INCOSE-IL - Herzlia
9 March 2011

Niels Malotaux

How Proactive Systems Engineers can realize Predictable Projects

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 electronic 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.

N R Malotaux
Consultancy

Niels Malotaux
project coach

Bongerdlaan 53
3723 VB Bilthoven
The Netherlands
tel +31-30-228 88 68
fax +31-30-228 88 69
mob +31-655 753 604
niels@malotaux.nl
www.malotaux.nl

Result Management

How Proactive Systems Engineers can realize Predictable Projects

Niels Malotaux

N R Malotaux
Consultancy

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

1

Niels Malotaux

Result Management

- Project Coach
  - Evolutionary Project Management (Evo)
  - Requirements Engineering
  - Reviews and Inspections
  - Dependability (Systems that simply work)

Helping projects and organizations to become predictable and deliver successfully in much shorter time

2

Predictable Projects ?

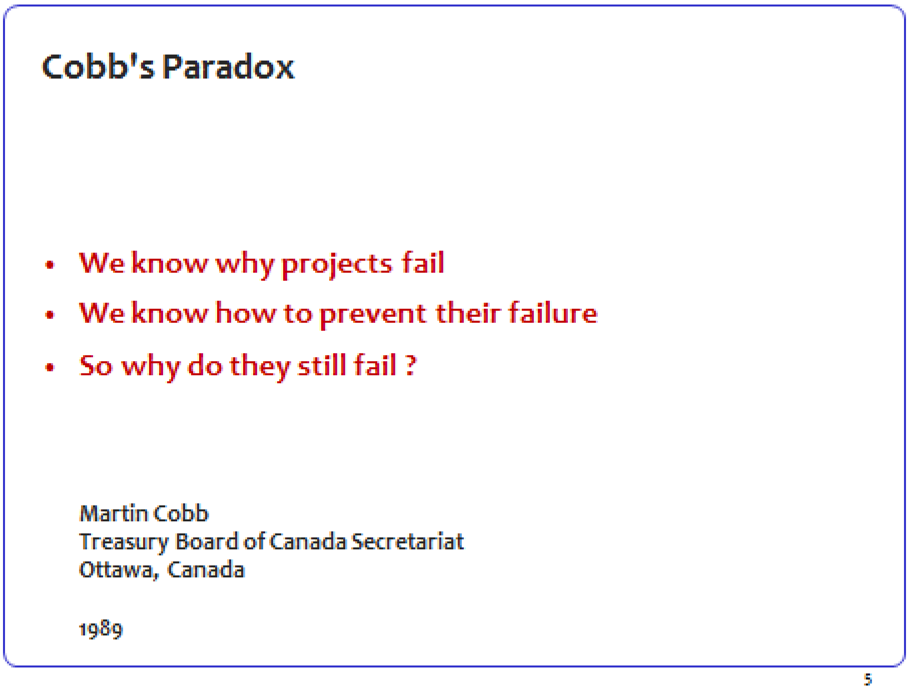
- Any problems with projects ?

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

4

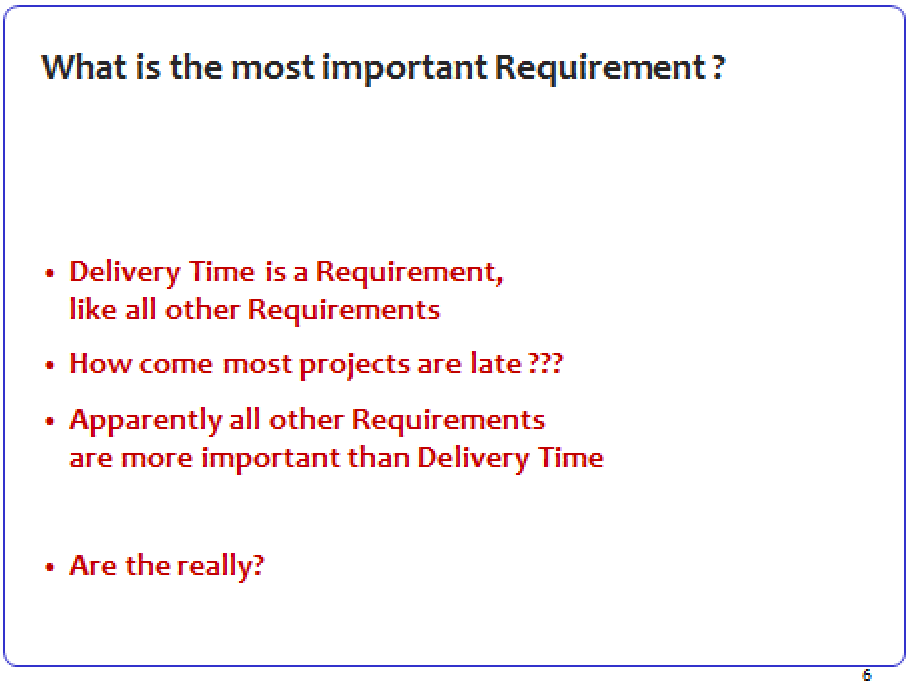


Cobb's Paradox

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

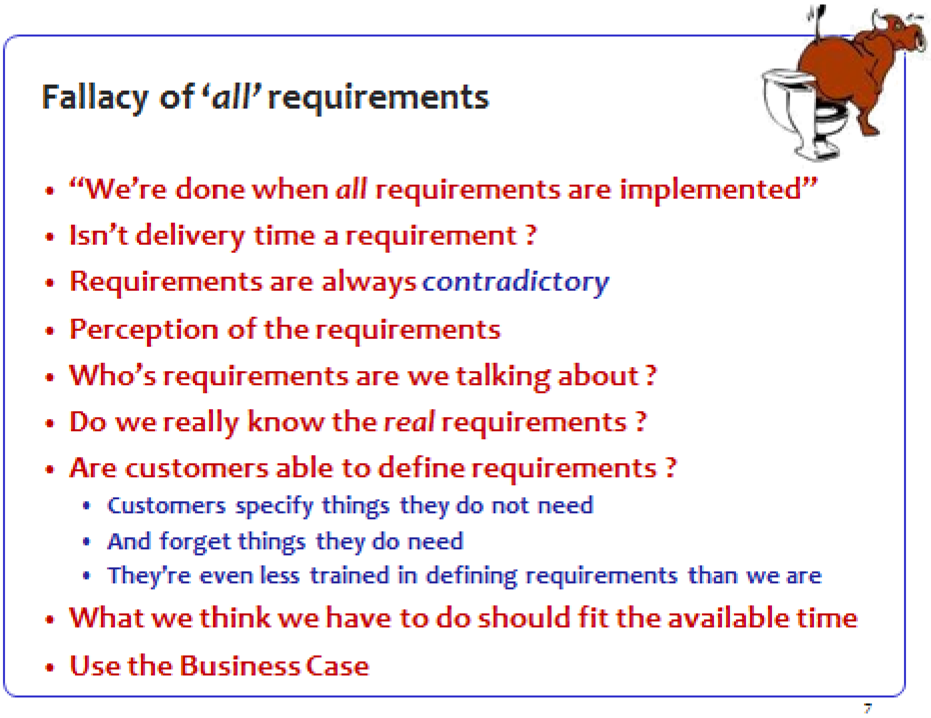
Martin Cobb
Treasury Board of Canada Secretariat
Ottawa, Canada

1989



What is the most important Requirement ?

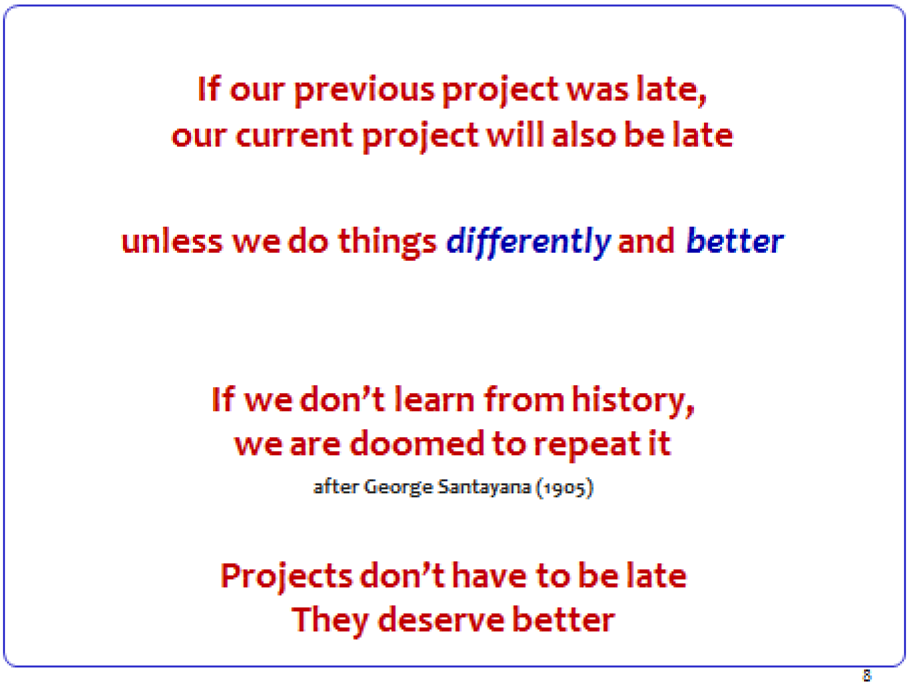
- Delivery Time is a Requirement, like all other Requirements
- How come most projects are late ???
- Apparently all other Requirements are more important than Delivery Time
- Are the really?



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

7



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

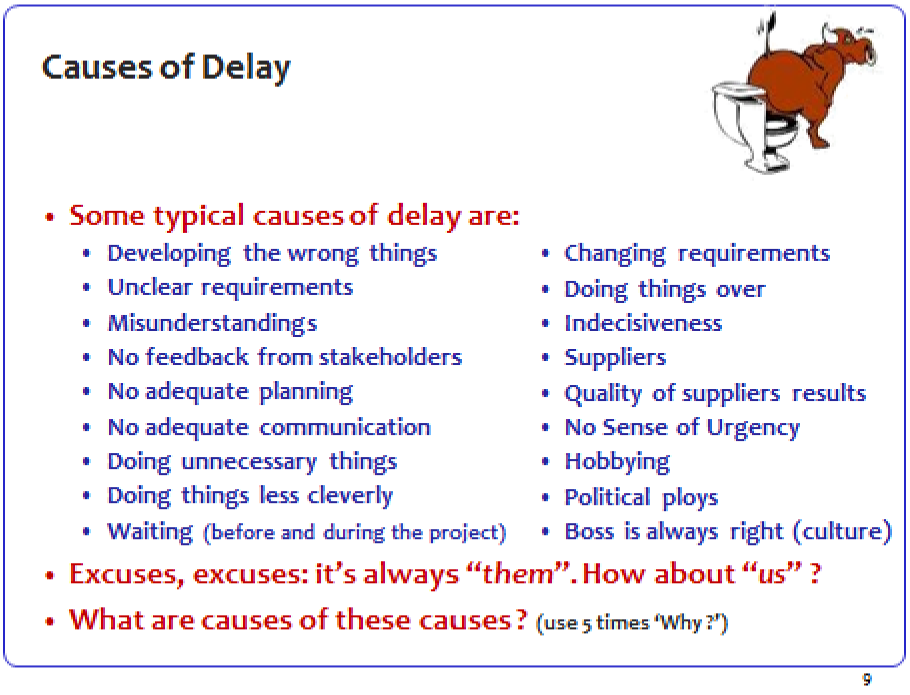
unless we do things *differently* and *better*

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

after George Santayana (1905)

Projects don't have to be late
They deserve better

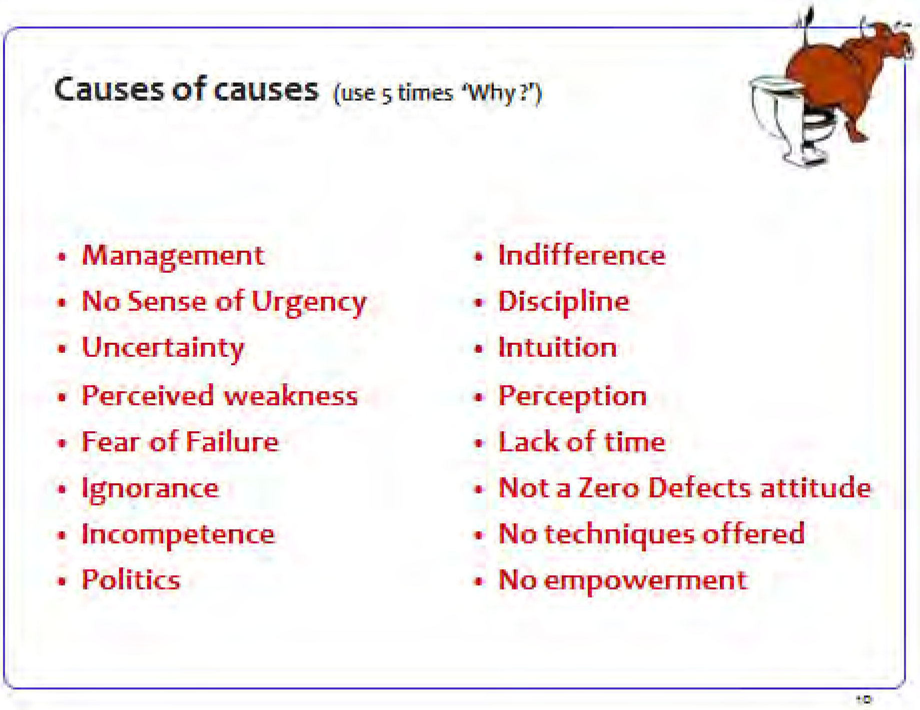
8



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*" ?
- What are causes of these causes ? (use 5 times 'Why ?')

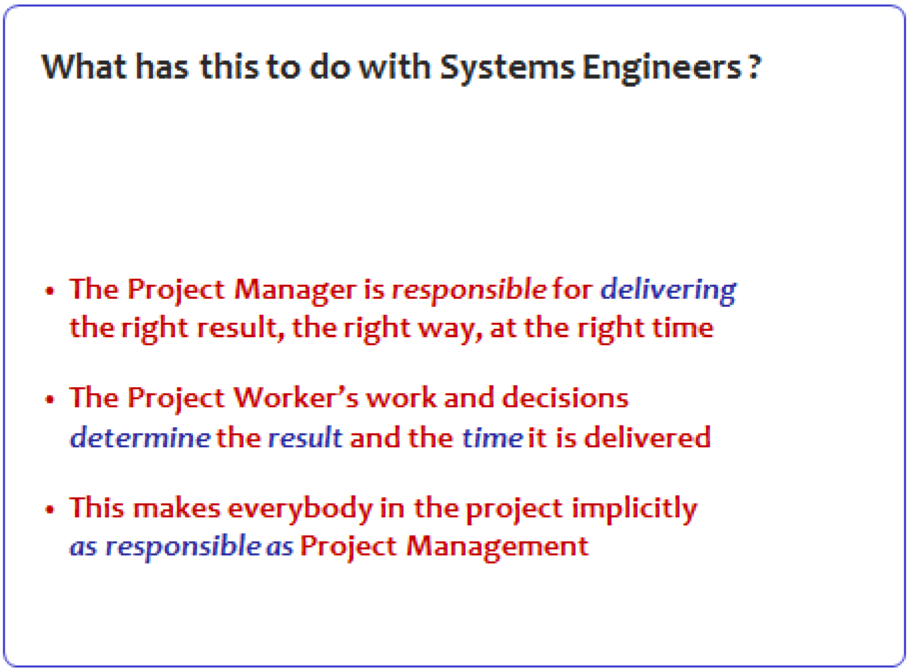
9



Causes of causes (use 5 times 'Why ?')

- Management
- No Sense of Urgency
- Uncertainty
- Perceived weakness
- Fear of Failure
- Ignorance
- Incompetence
- Politics
- Indifference
- Discipline
- Intuition
- Perception
- Lack of time
- Not a Zero Defects attitude
- No techniques offered
- No empowerment

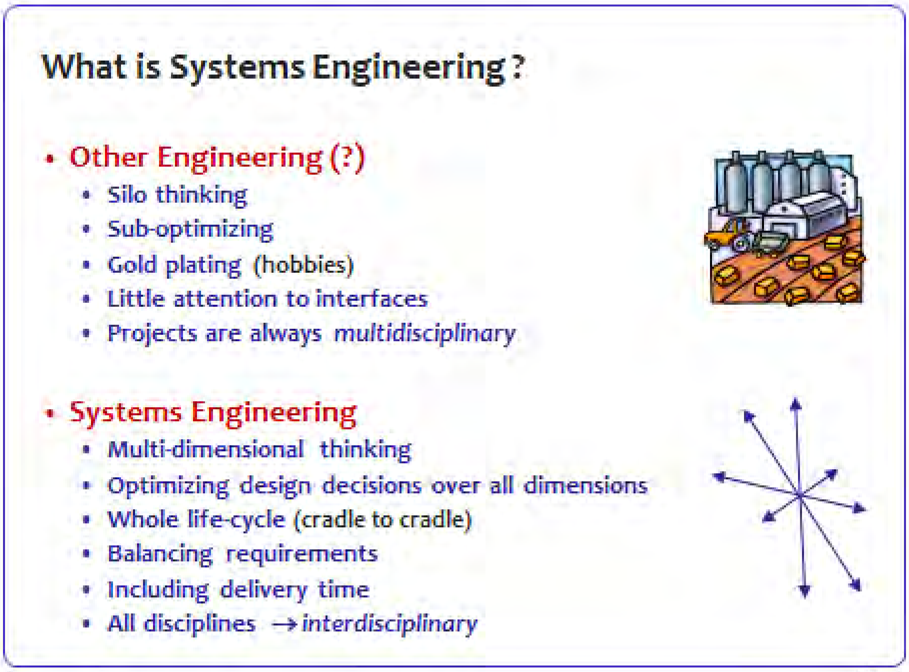
10



What has this to do with Systems Engineers ?

- The Project Manager is *responsible* for *delivering* the right result, the right way, 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

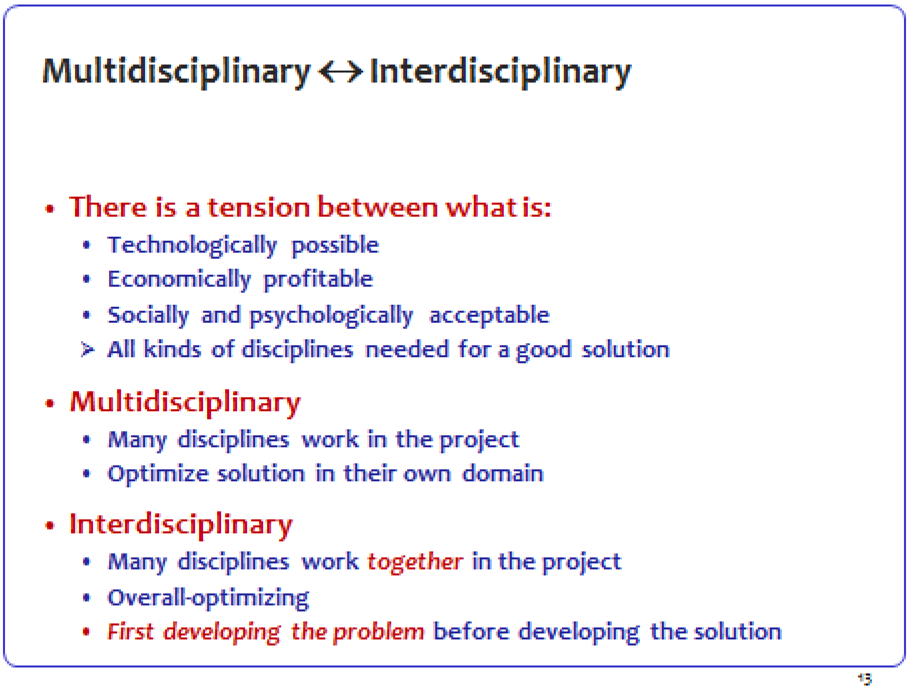
11



What is 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*

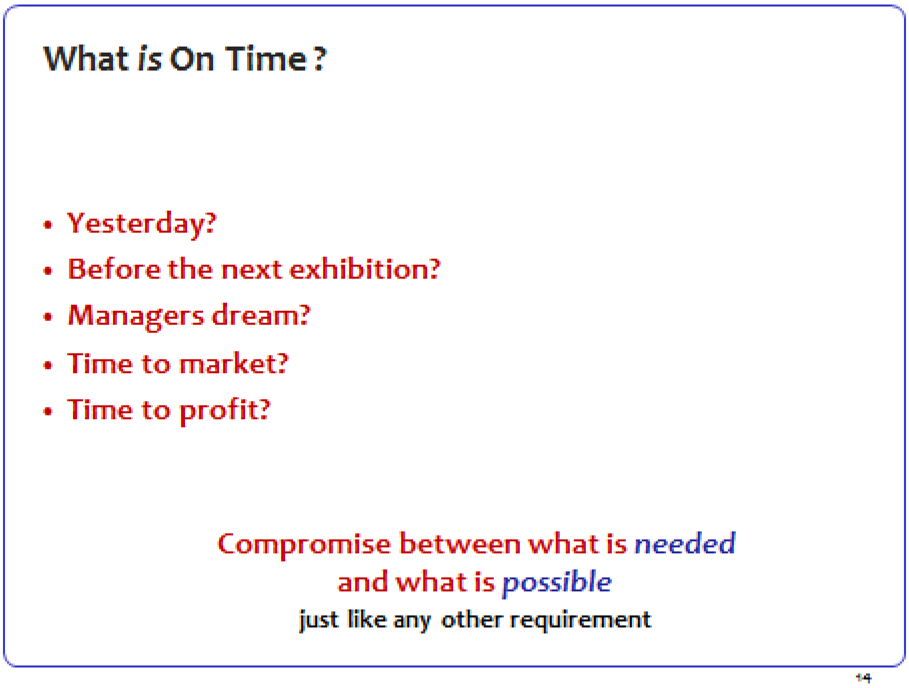
12



Multidisciplinary ↔ Interdisciplinary

- There is a tension between what is:
  - 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

13

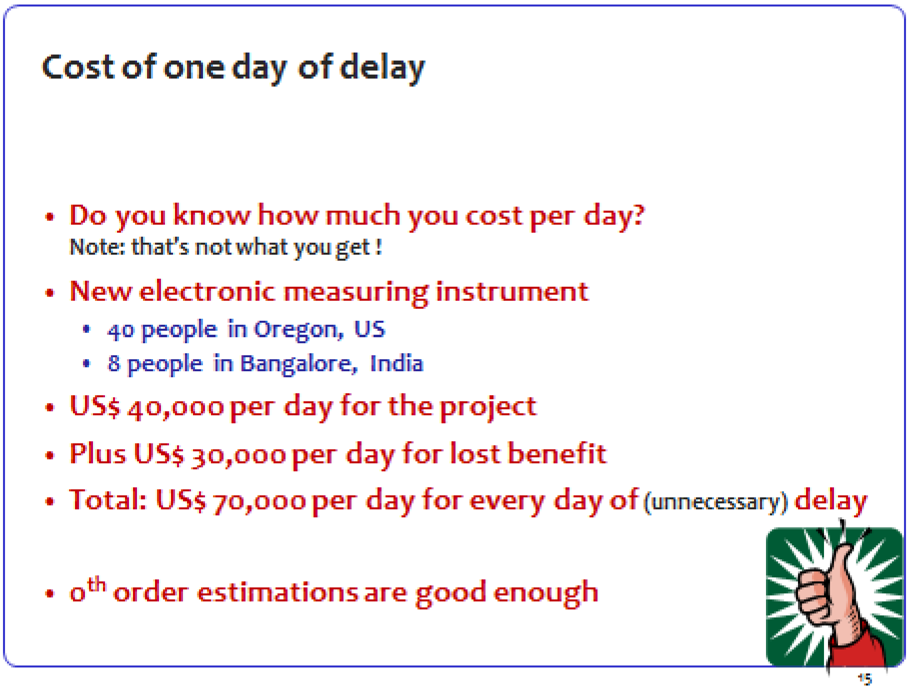


What *is* On Time ?

