
Keio-SDM - Yokohama
12 October 2010

Niels Malotaux

# Predictable Projects

## Delivering the Right Result at the Right Time

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

# Niels Malotaux

Niels Malotaux is an independent Project Coach and expert in optimizing project performance. He has some 35 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 helping projects to deliver Quality On Time: delivering what the customer needs, when he needs it, to enable customer success. Niels effectively teaches Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached well over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, Ireland, India, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less well in practice. He is a frequent speaker at conferences, see www.malotaux.nl/nrm/Conf

Niels puts development teams on the Quality On Time track and coaches them to stay there and deliver their quality software or systems on time, without overtime, without the need for excuses. Practical methods are developed, used, taught and continually optimized for:

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

Within a few weeks of turning a development project into an Evo project, the team has control and can tell the customer when the required features will all be done, or which features will be done at a certain date. Niels enjoys greatly the moments of enlightenment experienced by his clients when they find out that they can do it, that they are really in control, for the first time in their lives.

**Niels Malotaux**

# Predictable Projects

## Delivering the Right Result at the Right Time

# Predictable Projects

## Delivering the Right Result at the Right Time

Niels Malotaux

N R Malotaux
Consultancy

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

EvoShort- 2010 1

Niels Malotaux

## Result Management

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

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

EvoShort- 2010 2

**Niels Malotaux**

# Predictable Projects

## Delivering the Right Result at the Right Time

### Predictable Projects ?

- Any problems with projects ?

*

EvoShort- 2010 3

### Not every project is successful

(at first)

- Apparently we're doing something wrong
- Otherwise projects would succeed and be on time

- Heathrow Terminal 5: "Great success !"
  - Normal people aren't interested in the technical details of a terminal
  - They only want to check-in their luggage as *easily* as possible and
  - Get their luggage back as *quickly* as possible in *acceptable condition at their destination*
  - They didn't
- One of the problems is to determine what the project (or our work in general) really is about

EvoShort- 2010 4

## What is the most important Requirement ?

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



## Fallacy of '*all*' requirements

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

If our previous project was late,
our current project will also be late

unless we do things *differently* and *better*

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

Projects don't have to be late
They deserve better

EvoShort- 2010 7

## Causes of Delay

- Some typical causes of delay are:
  - Developing the wrong things
  - Unclear requirements
  - Misunderstandings
  - No feedback from stakeholders
  - No adequate planning
  - No adequate communication
  - Doing unnecessary things
  - Doing things less cleverly
  - Waiting (before and during the project)
  - Changing requirements
  - Doing things over
  - Indecisiveness
  - Suppliers
  - Quality of suppliers results
  - No Sense of Urgency
  - Hobbying
  - Political ploys
  - Boss is always right (culture)
- Excuses, excuses: it's always "*them*". How about "*us*" ?
- A lot of delay is avoidable and therefore unjustifiable

EvoShort- 2010 8

## What has this to do with Product Development?

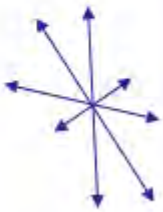
- The Project Manager is *responsible* for *delivering* the right result at the right time
- The Project Worker's work and decisions *determine* the *result* and the *time* it is delivered
- This makes everybody in the project implicitly *as responsible as* Project Management

EvoShort- 2010 9

## Systems Engineering

- Other Engineering (?)
  - Silo thinking
  - Sub-optimizing
  - Gold plating (hobbies)
  - Little attention to interfaces
  - Projects are always *multidisciplinary*

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



EvoShort- 2010 10

## Multidisciplinary ↔ Interdisciplinary

- Tension between
  - Technologically possible
  - Economically profitable
  - Socially and psychologically acceptable
  - All kinds of disciplines needed for a good solution
- Multidisciplinary
  - Many disciplines work in the project
  - Optimize solution in their own domain
- Interdisciplinary
  - Many disciplines work *together* in the project
  - Overall-optimizing
  - First *developing the problem* before developing the solution

EvoShort- 2010 11

## What *is* On Time ?

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

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

EvoShort- 2010 12

## Cost of one day of delay

- Do you know how much you cost per day?
  Note: that's not what you get !
- New electronic measuring instrument
  - 40 people in Oregon, US
  - 8 people in Bangalore, India
- US$ 40,000 per day for the project
- Plus US$ 30,000 per day for lost benefit
- Total: US$ 70,000 per day for every day of (unnecessary) delay
- $0^{th}$ order estimations are good enough

EvoShort- 2010 13

## Time to market

- Project cost per day € 2.500
- 5000 products per year ≈ 20 products per day
- € 5000 per product
- Profit € 500 per product
- Profit € 10.000 per day

- Every day we start later,
  we'll be done a day later and miss € 10.000
- Every day we work more, we'll be done a day later
  and spend € 2.500 more and miss € 10.000

EvoShort- 2010 14

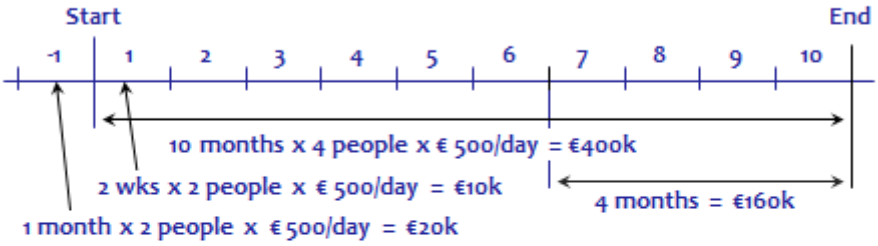
## Project ROI


**Return on Investment (ROI)**

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

EvoShort - 2010 15

## The Cost of Time




- We can save 4 months by investing €200k → "That's too much !"
- It's a *nicer* solution - Let's do 2 weeks more research on the benefits
- What are the expected revenues when all is done? → €16M/yr (1.3M/mnd)
- So 2 weeks extra doesn't cost €10k, but rather €16M/24 = €670k
- And saving 4 months brings €16M/3 = €5M extra

→ Invest that €200k *NOW* and *don't waste time*!

EvoShort - 2010 16

**Niels Malotaux**

# Predictable Projects

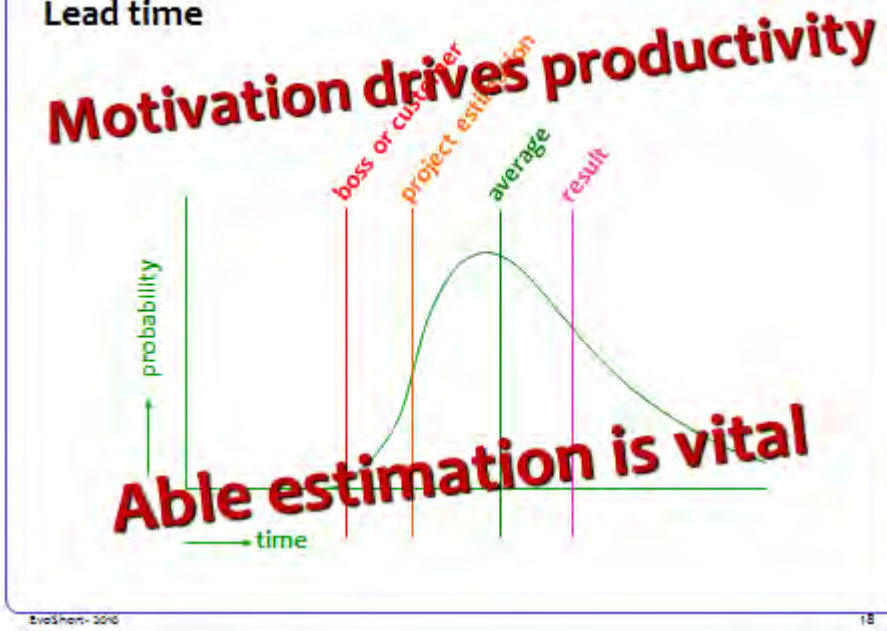
## Delivering the Right Result at the Right Time

### The challenge

- Getting and keeping the project under control
- Never to be late
- If we are late, we *failed*
- No excuses when we're not done at the FatalDay
- Not stealing from our customer's (boss) purse
- The only justifiable cost is the cost of developing the right things at the right time
- The rest is *waste*
- Would we enjoy producing waste ?



### Lead time



## Estimation Exercise

Are you an optimistic or a realistic estimator?

Let's find out !

Project:
Multiplying two numbers of 4 figures

How many seconds would you need to complete this Project?

EvoShort- 2010 19

## Is this what you did?

EvoShort- 2010 20

## Defect rate

- Before test ?
- After test ?

EvoShort- 2010 21

## Alternative Design (how to solve the requirement)

EvoShort- 2010 22

## Another alternative design

There are usually more, and possibly better solutions than the obvious one

EvoShort- 2010 23

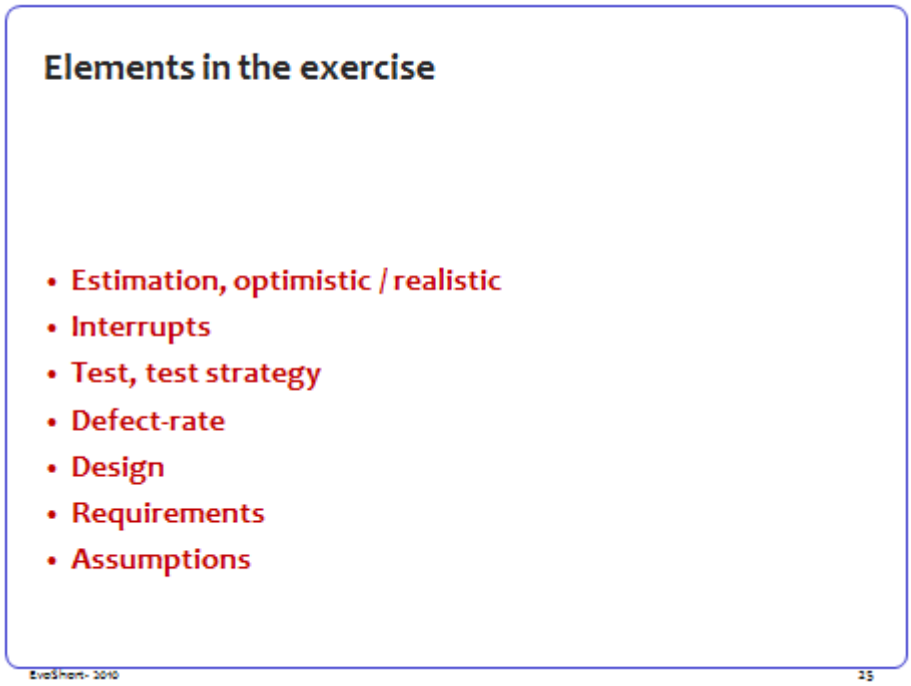
## What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

EvoShort- 2010 24


Elements in the exercise
• Estimation, optimistic / realistic
• Interrupts
• Test, test strategy
• Defect-rate
• Design
• Requirements
• Assumptions
EvoShort- 2010
35

# Human Behavior

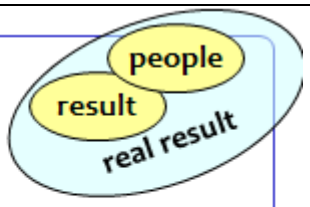


## Human Behavior

- Systems are conceived, designed, implemented, maintained, used, and tolerated (or not) by people
- People react quite predictably
- However, often differently from what we intuitively think

- Most project process approaches (PMI, INCOSE, as well as developers)
  - ignore human behavior,
  - incorrectly assume behavior,
  - or decide how people should behave *(ha ha)*
- To succeed in projects, we must study and adapt to real behavior rather than assumed behavior
- *Even if we don't agree with that behavior*

## Is Human Behavior a risk?




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

EvoShort- 2010 18

## People responsible for success

- *During* the project
  - Can still influence the performance of the project
  - First responsibility of the Project Manager
  - Actually responsibility of the whole development organization
- *After* the project, once the system is out there
  - No influence on the performance of the system any more
  - System must perform autonomously
  - So the performance must be there *by design*
  - Including appropriate interface with humans
  - Responsibility and required skill of Systems Engineering

EvoShort- 2010 19

## Discipline

- Control of wrong inclinations
- Even if we know how it should be done …
  (if nobody is watching …)
- Discipline is very difficult
- Romans 7:19
  - The good that I want to do, I do not …

→ Helping each other (watching over the shoulder)
→ Rapid success (do it 3 weeks for me…)
→ Making mistakes (provides short window of opportunity)
→ Openness (management must learn how to cope)

EvoShort- 2010 30

## Intuition

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

EvoShort- 2010 31

## Is intuition wrong, or is the design wrong ?


EvoShort- 2010 32

## 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

EvoShort- 2010 33

## Perception

- Quick, acute, and intuitive cognition (www.M-W.com)
- What people say and what they do is not always equal
- The head knows, but the heart decides
- Hidden emotions are often the drivers of behavior
- Customers who said they wanted lots of different ice cream flavors from which to choose,
  still tended to buy those that were fundamentally vanilla
- So, trying to find out what the real value to the customer is, can show many paradoxes
- Better not simply believe what they say: check!

EvoShort- 2010 34

## Logical thinking is not always better

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

EvoShort- 2010 35

## People *like* change, if it's an improvement!

- People are not against change
- People (sub-consciously) don't like uncertainty
- Any project changes something and thus introduces uncertainty
- People can cope with uncertainty for a short time

EvoShort- 2010 36

## Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures
- At the Fatal Day, any excuse is in vain: we failed
- Even if we "couldn't do anything about it"
- Failure is a very hard word. That's why we are using it!
- No pain, no gain
- We never say: "*You* failed", better: "*We* failed"
  - After all, *we* didn't help the person not to fail

EvoShort- 2010 37

## We failed because of politics

- Good politics:
  - People decide differently based on different values
- Bad politics: hidden agenda's
  - Say this, mean that - often even unintentionally
  - Politics thrive by vagueness
  - Facts can make bad politics loose ground
- If you accepted the responsibility for the project, failure because of "politics" is just an excuse
- What did you really do about it ?

EvoShort - 2010 38

## Culture

- It failed because of the existing culture

(no good excuse !)

