PNSQC 2005

10 October 2005

Workshop

Evolutionary Project Management Methods

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Niels Malotaux

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

PNSQC 2005

Niels Malotaux

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

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

Slash Project Time with Evolutionary Methods

Delivering Better Results Faster

Niels Malotaux

N R Malotaux Consultancy

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

1

Niels Malotaux

Project Coach

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

2

Who are you?

- Industry
- Types of product
- Types of work

3

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

Ambition

- **What problem would you like to solve most?**
  (In your work)

4

Immediately OK?

- **Do your projects regularly deliver immediately 100% good results?**
- **Why not?**
- **Is this normal?**
- **What can we do about it?**

5

Agreed time

- **Do your projects regularly deliver within the time agreed?**
- **Why not?**
- **Is this normal?**
- **What can we do about it?**

6

Intro

Problems with projects

- It is not immediately right
- It takes too much time
- It costs more than necessary

7

Do you mind?

- Does anybody mind
  - projects being late
  - costing too much … ?

8

Can you afford it?

- Can you afford
  - projects being late
  - costing too much ….. ?

9

Goal

- The right product
- The right quality

} The right results

- Within the time and budget agreed
- Pleasantly for everyone involved

Quality On Time

What the customer needs *when* he needs it
to make *more* money than *we* need

10

Higher productivity of the User

- All the functions we are providing *are already there*
- All we are adding is *better performance*
- Performance increasing *productivity*, adding *value*, focused on *success* for our *customers*

We should constantly be aware: "Are we *Creating Value?*"

11

Adding performance

	V8.5	V9.0	
• **Usability.Productivity:**			
• Time to set up a typical specified report	65	20	min
• Time to generate a survey	120	0.25	min
• Time to grant access to report, distribute logins to end-users	80	5	min
• **Usability.Intuitiveness:**	265	25.25	min
• Time for medium experienced programmer to find out how to do ...	15	5	min
• **Capacity.RuntimeConcurrency**			
• Max number of concurrent users, click-rate 20 sec, response time < 0.5 sec	250	6000	users

after FIRM / Gilb 2005

12

Intro

Stakeholders and Requirements

- A Stakeholder is anybody with a stake in what we are working on
- Customer, user, ... up to ourselves
- Every project has about 30 Stakeholders
- The set of Stakeholders doesn't change much

- *Requirements* are what the Stakeholders require
- but for a project ...
- Requirements are the set of stakeholder needs that a project is planning to satisfy

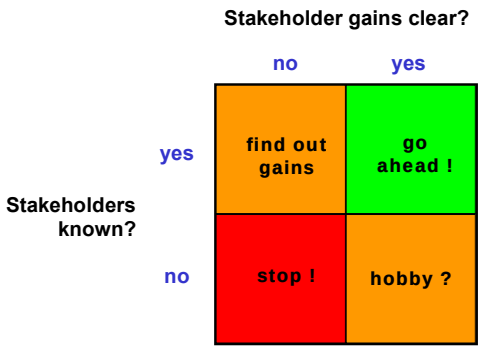
13

No Stakeholder?

- No Stakeholder: no requirements
- No requirements: nothing to do
- No requirements: nothing to test
- If you find a requirement without a Stakeholder:
  - Either the requirement isn't a requirement
  - Or, you haven't determined the Stakeholder yet
- If you don't know the Stakeholder:
  - Who's going to pay you for your work?
  - How do you know that you are doing the right thing?
  - When are you ready?

14

Make sure to know *what* to do *for whom*



15

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

No cure - no pay

- If whatever you do doesn't yield a positive ROI, you shouldn't get paid
- So you better shouldn't do it

- Who dares working on a no-cure no-pay basis?

16

Stakeholder exercise

- Every project has about 30 Stakeholders
- The set of Stakeholders doesn't change much

1. Make a list of Stakeholders
2. Select which Stakeholders you should serve in your current project
3. What is the most important requirement for each Stakeholder?
4. Are you serving all these Stakeholders?

17

Quality On Time

Niels Malotaux

N R Malotaux Consultancy

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

1

Quality On Time

- What is Quality?
- What is On Time?

2

Quality

- I know it when I see it …?

- Should be *measurable*
- Should be *predictable*

3

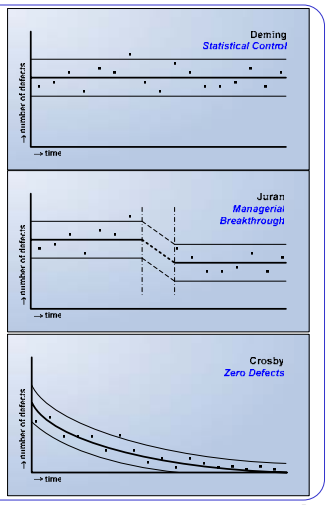
Booklets:
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf
http://www.malotaux.nl/nrm/pdf/Booklet2.pdf
http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf

Product quality

- **Shewhart** - Economic Control of Quality 1930
- **Deming** - Japan 1949, Out of the crisis 1986
- **Juran** - Japan 1950, Quality handbook 1951
- **Crosby** - Zero Defects 1961, Quality is Free 1979

4

Deming - Juran - Crosby



5

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

6

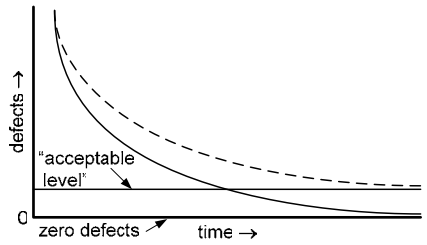
Quality on Time

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Is defect free software possible?

