CHINA INSTITUTE for
INNOVATION

Workshop

Beijing - Shanghai
December 2010

Niels Malotaux

Predictable Projects

Delivering the Right Result at the Right Time

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

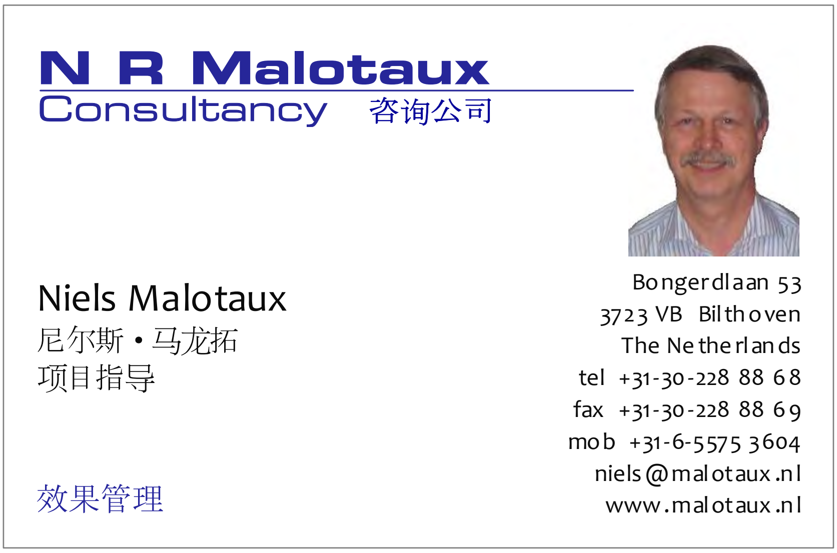
Niels Malotaux

Niels Malotaux is an independent Project Coach and expert in optimizing project performance. He has some 35 years experience in designing hardware and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading his own systems design company. Since 1998 he devotes his expertise to helping projects to deliver Quality On Time: delivering what the customer needs, when he needs it, to enable customer success. Niels effectively teaches Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached well over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, Ireland, India, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less well in practice. He is a frequent speaker at conferences, see www.malotaux.nl/nrm/Conf

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

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

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



N R Malotaux
Consultancy 咨询公司

Niels Malotaux
尼尔斯・马龙拓
项目指导

效果管理

Bongerdlaan 53
3723 VB Bilthoven
The Netherlands
tel +31-30-228 88 68
fax +31-30-228 88 69
mob +31-6-5575 3604
niels@malotaux.nl
www.malotaux.nl

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

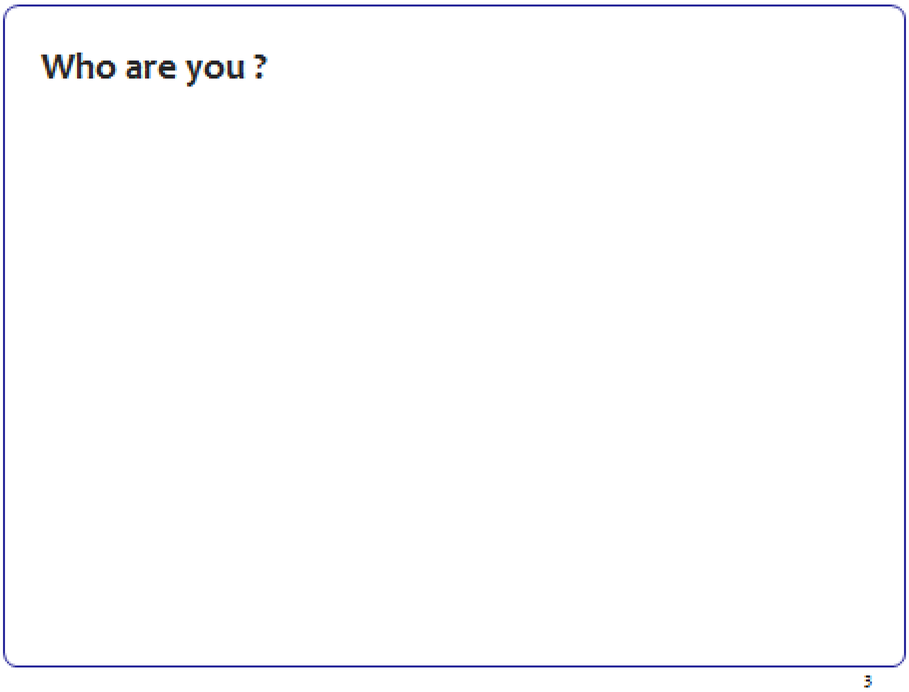
Niels Malotaux

Result Management

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

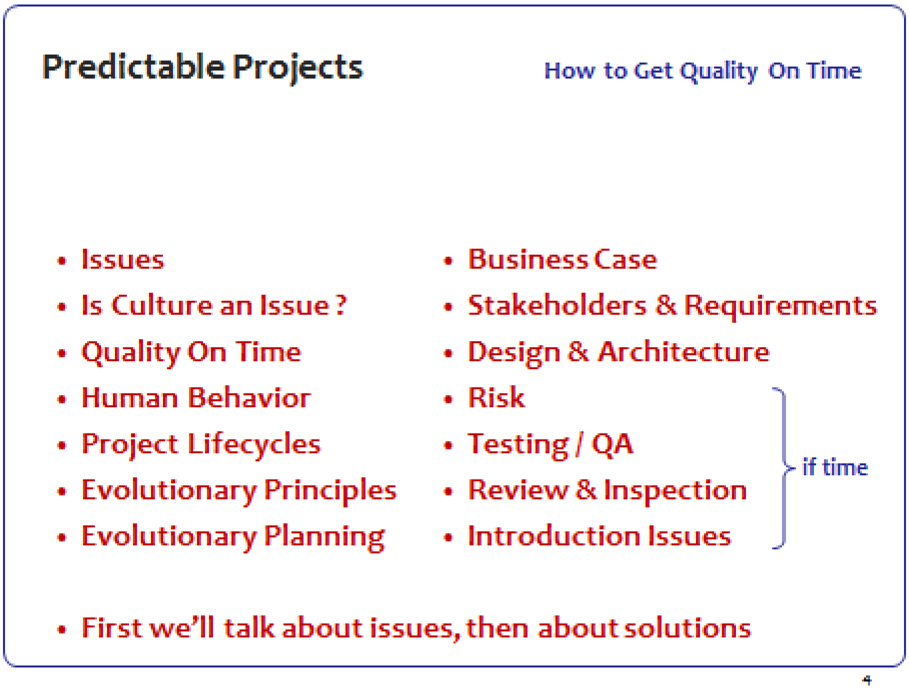
- Researching problems in projects
- Finding ways to fundamentally overcoming these problems
- Ploughing back into projects
- Tuning of the results (because theory isn't practice)

2



Who are you ?

3



Predictable Projects

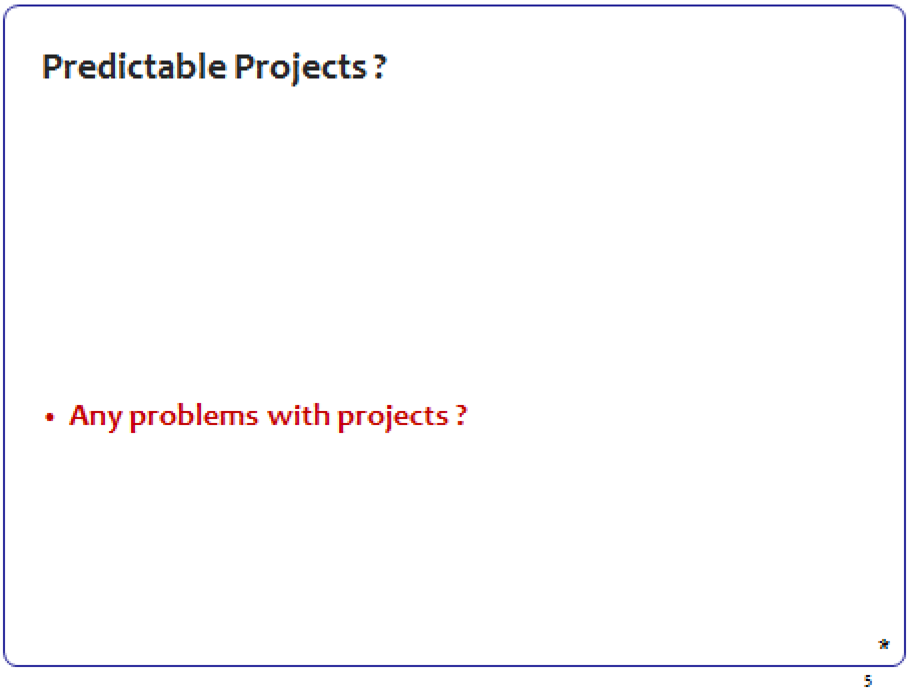
How to Get Quality On Time

- Issues
- Is Culture an Issue ?
- Quality On Time
- Human Behavior
- Project Lifecycles
- Evolutionary Principles
- Evolutionary Planning

- Business Case
- Stakeholders & Requirements
- Design & Architecture
- Risk
- Testing / QA
- Review & Inspection
- Introduction Issues

- First we'll talk about issues, then about solutions

4

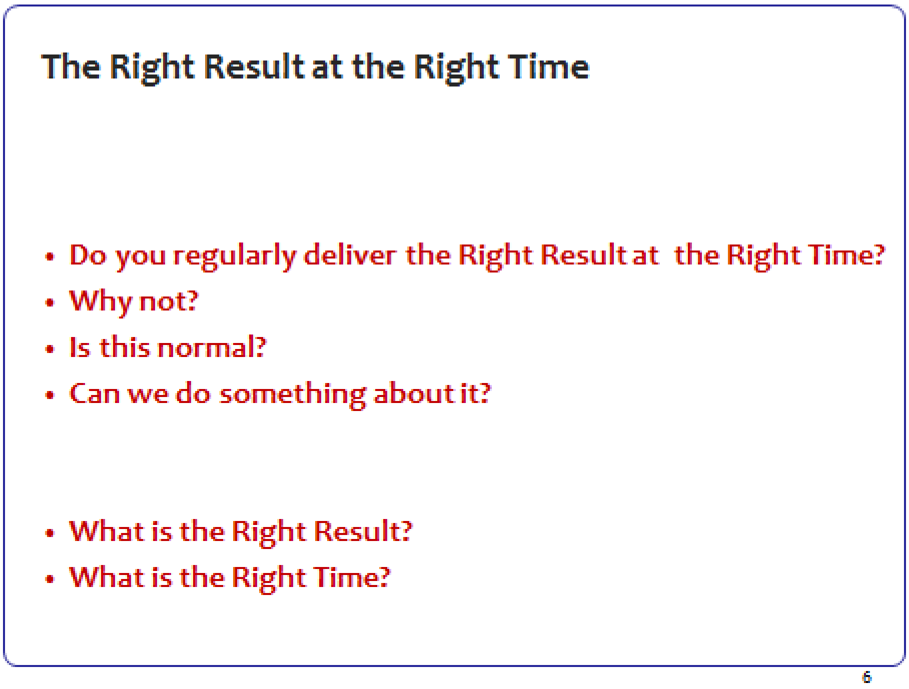


Predictable Projects ?

- Any problems with projects ?

*

5

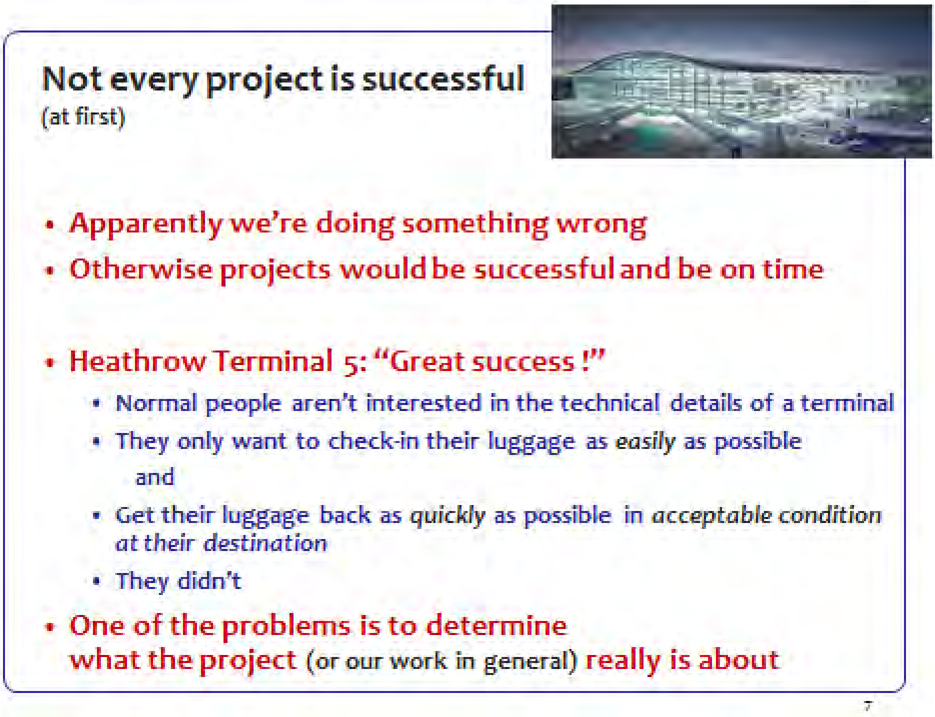


The Right Result at the Right Time

- Do you regularly deliver the Right Result at the Right Time?
- Why not?
- Is this normal?
- Can we do something about it?

- What is the Right Result?
- What is the Right Time?

6



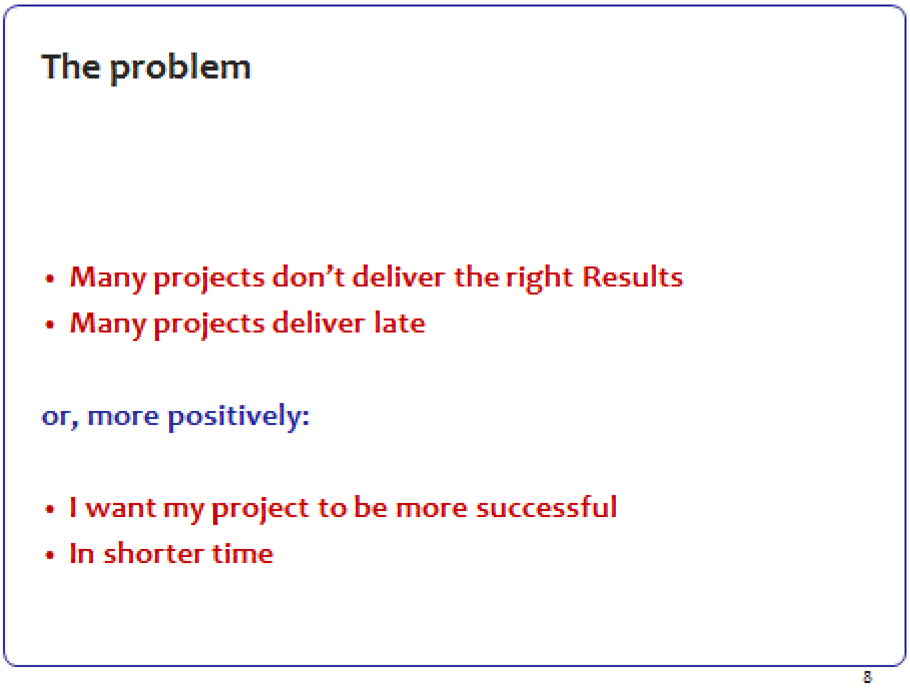
Not every project is successful
(at first)

- Apparently we're doing something wrong
- Otherwise projects would be successful 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

7



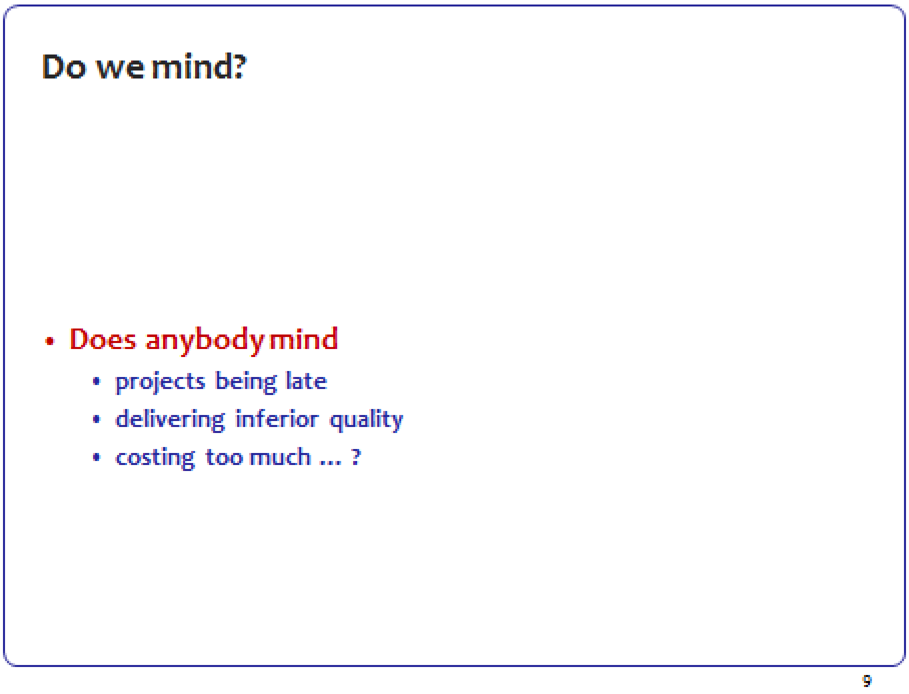
The problem

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

or, more positively:

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

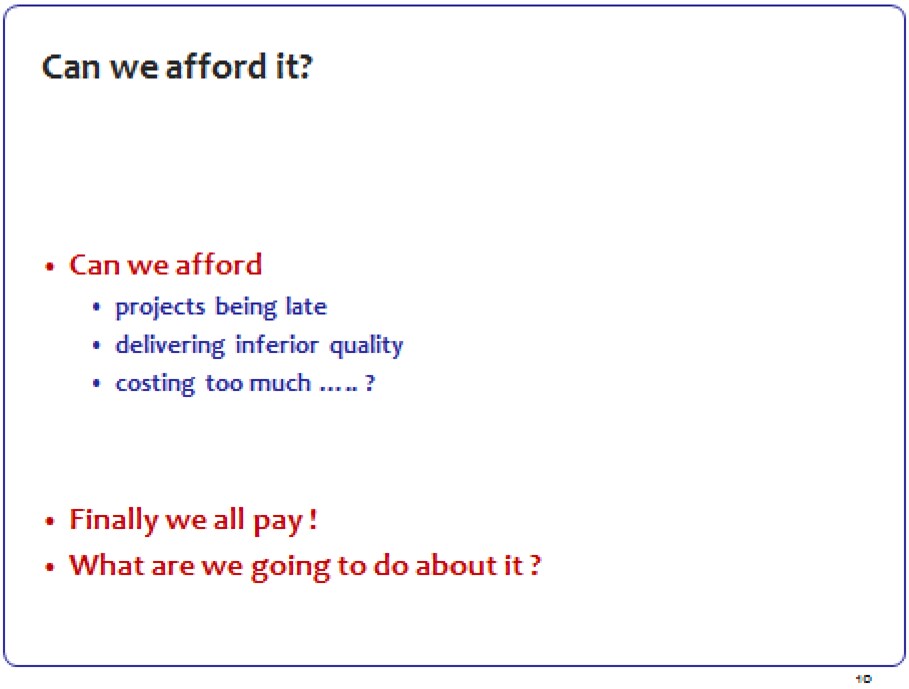
8



Do we mind?

- Does anybody mind
  - projects being late
  - delivering inferior quality
  - costing too much ... ?