- Culture is the result of how people work together
- Culture can't be changed
- Culture *can* change
- By doing things differently

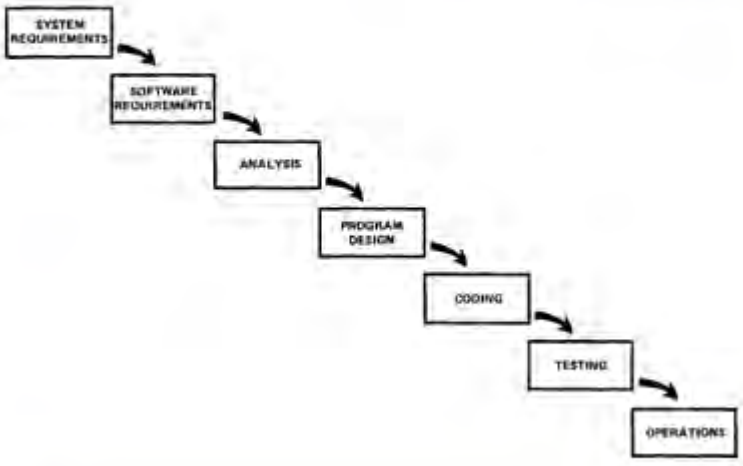
EvoShort - 2010 39

# Project Life Cycles

EvoShort- 2010 40

## Waterfall ?

Winston Royce 1970




EvoShort- 2010 41

## When can we use waterfall ?

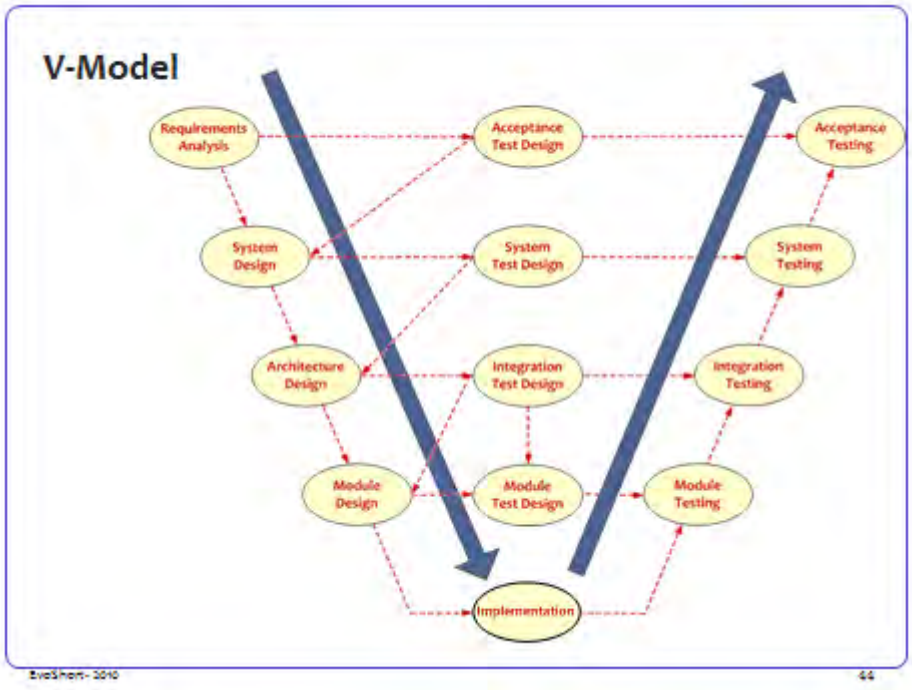
- Requirements are completely clear, nothing will change
- We've done it may times before
- Everybody knows exactly what to do
- We call this *production*

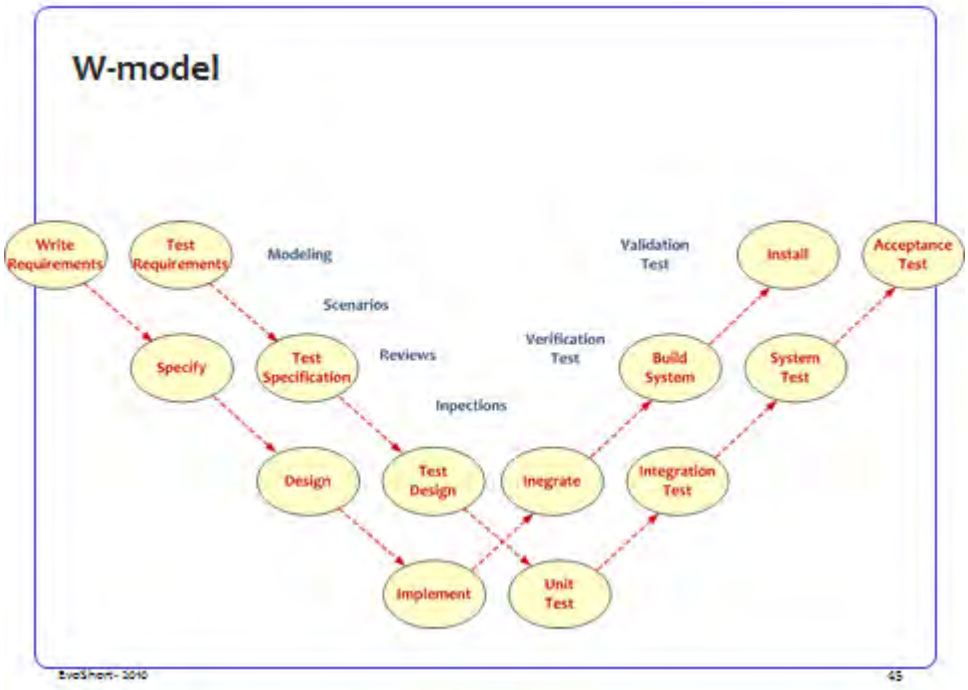
- In your projects:
  - Is everything completely clear ?
  - Will nothing change ?
  - Does everybody know exactly what to do ?
  - Are you sure ?

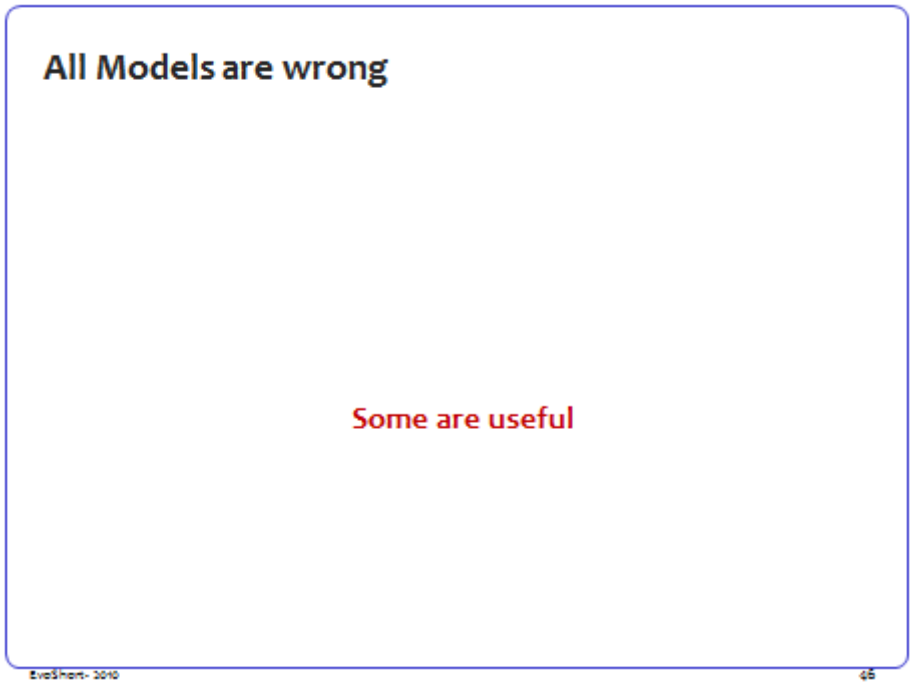
EvaShort- 2010 42

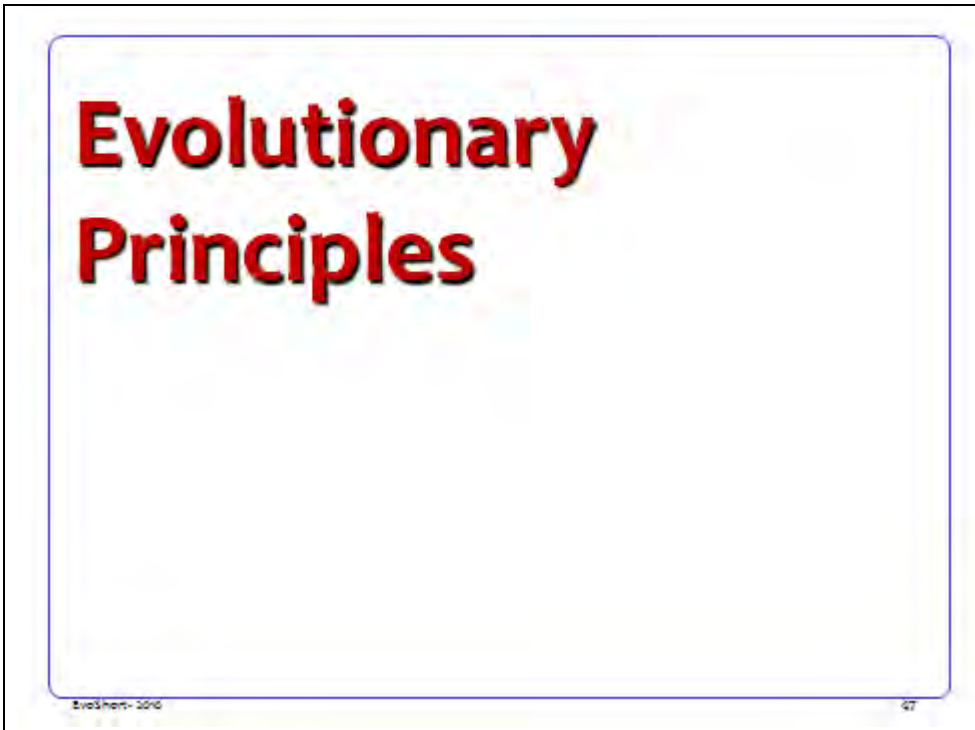
## How management likes it

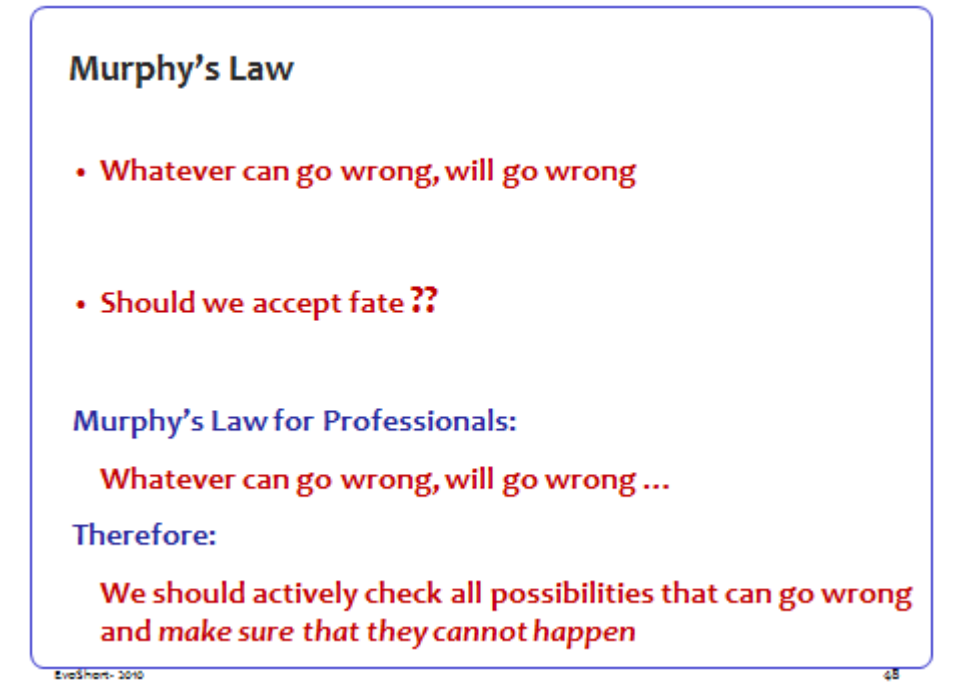

EvaShort- 2010 43


V-Model
Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Module Test Design
Module Testing
Implementation
EvoShort- 2010
44


W-model
Write Requirements
Test Requirements
Modeling
Validation Test
Install
Acceptance Test
Scenarios
Specify
Test Specification
Reviews
Verification Test
Build System
System Test
Inpections
Design
Test Design
Integrate
Integration Test
Implement
Unit Test
EvoShort- 2010
45


All Models are wrong
Some are useful
EvoShort- 2010
46


Evolutionary
Principles
EvoShort- 2010
47


Murphy's Law
• Whatever can go wrong, will go wrong
• Should we accept fate ??
Murphy's Law for Professionals:
Whatever can go wrong, will go wrong ...
Therefore:
We should actively check all possibilities that can go wrong and make sure that they cannot happen
EvoShort- 2010
48

## Preflection, foresight, prevention

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

Albert Einstein 1879-1955, Benjamin Franklin 1706-1790, it seems Franklin was first

Only if we change our way of working,
the result may be different

- Hindsight is easy, but reactive
- Foresight is less easy, but proactive
- Reflection is for hindsight and learning
- Preflection is for foresight and prevention

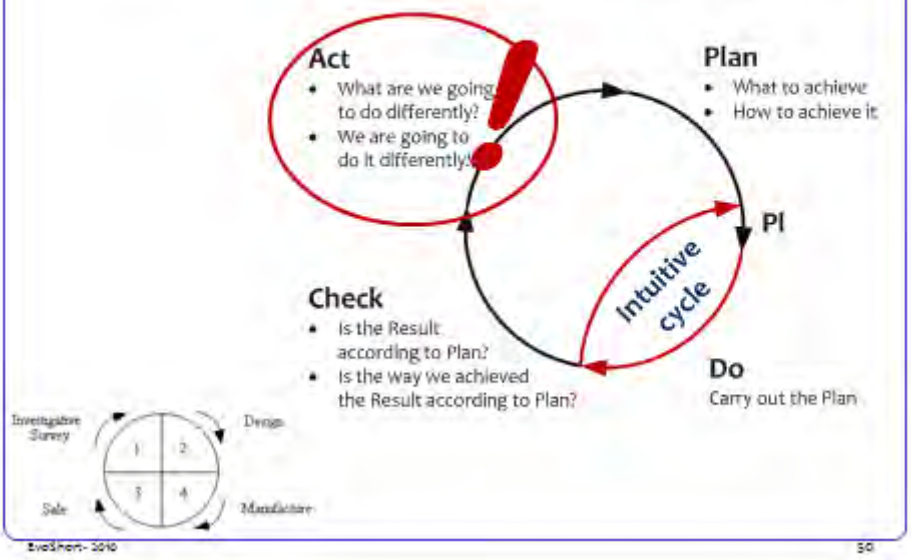
Only with *prevention* we can save precious time

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

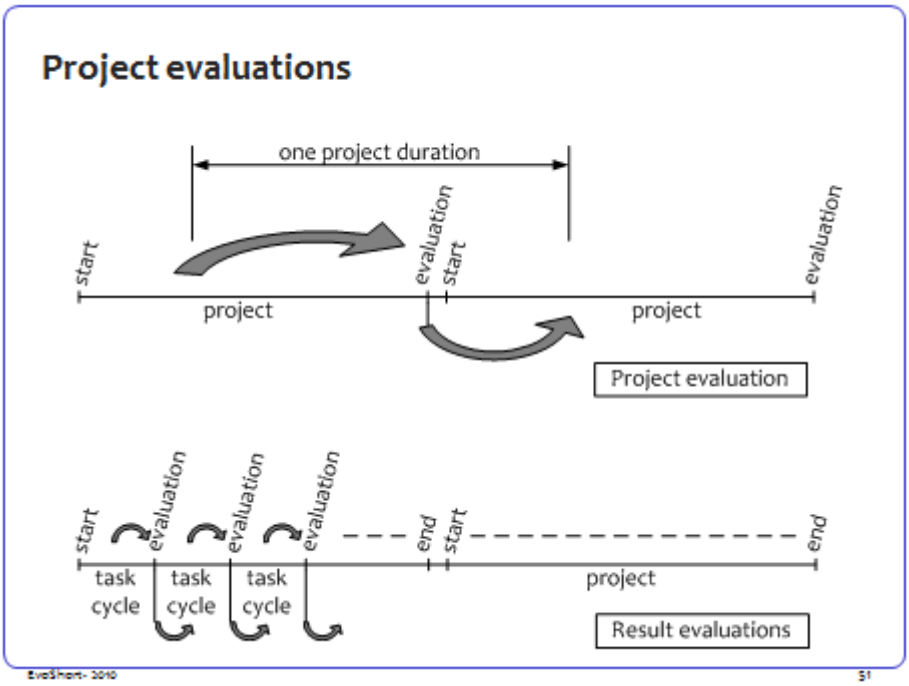
EvoShort- 2010 49

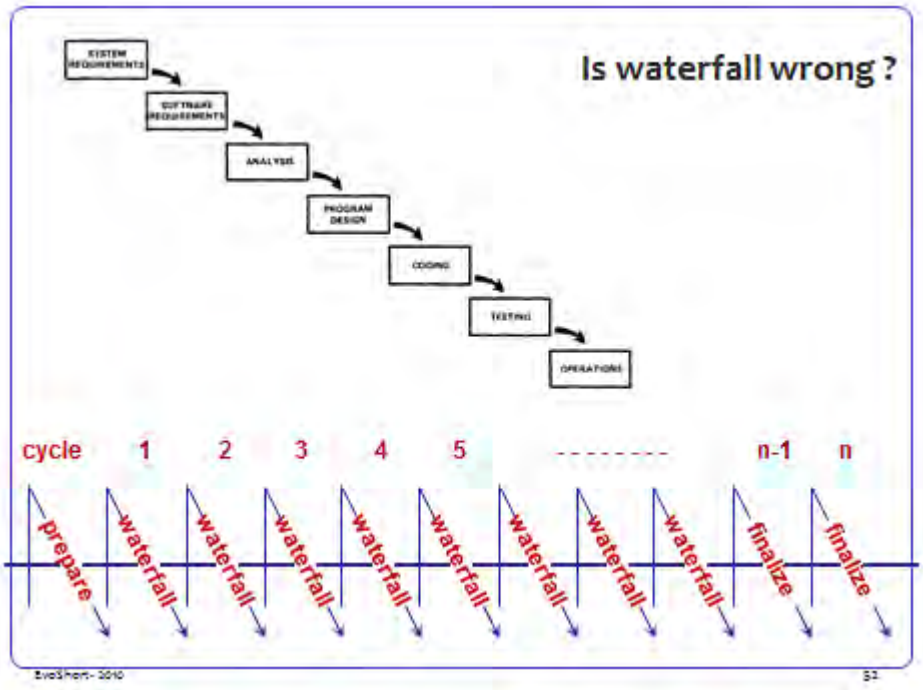
## The essential ingredient: the PDCA Cycle