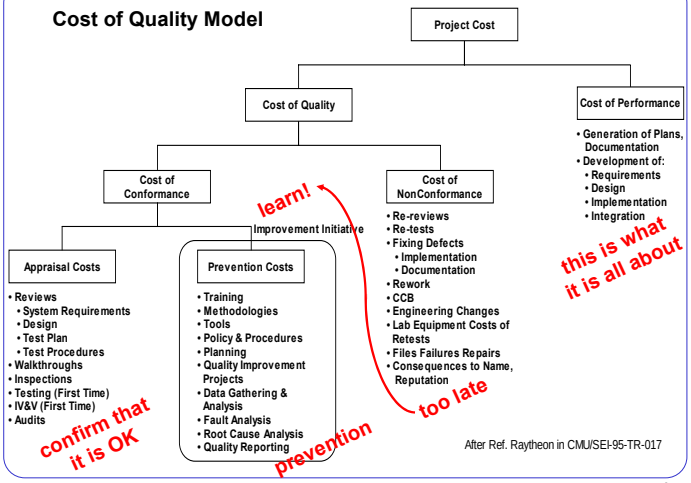
- **Zero Defects is an asymptote**



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

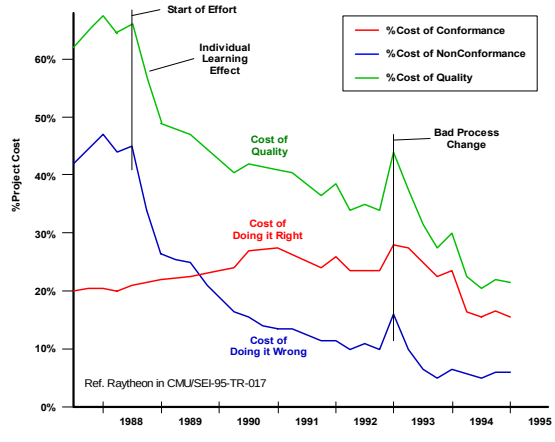
7

Cost of Quality Model



8

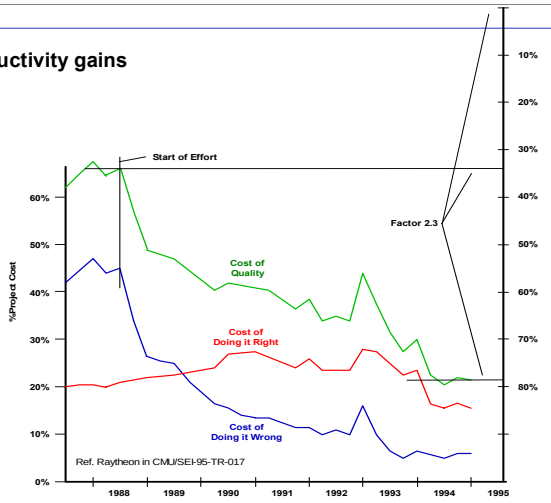
Cost of Quality



9

Quality on Time

Productivity gains



10

On Time

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

- Compromise between what is *needed* and what is *possible*

11

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

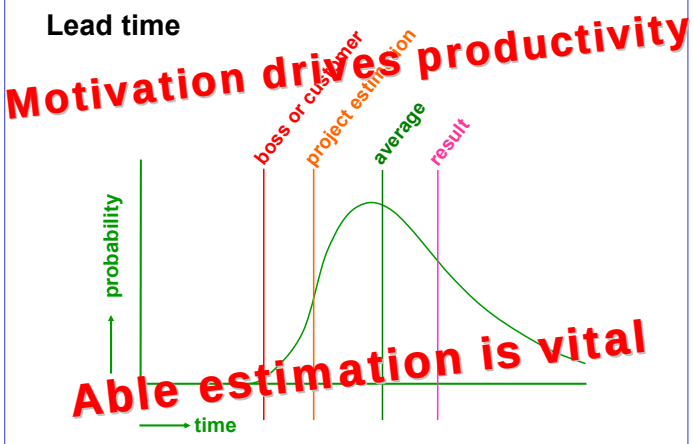
12

Quality on Time

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Lead time



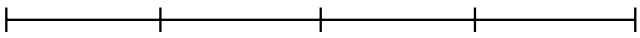
13

Realistic estimation in 3 weeks

- In 3 weeks people can change estimation from optimistic to realistic
- 1st week 40%, 2nd 80%, 3rd week 100%
- Commitment
- Use 'the mirror'
  - Commitment: they see themselves in the mirror
  - No commitment: they see you

14

4 week project

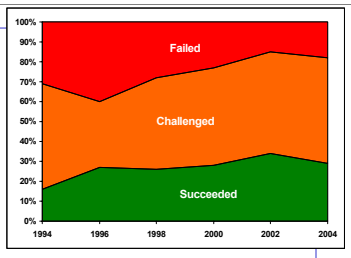


25%	25%	25%	25%
10%		90%	
10%	10%		80%
10%	10%	10%	70%

15

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

What's wrong with projects?



IT projects

		1994	1996	1998	2000	2002	2004
• Succeeded	delivered on time, on budget, with required features and functions	16%	27%	26%	28%	34%	29%
• Challenged	late, over budget and/or with less than the required features and functions	53%	33%	46%	49%	51%	53%
• Failed	cancelled prior to completion or delivered and never used	31%	40%	28%	23%	15%	18%

2/3 still fails on Quality On Time

However: "Succeeded" projects actually were late from the beginning:
Management told that they multiplied "best guess" by 2.5

Standish Group International

16

Top 5 success factors:

1. Executive Support
2. User Involvement
3. Experienced Project Manager
4. Clear Business Objectives
5. Minimized Scope

Standish Group International

17

Deming

- Quality comes not from inspection, but from improvement of the production process
- Inspection (testing) does not improve quality, nor guarantee quality
- Inspection is too late
- The quality, good or bad, is already in the product
- You cannot inspect quality into a product

18

Quality on Time

Is it difficult to be on time?

- **Did anyone miss a plane?**
- **What did you feel?**
- **Why did it happen?**
- **Did it happen again?**

19

Issues

Niels Malotaux

N R Malotaux Consultancy

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

1

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
  - For the good that I would I do not …

→ We must help each other (watching over the shoulder)
→ Rapid success helps

2

Intuition

- Makes you react on every situation
- Intuition is fed by experience
- It is free, we always carry it with us
- Sometimes intuition is simply wrong
- In many cases the head knows, the heart not
- Coaching is about redirecting intuition

3

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

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Communication

- 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
- E-mail communication evaporates in a few days

4

Ready in January

- Stick to your agreement
  - Can you do that?
  - Yes
  - *When* is it done?

Be as explicit as needed

5

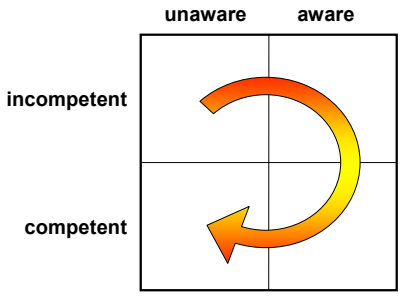
Ignore the first reaction

- If you show something is wrong
- Even if the person agrees, first you'll get:
  "Yes, but ... blah blah" or,
  "That's because ... blah blah"
- We have been trained from childhood to make excuses
- Ignore the blah blah
- Wait for the next reaction

6

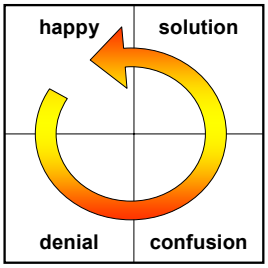
Issues

Competence square



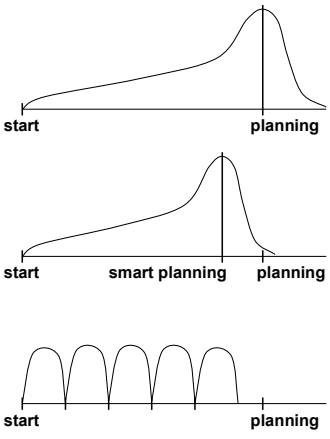
7

The problem of problem denial



8

Development cycles

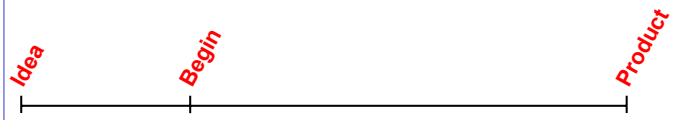


9

Issues

The order of things

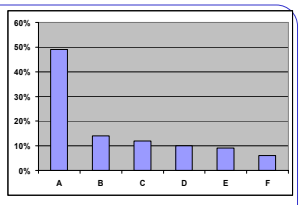
- Known and clear issues
- New / unknown / unclear issues



- Known …
- Clear …
- Unknown …
- Unclear …

10

The Pareto principle (20 - 80 rule)



A collection of problems always can be divided into a small number of large problems and a large number of smaller problems

- The *vital few* are dealt with individually
- The *useful many* are dealt with as a group

Juran, 1960

11

The Requirements Paradox

- **Requirements must be stable**
- **Requirements always change**

→ **Use a process that can cope with the requirements paradox**

You cannot foresee every change, but you can foresee change itself

12

Issues

The 2nd requirements paradox

- We don't want requirements to change
- Because requirements change now is a *known risk:*
  We must *provoke* requirements change
  as early as possible

13

Effort and Lead Time

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

- Effort variations and lead time variations have different causes
- So, treat them differently and keep them separate
  - Effort: complexity
  - Lead Time: time-management
    - (effort / lead-time ratio)

