INCOSE nederland
International Council on Systems Engineering

Workshop
27 November 2010

Niels Malotaux

Predictable Projects
Delivering the Right Results 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 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: being predictable while delivering what the customer needs, when he needs it, to enable customer success. To this effect, Niels developed an approach for effectively teaching 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, India, Ireland, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less in the practice of real projects. He is a frequent speaker at conferences and published several booklets around the topic of the presentation (see www.malotaux.nl/Booklets).

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.

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

1

Niels Malotaux

Result Management

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

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

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

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?

5

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

6

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

7

Causes of Delay

- Some typical causes of delay are:
  - Developing the wrong things
  - Unclear requirements
  - The fallacy of 'all' requirements
  - Misunderstandings
  - No feedback from stakeholders
  - No adequate planning
  - No adequate communication
  - Doing unnecessary things
  - Waiting (before and during the project)
  - Changing requirements
  - Doing things less cleverly
  - Doing things over
  - Indecisiveness
  - Suppliers
  - Quality of suppliers results
  - 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

8

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

- So called Scientific Management Techniques
- No dissemination through the whole organization

9

But we are Systems Engineers!

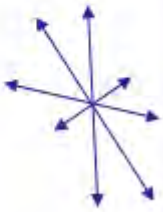
- The Project Manager is *responsible* for *delivering* the right result at the right time
- The Project Worker's work and decisions *determine* the *result* and the *time* it is delivered
- This makes everybody in the project implicitly *as responsible as* Project Management

10

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*



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

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

13

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

14

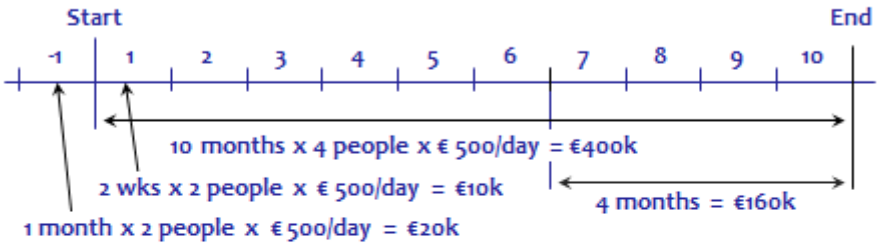
Project ROI

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

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*!

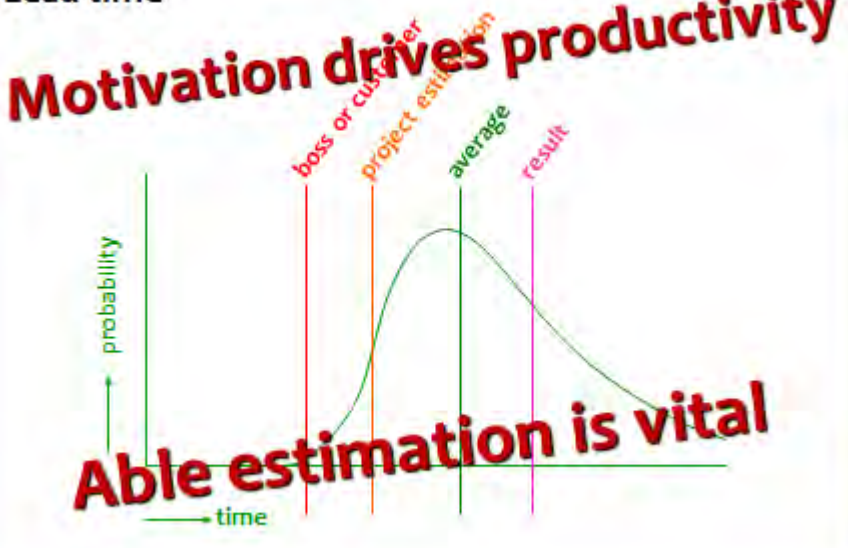
16

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 ?

17

Lead time



18

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?

19

Is this what you did?

20

Defect rate

- Before test ?
- After test ?

21

Alternative Design (*how* to solve the requirement)

22

Another alternative design

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

23

What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

24

Elements in the exercise

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

35

Human Behavior

26

Human Behaviour

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

27

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

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

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)

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

Is the design wrong, or the users of the result?

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

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!

34

Logical thinking is not always better

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

Ref Ap Dijksterhuis: *Het Slimme Ombewuste*

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

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

37

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 ?

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

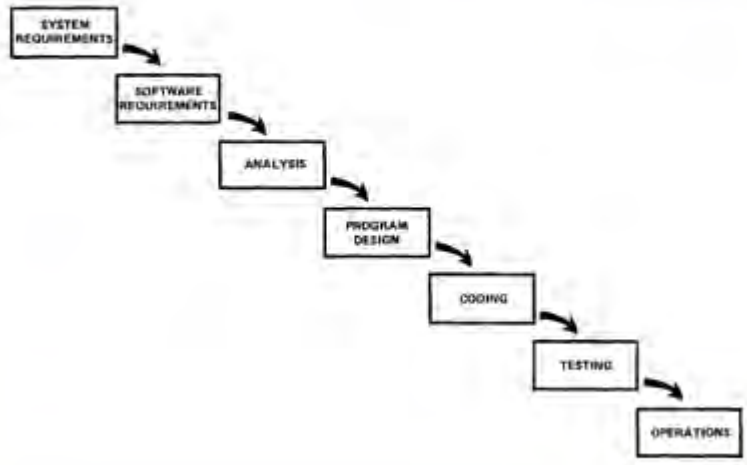
39

Project Life Cycles

40

Waterfall ?

Winston Royce 1970



41

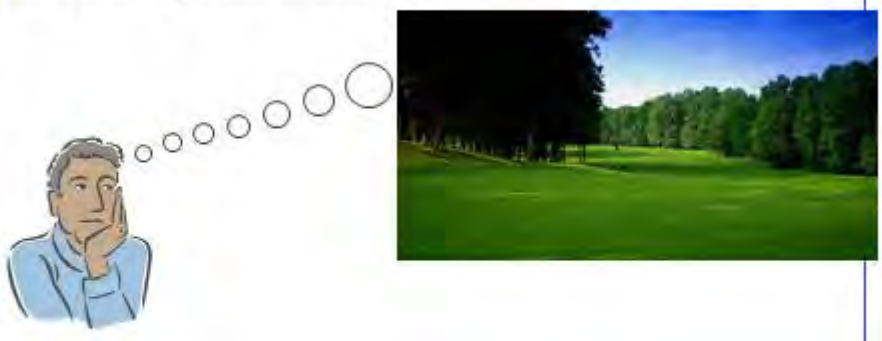
When can we use waterfall ?

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

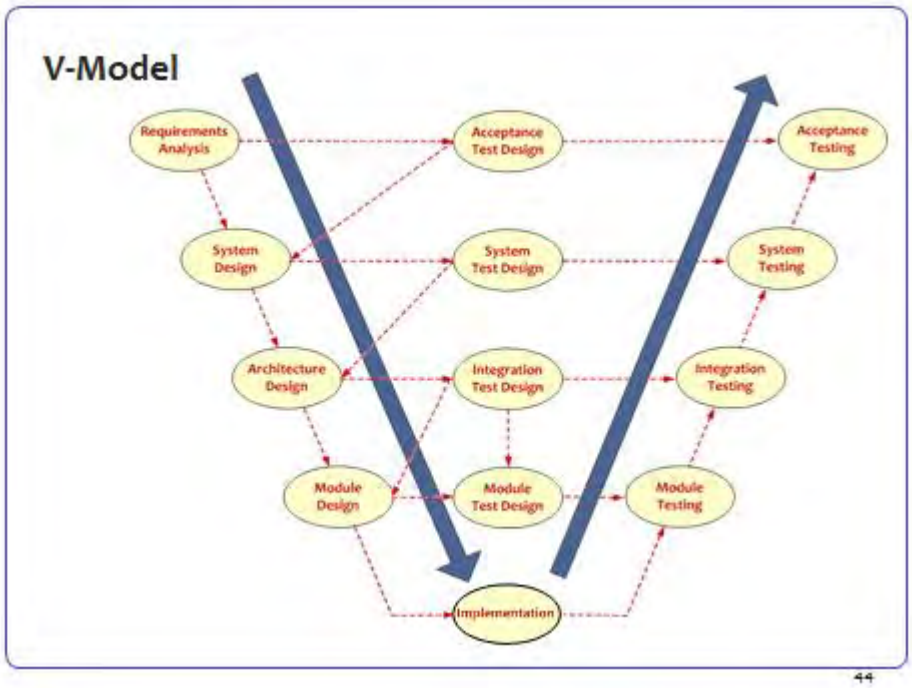
- In your projects:
  - Is everything completely clear ?
  - Will nothing change ?
  - Does everybody know exactly what to do ?
  - Are you sure ?