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

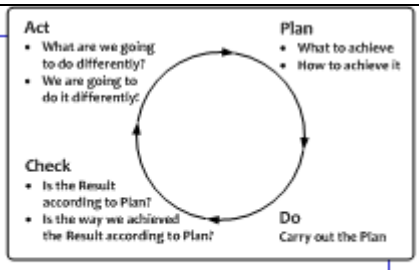



EvoShort- 2010 50


Project evaluations
one project duration
start
evaluation
start
evaluation
project
project
Project evaluation
start
evaluation
evaluation
evaluation
end
start
end
task cycle
task cycle
task cycle
project
Result evaluations


Is waterfall wrong ?
cycle
1
2
3
4
5
n-1
n
prepare
waterfall
finalize

## Knowledge how to achieve the goal




If we
- Use very short Plan-Do-Check-Act cycles
- Constantly selecting the most important things to do
- Don't do unnecessary things

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

doing the right things right

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

EvoShort- 2010 53

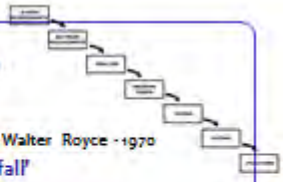
## Known for decades

Eliminating Waste
Not doing what doesn't yield value

- Benjamin Franklin (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- Henry Ford (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - Learning from waste, keeping things clean and safe, better treated people produce more
- Toyoda's (Sakichi, Kiichiro, Eiji) (1867-1930, 1894-1952, 1913-)
  - Jidoka: Zero-Defects, stop the production line (1926)
  - Just-in-time – flow – pull
- W. Edwards Deming (1900-1993)
  - Shewart cycle: Design-Produce-Sell-Study-Redesign (Japan – 1950)
  - Becoming totally focused on quality improvement (Japan – 1950)
    Management to take personal responsibility for quality of the product
  - Out of the Crisis (1986) - Reduce waste
- Joseph M. Juran (1904-2008)
  - Quality Control Handbook (1951, Japan – 1954)
  - Total Quality Management – TQM
  - Pareto Principe
- Philip Crosby (1926-2001)
  - Quality is Free (1980)
    - Zero-defects (1961)
- Taiichi Ohno (1912-1990)
  - (Implemented the) Toyota Production System (Beyond Lange-Scale Production) (1988)
  - Absolute elimination of waste - Optimizing the TimeLine from order to cash
- Masaaki Imai (1930-)
  - Kaizen: The Key to Japan's Competitive Success (1986)
  - Gemba Kaizen: A Commonsense, Low-Cost Approach to Management (1997)

EvoShort- 2010 54

## There is nothing new in software too



- Managing the development of large software systems - Walter Royce - 1970
  - Famous "Waterfall document": figure 2 showed a 'waterfall'
  - Text and other figures showed that Waterfall doesn't work
  - Anyone promoting Waterfall doesn't know or didn't learn from history
- Incremental development - Harlan Mills - 1971
  - Continual Quality feedback by Statistical Process Control (Deming !)
  - Continual feedback by customer use
  - Accommodation of change - Always a working system
- Cleanroom software engineering - Harlan Mills - 1970's
  - Incremental Development - Short Iterations
  - Defect *prevention* rather than defect removal
  - Statistical testing
  - 10-times less defects at lower cost
  - Quality is *cheaper*
- Evolutionary Delivery - Evo - Tom Gilb - 1974, 1976, 1988, 2005
  - Incremental + Iterative + *Learning and consequent adaptation*
  - Fast and Frequent Plan-Do-Check-Act
  - Quantifying Requirements - Real Requirements
  - Defect *prevention* rather than defect removal




EvoShort- 2010 55

## Cobb's Paradox

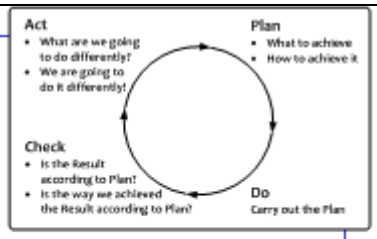
- We know why projects fail
- We know how to prevent their failure
- So why do they still fail ?

Martin Cobb
Treasury Board of Canada Secretariat
Ottawa, Canada

1989

EvoShort- 2010 56

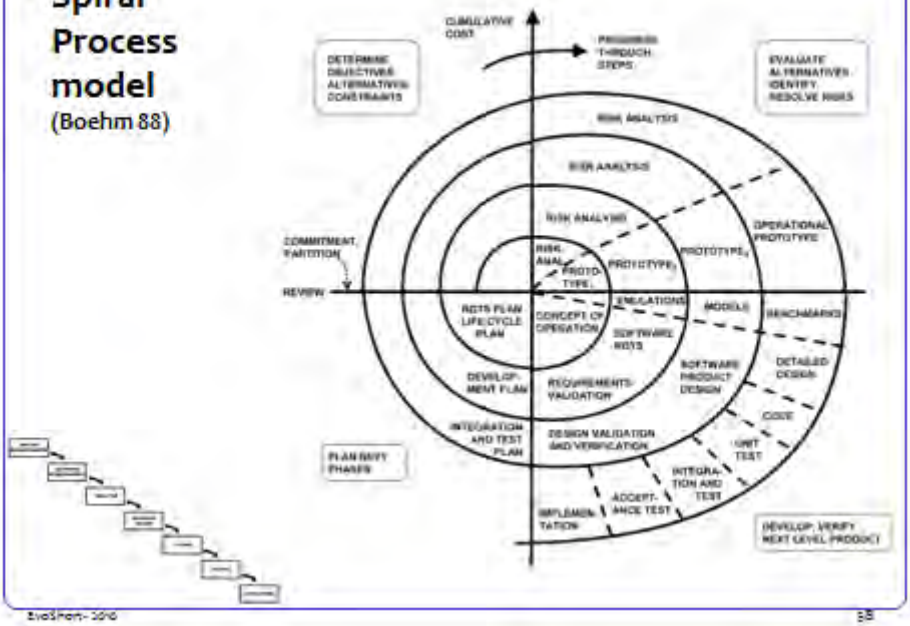
## Evo




- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle actively, deliberately, rapidly and frequently, for *Product*, *Project* and *Process*, based on ROI and highest value
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- We know we are not perfect, but the customer shouldn't be affected
- Evo is about delivering Real Stuff to Real Stakeholders doing Real Things *"Nothing beats the Real Thing"*
- Projects seriously applying Evo, routinely conclude successfully on time, or earlier

EvoShort- 2010 57

## Spiral Process model (Boehm 88)

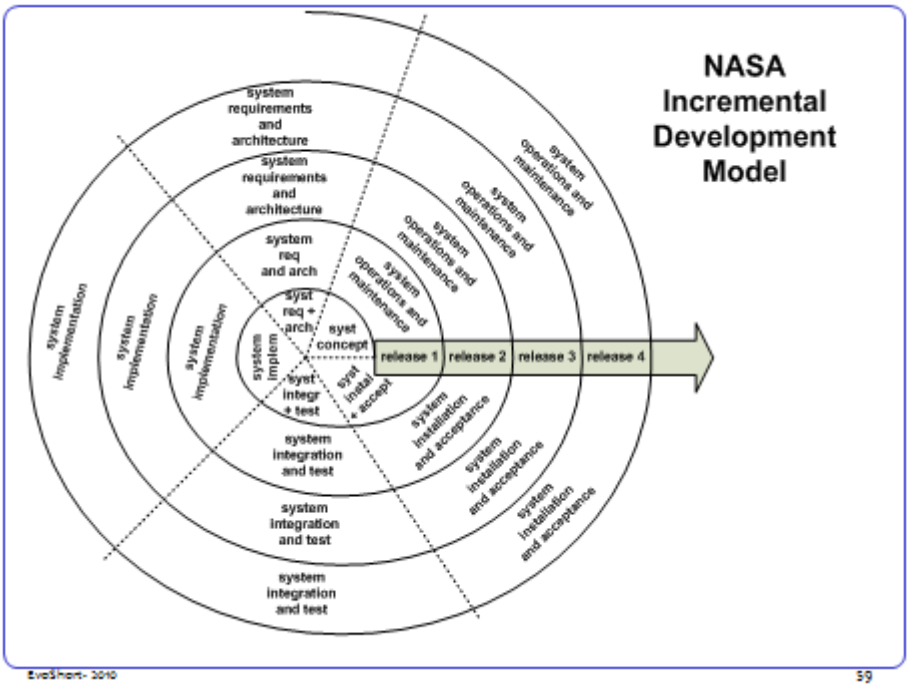


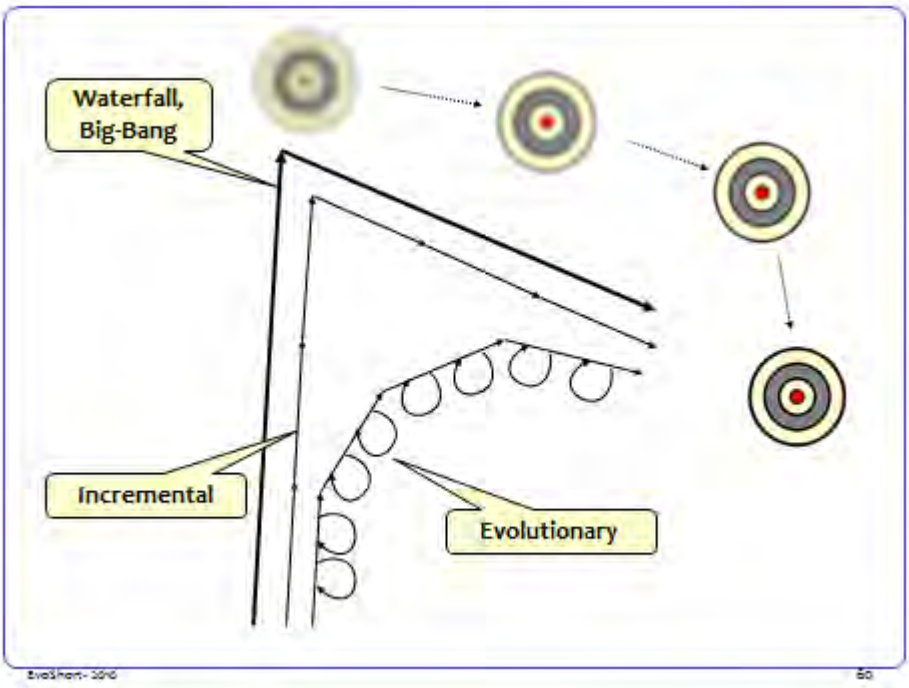

EvoShort- 2010 58

**Niels Malotaux**

# Predictable Projects

## Delivering the Right Result at the Right Time



## Evolutionary Project Management (Evo)

- Plan-Do-Check-Act
  - The powerful ingredient for success
- Business Case
  - Why we are going to improve *what*
- Requirements Engineering
  - What we are going to improve *and what not*
  - How much we will improve: quantification
- Architecture and Design
  - Selecting the optimum compromise for the conflicting requirements
- Early Review & Inspection
  - Measuring quality while doing, learning to prevent doing the wrong things

Zero Defects Attitude

Right product

## Evo Project Planning

- Weekly TaskCycle
  - Short term planning
  - Optimizing estimation
  - Promising what we can achieve
  - Living up to our promises
- Bi-weekly DeliveryCycle
  - Optimizing the requirements and checking the assumptions
  - Soliciting feedback by delivering Real Results to *eagerly waiting* Stakeholders
- TimeLine
  - Getting and keeping control of Time: Predicting the future
  - Feeding program/portfolio/resource management

Right time

EvoShort - 2010 61

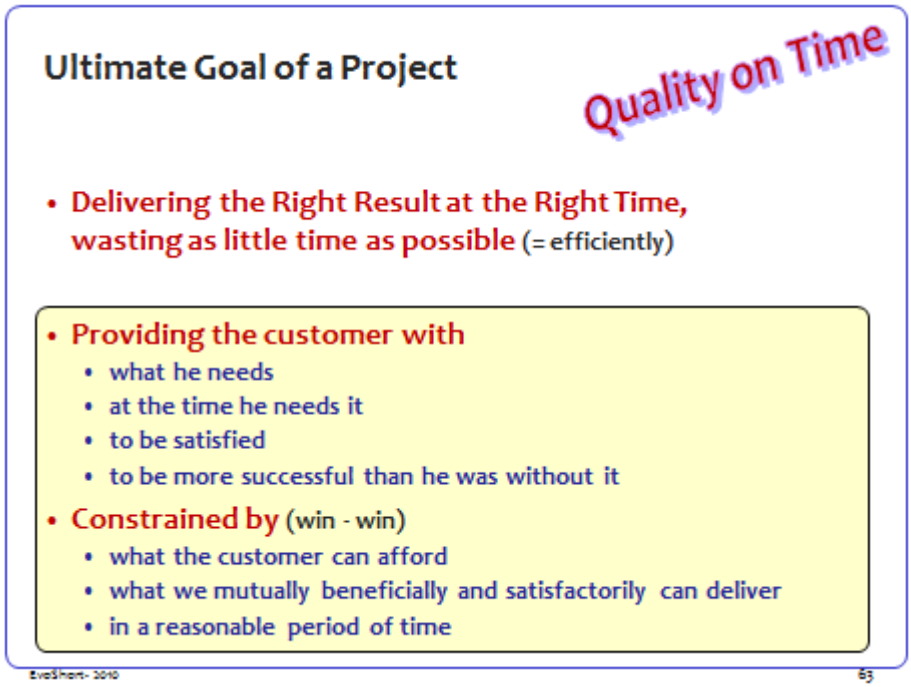
## Is Zero Defects possible?