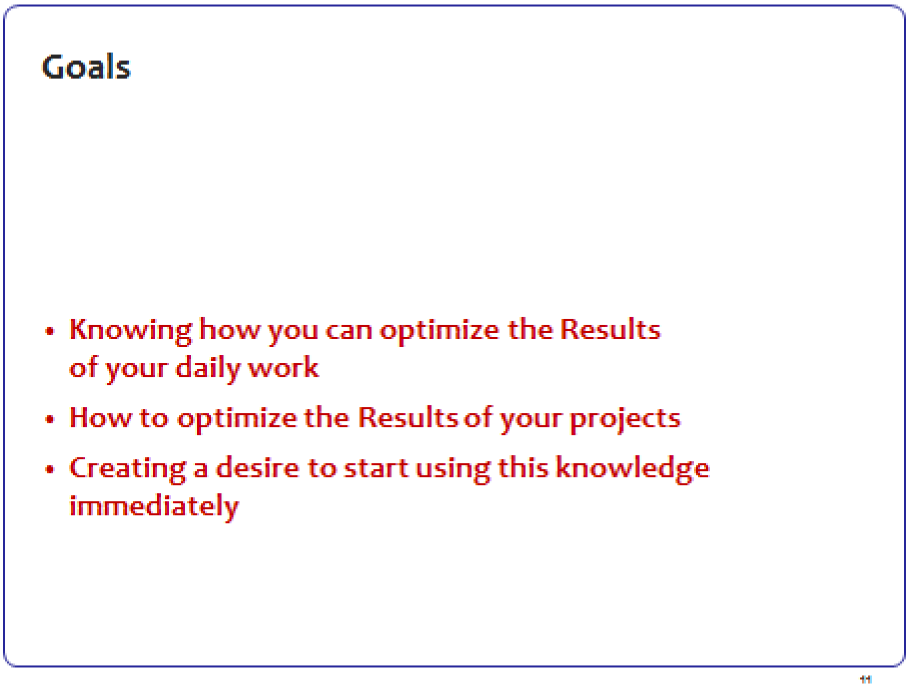
9



Can we afford it?

- Can we afford
  - projects being late
  - delivering inferior quality
  - costing too much ….. ?
- Finally we all pay !
- What are we going to do about it ?

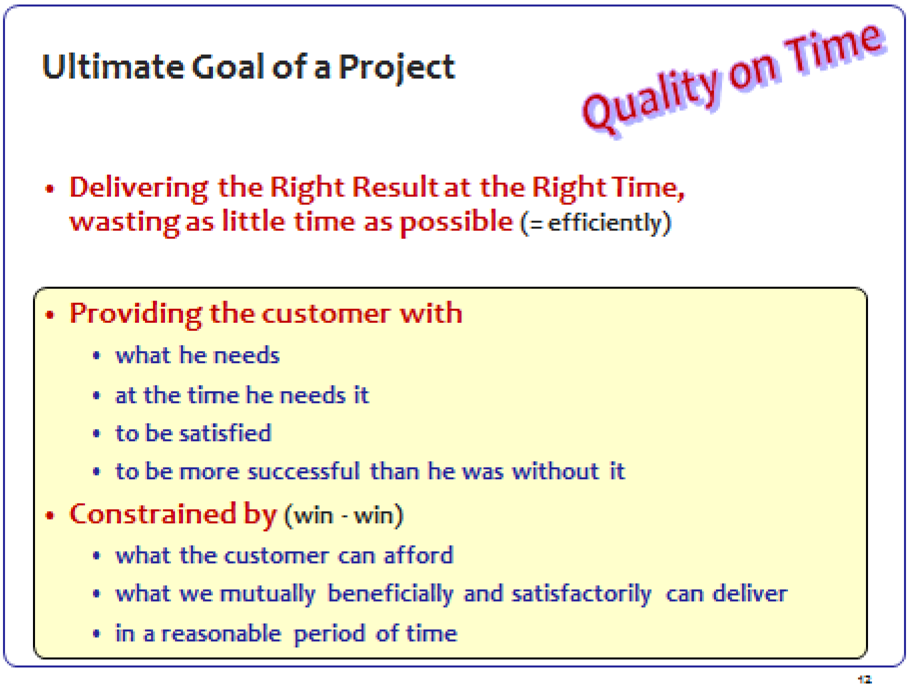
10



Goals

- Knowing how you can optimize the Results of your daily work
- How to optimize the Results of your projects
- Creating a desire to start using this knowledge immediately

11



Ultimate Goal of a Project

Quality on Time

- Delivering the Right Result at the Right Time, wasting as little time as possible (= efficiently)

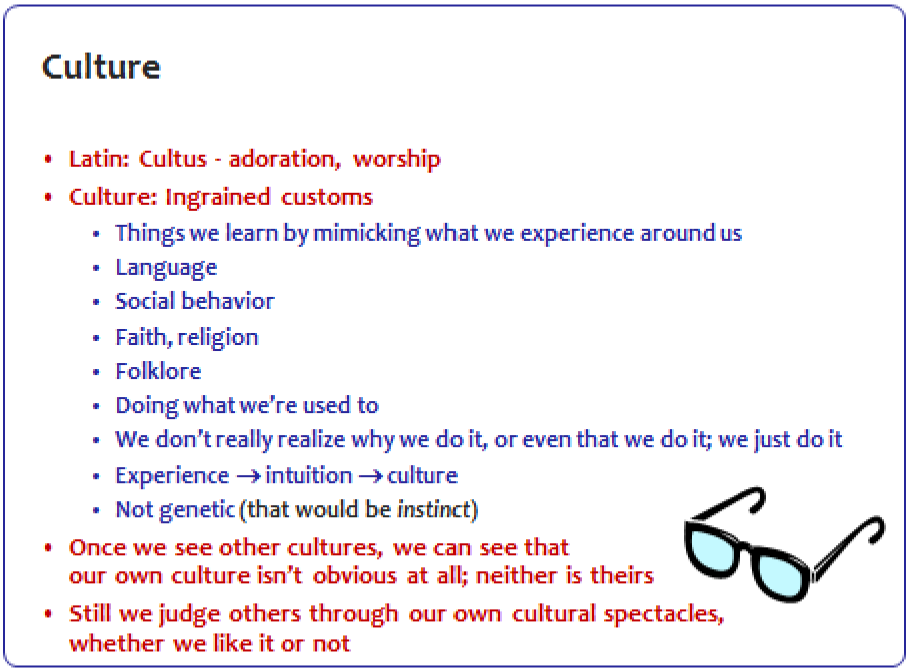
- Providing the customer with
  - what he needs
  - at the time he needs it
  - to be satisfied
  - to be more successful than he was without it
- Constrained by (win - win)
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of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

12

Is Culture an Issue ?

Persons per sq. km
Over 100
50-100
10-50
1-10
0-1
Persons per sq. mile
Over 250
125-250
25-125
2-25
0-2

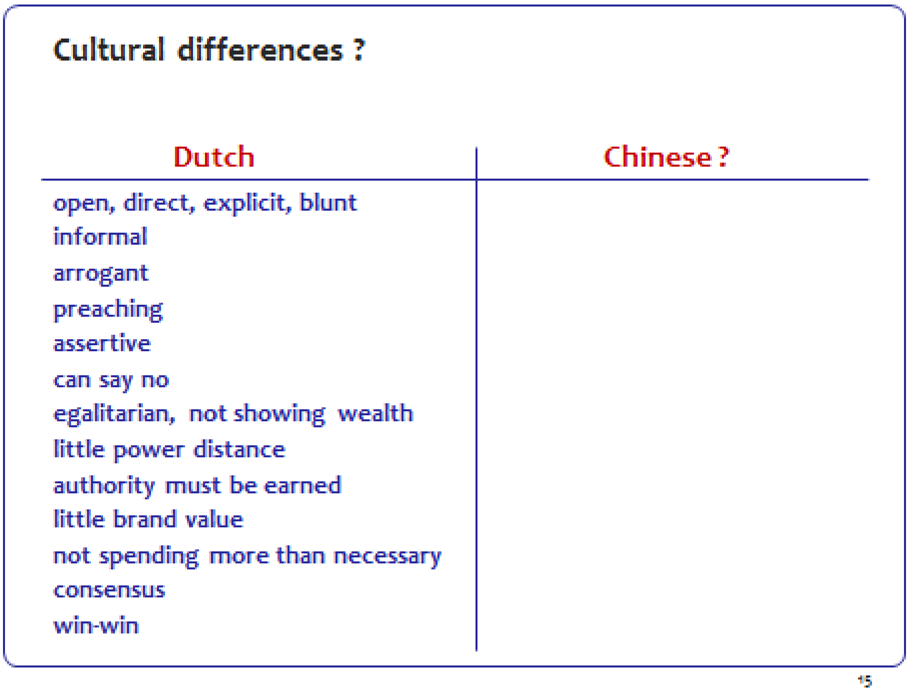
13



Culture

- Latin: Cultus - adoration, worship
- Culture: Ingrained customs
  - Things we learn by mimicking what we experience around us
  - Language
  - Social behavior
  - Faith, religion
  - Folklore
  - Doing what we're used to
  - We don't really realize why we do it, or even that we do it; we just do it
  - Experience → intuition → culture
  - Not genetic (that would be *instinct*)
- Once we see other cultures, we can see that our own culture isn't obvious at all; neither is theirs
- Still we judge others through our own cultural spectacles, whether we like it or not

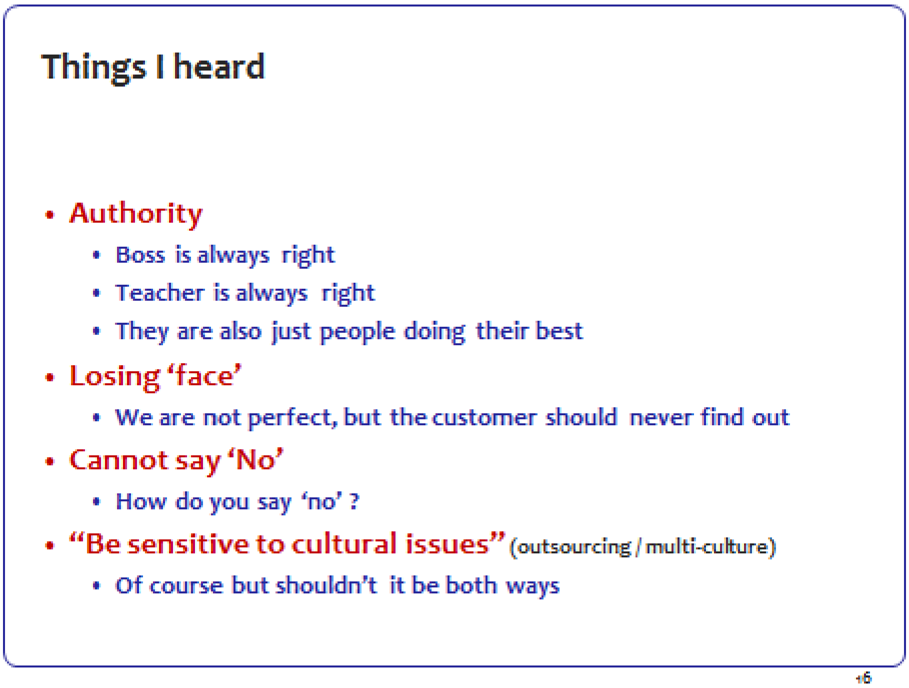
14



Cultural differences ?

Dutch	Chinese ?
open, direct, explicit, blunt	
informal	
arrogant	
preaching	
assertive	
can say no	
egalitarian, not showing wealth	
little power distance	
authority must be earned	
little brand value	
not spending more than necessary	
consensus	
win-win	

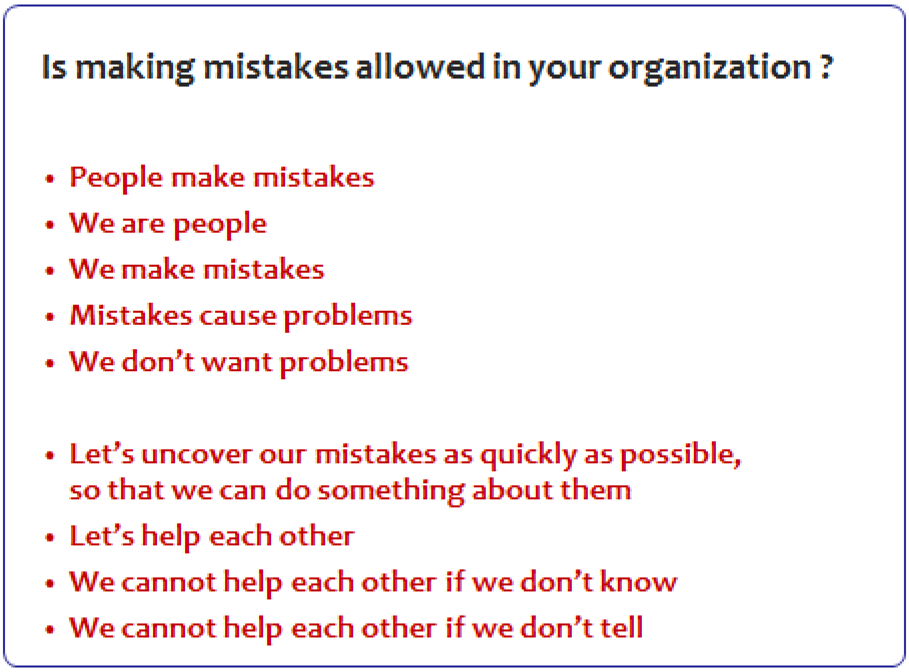
15



Things I heard

- Authority
  - Boss is always right
  - Teacher is always right
  - They are also just people doing their best
- Losing 'face'
  - We are not perfect, but the customer should never find out
- Cannot say 'No'
  - How do you say 'no' ?
- "Be sensitive to cultural issues" (outsourcing / multi-culture)
  - Of course but shouldn't it be both ways

16

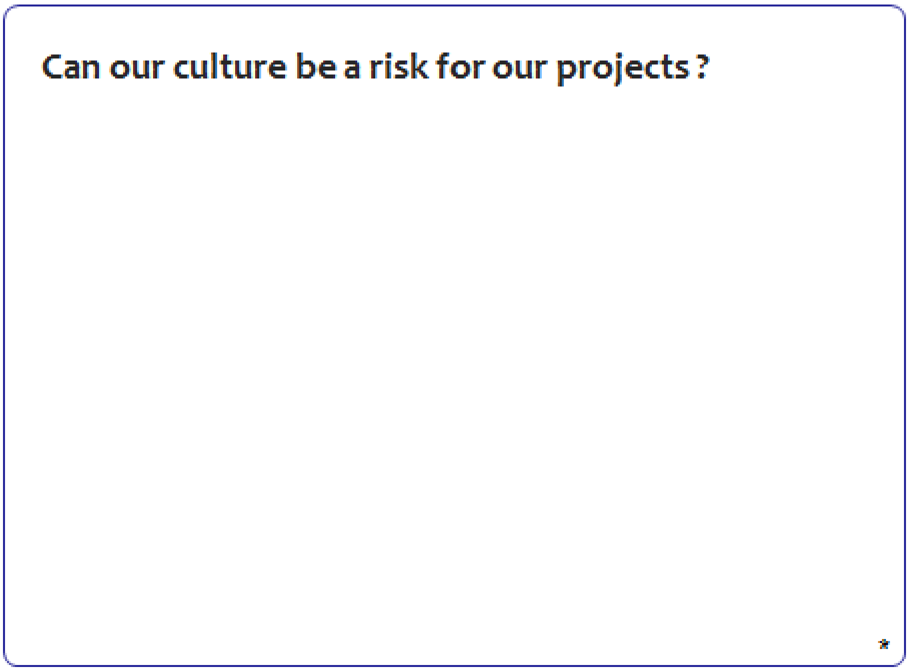


Is making mistakes allowed in your organization ?

- People make mistakes
- We are people
- We make mistakes
- Mistakes cause problems
- We don't want problems

- Let's uncover our mistakes as quickly as possible, so that we can do something about them
- Let's help each other
- We cannot help each other if we don't know
- We cannot help each other if we don't tell

17



Can our culture be a risk for our projects ?

18

Quality on Time

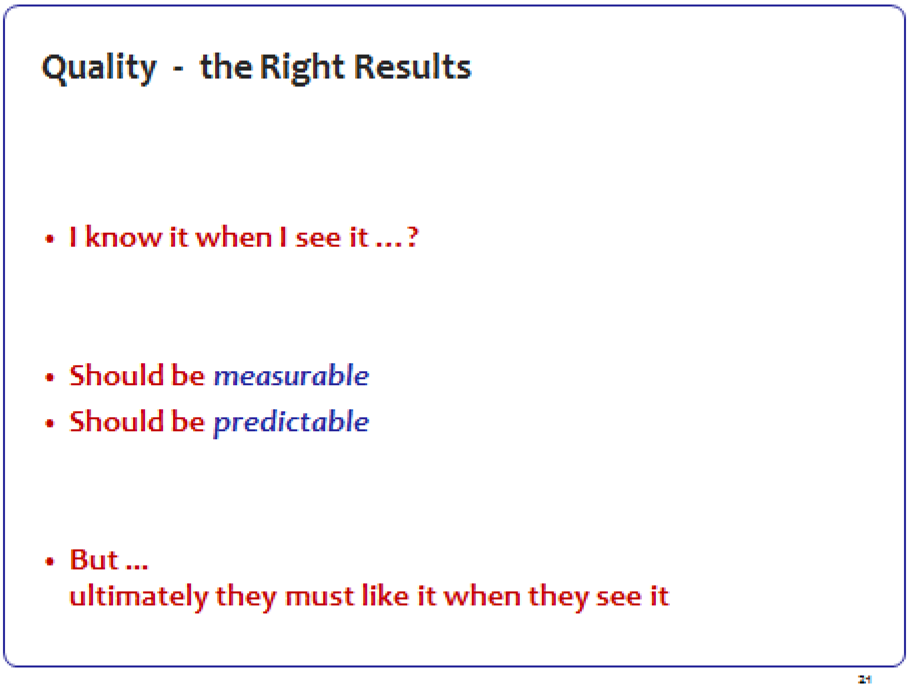
The Right things at the Right time

19

Quality On Time

- What is Quality?
- What is On Time?

