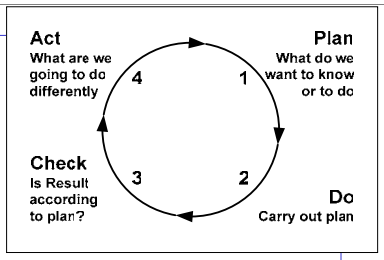
Using many waterfalls

of growing functionality



7

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**

8

Why would the product need Evo ?

- ***We*** **don't know the real requirements**
- ***They*** **don't know the real requirements**
- **Together we have to find out** (stop playing macho!)
- **What the customer wants he cannot afford**
- **Is what the customer wants what he needs?**
- **People tend to do more than necessary**
  especially if they don't know exactly what to do

If time, money, resources are limited,
we should not overrun the budgets

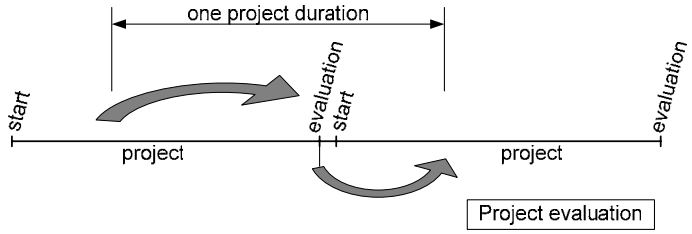
9

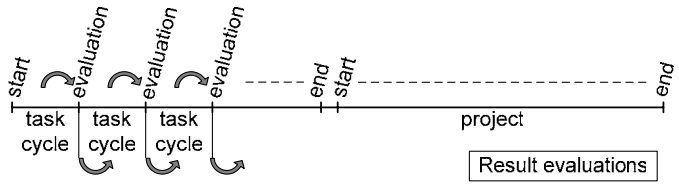
Why would the project need Evo ?

- **Are we effective?** (producing Results)
- **Are we efficient?** (optimally using the available time)
- **Are we actively learning from our mistakes?** (PDCA)
- **How do we estimate, plan and track progress?**
- **How do we handle interruptions?**
- **Did we learn from feedback per project** (project evaluation)**?**

10

Project evaluations





11

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**

12

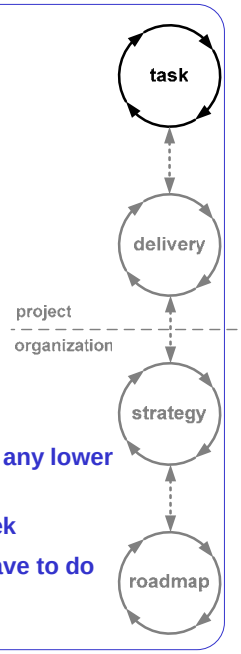
When would we *not* need Evo

- **Requirements are completely clear, nothing will change: use waterfall** (= production)
- **Requirements can be easily met with the available resources, within the available time** (Still, Evo can make it faster)
- **Everybody knows exactly what to do**
- **Customer can wait until you are ready**
- **Management doesn't know what to do with the time saved**
- **No Sense of Urgency**