14

Project leader categories

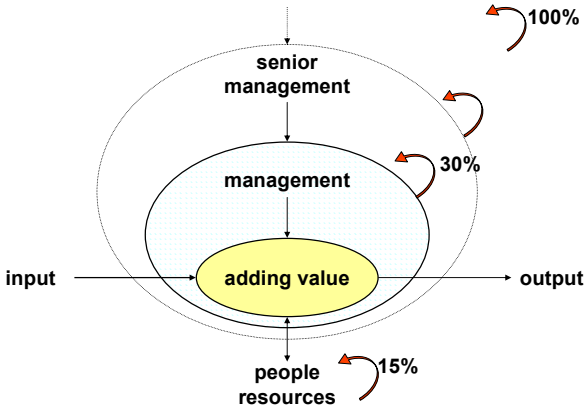
1. There is no project leader
2. He does not know, others don't know or nobody knows what it means
3. Project follower:
   Hopes that it will get on track eventually
4. Project leader: vision, strategy, scenario's, first time right, zero defects, time to market: *makes it happen*

Projects without project leader fail

15

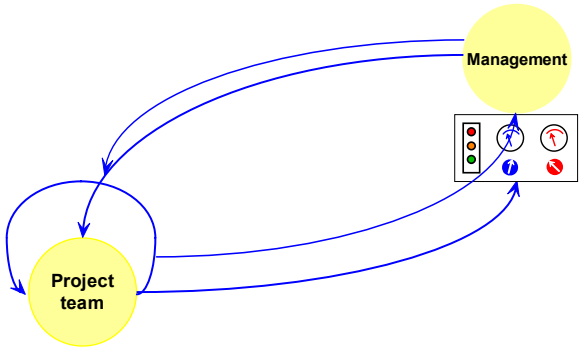
Issues

The managers task



16

Local loop principle



17

Process Improvement

- Call it Process Change until you can *prove* Improvement
- Does the Change have a positive ROI?
- Does your work have a positive ROI?
- In stead of *improving* non-value adding activities, better *eliminate* them

18

Issues

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**

19

Issues

Evo elements

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

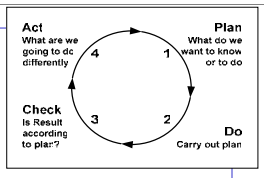
Booklets:
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf
http://www.malotaux.nl/nrm/pdf/Booklet2.pdf
http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

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

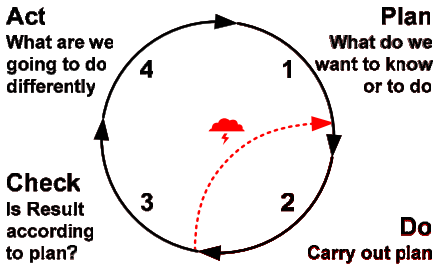
- Most quickly learn what the real requirements are
- Learn how to most effectively and efficiently realize these requirements

and we can

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

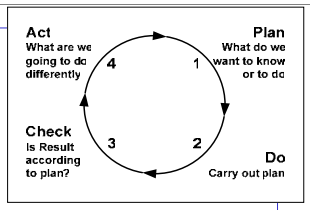
4

The PDCA cycle



5

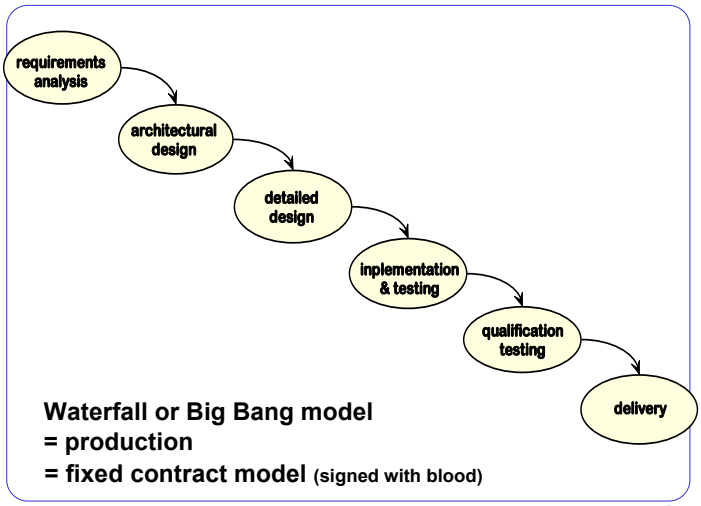
Evo

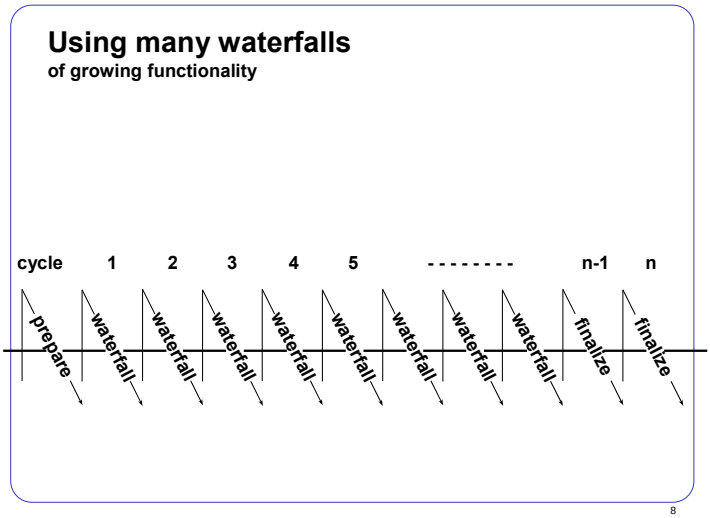


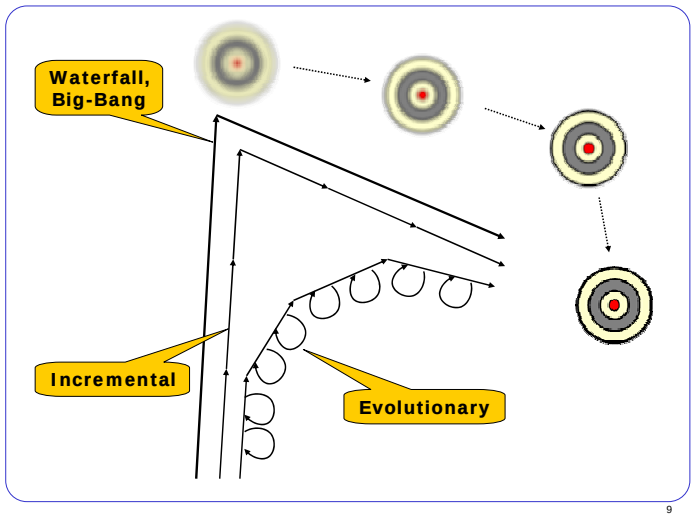
- Evo (short for Evolutionary...) uses this knowledge to the full
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for *Product*, *Project* and *Process*, based on ROI
- A desire to Learning how 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