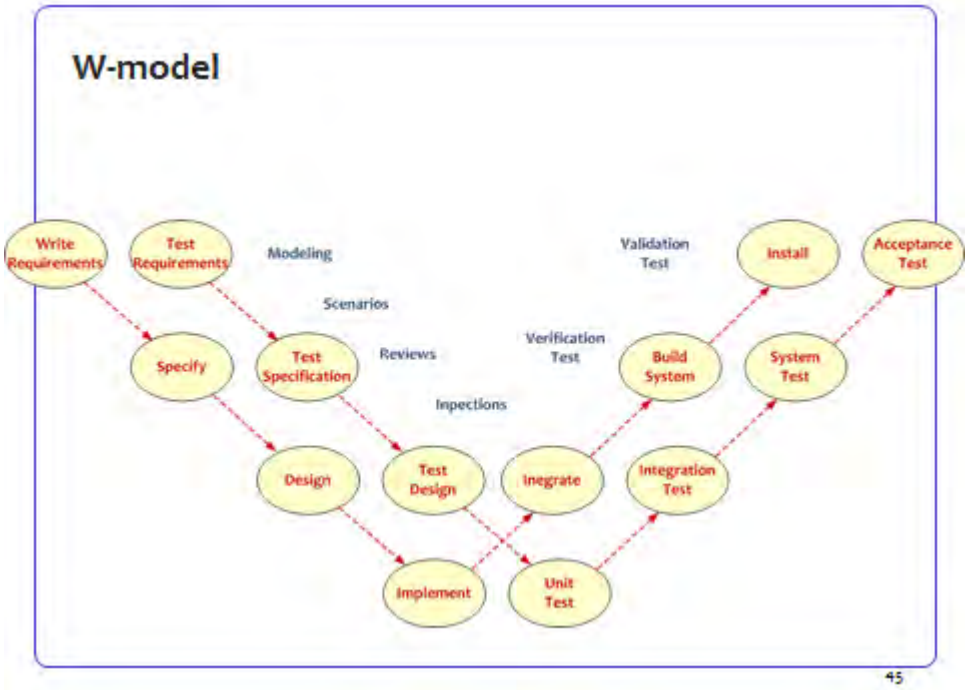
42

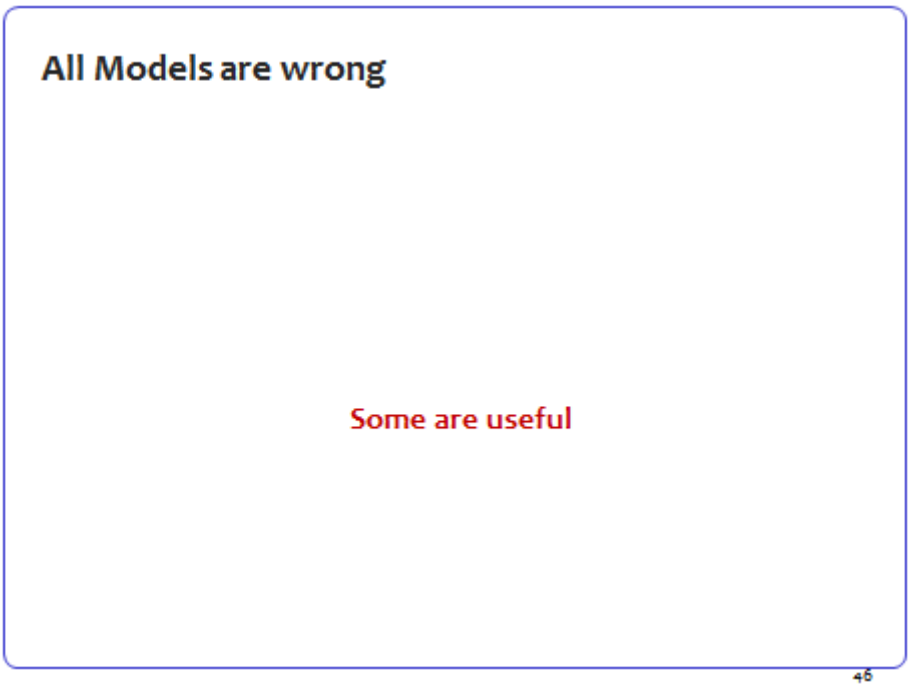
How management likes it



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
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
Inegrate
Integration Test
Implement
Unit Test
45


All Models are wrong
Some are useful
46

Evolutionary Principles

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*

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

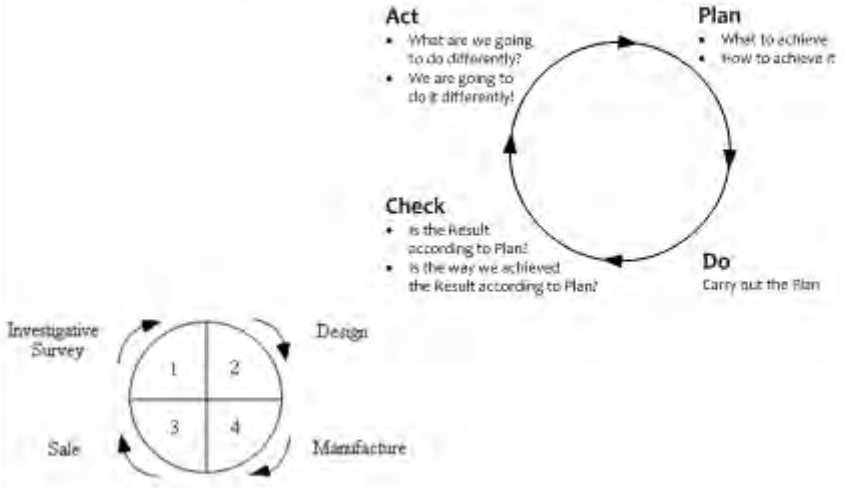
Only with *prevention* we can save precious time

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

49

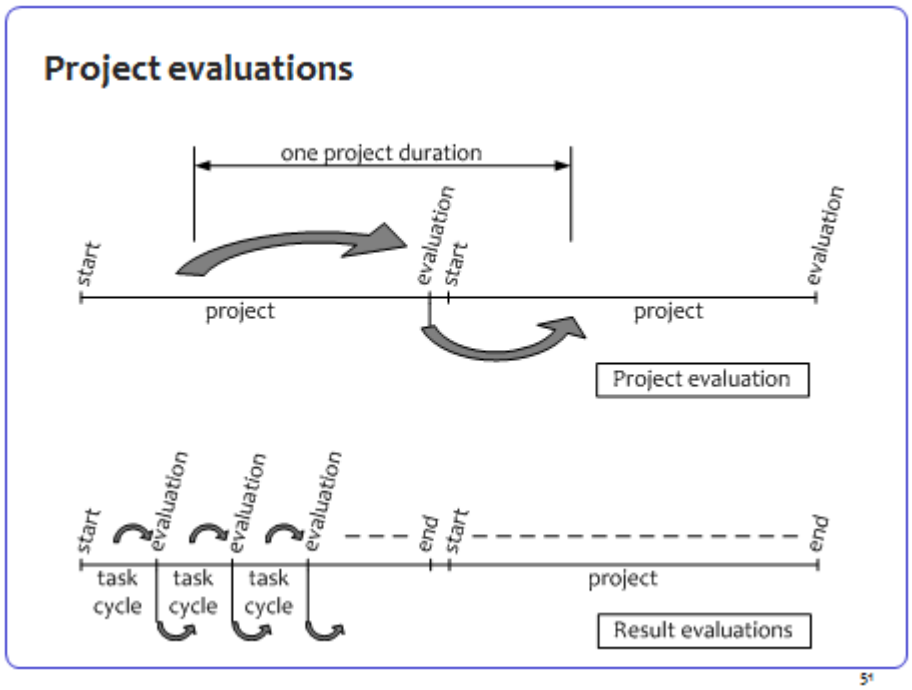
The essential ingredient: the PDCA Cycle

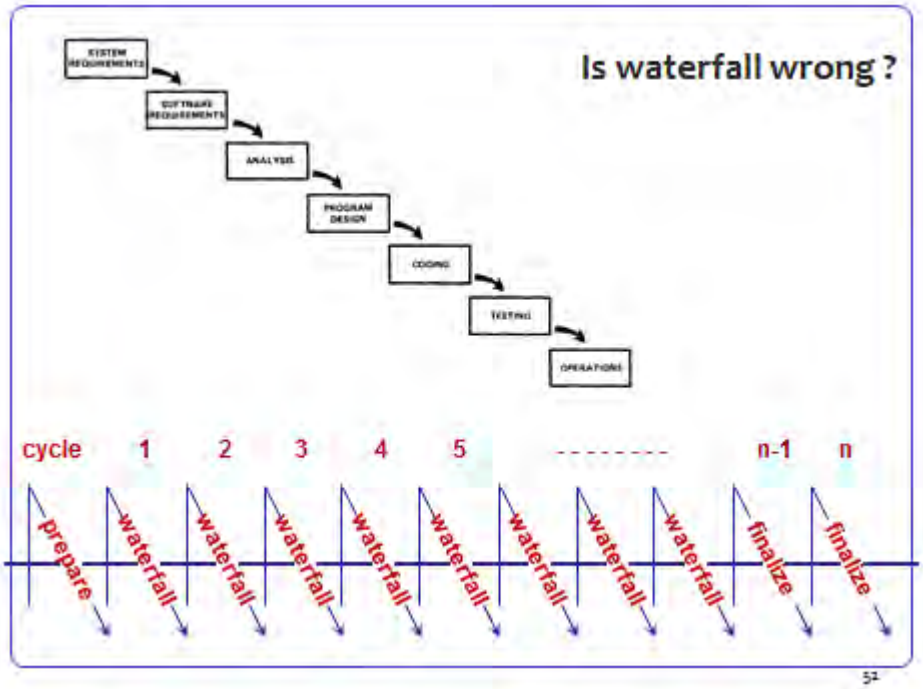
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)