- 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

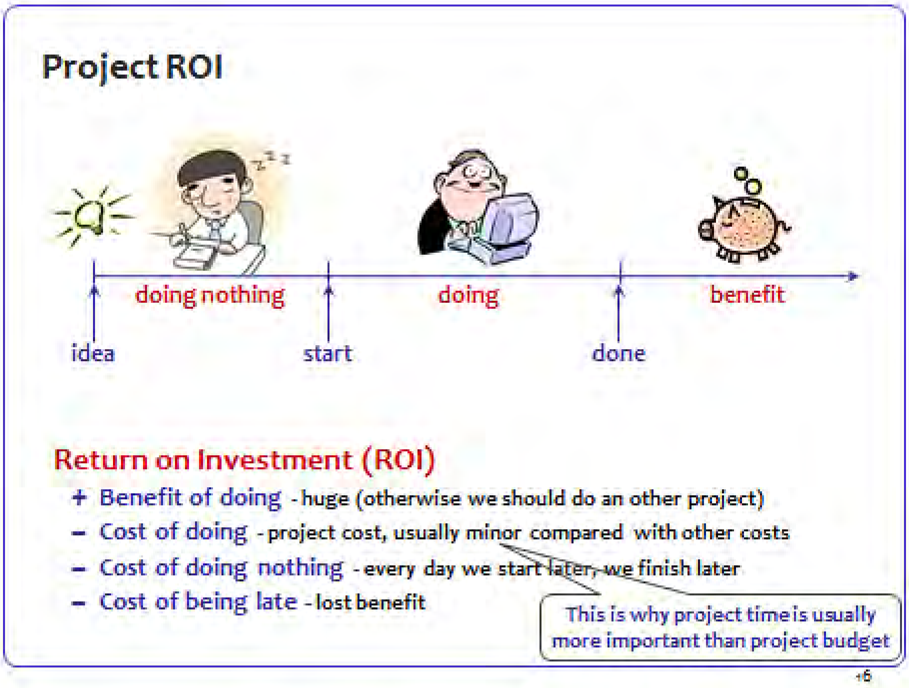
14



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
- 0th order estimations are good enough

15



Project ROI

Return on Investment (ROI)

+ Benefit of doing - huge (otherwise we should do an other project)

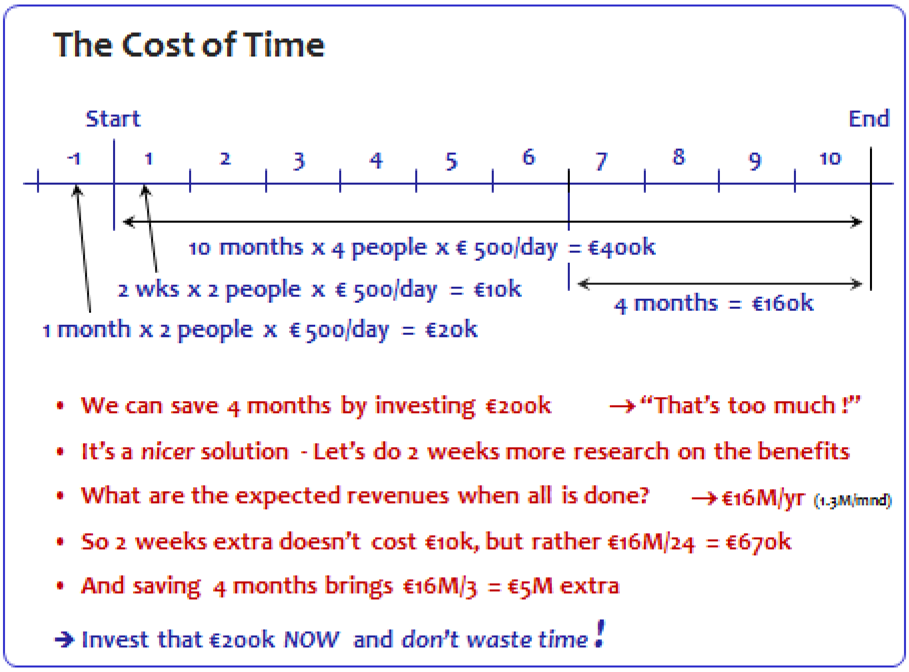
- Cost of doing - project cost, usually minor compared with other costs

- Cost of doing nothing - every day we start later, we finish later

- Cost of being late - lost benefit

This is why project time is usually more important than project budget

16



The Cost of Time

10 months x 4 people x € 500/day = €400k

2 wks x 2 people x € 500/day = €10k

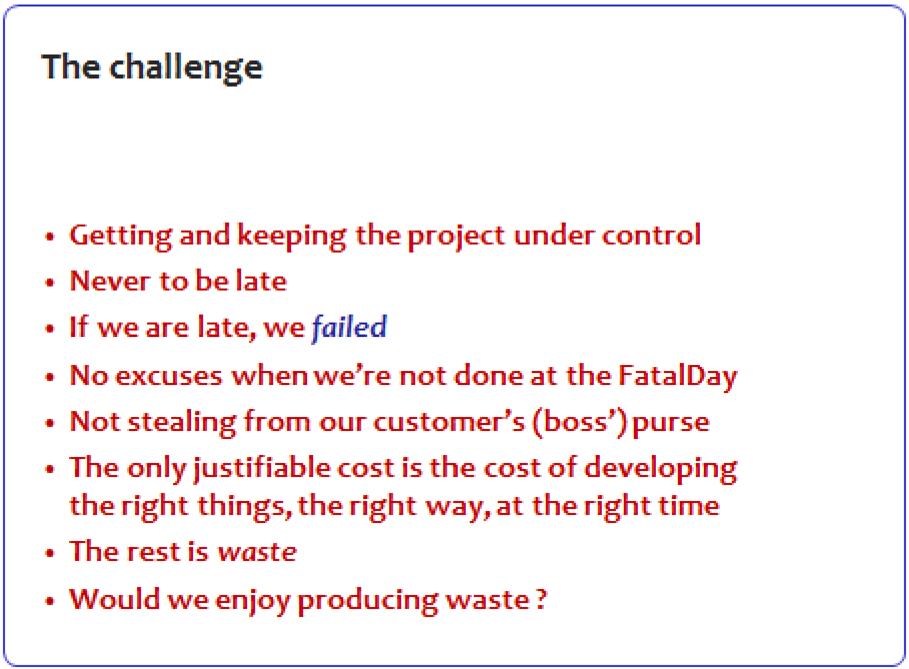
4 months = €160k

1 month x 2 people x € 500/day = €20k

- 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* !

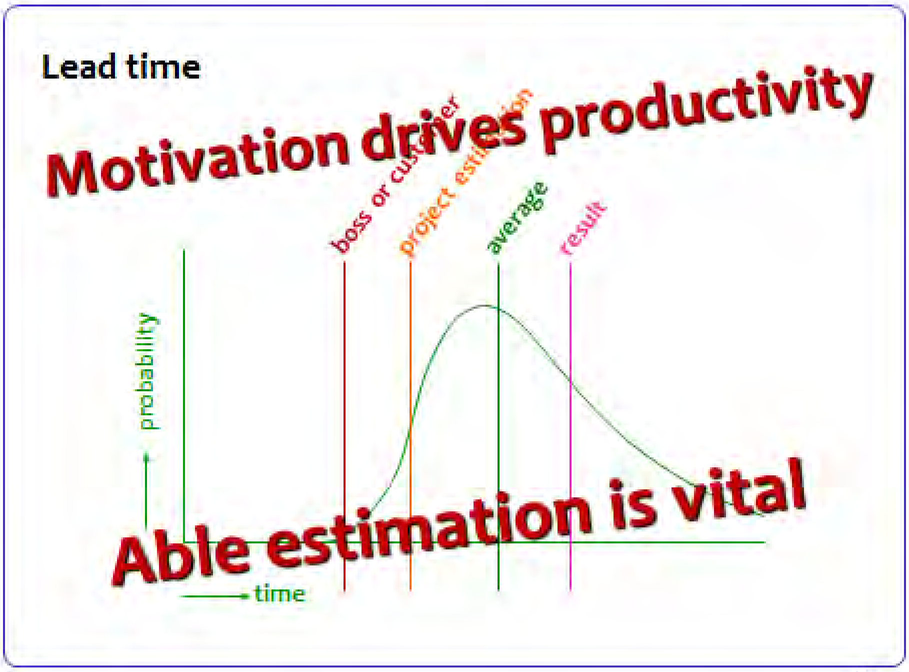
17



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, the right way, at the right time
- The rest is *waste*
- Would we enjoy producing waste ?

18

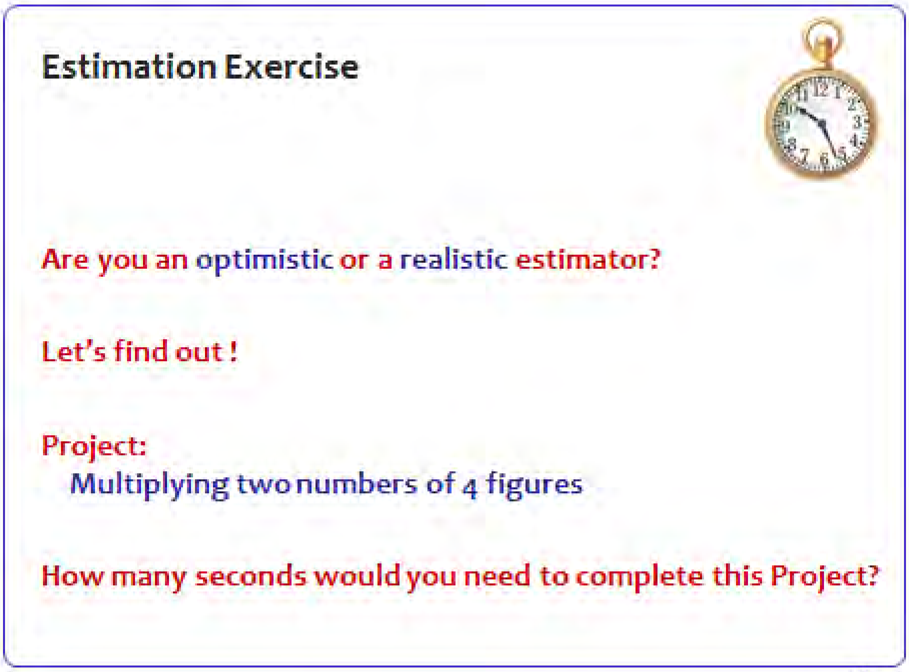


Lead time

Motivation drives productivity

Able estimation is vital

19



Estimation Exercise

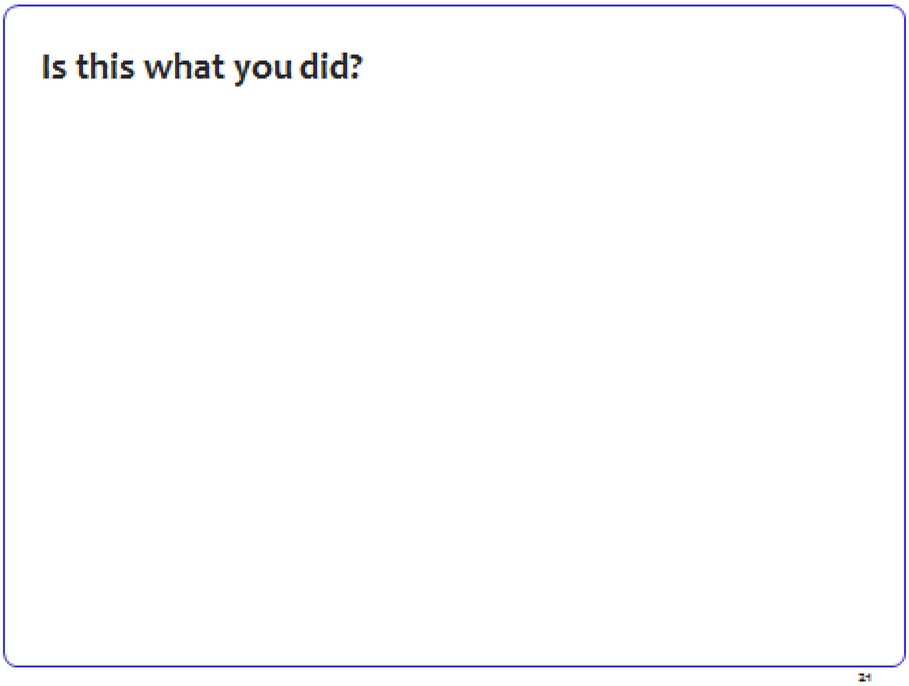
Are you an optimistic or a realistic estimator?

Let's find out !

Project:
Multiplying two numbers of 4 figures

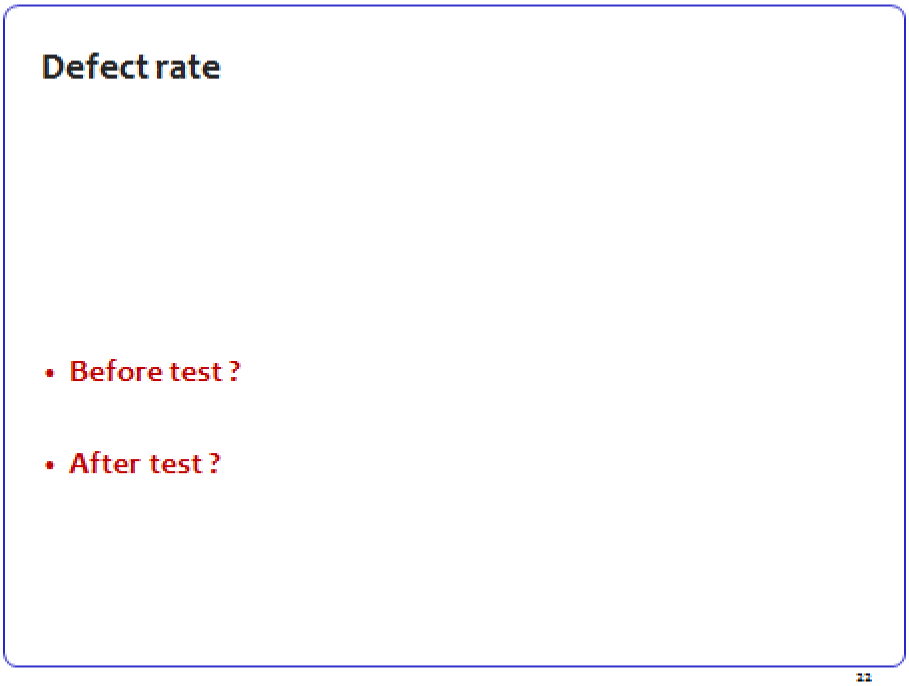
How many seconds would you need to complete this Project?

20



Is this what you did?

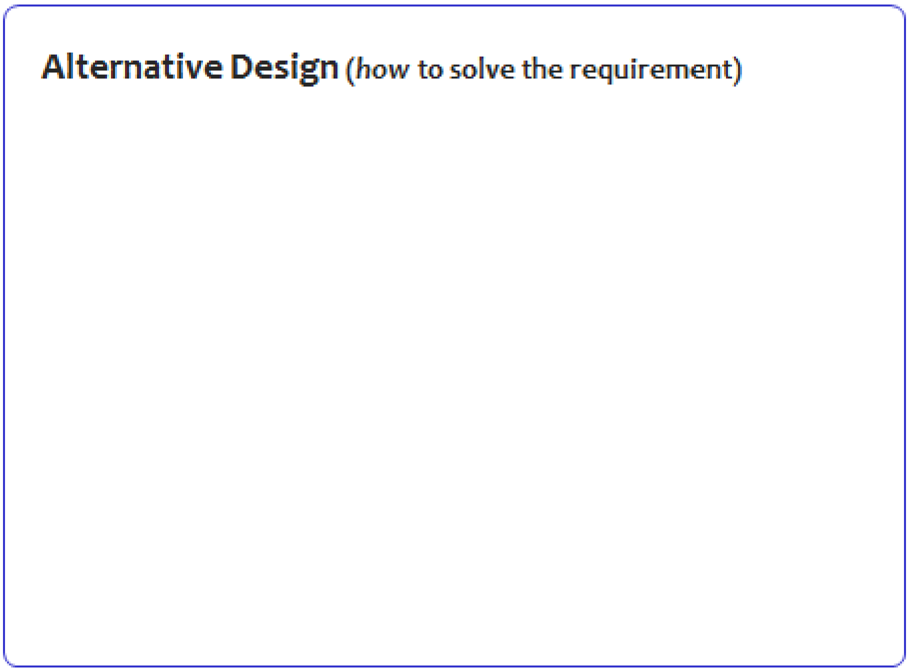
21



Defect rate

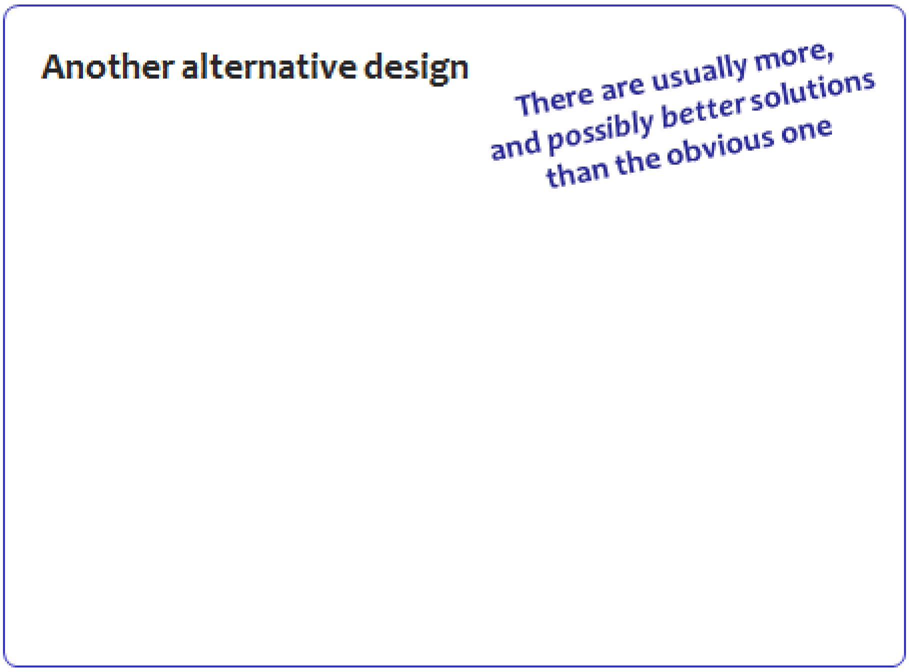
- Before test ?
- After test ?

22



Alternative Design (*how to solve the requirement*)

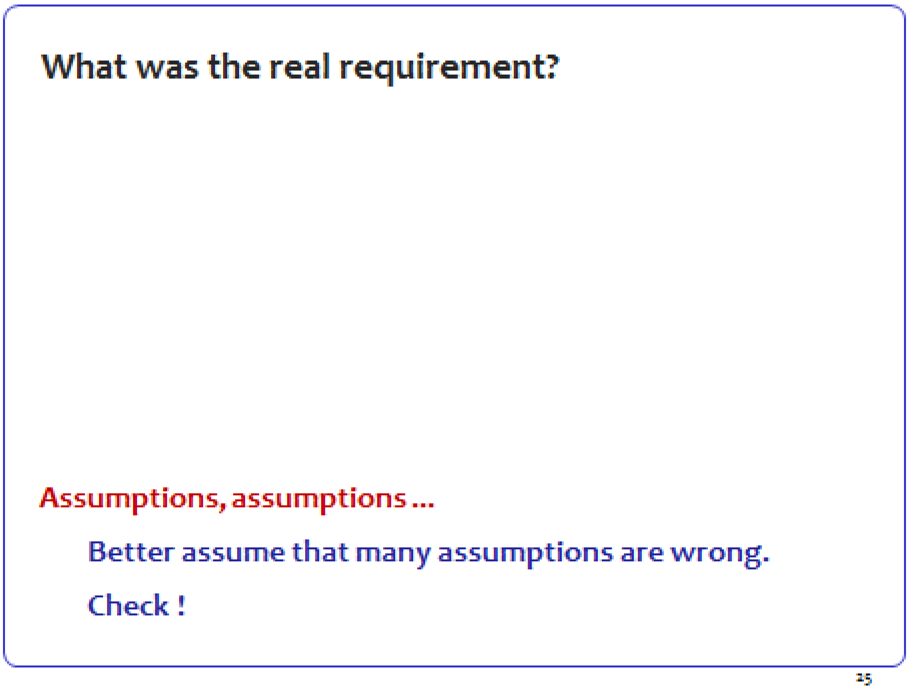
23



Another alternative design

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

24



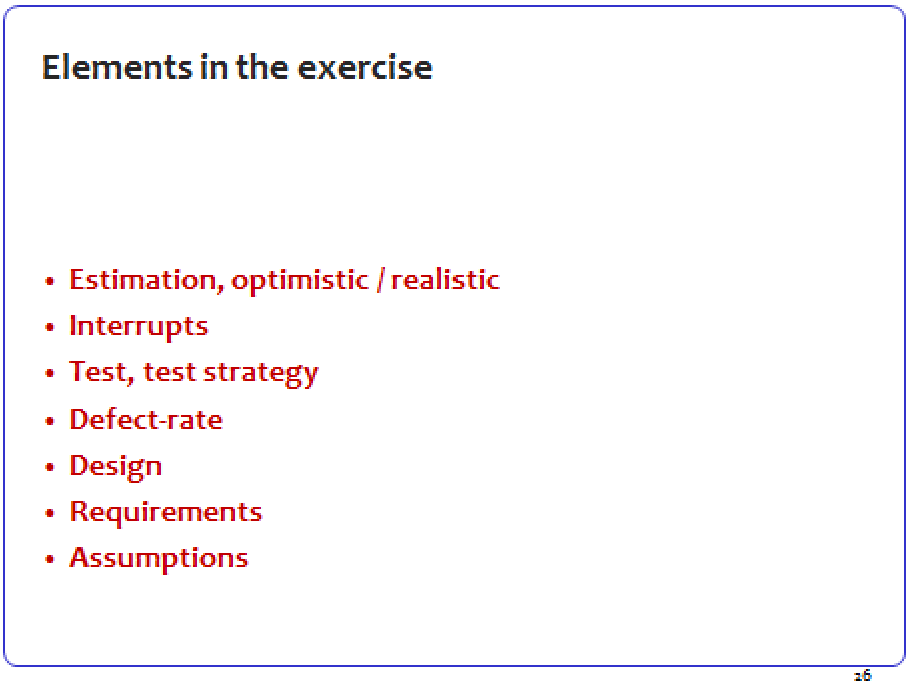
What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

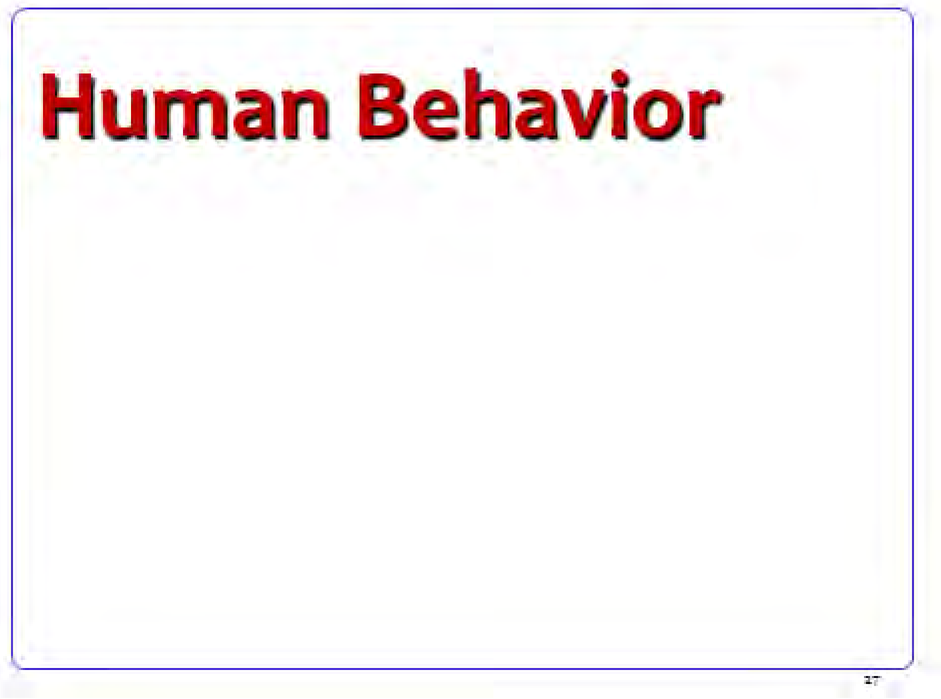
25



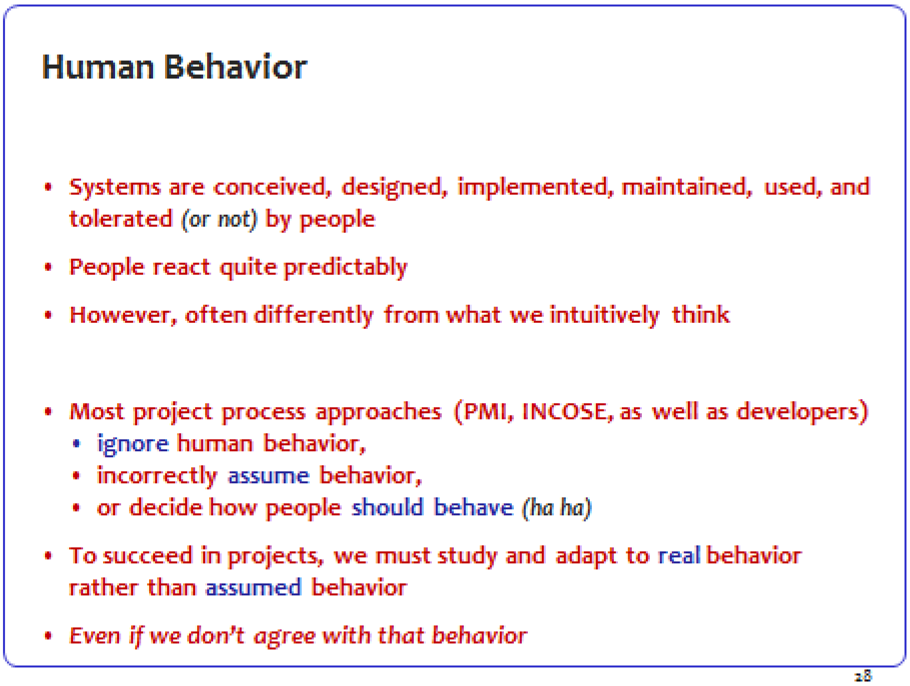
Elements in the exercise

- Estimation, optimistic / realistic
- Interrupts
- Test, test strategy
- Defect-rate
- Design
- Requirements
- Assumptions

26



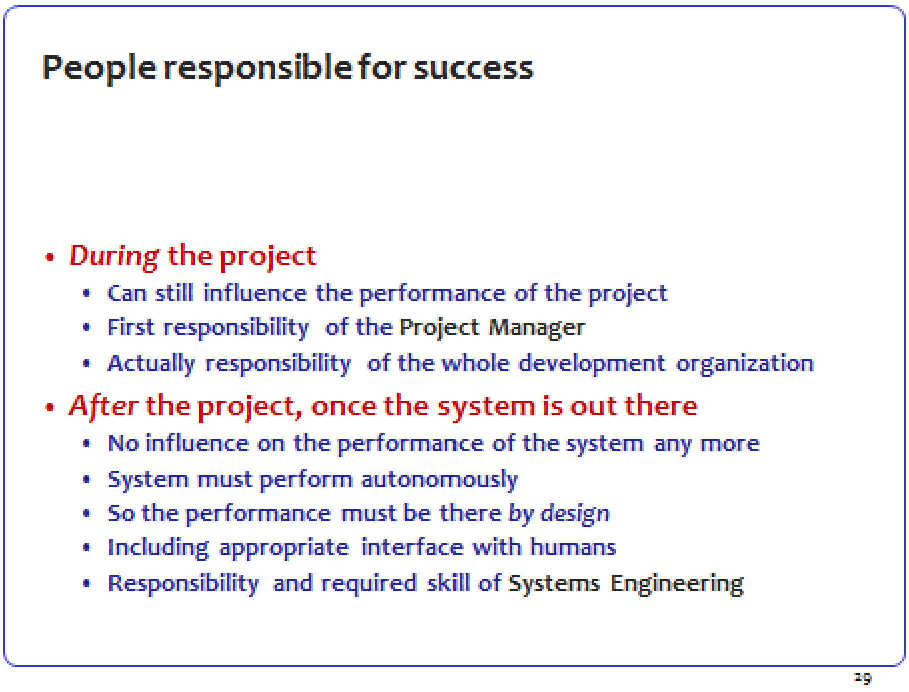
Human Behavior



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

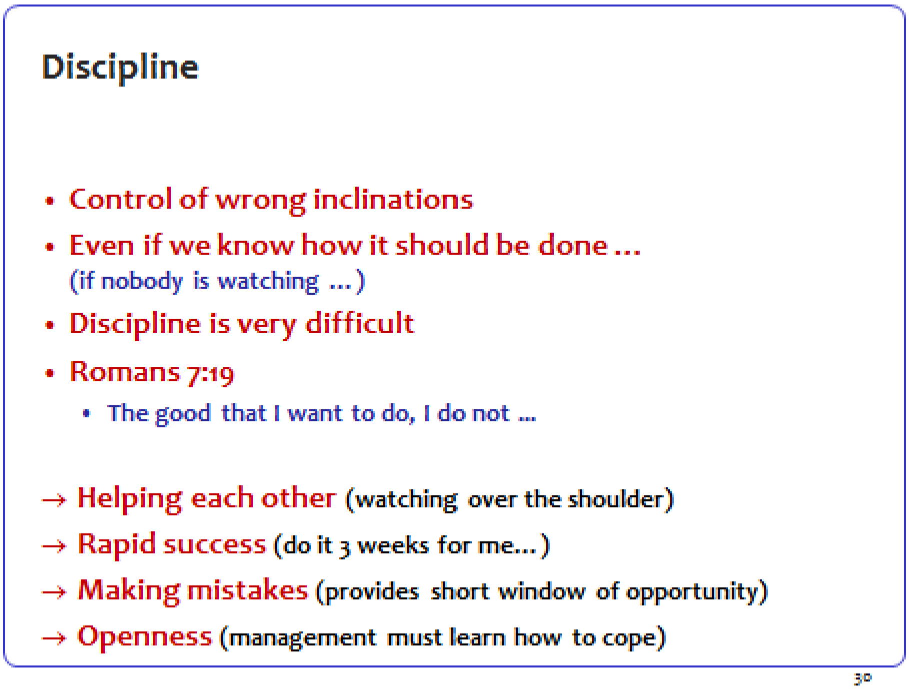
- Most project process approaches (PMI, INCOSE, as well as developers)
  - ignore human behavior,
  - incorrectly assume behavior,
  - or decide how people should behave *(ha ha)*
- To succeed in projects, we must study and adapt to real behavior rather than assumed behavior
- *Even if we don't agree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