6

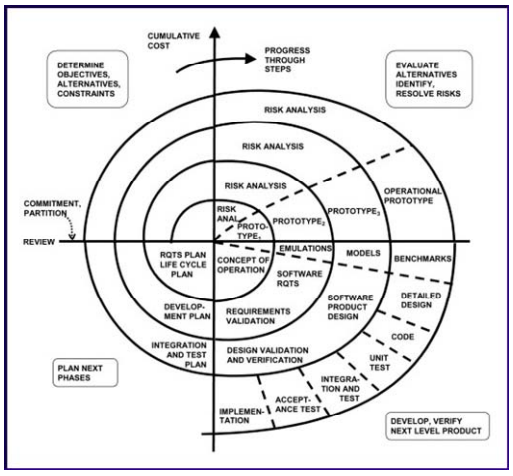
Evo elements





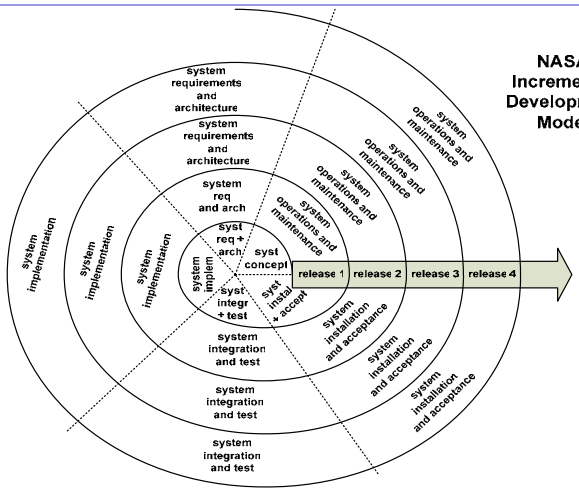


Evo elements



Spiral Process model
(Boehm 88)

10



NASA Incremental Development Model

11

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

12

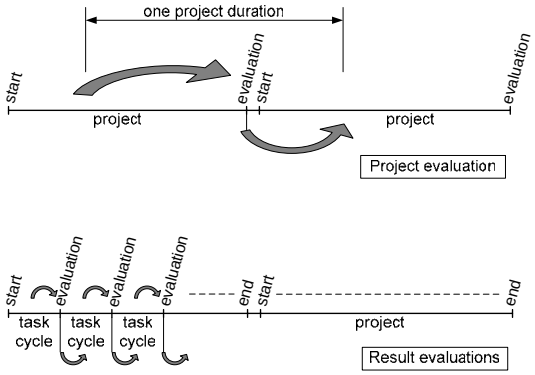
Evo elements

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)**?**

13

Project evaluations



14

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**

15

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

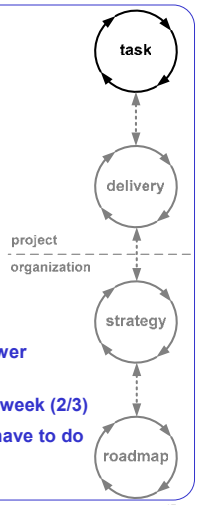
When would we *not* need Evo

- 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

16

Cycles in Evo



- Weekly Task Cycle
  - 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

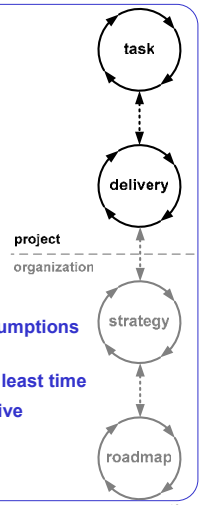
What to plan and what not to plan

- We plan any task that does not get done unless it is planned
- We do not plan any 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

18

Evo elements

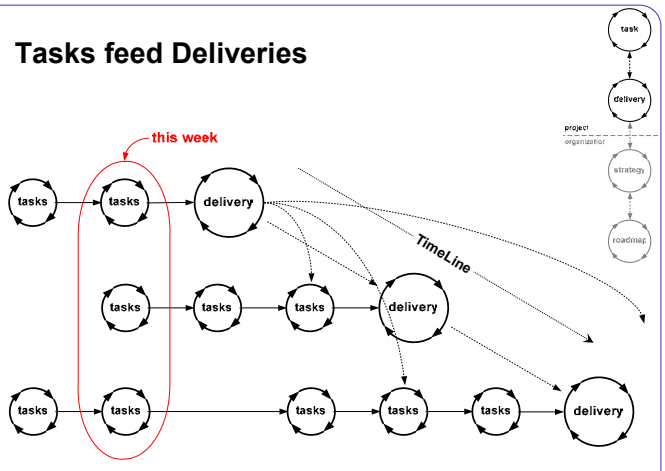
Cycles in Evo



- **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
  - Delivering the juiciest, most important stakeholder values that can be made in the least time
  - What will make Stakeholders more productive
  - What will generate the optimum feedback
  - Not more than 2 weeks

19

Tasks feed Deliveries



20

Task Cycle ↔ Delivery Cycle

Doing	Delivering
the *right things*, in the *right order* to the *right level of detail*	
Optimising	
Estimation, planning, tracking	Requirements, assumptions
Selecting	
Highest priority tasks	Most important values
≤ 1 week	≤ 2 weeks
Always done, 100% done	

21

Evo elements

Task selection criteria

- Most important requirements first
- Highest risks first
- Most educational or supporting for development first
- Actively Synchronize with other developments
- Every cycle delivers a useful, *completed*, result

22

Delivery selection criteria

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

- 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

23

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

24

Evo elements

Slash Project Time with Evolutionary Methods

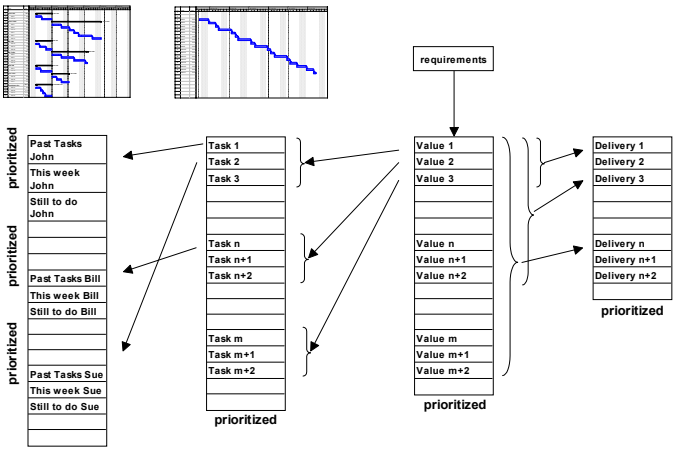
How to deliver the best possible results in the shortest possible time

TimeLine: the *design* of the *project*

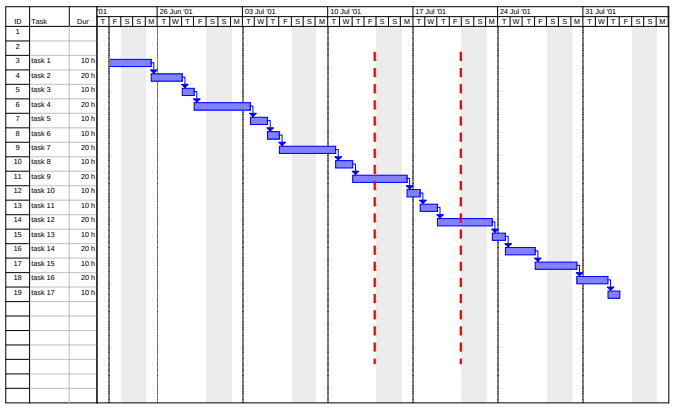
- Write down whatever you have to accomplish
- List in order of priority
- Write the same down in chapters of Results
- List chapters in order of priority
- Translate chapters into Tasks: what you have to do
- Estimate the Tasks in hours of effort
- Cut most urgent Tasks into work-Tasks (max ~6 hrs effort)
- Review the order of the list
- Ask team to add forgotten tasks and add effort estimates
- Get consensus on large variations of estimates (Delphi process)
- Add up the number of effort hours
- Divide by number of available effort hours
- This is the first estimate of the project duration
- Now we know what, at the *fatal date*, will be:
  - Surely done
  - Surely not done
  - May-be done

  And we *do* something with this knowledge
- Define Deliveries of max. two weeks at the top of the list
- Decide on first few deliveries
- Keep iterating the Timeline exercise, replanning the order of Deliveries based on the continuous change of Priorities