- Zero Defects is an asymptote

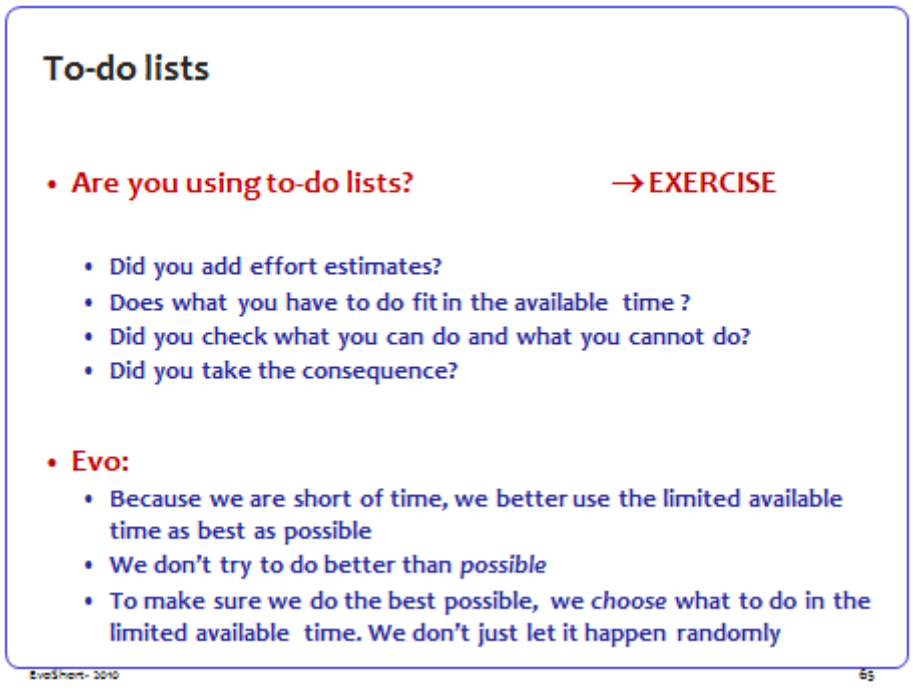

- We aren't perfect, but the customer shouldn't find out
- What we deliver simply works

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

EvoShort - 2010 62


Ultimate Goal of a Project
Quality on Time
• Delivering the Right Result at the Right Time, wasting as little time as possible (= efficiently)
• Providing the customer with
• what he needs
• at the time he needs it
• to be satisfied
• to be more successful than he was without it
• Constrained by (win - win)
• what the customer can afford
• what we mutually beneficially and satisfactorily can deliver
• in a reasonable period of time
EvoShort- 2010
63


Evolutionary
Planning
TaskCycle
DeliveryCycle
EvoShort- 2010
64


To-do lists
• Are you using to-do lists? → EXERCISE
• Did you add effort estimates?
• Does what you have to do fit in the available time ?
• Did you check what you can do and what you cannot do?
• Did 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 choose what to do in the limited available time. We don't just let it happen randomly
EvoShort- 2010
65

## Evo Planning: Weekly TaskCycle


- Are we *doing* the right things, in the right order, to the right level of detail for now
- Optimizing estimation, planning and tracking abilities to better predict the future
- Select highest priority tasks, never do any lower priority tasks, never do undefined tasks
- There are only about 26 plannable hours in a week (2/3)
- In the remaining time: do whatever else you have to do
- Tasks are always done, 100% done

EvoShort- 2010 66

## Effort and Lead Time

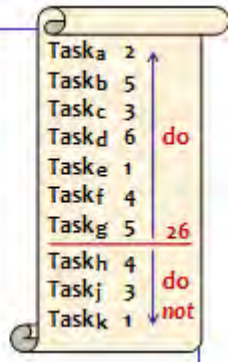
- Days estimation → lead time (calendar time)
- Hours estimation → effort

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

EvoShort- 2010 67

## Every week we plan

- How much time do we have available
- 2/3 of available time is net plannable time
- What is most important to do
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr per week)
- What can, and are we going to do
- What are we not going to do
- *Write it down! Our fuzzy mind isn't good enough!*



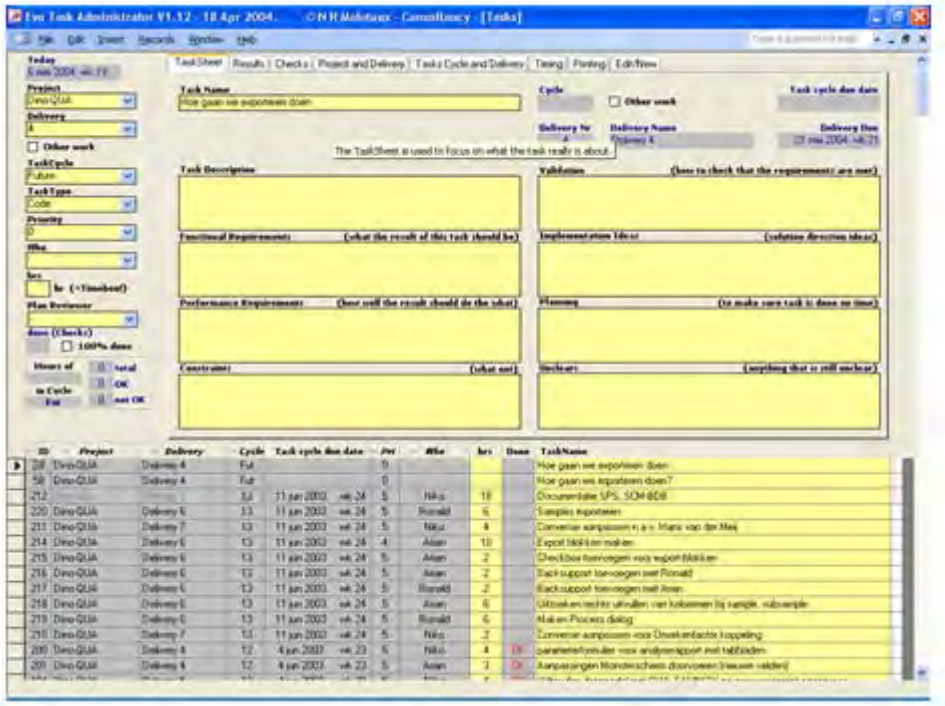

2/3 is default start value
this value works well in development projects

EvoShort- 2010 68

## Weekly 3-Step Procedure

- Individual preparation
  - Conclude current tasks
  - What to do next
  - Estimations
  - How much time available
- Modulation with / coaching by Project Management
  - Status
  - Priority check
  - Feasibility
  - Commitment and decision
- Synchronization with group (team meeting)
  - Formal confirmation
  - Concurrency
  - Learning
  - Helping
  - Socializing

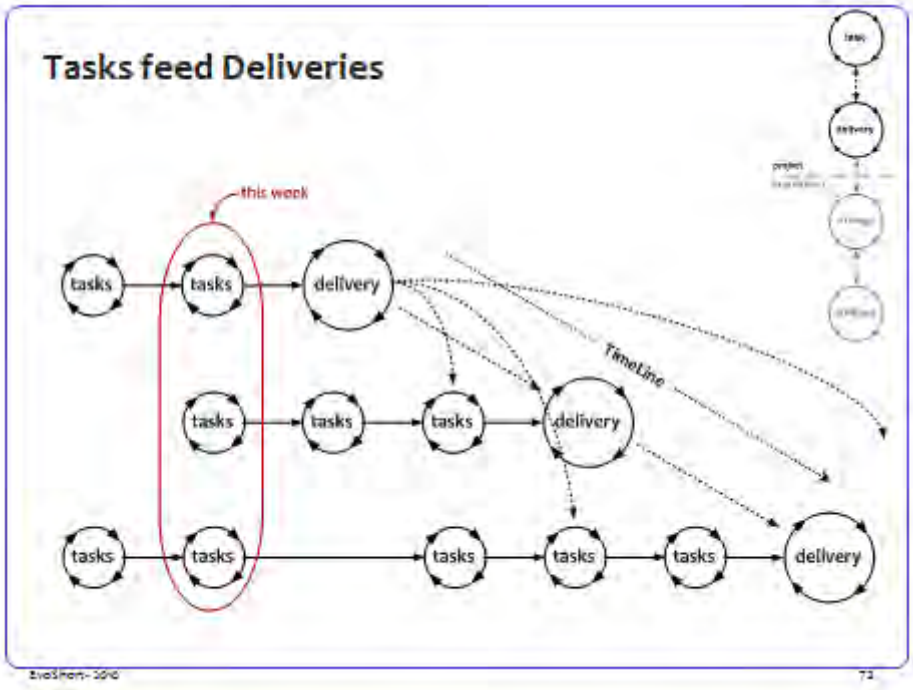
EvoShort- 2010 69

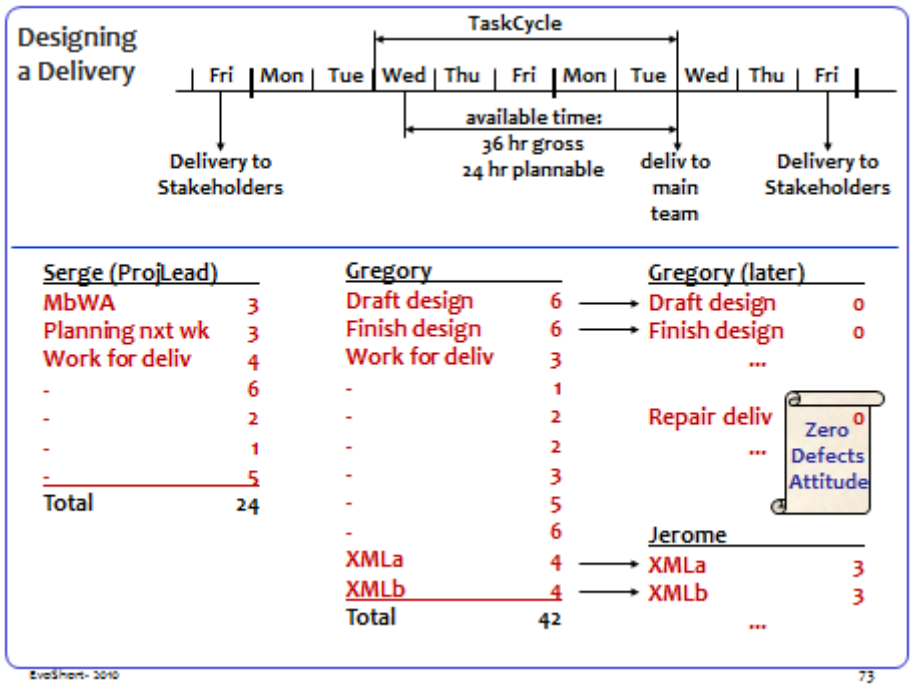


## DeliveryCycle

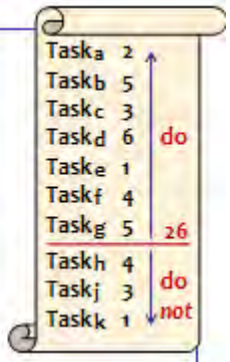
- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
     stakeholder values that can be made in the least time
     - What will make Stakeholders more productive now
- Not more than 2 weeks


EvoShort- 2010 71


Tasks feed Deliveries
this week
tasks
tasks
delivery
TimeLine
tasks
tasks
tasks
delivery
tasks
tasks
tasks
tasks
tasks
delivery


Designing a Delivery
TaskCycle
Fri Mon Tue Wed Thu Fri Mon Tue Wed Thu Fri
available time:
36 hr gross
24 hr plannable
Delivery to Stakeholders
deliv to main team
Delivery to Stakeholders
Serge (ProjLead)
MbWA 3
Planning nxt wk 3
Work for deliv 4
- 6
- 2
- 1
- 5
Total 24
Gregory
Draft design 6
Finish design 6
Work for deliv 3
- 1
- 2
- 2
- 3
- 5
- 6
XMLa 4
XMLb 4
Total 42
Gregory (later)
Draft design 0
Finish design 0
...
Repair deliv 0
...
Zero Defects Attitude
Jerome
XMLa 3
XMLb 3
...

## TaskCycle Exercise




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

EvoShort- 2010 74

## Agile, but will we be on time ?

- Organizing the work in very short cycles
- Making sure we are doing the right things
- Doing the right things right
- Continuously optimizing (what not to do)
- So, we already work more efficiently

but ...

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

EvoShort- 2010 75

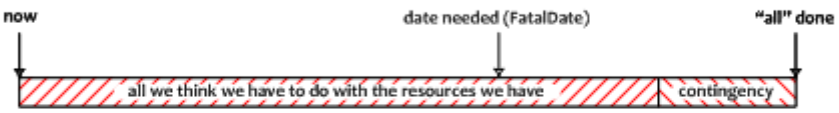
# Evolutionary Planning

## TimeLine

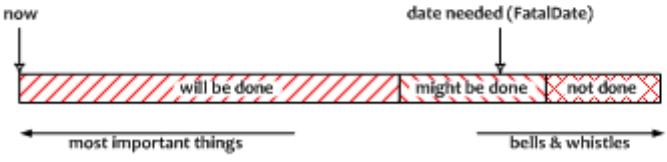


## TimeLine

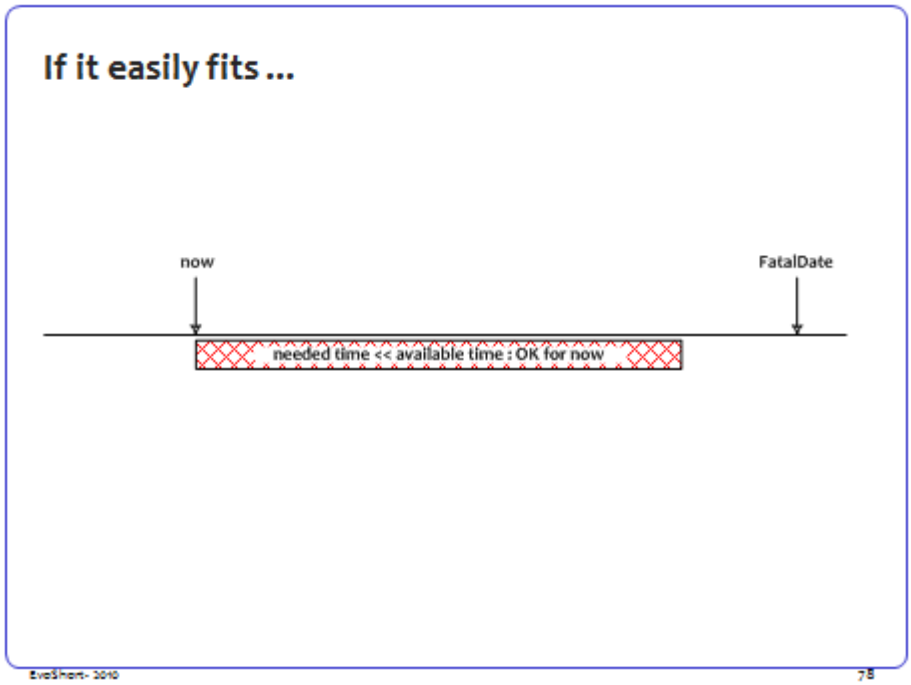
What the customer wants, he cannot afford