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

29

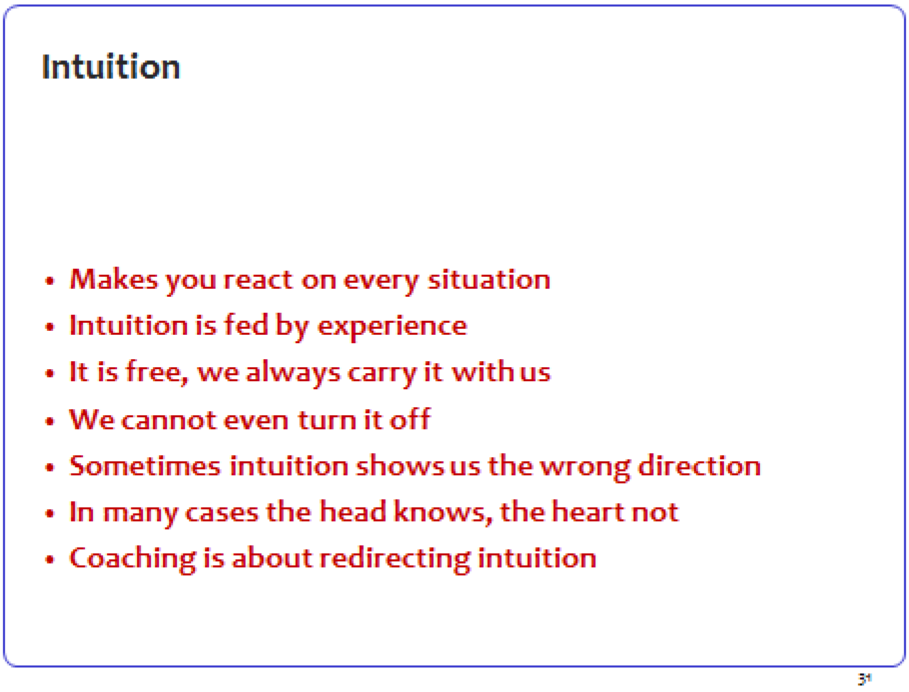


Discipline

- 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)

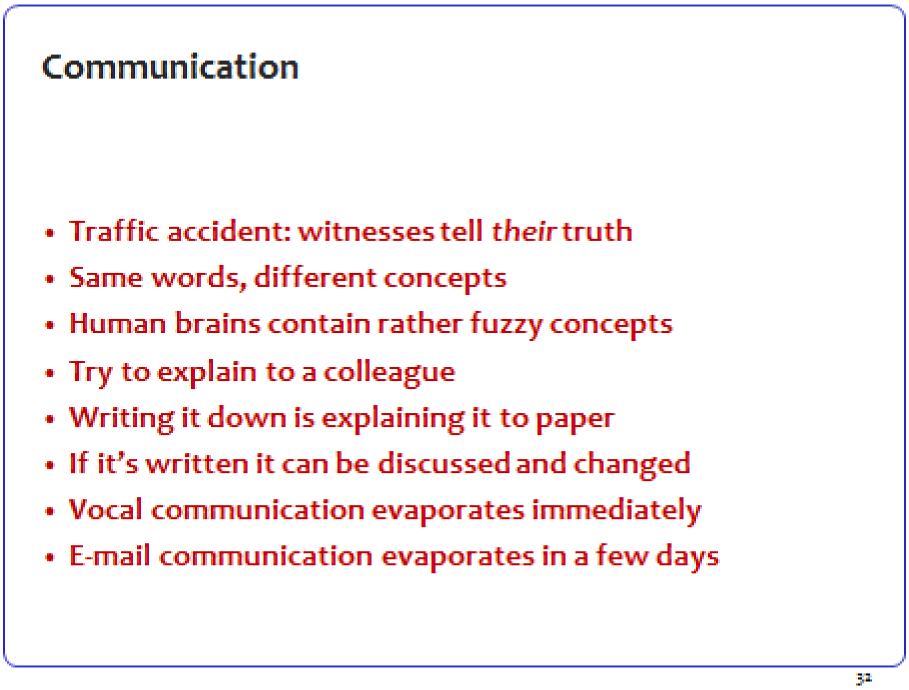
30



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

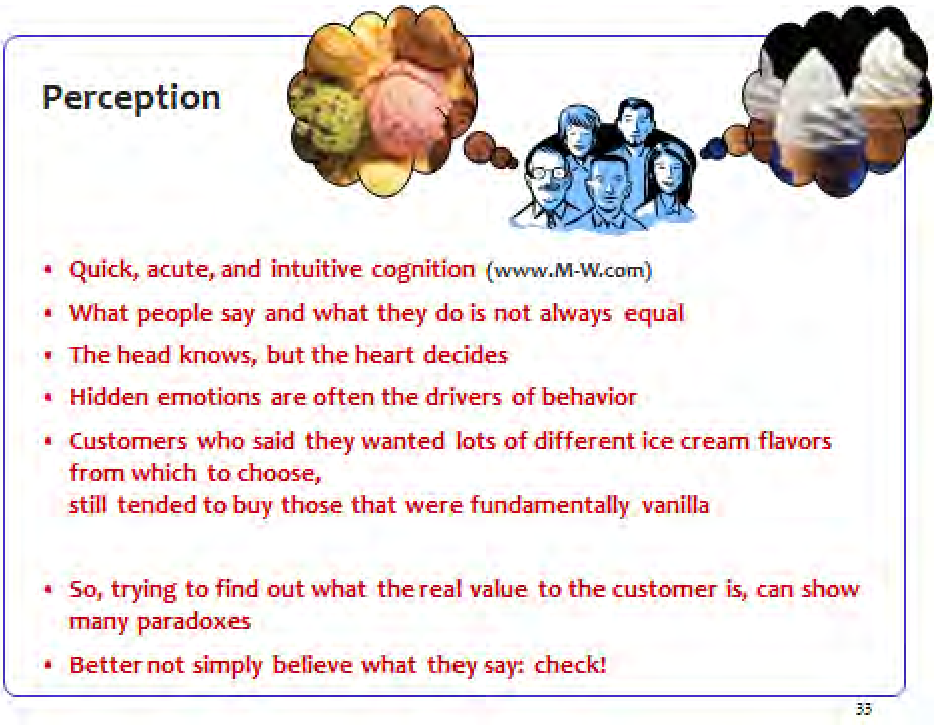
31



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

32

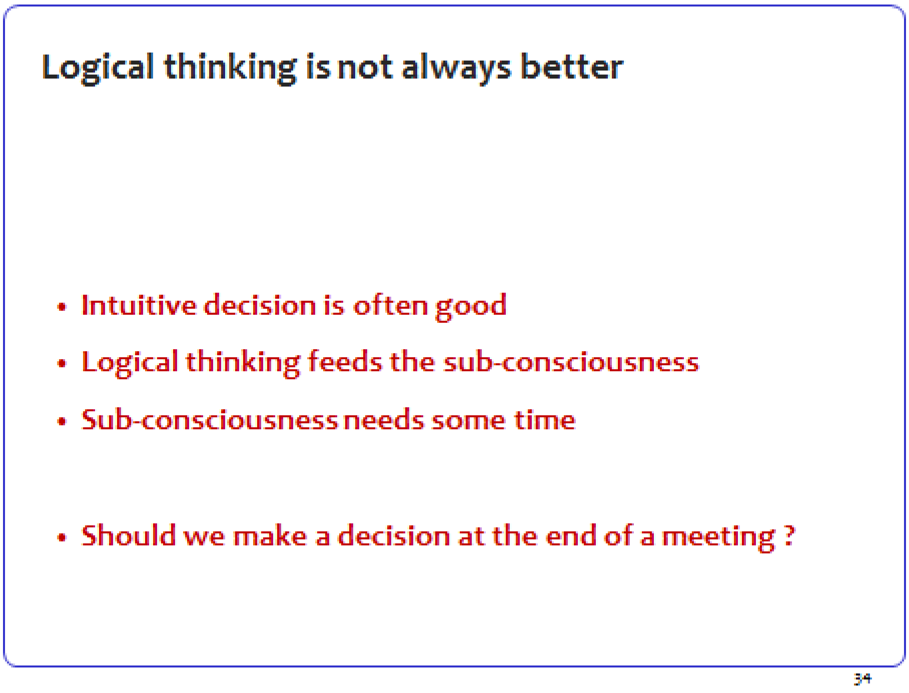


Perception

- 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!

33

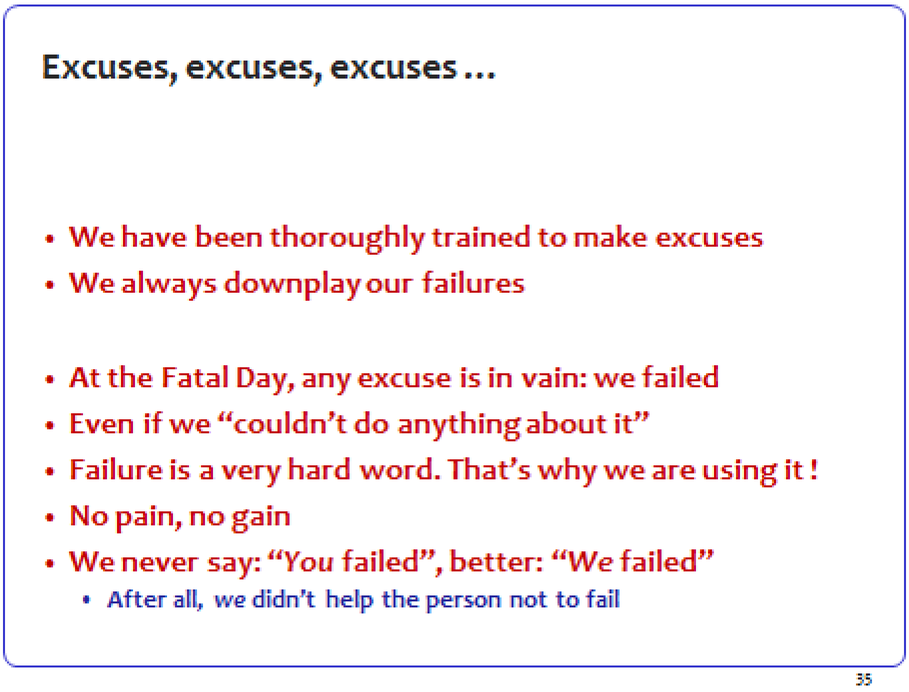


Logical thinking is not always better

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

- Should we make a decision at the end of a meeting ?

34



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

35

We failed because of politics

- Good politics:
  - People decide differently 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 ?

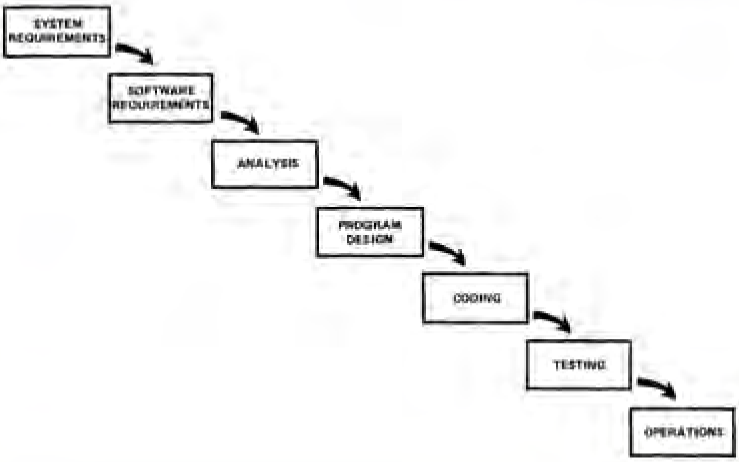
36

Models

37

Waterfall ?

Winston Royce 1970



38

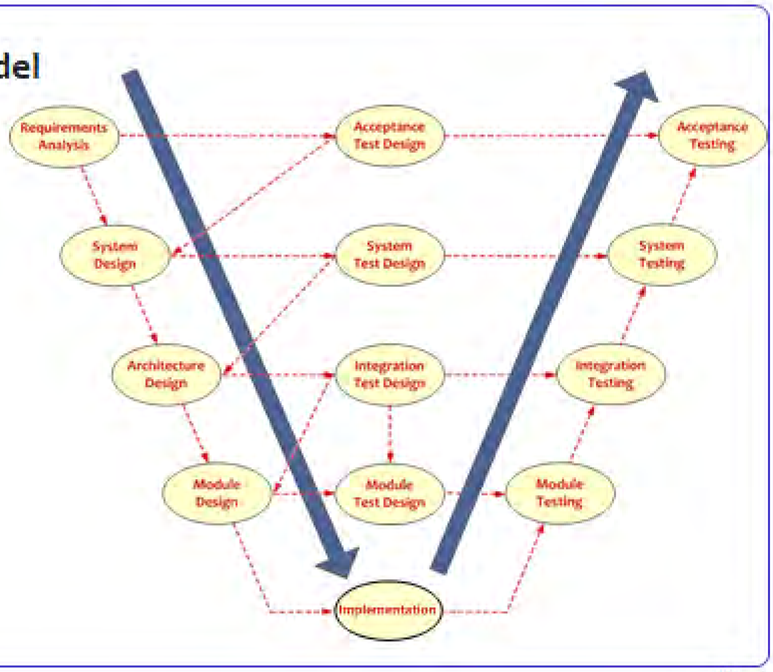
When can we use waterfall ?

- 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 ?

39

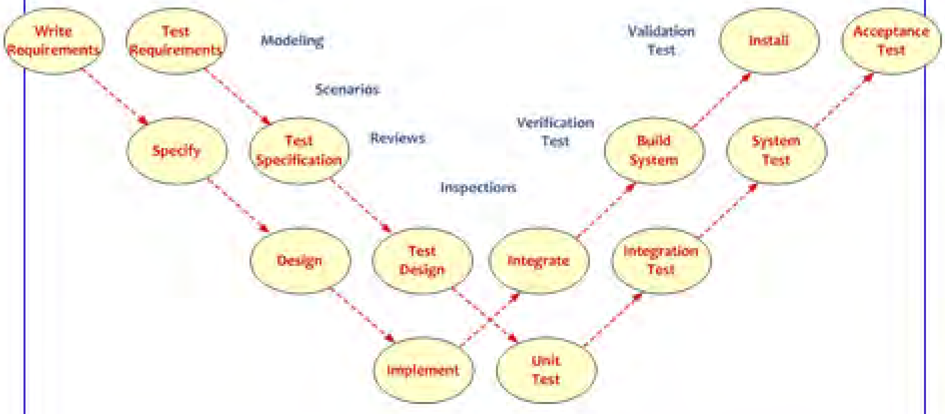
V-Model



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

40

W-model

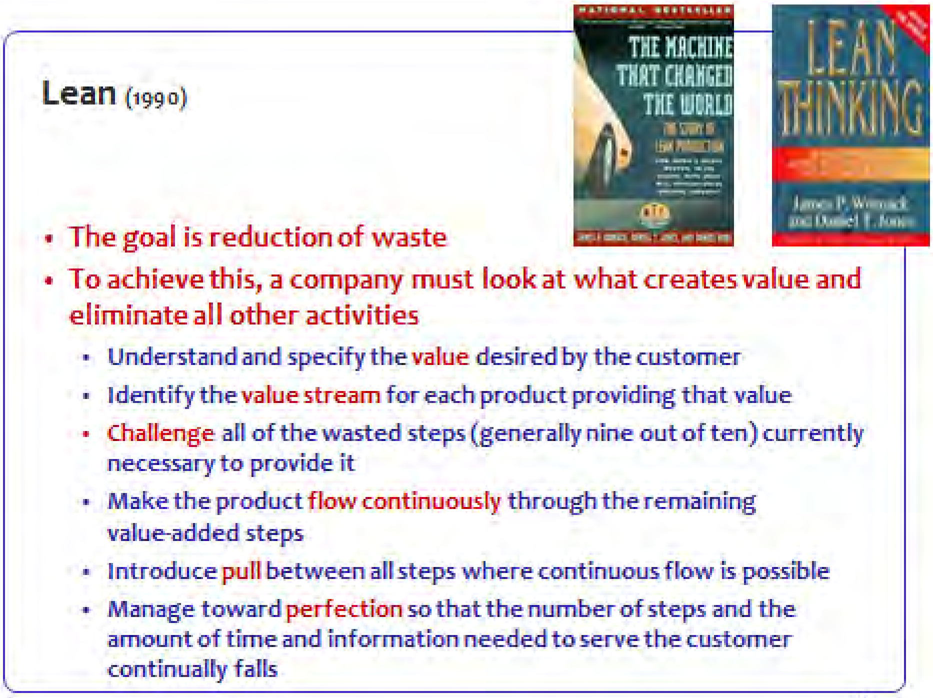


41

All Models are wrong

Some are useful

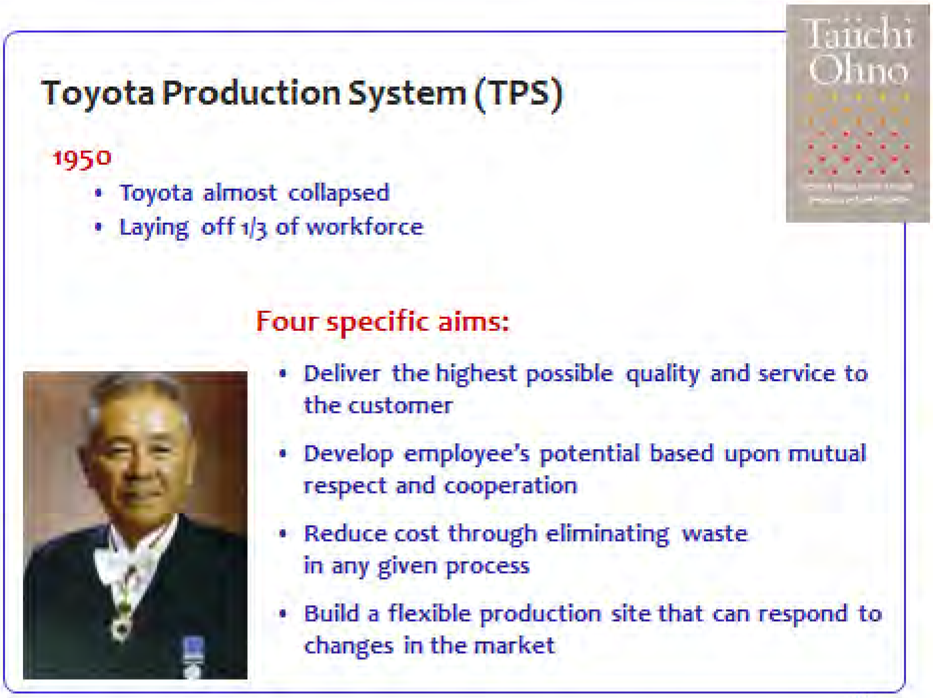
42


THE MACHINE THAT CHANGED THE WORLD
LEAN THINKING
James P. Womack and Daniel T. Jones

Lean (1990)

- The goal is reduction of waste
- To achieve this, a company must look at what creates value and eliminate all other activities
  - Understand and specify the value desired by the customer
  - Identify the value stream for each product providing that value
  - Challenge all of the wasted steps (generally nine out of ten) currently necessary to provide it
  - Make the product flow continuously through the remaining value-added steps
  - Introduce pull between all steps where continuous flow is possible
  - Manage toward perfection so that the number of steps and the amount of time and information needed to serve the customer continually falls

43



Toyota Production System (TPS)

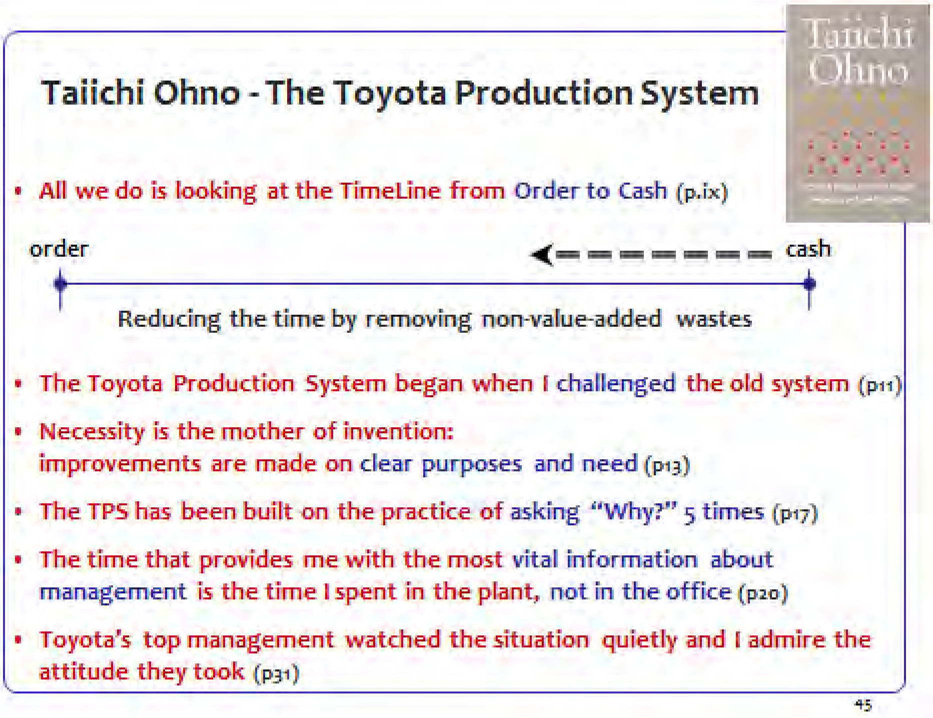
1950

- Toyota almost collapsed
- Laying off 1/3 of workforce

Four specific aims:

- Deliver the highest possible quality and service to the customer
- Develop employee's potential based upon mutual respect and cooperation
- Reduce cost through eliminating waste in any given process
- Build a flexible production site that can respond to changes in the market

44


Taiichi Ohno

Taiichi Ohno - The Toyota Production System

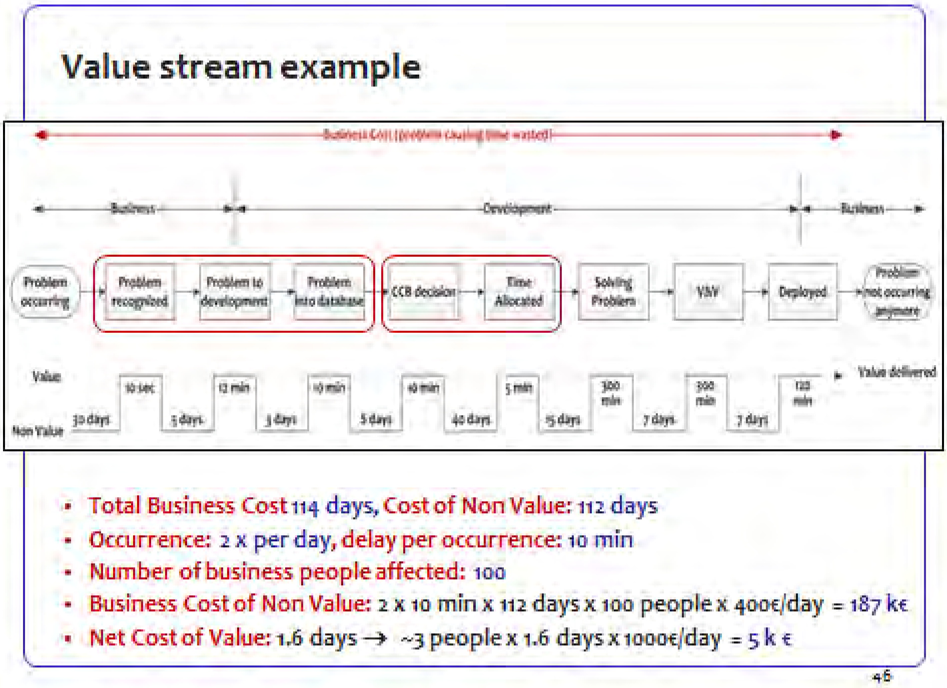
- All we do is looking at the TimeLine from Order to Cash (p.ix)

order ⟵ ════════ cash

Reducing the time by removing non-value-added wastes

- The Toyota Production System began when I challenged the old system (p11)
- Necessity is the mother of invention:
  improvements are made on clear purposes and need (p13)
- The TPS has been built on the practice of asking “Why?” 5 times (p17)
- The time that provides me with the most vital information about management is the time I spent in the plant, not in the office (p20)
- Toyota’s top management watched the situation quietly and I admire the attitude they took (p31)

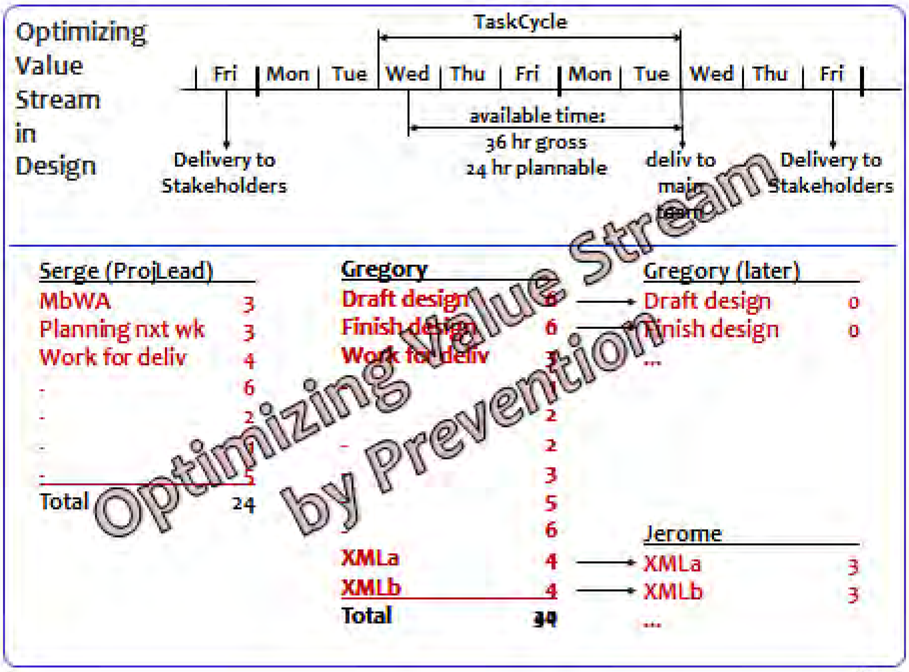
45



Value stream example

- Total Business Cost 114 days, Cost of Non Value: 112 days
- Occurrence: 2 x per day, delay per occurrence: 10 min
- Number of business people affected: 100
- Business Cost of Non Value: 2 x 10 min x 112 days x 100 people x 400€/day = 187 k€
- Net Cost of Value: 1.6 days → ~3 people x 1.6 days x 1000€/day = 5 k €

46



Optimizing Value Stream in Design

Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

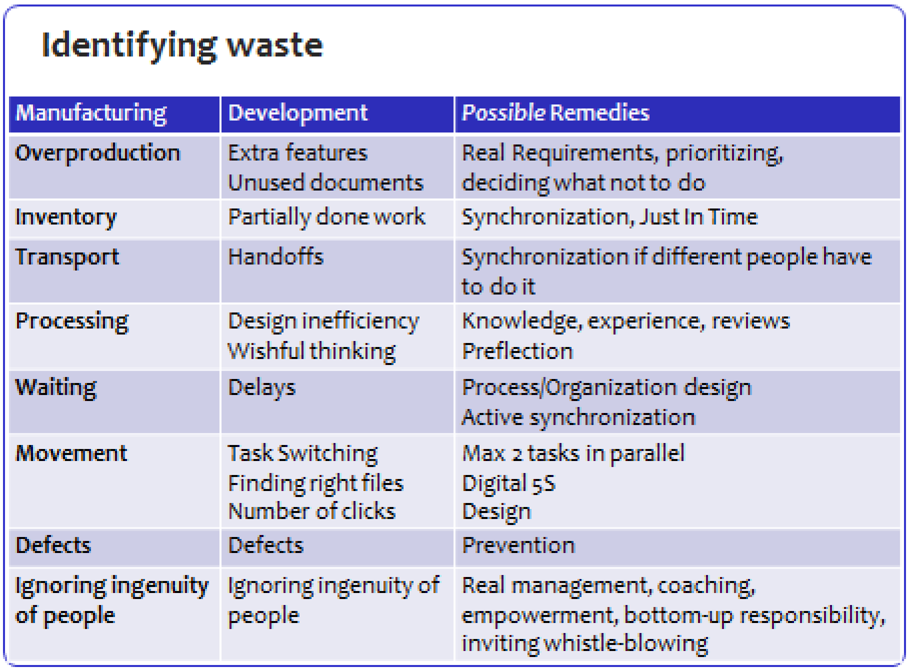
Gregory	
Draft design	6
Finish design	6
Work for deliv	3
	1
	2
-	2
	3
	5
-	6
XMLa	4
XMLb	4
Total	40

Gregory (later)	
Draft design	0
Finish design	0
...	