25

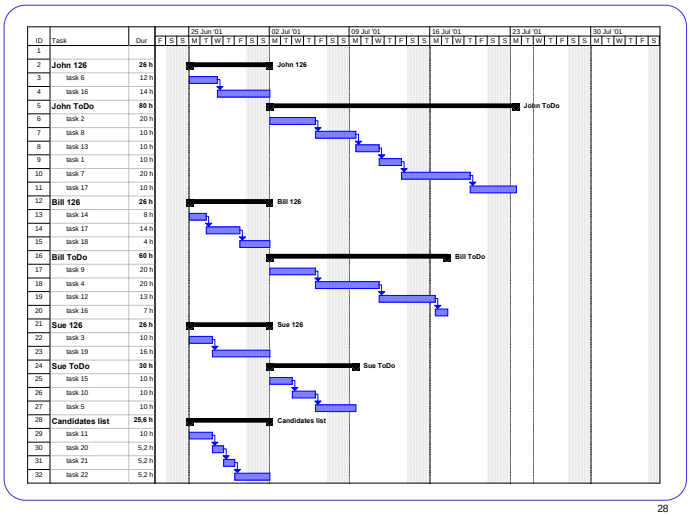


26



27

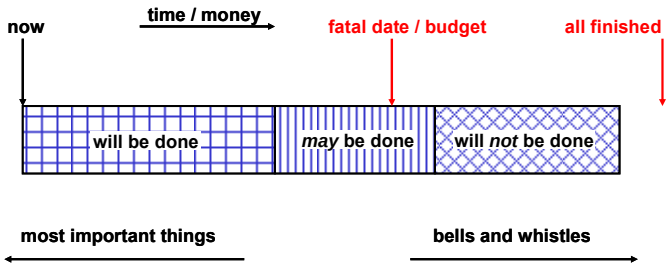
Evo elements



28

TimeLine

What the customer wants, he cannot afford



29

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

30

Evo elements

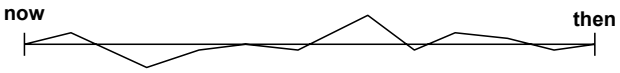
Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Tasks - Deliveries - Projects

Tasks - Deliveries - Projects
actually are similar, except for
the time and complexity scales

- At the end there is a defined Result, 100% done
- The journey to the Result should be *designed*



31

Tasks			Priority	Effort (hr)	Delivery
Onduidelijkheden in eisen en ontwerp oplossen	**ANALYSIS TASKS**				
	Nummers in formulieren en rapporten		5		
	Hoe omgaan met personen en instanties?		5		
	Hoe gaat het importeren van bestanden werken?		4		
	Hoe gaat het exporteren van bestanden werken?				
	Uitzoeken nodige functionaliteit/inspanning voor berekende parameters		5		
	Analyseren gegevensmodel oude meetnetten.		4		
Afronden Raamwerk2			5		**1**
Overzetten oude gegevensmodel naar nieuwe gegevensmodel					
	Hernoemen tabellen, constraints, enz.		5		
	Toevoegen van DBK velden in alle niet REF tabellen		5		
	Creeren 'rebuild' scripts		5		
Overzetten oude Software naar nieuwe Software			4		**2**
	Hernoemen formulieren en rapporten		4		
	Bijwerken van alle tabelreferenties naar de nieuwe tabellen		4		
	Toevoegen van DBK velden verwerken in alle PersistanceControllers		4		
	Maken plaatje voor hoofdscherm		1		
Nieuwe uitbreidingen aan gegevensmodel toevoegen volgens GDF			4		
Conversie					
	Conversie schrijven voor kwaliteitsdatabasegegevens		3		**2**
	Conversie schrijven om oude gegevens over te kunnen zetten naar nieuw		2		**3**
	Uitvoeren kwaliteitsconversie (meerdere malen)		3		**2**
	Uitvoeren van conversie (meerdere malen)		2		**3**
Beheersformulieren realiseren					**3/4**
	Afmaken Monsterscherm		2		
	Afmaken Indicatietestenscherm		2		

32

Management Questions on TimeLine

- **Will the Result be On Time?**
- **Show me !**
  - Can immediately draw TimeLine
  - Deliveries planned
  - Horizon defined
  - What will be done, not done, may-be done
  - Prompt explanation in case of discrepancies

33

Evo elements

Slash Project Time with Evolutionary Methods

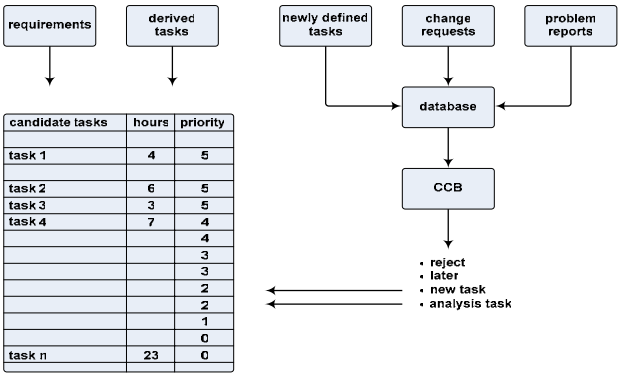
How to deliver the best possible results in the shortest possible time

To-do lists

- Are you using to-do lists? → EXERCISE
- Do you add effort estimates?
- Do you check how much time you have available?
- Do you check what you can do and what you cannot do?
- Do you take the consequence?
- Evo:
  - Because we are short of time, we better use the limited available time as best as possible
  - We don't try to do better than possible
  - To make sure we do the best possible, we have to *choose* what to do in the limited available time and don't just let it happen *randomly*
  - We don't try to do as much as possible. We rather try to do as little as possible (but not less)

34

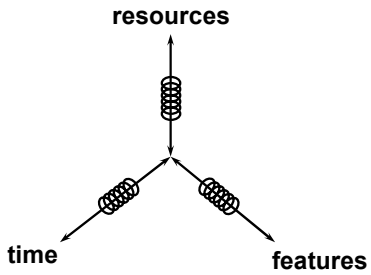
Anything that must be done goes through the *Candidate Task* Mechanism



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

35

Dependencies



36

Evo elements

Priorities

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

Let it be the most important 80%

37

Elements of Evo

- Focus on delivering *value* and *productivity gain* to stakeholders
- Constantly, actively learning. To optimize our Results
- Task Cycles to organize the work (estimation, planning, tracking, priority)
- Delivery Cycles to verify the requirements and assumptions
- Delivery Cycles to provide early productivity to the stakeholders
- Analysis Tasks to find out what we don't know yet
- TimeLine to keep vision and control over the whole project
- Working in a strict time box mode:
  Solving the estimation-planning-tracking weakness
- Active synchronization with related parties (e.g. hardware, other team, suppliers)
- Requirements and Risk Management are part of daily life
- Working on a strict priority basis
  (Why are we doing this? Why now? Who's waiting for this?)