Use Evo *only* on projects you want to succeed

13

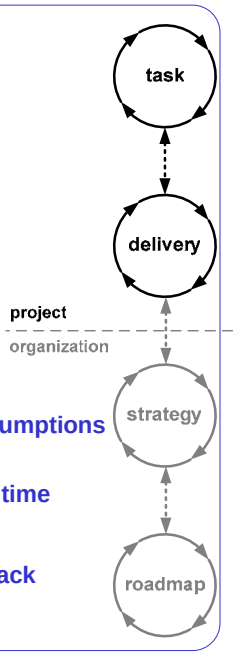
Cycles in Evo



- **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**

14

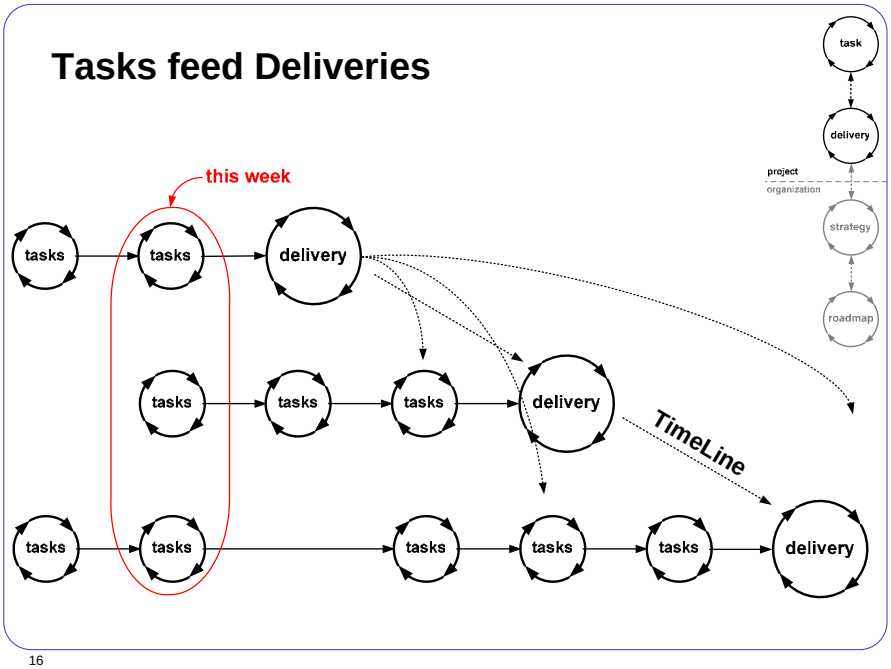
Cycles in Evo



- **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

15

Tasks feed Deliveries



16

Task selection criteria

- Most important requirements first
- Highest risks first
- Most educational or supporting first
- Actively Synchronize with others beyond your control
- Every cycle delivers a useful, *completed*, result

17

Delivery selection criteria

1. What will generate the optimum feedback
2. Delivering the juiciest, most important stakeholder values that can be made in the least time
3. What will make Stakeholders more productive *now*

- Every delivery must have a useful set of stakeholder values (features, qualities), otherwise the stakeholders get stuck
  - Delete ↔ Add
  - Copy ↔ Paste
- Every new delivery must have clear extras, otherwise the stakeholders won't keep producing feedback
- Every delivery delivers smallest clear increment, to get the most rapid and most frequent feedback
- If a delivery takes more than two weeks, it can usually be shortened: try harder

18

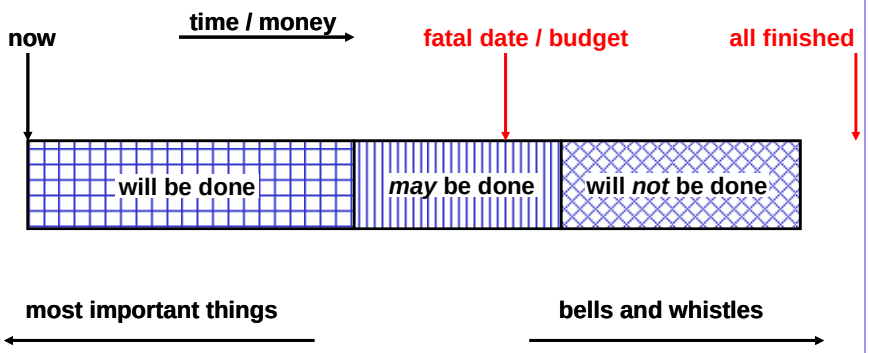
My project is different

- **On *every* project somebody will claim:**
  "Nice story, but *my* project is different.
  It cannot be cut into two-week deliveries"
- **On *every* project, it takes less than an hour to define the first short deliveries**
- **This is one of the less easy issues of Evo.**
  We must learn to turn a switch

19

TimeLine

What the customer wants, he cannot afford



20

Two options

1. **Conventional option**
   At the fatal day we'll tell we didn't succeed
2. **Evo option**
   We already know we won't succeed, so we can tell it now, then together we can decide what to do

Which option do you want?