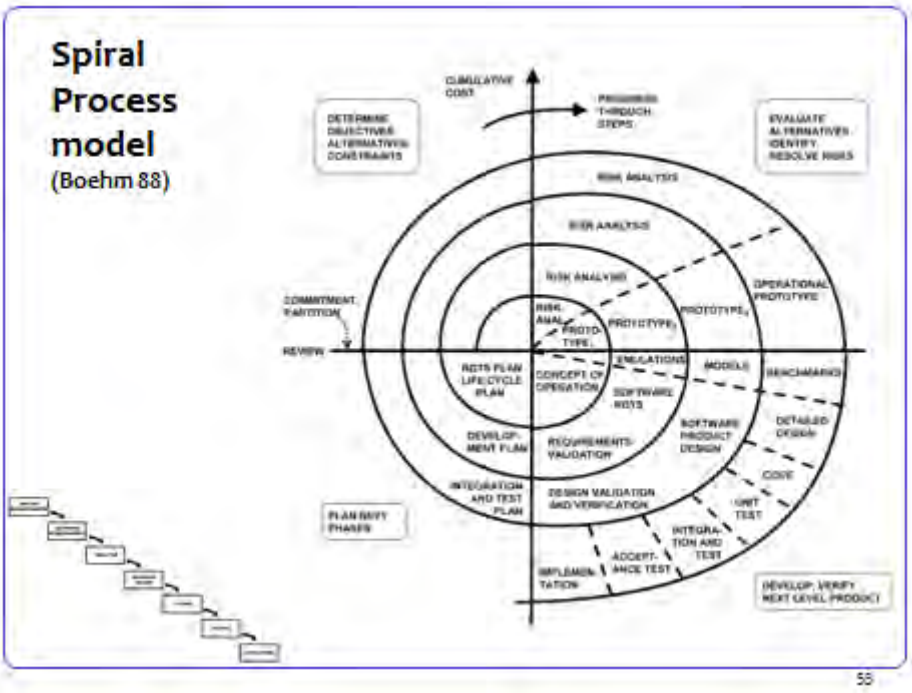


What Deming showed in Japan in 1950

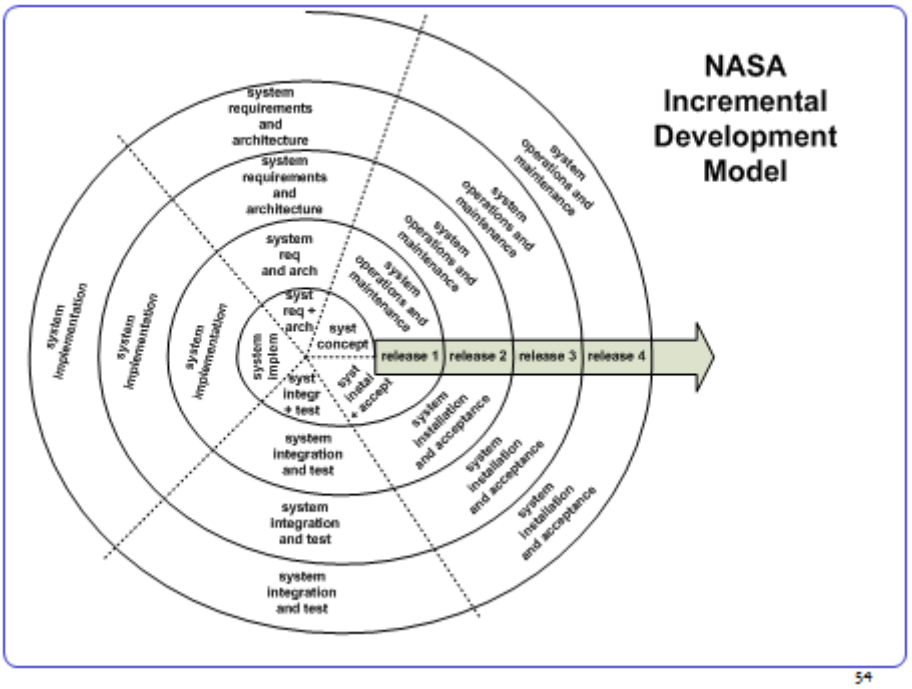
50


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
51


Is waterfall wrong ?
cycle 1 2 3 4 5 n-1 n
prepare
waterfall
finalize
52

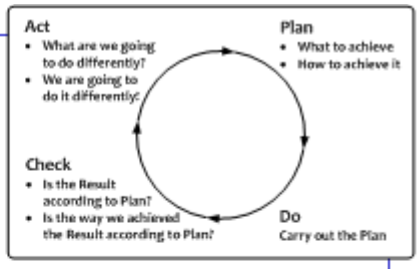

Spiral
Process
model
(Boehm 88)
CUMULATIVE COST
DETERMINE OBJECTIVES ALTERNATIVES CONSTRAINTS
EVALUATE ALTERNATIVES IDENTIFY RESOLVE RISKS
RISK ANALYSIS
RISK ANALYSIS
RISK ANALYSIS
COMMITMENT PARTITION
REVIEW
RISK ANALYSIS
PROTOTYPE
PROTOTYPE
OPERATIONAL PROTOTYPE
RQTS PLAN LIFECYCLE PLAN
CONCEPT OF OPERATION
EMULATIONS
MODELS
BENCHMARKS
SOFTWARE RQTS
DEVELOPMENT PLAN
REQUIREMENTS VALIDATION
SOFTWARE PRODUCT DESIGN
DETAILED DESIGN
CODE
INTEGRATION AND TEST PLAN
DESIGN VALIDATION AND VERIFICATION
UNIT TEST
INTEGRATION AND TEST
ACCEPTANCE TEST
IMPLEMENTATION
PLAN NEXT PHASES
DEVELOP, VERIFY NEXT-LEVEL PRODUCT

55


NASA
Incremental
Development
Model
system requirements and architecture
system requirements and architecture
system req and arch
syst req + arch
syst concept
system implem
syst integr + test
syst instal + accept
release 1
release 2
release 3
release 4
system operations and maintenance
system operations and maintenance
system operations and maintenance
system operations and maintenance
system implementation
system implementation
system implementation
system integration and test
system integration and test
system integration and test
system installation and acceptance
system installation and acceptance
system installation and acceptance

54

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

55

Building on quite some history

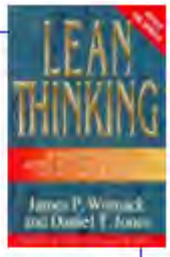
- Benjamin Franklin (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- Henry Ford (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - 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 Principe
- 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

56

Lean

A lot of the cost of vehicles is based on:
- bad design
- poor management
- an attitude that problems, no matter how small, can be overlooked



- 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

57

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

58

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)

59

Pillars of the TPS

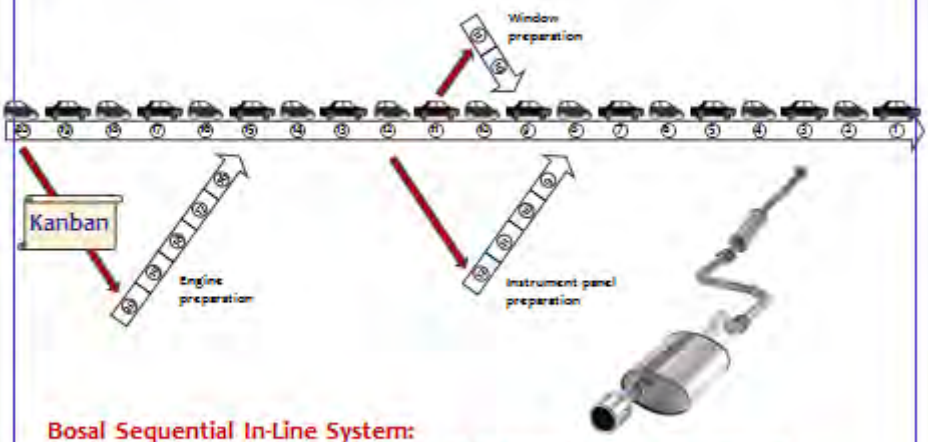
- Just in Time
  - No inventory
  - Doing the right things at the right time
- Perfection
  - Perfection is a condition for JIT to work
  - If a defect is found, stop the line, find cause, fix immediately
  - Continuous improvement of product, project and process
- Autonomation
  - The loom runs unattended until signalling it needs help

  For development:
  - The development team runs unattended until signalling they need help
    (caused by an issue beyond their control)
  - Management observes the team and facilitates them to become ever more efficient, to *prevent* issues delaying them beyond the teams control – *Education*, *Empowerment* and *Responsibility* of people
  - If an issue does occur, management helps to remove obstacles quickly, making sure it doesn't happen again

60

Just In Time delivery – *no inventory*

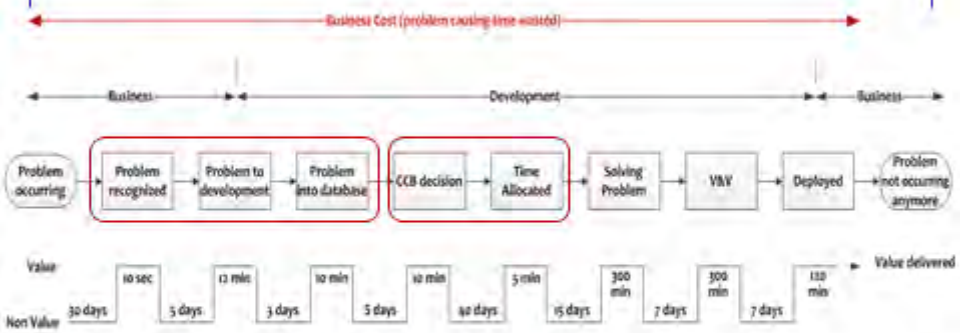
(after Ohno)



Bosal Sequential In-Line System:
We pioneered just-in-time delivery of exhaust systems - supplying systems to the assembly line within 80 minutes of receiving the order

61

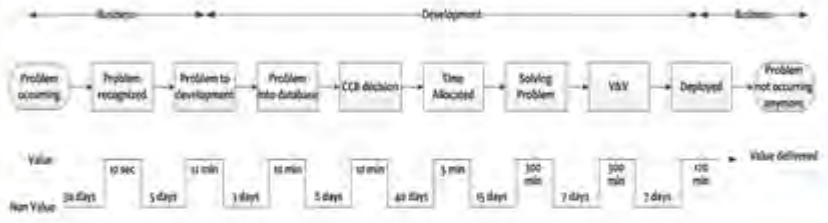
Value stream example



- 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 100 people x 10 min x 112 days x 400€/day = 187 k€
- Net Cost of Value: 1.6 days: ~3 people x 1.6 days x 1000€/day = 5 k€

62

Capacity = Work + Waste



Work Capacity

- Net Work, creating value
- Non-value adding, but necessary work
- Waste

Because it costs nothing, eliminating waste is one of the easiest ways for an organization to improve it's operations

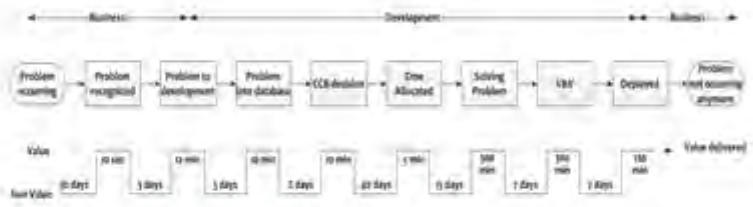
63

Identifying waste

Manufacturing	Development	*Possible* Remedies
Overproduction	Extra features Unused documents	Prioritizing, Real Requirements, Deciding what not to do
Inventory	Partially done work	Synchronization, Just In Time
Transport	Handoffs	Keeping in one hand/mind: - Responsibility (what to do) - Knowledge (how to do it) - Action (doing it) - Feedback (learning from Result)
Processing	Design inefficiency Wishful thinking	Knowledge, experience, reviews Preflection
Waiting	Delays	Process/Organization redesign
Movement	Task Switching	Max 2 tasks in parallel
Defects	Defects	Prevention
Ignoring ingenuity of people	Ignoring ingenuity of people	Real management, Empowerment Bottom-up responsibility