38

Elements of Evo

- What we've done is done, we cannot change it any more
- What we do from now, we can control
- Constantly asking ourselves:
  What should we do now, in *which order*,
  to *which level of detail for this moment*
- Don't ostrich, we deliberately pull the head out of the sand
- The methods really work (otherwise we would discard them)
- No other method delivers better results faster (otherwise we would be using that method; nothing is sacred)
- You can start saving time, saving money immediately
- Relaxed working, yet higher productivity, no need for excuses any more
- Happy developers, happy customers, happy management
- Customer has choice in the time-to-market and features battle
- Quality is cheaper

39

Evo elements

Evo practice

Niels Malotaux

N R Malotaux Consultancy

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

1

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

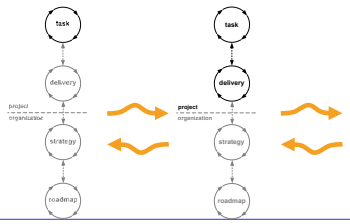
2

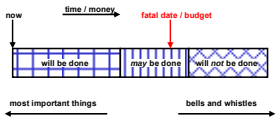
Evolutionary introduction pattern

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





3

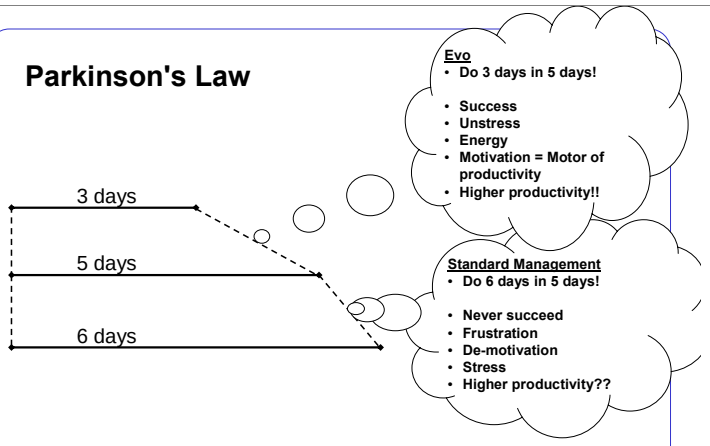
Booklets:
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf
http://www.malotaux.nl/nrm/pdf/Booklet2.pdf
http://www.malotaux.nl/nrm/pdf/EvoTesting.pdf

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

4

Parkinson's Law



"Work expands to fill the time available"

5

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
- We may include tasks in the planning to show that the hours for these tasks are not available for other work
- Plan *all* plannable hours

6

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

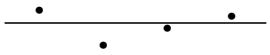
Types of Tasks

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

7

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**

8

TimeBox - taking Time seriously

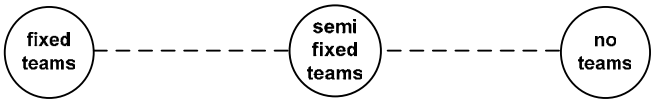
- **A TimeBox is the maximum time available for a Task**
- **When the time is up, the Task should be completely done: there is no more time !**
- **Because people tend to do more than necessary**
  (especially if the requirements of the Task are unclear)
  - Check halfway whether you're going to succeed 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)

9

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:**



10

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.**

11

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**

12

Evo practice

Meetings

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

13

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?
    - If you sense lack of 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**

14

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?

15

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

16

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 ...

17

Architect ↔ Project Manager

- Architect: Master Builder
- Architect is the conductor of the Product
- Project Manager is the conductor of the Project
- There is only one captain on the ship: the Project Manager
- Test Manager is the conductor of the Test Process

18

Evo practice

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

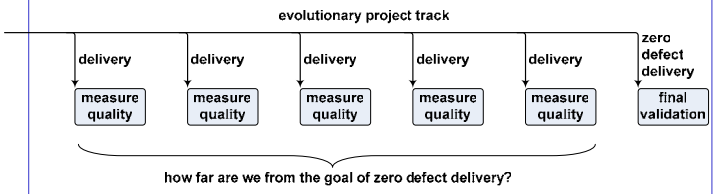
19

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

20

Testing in Evo



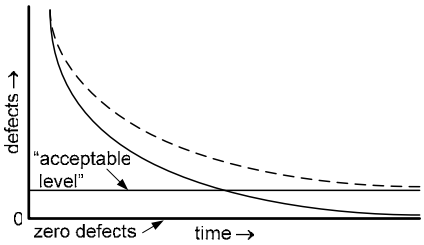
- Final validation shouldn't find any problems
- Earlier verifications mirror quality level to developers: how far from goal and what still to learn
- Evo has *no debugging phase!*

21

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

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

22

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)

23

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

24

Evo practice

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

25

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!"

26

Magic words

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

27

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

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)

28

Results of Evo

Solid control of development projects
by doing the *right things* in the *right order* to the *right level of detail*

- **Early results** regular, frequent deliveries of stakeholder value: right order
- **Better results** rapid, frequent feedback: do right things right
- **Faster results** only what is needed in the right order
- **Risk reduction** no missed deadlines, no unusable results
- **Less stressed developers** stress disappears *while producing more*
- **Happy customers** getting early and regular deliveries that can be *used*
- **More profits** better results in 30% less time saves costs
- **Magic bullet** remarkable results, no better alternative

29

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

30

ETA

Evo Task Administrator

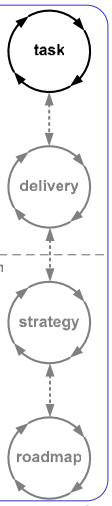
Niels Malotaux

N R Malotaux Consultancy

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

1

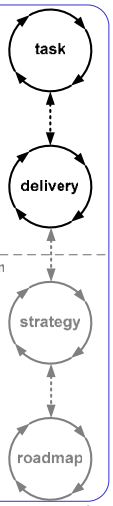
Cycles in Evo



- **Weekly Task Cycle**
  - 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 real effort hours in a week
  - In the remaining time: do whatever else you have to do
  - Tasks are always done, 100% done

2

Cycles in Evo

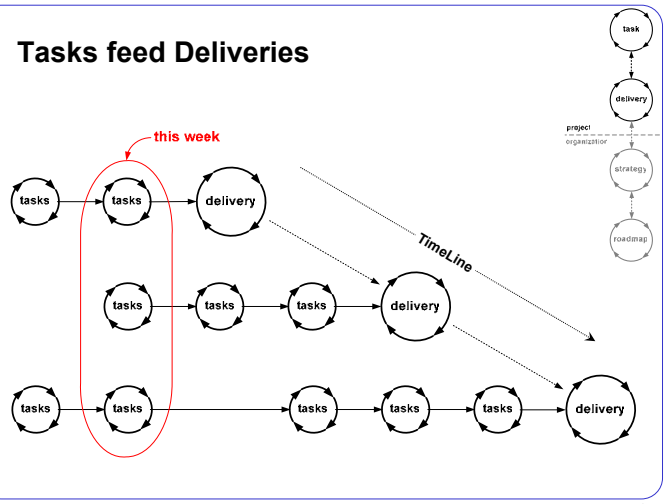


- **Weekly Task Cycle**
- **Value Delivery Cycle**
  - Are we *delivering* the right things, in the right order to the right level of detail
  - Optimising requirements and checking assumptions
  - Delivering the juiciest, most important stakeholder values that can be made in the least time
  - What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

3

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

Tasks feed Deliveries



4

ETA

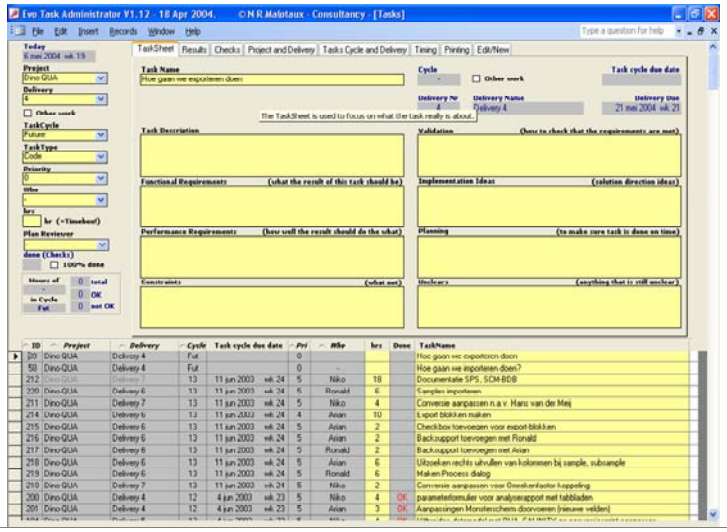
Evo Task Administrator

Download MS Access 2000 database:
http://www.malotaux.nl/nrm/Evo/ETAF.htm

Booklet "Evo Project Management Methods"
http://www.malotaux.nl/nrm/pdf/MxEvo.pdf

Booklet "How Quality is Assured by Evolutionary Methods"
http://www.malotaux.nl/nrm/pdf/Booklet2.pdf

5



Evo Task Administrator

TaskSheet: More time needed?