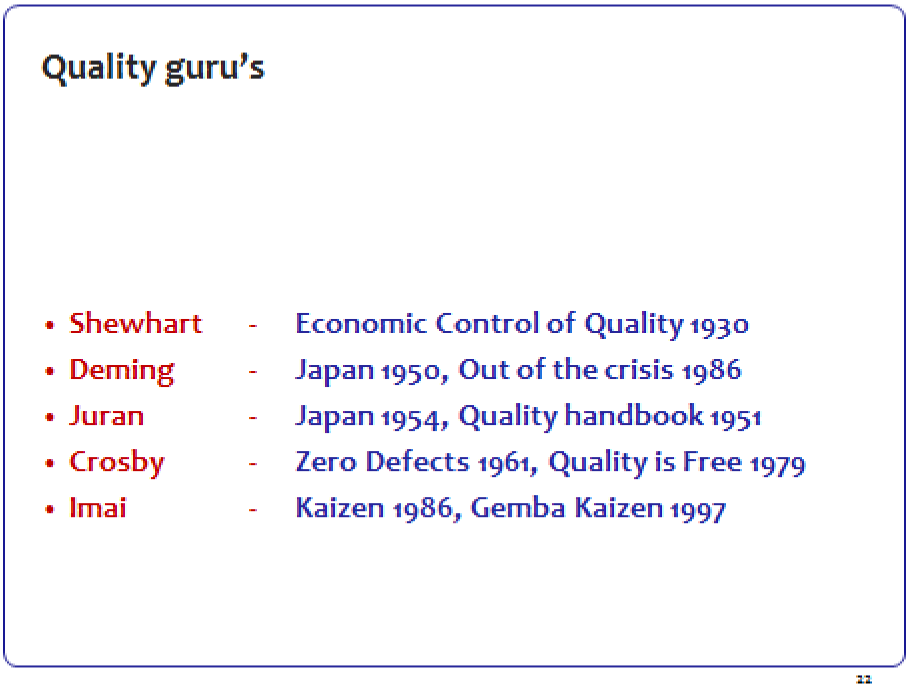
20



Quality - the Right Results

- I know it when I see it ... ?
- Should be *measurable*
- Should be *predictable*
- But ...
  ultimately they must like it when they see it

21

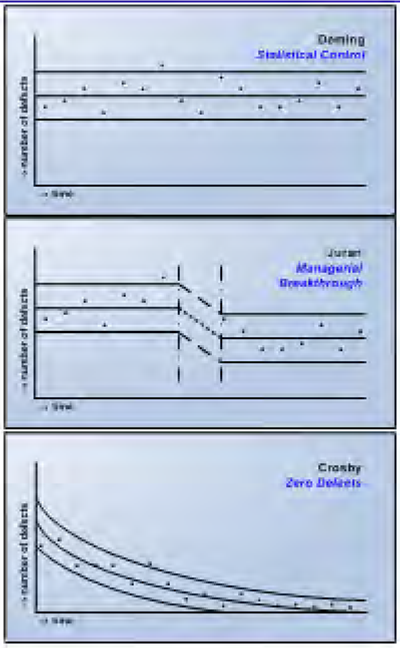


Quality guru's

• Shewhart	-	Economic Control of Quality 1930
• Deming	-	Japan 1950, Out of the crisis 1986
• Juran	-	Japan 1954, Quality handbook 1951
• Crosby	-	Zero Defects 1961, Quality is Free 1979
• Imai	-	Kaizen 1986, Gemba Kaizen 1997

22

Deming - Juran - Crosby



23

Deming

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

24

Absolutes of Quality

- Conformance to requirements
- Obtained through prevention
- Performance standard is zero defects
- Measured by the price of non-conformance (PONC)

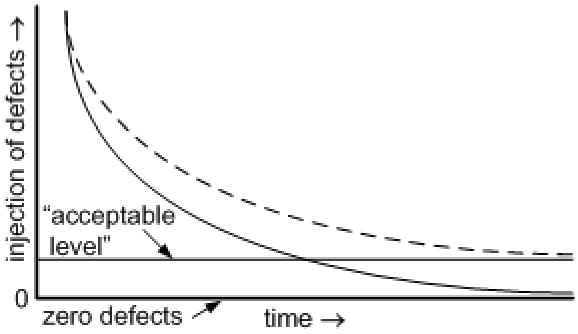
Philip Crosby, 1970

- The purpose is customer success (not customer satisfaction)

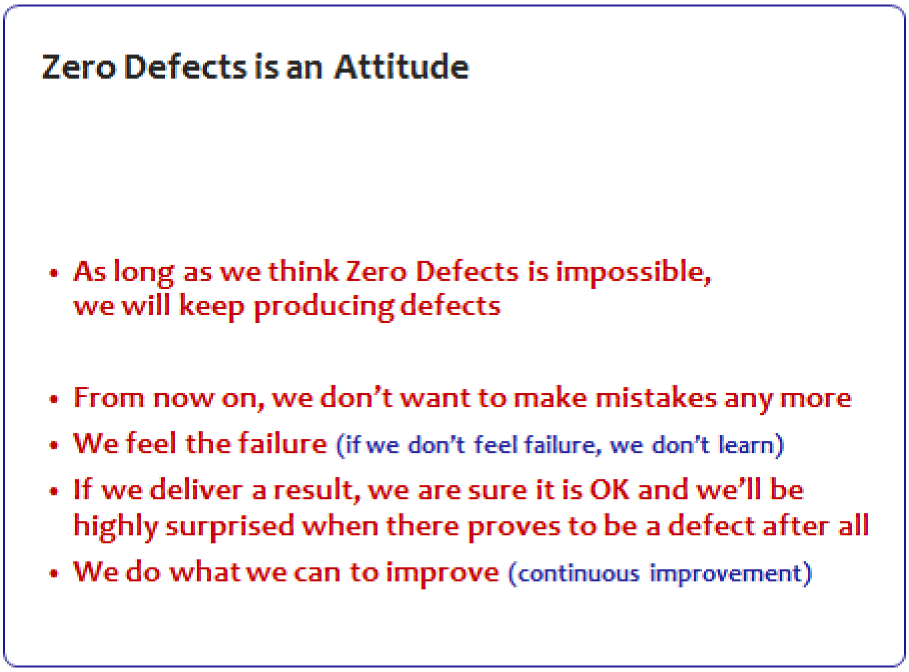
Added by Philip Crosby Associates, 2004

Is Zero Defects possible ?

- Zero Defects is an asymptote



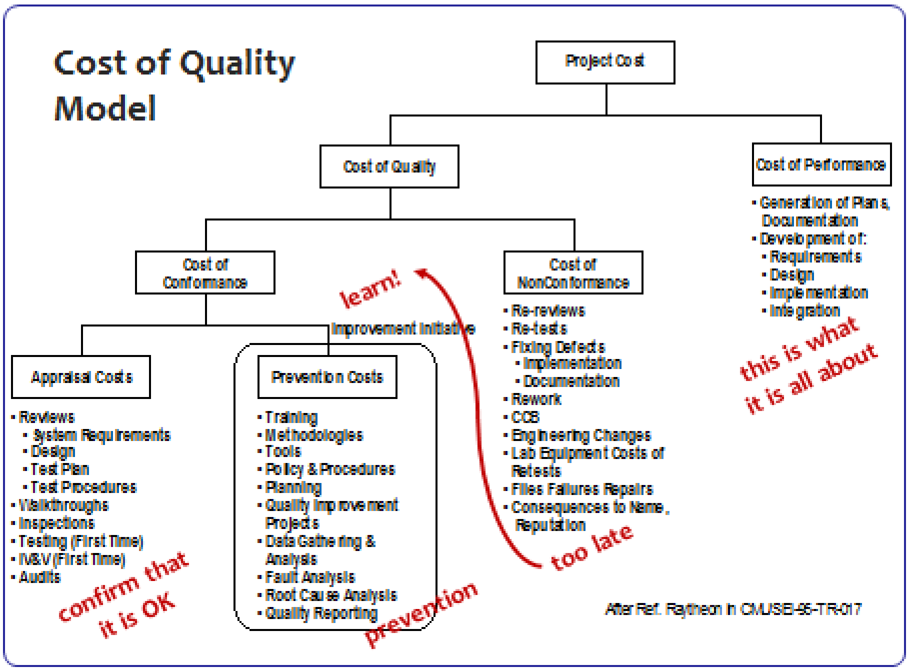
- When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately



Zero Defects is an Attitude

- As long as we think Zero Defects is impossible, we will keep producing defects
- From now on, we don't want to make mistakes any more
- We feel the failure (if we don't feel failure, we don't learn)
- If we deliver a result, we are sure it is OK and we'll be highly surprised when there proves to be a defect after all
- We do what we can to improve (continuous improvement)

27



Cost of Quality Model

Project Cost

Cost of Quality

Cost of Performance
- Generation of Plans, Documentation
- Development of:
  - Requirements
  - Design
  - Implementation
  - Integration

Cost of Conformance

Cost of NonConformance
- Re-reviews
- Re-tests
- Fixing Defects
  - Implementation
  - Documentation
- Rework
- CCB
- Engineering Changes
- Lab Equipment Costs of Retests
- Files Failures Repairs
- Consequences to Name, Reputation

Appraisal Costs
- Reviews
  - System Requirements
  - Design
  - Test Plan
  - Test Procedures
- Walkthroughs
- Inspections
- Testing (First Time)
- IV&V (First Time)
- Audits

Improvement Initiative

Prevention Costs
- Training
- Methodologies
- Tools
- Policy & Procedures
- Planning
- Quality Improvement Projects
- Data Gathering & Analysis
- Fault Analysis
- Root Cause Analysis
- Quality Reporting

learn!

confirm that it is OK

prevention

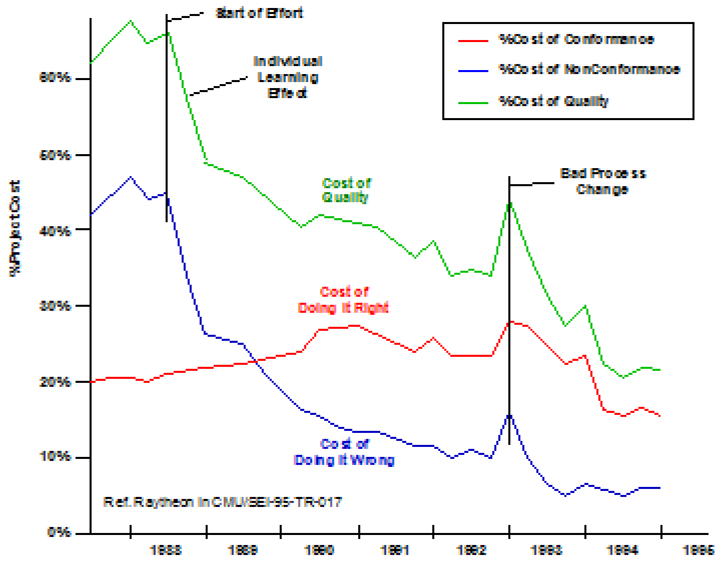
too late

this is what it is all about

After Ref. Raytheon in CMU/SEI-95-TR-017

28

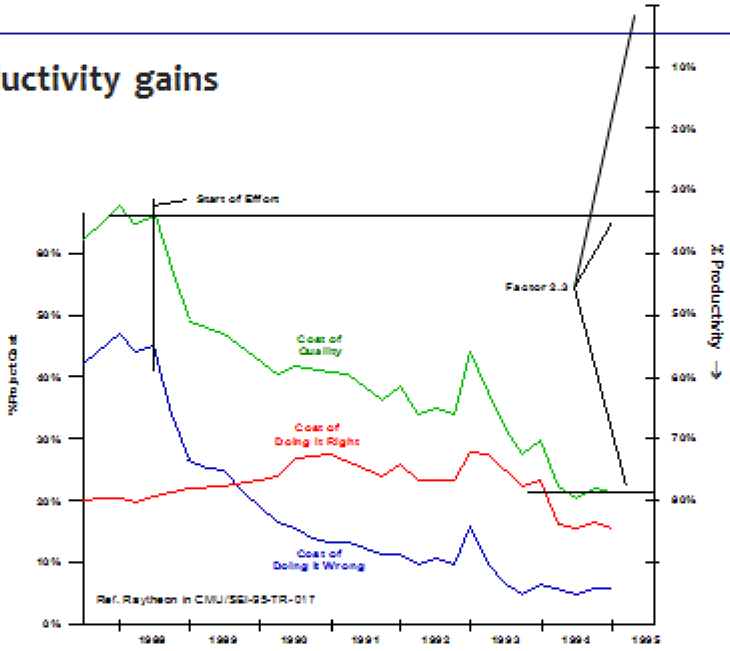
Cost of Quality



Ref. Raytheon in CMU/SEI-95-TR-017

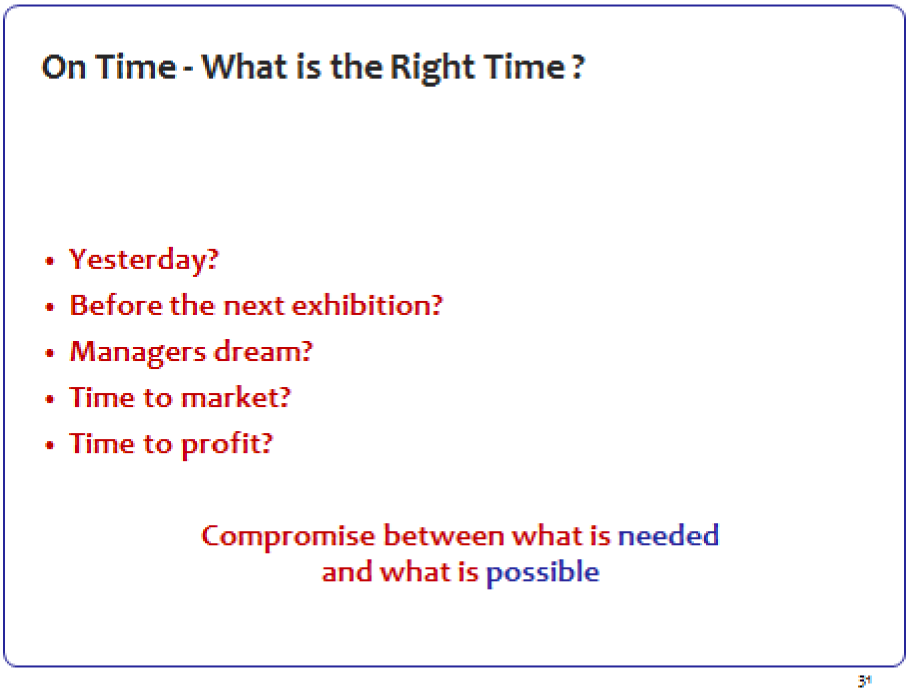
29

Productivity gains



Ref. Raytheon in CMU/SEI-95-TR-017

30

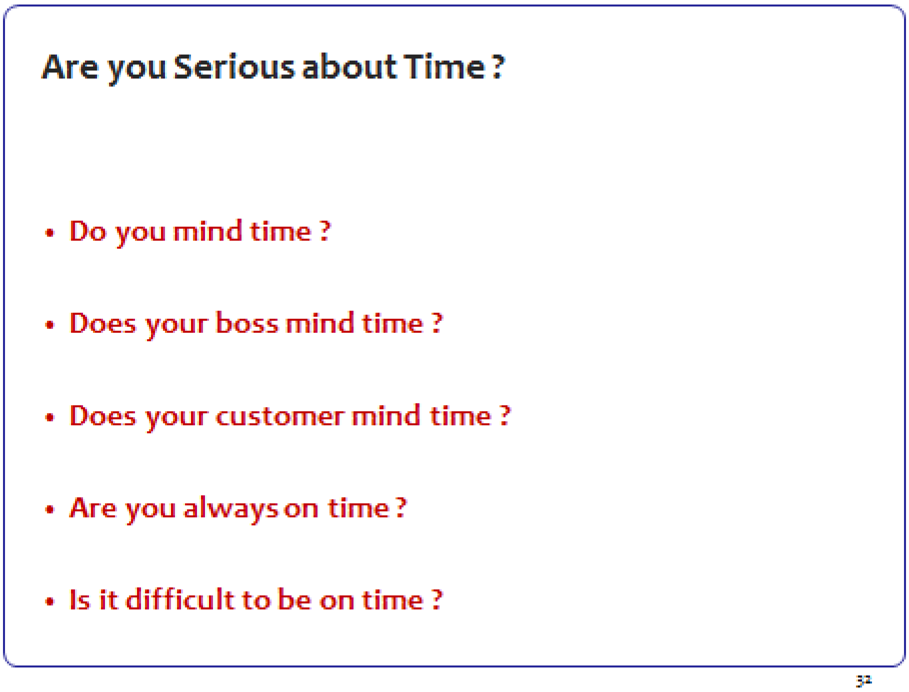


On Time - What is the Right Time ?

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

Compromise between what is needed
and what is possible

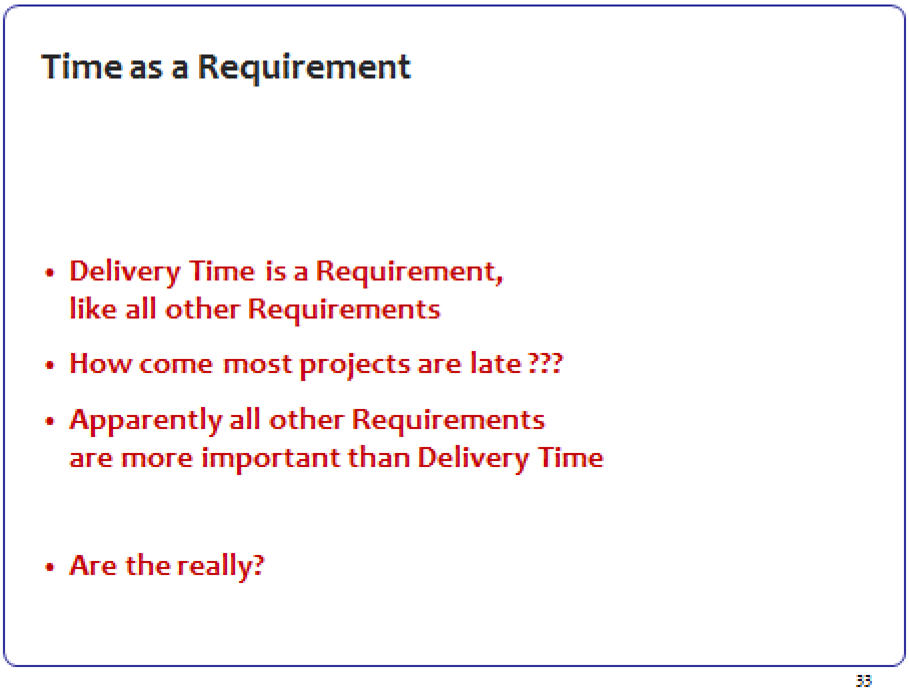
31



Are you Serious about Time ?

- Do you mind time ?
- Does your boss mind time ?
- Does your customer mind time ?
- Are you always on time ?
- Is it difficult to be on time ?

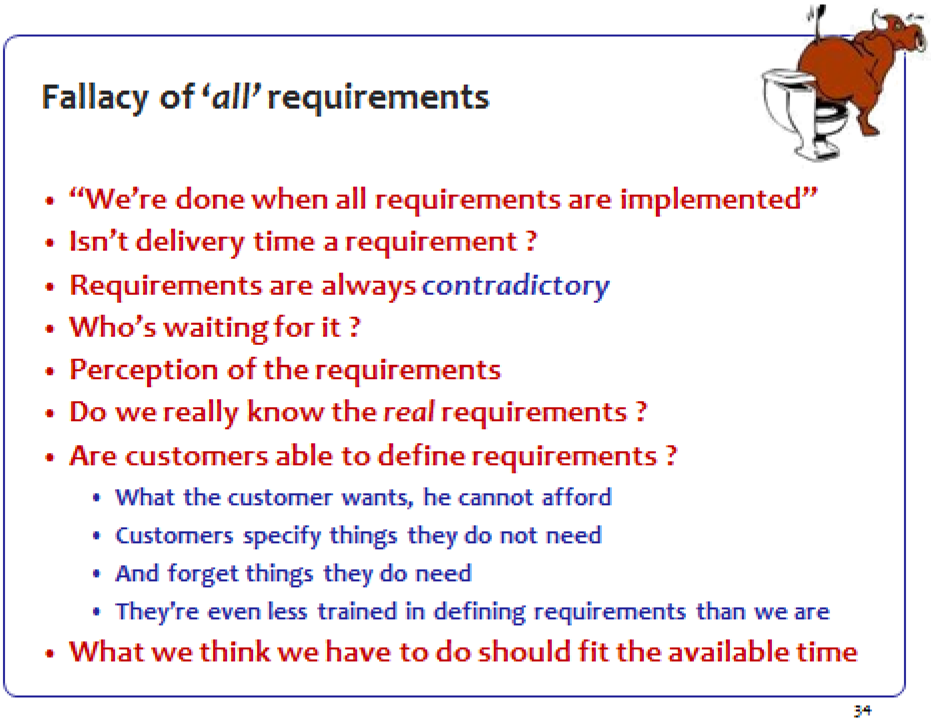
32



Time as a 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?

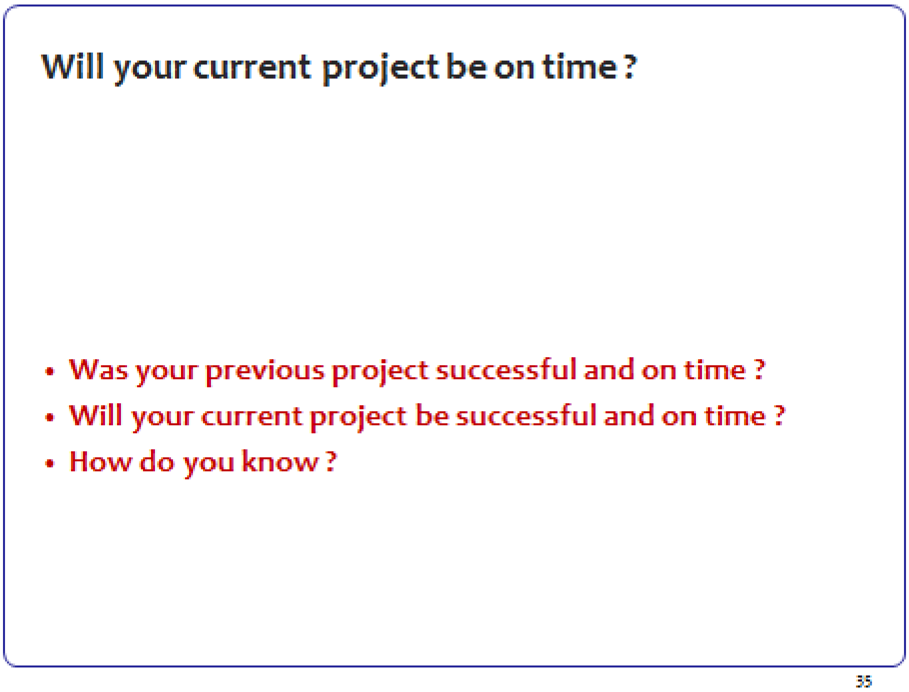
33



Fallacy of ‘*all*’ requirements

- “We’re done when all requirements are implemented”
- Isn’t delivery time a requirement ?
- Requirements are always *contradictory*
- Who’s waiting for it ?
- Perception of the requirements
- Do we really know the *real* requirements ?
- Are customers able to define requirements ?
  - What the customer wants, he cannot afford
  - 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

34



Will your current project be on time ?