64

5-S



- Seiri - Remove unnecessary things → waste
- Seiton - Arrange remaining things orderly → flow
- Seiso - Keep things clean → uncovers hidden problems
- Seiketsu - Keep doing it, standardize → know what to improve
- Shitsuke - Keep training it → fighting entropy

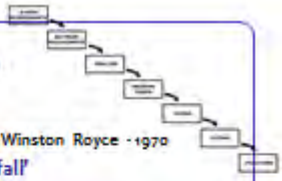
65

The 3 Mu's to remove

- Muda - Waste → minimize waste
- Mura - Irregularities → optimize flow
- Muri - Stress → sustainable pace

66

There is nothing new in software too



- 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
- 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

67

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

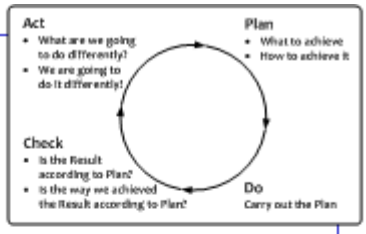
68

Lean things

- Most managers think their greatest contribution to the business is doing work-arounds on broken processes, rather than doing the hard work to get the process right so that it never breaks down (Womack)
- 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)
- Plan-Do-Check-Act cycle was by far the most important thing we did in hindsight (Tom Harada)

69

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

70

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: quantificati
- Architecture and De
  - Selecting the optim 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

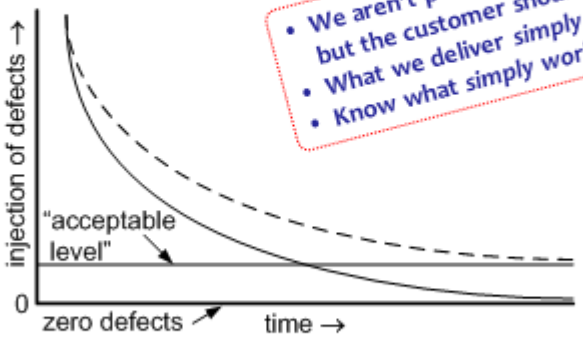
- Weekly TaskCycle
  - Short term planning
  - Optimizing estimation
  - Promising what we can achieve
  - Living up to our promises
- Bi-weekly DeliveryCycle
  - Optimizing the requirements and che ing the assumptions
  - Soliciting feedback by delivering Real Results to *eagerly waiting* Stakeholders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

Right time

71

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
- Know what simply works means !

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

72

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

73

Evolutionary Planning

TaskCycle
DeliveryCycle

74

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

75

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

76

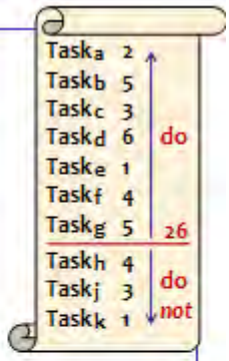
Effort and Lead Time

- 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)

77

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!*

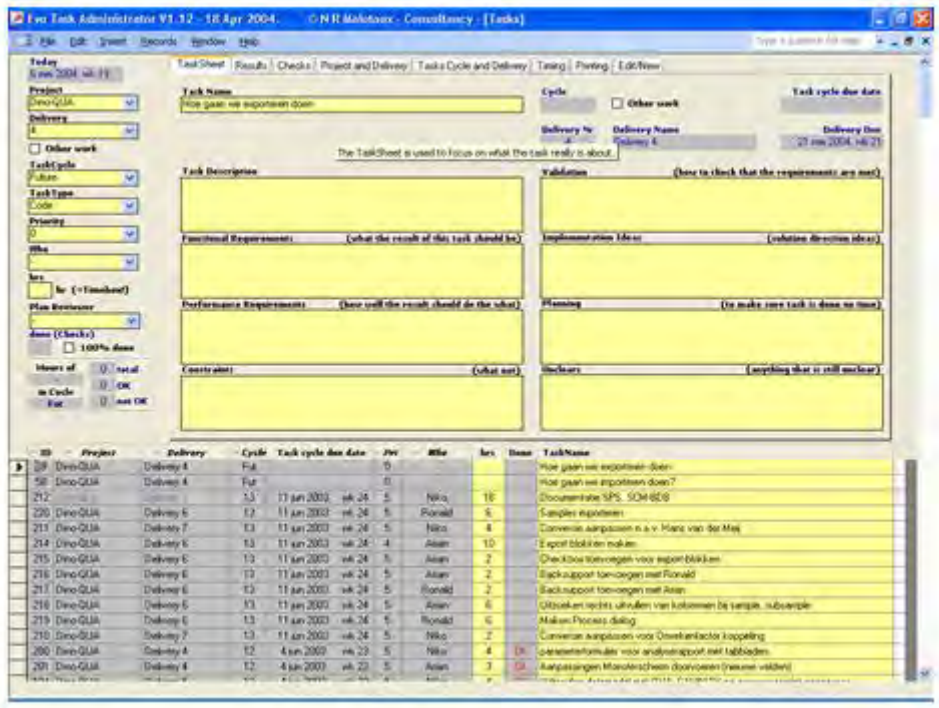
2/3 is default start value
this value works well in development projects

78

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

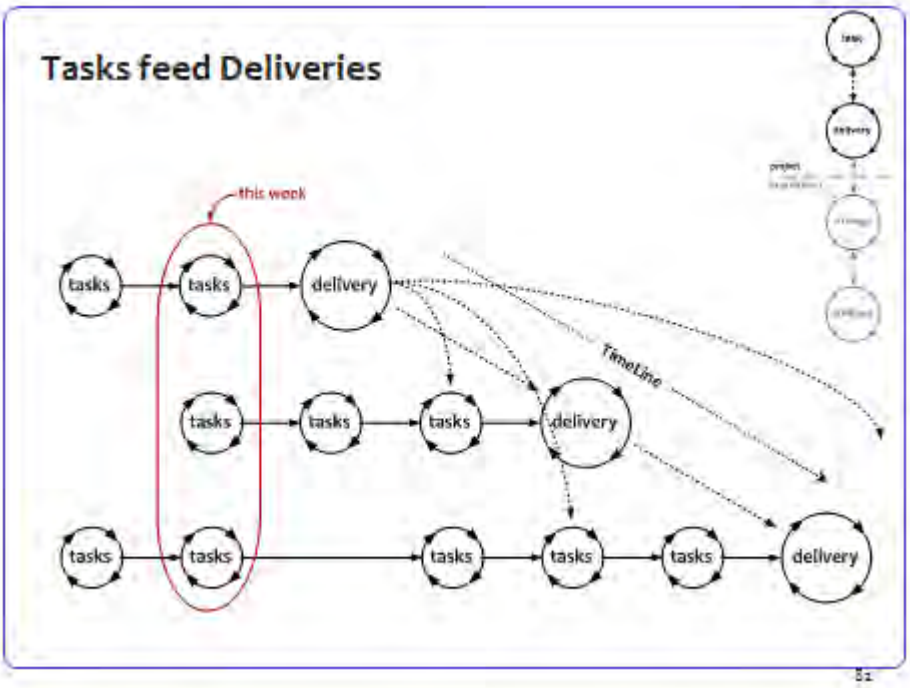
79

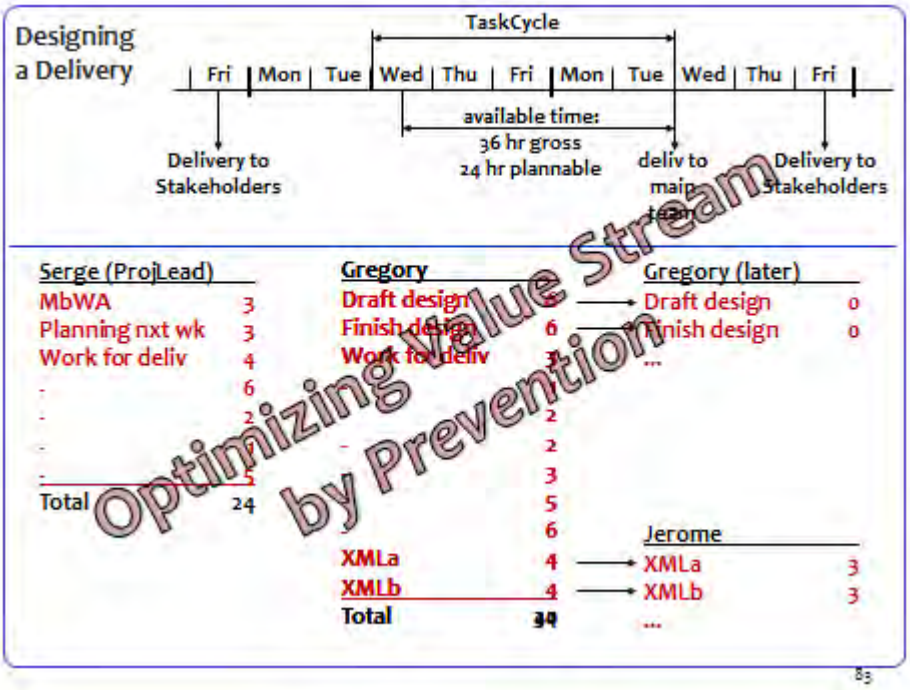


DeliveryCycle

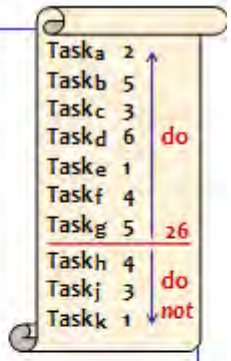
- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
    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

81


Tasks feed Deliveries
this week
tasks
tasks
delivery
tasks
tasks
tasks
delivery
tasks
tasks
tasks
tasks
tasks
delivery
TimeLine
81


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
- 2
- 5
Total 24
Gregory
Draft design 5
Finish design 6
Work for deliv 3
1
2
2
3
5
6
XMLa 4
XMLb 4
Total 34
Gregory (later)
Draft design 0
Finish design 0
...
Jerome
XMLa 3
XMLb 3
...
Optimizing Value Stream by Prevention
83

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 ?