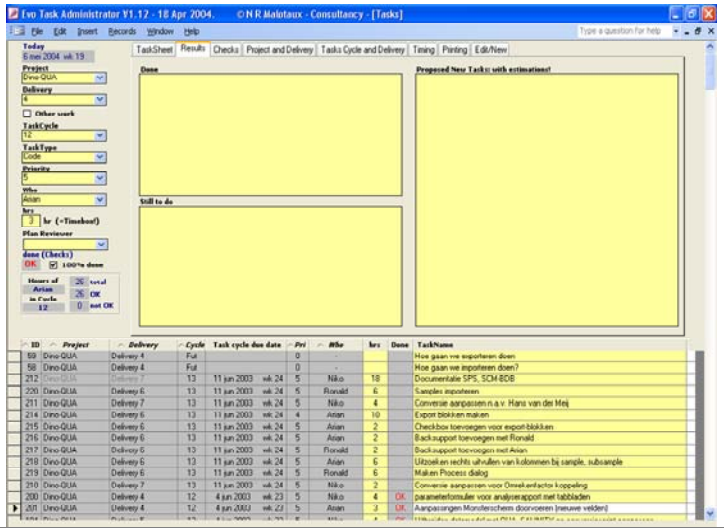
- Some people think they need more time for the Task if they "must" fill in the TaskSheet
- If you feel you "must fill in": Don't do it !
- If you think you need more time: add more time
- You will need this information during the Task anyway, so you should want it and
- It should *save* time

7

Analysis Tasks

- I don't know…
  - That's an Analysis Task!
  - How much time are you going to give yourself?
- To find out something we do not know
  - Use short TimeBox
  - Documented at the end of the TimeBox:
    - What do we know now
    - What do we not yet know
    - What should we know more
    - Which New Tasks can we define?
    - Estimation and priority of these tasks defined
- Typically Architecture and Design issues!

8



Evo Task Administrator

Accepting a Task

Acceptance of a Task means:

- Taking full responsibility for the successful conclusion of the Task within the time agreed

This also means:

- As soon as you know that you will not be able to conclude the task successfully, then notify Project Management to decide what to do with this information
- When the agreed time has come, no excuse (except act of God) is good enough for not having successfully concluded the Task: you simply failed your responsibility

10

Management Questions on Tasks

- Is the Project under Control?
- Show me !
  - No "holes" in OK's
  - All available plannable time planned
  - TaskSheets used
  - Results used
  - Prompt explanation in case of discrepancies

11

Evo Task Administrator

Supplemental slides

Niels Malotaux

N R Malotaux Consultancy

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

1

Case 1: First project (Q4 2000)

- Busy for some 9 months (15 people, € 2M spent)
- Still working to get "complete requirements"
- Organize software in 3 week increments
- Project was cancelled in first week
- Results came out one week later

2

Case 2: 2nd project (Jan ~ March 2001)

- Busy for some 9 months (5 people)
- Project already over time
- Software desperately unstable
- Motivation below zero
- Management had given up
- One more chance:
  Stable within 6 weeks or else …

3

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

Case 2: 2nd project (Jan ~ March 2001)

- All unnecessary functionality out
- One goal: stable software
- 1 week cycles
- Main problem: keeping focus
- Cause of instabilities: memory leaks
- Stable in 3 weeks

4

Case 2: 2nd project (Jan ~ March 2001)

- 6 more weeks to add lacking functionality
- Absolutely not more than 6 weeks *!*
- Introduced Evo planning: 6 cycles of 1 week
- People learn estimation in 3 weeks
- People get relaxed, start smiling again
- Motivation restored
- Hardly any new defects injected

5

Case 2: Result of 2nd project (Jan ~ March 2001)

- Project on track and successfully done
- Organization convinced:
  "Reorganize all other projects"

6

Supplemental slides

Case 3: Result of reorganization (end 2001)

- 4 projects delivered on time
  - 1 project had some troubles:
    - Without Evo planning they would not even have known
    - Not understanding crucial details of Evo, like:
      - Tasks done in a cycle must be 100% done
      - Producing stakeholder deliveries as soon as possible
      - Big bang delivery caused debugging phase, however debugging was done in evolutionary way (1 day cycles!)
    - Teaching is not enough, coaching is absolutely necessary
    - Still, management perception is that Evo saved the project
- 1 project continues after product release
- 2 new projects starting
- Coaching time per project: 5 - 15 days

7

Case 3: Result of reorganization (2002)

- CMM level 2 efforts:
  - Now they know their process
  - So they can document their process
  - Process is successful
  - So they can keep following the documented process
  - No discrepancies between CMM and Evo
  - CMM sets goals
  - Evo provides the way to the goals

8

Case 4: Other company

- Software for installation and service support
  - Budget of 10 days coaching
  - First Evo day: nobody knew what to do
  - First weeks: find out what has to be known
  - Second Evo day: two deliveries planned
  - Xmas 2001: two deliveries (CD) delivered on time
  - June 2002:
    - Continued to deliver CD's every two weeks
    - Developers ready, testers doing acceptance test
    - No debugging phase whatsoever
  - Enthusiastic response and feedback from customer
  - Only 9 days of coaching used
  - Everybody very happy

9

Supplemental slides

Case 5:

- 3 projects, 13 people,
  expecting 15 weeks of average 34 hours
  would have used ~ 6600 hours.
- With the investment of total 110 hours coaching,
  we expect to save 2200 hours.
- Assume coach cost double of project members,
  ROI will be 2200 / (110 * 2) is 10 to 1.

10

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

11

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

12

Supplemental slides

Case 8: US company

- Artificial Intelligence research project
- Three 3-day Evo coaching sessions, 5 weeks apart
- Government customer pleasantly surprised
- Project won 20 months follow-up project from 20 contenders, due to use of Evo methods

13

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

14

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!"

15

Supplemental slides

CMM compliance of Evo → Level 2

- **RM:** RM (Req's *and* Risk) in Evo is part of every day life
- **PP:** Keep existing estimating techniques for size, complexity, effort. Schedule to dynamic Evo priorities.
- **PTO:** Evo = continuous tracking & correction of plans. *100% done* principle allows better tracking.
- **SM:** Apply Evo principles to the subcontractor.
- **QA:** Very frequent review & testing (QC), Independent QA must be covered separately.
- **CM:** Apply all existing CM procedures (more integration cycles).
- **M&A:** Well implemented Evo provides weekly product completion & quality measures. Process Performance Measurement must be added.