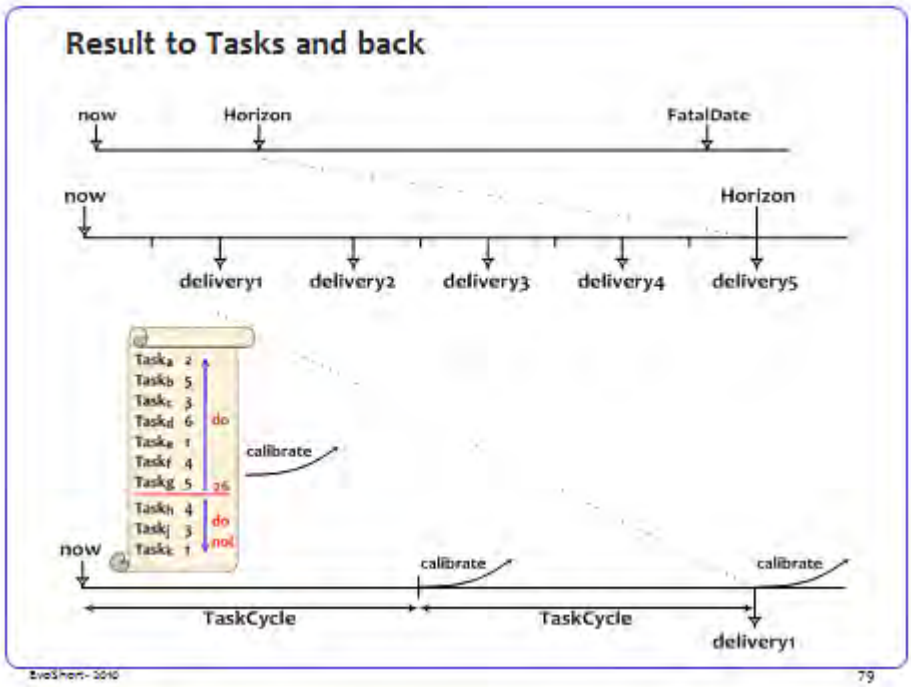


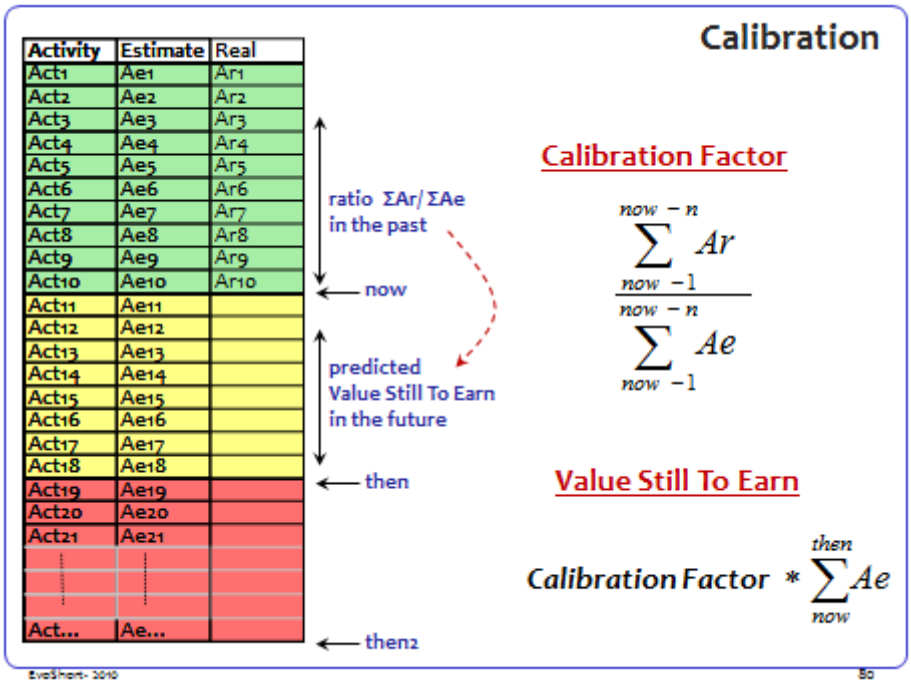

Standard Projects

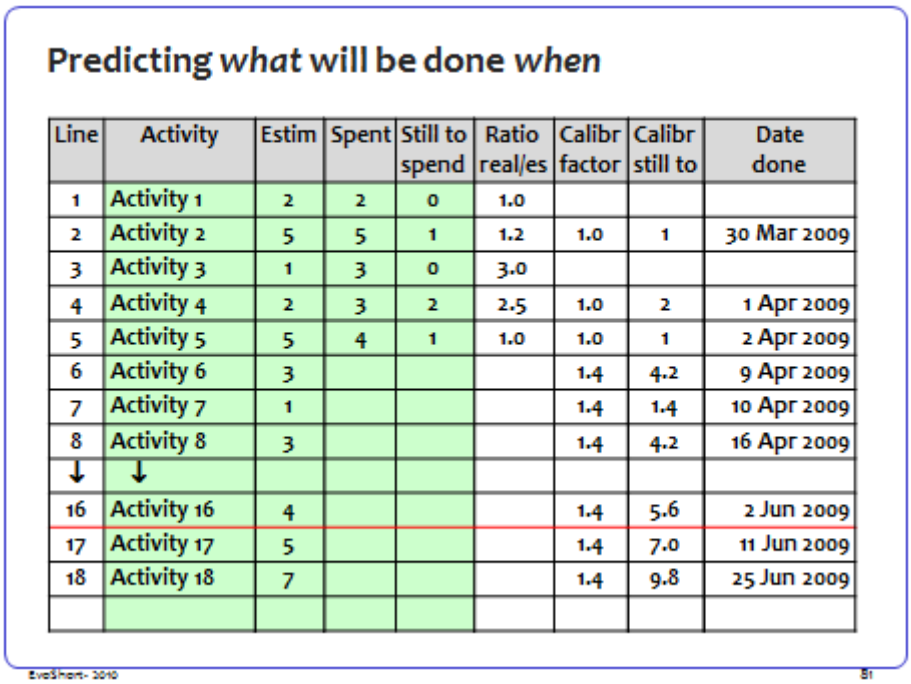



Evo


If it easily fits ...
now
FatalDate
needed time << available time : OK for now
EvoShort- 2010
78


Result to Tasks and back
now
Horizon
FatalDate
now
Horizon
delivery1
delivery2
delivery3
delivery4
delivery5
Taska 2
Taskb 5
Taskc 3
Taskd 6
Taske 1
Taskf 4
Taskg 5
Taskh 4
Taskj 3
Taskk 1
do
do not
calibrate
now
calibrate
calibrate
TaskCycle
TaskCycle
delivery1
EvoShort- 2010
79


Calibration
Activity
Estimate
Real
Act1 Ae1 Ar1
Act2 Ae2 Ar2
Act3 Ae3 Ar3
Act4 Ae4 Ar4
Act5 Ae5 Ar5
Act6 Ae6 Ar6
Act7 Ae7 Ar7
Act8 Ae8 Ar8
Act9 Ae9 Ar9
Act10 Ae10 Ar10
Act11 Ae11
Act12 Ae12
Act13 Ae13
Act14 Ae14
Act15 Ae15
Act16 Ae16
Act17 Ae17
Act18 Ae18
Act19 Ae19
Act20 Ae20
Act21 Ae21
Act... Ae...
ratio ΣAr/ΣAe in the past
now
predicted Value Still To Earn in the future
then
then2
Calibration Factor
$\frac{\sum_{now-1}^{now-n} Ar}{\sum_{now-1}^{now-n} Ae}$
Value Still To Earn
Calibration Factor * $\sum_{now}^{then} Ae$
EvoShort- 2010
80

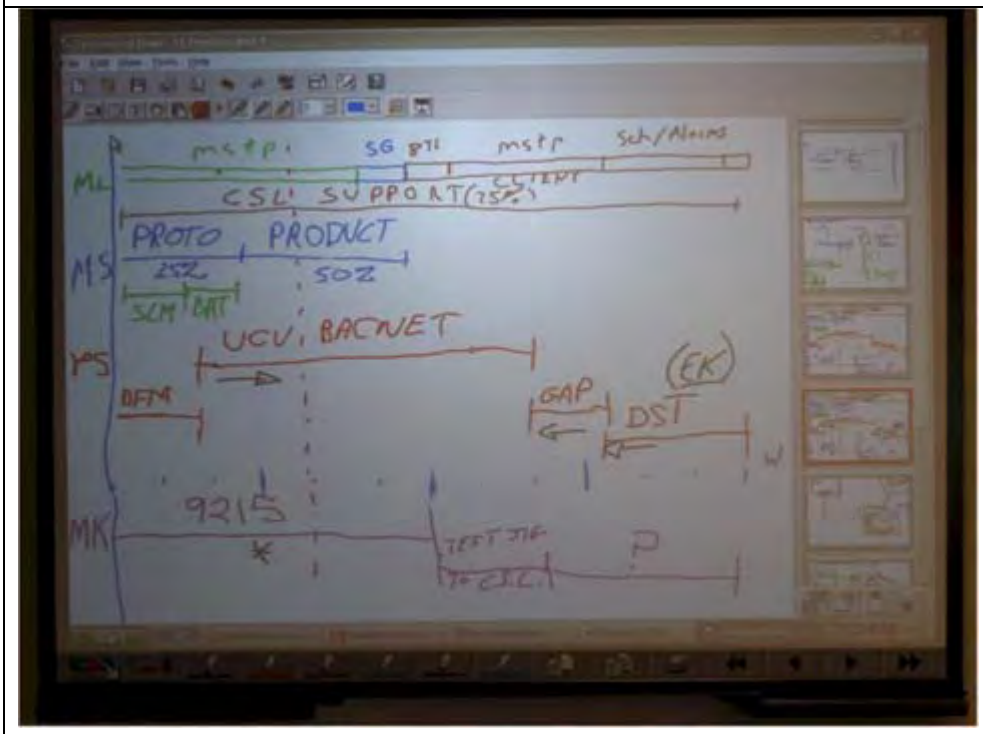

Predicting what will be done when
Line
Activity
Estim
Spent
Still to spend
Ratio real/es
Calibr factor
Calibr still to
Date done
1 Activity 1 2 2 0 1.0
2 Activity 2 5 5 1 1.2 1.0 1 30 Mar 2009
3 Activity 3 1 3 0 3.0
4 Activity 4 2 3 2 2.5 1.0 2 1 Apr 2009
5 Activity 5 5 4 1 1.0 1.0 1 2 Apr 2009
6 Activity 6 3 1.4 4.2 9 Apr 2009
7 Activity 7 1 1.4 1.4 10 Apr 2009
8 Activity 8 3 1.4 4.2 16 Apr 2009
16 Activity 16 4 1.4 5.6 2 Jun 2009
17 Activity 17 5 1.4 7.0 11 Jun 2009
18 Activity 18 7 1.4 9.8 25 Jun 2009
EvoShort- 2010
81

## Product/Portfolio/Resource Management

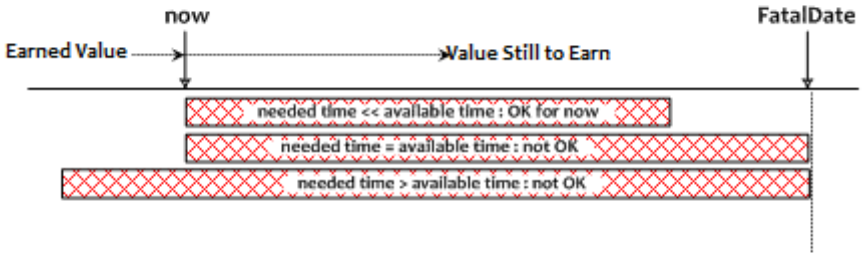
- Current Program/Portfolio/Resource Management is based on hope
- More a game than management

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

EvoShort- 2010 81



## What do we do if we see we won't make it on time?




- If it doesn't fit ... count backwards

EvoShort- 2010 84

## Deceptive options

- Hoping for the best (fatalistic)
- Going for it (macho)
- Working Overtime (fooling ourselves)
- Moving the deadline
  - Parkinson's Law
    - Work expands to fill the time for its completion
  - Student Syndrome
    - Starting as late as possible, only when the pressure of the FatalDate is really felt

EvoShort- 2010 85

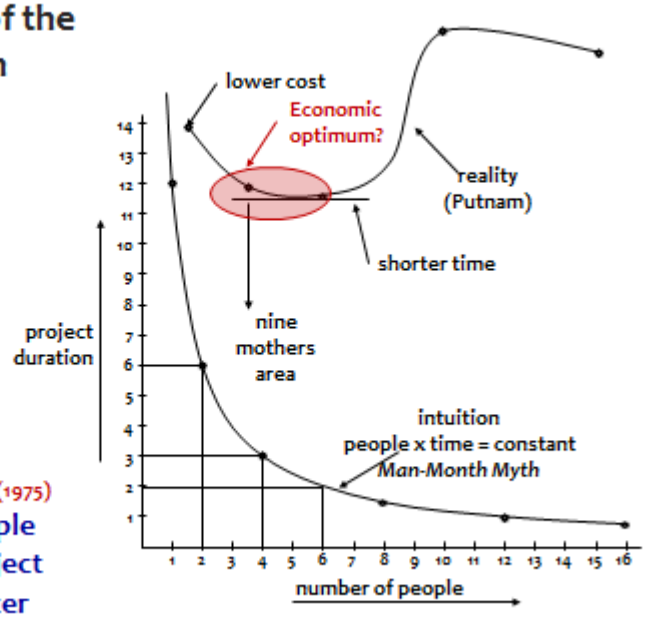
## Adding people to a late project ...

**makes it later**

(Brooks' Law, 1975)

EvoShort- 2010 86

## The Myth of the Man-Month




**Brooks' Law** (1975)
Adding people to a late project makes it later

EvoShort- 2010 87

## Saving time

Continuous elimination of waste

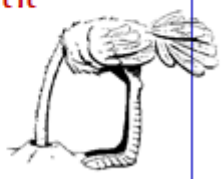
We don't have enough time, but we can save time *without negatively affecting the Result !*

- Efficiency in *what (why, for whom)* we do - doing the right things
  - Not doing what later proves to be superfluous
- Efficiency in *how* we do it - doing things differently
  - The product
    - Using proper and most efficient solution, instead of the solution we always used
  - The project
    - Doing the same in less time, instead of immediately doing it the way we always did
  - Continuous improvement and prevention processes
    - Constantly learning doing things better and overcoming bad tendencies
- Efficiency in *when* we do it - right time, in the right order
- TimeBoxing - much more efficient than FeatureBoxing

EvoShort- 2010 88

## TimeLine

- The TimeLine technique doesn't solve our problems
- It helps to expose the real status early and continuously
- Instead of accepting the undesired outcome, *we do something about it*
- The earlier we know, the more we can do about it
- We start saving time from the very beginning
- We can save a lot of time in any project, *while producing a better outcome*



If, and only if, we are serious about time !

EvoShort- 2010 89

## Estimation techniques used

- **Just-enough estimation** (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- **Changing from optimistic to realistic predictions**
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- **$o^{th}$ order estimations** (ball-park figures)
  - For decision-making in Business Case and Design
- **Simple Delphi**
  - For estimating longer periods of time in TimeLine
  - For duration of several (15 or more) elements of work
- **Simpler Delphi** (just enough !)
  - Same, but for quicker insight
  - Recently added by practice
- **Calibration**
  - Coarse metrics provide accurate predictions
- ***Doing something about it*** (if we don't like what we see)
  - Taking the consequence
  - Saving time

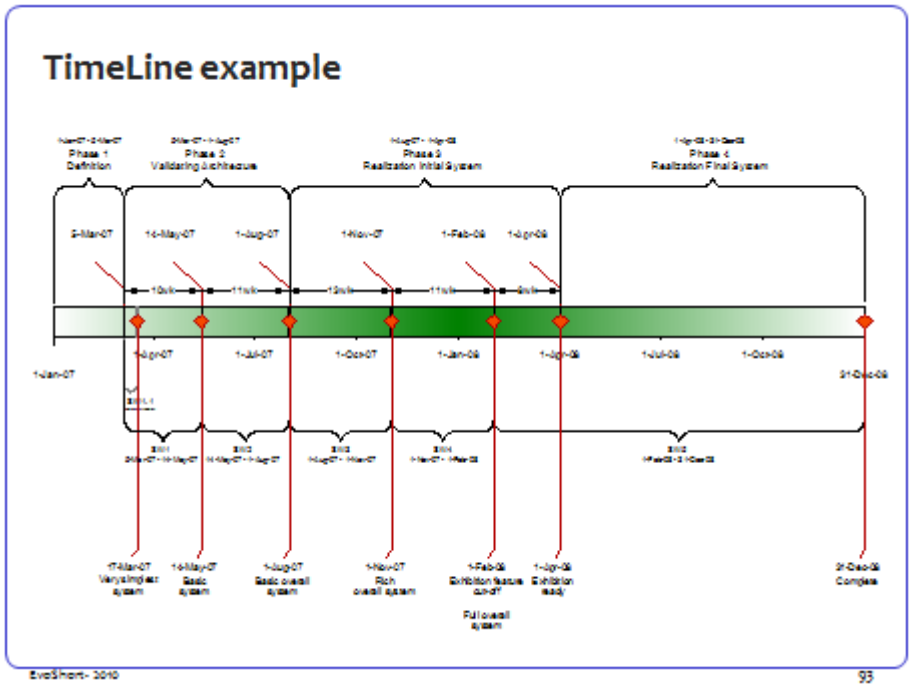
EvoShort- 2010 90

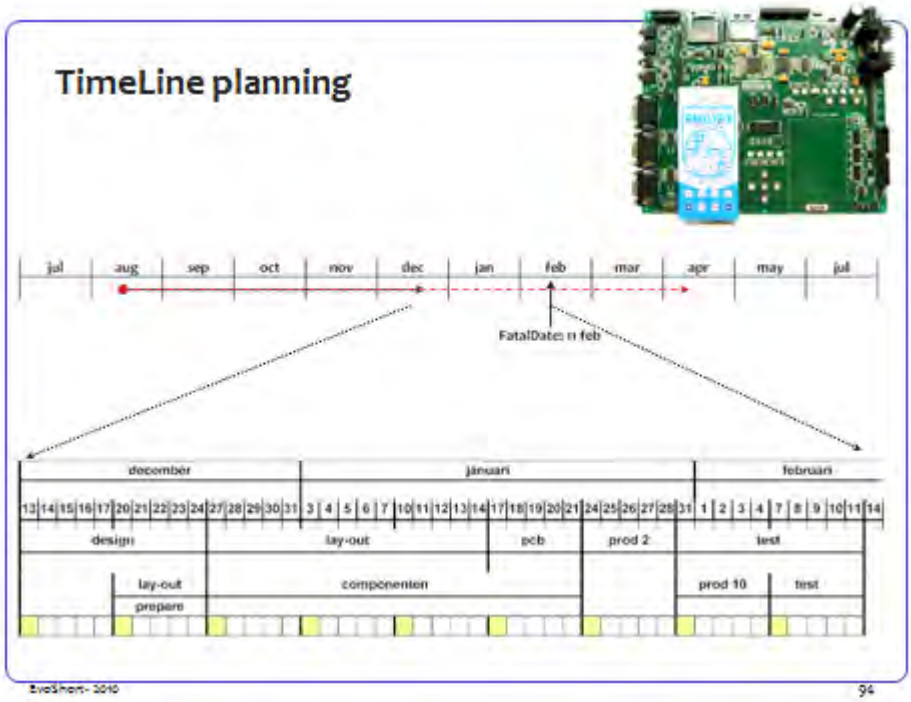
# TimeLine examples



## If we add something…

If we add something, something else will not be done


TimeLine example
Phase 1
Definition
Phase 2
Validating Architecture
Phase 3
Realization Initial System
Phase 4
Realization Final System
2-Mar-07
14-May-07
1-Aug-07
1-Nov-07
1-Feb-08
1-Apr-08
1-Jan-07
1-Apr-07
1-Jul-07
1-Oct-07
1-Jan-08
1-Apr-08
1-Jul-08
1-Oct-08
31-Dec-08
17-Mar-07
1-May-07
1-Aug-07
1-Nov-07
1-Feb-08
1-Apr-08
Full overall system
EvoShort- 2010
93