84

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 ?

85

Evolutionary Planning

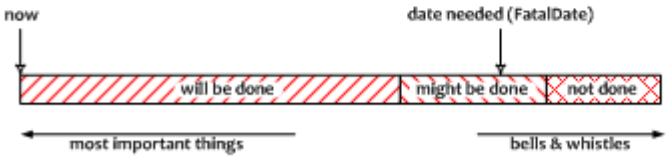
TimeLine

86

TimeLine

What the customer wants, he cannot afford

Standard Projects



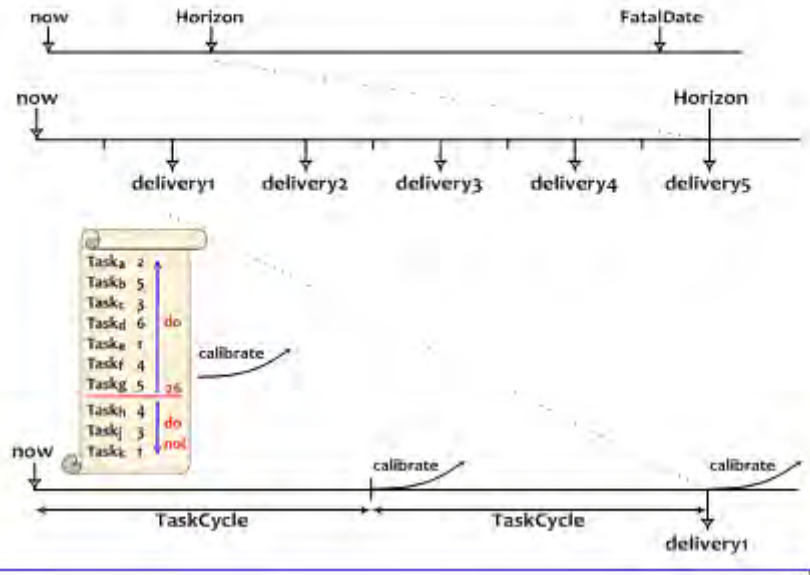
Evo

87

If it easily fits ...

88

Result to Tasks and back



89

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

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

90

Predicting *what* will be done *when*

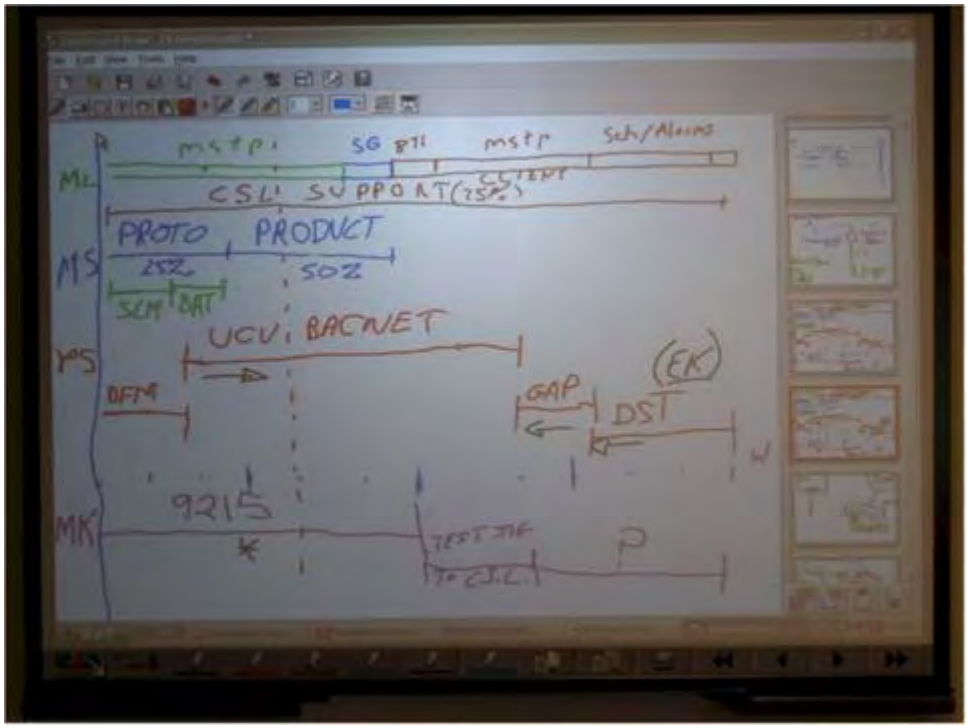
Line	Activity	Estim	Spent	Still to spend	Ratio real/es	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009
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

91

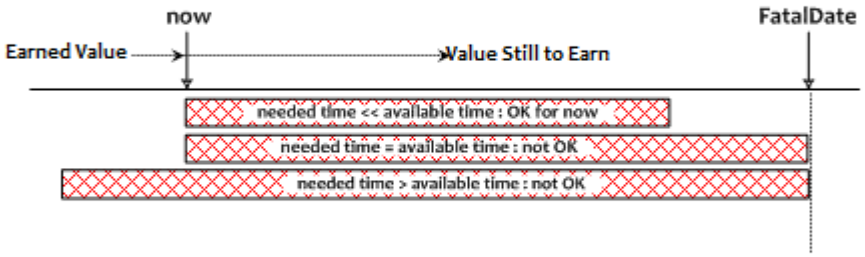
Product/Portfolio/Resource Management

- 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



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



- If it doesn't fit ... count backwards
- When the match is over, you can't score a goal any more

94

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

95

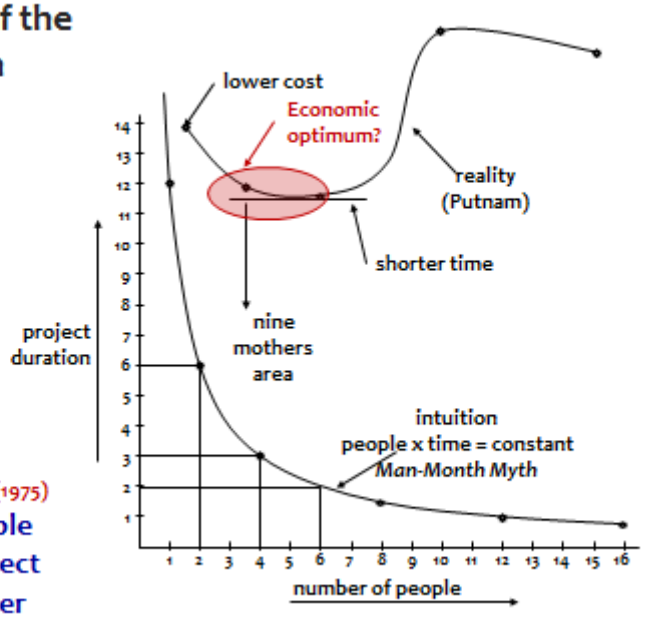
Adding people to a late project ...

makes it later

(Brooks' Law, 1975)

96

The Myth of the Man-Month



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

97

Saving time

Continuous elimination of waste

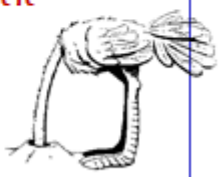
We don't have enough time, but we can save time
without negatively affecting the Result !

- 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

98

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 !
Lean: if we eliminate the waste!

99

Estimation techniques used

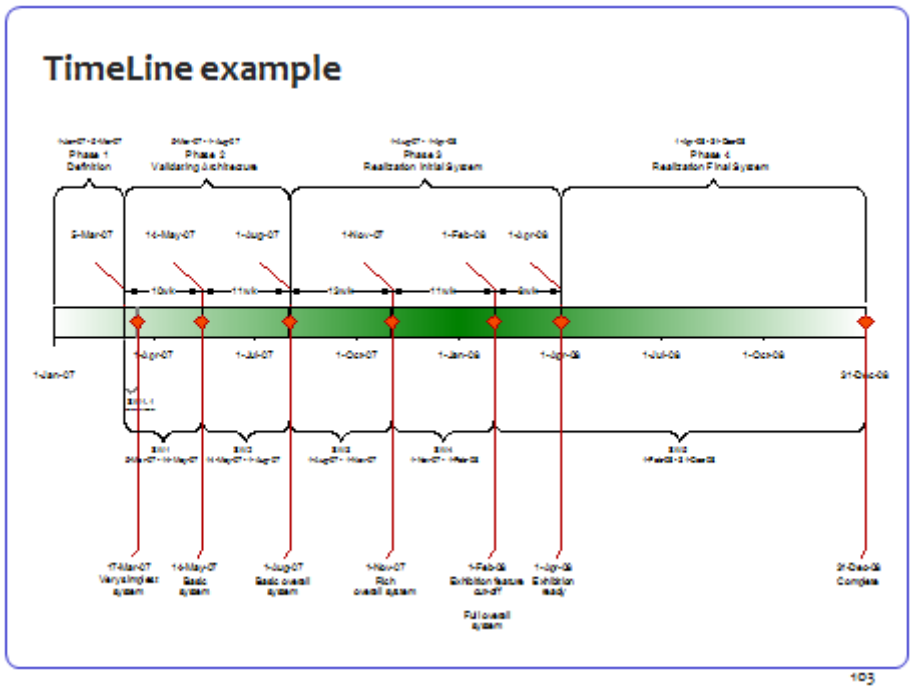
- **Just-enough estimation** (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- **Changing from optimistic to realistic predictions**
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- **0^{th} order estimations** (ball-park 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