16

CMM compliance of Evo → Levels 3,4,5

- **OPF:** After introduction Evo in projects, people only want to work the Evo way, allowing organization-wide use
- **OPD:** Dynamic Evo environment more a problem for assessors
- **ISM:** Evolutionary tailoring is a way of life in Evo
- **PE:** Product engineering tailoring is a way of life in Evo
- **IC:** Active synchronization provides for Intergroup Coordination
- **PR:** All forms of review are tasks, just as anything else
- **QPM:** Evo emphasizes numerical definition of goals and requirements
- **SQM:** Evo emphasizes numerical definition of goals and requirements
- **DP:** Defect Prevention is a way of life
- **TCM:** Evo heavily supports change management
- **PCM:** Evo heavily supports change management

17

Overlaps between Evo and XP (BLUE)

Planning

- User stories are written
- Release planning creates the schedule
- **Make frequent small releases**
- **The Project Velocity is measured**
- **The project is divided into iterations**
- **Iteration planning starts each iteration**
- Move people around
- A stand-up meeting starts each day
- **Fix XP when it breaks**

Designing

- **Simplicity**
- Choose a system metaphor
- Use CRC cards for design sessions
- **Create spike solutions to reduce risk**
- **No functionality is added early**
- Refactor whenever and wherever possible

Coding

- The customer is always available
- **Code to be written to agreed standards**
- Code the unit test first
- All production code is pair programmed
- Only one pair integrates code at a time
- **Integrate often**
- Use collective code ownership
- Leave optimization till last
- **No overtime**

Testing

- All code must have unit tests
- **All code must pass all unit tests before it can be released**
- When a bug is found tests are created
- **Acceptance tests are run often and the score is published**

18

Supplemental slides

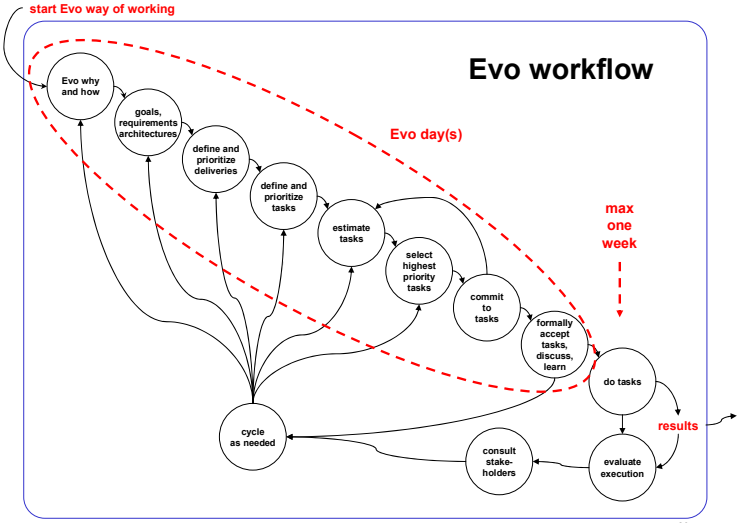
Differences between Evo and XP

Evo	XP
• Suited for large & small Systems & Software Development	• Suited for small Software Development
• Results Centric	• Code Centric
• Stakeholder focus	• Developers focus above Process focus
• Works with anybody	• Need seasoned programmers

• Numeric
  • Specification of (strategic) objectives
  • Prioritization
  • Progress tracking
	• NO numeric specification of objectives, prioritization nor tracking

• Requirements/Risk management using Planguage with multiple Stakeholders	• One Stakeholder in the room is walking Requirements document

19



20

Supplemental slides

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Quality On Time

Selection of posters

Glue slides 2 ~ 12 on a wall or door for all to see and study

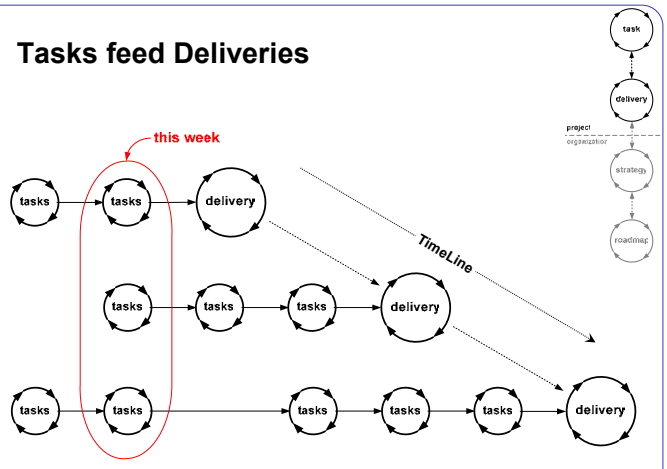
Niels Malotaux

N R Malotaux Consultancy

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

1

Tasks feed Deliveries



2

Task selection criteria

- Most important requirements first
- Highest risks first
- Most educational or supporting for development first
- Actively Synchronize with other developments
- Every cycle delivers a useful, *completed*, result

3

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

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

Delivery selection criteria

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

- 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

4

Types of Tasks

1. Tasks done within estimated time (= timebox)
2. Analysis Tasks
   - 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

5

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
- We may include tasks in the planning to show that the hours for these tasks are not available for other work
- Plan *all* plannable hours

6

Selection of posters

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

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.

7

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

8

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

9

Selection of posters

Slash Project Time with Evolutionary Methods

How to deliver the best possible results in the shortest possible time

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)

10

Accepting a Task

Acceptance of a Task means:

- Taking full responsibility for the successful conclusion of the Task within the time agreed

This also means:

- As soon as you know that you will not be able to conclude the task successfully, then notify Project Management to decide what to do with this information
- When the agreed time has come, no excuse (except act of God) is good enough for not having successfully concluded the Task: you simply failed your responsibility

11

Requirements in TaskSheets

All Tasks shall at least contain in the TaskSheet:

- Stakeholder(s) for this Task
- Requirements of the Stakeholder(s) for this Task

Rationale:

- No Stakeholder means: no Requirement
- No Requirement means: no work to do

- Without a stated requirement: how do we know that the Task is concluded OK?

12

Selection of posters

TaskCycle preparation		
Name TaskCycle Available plannable hours	Your Goal	
Project	Project Goal	
Stakeholders	Stakeholder Requirements	
Deliveries	Most productive delivery	
Tasks to do	Estimation = timebox	Priority

Task Cycle Plan	
name	
Taskcycle	Available plannable hrs

project	delivery due	task due	hrs	prio	tsk sht	done	description

TaskSheet for week	Assigned to	Estimated duration
Task description		
☑ **Requirements for this task to be used as reference for verification** • **Functions (what should it do?)**		
• **Qualities (how well should it do it)** State definitions of e.g. "usability", "user-friendly", "response time", etc. Don't state trivial qualities of your work, like "no bugs", or "no leaks": your work is supposed to be Quality On Time. That is, simply the right things, simply within the time agreed.		
• **Constraints**		
☑ **Which activities must be done to realize the requirements stated?** What has to be done before I can say "It is completely finished, I don't have to think about it any more". If the task is a modification, state what modifications have to be done.		
☑ **Implementation details (how am I going to implement it)**		
☑ **Verification approach – test design** How can I make sure that it does what it should do and that it does not do what it should not do.		
☑ **Planning** (in which order am I going to do things to move efficiently towards the final result?) What to do first, what to do then: evolutionary steps, no big bang		
☑ **Is everything really clear?**		
☑ **Have this document (and related docs, if any) reviewed**		
☑ **Clarify any unclearness until everything is clear and agreed with the reviewer**		
☑ **Detail the design**		
☑ **Convert the detailed design to code**		
☑ **Verify against the written requirements (not less, not more) and against the design according to the defined test. Comments:**		
☑ **Checklist for 100% done:**		
☑ **The code compiles and links with all files in integration promotion level** ☑ **The code simply does what it should do: no bugs** ☑ **There are no memory leaks** ☑ **Defensive programming measures have been implemented** ☑ **All files are labeled according to the rules agreed** ☑ **File promotion is done** ☑ **I feel confident that the tester will find no problems**		
☑ **Project manager is informed about task completion**		