- Was your previous project successful and on time ?
- Will your current project be successful and on time ?
- How do you know ?

35

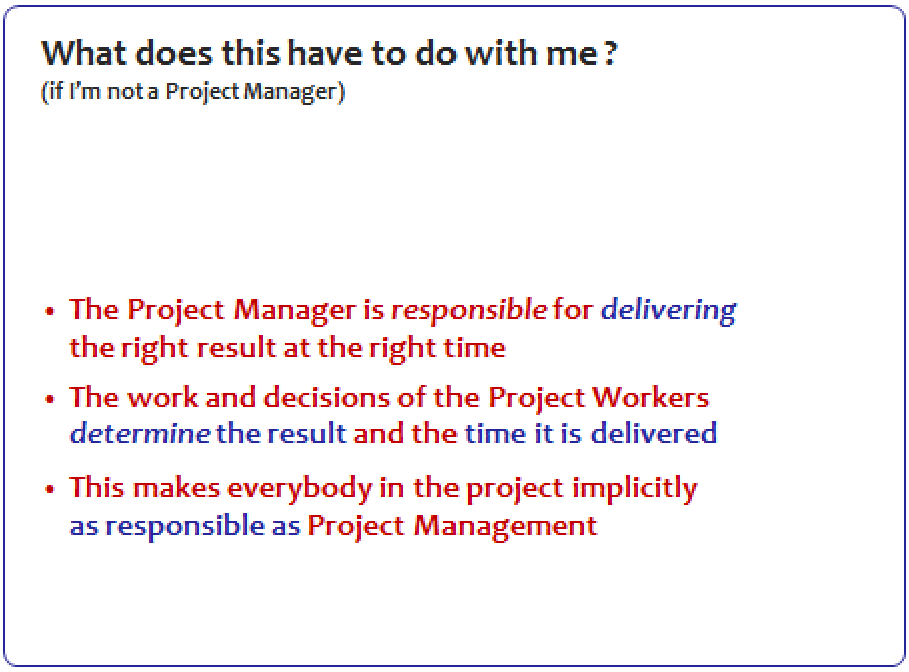
If our previous project was late,
our current project will also be late

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

36

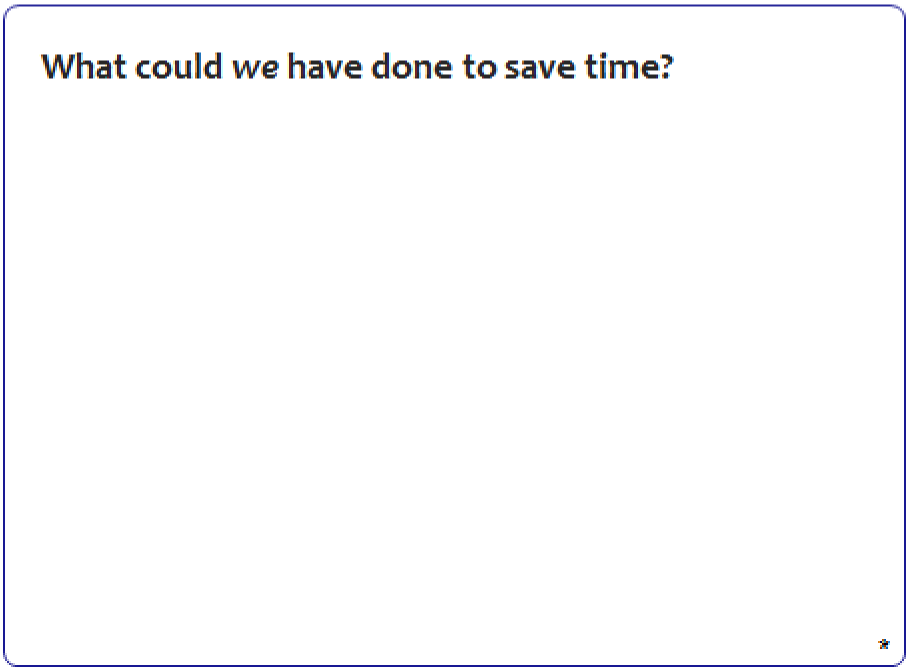


What does this have to do with me ?

(if I'm not a Project Manager)

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

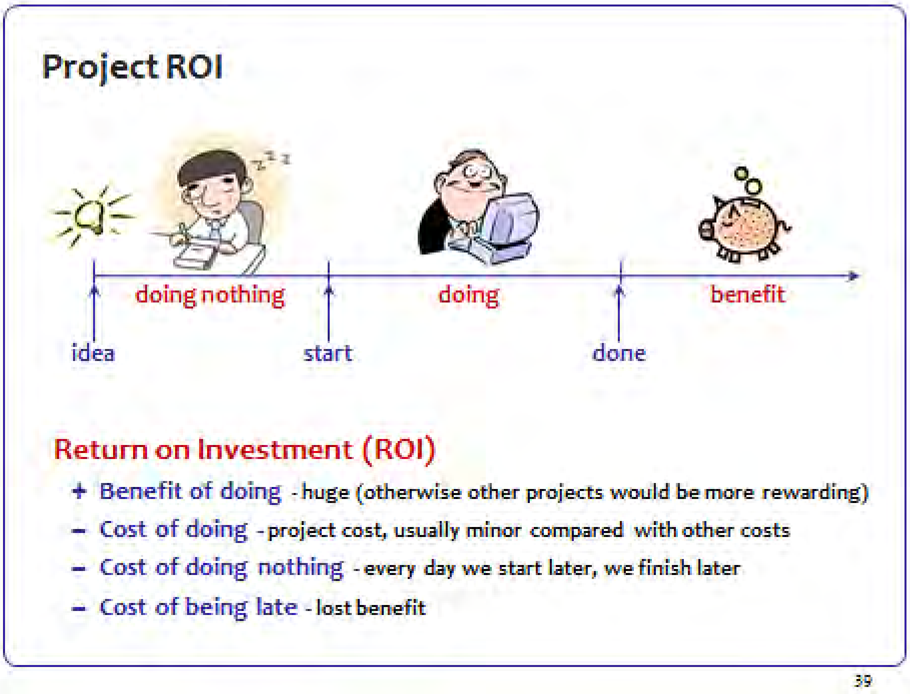
37



What could *we* have done to save time?

*

38



Project ROI

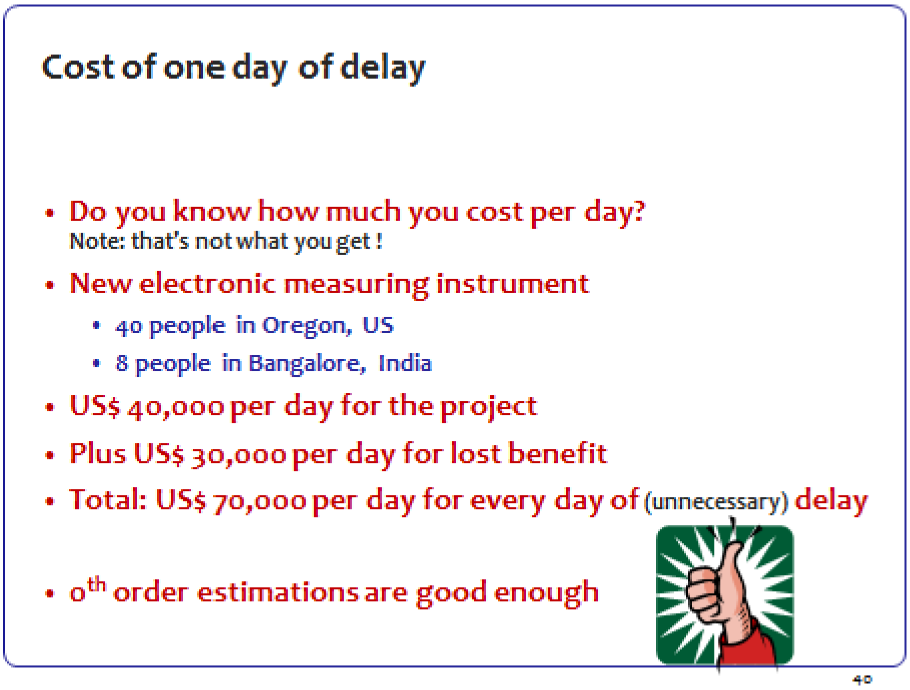
doing nothing
doing
benefit

idea
start
done

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

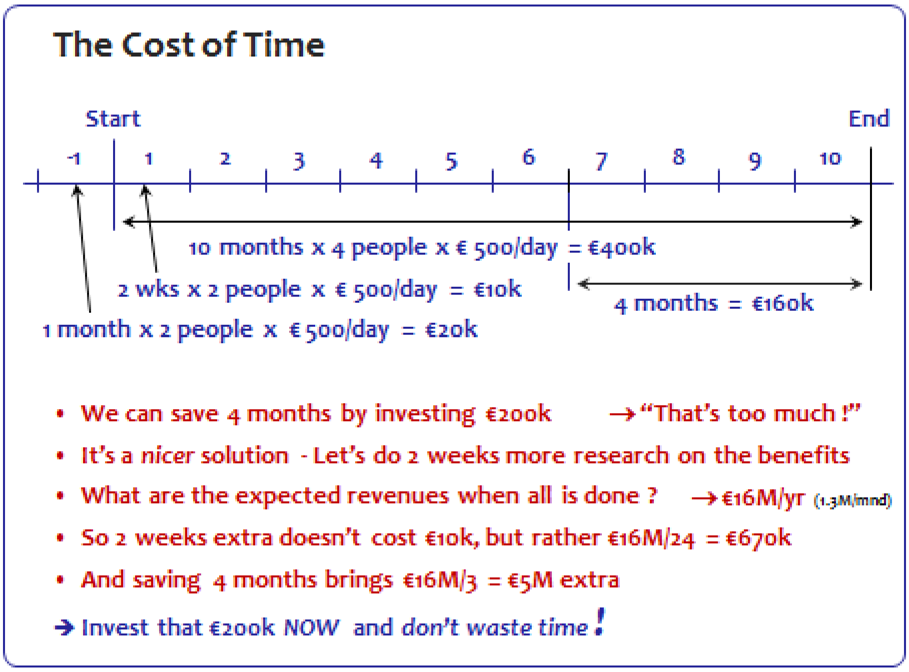
39



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

40



The Cost of Time

Start

End

-1 1 2 3 4 5 6 7 8 9 10

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

2 wks x 2 people x € 500/day = €10k

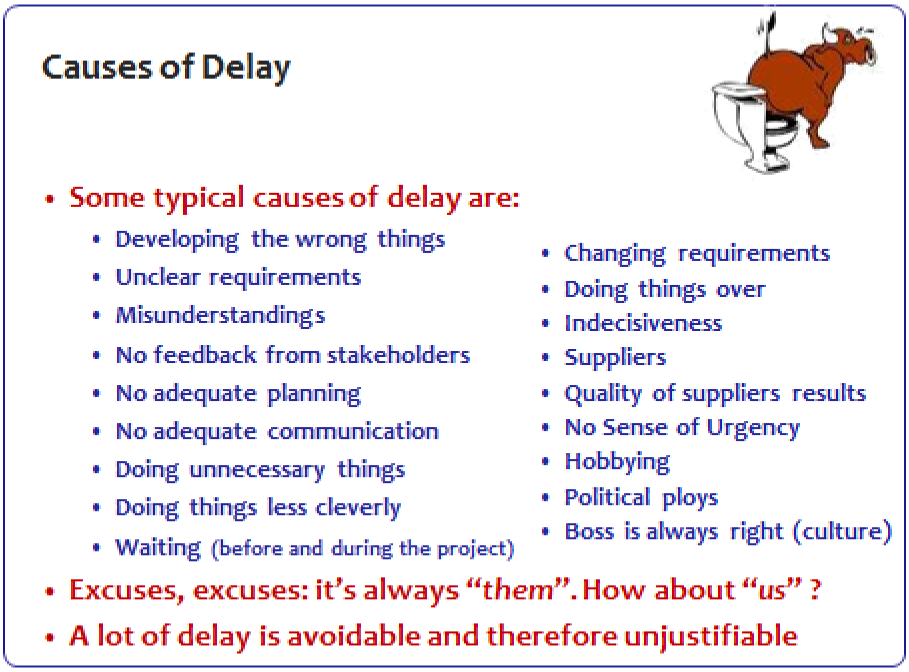
1 month x 2 people x € 500/day = €20k

4 months = €160k

- 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* !

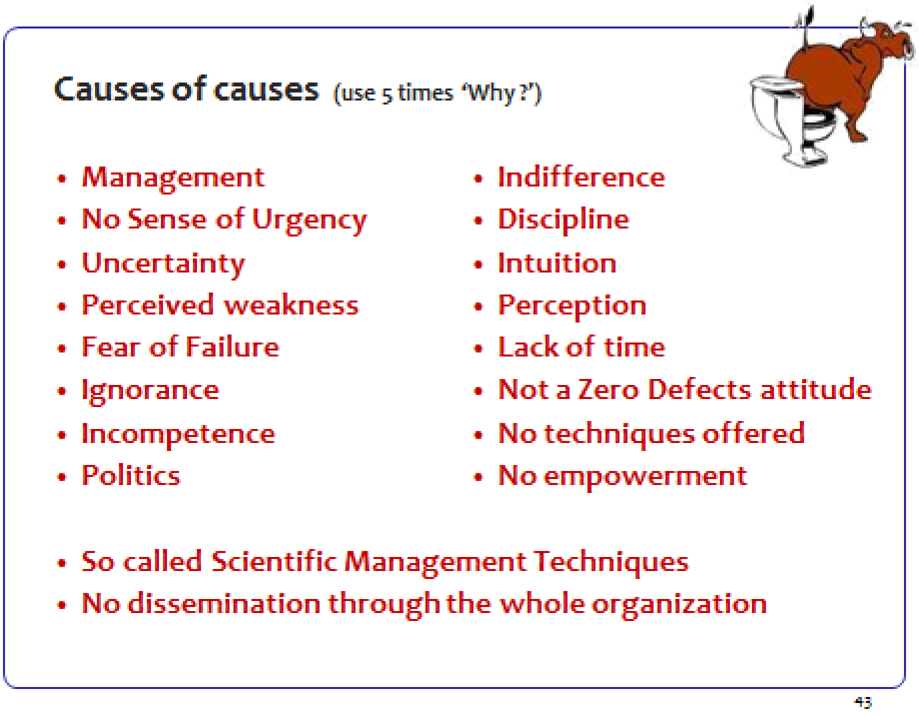
41



Causes of Delay

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

42

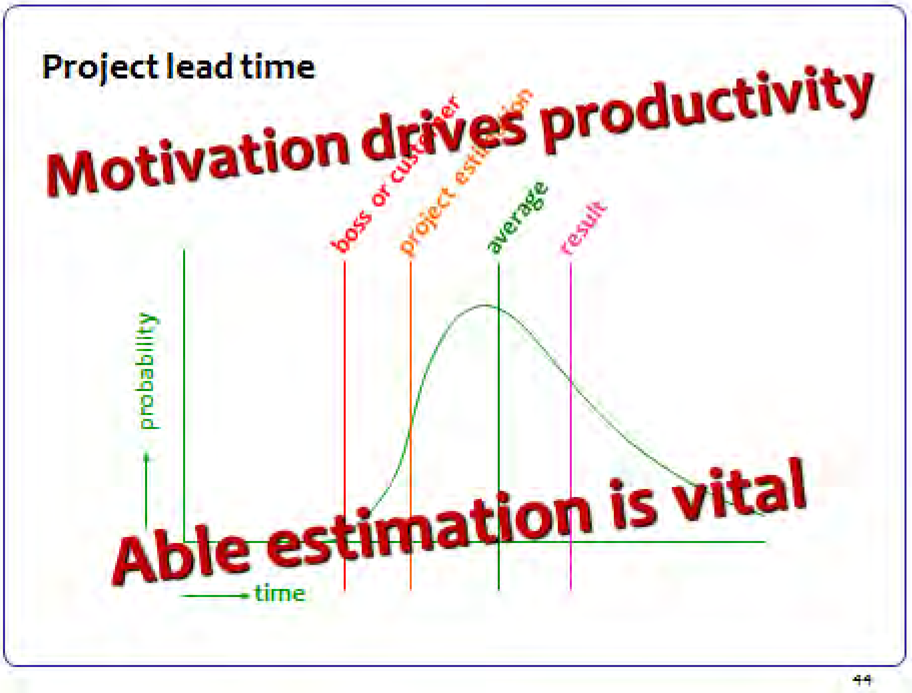


Causes of causes (use 5 times 'Why?')

- 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

43

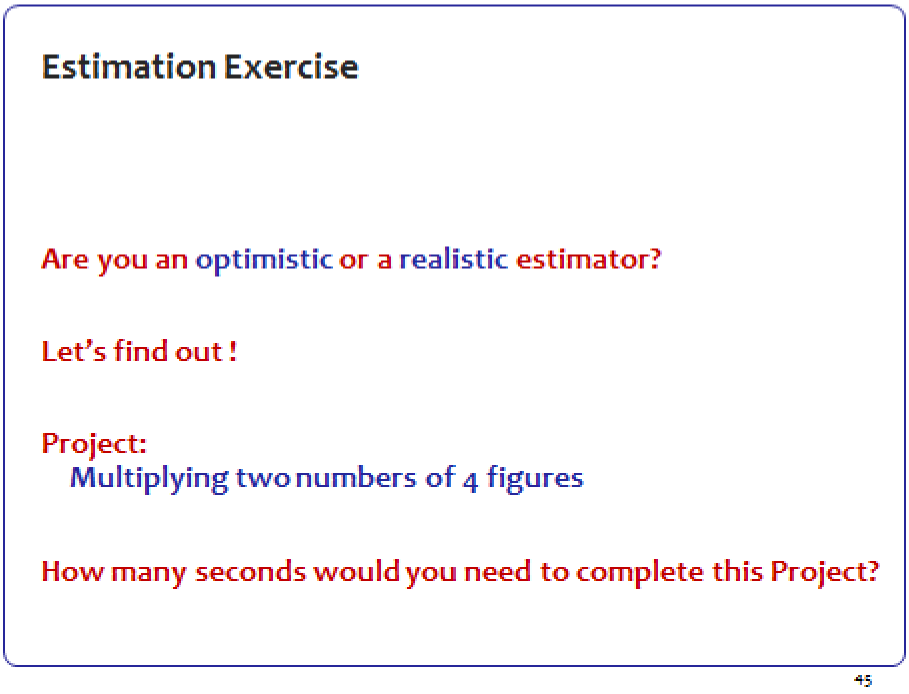


Project lead time

Motivation drives productivity

Able estimation is vital

44



Estimation Exercise

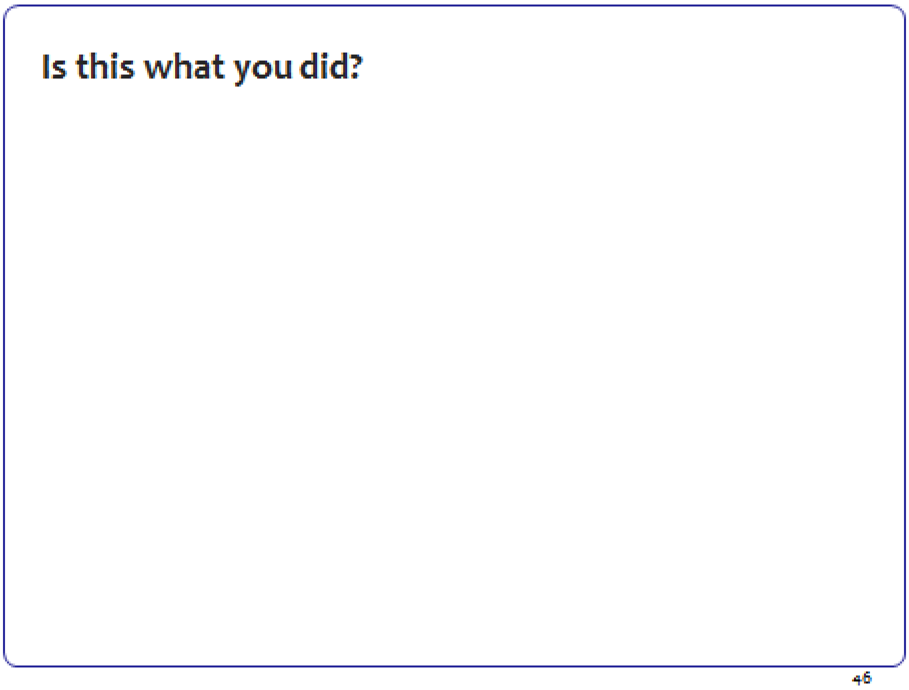
Are you an optimistic or a realistic estimator?