Jerome	
XMLa	3
XMLb	3
...	

Optimizing Value Stream by Prevention

47



Identifying waste

Manufacturing	Development	*Possible* Remedies
Overproduction	Extra features Unused documents	Real Requirements, prioritizing, deciding what not to do
Inventory	Partially done work	Synchronization, Just In Time
Transport	Handoffs	Synchronization if different people have to do it
Processing	Design inefficiency Wishful thinking	Knowledge, experience, reviews Preflection
Waiting	Delays	Process/Organization design Active synchronization
Movement	Task Switching Finding right files Number of clicks	Max 2 tasks in parallel Digital 5S Design
Defects	Defects	Prevention
Ignoring ingenuity of people	Ignoring ingenuity of people	Real management, coaching, empowerment, bottom-up responsibility, inviting whistle-blowing

48

Building on quite some history

Do we still have to talk about this ?

- 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) – Quality reduces waste
- Joseph M. Juran (1904-2008)
  - Quality Control Handbook (1951, Japan – 1954)
  - Total Quality Management – TQM
  - Pareto Principle
- Philip Crosby (1926-2001)
  - Quality is Free (1980)
  - Zero-defects (1961)
- Taiichi Ohno (1912-1990)
  - (Implemented the) Toyota Production System (Beyond Large-Scale Production (1978, 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)

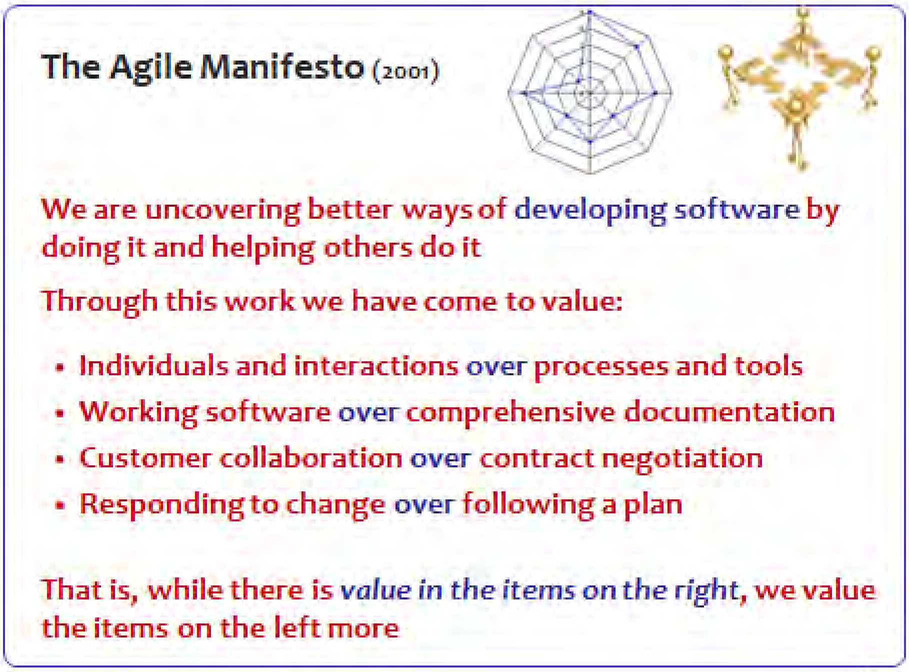
Eliminating Waste
Not doing what doesn't yield value

49

What is Agile ?

- A philosophy (Agile Manifesto)

50



The Agile Manifesto (2001)

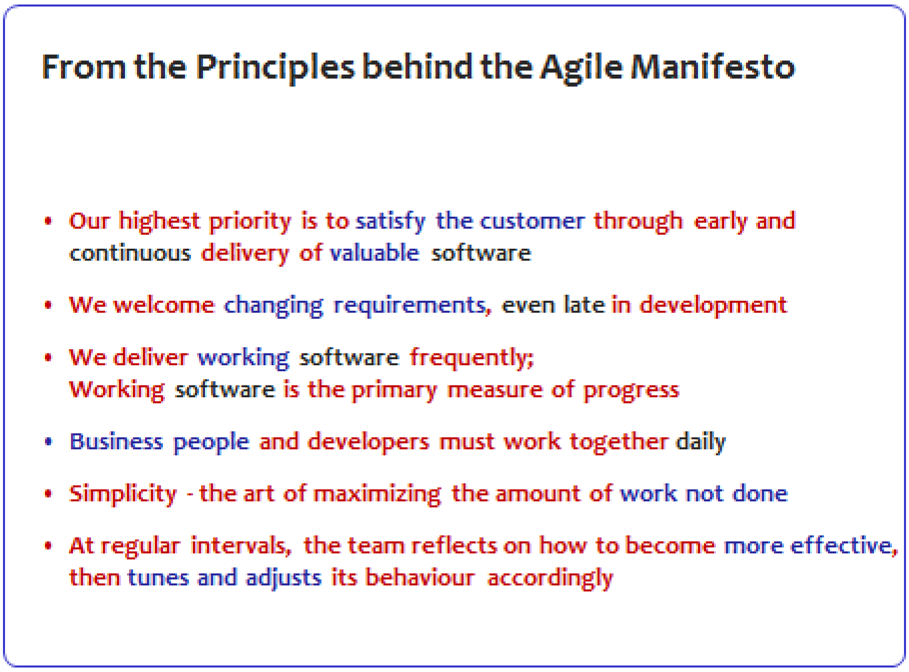
We are uncovering better ways of developing software by doing it and helping others do it

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is *value in the items on the right*, we value the items on the left more

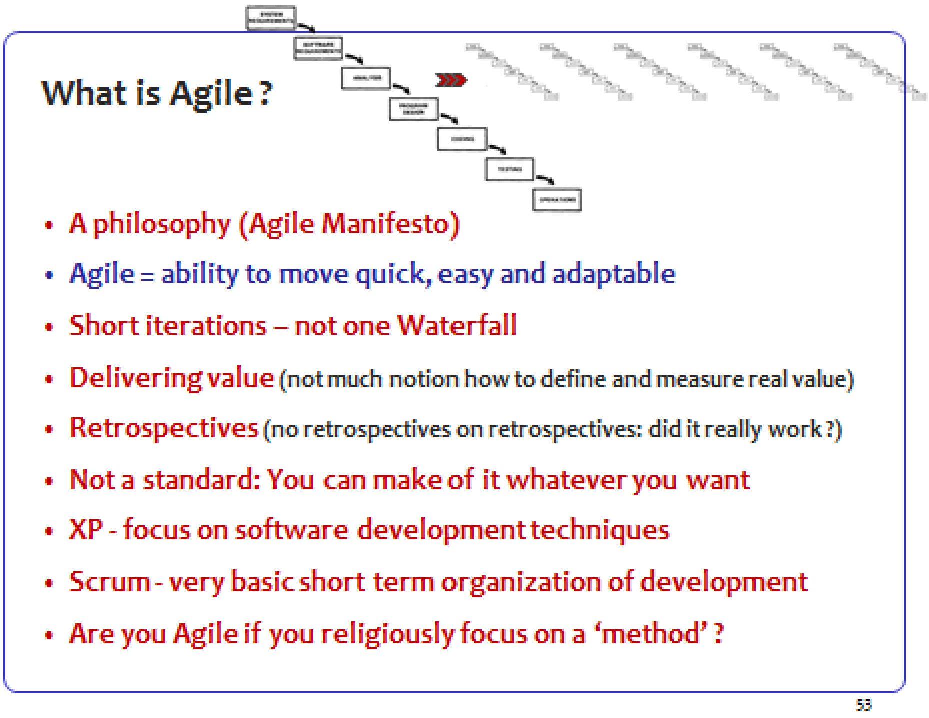
51



From the Principles behind the Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- We welcome changing requirements, even late in development
- We deliver working software frequently;
  Working software is the primary measure of progress
- Business people and developers must work together daily
- Simplicity - the art of maximizing the amount of work not done
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

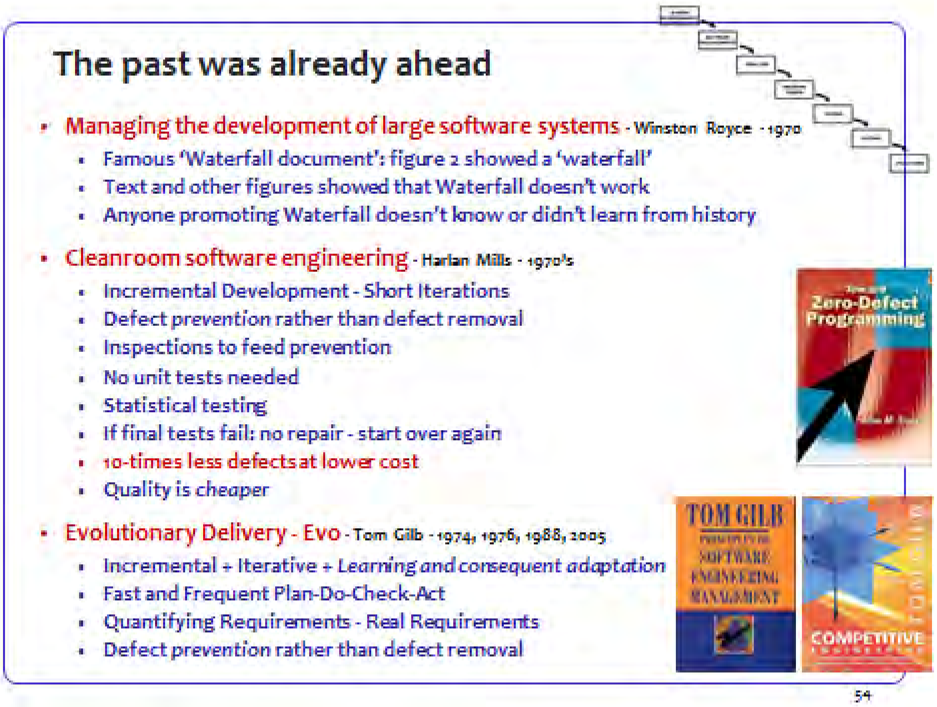
52



What is Agile ?

- A philosophy (Agile Manifesto)
- Agile = ability to move quick, easy and adaptable
- Short iterations – not one Waterfall
- Delivering value (not much notion how to define and measure real value)
- Retrospectives (no retrospectives on retrospectives: did it really work ?)
- Not a standard: You can make of it whatever you want
- XP - focus on software development techniques
- Scrum - very basic short term organization of development
- Are you Agile if you religiously focus on a 'method' ?

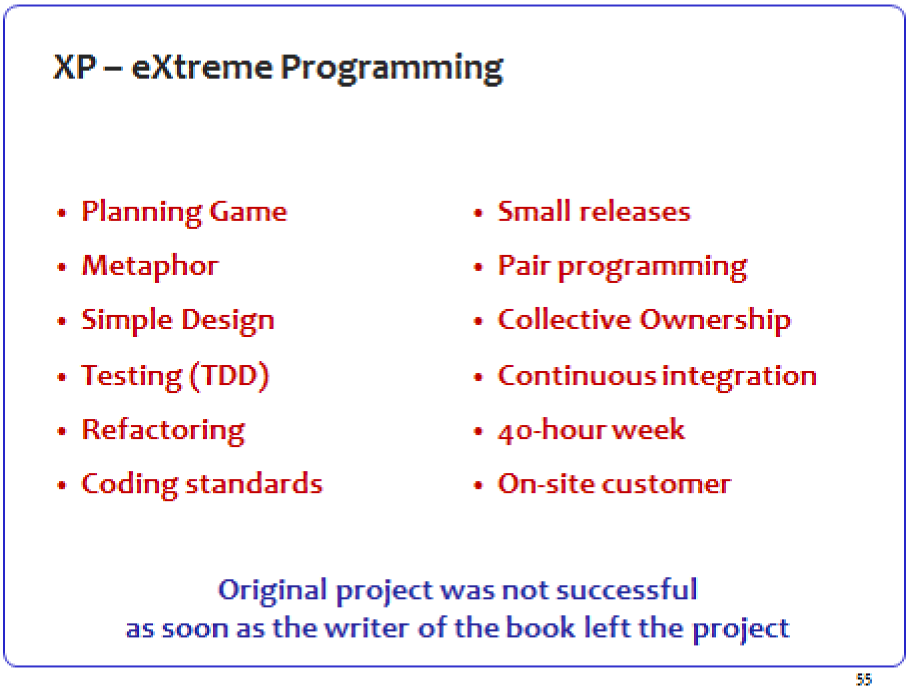
53



The past was already ahead

- Managing the development of large software systems - Winston 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
- Cleanroom software engineering - Harlan Mills - 1970's
  - Incremental Development - Short Iterations
  - Defect prevention rather than defect removal
  - Inspections to feed prevention
  - No unit tests needed
  - Statistical testing
  - If final tests fail: no repair - start over again
  - 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

54

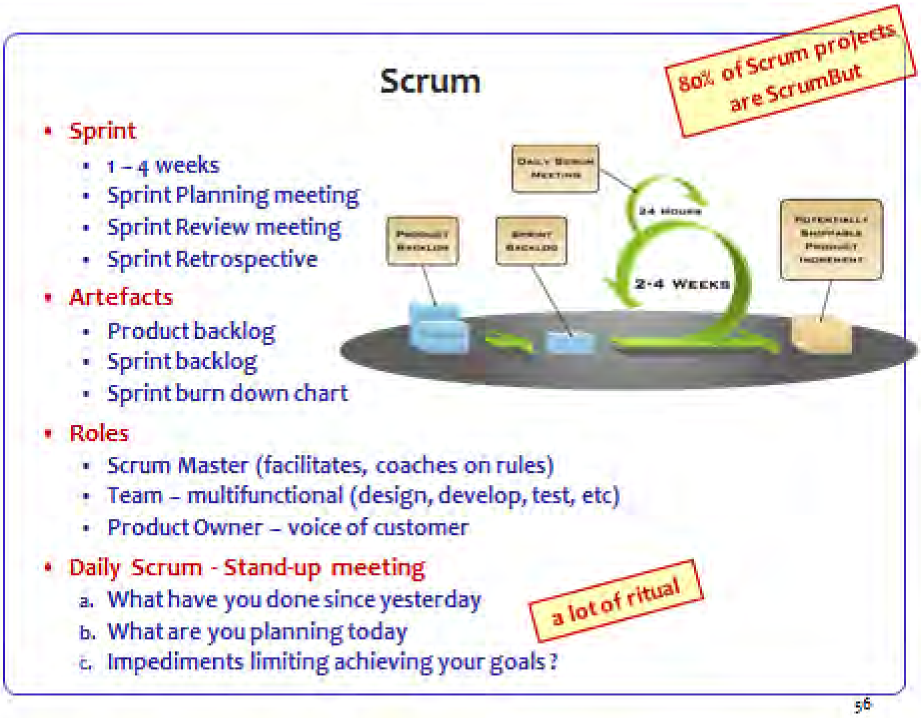


XP – eXtreme Programming

- Planning Game
- Metaphor
- Simple Design
- Testing (TDD)
- Refactoring
- Coding standards
- Small releases
- Pair programming
- Collective Ownership
- Continuous integration
- 40-hour week
- On-site customer

Original project was not successful
as soon as the writer of the book left the project

55



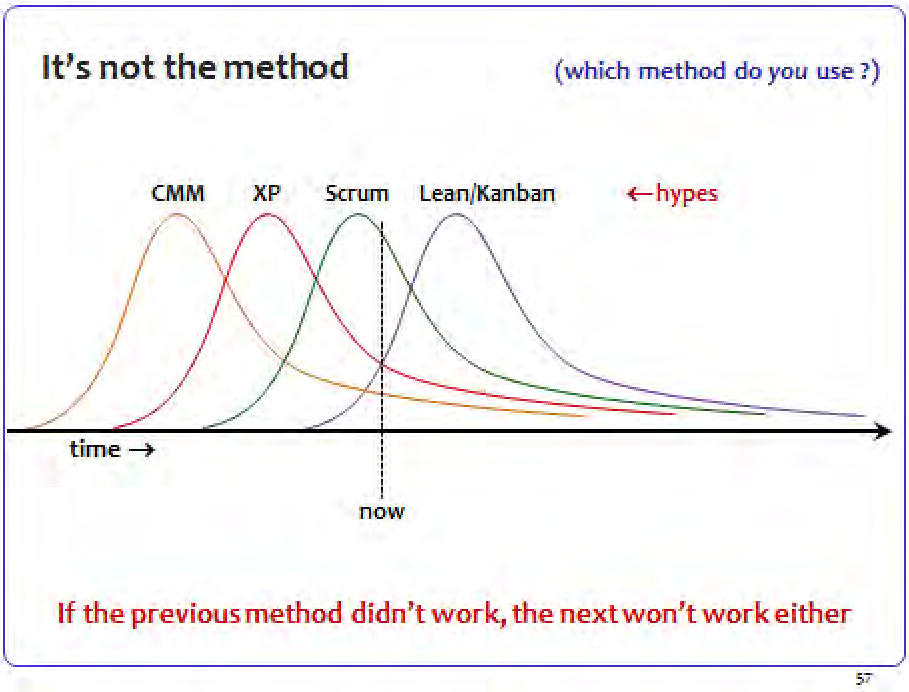
Scrum

80% of Scrum projects
are ScrumBut

- Sprint
  - 1 – 4 weeks
  - Sprint Planning meeting
  - Sprint Review meeting
  - Sprint Retrospective
- Artefacts
  - Product backlog
  - Sprint backlog
  - Sprint burn down chart
- Roles
  - Scrum Master (facilitates, coaches on rules)
  - Team – multifunctional (design, develop, test, etc)
  - Product Owner – voice of customer
- Daily Scrum - Stand-up meeting
  a. What have you done since yesterday
  b. What are you planning today
  c. Impediments limiting achieving your goals ?

a lot of ritual

56

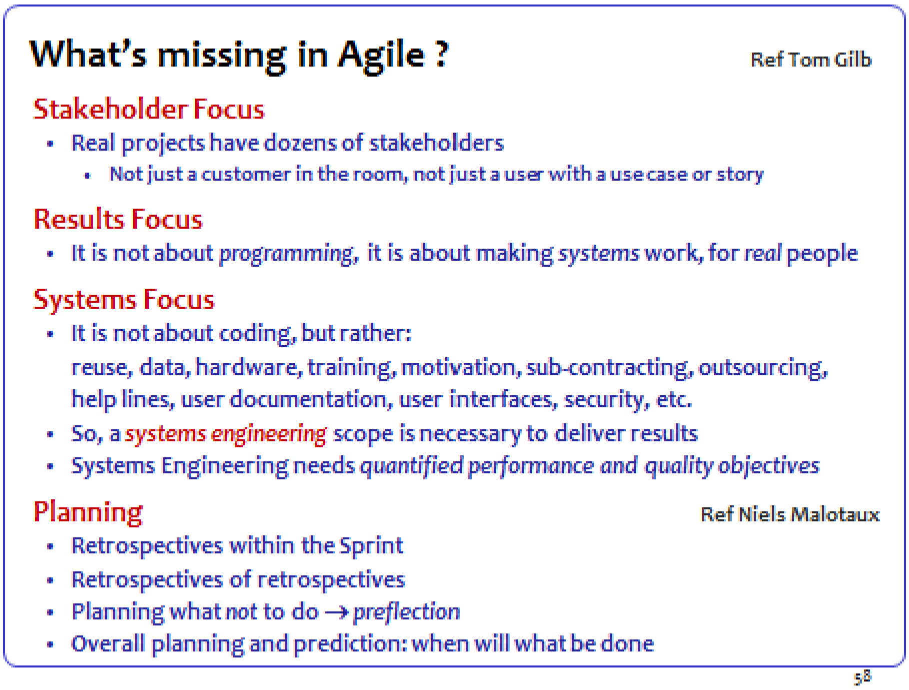


It's not the method

(which method do you use ?)

If the previous method didn't work, the next won't work either

57



What's missing in Agile ?