TimeLine planning
jul
aug
sep
oct
nov
dec
jan
feb
mar
apr
may
jul
FatalDate: 1 feb
december
januari
februari
design
lay-out
pcb
prod 2
test
lay-out
prepare
componenten
prod 10
test
EvoShort- 2010
94

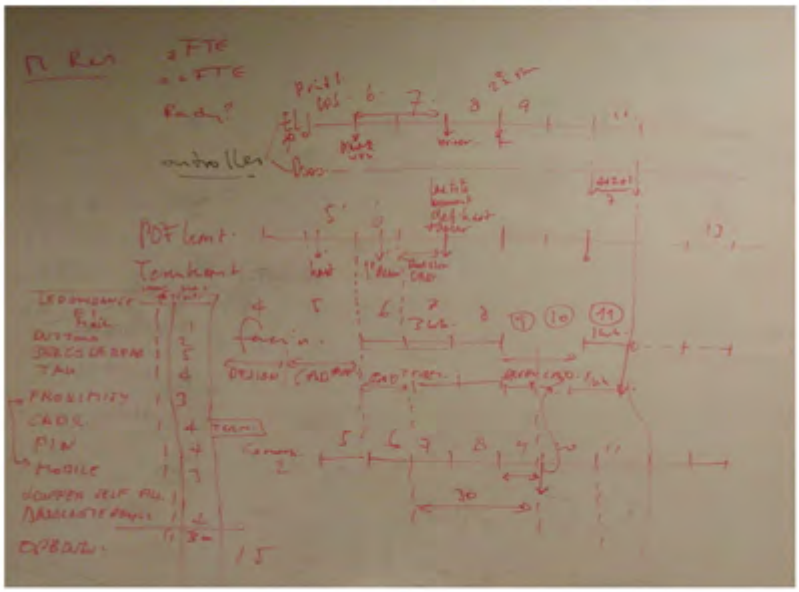
## TimeLine exercise example

- Preparing for student exams



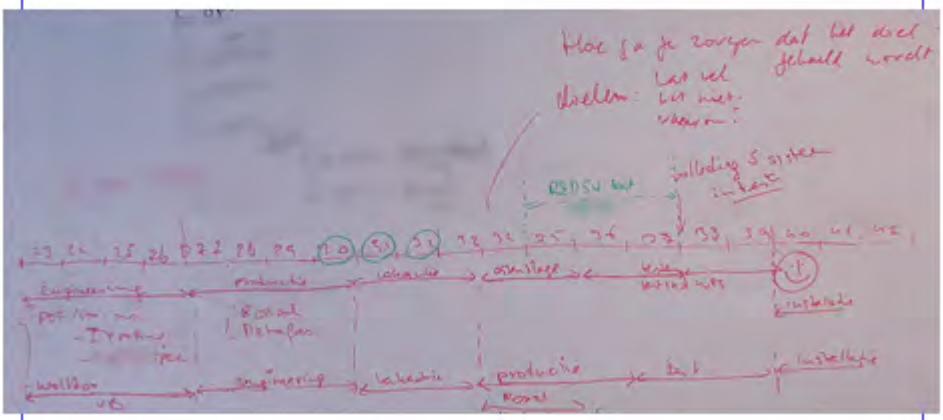
## What we did

## Whiteboard TimeLine Planning

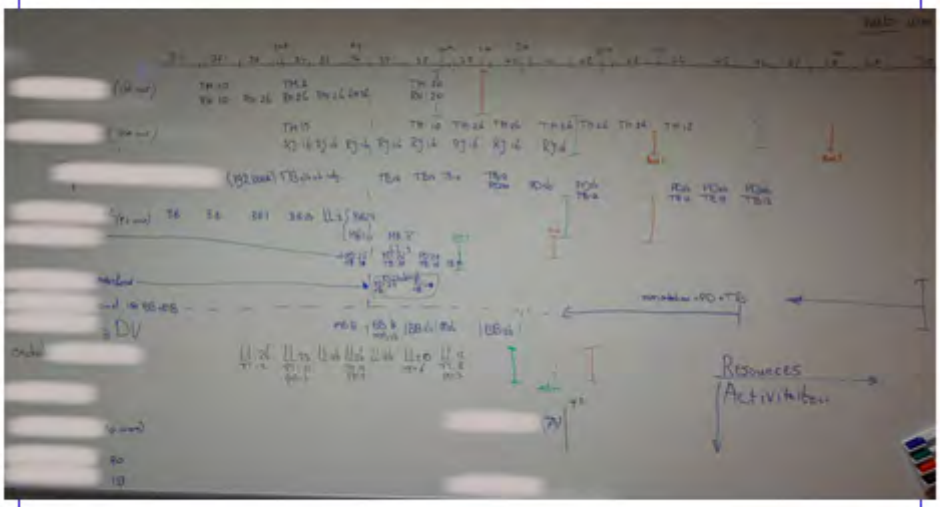




## Whiteboard TimeLine Planning



## Whiteboard TimeLine Resource Planning




EvoShort- 2010 99

## Help!
## We have a QA problem!

- Large stockpile of modules to test
  (hardware, firmware, software)
- You shall do Full Regression Tests
- Full Regression Tests take about 15 days each
- Too few testers ("Should we hire more testers?")
- Senior Tester paralyzed
- Can we do something about this?

EvoShort- 2010 100


Do you think you can help us ?
Act
What are we going to do differently?
We are going to do it differently!
Plan
What to achieve
How to achieve it
Check
Is the Result according to Plan?
Is the way we achieved the Result according to Plan?
Do
Carry out the Plan
EvoShort- 2010
101

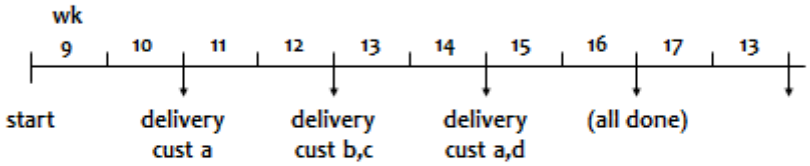

In stead of complaining about a problem ...
(Stuck in the Check-phase)
Let's do something about it !
(Moving to the Act-phase)
EvoShort- 2010
103

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

| Line | Activity | Estim | Alter native | Junior tester | Devel opers | Customer | Will be done (now=22Feb) |
|---|---|---|---|---|---|---|---|
| 1 | Package 1 | 17 | 2 | 17 | 4 | HT | |
| 2 | Package 2 | 8 | 5 | | 10 | Chrt | |
| 3 | Package 3 | 14 | 7 | 5 | 4 | BMC | |
| 4 | Package 4 (wait for feedback) | 11 | | | | McC? | |
| 5 | Package 5 | 9 | 3 | | 5 | Ast | |
| 6 | Package 6 | 17 | 3 | 10 | 10 | ? | |
| 7 | Package 7 | 4 | 1 | | 3 | Cli | |
| 8 | Package 8.1 | 26 | 1 | | | Sev | |
| 9 | Package 8.2 | 1 | 1 | | | ? | |
| 10 | Package 8.3 | 1 | 1 | | | Chrt | 24 Feb |
| 11 | Package 8.4 | 1 | 1 | | | Chrt | |
| 12 | Package 8.5 | 1.1 | 1.1 | | | Yet | 28 Feb |
| 13 | Package 8.6 | 3 | 3 | | | Yet | 24 Mar |
| 14 | Package 8.7 | 0.1 | 0.1 | | | Cli | After 8.5 OK |
| 15 | Package 8.8 | 18 | 18 | | | Ast | |
| | totals | 106 | 47 | 32 | 36 | | |

EvoShort- 2010 103

## TimeLine




**Selecting the priority order of customers to be served**

- "We'll have a solution at that date ... Will you be ready for it ?"
  An other customer could be more eagerly waiting
- Most promising customers

EvoShort- 2010 104

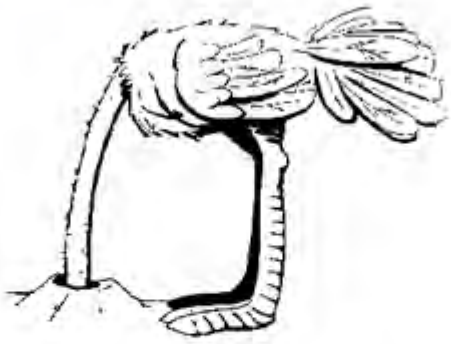
## Result

- Tester empowered
- Done in 9 weeks
- So called "Full Regression Testing" was redesigned
- Customers systematically happy and amazed
- Kept up with development ever since
- Increased revenue

Recently:

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

EvoShort- 2010 105



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

EvoShort- 2010 106

# Some extra



## 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

## Interrupts

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


- In case of interrupt, use interrupt procedure

EvoShort- 2010 109

## 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:

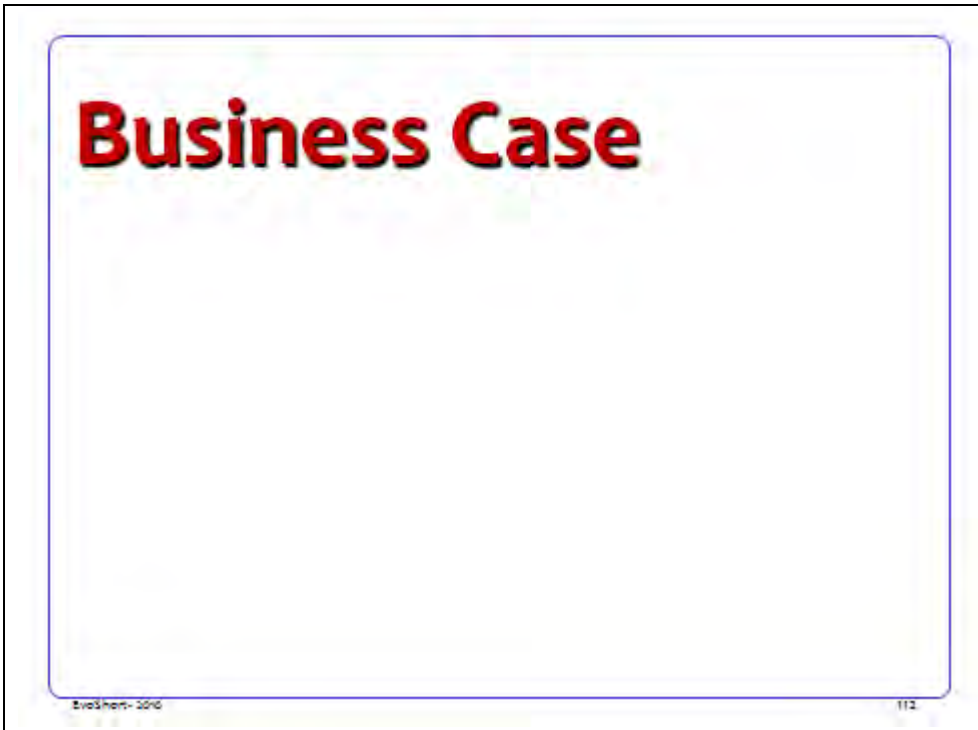
1. Define the expected Results of the new Task properly
2. Estimate the time needed to perform the new Task, to the level of detail really needed
3. Go to your task planning tool (many projects use the ETA tool)
4. Decide which of the planned Tasks is/are going to be sacrificed (up to the number of hours 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.

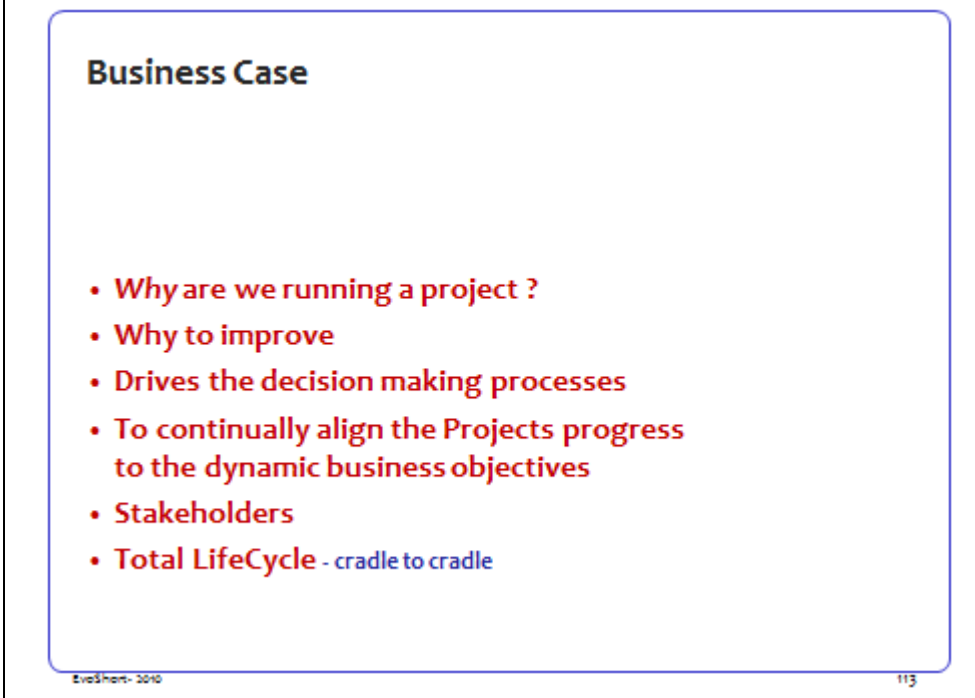
EvoShort- 2010 110

**Niels Malotaux**

# Predictable Projects

## Delivering the Right Result at the Right Time


Business Case
EvoShort- 2010
112


Business Case
• Why are we running a project ?
• Why to improve
• Drives the decision making processes
• To continually align the Projects progress to the dynamic business objectives
• Stakeholders
• Total LifeCycle - cradle to cradle
EvoShort- 2010
113

## Higher Productivity

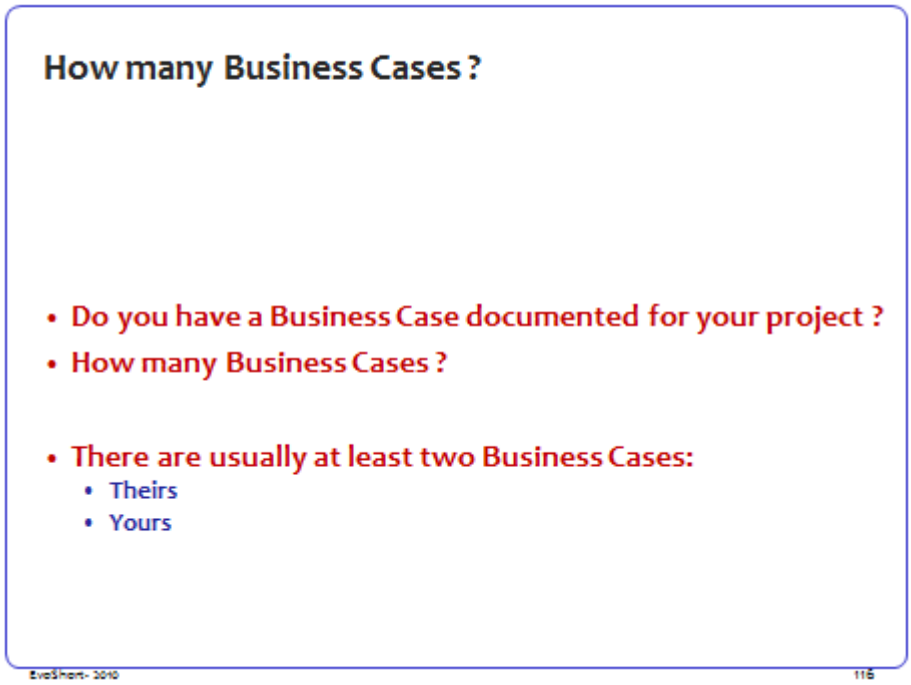
- All functionality we produce *does already exist*
- The real reason for running our projects is creating *better performance*
- Types of improvement:
  - Less loss
  - More profit
  - Doing the same in shorter time
  - Doing more in the same time
  - Being happier than before
- In short: *Adding Value*



## Return on Investment


**Return on Investment (RoI)**

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


How many Business Cases ?
• Do you have a Business Case documented for your project ?
• How many Business Cases ?
• There are usually at least two Business Cases:
• Theirs
• Yours
EvoShort- 2010
116

# Stakeholders & Requirements



## Stakeholders are people


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

## What are the Requirements for a Project ?

- Requirements are what the Stakeholders require

but for a project ...