Let's find out !

Project:
Multiplying two numbers of 4 figures

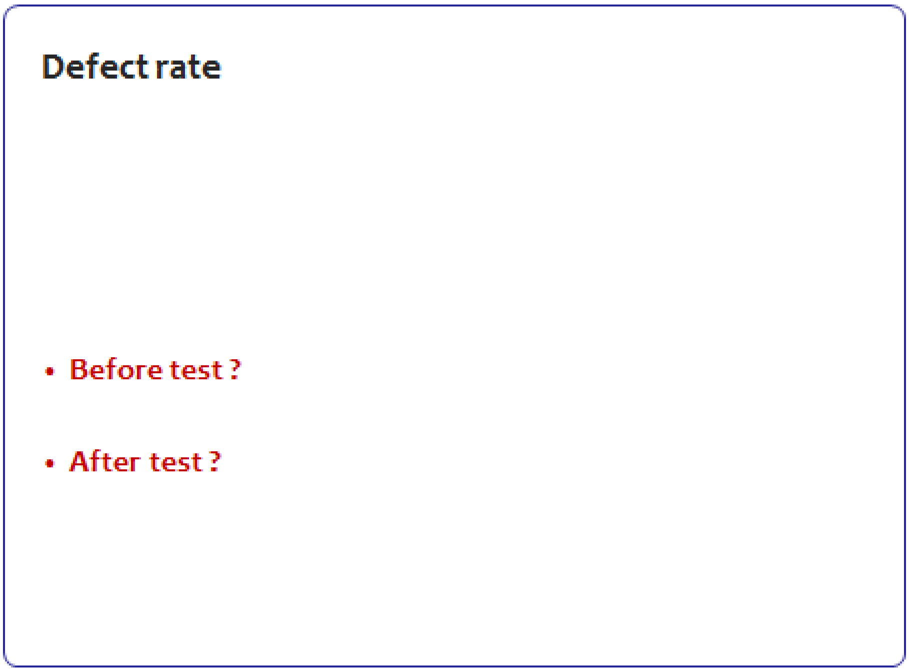
How many seconds would you need to complete this Project?

45



Is this what you did?

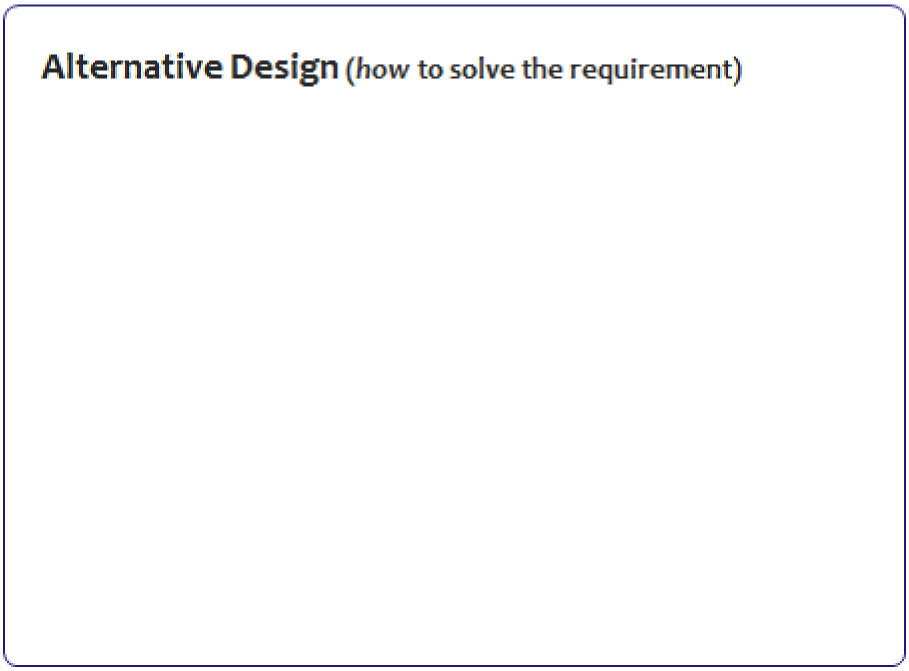
46



Defect rate

- Before test ?
- After test ?

47



Alternative Design (*how* to solve the requirement)

48

Another alternative design

There are usually more solutions than the obvious one

49

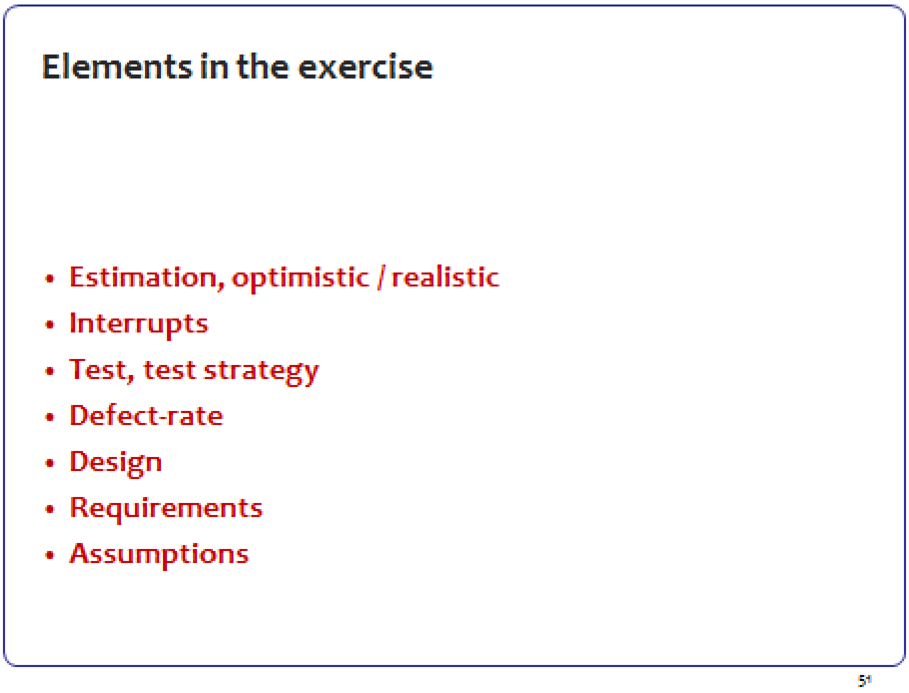
What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong

Check !

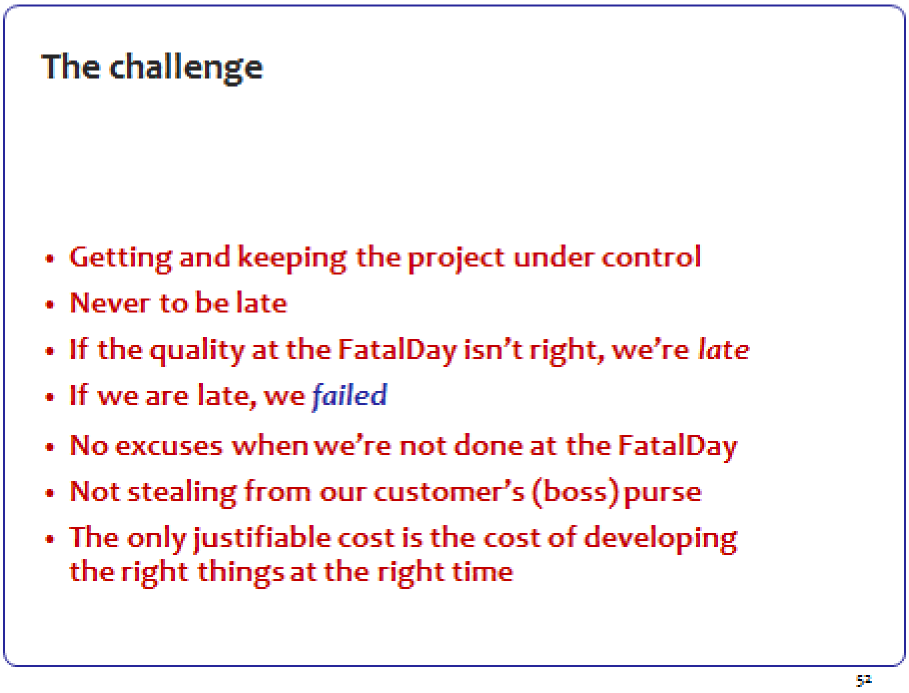
50



Elements in the exercise

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

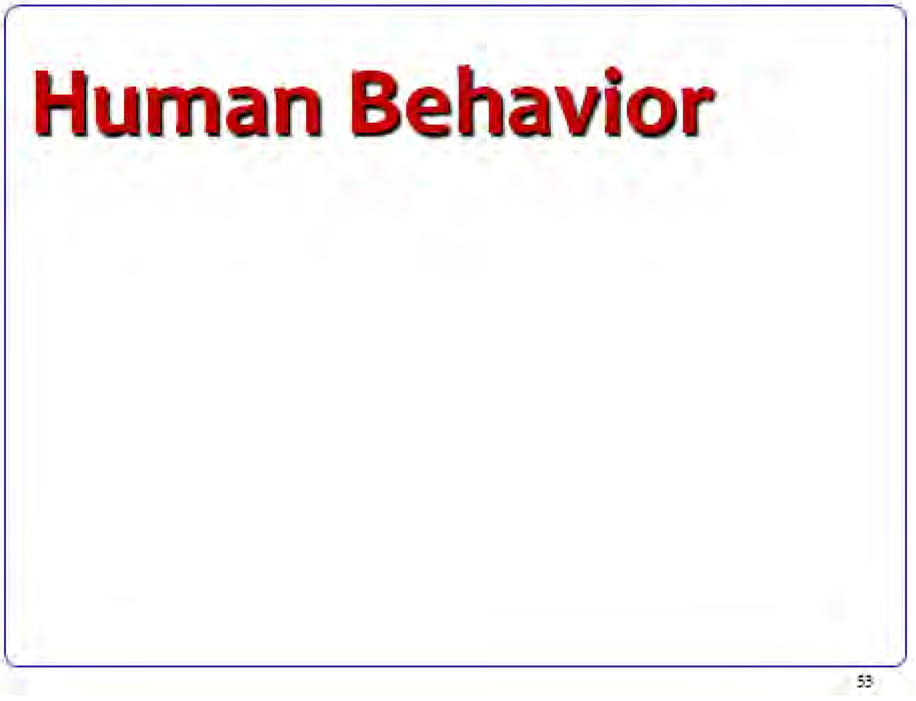
51



The challenge

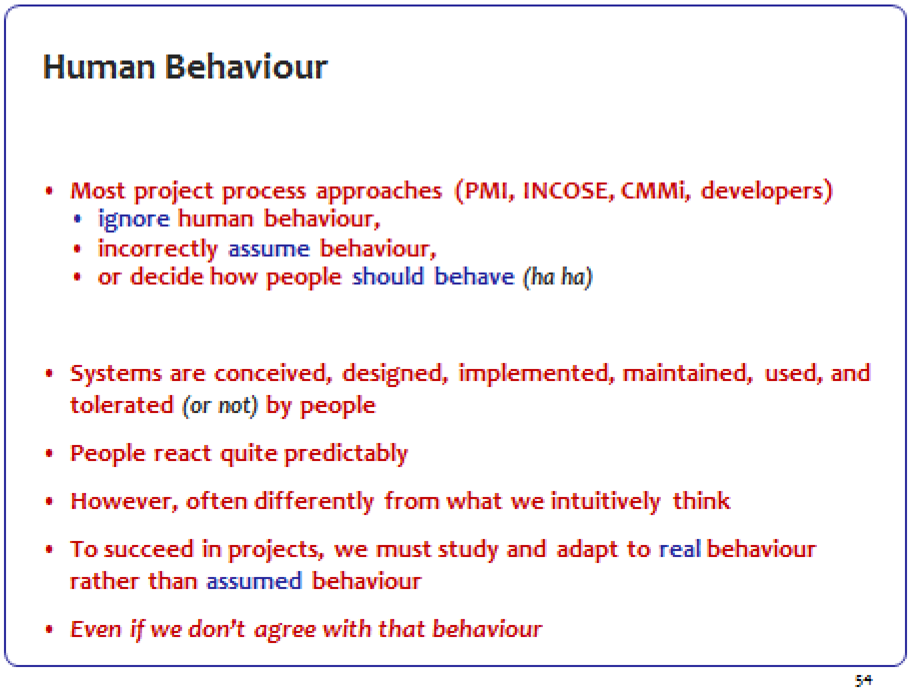
- Getting and keeping the project under control
- Never to be late
- If the quality at the FatalDay isn't right, we're *late*
- If we are late, we *failed*
- No excuses when we're not done at the FatalDay
- Not stealing from our customer's (boss) purse
- The only justifiable cost is the cost of developing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

52



Human Behavior

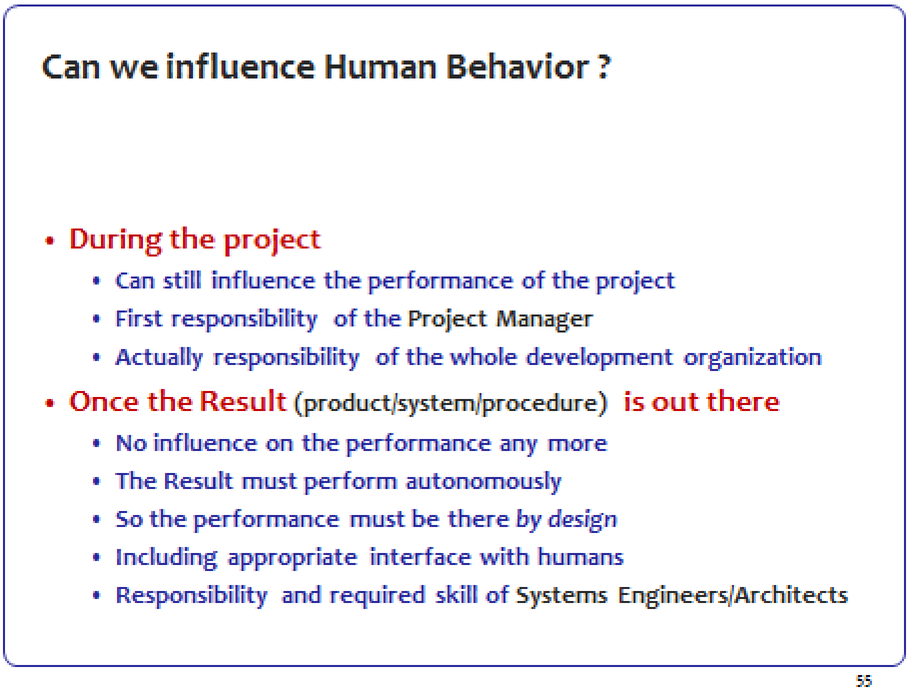
53



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*

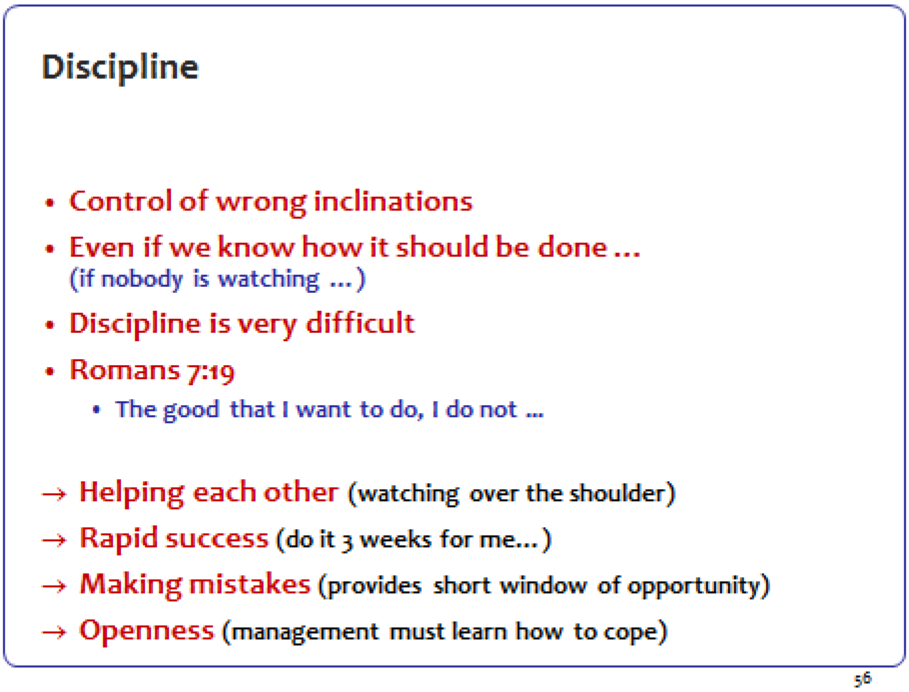
54



Can we influence Human Behavior ?

- During the project
  - Can still influence the performance of the project
  - First responsibility of the Project Manager
  - Actually responsibility of the whole development organization
- Once the Result (product/system/procedure) is out there
  - No influence on the performance any more
  - The Result must perform autonomously
  - So the performance must be there *by design*
  - Including appropriate interface with humans
  - Responsibility and required skill of Systems Engineers/Architects

55



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 ...