Ref Tom Gilb

Stakeholder Focus

- Real projects have dozens of stakeholders
  - Not just a customer in the room, not just a user with a use case or story

Results Focus

- It is not about *programming*, it is about making *systems* work, for *real* people

Systems Focus

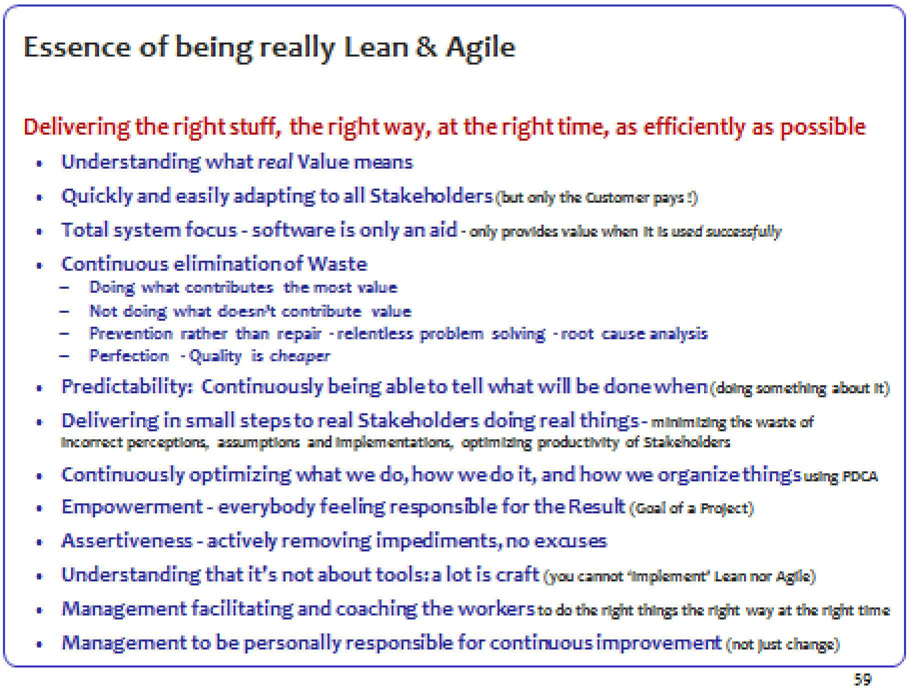
- It is not about coding, but rather:
  reuse, data, hardware, training, motivation, sub-contracting, outsourcing, help lines, user documentation, user interfaces, security, etc.
- So, a *systems engineering* scope is necessary to deliver results
- Systems Engineering needs *quantified performance and quality objectives*

Planning

Ref Niels Malotaux

- Retrospectives within the Sprint
- Retrospectives of retrospectives
- Planning what *not* to do → *preflection*
- Overall planning and prediction: when will what be done

58

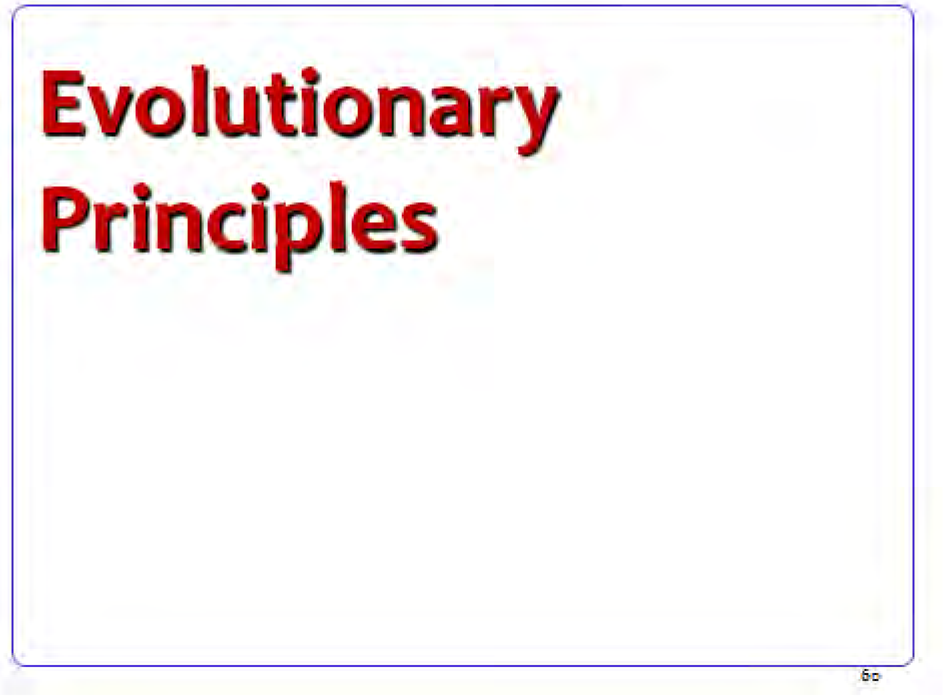


Essence of being really Lean & Agile

Delivering the right stuff, the right way, at the right time, as efficiently as possible

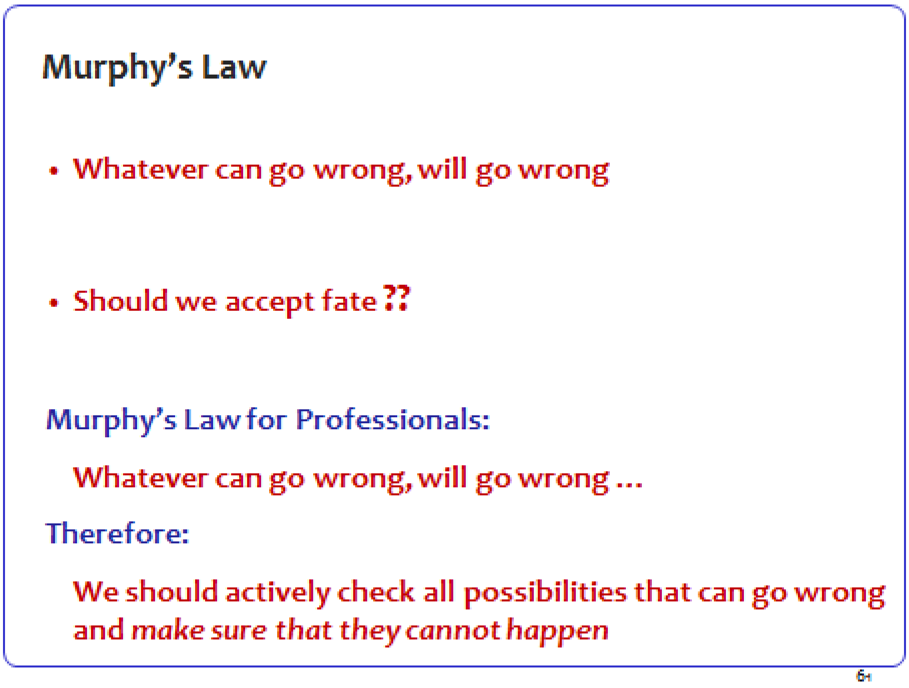
- Understanding what *real* Value means
- Quickly and easily adapting to all Stakeholders (but only the Customer pays !)
- Total system focus - software is only an aid - *only provides value when it is used successfully*
- Continuous elimination of Waste
  - Doing what contributes the most value
  - Not doing what doesn't contribute value
  - Prevention rather than repair - relentless problem solving - root cause analysis
  - Perfection - Quality is *cheaper*
- Predictability: Continuously being able to tell what will be done when (doing something about it)
- Delivering in small steps to real Stakeholders doing real things - minimizing the waste of incorrect perceptions, assumptions and implementations, optimizing productivity of Stakeholders
- Continuously optimizing what we do, how we do it, and how we organize things using PDCA
- Empowerment - everybody feeling responsible for the Result (Goal of a Project)
- Assertiveness - actively removing impediments, no excuses
- Understanding that it's not about tools: a lot is craft (you cannot 'implement' Lean nor Agile)
- Management facilitating and coaching the workers to do the right things the right way at the right time
- Management to be personally responsible for continuous improvement (not just change)

59



Evolutionary Principles

60



Murphy's Law

- Whatever can go wrong, will go wrong
- Should we accept fate ⁇

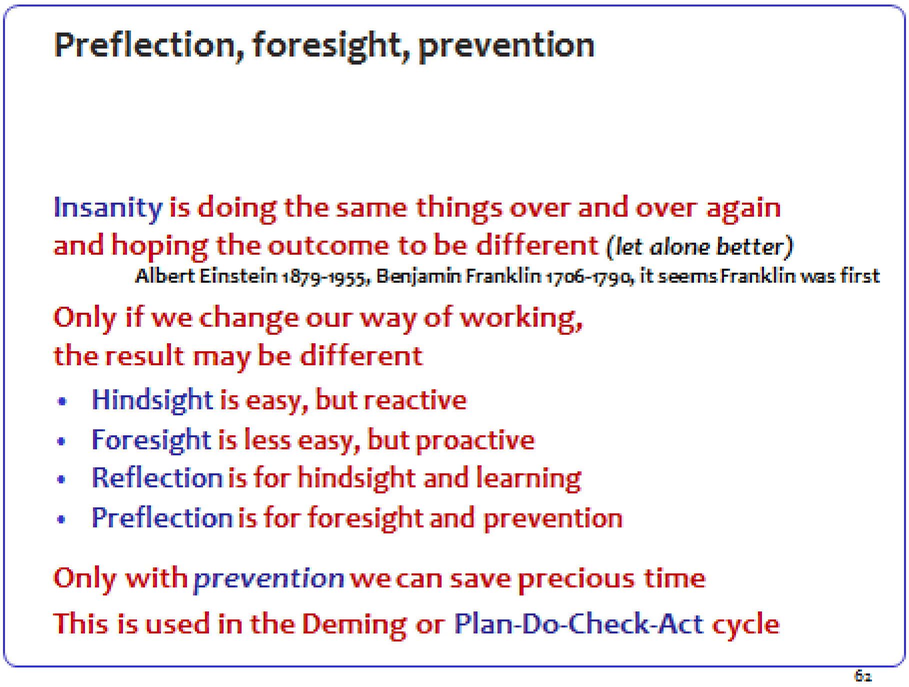
Murphy's Law for Professionals:

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*

61



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

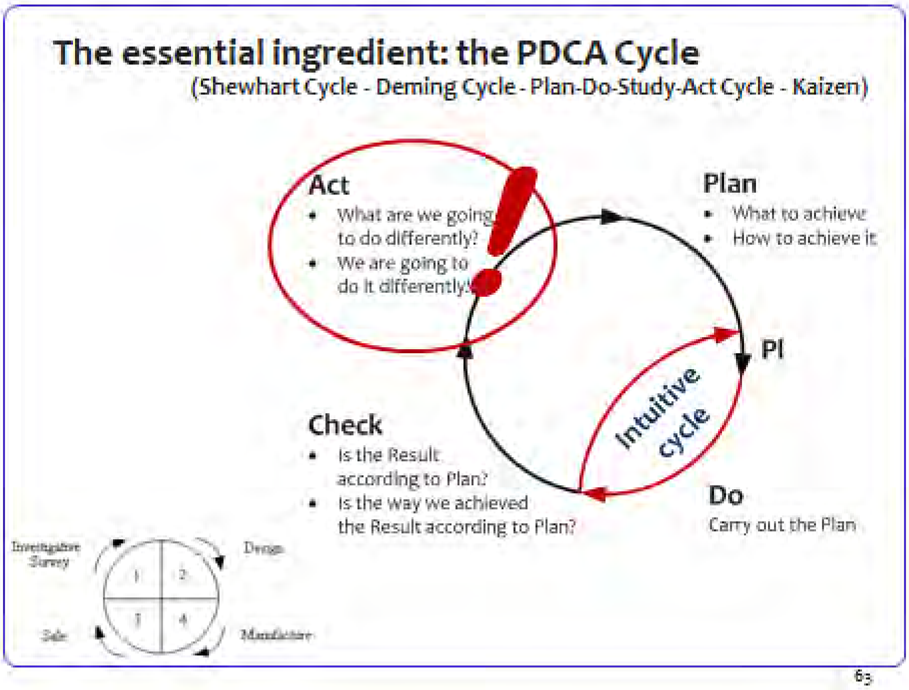
Only if we change our way of working, the result may be different

- 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

62



The essential ingredient: the PDCA Cycle

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

Act
- What are we going to do differently?
- We are going to do it differently!

Plan
- What to achieve
- How to achieve it

Pl

Intuitive cycle

Check
- Is the Result according to Plan?
- Is the way we achieved the Result according to Plan?

Do

Carry out the Plan

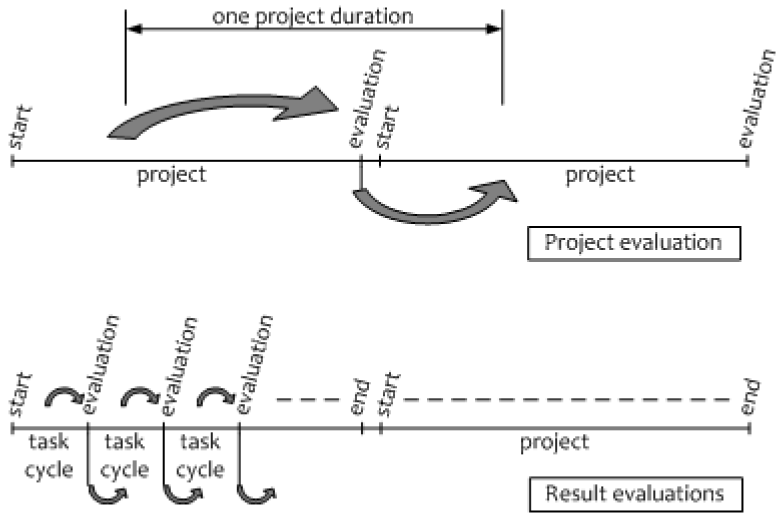
63

Lean things

- Plan-Do-Check-Act cycle was by far the most important thing we did in hindsight (Tom Harada)
- 90 per cent of all corporate problems can be solved using common sense and improving quality while reducing cost through the elimination of waste
  Imai: *Gemba Kaizen* - A Commonsense Low-Cost Approach to Management
- Root-Cause-Analysis on every defect found ?
  We don't have time for that ! (project manager)

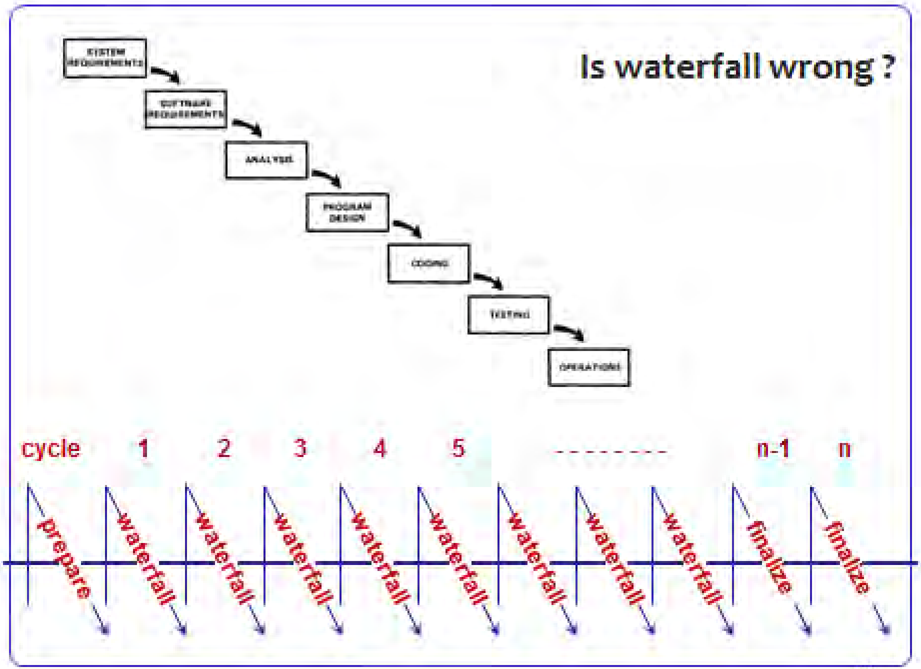
64

Project evaluations



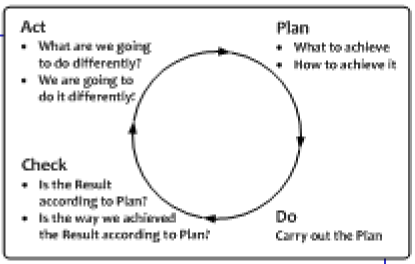
65

Is waterfall wrong ?



66

Knowledge
how to achieve the goal



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

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

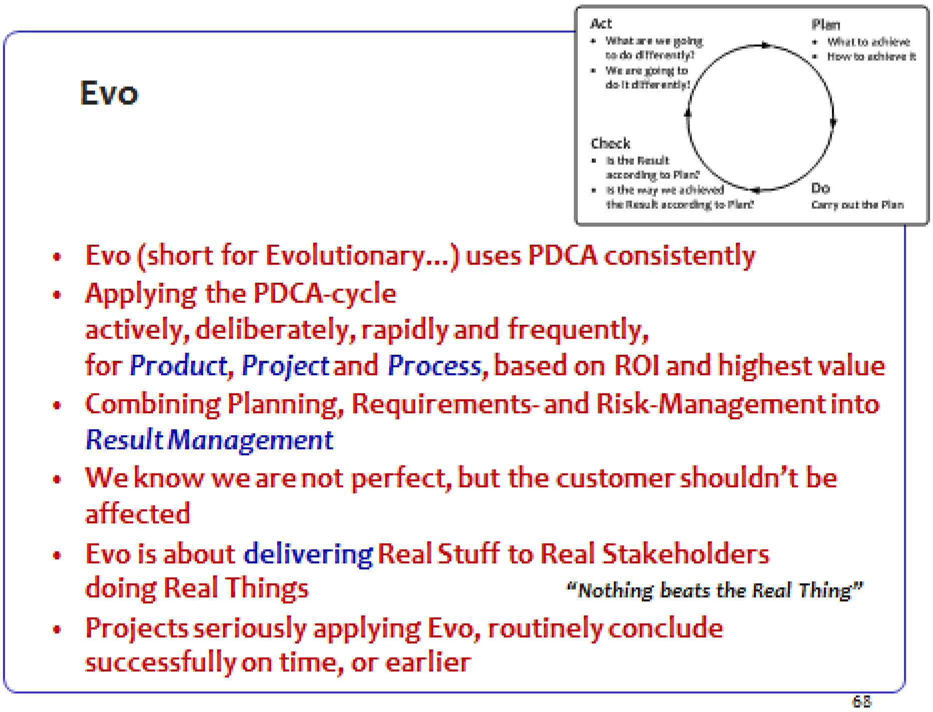
doing the right things

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

67

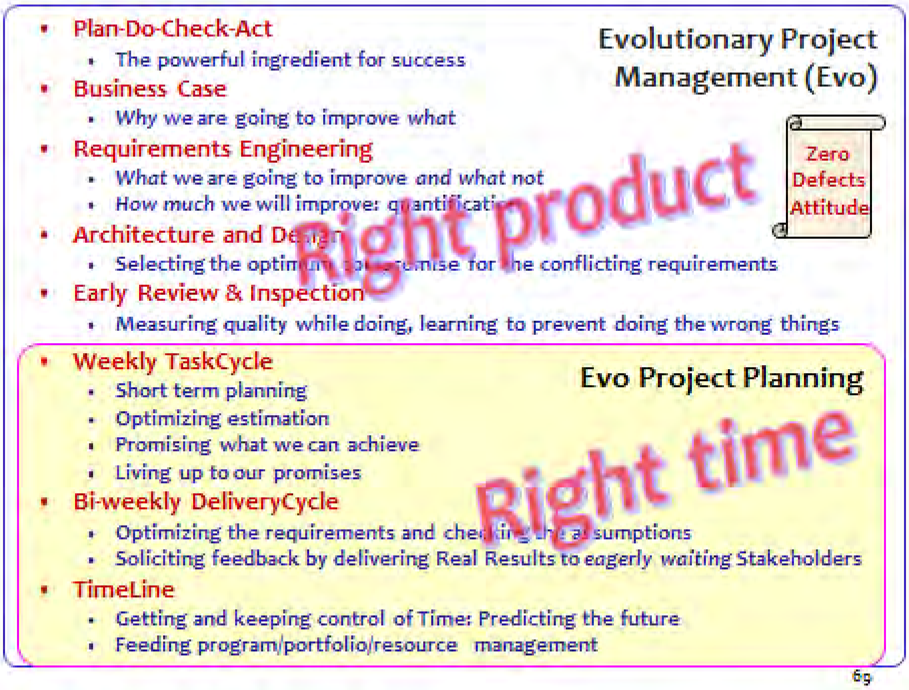


Evo

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

- 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

68



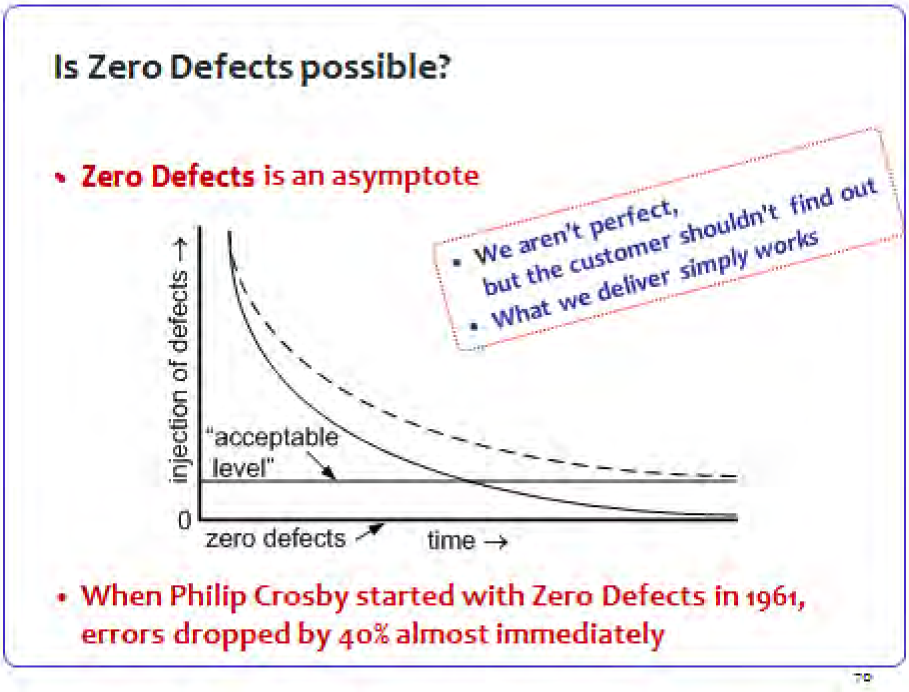
Evolutionary Project Management (Evo)

- 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

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

69



Is Zero Defects possible?

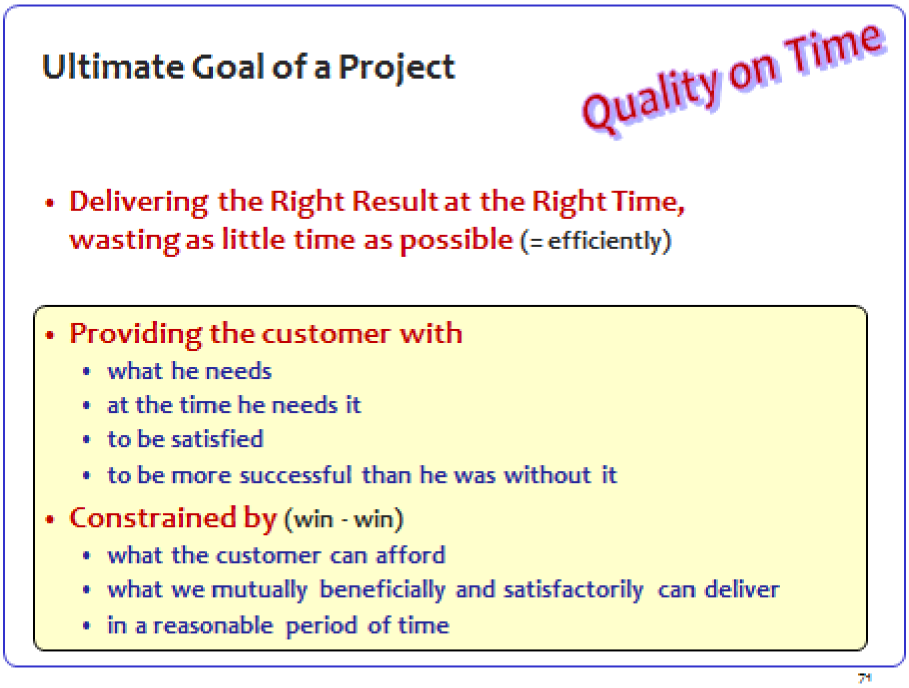
- **Zero Defects** is an asymptote

injection of defects →
0
"acceptable level"
zero defects
time →

- 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

70



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

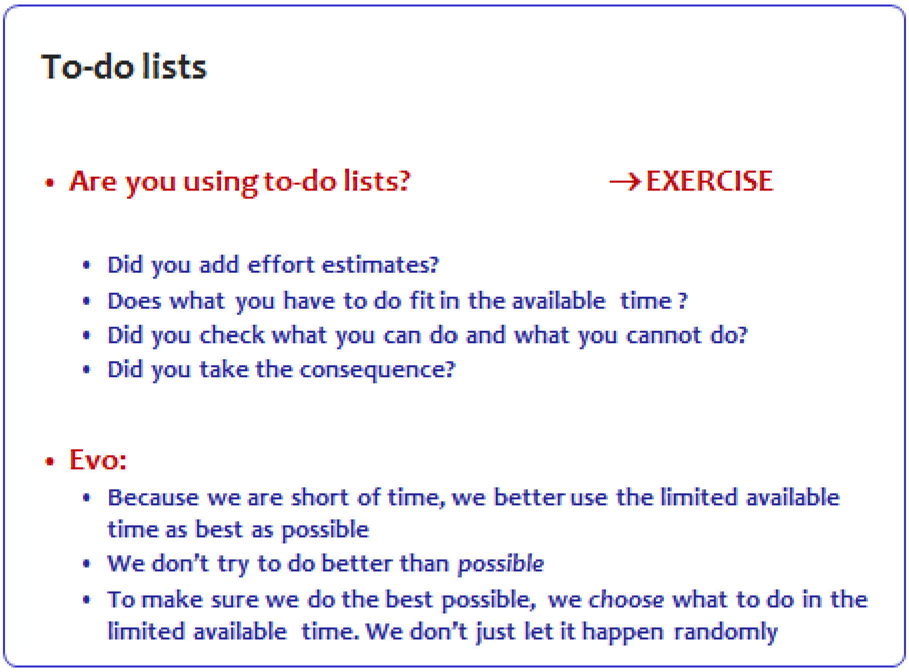
71

Evolutionary Planning

TaskCycle

DeliveryCycle

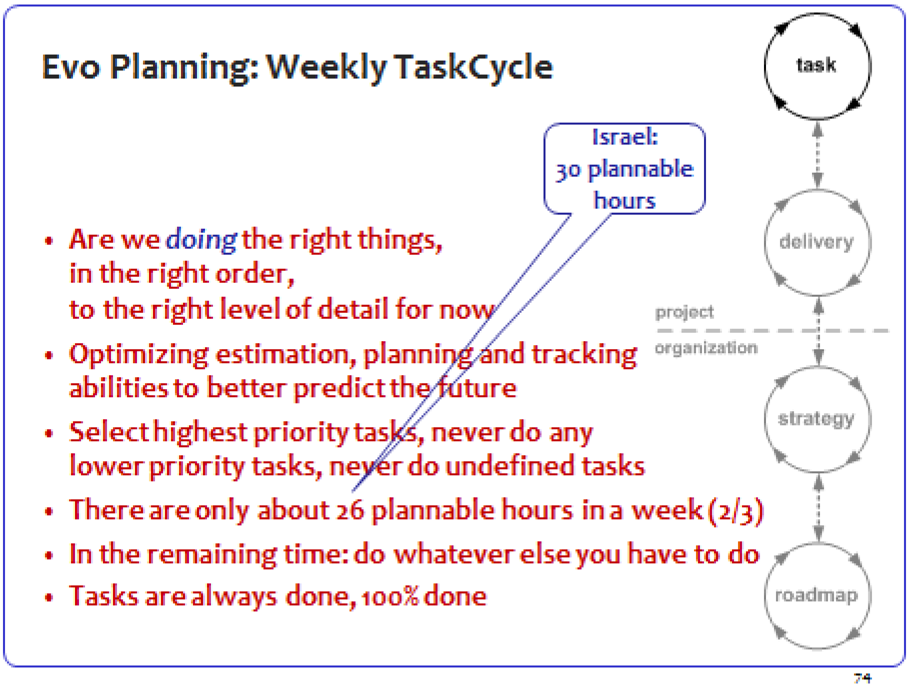
72



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

73

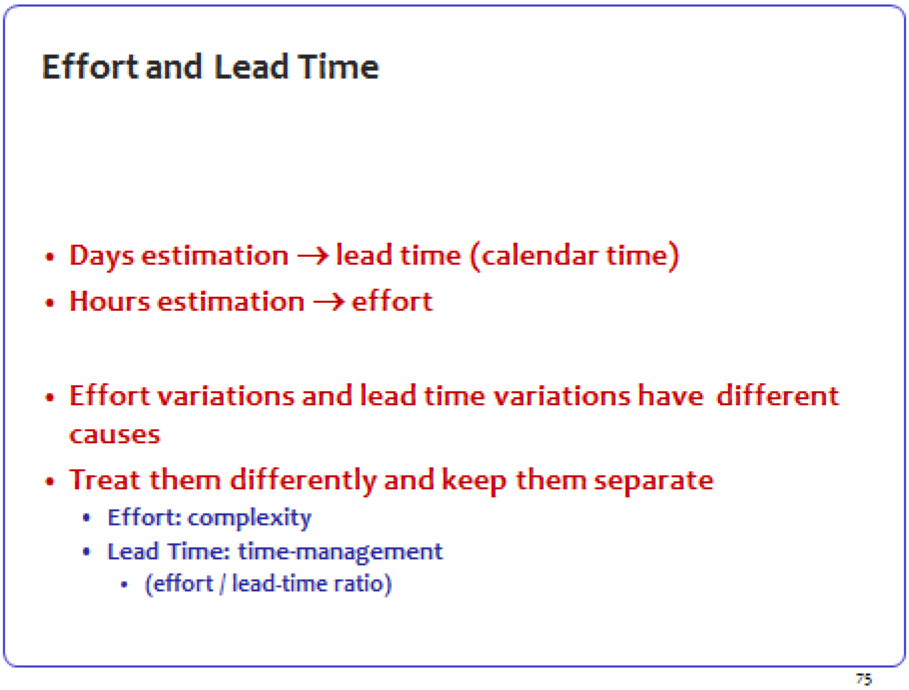


Evo Planning: Weekly TaskCycle

Israel: 30 plannable hours

- 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

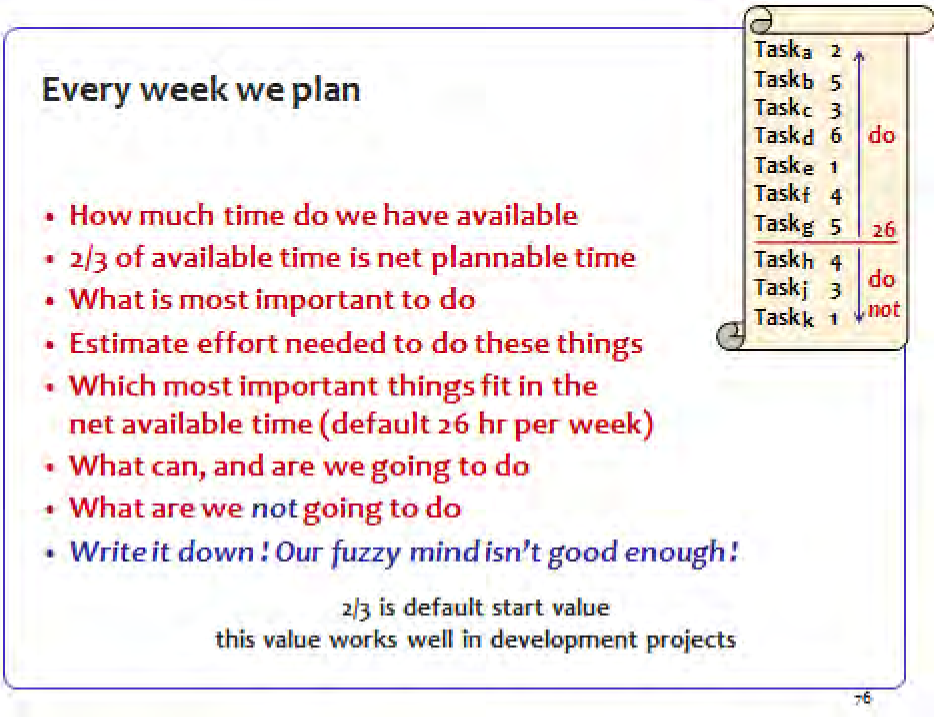
74



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)