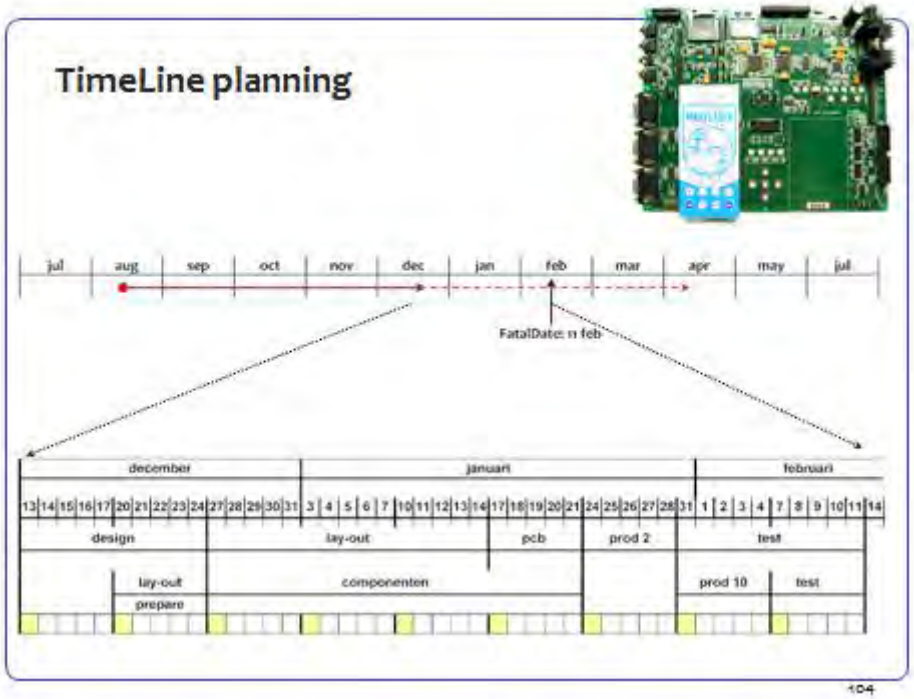
100

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
Very simple system
14-May-07
Basic system
1-Aug-07
Basic overall system
1-Nov-07
Rich overall system
1-Feb-08
Exhibition feature complete
1-Apr-08
Exhibition ready
Full overall system
31-Dec-08
Complete
103


TimeLine planning
jul
aug
sep
oct
nov
dec
jan
feb
mar
apr
may
jul
FatalDate: n feb
december
januari
februari
design
lay-out
pcb
prod 2
test
lay-out
prepare
componenten
prod 10
test
104

TimeLine exercise example

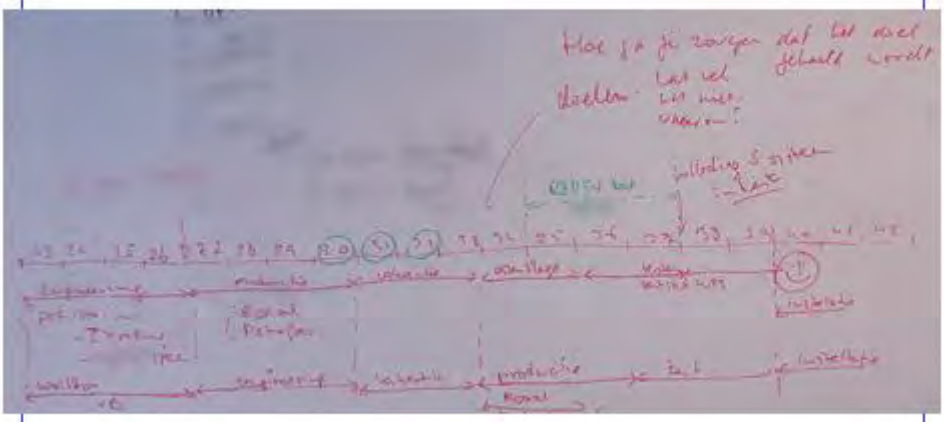
- Preparing for student exams

105

What we did

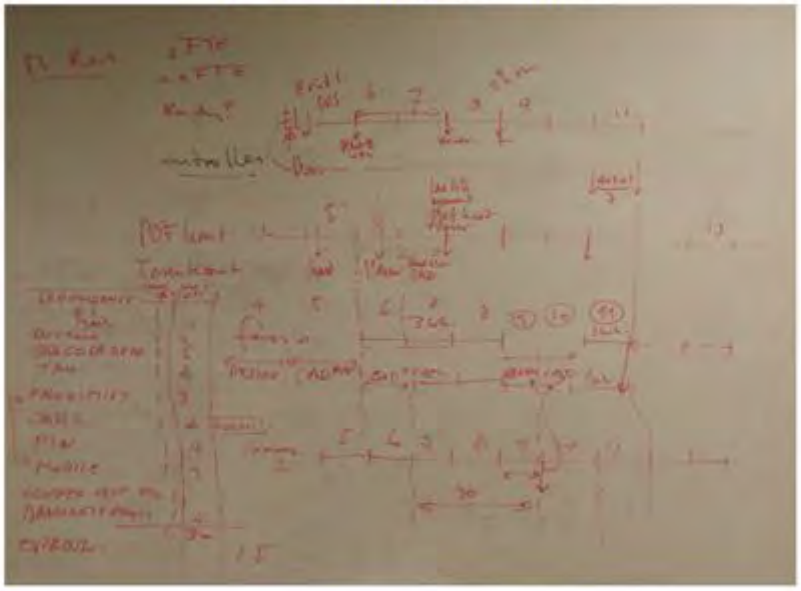
106

Whiteboard TimeLine Planning



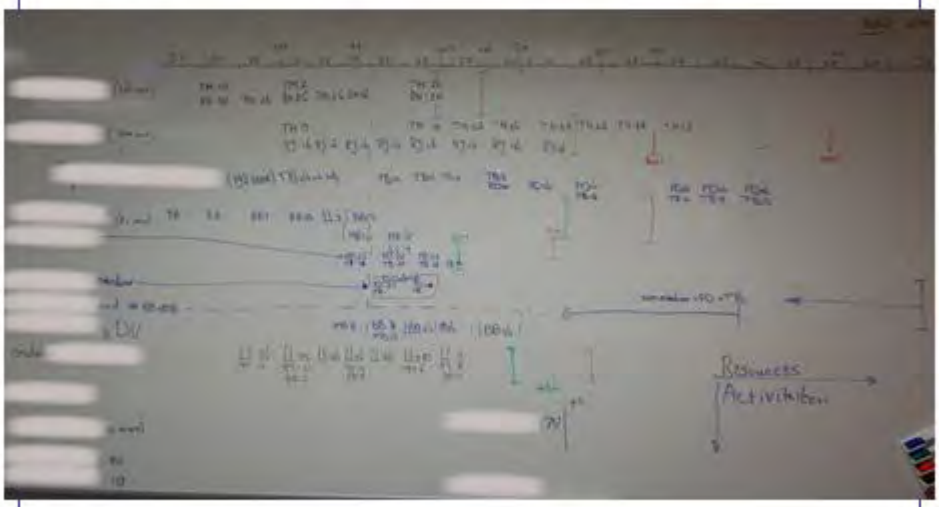
107

Synchronizing TimeLines



108

Whiteboard TimeLine Resource Planning

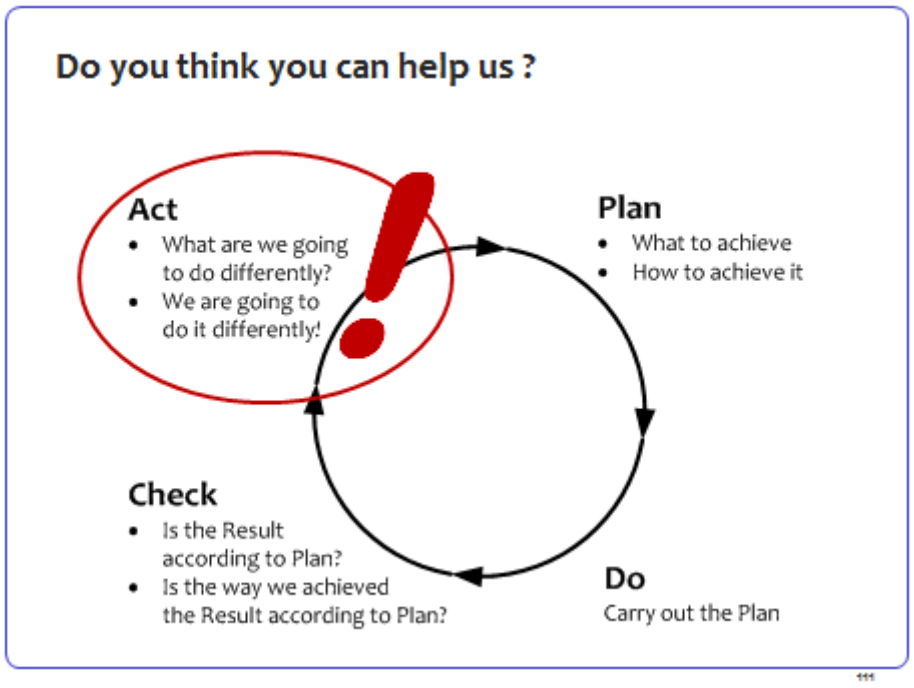


109

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?