Quality On Time is also being honest as soon as you can

The challenge is to find out as soon as you can

21

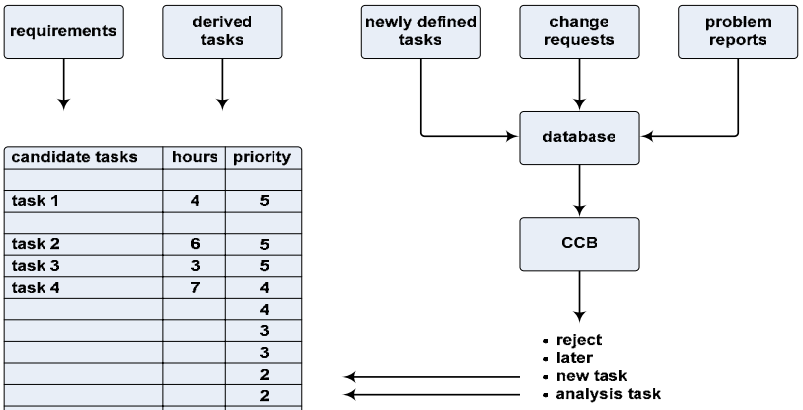
Priorities

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

Let it be the most important 80%

22

Anything that must be done goes through the *Candidate Task* Mechanism



candidate tasks	hours	priority
task 1	4	5
task 2	6	5
task 3	3	5
task 4	7	4
		4
		3
		3
		2
		2
		1
		0
task n	23	0

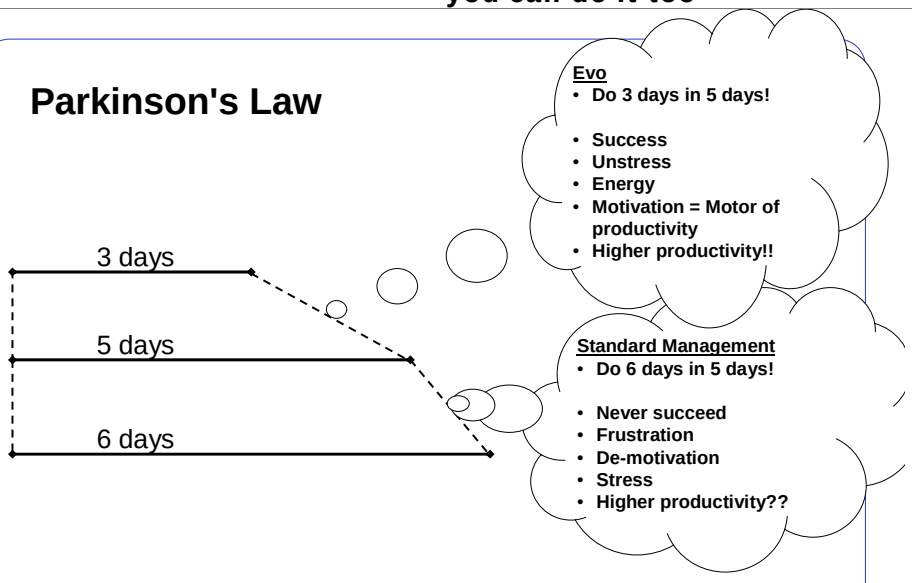
23

How to start with tasks

- **Take the requirements, architecture and design**
- **Make a list of things to do**
- **Split in tasks of max ~6 hrs max (estimate TimeBox)**
- **Put on Candidate Tasks List**
- **Prioritize the tasks on the Candidate Tasks List**
- **Select 26 hrs of tasks from top of the list**
- **Agree and commit to work packages (100% done!!!)**
- **Do the work**
- **Learn**

24

Parkinson's Law



"Work expands to fill the time available"

25

What to plan and what not to plan

- **We plan tasks that don't get done unless planned**
- **We do not plan tasks that don't have to be planned to get done.** **Such planning costs more than it saves**
- **Account for these tasks as "unplannable tasks"**
- **Default we allocate 2/3 for plannable tasks and 1/3 for unplannable tasks**
- **Plan *all* plannable hours**

26

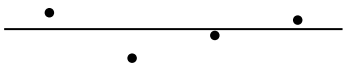
Types of Tasks

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**

27

Beware of longer Tasks

- **Beware of Tasks longer than about 6 hrs**
- **Estimation is never exact**
- **If you have 4 or more Tasks in a week, the variation in the Tasks estimations should average**



Only the average should be OK: *Result* is all that counts

- **You have only 2/3 plannable time, so you can cheat a bit to get all the committed tasks done**