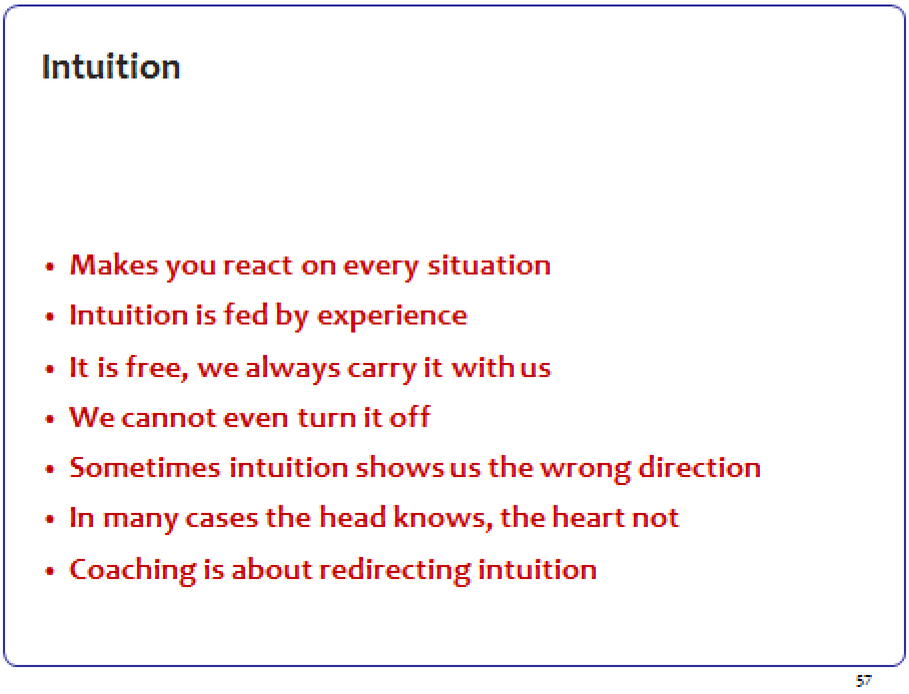
→ Helping each other (watching over the shoulder)

→ Rapid success (do it 3 weeks for me…)

→ Making mistakes (provides short window of opportunity)

→ Openness (management must learn how to cope)

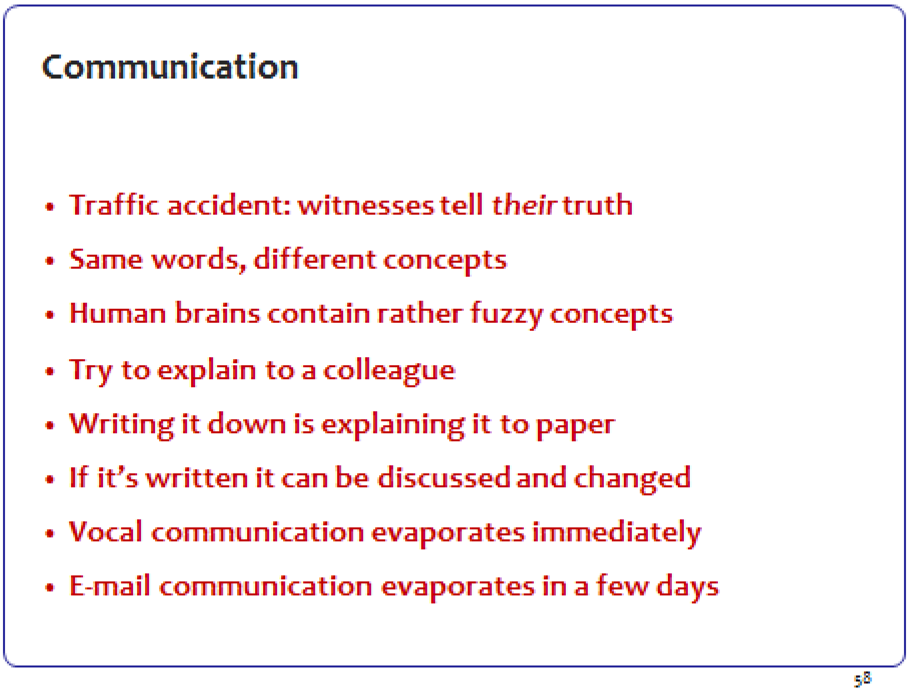
56



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

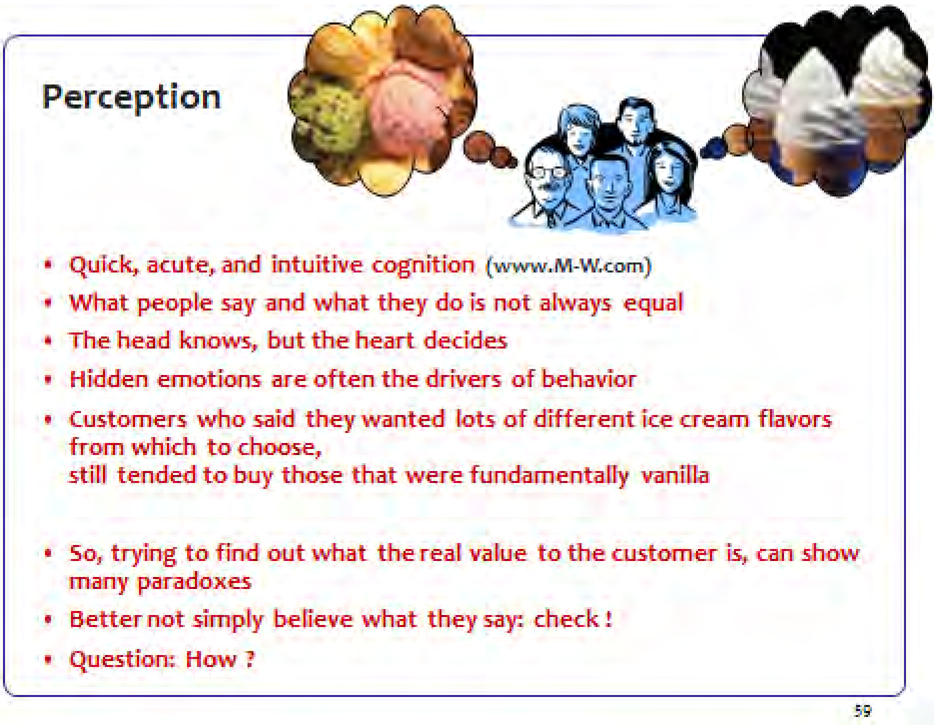
57



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

58



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 !
- Question: How ?

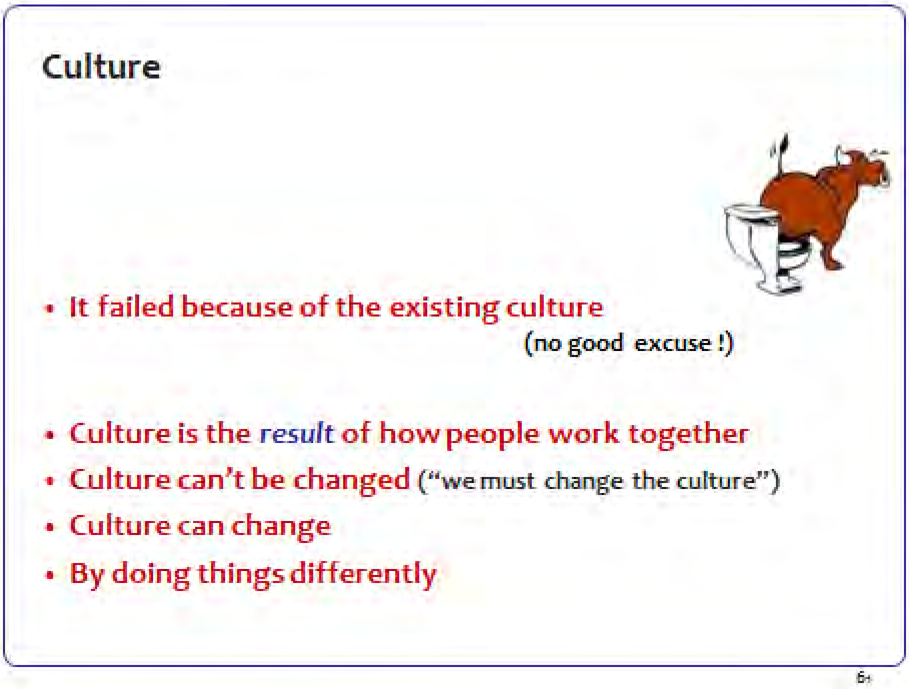
59

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 Onbewuste (Sorry, in Dutch)

60

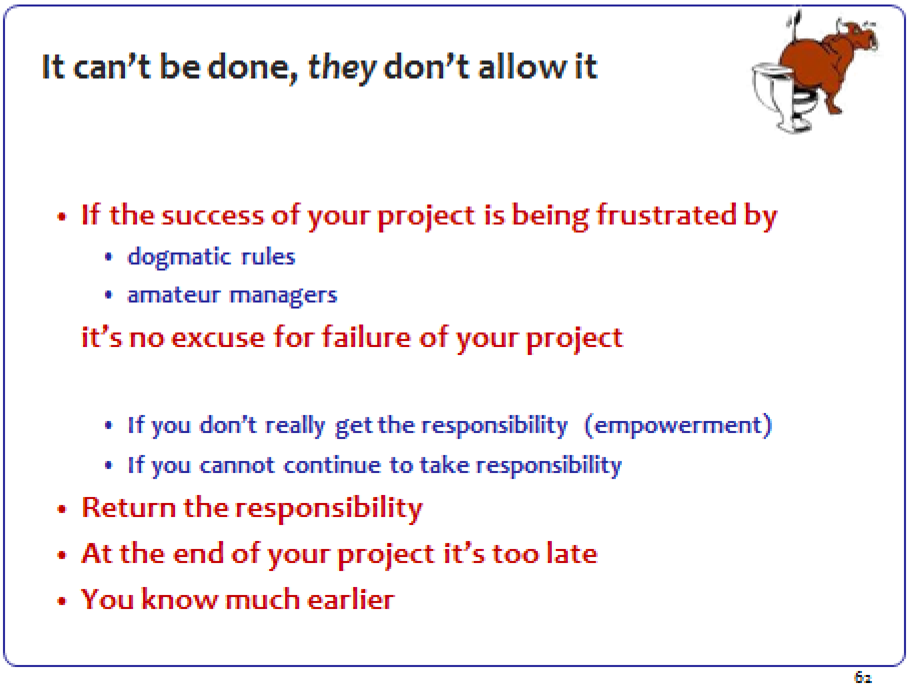


Culture

- It failed because of the existing culture
  (no good excuse !)

- Culture is the *result* of how people work together
- Culture can't be changed ("we must change the culture")
- Culture can change
- By doing things differently

61



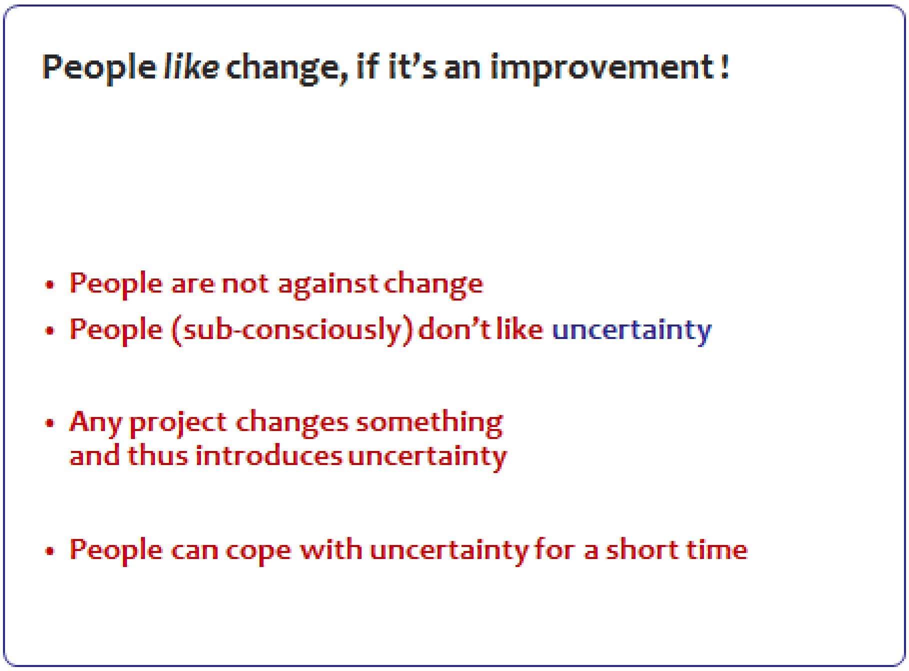
It can't be done, *they* don't allow it

- If the success of your project is being frustrated by
  - dogmatic rules
  - amateur managers

  it's no excuse for failure of your project

  - If you don't really get the responsibility (empowerment)
  - If you cannot continue to take responsibility
- Return the responsibility
- At the end of your project it's too late
- You know much earlier

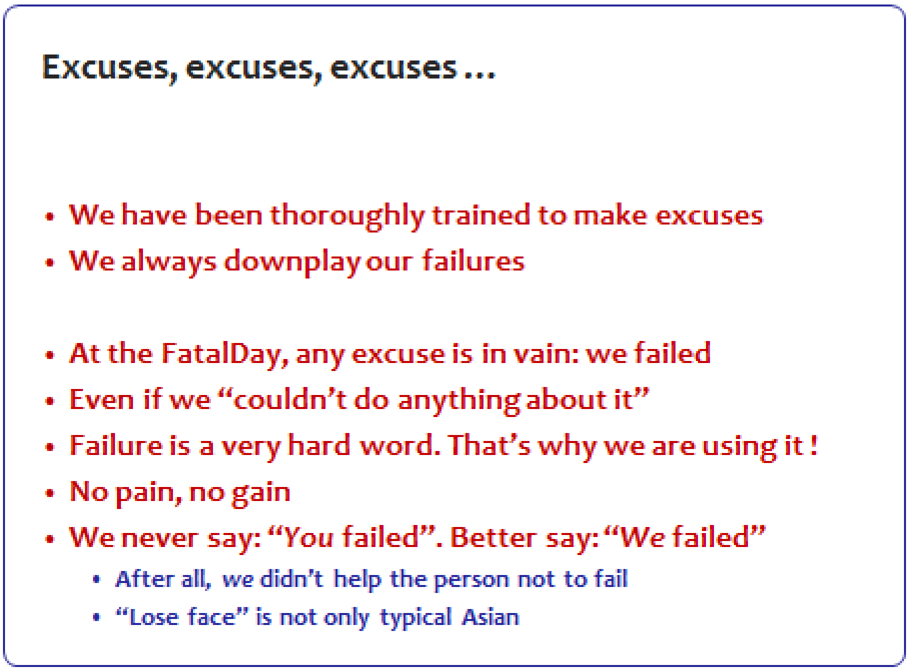
62



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

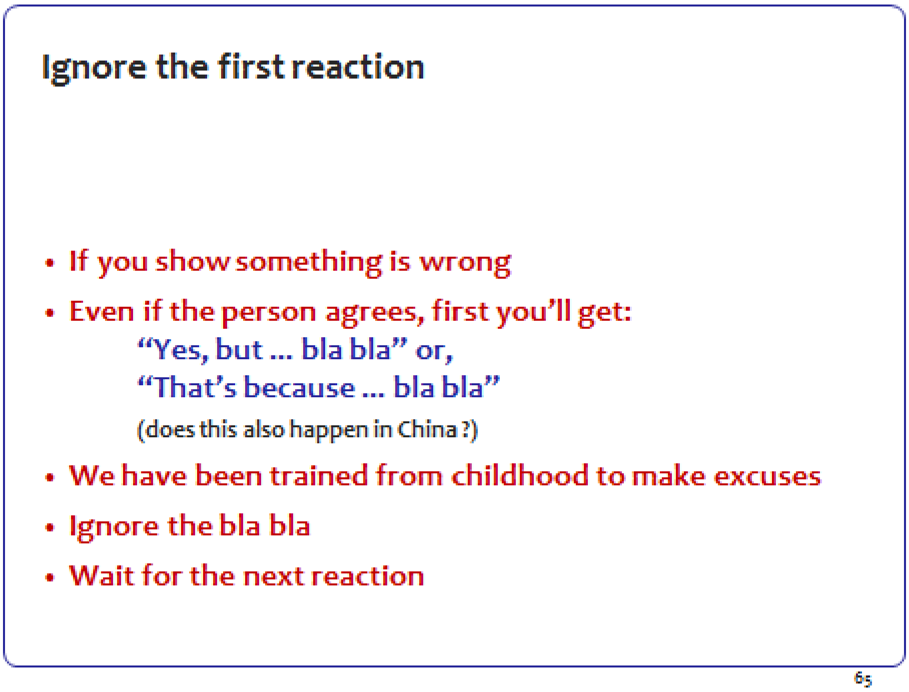
63



Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures
- At the FatalDay, 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 say: "*We* failed"
  - After all, we didn't help the person not to fail
  - "Lose face" is not only typical Asian

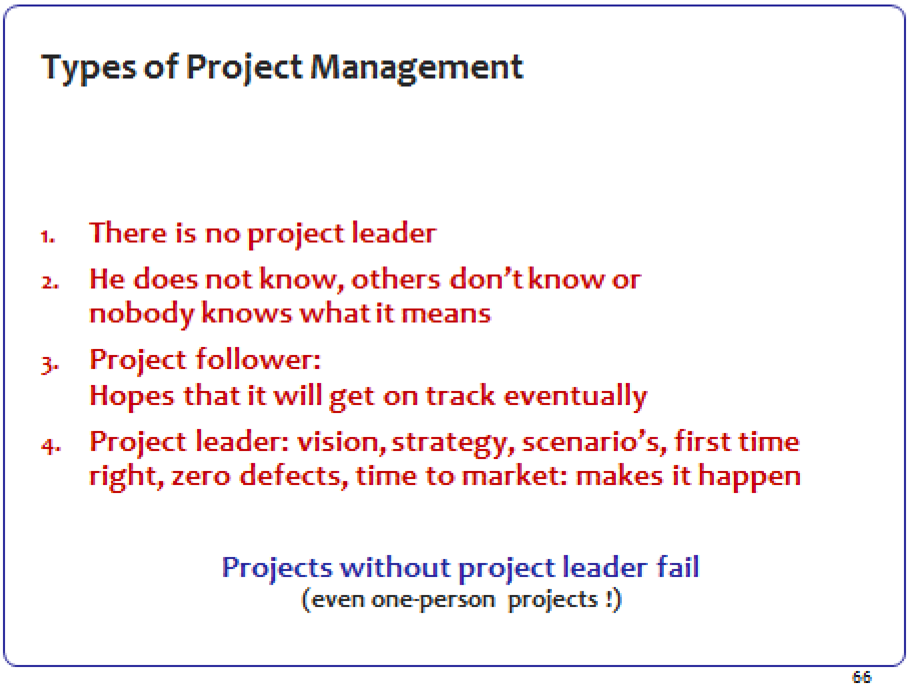
64



Ignore the first reaction

- If you show something is wrong
- Even if the person agrees, first you'll get:
  "Yes, but ... bla bla" or,
  "That's because ... bla bla"
  (does this also happen in China ?)
- We have been trained from childhood to make excuses
- Ignore the bla bla
- Wait for the next reaction

65



Types of Project Management

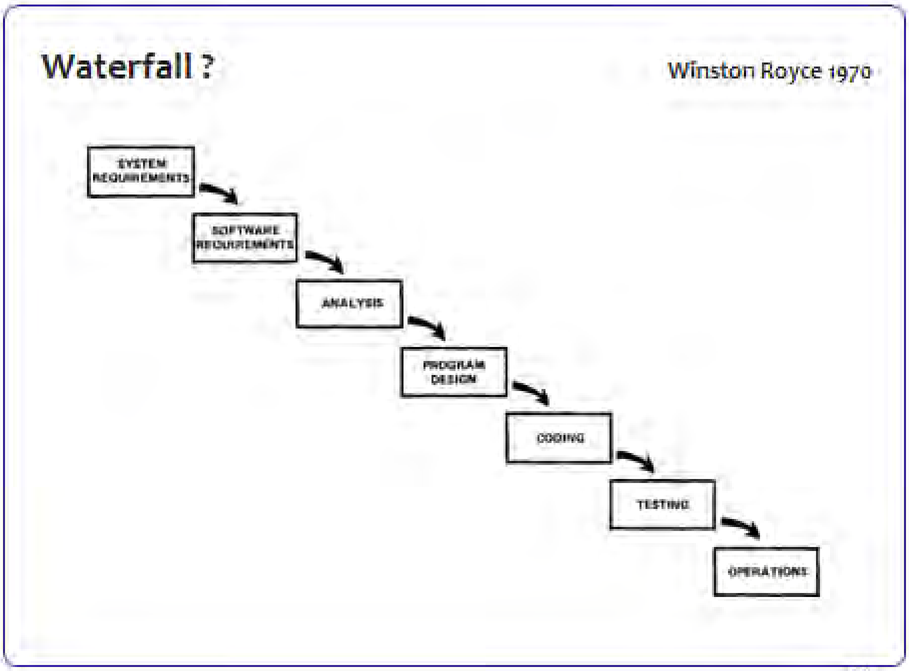
1. There is no project leader
2. He does not know, others don't know or nobody knows what it means
3. Project follower:
   Hopes that it will get on track eventually
4. Project leader: vision, strategy, scenario's, first time right, zero defects, time to market: makes it happen

Projects without project leader fail
(even one-person projects !)

66

Project Life Cycles

67

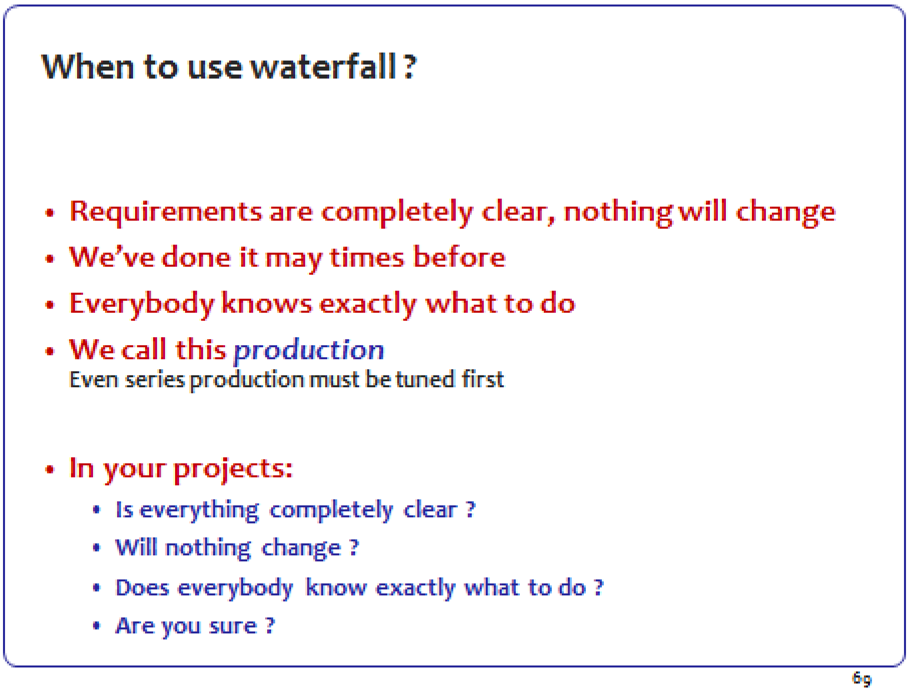


Waterfall ?