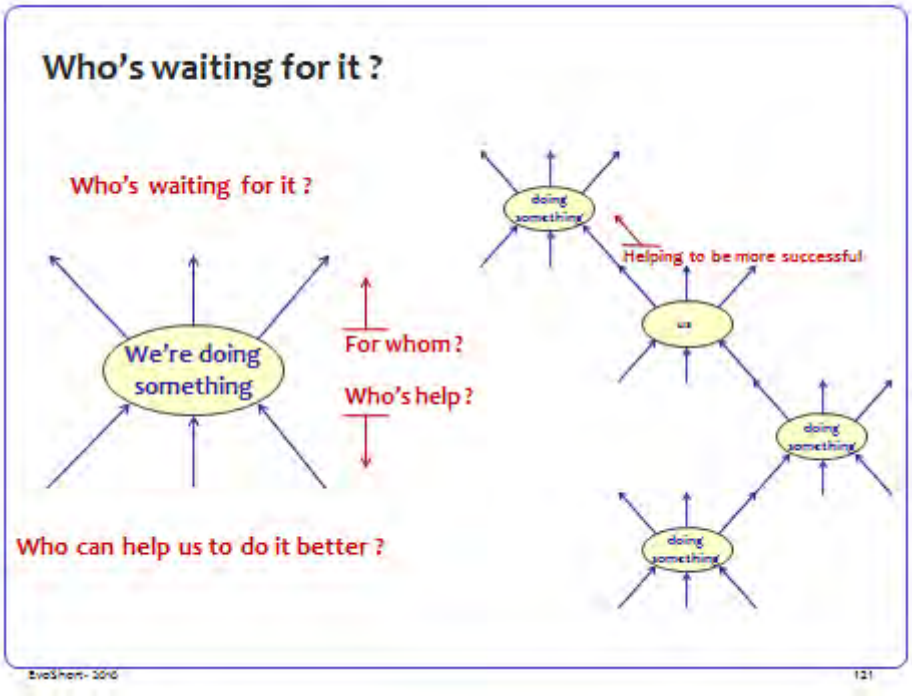
- Requirements are the set of stakeholder needs that the project is *planning* to *satisfy*

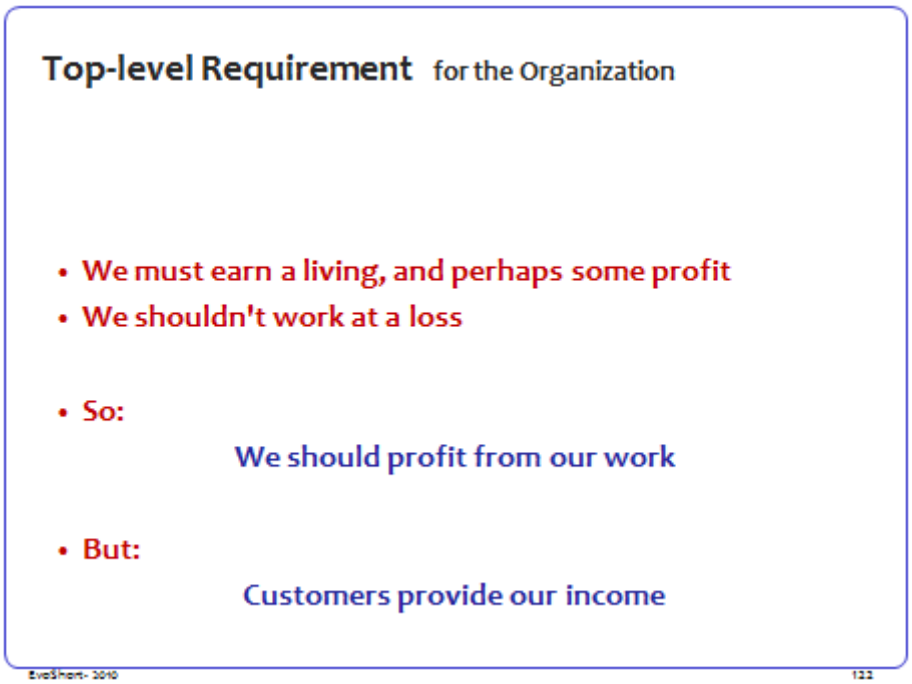
- The set of Stakeholders doesn't change much
- Do you have a checklist of possible Stakeholders ?



## 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?


Who's waiting for it ?
Who's waiting for it ?
We're doing something
For whom ?
Who's help ?
Who can help us to do it better ?
doing something
Helping to be more successful
us
doing something
doing something
EvoShort- 2010
121


Top-level Requirement for the Organization
• We must earn a living, and perhaps some profit
• We shouldn't work at a loss
• So:
We should profit from our work
• But:
Customers provide our income
EvoShort- 2010
122

## Top-level Requirement for the Project

- Providing the customer with
  - what he needs
  - at the time he needs it
  - to be satisfied
  - to be more successful than he was without it
- Constrained by (win - win)
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

EvoShort- 2010 123

## Somebody said the requirements should be *SMART*

- Do we have documented requirements ?
- Are they SMART ?

- S Specific
- M Measurable
- A Attainable
- R Realisable
- T At the right Time (some say: Traceable)

EvoShort- 2010 124

## Requirements with Planguage

ref Tom Gilb

Definition:

Specific
Measurable

RQ27: Speed of Luggage Handling at Airport
Scale: Time between <arrival of airplane> and first luggage on belt
Meter: <measure arrival of airplane>, <measure arrival of first luggage on belt>, calculate difference

Benchmarks (Playing Field):

Attainable

Past: 2 min [minimum, 2009], 8 min [average, 2009], 83 min [max, 2009]
Current: < 4 min [competitor y, Jan 2010] ← <who said this?>, <Survey Feb2010>
Record: 57 sec [competitor x, Jan 2010]
Wish: < 2 min [2011Q3] ← CEO, 19 Feb 2010, <document...>

Requirements: Time

Realizable

Must: < 10 min [99%, Q4] ←SLA
Must: < 15 min [100%, Q4, Schiphol] ←SLA
Goal: < 15 min [99%, Q2], < 10 min [99%, Q3], < 5 min [99%, Q4] ←marketing

EvoShort- 2010 125

## Requirements carved in stone ?

- *We* don't know the real requirements
- *They* don't know the real requirements
- Together we'll 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

EvoShort- 2010 126

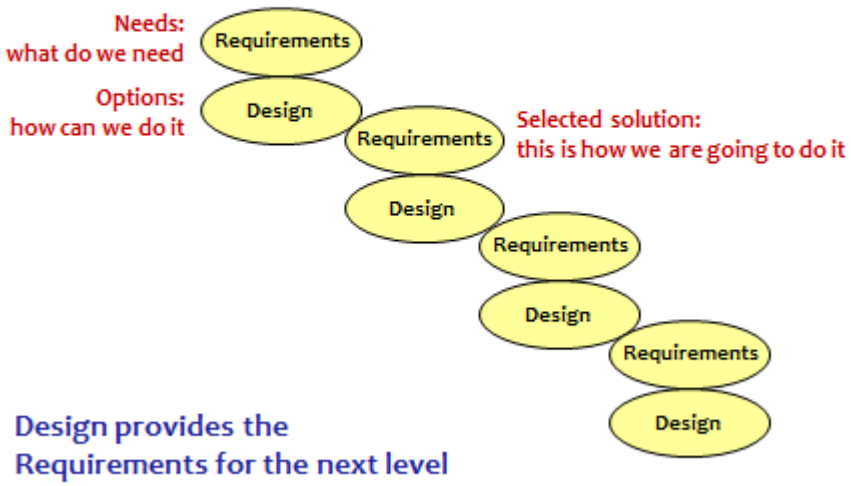
## 5 times "Why?" technique

First develop the problem interdisciplinarily, then develop the solution and then the implementation

- Freud and Jung:
  - Problems are in our sub-consciousness
  - Solutions pop up
  - Solutions are how people tell their problems
- What's your problem ?
  - If there's no problem, we don't have to do something
- Within 5 times "Why?"
  we usually come down to the real problem to solve
  - Otherwise we will be perfectly solving the wrong problem

EvoShort- 2010 127

## No Design in the requirements, but ...




Design provides the
Requirements for the next level

EvoShort- 2010 128

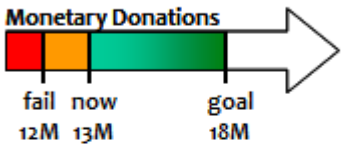
## Requirements Case

- Organization collecting online giving for charities
- CEO: "Improve website to increase online giving for our 'customers' (charities)"
- Increasing market share for online giving
- Budget: 1M€ - 10 months
- Show results fast

Ref Ryan Shriver
ACCU Overload Feb 2009

EvoShort- 2010 129

## Objective: Monetary Donations

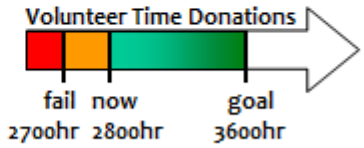



| | |
|---|---|
| Name | Monetary Donations |
| Scale | Euro's donated to non-profits through our website |
| Meter | Monthly Donations Report |
| Fail | 12M |
| Now | 13M [2008] ←Annual Report 2008 |
| Goal | 18M [2009] |

Ref Ryan Shriver
ACCU Overload Feb 2009

EvoShort- 2010 130

## Objective: Volunteer Time (Natura) Donations




Name Volunteer Time Donations

Scale Hours donated to non-profits through our website

Meter Monthly Donations Report

Fail 2700 hr

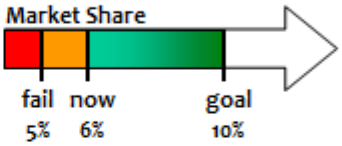
Now 2800 hr [2008] ←Annual Report 2008

Goal 3600 hr [2009]

Ref Ryan Shriver
ACCU Overload Feb 2008

EvoShort- 2010 131

## Goal: Market Share




Name Market Share

Scale Market Share %% online giving

Meter Quarterly Industry Report

Fail 5%

Now 6% [Q1-2009] ←Quarterly Industry Report

Goal 10% [Q1-2010]

Ref Ryan Shriver
ACCU Overload Feb 2008

EvoShort- 2010 132

## Impact Estimation example

| Impact Estimation | Monthly Donations | Facebook integration | Image & video uploads | Total effect for requirement |
|---|---|---|---|---|
| €€ donations 13M€ → 18M€ | 80% ±30% | 30% ±30% | 50% ±20% | 160% ±80% |
| Time donations 2800hr→3600hr | 10% ±10% | 50% ±20% | 80% ±20% | 140% ±50% |
| Market share 6% → 10% | 30% ±20% | 30% ±20% | 20% ±10% | 80% ±50% |
| Total effect per solution | 120% ±60% | 110% ±70% | 150% ±50% | 380% ±180% |
| Cost - money % of 1M€ | 30% ±10% | 20% ±10% | 50% ±20% | 100% ±40% |
| Cost - time % of 10 months | 40% ±20% | 20% ±10% | 50% ±20% | 110% ±50% |
| Total effect / money budget | 120/30 = 4 1.5 … 9 | 110/20 = 5.5 1.3 … 18 | 150/50 = 3 1.4 … 6.7 | |
| Total effect / time budget | 120/40 = 3 1 … 9 | 120/20 = 6 1.3 … 18 | 120/50 = 2.4 1.4 … 6.7 | |

Rot Ryan Shriver - ACCU Overload Feb 2009

EvoShort- 2010 133

# Architecture and Design



## Design is always a compromise

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

example:

- Performance ↑
- Budget (time, money) ↓

## Design and requirements

- Design:
  Finding the best compromise between the conflicting requirements

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

MoSCoW ?

EvoShort- 2010 136

## Design Process

- Collect obvious design(s)
- Search for *one* non-obvious design
- Compare the relative ROI of the designs
- Select the best compromise based on defined criteria
- Describe the selected design

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

EvoShort- 2010 137

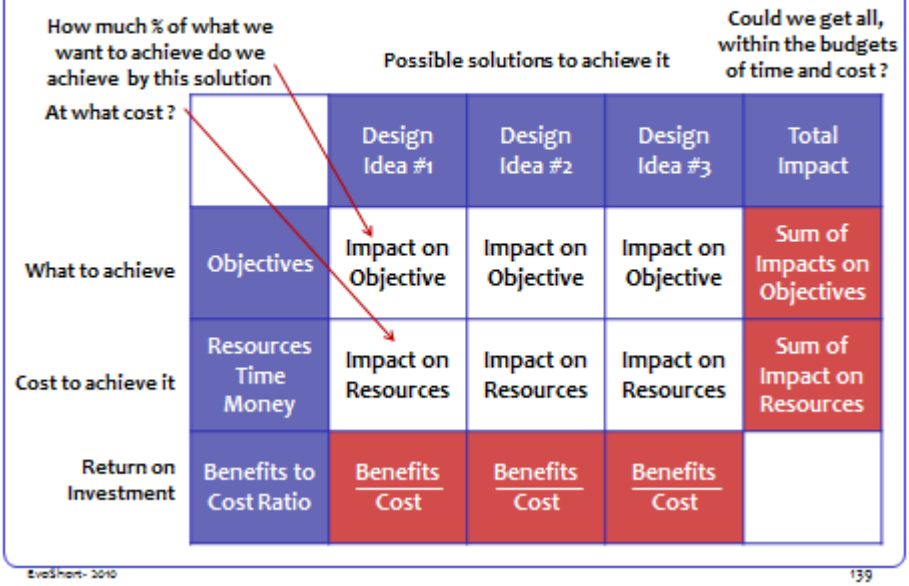
## Impact Estimation example

| Impact Estimation | Monthly Donations | Facebook integration | Image & video uploads | Total effect for requirement |
|---|---|---|---|---|
| €€ donations 13M€ → 18M€ | 80% ±30% | 30% ±30% | 50% ±20% | 160% ±80% |
| Time donations 2800hr→3600hr | 10% ±10% | 50% ±20% | 80% ±20% | 140% ±50% |
| Market share 6% → 10% | 30% ±20% | 30% ±20% | 20% ±10% | 80% ±50% |
| Total effect per solution | 120% ±60% | 110% ±70% | 150% ±50% | 380% ±180% |
| Cost - money % of 1M€ | 30% ±10% | 20% ±10% | 50% ±20% | 100% ±40% |
| Cost - time % of 10 months | 40% ±20% | 20% ±10% | 50% ±20% | 110% ±50% |
| Total effect / money budget | 120/30 = 4 1.5 … 9 | 110/20 = 5.5 1.3 … 18 | 150/50 = 3 1.4 … 6.7 | |
| Total effect / time budget | 120/40 = 3 1 … 9 | 120/20 = 6 1.3 … 18 | 120/50 = 2.4 1.4 … 6.7 | |