110


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

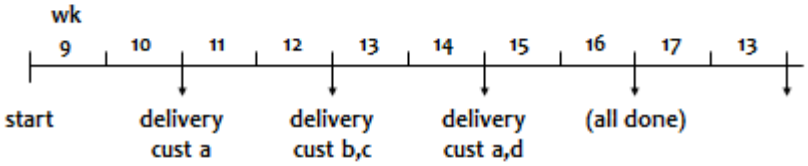

In stead of complaining about a problem ...
(Stuck in the Check-phase)
Let's do something about it !
(Moving to the Act-phase)

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		

113

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

114

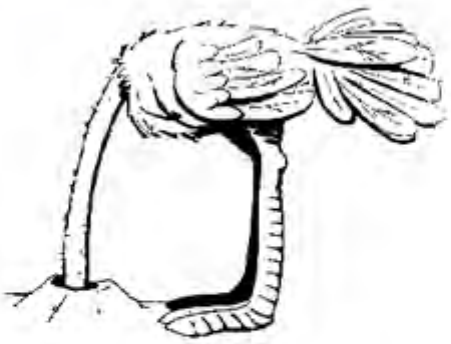
Result

- 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

115



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

116

Some extra

117

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

118

Interrupts

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

- In case of interrupt, use interrupt procedure

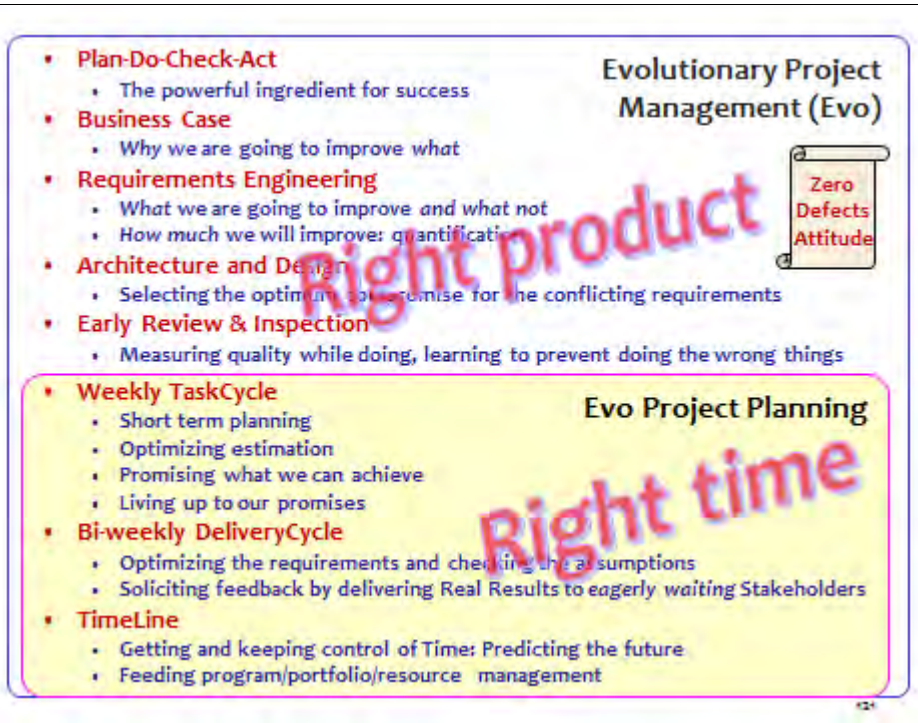
119

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.

120


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: quantificati
Architecture and De
Selecting the optim
for the conflicting requirements
Early Review & Inspection
Measuring quality while doing, learning to prevent doing the wrong things
Evolutionary Project Management (Evo)
Zero Defects Attitude
Right product
Weekly TaskCycle
Short term planning
Optimizing estimation
Promising what we can achieve
Living up to our promises
Bi-weekly DeliveryCycle
Optimizing the requirements and chec
sumptions
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
Evo Project Planning
Right time

Business Case

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

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*

124

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

125

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

- Actually, every Stakeholder has his Business Case

but

- Only the Customer pays and can make tradeoffs
- The other Stakeholders couldn't care less about the cost

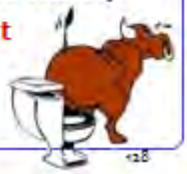
126

Stakeholders & Requirements

127

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!



128

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*

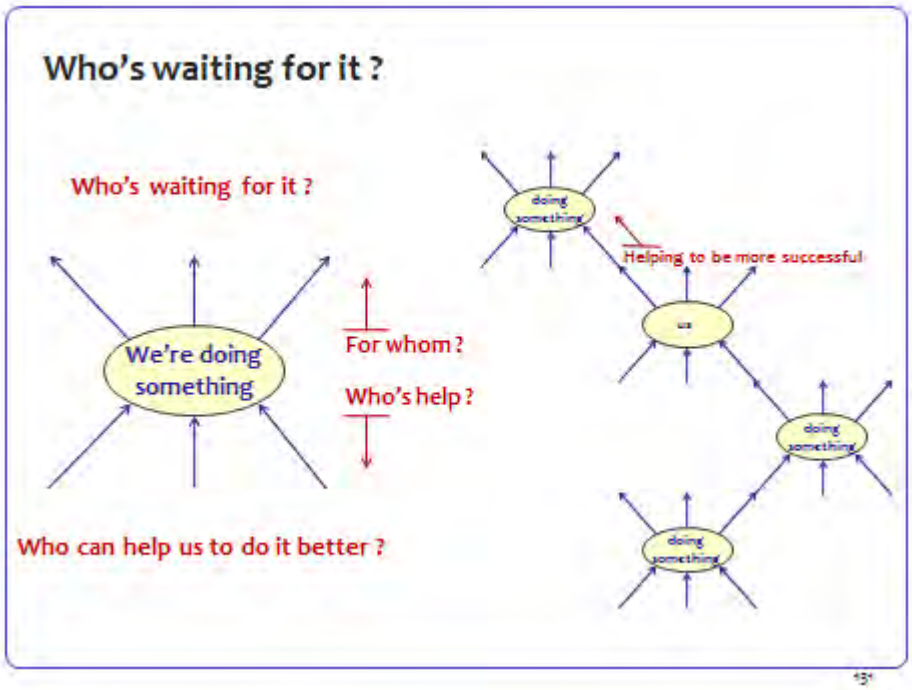
- The set of Stakeholders doesn't change much
- Do you have a checklist of possible Stakeholders ?

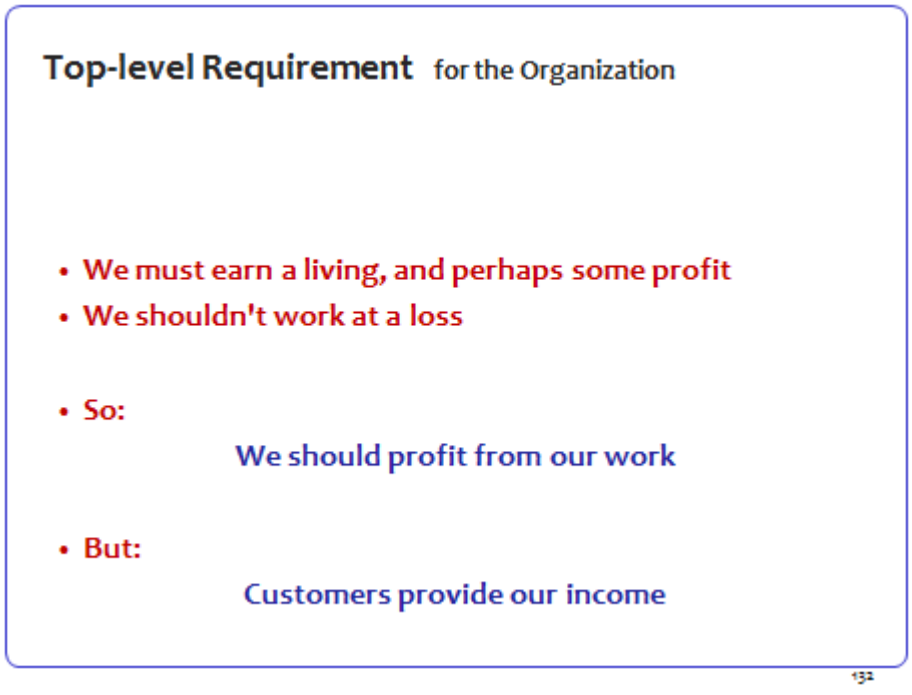
129

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?

130


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
131


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
132

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

133

Example

Customer Relations Management project

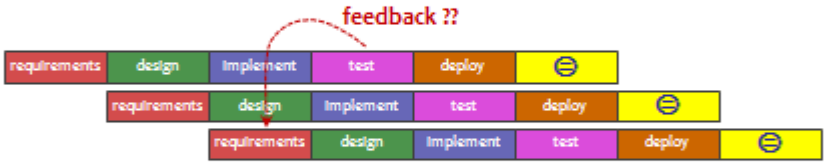
- CRM system, original plan: 6 months and € 1M
- Spent 1.5 years and € 5M
- Business hasn't seen any result whatsoever
- Systems Integrator still "working hard"

- New Project Manager, new System Integrator
- Started working in exactly the same fashion…

134

Few larger deliveries

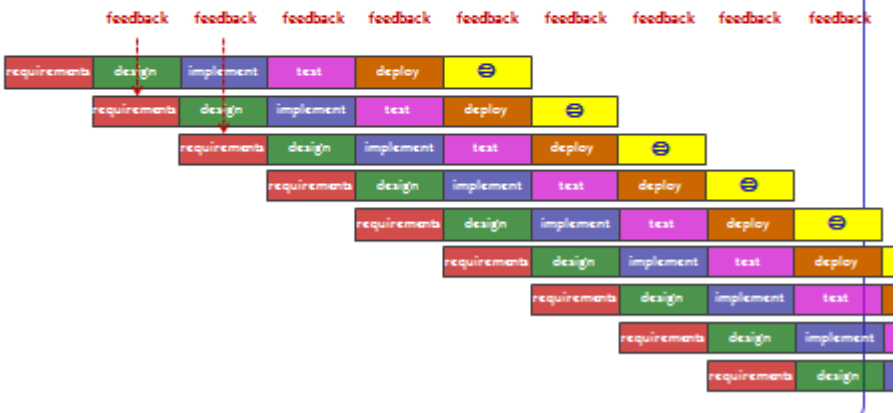
- Evolutionary Delivery ?