Winston Royce 1970

SYSTEM REQUIREMENTS → SOFTWARE REQUIREMENTS → ANALYSIS → PROGRAM DESIGN → CODING → TESTING → OPERATIONS

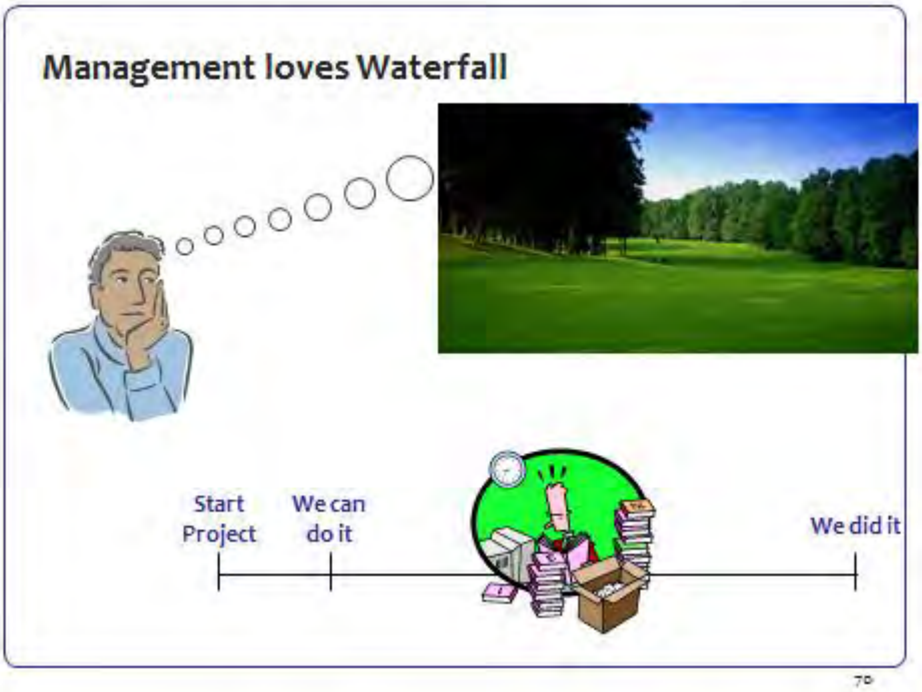
68



When to use waterfall ?

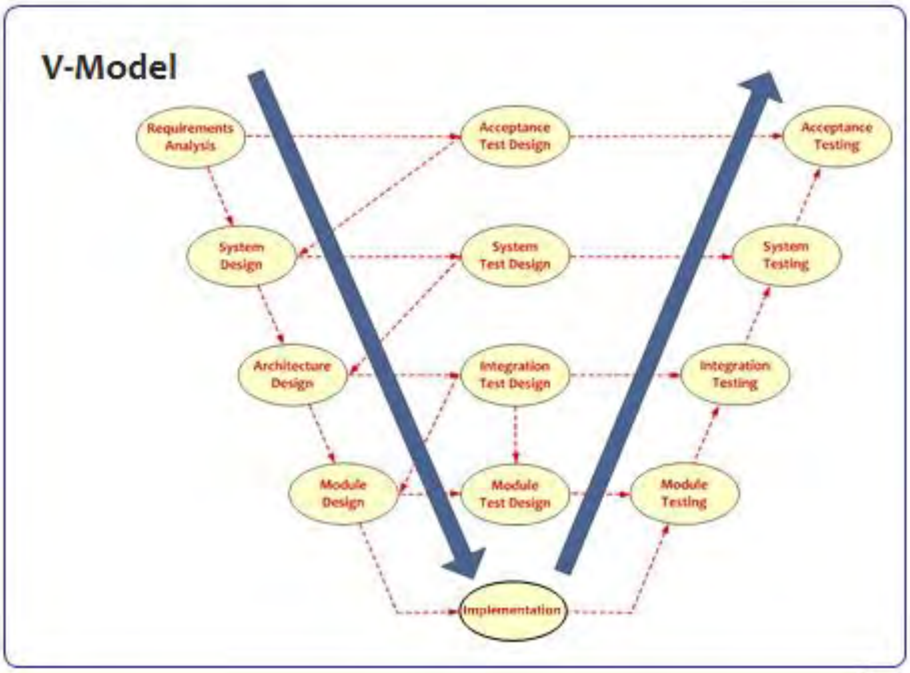
- Requirements are completely clear, nothing will change
- We've done it may times before
- Everybody knows exactly what to do
- We call this *production*
  Even series production must be tuned first
- In your projects:
  - Is everything completely clear ?
  - Will nothing change ?
  - Does everybody know exactly what to do ?
  - Are you sure ?

69



Management loves Waterfall

70



V-Model

Requirements Analysis

System Design

Architecture Design

Module Design

Implementation

Acceptance Test Design

System Test Design

Integration Test Design

Module Test Design

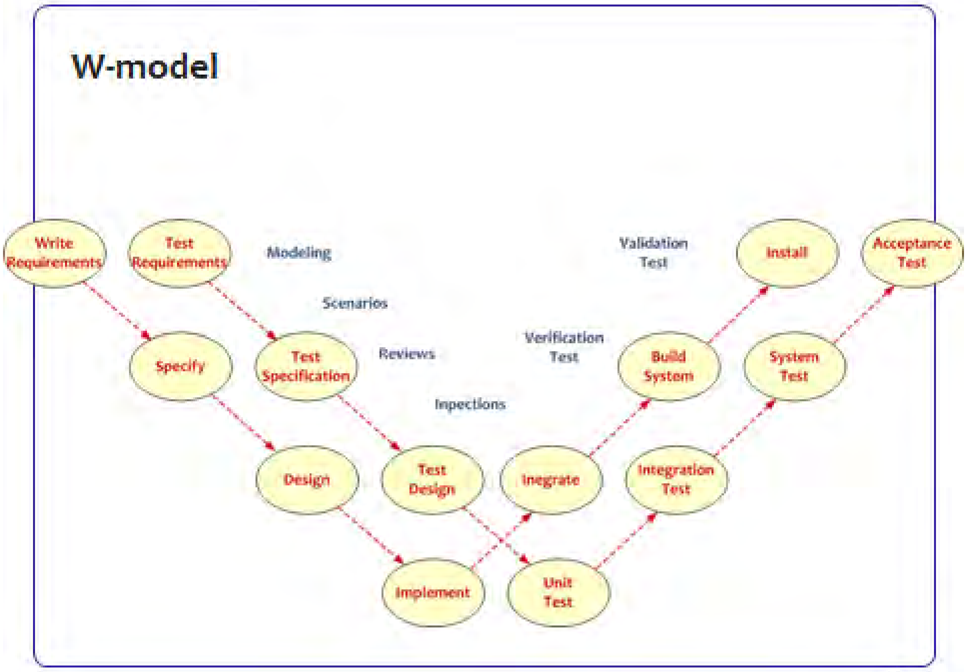
Acceptance Testing

System Testing

Integration Testing

Module Testing

71



W-model

Write Requirements

Test Requirements

Modeling

Scenarios

Specify

Test Specification

Reviews

Inpections

Design

Test Design

Implement

Unit Test

Inegrate

Integration Test

Verification Test

Build System

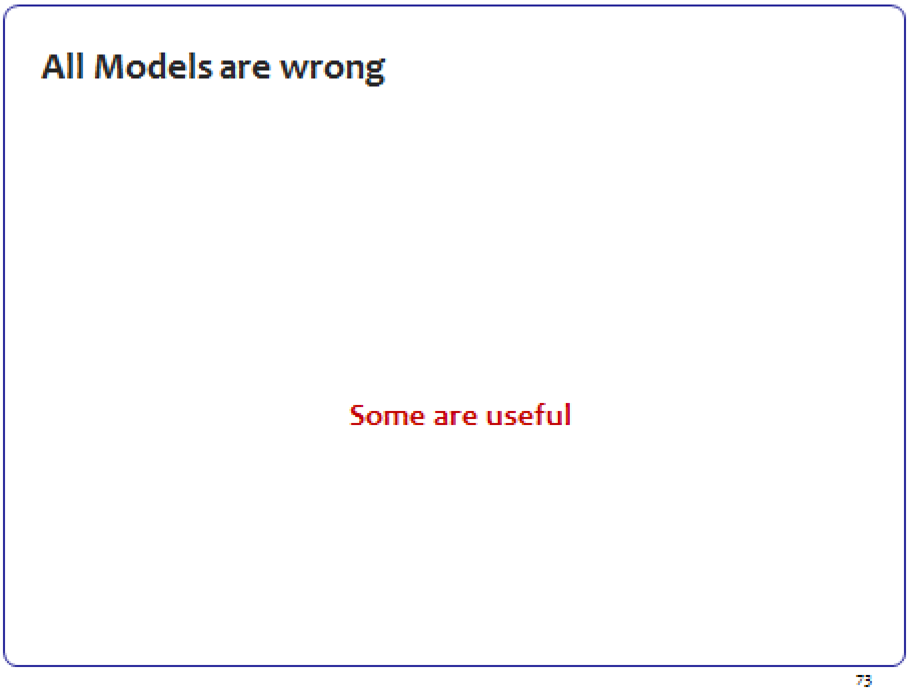
Validation Test

Install

System Test

Acceptance Test

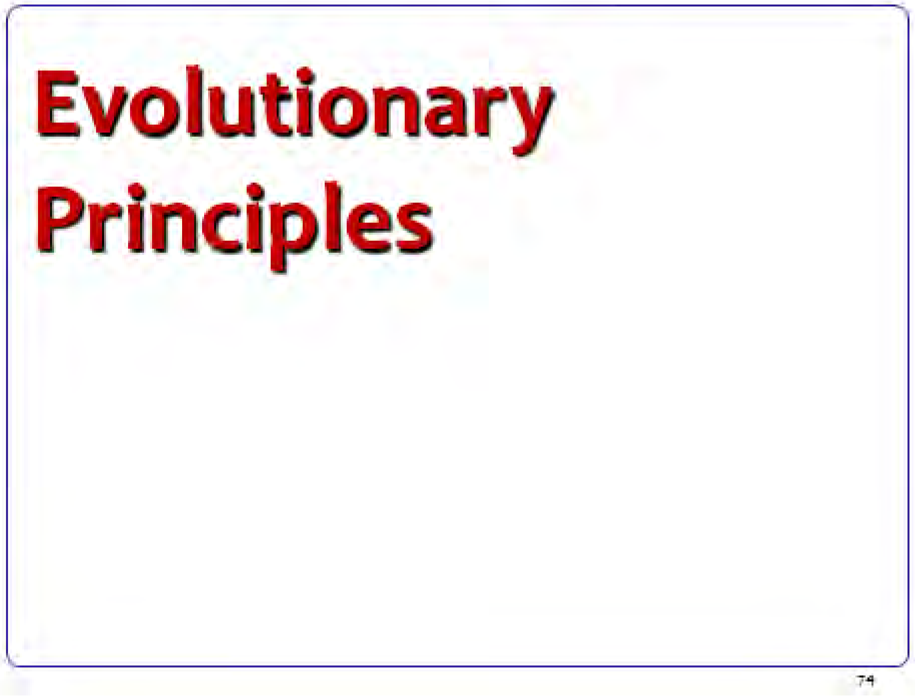
72



All Models are wrong

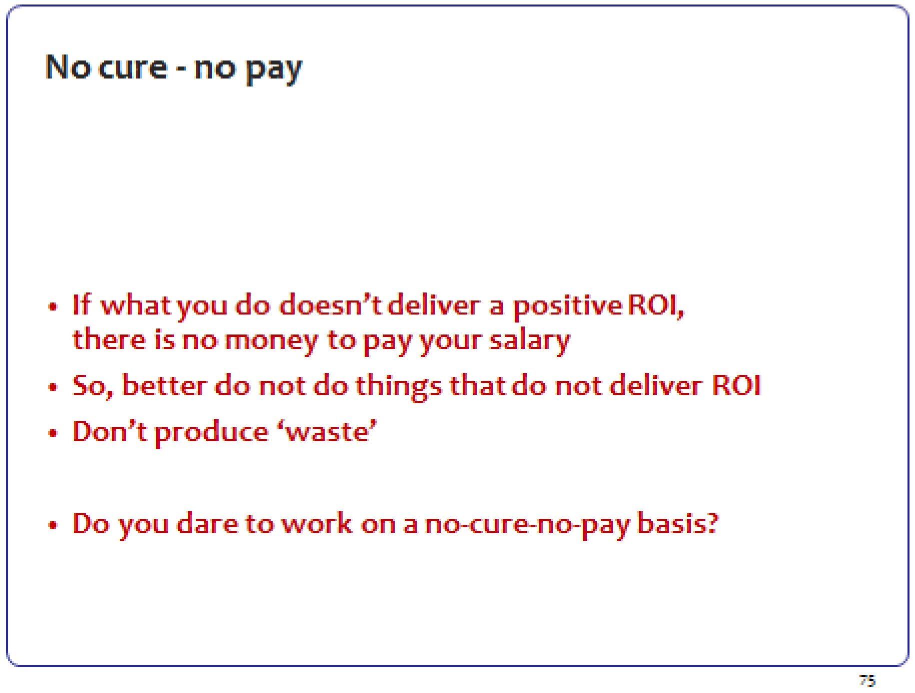
Some are useful

73



Evolutionary Principles

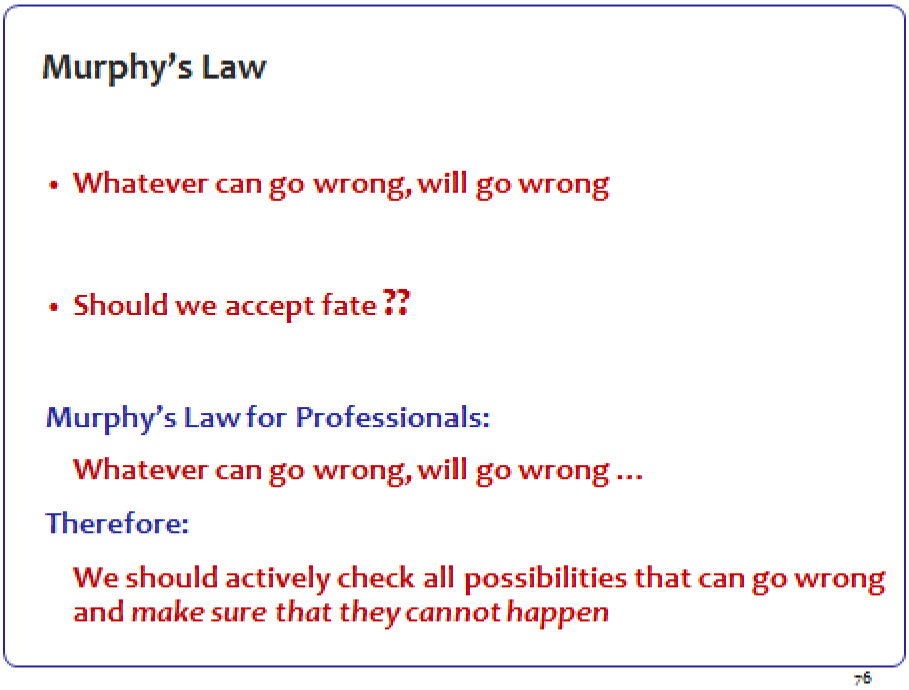
74



No cure - no pay

- If what you do doesn't deliver a positive ROI, there is no money to pay your salary
- So, better do not do things that do not deliver ROI
- Don't produce 'waste'
- Do you dare to work on a no-cure-no-pay basis?

75



Murphy's Law

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

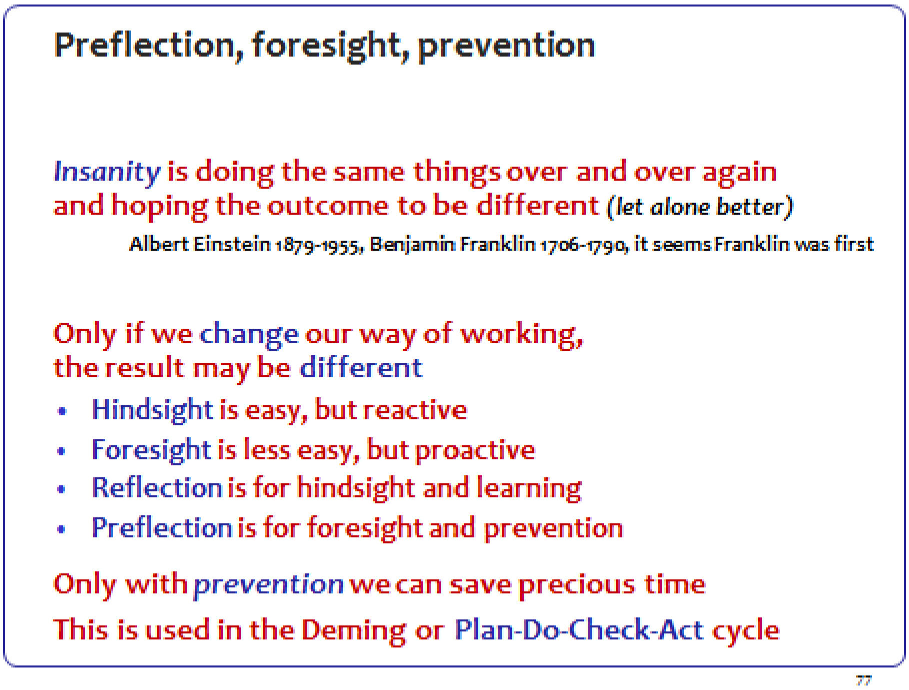
Murphy's Law for Professionals:

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*

76



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

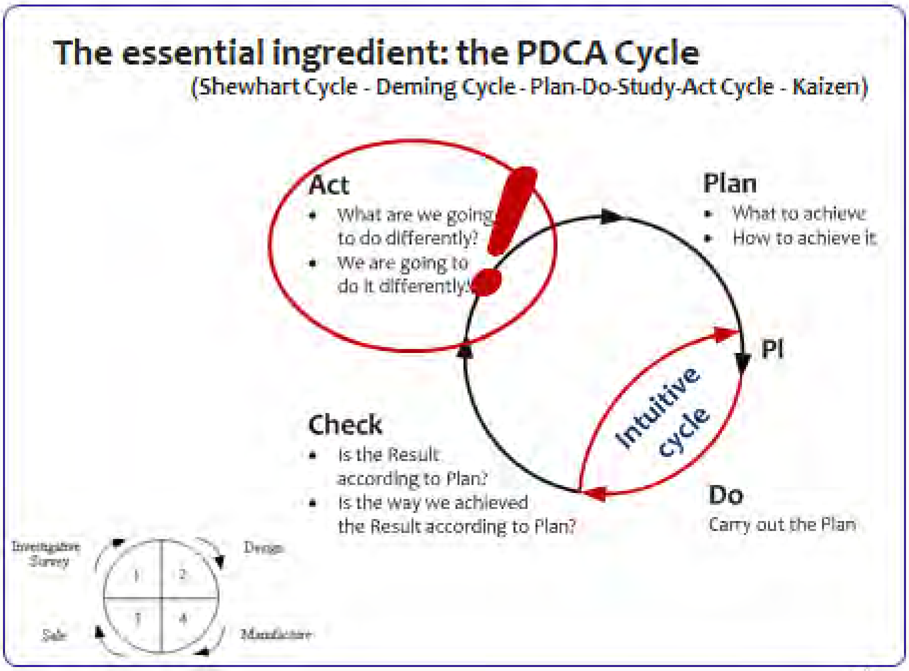
Only if we change our way of working, the result may be different