Ref Ryan Shriver - ACCU Overload Feb 2009

EvoShort- 2010 138

## Impact Estimation principle

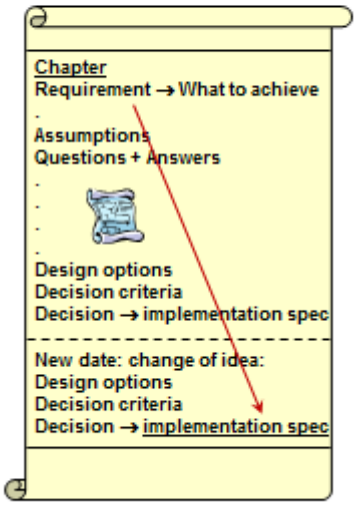



EvoShort- 2010 139

## DesignLog

(project level)

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

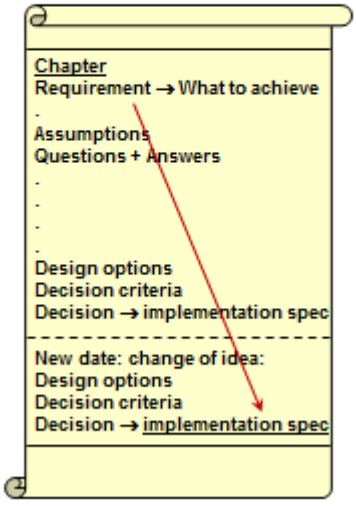



EvoShort- 2010 140

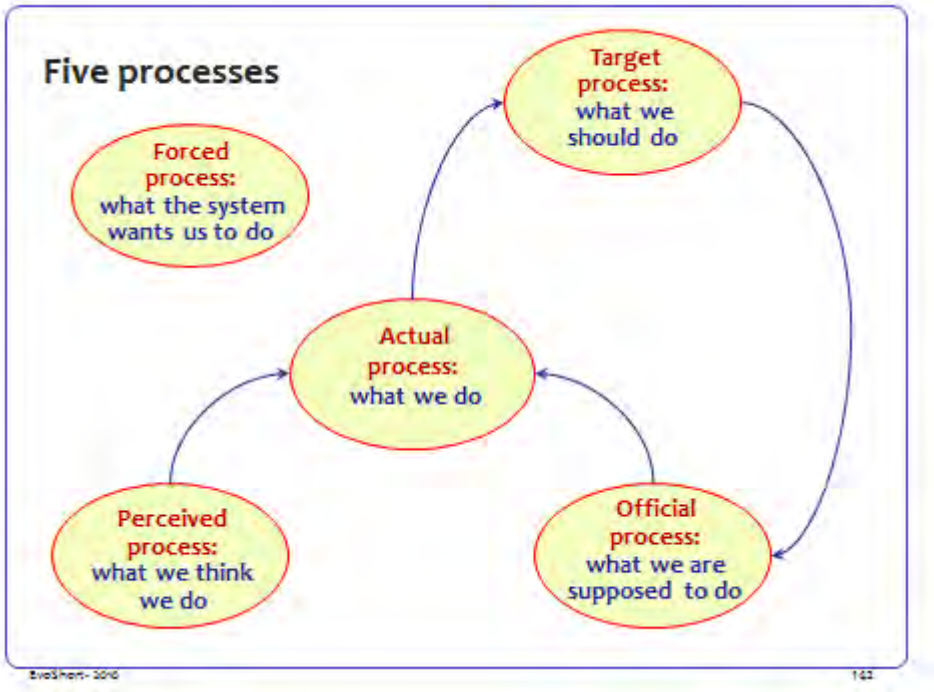
## ProcessLog

(department / organization level)

- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Graphics (drawings)
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Known techniques
  - Choices + reasoning :
    - If rejected: why?
    - If chosen: why?
- Rejected choices
- Final (current) choices
- Implementation



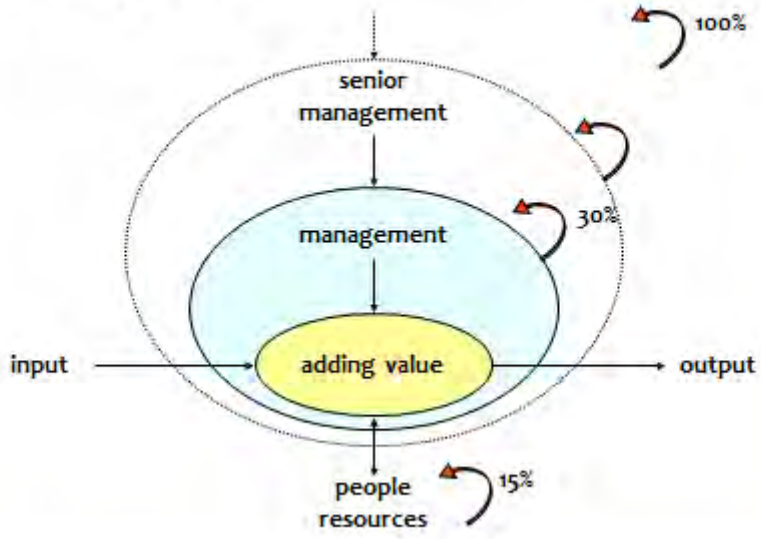

EvoShort- 2010 141


Five processes
Target process: what we should do
Forced process: what the system wants us to do
Actual process: what we do
Perceived process: what we think we do
Official process: what we are supposed to do

# Managers Tasks



## The managers task

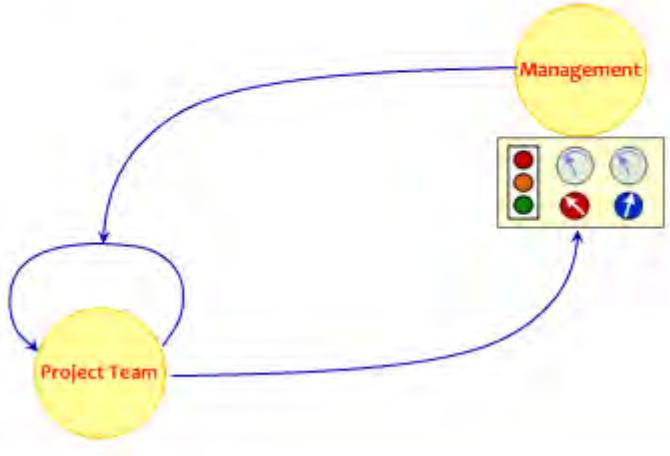


## Managers have to learn

- Managers *facilitate* their people to be successful
- Managers should be coaches
- Not police
- Managers have to understand the Evo approach

EvoShort- 2010 145

## Local Loop Principle

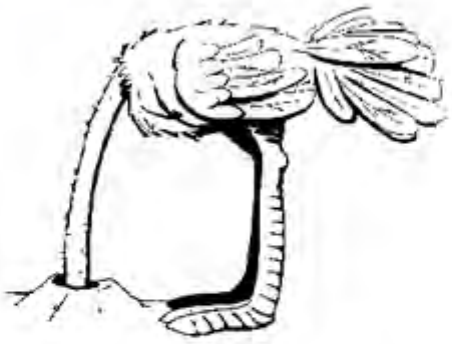



EvoShort- 2010 146

## 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

EvoShort- 2010 147



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

EvoShort- 2010 148

## More

www.malotaux.nl/Booklets

1 Evolutionary Project Management Methods (2001)
Issues to solve, and first experience with the Evo Planning approach

2 How Quality is Assured by Evolutionary Methods (2004)
After a lot more experience: rather mature Evo Planning process

3 Optimizing the Contribution of Testing to Project Success (2005)
How Testing fits in

3a Optimizing Quality Assurance for Better Results (2005)
Same as Booklet 3, but for non-software projects

4 Controlling Project Risk by Design (2006)
How the Evo approach solves Risk by Design (by process)

5 TimeLine: How to Get and Keep Control over Longer Periods of Time (2007)
Replaced by Booklet 7, except for the step-by-step TimeLine procedure

6 Human Behavior in Projects (APCOSE 2008)
Human Behavioral aspects of Projects

7 How to Achieve the Most Important Requirement (2008)
Planning of longer periods of time, what to do if you don't have enough time

8 Help! We have a QA Problem! (2009)
Use of TimeLine technique: How we solved a 6 month backlog in 9 weeks

RS Measurable Value with Agile (Ryan Shriver - 2009)
Use of Evo Requirements and Prioritizing principles

www.malotaux.nl/nrm/Insp
Inspection pages

EvoShort- 2010 149

## What now ?

EvoShort- 2010 150

# Early Reviews & Inspections

EvoShort- 2010 151

## Inevitable consequence

People make mistakes

We are people

If we do something, we introduce problems

Repair of problems costs exponentially more if found later

So, when to solve the problems? Immediately after making the mistake, or even preferably: by *preventing* mistakes

EvoShort- 2010 152

## Where do we make mistakes?

| | |
|---|---|
| • Wish specification | Thank you, nice input |
| • Business Case | Why are we doing it |
| • Requirements | What the project agrees to satisfy |
| • DesignLog | Selecting the 'optimum' compromise and how we arrived at this decision |
| • Specification | This is how we are going to implement it |
| • Implementation | Code, schematics, plans, procedures, hardware, documentation, training |
| • Process Log | Describing how and why we arrived at which current practices |

EvoShort- 2010 153

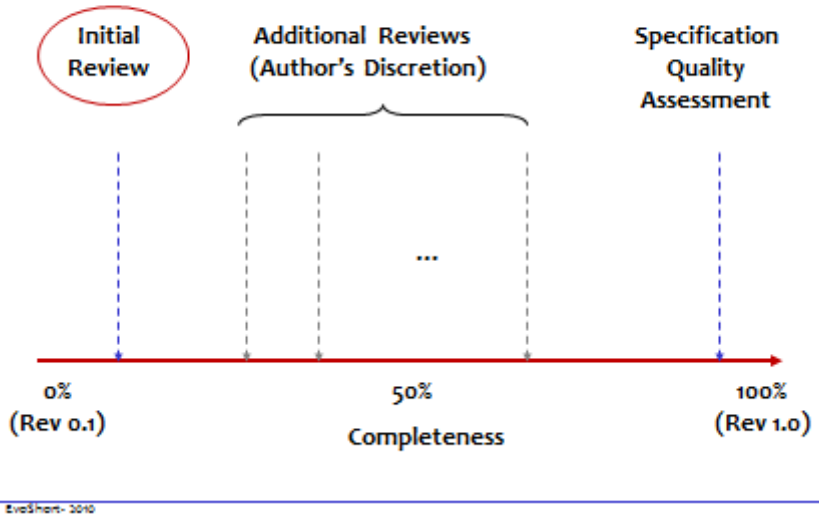
16 page
Inspection
Manual

www.malotaux.nl/nrm/pdf/InspManual.pdf


EvoShort- 2010 154

## Early Inspection

Prevention costs less than Repair




E8 EvoShort- 2010 155

## Initial Review

**Purpose:** Locating mistakes and tendencies that could lead to injecting major defects if not corrected

**When:** As soon as the author has completed a small representative portion of the specification, typically a few pages or 600-1200 words (e.g. few requirements)

**Who:** Individual or small team (1 or 2)
- Expertise in the subject matter
- Expertise in generic principles (such as requirements engineering, design, specific language)

**What:** Detailed review of the specification against rules and checklists for known error conditions and dangerous tendencies; formal inspection may be used

**Duration:** Because the sample is small, the initial review takes only 1-2 hr

The earlier it's reviewed, the more defects we can *prevent*

E8 EvoShort- 2010 156

## Initial Review Checklist

- ✓ Use a small team of experienced reviewers
- ✓ Schedule the review to minimize author waiting time
- ✓ Focus on issues that are or will cause major defects
- ✓ Avoid elements of style
- ✓ Be constructive at all times
- ✓ Focus on the work product, and never on the author
- ✓ Maintain confidentiality!
  The review is for the author's benefit

Reviewers: Your job is to make the author look like a hero

E8 EvoShort- 2010 157

## Case Study 1 - Situation

- Large e-business integrated application with 8 requirements authors, varying experience and skill
  - Each sent the first 8-10 requirements of estimated 100 requirements per author (table format, about 2 requirements per page including all data)
  - Initial reviews completed within a few hours of submission
  - Authors integrated the suggestions and corrections, then continued to work
  - Some authors chose additional reviews; others did not
  - Inspection performed on document to assess final quality level

E8 EvoShort- 2010 158

## Case Study 1 - Results

| | |
|---|---|
| Average major defects per requirement in initial review | 8 |
| Average major defects per requirement in completed document | 3 |

- Time investment: 26 hr
  - 12 hours in initial review (1.5 hrs per author)
  - About 8 hours in additional reviews
  - 6 hours in final inspection (2 hrs, 2 checkers, plus prep and debrief)
- Major defects prevented: 5 per requirement in ~750 total
- Saved 5 x 750 x 10 hr = 37500 hr / 3 = 12500 x $50 = $625000

E8 EvoShort- 2010 159

## Why Early Inspection Works

- Many defects are repetitive and can be prevented
  - Early review allows an author to get independent feedback on individual tendencies and errors
  - By applying early learning to the rest (~90%) of the writing process, many defects are prevented before they occur
  - Reducing rework in both the document under review and all downstream derivative work products

E8 EvoShort- 2010 160

## Case Study 2 - Situation

- A tester's improvement writing successive test plans:
  - Early Inspection used on an existing project to improve test plan quality
  - Test plan nearly "complete", so simulated Early Inspection
  - First round, inspected 6 randomly-selected test cases
  - Author notes systematic defects in the results, reworks the document accordingly (~32 hrs.)
  - Second round, inspected 6 more test cases; quality vastly improved
  - Test plan exits the process and goes into production
  - The author goes on to write another test plan on the next project...



## Case Study 2 - Results

| | |
|---|---|
| First round inspection | 6 major defects per test case |
| Second round | 0.5 major defects per test case |

- Time investment: 2 hours in initial review, 36 hours total in inspection, excluding rework (2 inspections, 4 hrs each, 4 checkers, plus preparation and debrief)
- Historically about 25% of all defects found by testing, were closed as "functions as designed", still 2-4 hrs spent on each
- This test plan yielded over 1100 software defects with only 1 defect (0.1%) closed as "functions as designed"
- Time saved on the project: 500 - 1000 hrs (25% x 1100 x 2-4 hrs )

Defect Prevention in action: First inspection of this tester's next test plan: 0.2 major defects per test case

## Early Detection vs. Prevention

Denise Leigh (Sema group, UK), British Computer Society address, 1992:

An eight-work-year development, delivered in five increments over nine months for Sema Group (UK), found:

- 3512 defects through inspection
- 90 through testing
- and 35 (including enhancement requests) through product field use

After two evolutionary deliveries, *unit testing of programs was discontinued because it was no longer cost-effective*

Nice job! Early detection has big benefits - BUT...

How many of the 3512 defects found in end-of-line inspections could have been completely prevented by Early Inspection?

Cost-effective defect prevention is the bottom line

E8 EvoShort- 2010 163

## Schedule

| October | Tue 12 |
|---|---|
| 09:00~10:30 | 1:30 |
| break | 0:10 |
| 10:40~11:40 | 1:00 |
| break | 0:10 |
| 11:50~12:50 | 1:00 |
| lunch | 0:40 |
| 13:30~14:30 | 1:00 |
| break | 0:10 |
| 14:40~15:40 | 1:00 |
| break | 0:10 |
| 15:50~16:50 | 1:00 |

EvoShort- 2010 164