75



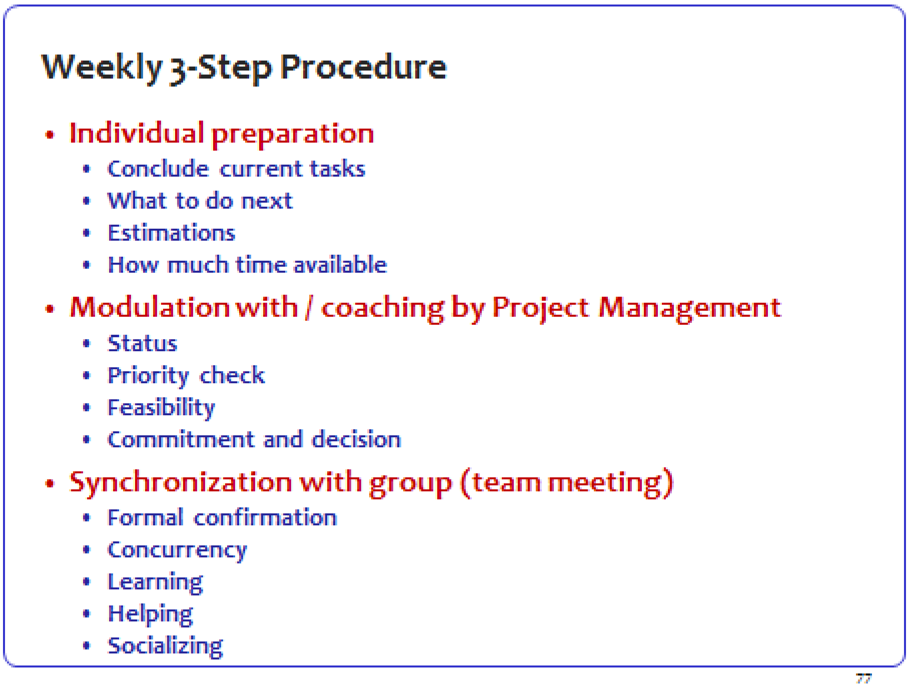
Every week we plan

- 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 !*

Task a	2	do
Task b	5	
Task c	3	
Task d	6	
Task e	1	
Task f	4	
Task g	5	26
Task h	4	do not
Task j	3	
Task k	1	

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

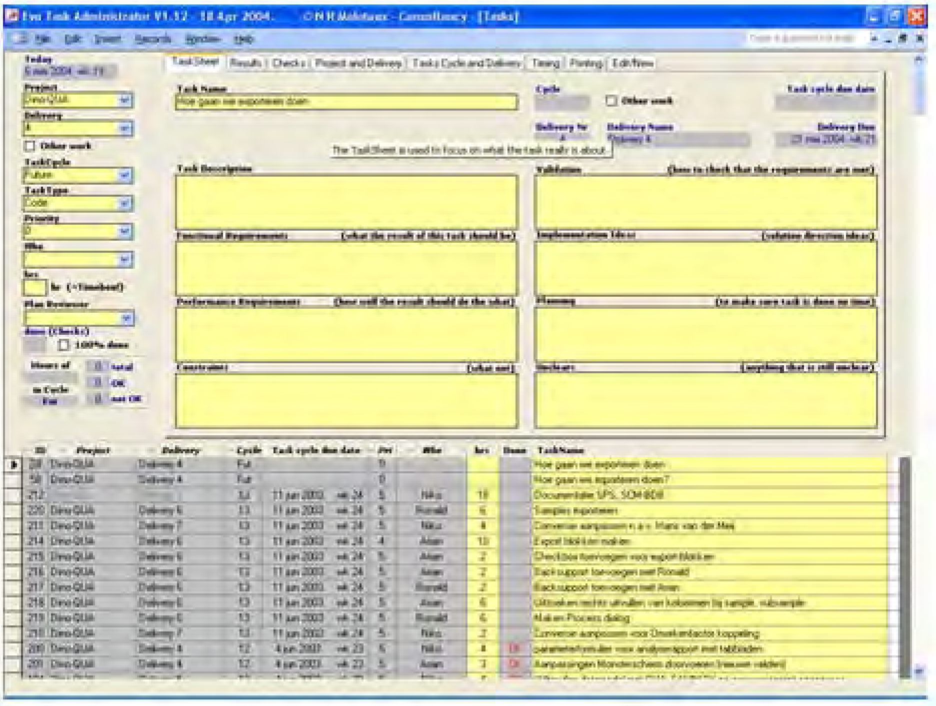
76



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

77



Evo Task Administrator V1.12 - 18 Apr 2004. © N R Malotaux - Consultancy - [Tasks]

File Edit Insert Records Window Help
Task Sheet | Results | Checks | Project and Delivery | Tasks Cycle and Delivery | Timing | Printing | Edit/New

Today: 6 mei 2004 wk 19
Project: Dino-QUA
Delivery: 4
☐ Other work
TaskCycle: Future
TaskType: Code
Priority: 0
Who:
hrs: hr (+Timesheet)
Plan Reviewer:
done (Checks) ☐ 100% done
Hours of in Cycle Fut: 0 total, 0 OK, 0 not OK

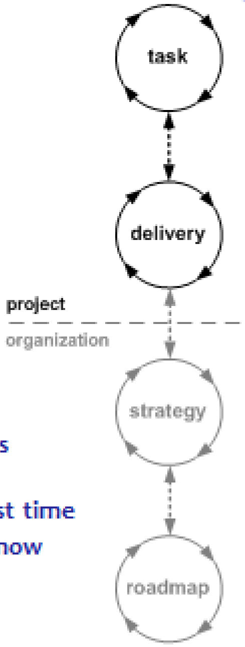
Task Name: Hoe gaan we exporteren doen
Cycle: ☐ Other work
Task cycle due date:
Delivery Nr: 4
Delivery Name: Delivery 4
Delivery Due: 21 mei 2004 wk 21

The TaskSheet is used to focus on what the task really is about.

Task Description
Validation (how to check that the requirements are met)
Functional Requirements (what the result of this task should be)
Implementation Ideas (solution direction ideas)
Performance Requirements (how well the result should do the what)
Planning (to make sure task is done on time)
Constraints (what not)
Unclear (anything that is still unclear)

ID	Project	Delivery	Cycle	Task cycle due date	Pri	Who	hrs	Done	TaskName
28	Dino-QUA	Delivery 4	Fut		0				Hoe gaan we exporteren doen
58	Dino-QUA	Delivery 4	Fut		0				Hoe gaan we exporteren doen?
212			14	11 jun 2003 wk 24	5	Niko	18		Documentatie SPS, SOM BDB
220	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Samples exporteren
211	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	4		Conversie aanpassen n.a.v. Hans van der Meij
214	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	4	Alain	10		Export blokken maken
215	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Alain	2		Checkbox toevoegen voor export blokken
216	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Alain	2		Back support toevoegen met Ronald
217	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	2		Back support toevoegen met Alain
218	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Alain	6		Uitzoeken rechts uitvullen van kolommen bij sample, subsample
219	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Maken Process dialog
210	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	2		Conversie aanpassen voor Omrekenfactor koppeling
200	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Niko	8	Ok	parameterformulier voor analyserapport met tabbladen
201	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Alain	3	Ok	Aanpassingen Monsterschema doorvoeren (nieuwe velden)

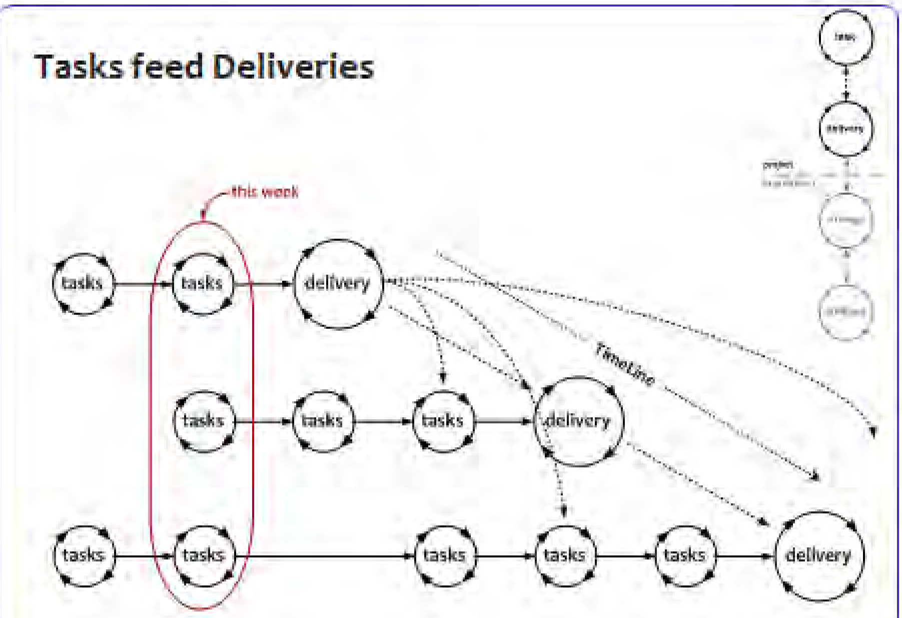
DeliveryCycle



- 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

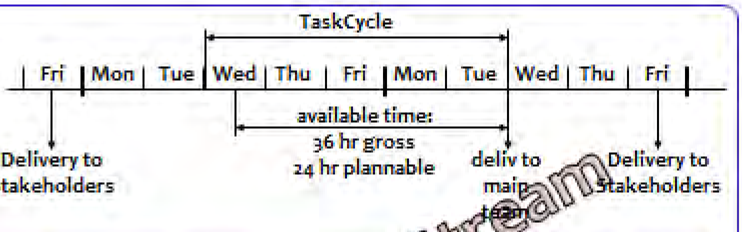
79

Tasks feed Deliveries



80

Designing a Delivery



Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	[illegible]
-	5
Total	24

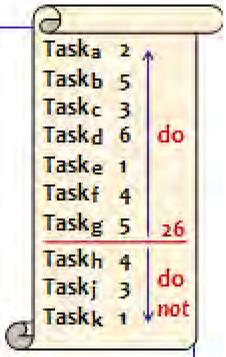
Gregory	
Draft design	6
Finish design	0
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	1
XMLb	1
Total	24

Gregory (later)	
Draft design	0
Finish design	0
...	

Jerome	
XMLa	3
XMLb	3
...	

81

TaskCycle Exercise



Task$_a$	2
Task$_b$	5
Task$_c$	3
Task$_d$	6
Task$_e$	1
Task$_f$	4
Task$_g$	5
Task$_h$	4
Task$_j$	3
Task$_k$	1

do

26

do not

- 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 (30) 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 ?

82

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 ...

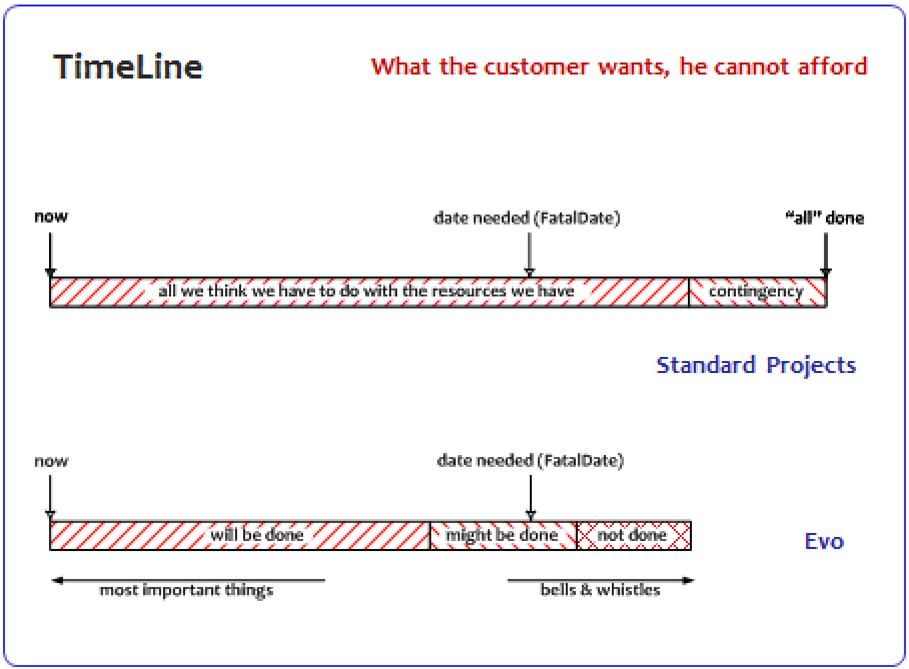
- How do we make sure the whole project is done on time ?

83

Evolutionary Planning

TimeLine

84



TimeLine

What the customer wants, he cannot afford

now

date needed (FatalDate)

"all" done

all we think we have to do with the resources we have

contingency

Standard Projects

now

date needed (FatalDate)

will be done

might be done

not done

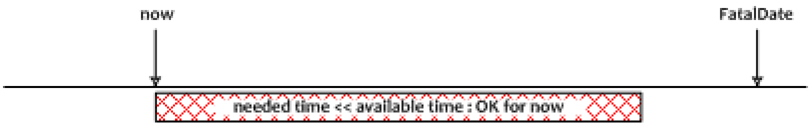
Evo

most important things

bells & whistles

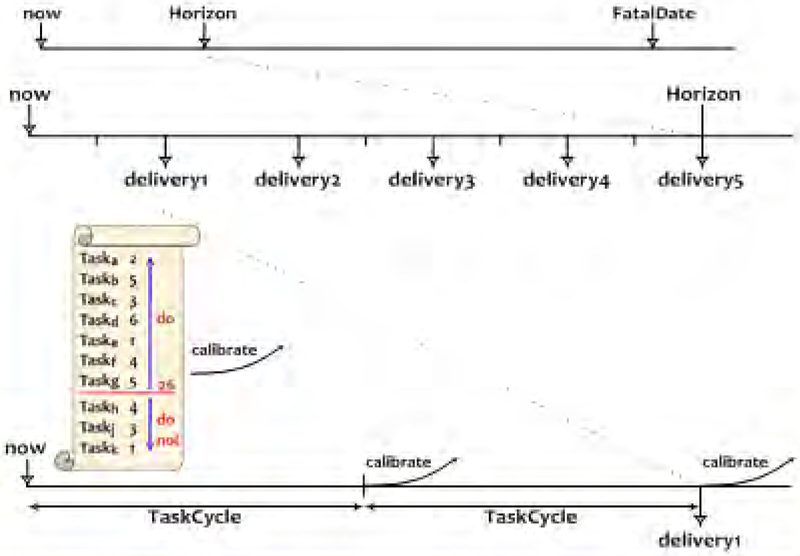
85

If it easily fits ...

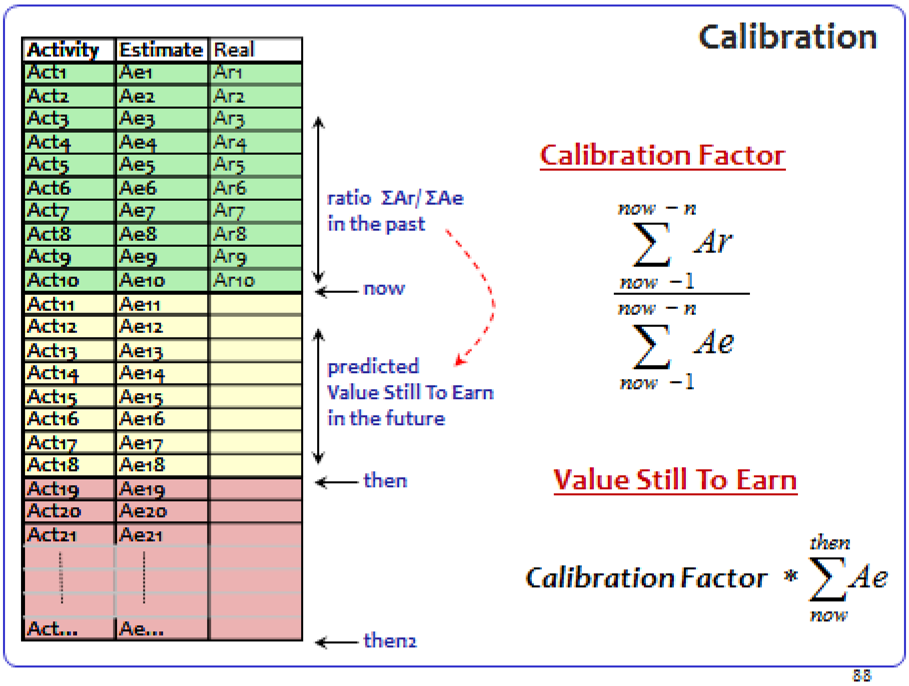


86

Result to Tasks and back



87



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...	

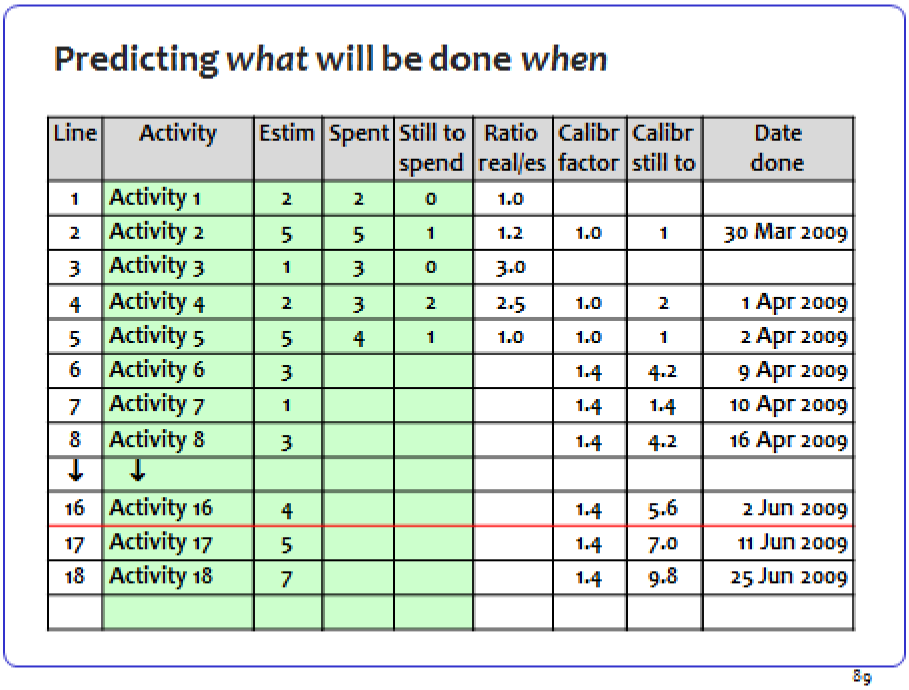
Calibration Factor

$$\frac{\sum_{now\ -1}^{now\ -n} Ar}{\sum_{now\ -1}^{now\ -n} Ae}$$

Value Still To Earn

$$Calibration\ Factor\ *\ \sum_{now}^{then} Ae$$

88



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

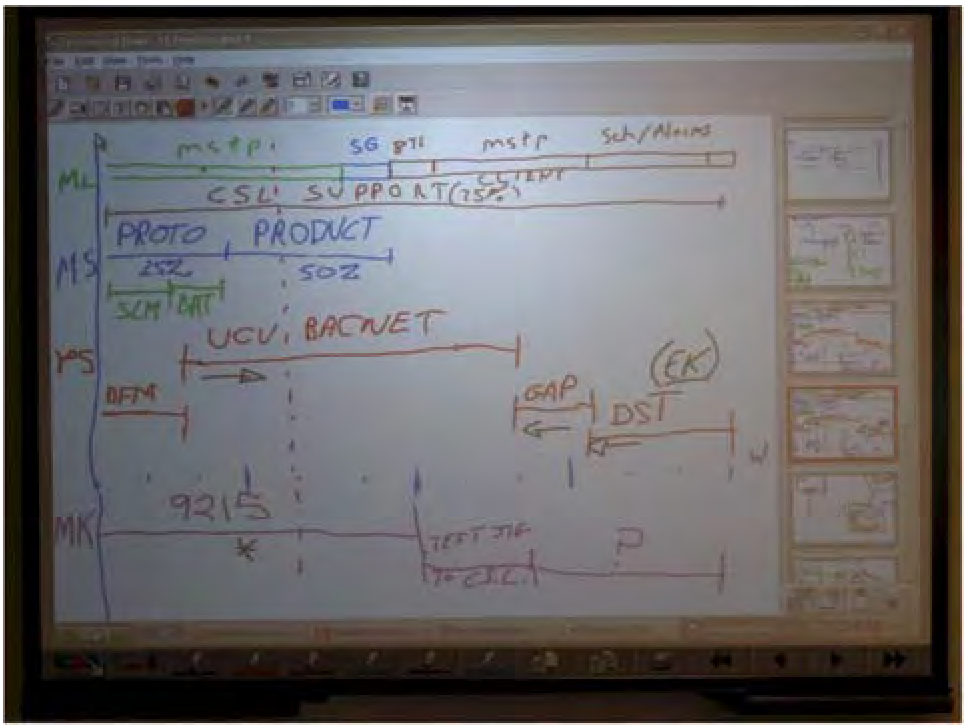
89

Product/Portfolio/Resource Management

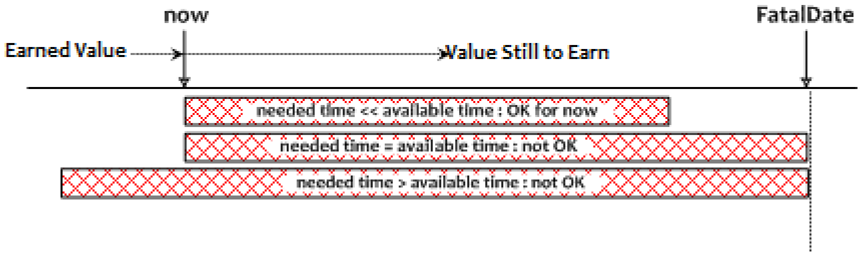
- Current Program/Portfolio/Resource Management is based on hope
- More a game than management

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

90



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


now
FatalDate
Earned Value
Value Still to Earn
needed time << available time : OK for now
needed time = available time : not OK
needed time > available time : not OK

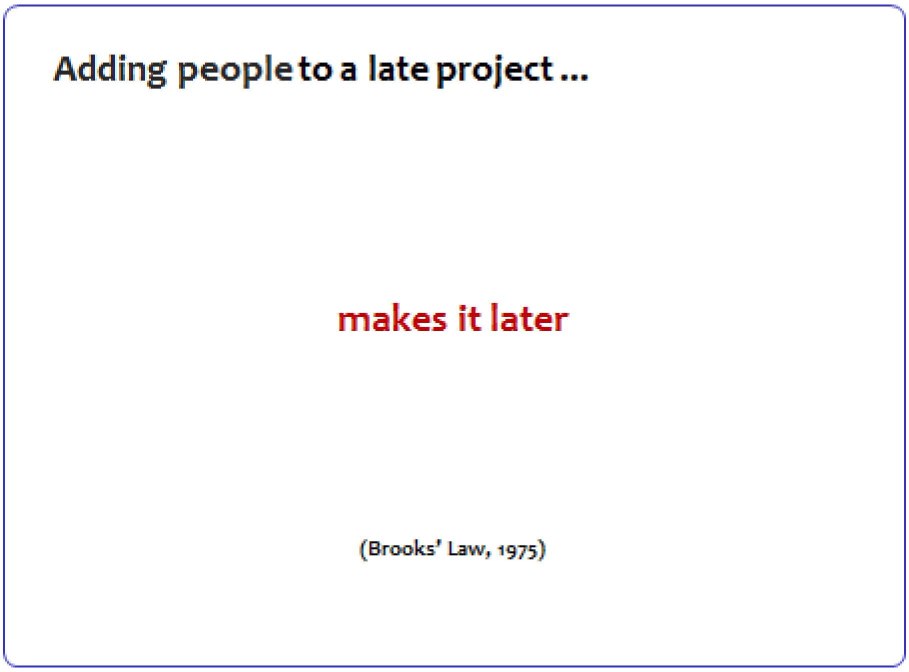
- If it doesn't fit ... count backwards
- If the match is over, you cannot score a goal