28

TimeBox - taking Time seriously

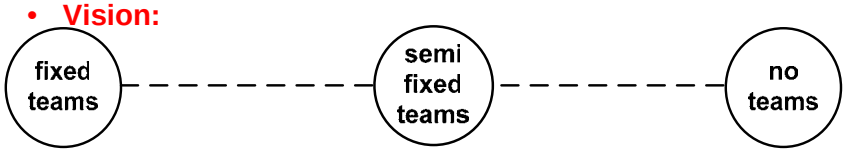
- A TimeBox is the maximum time available for a Task
- When the time is up, the Task should be completely done: there is no more time !
- Because people tend to do more than necessary
  (especially if the requirements of the Task are unclear)
  - Check halfway whether you're going to succeed on time
  - If not: what can you do less, without doing too little
  - Define the requirements of the Task well
  - If the TimeBox is unrealistic: take the consequences (pdc*Act*) immediately
    (if a Task suddenly proves to need much more time, is it still worth the investment?)
- If you really cannot succeed within the TimeBox:
  - 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)

29

We work on more projects

- Define how many hours available for this project
- Deliver these hours
- In case of interrupt, use interrupt procedure
- Boss comes in: "Can you paint my fence?"
- What do you do?

- Vision:



30

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.**

31

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**

32

Meetings

- **Pitfalls**
  - One to ones, others waiting
    - status round ("round of excuses")
    - filling the excuses-toolbox
    - detailed discussion
  - Discussing less important subjects for too long
  - Not reaching set goals
- **Meetings are very costly** (ROI?)
  - Try the meeting-meter
    number of people * average hourly rate: show $$ ticking

33

Taking Status out of the meeting: 1 to 1's

- **Team member with Project Management: 1 to 1**
  - Status of Task: If task not done, coach
    - Timebox used? → complexity problem
    - Timebox not used? → time management problem
  - New tasks: what is most important to do
  - Estimate new tasks: timeboxes
  - Commitment:
    - Do you agree this is the most important
    - Will you really finish these tasks completely?
    - Sensing not real commitment: do something!
  - Decision: new task list for next cycle
  - Can also talk about specific details of the work
- **Evo day: First 1 to 1's, then Team meeting**

34

Problem with 1 to 1's

- Took about 1 hr per person
- Homework:
  - What to do next
  - Estimations
  - How much time available
- Result: now 20 min per person
  - How come?

35

Result: 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

36

Experiment!

- **Every meeting with more than one person uses a projector**
- **Why?**
- **1-to-1's should be held on neutral ground**
- **Why?**
- **Don't believe me. Try it out yourself. *Experiment!***
- **But then ...**

37

Evolutionary start pattern

- **Evo day**
  - **Explanation of the Evo approach**
  - **Organizing the work of the coming week**
  - **Goal: at the end of the day, people of the team know what they are going to work on and why**
- **Next day**
  - **Defining Tasks by the remaining team members (larger team)**
- **Weekly Evo day**
  - **Execution of the 3-step procedure**

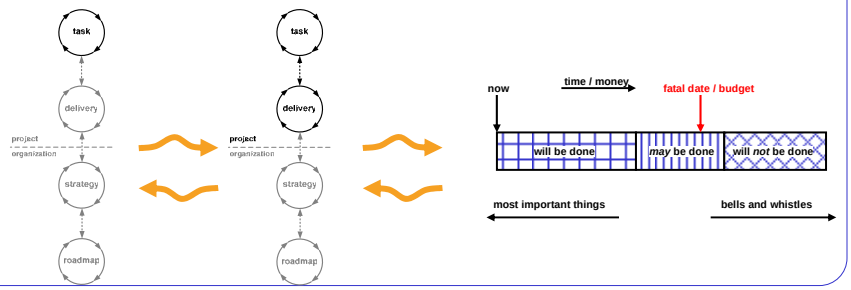
38

Evolutionary introduction pattern

1. **Introducing *Tasks***
   How to organize the work
2. **Introducing *Deliveries***
   Focusing on Results
3. **Introducing *TimeLine***
   The *design* of the project

Short term view

Project view



39

What to do with the time gained?