- 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

135

Many small deliveries



136

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)

137

Requirements with Planguage

ref Tom Gilb

Definition:

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

Specific
Measurable

Benchmarks (Playing Field):

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...>

Attainable

Requirements: Time

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

Realizable

138

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

139

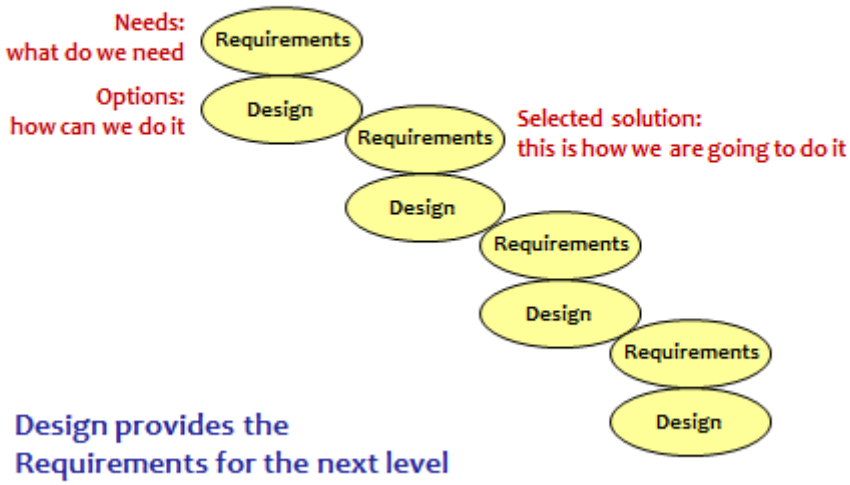
5 times "Why?" technique

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

140

No Design in the requirements, but ...



Design provides the
Requirements for the next level

141

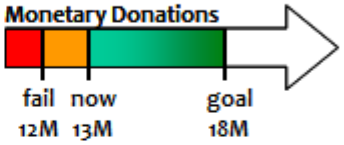
Requirements Case

- 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

142

Objective: Monetary Donations

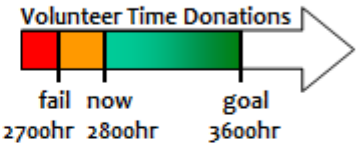


Name	Monetary Donations
Scale	Euro's donated to non-profits through our website
Meter	Monthly Donations Report
Fail	12M
Now	13M [2008] ←Annual Report 2008
Goal	18M [2009]

Ref Ryan Shriver
ACCU Overload Feb 2008

143

Objective: Volunteer Time (Natura) Donations

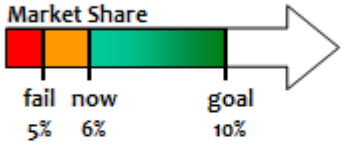


Name	Volunteer Time Donations
Scale	Hours donated to non-profits through our website
Meter	Monthly Donations Report
Fail	2700 hr
Now	2800 hr [2008] ←Annual Report 2008
Goal	3600 hr [2009]

Ref Ryan Shriver
ACCU Overload Feb 2008

144

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

145

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 2008

146

Architecture and Design

147

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) ↓

148

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 ?

149

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

150

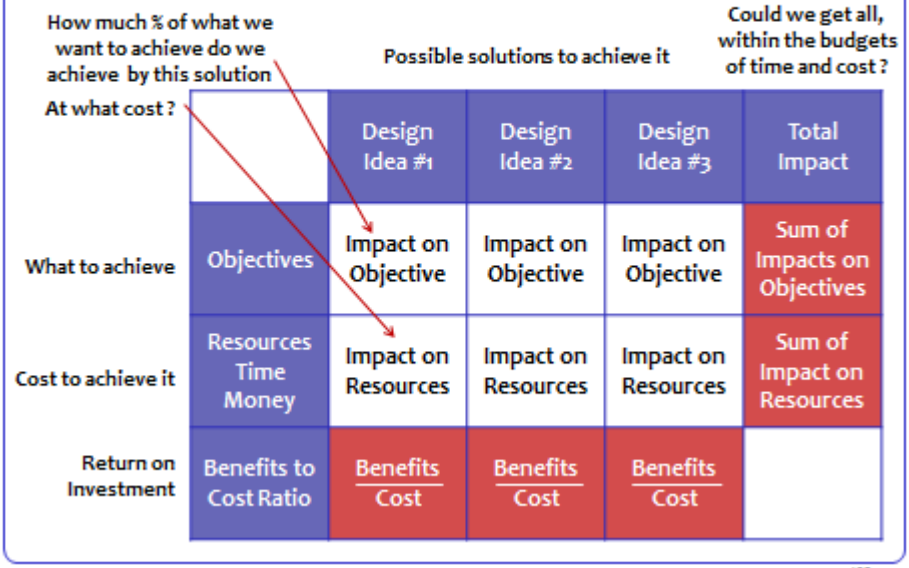
Impact Estimation example

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

151

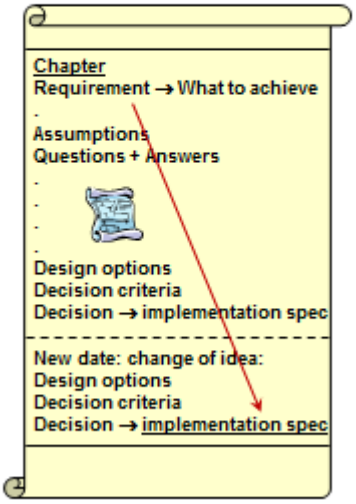
Impact Estimation principle



152

DesignLog (project level)

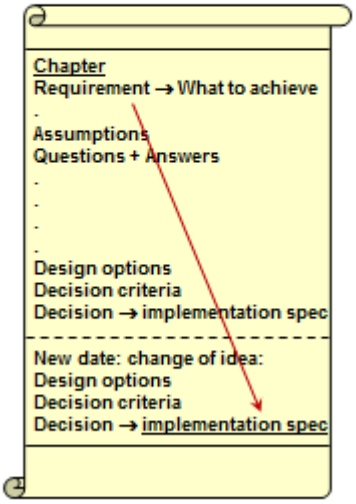
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + reasoning:
    - If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



153

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



154

ISO 15288 ?

- Keep Outcomes in chapters in the ProcessLog
  - Which Outcome is addressed ?
    - Otherwise the chapter is empty
  - What are the current activities for this Outcome ?
  - What should we do to improve?
  - Why?
  - Etc

155

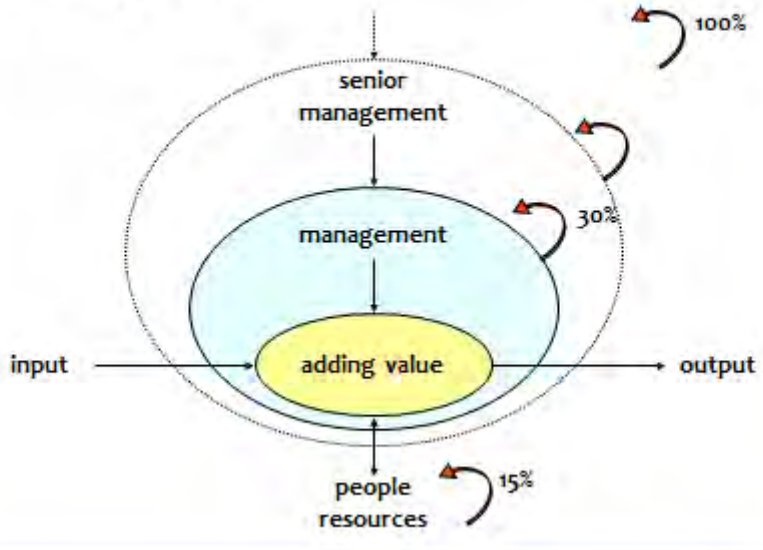
Five processes

156

Managers Tasks

157

The managers task



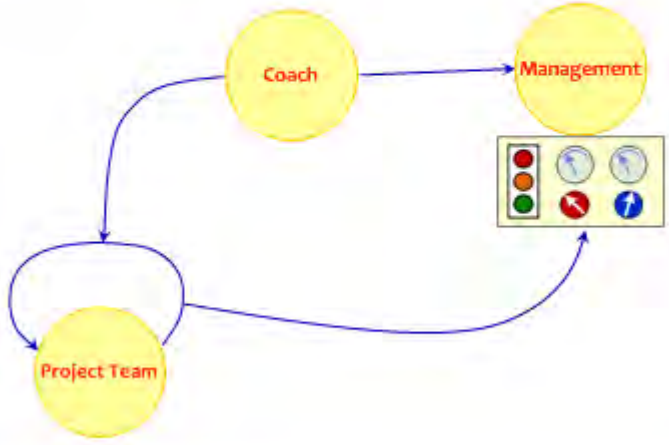
158

Managers have to learn

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

159

Local Loop Principle



160

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

161

What now ?

161

Early Reviews & Inspections

163

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

164

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

165

16 page Inspection Manual

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

166

Use the three rules on these Requirements

It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality

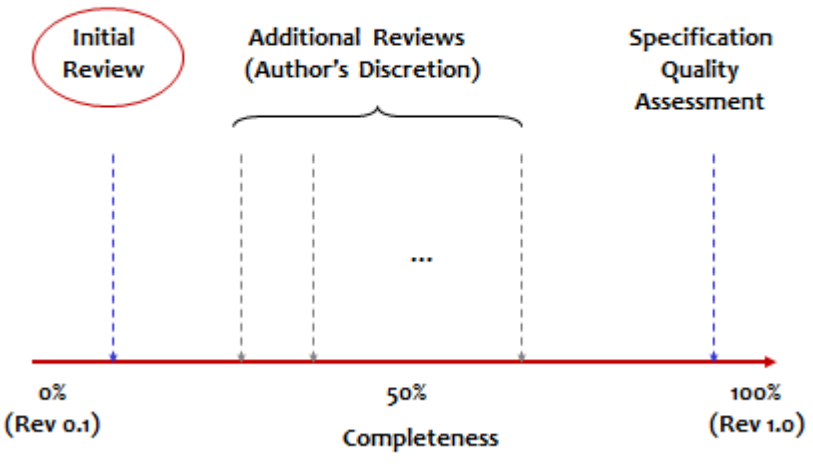
It shall be reasonably easy to recover the system from failures, e.g. without taking down the power

1. **Unambiguous to the intended readership**
   - Two designers arrive at the same result
2. **Clear enough to test**
   - Two testers get same result
3. **No design mixed in requirements**

167

Early Inspection

Prevention costs less than Repair



168

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 169

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 170

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 171

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 172

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 173

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...

E8 174

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

E8 175

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 176

Schedule

November	Sat 27
09:30~11:00	1:30
break	0:15
11:15~12:45	1:30
lunch	1:00
13:45~15:15	1:30
lunch	0:15
15:30~17:00	1:30

177