92

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

93

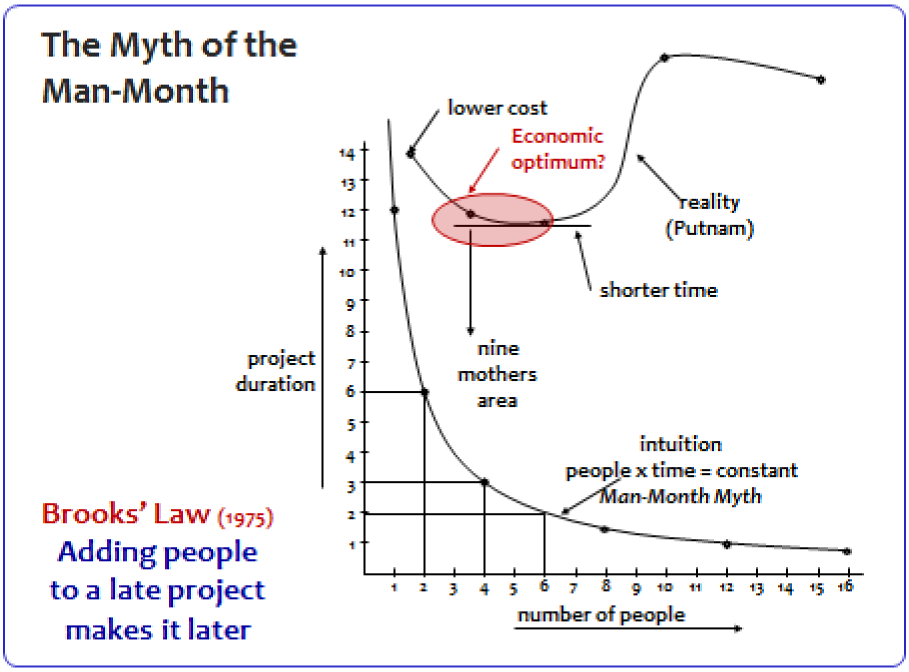


Adding people to a late project ...

makes it later

(Brooks' Law, 1975)

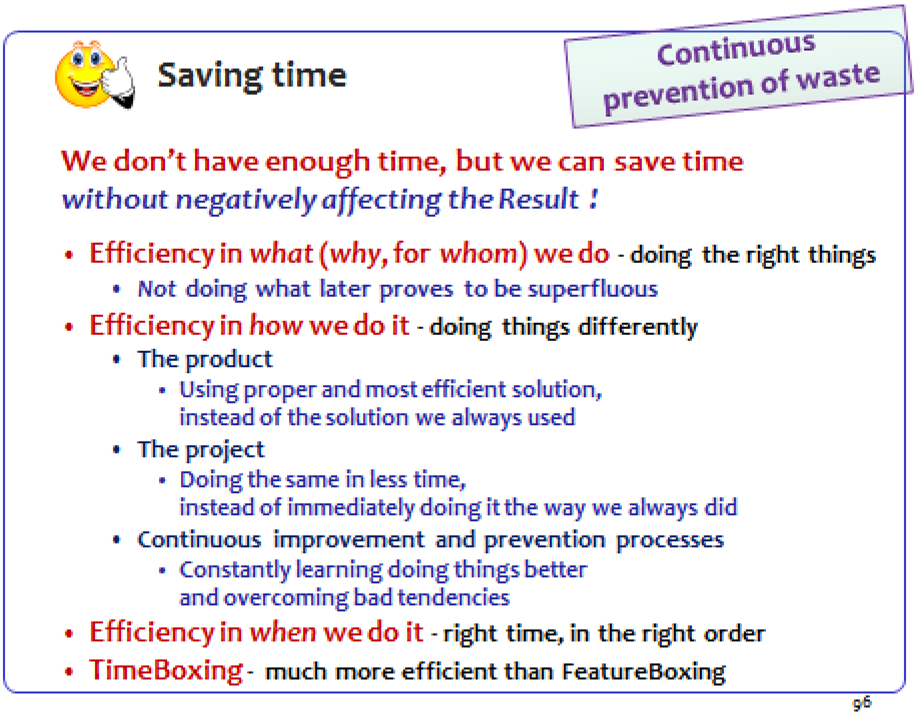
94



The Myth of the Man-Month

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

95



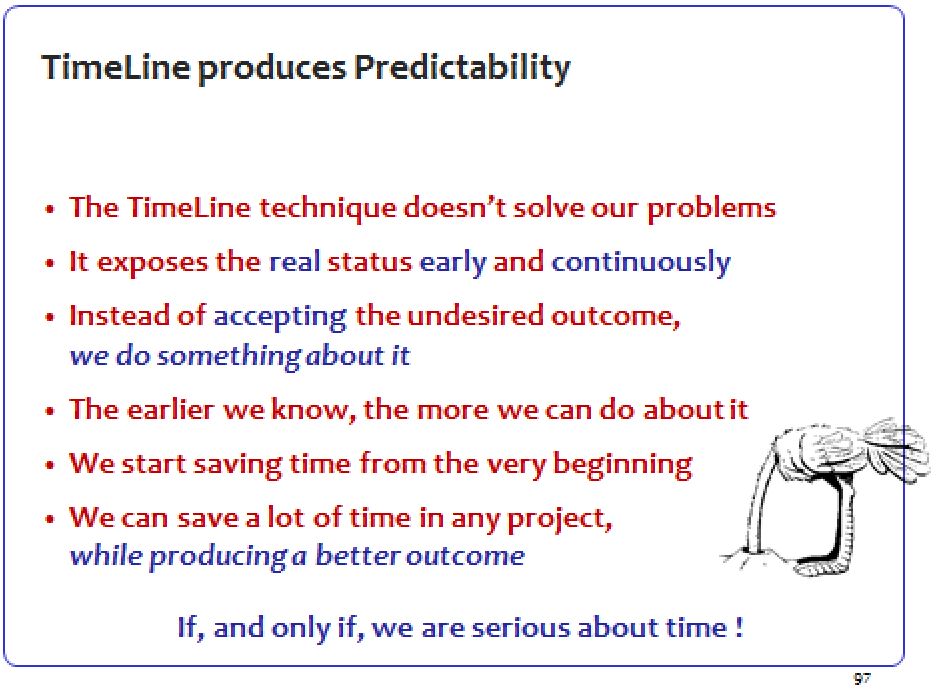
Saving time

Continuous prevention 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

96

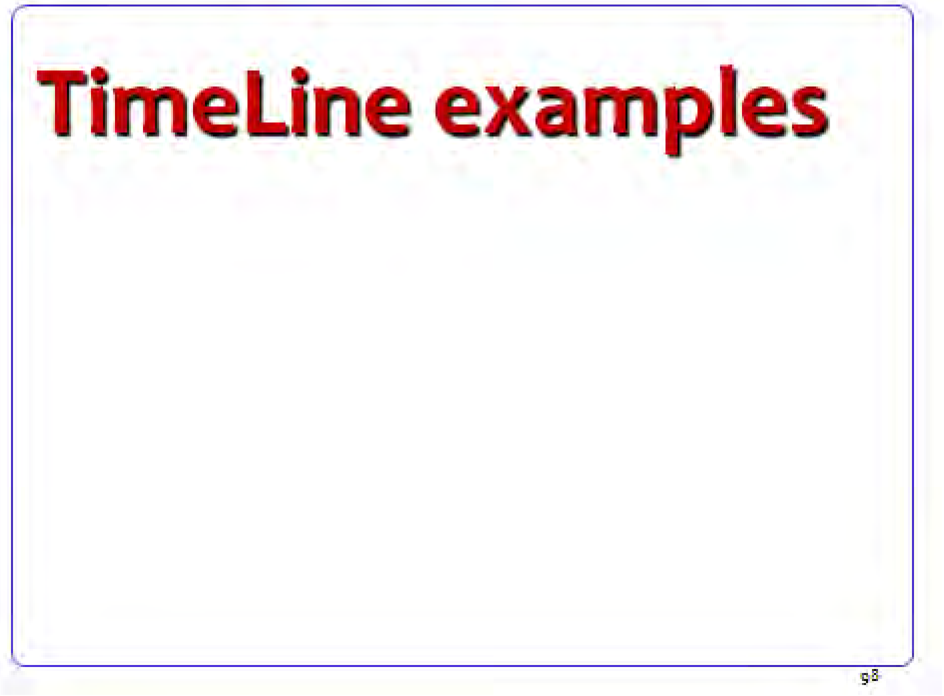


TimeLine produces Predictability

- The TimeLine technique doesn’t solve our problems
- It exposes 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*

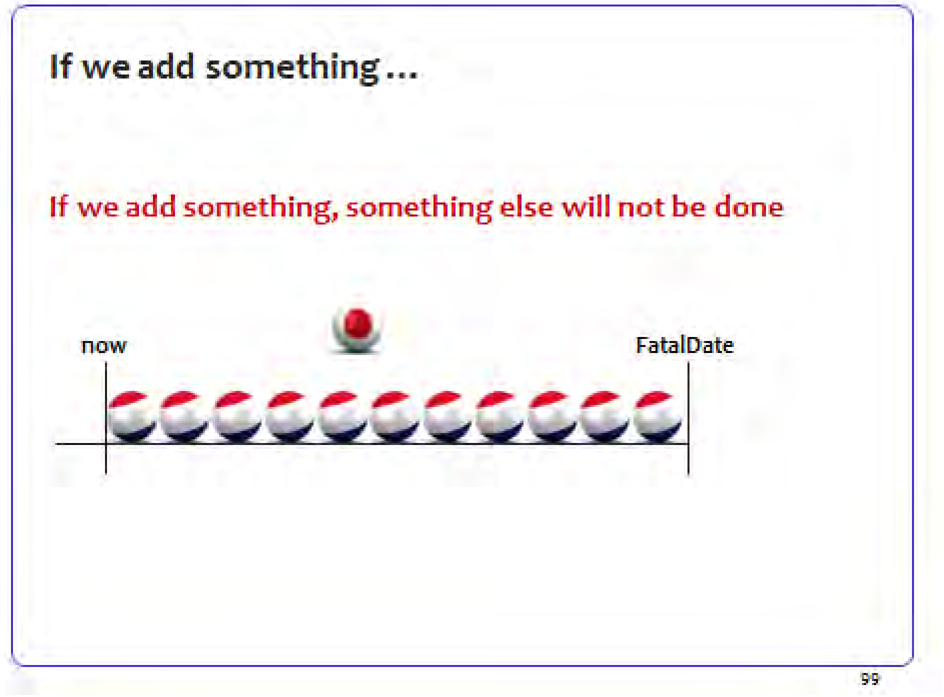
If, and only if, we are serious about time !

97



TimeLine examples

98

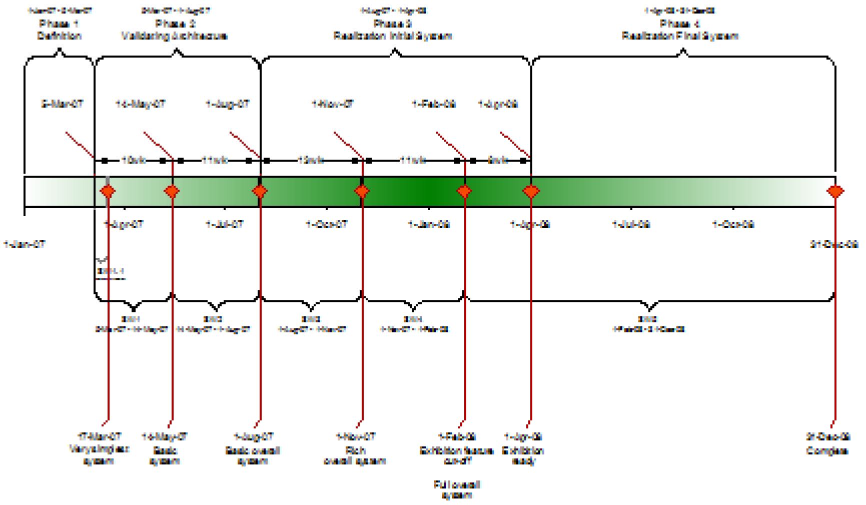


If we add something ...

If we add something, something else will not be done

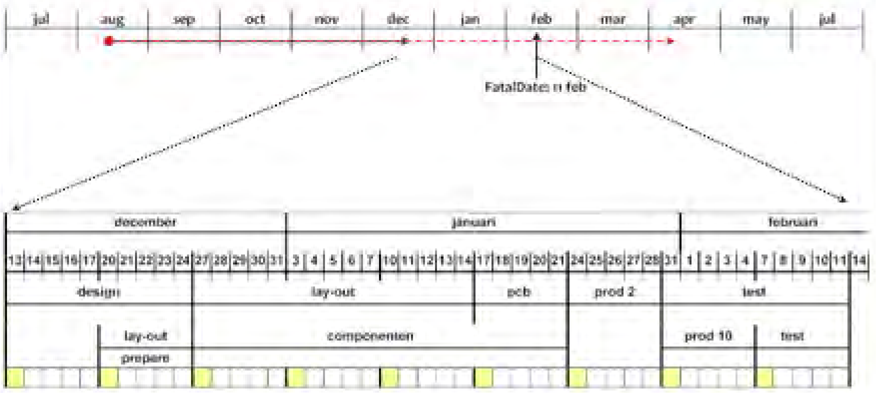
99

TimeLine example



100

TimeLine planning

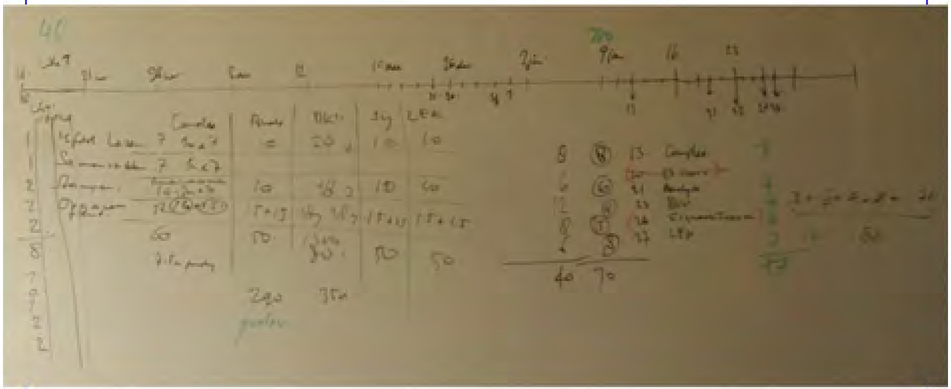


101

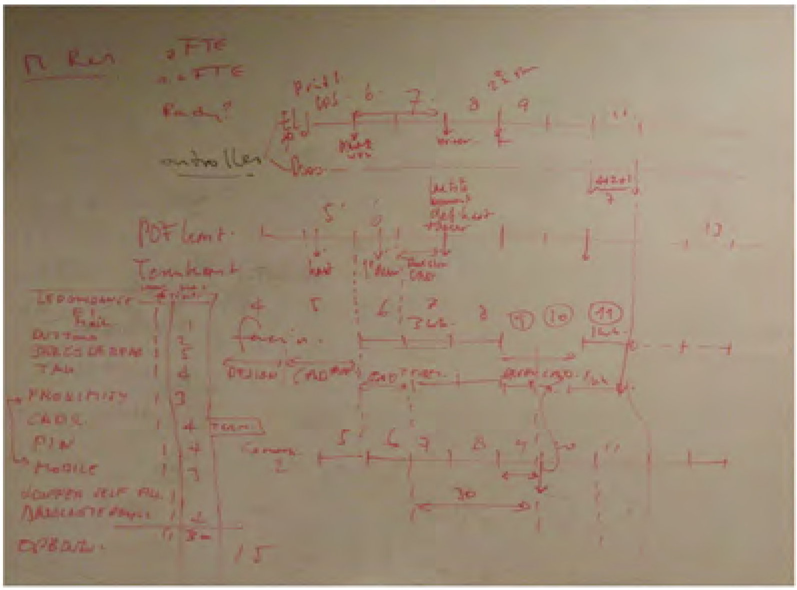
TimeLine exercise example

- Preparing for student exams

What we did

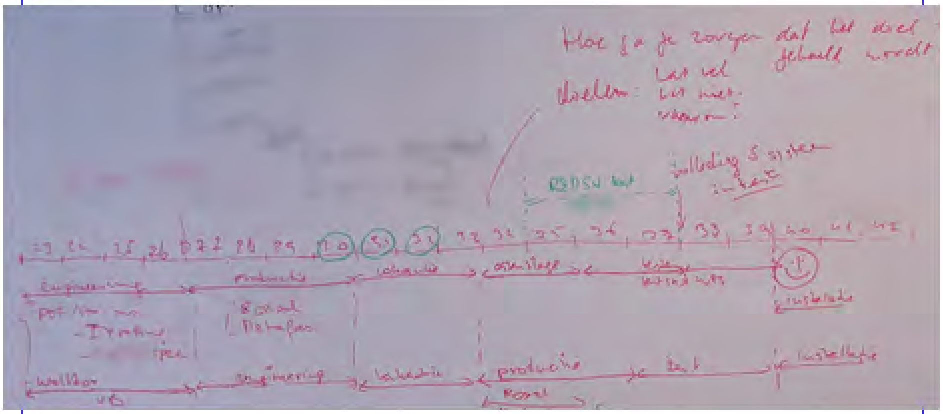


Whiteboard TimeLine Planning



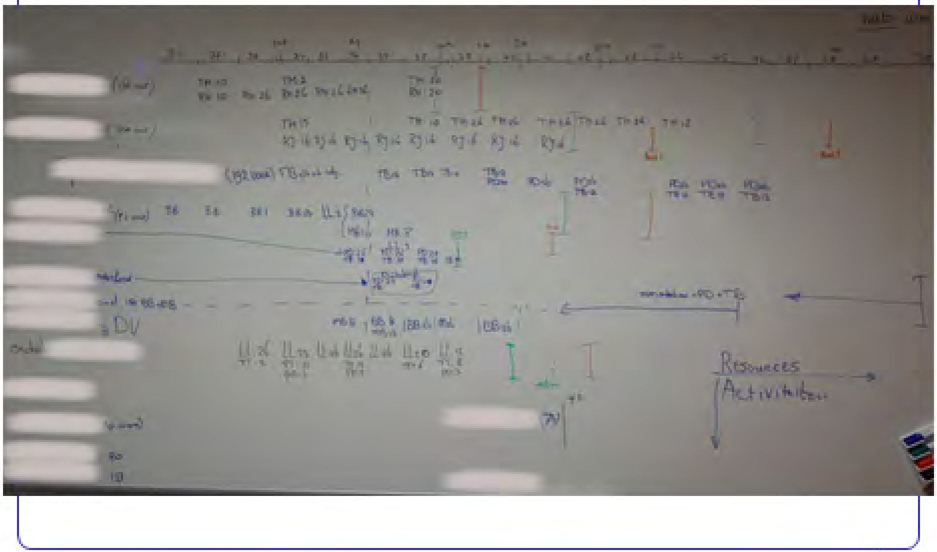
104

Whiteboard TimeLine Planning



105

Whiteboard TimeLine Resource Planning

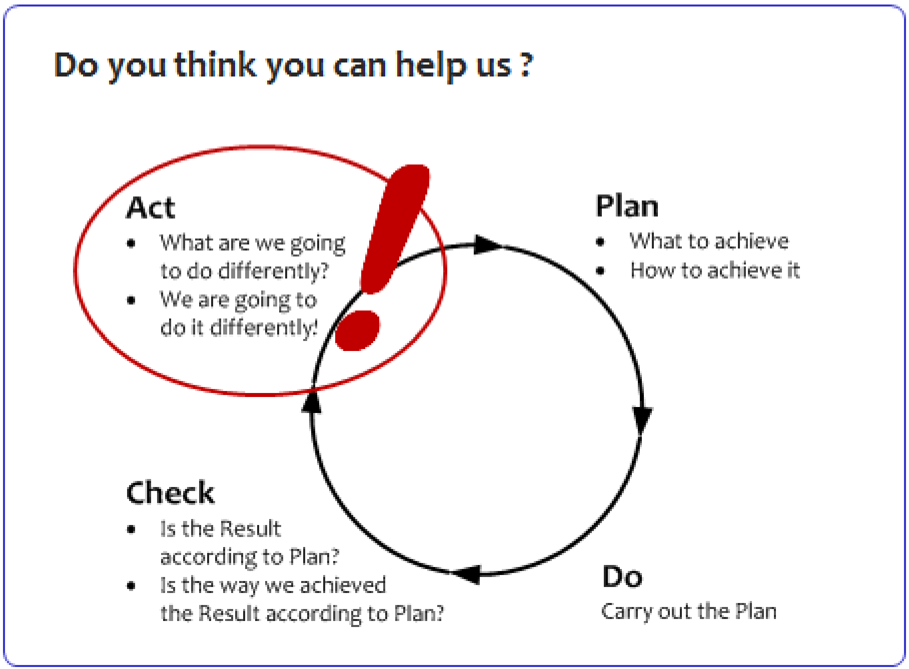


106

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?

107



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

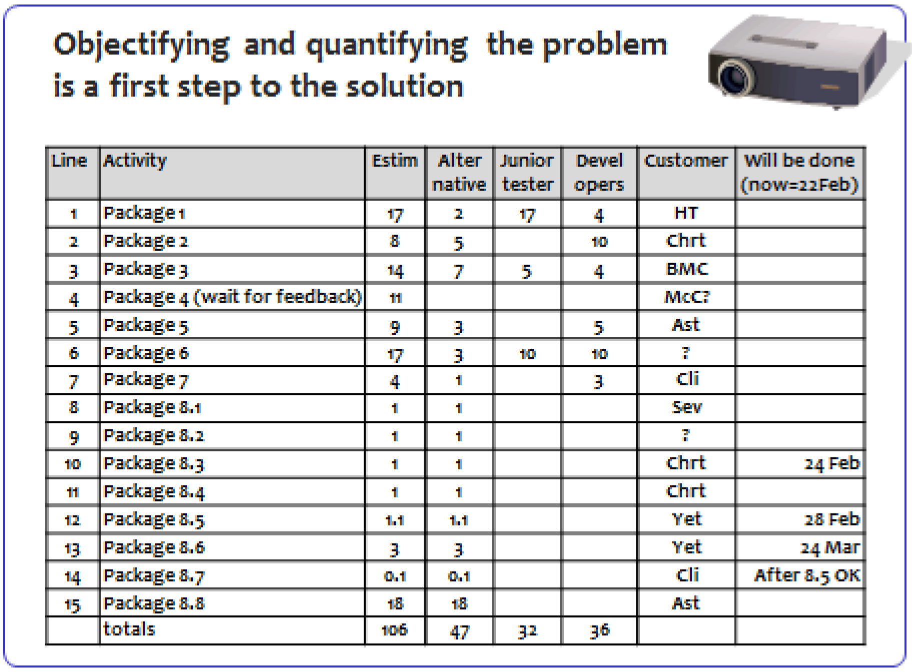
Carry out the Plan

In stead of complaining about a problem …

(Stuck in the Check-phase)

Let's do something about it !