- **If our original requirements are done in 70% of the time, what do we do with the 30% gained?**
  - **Choosing the next project**
  - **Continuing evolutionarily adding extras**
  - **Beware of Parkinson's Law!**
  - **Extending the horizon of the project to assure success**

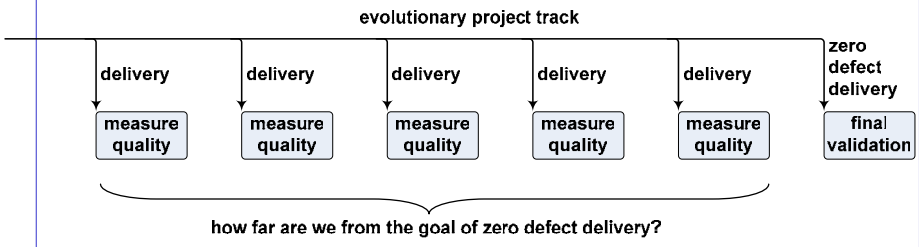
40

Extending the project horizon to success

- **Many projects end at: Hurray, it works!**
- **If customer success is paying our salaries, shouldn't we make sure the success is *going to happen***
- **Now a lot of quality requirements suddenly make sense:**
  - **User friendliness - Usability**
  - **Intuitiveness - Learnability**
  - **Installability**
  - **Serviceability - Maintainability**

41

Testing in Evo

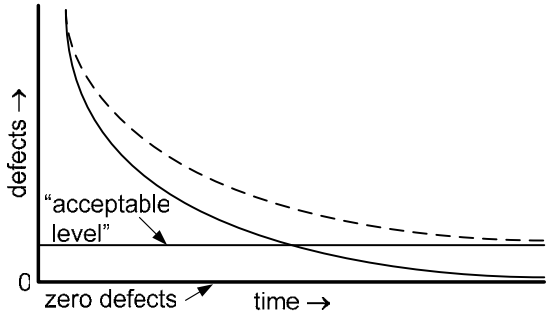


- **Final validation shouldn't find any problems**
- **Earlier verifications mirror quality level to developers: how far from goal and what still to learn**
- **Evo has *no debugging phase!***

42

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**

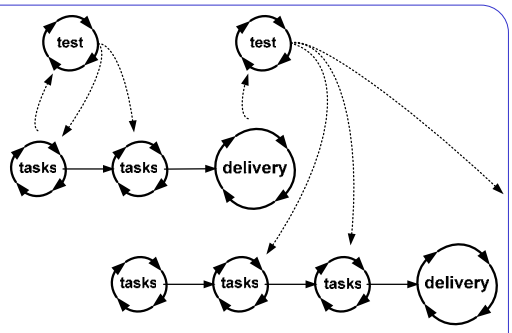
43

Attitude

- **As long as we think defect free software is impossible, we will keep producing defects**
- **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 defect after all**
- **We do what we can to improve (continuous PDCA)**

44

Evo cycles for Testing



- Testers organize their work in weekly TaskCycles
- DeliveryCycle is the Test-Feedback cycle
- Testers use their own TimeLine, synchronized with the developers TimeLine
- Testers conclude their work in sync with developers
- Testers check work in progress *even before* it is finished

45

Case 7: A "failure"

- Seasoned project manager: "Good idea, but..."
- No emphasis on TimeBoxing
- Didn't try to understand Delivery and TimeLine concepts
- Many "hero's" in the team
  - I can do whatever I want. I know so much, they won't fire me.
- No *Sense of Urgency* both in team and from management
  - Management by fear
  - Management asks different things every week
  - Management asks impossible results

If you don't apply Evo, Evo does not fail, the project does

46

Case 9: US company

- **Started with 15 people of a 40 people project** (don't over-eat)
- **We *designed* the Evolutionary Introduction of Evo**
- **Now the whole team is routinely working the Evo way**
- **Including 8 people in India**
- **Didn't miss a milestone since**
  (Average time overrun before Evo was 20%)
- **They still hardly believe this is possible**

Evo works with larger and distributed projects

47

Case 10: Managers

- **Managers asked**
  "Can I use this for my own busy schedule?"
  - Write down what you have to do
  - Add effort hours
  - List in order of priority
  - Check how much time available this week
  - Draw line at 2/3 of the available time
  - Decide what to do *and what not to do*
- **Managers Report:**
  "This made me 40% more productive!"

48

Case 6:

- **Project Manager needed two days of coaching for basic Evo**
- **Two more times with 3 weeks intervals**
- **Project well within the expected time**
- **Product manager absolutely happy with results**
- **This project produced a very useful Evo metric:**

The size of the smile of the customer

49

Magic words

- **Focus**
- **Priority**
- **Synchronize**
- **Why**
- **Dates are sacred**
- **Done**
- **Bug, debug**
- **Discipline**

50

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)

51

Can you afford not to use Evo?

Niels Malotaux

N R Malotaux
Consultancy

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

52

20050818