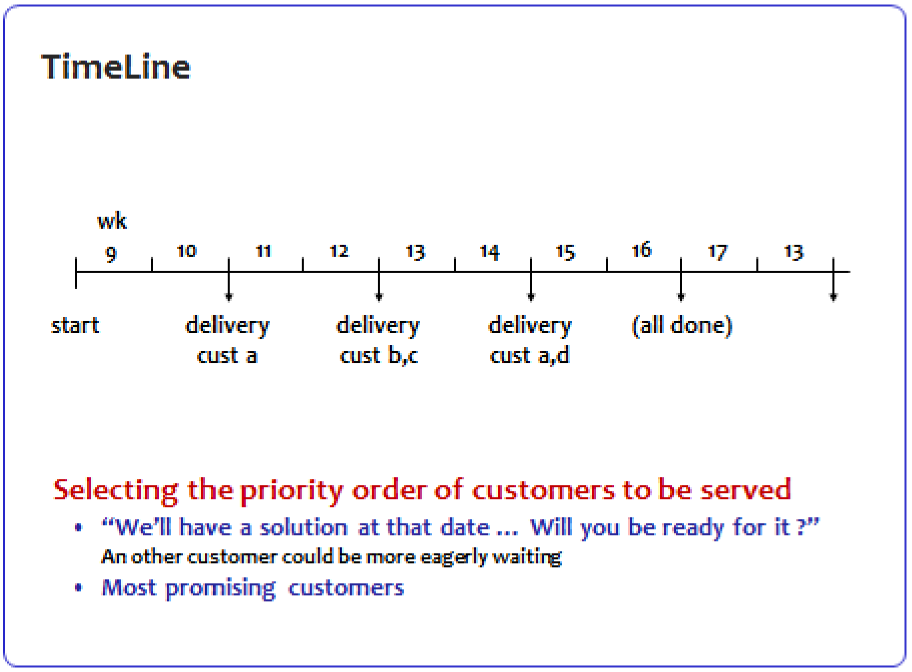
(Moving to the Act-phase)



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	1	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		

110



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

111

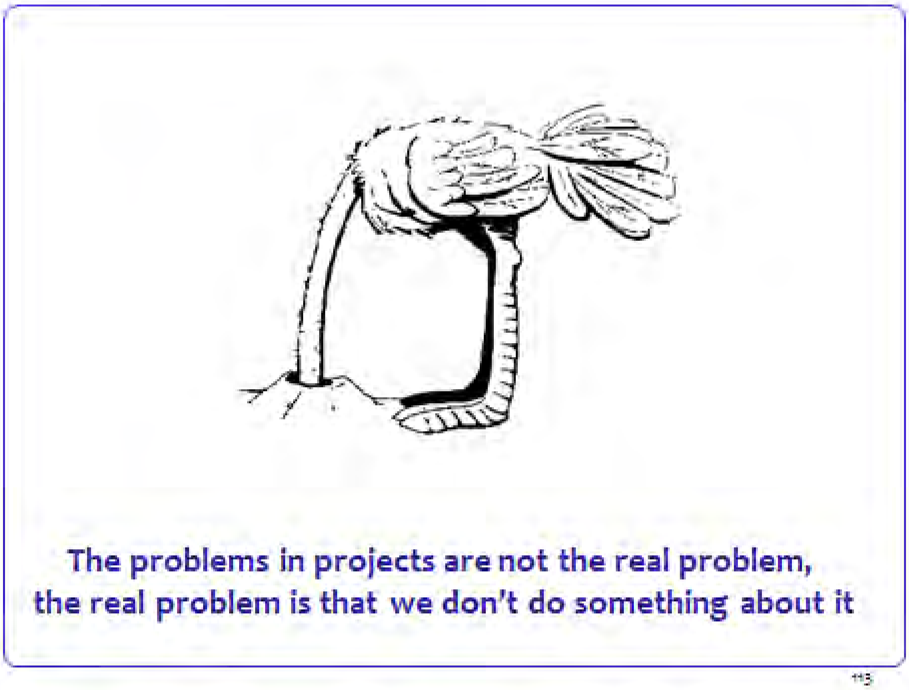
Result

- 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:

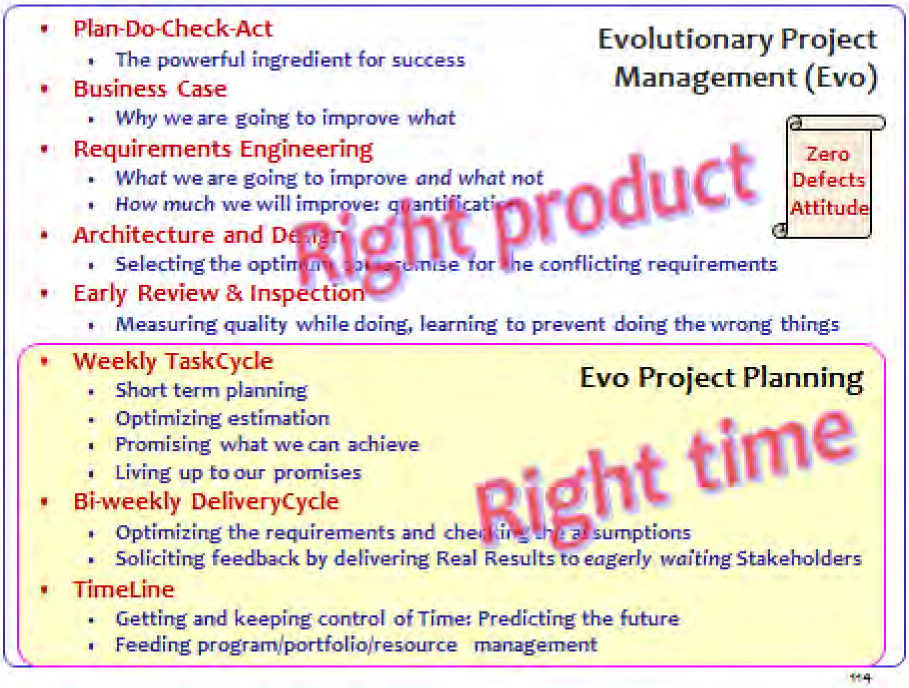
- Tester promoted to product manager
- Still coaching successors how to plan

112



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

113



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

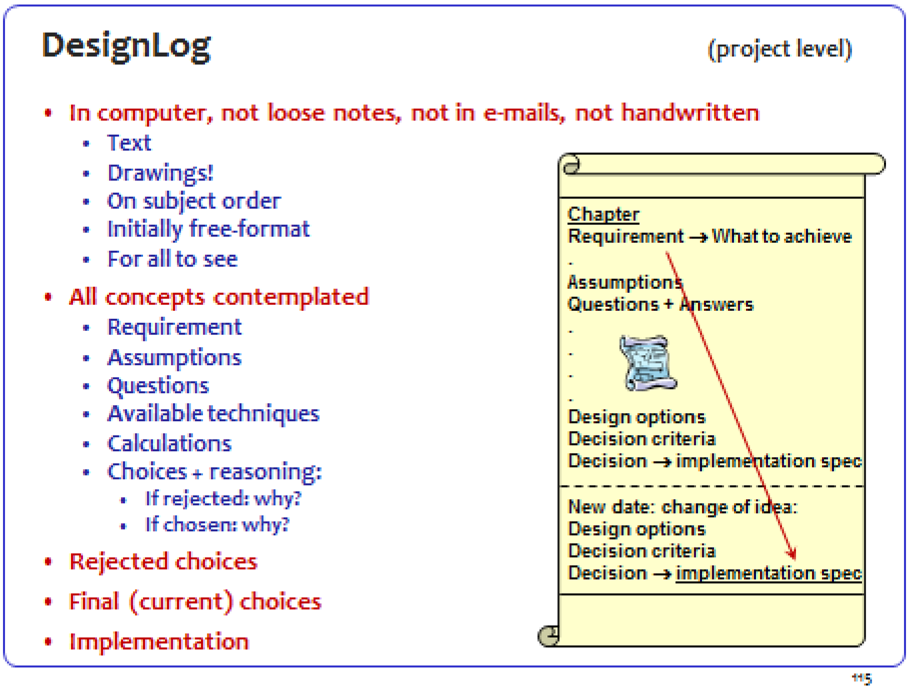
Right product

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

114



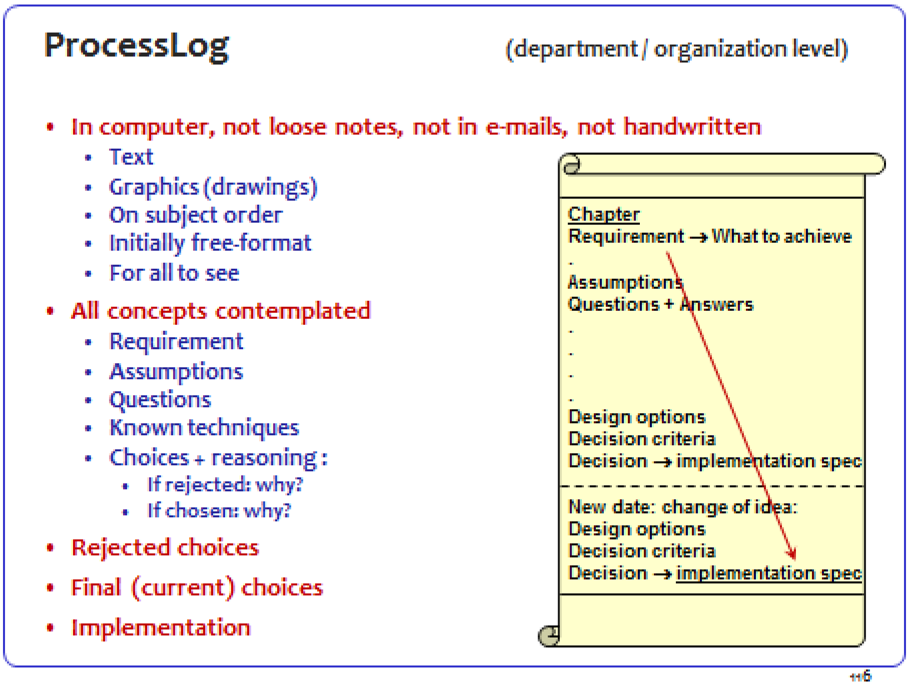
DesignLog (project level)

- 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

Chapter
Requirement → What to achieve
.
Assumptions
Questions + Answers
.
.
.
Design options
Decision criteria
Decision → implementation spec

New date: change of idea:
Design options
Decision criteria
Decision → implementation spec

115



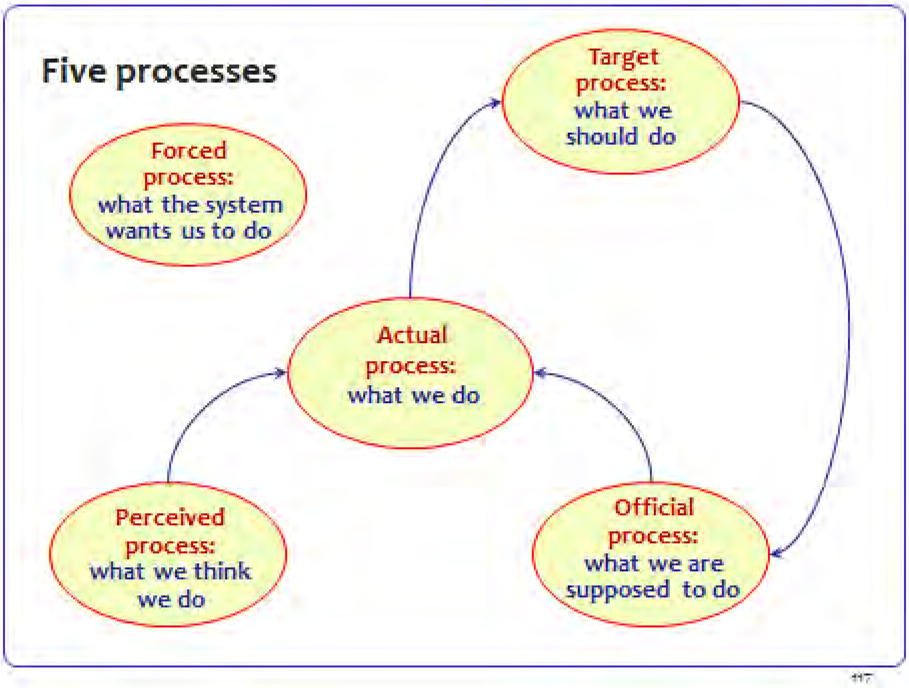
ProcessLog

(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

Chapter
Requirement → What to achieve
.
Assumptions
Questions + Answers
.
.
.
.
Design options
Decision criteria
Decision → implementation spec
New date: change of idea:
Design options
Decision criteria
Decision → implementation spec

116



Five processes

Forced process: what the system wants us to do

Target process: what we should do

Actual process: what we do

Perceived process: what we think we do

Official process: what we are supposed to do

117

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

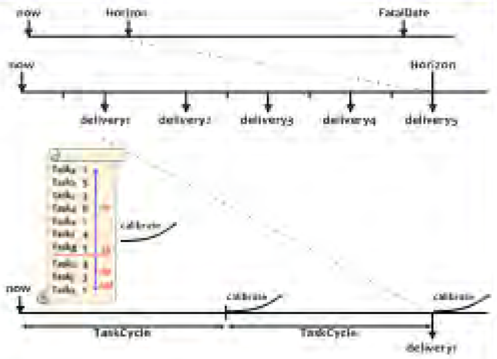
8 Help! We have a QA Problem! (2009)
Use of TimeLine technique: How we solved a 6 month backlog in 9 weeks

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

www.malotaux.nl/nrm/Insp
Inspection pages

118

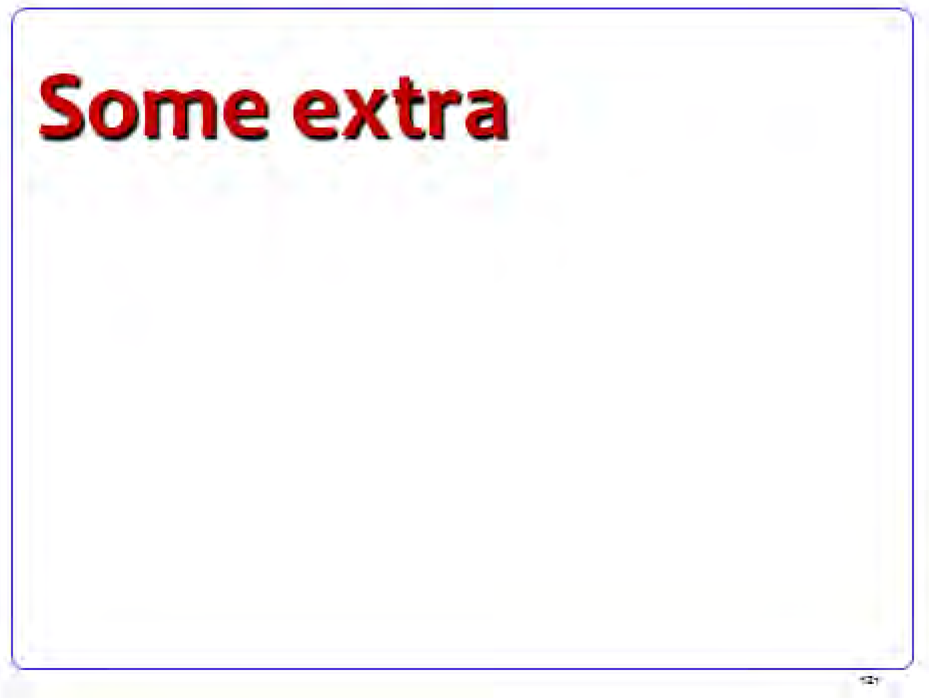
TimeLine exercise for your Project



- Try to describe the TimeLine for your project
- What do you have to do the coming 10 weeks
- Can you define the first few deliveries
  - What to do, for whom, and why
- Is your TaskCycle plan still adequate ?

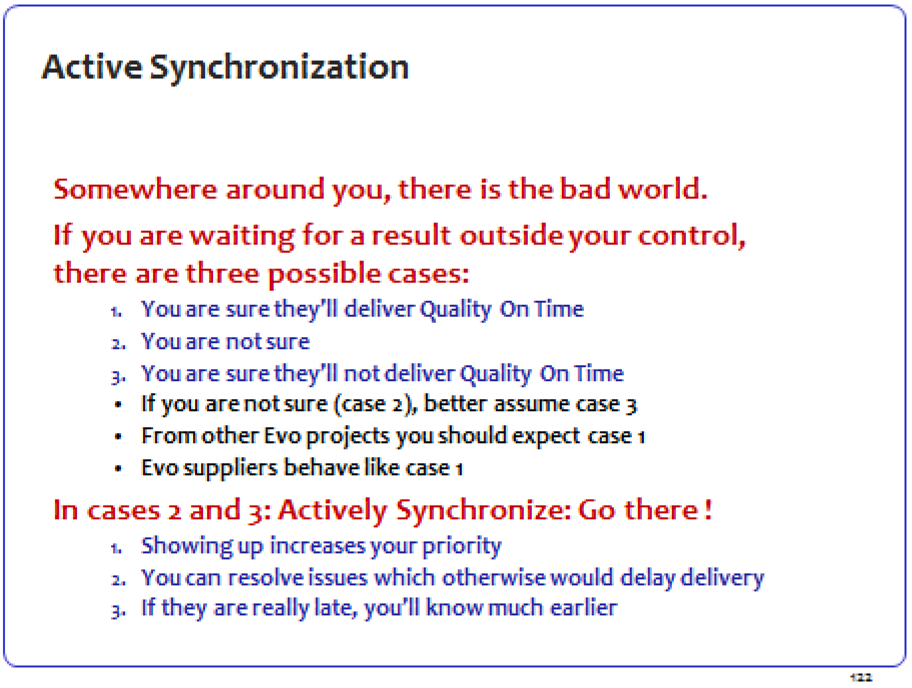
119

What now ?



Some extra

121



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:

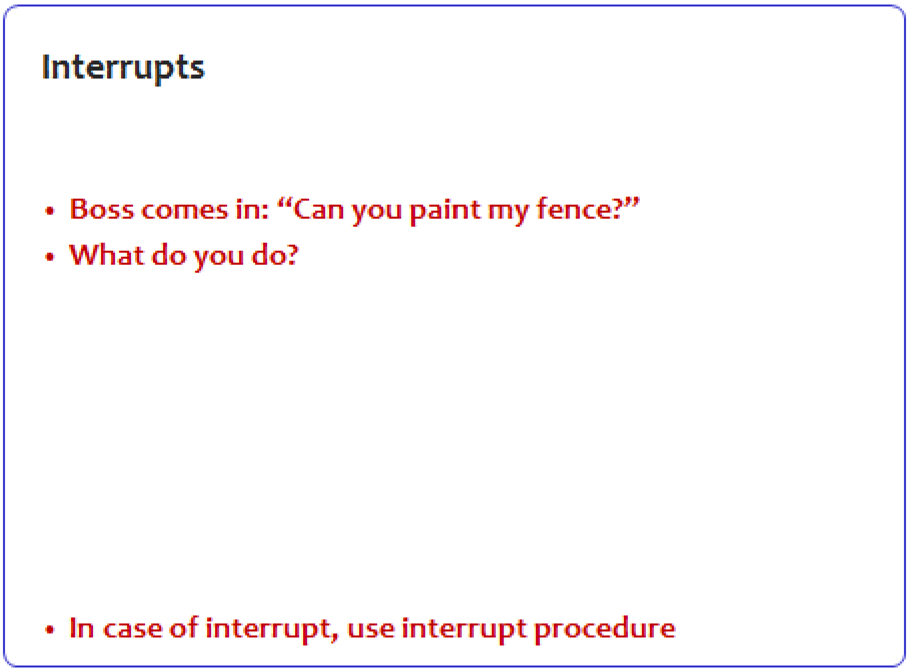
1. You are sure they'll deliver Quality On Time
2. You are not sure
3. You are sure they'll not deliver Quality On Time

- 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

122

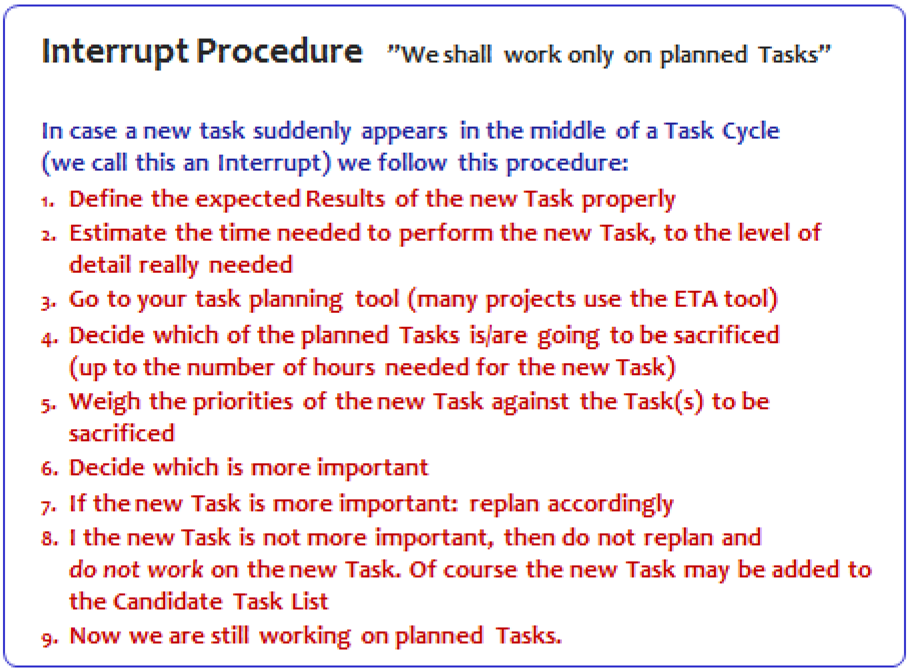


Interrupts

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

- In case of interrupt, use interrupt procedure

123

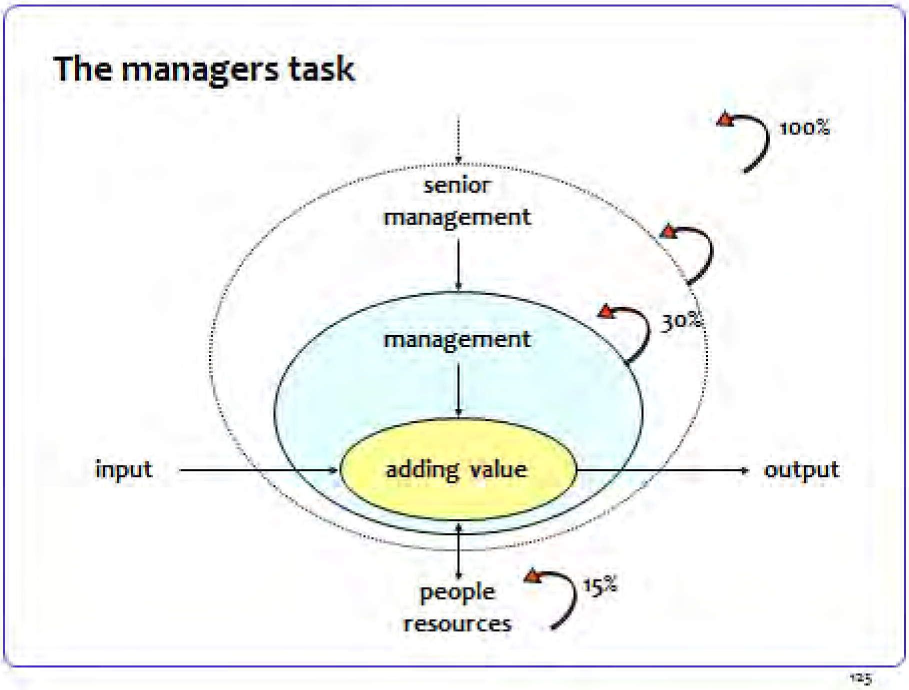


Interrupt Procedure ”We shall work only on planned Tasks”

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.

124



The managers task

senior management

management

adding value

input

output

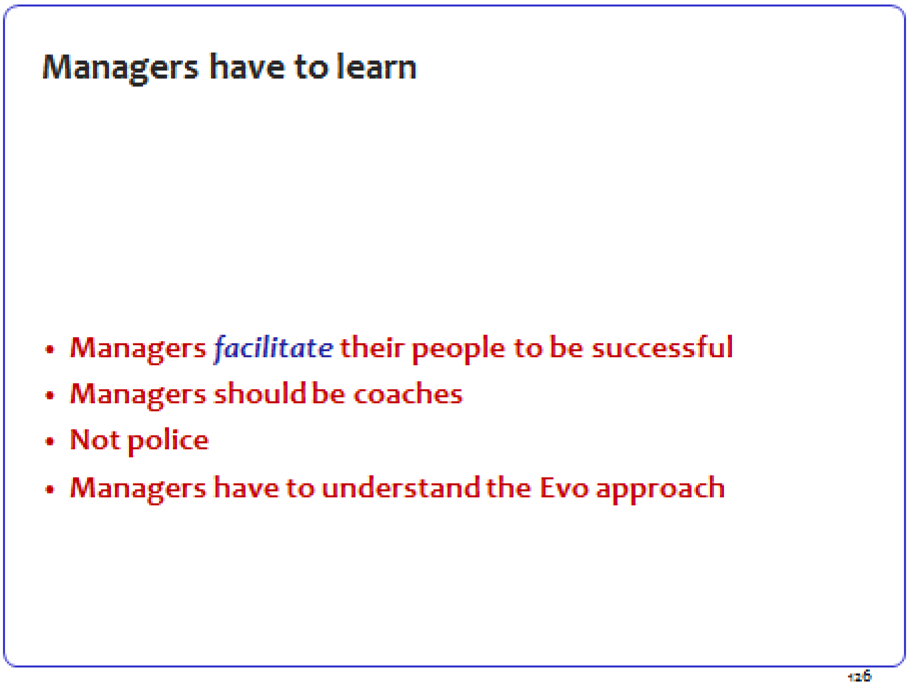
people resources

100%

30%

15%

125

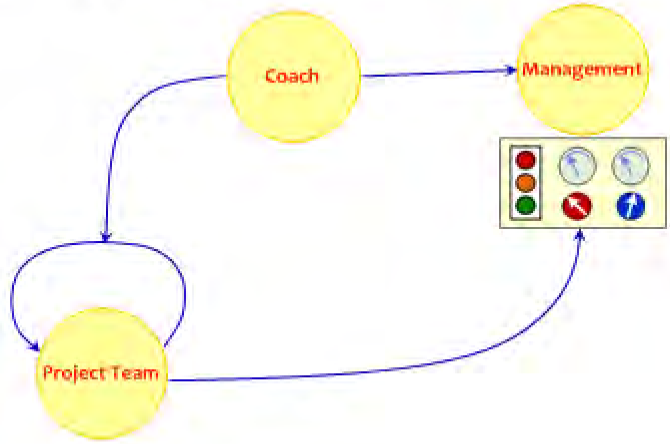


Managers have to learn

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

126

Local Loop Principle



127

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
- 0th 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

128

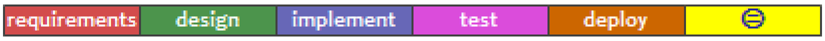
Requirements Example

- How many Requirements do you typically deal with ?

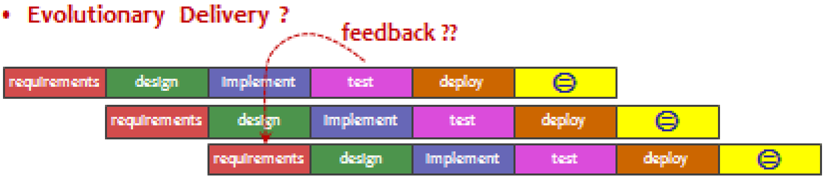
- CRM system, to replace 'old' CRM system
- Original plan: 6 months and € 1M
- Spent 1.5 years and € 5M: business hasn't seen any result
- New Project Manager, new System Integrator

- Who's project is this ?

Delivery Requirements

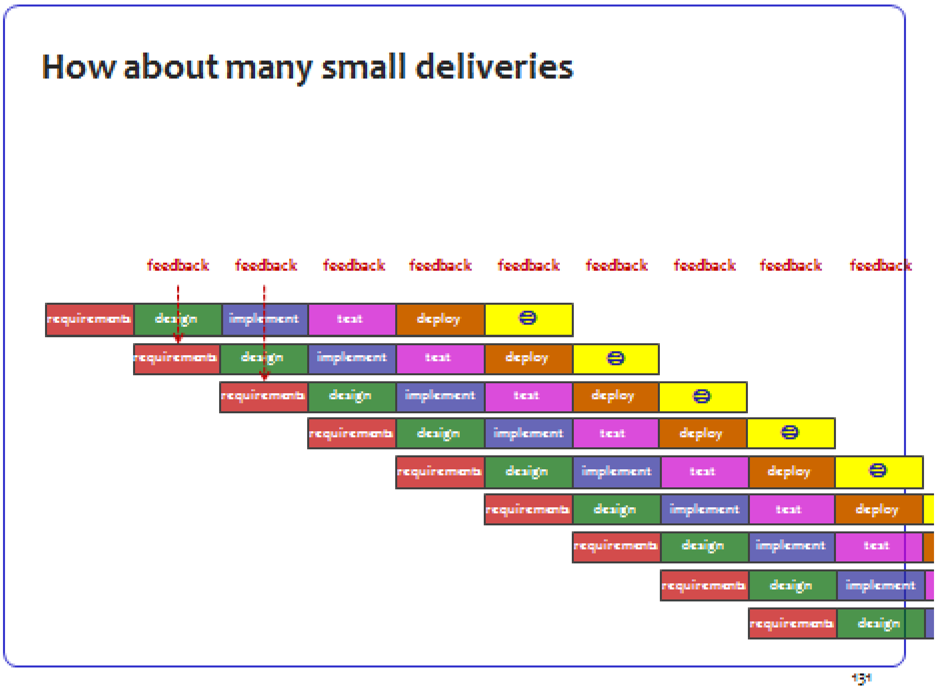

requirements design implement test deploy ☺

- Evolutionary Delivery ?


feedback ??

requirements design implement test deploy ☺
requirements design implement test deploy ☺
requirements design implement test deploy ☺

- Suggested Requirements:
  1. Within one week of any delivery, the business is not *less* efficient than before
  2. The business decides whether they are satisfied
- "Unacceptable" means supplier is saying:
  1. Within one week of a delivery, the business will be less efficient than before
  2. The business will not be satisfied



How about many small deliveries

feedback feedback feedback feedback feedback feedback feedback feedback feedback

requirements design implement test deploy ☺

requirements design implement test deploy ☺

requirements design implement test deploy ☺

requirements design implement test deploy ☺

requirements design implement test deploy ☺

requirements design implement test deploy

requirements design implement test

requirements design implement

requirements design

131