- 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

77



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

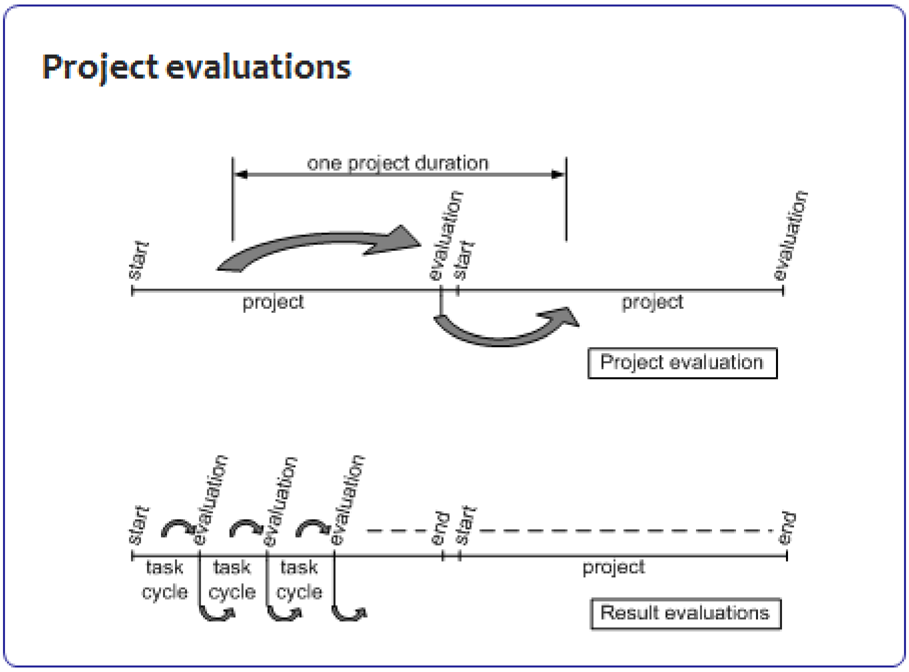
Check

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

Do

Carry out the Plan

78

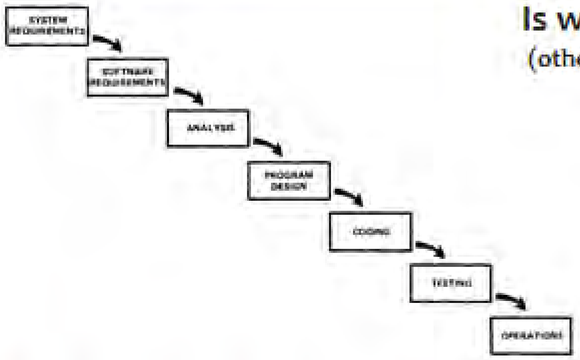


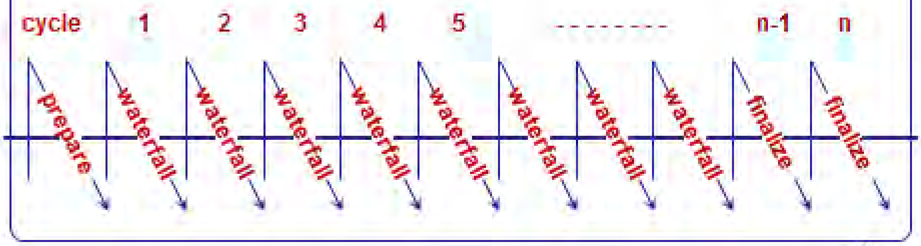
Project evaluations

79

Is waterfall wrong ?

(other than for production)

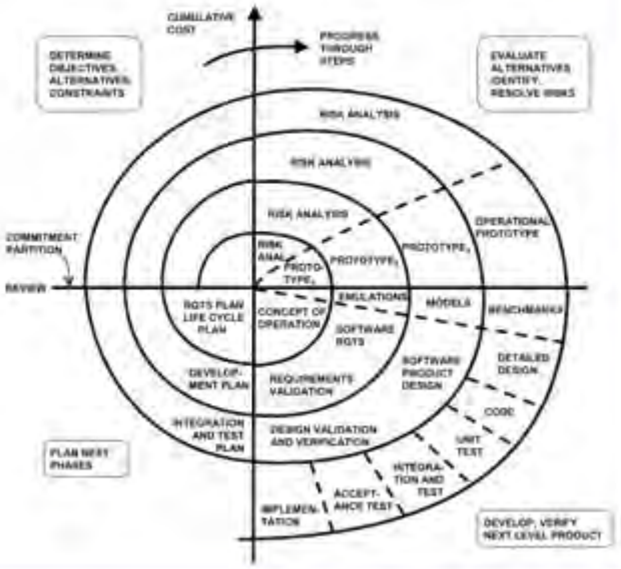




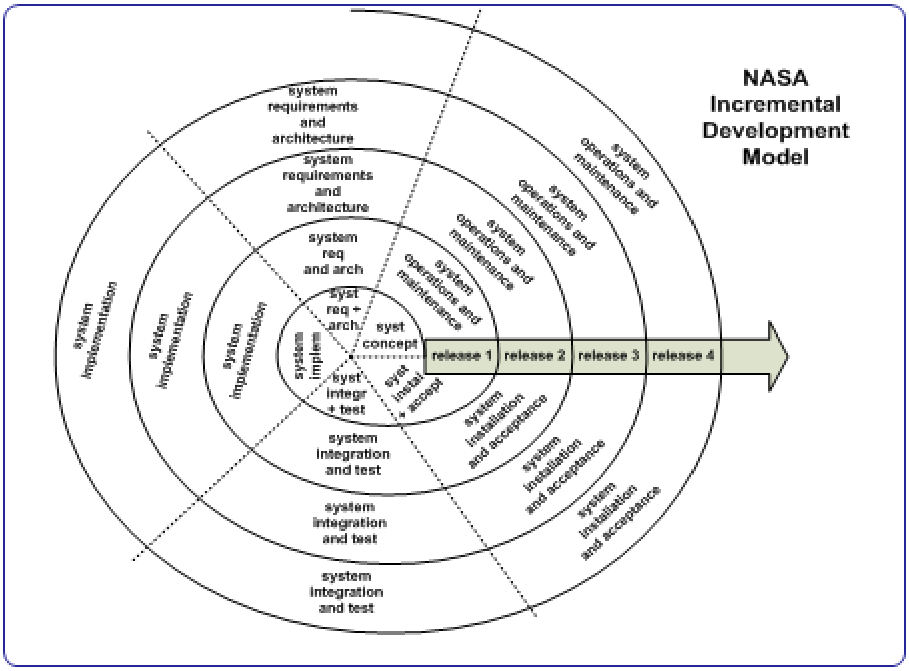
80

Spiral Process model

(Boehm 88)



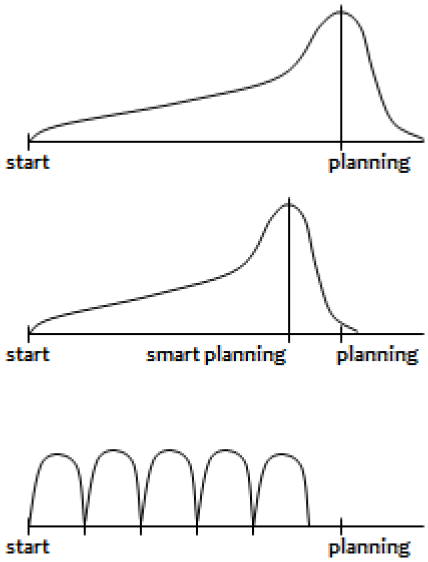
81

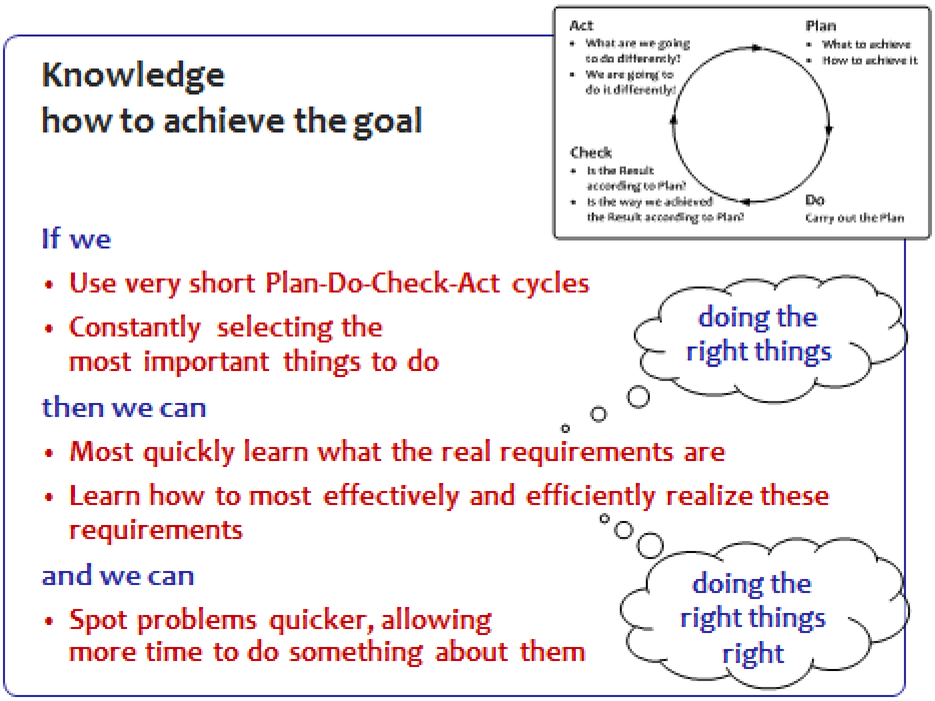


NASA Incremental Development Model

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

Development cycles





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?

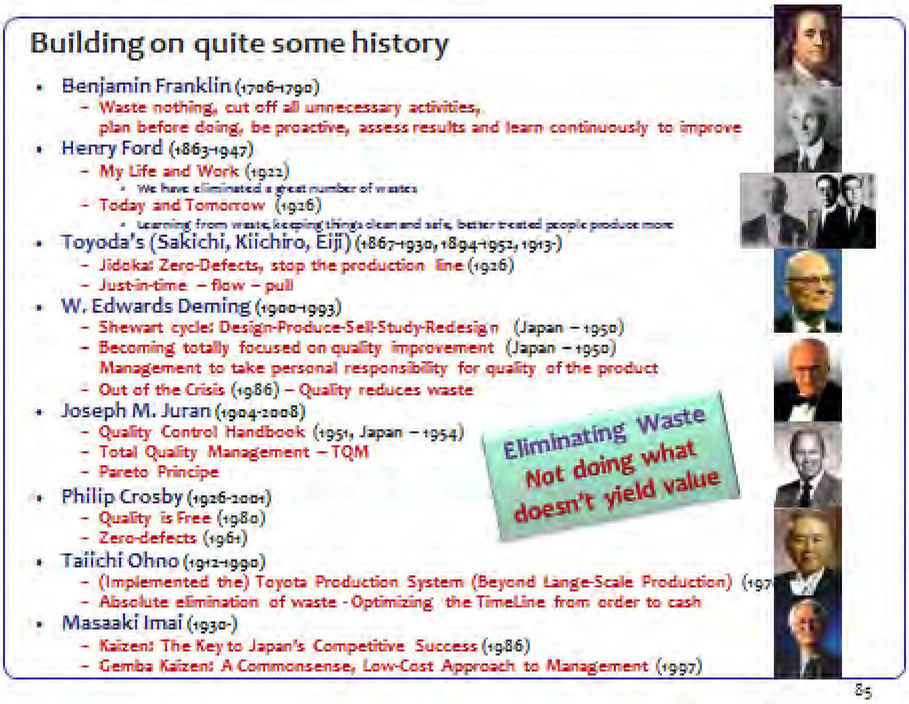
Do
Carry out the Plan

If we
- Use very short Plan-Do-Check-Act cycles
- Constantly selecting the most important things to do

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

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

84

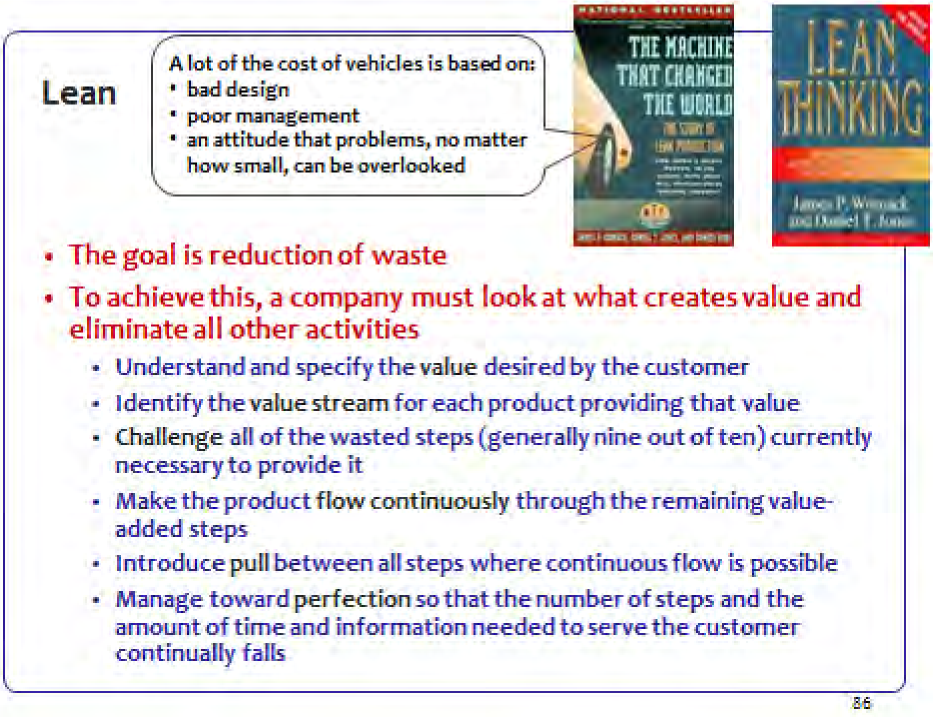


Building on quite some history

- 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) (197
  - 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

85



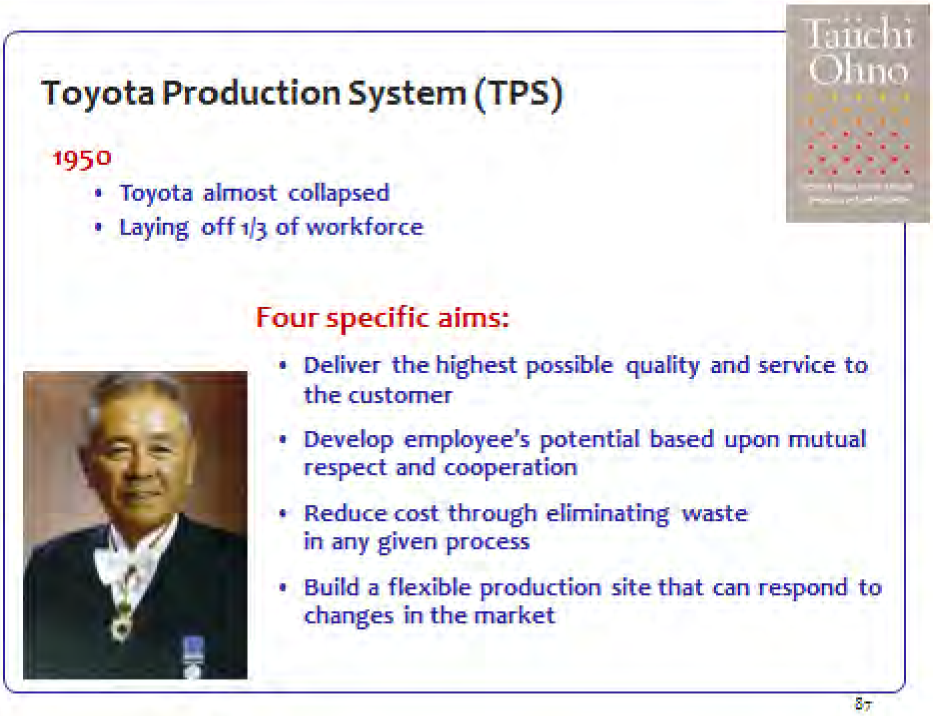
THE MACHINE THAT CHANGED THE WORLD
LEAN THINKING
James P. Womack and Daniel T. Jones

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

86



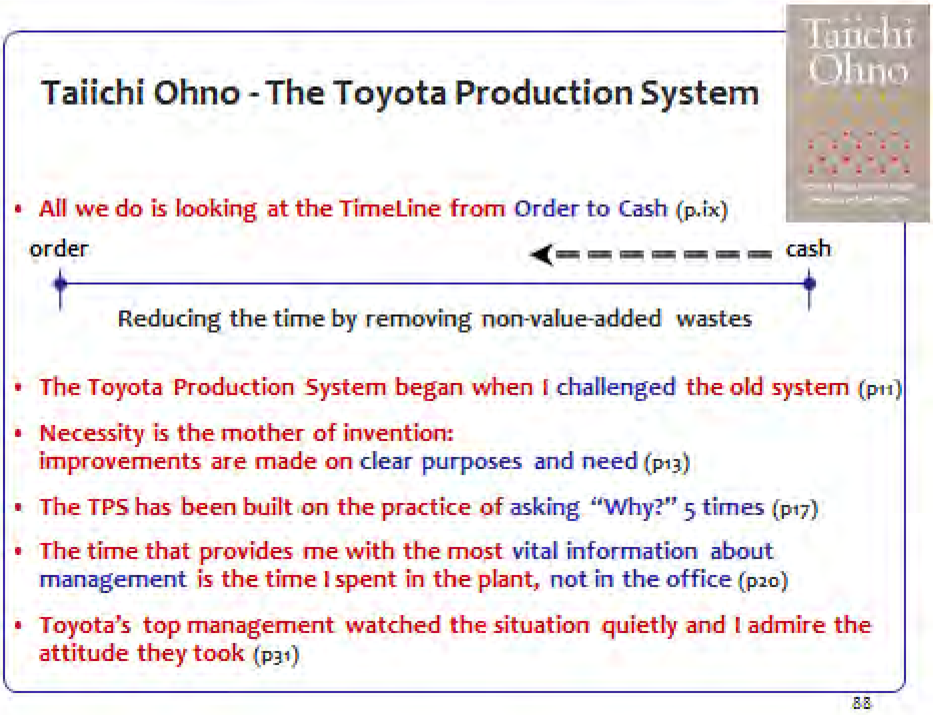
Taiichi Ohno

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

87



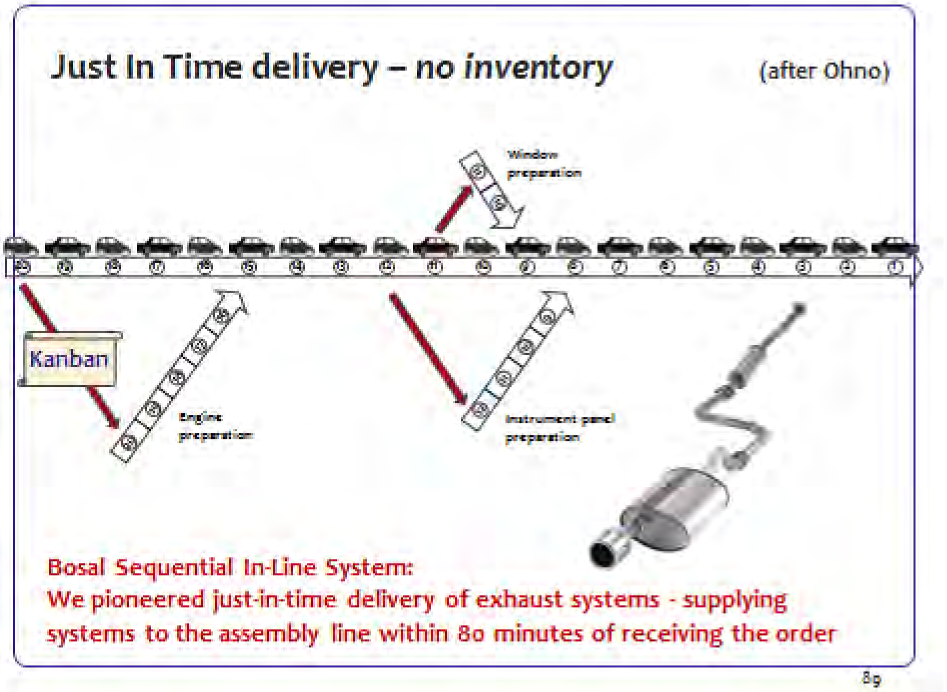
Taiichi Ohno - The Toyota Production System

- 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
Taiichi Ohno

- 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)

88



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

89

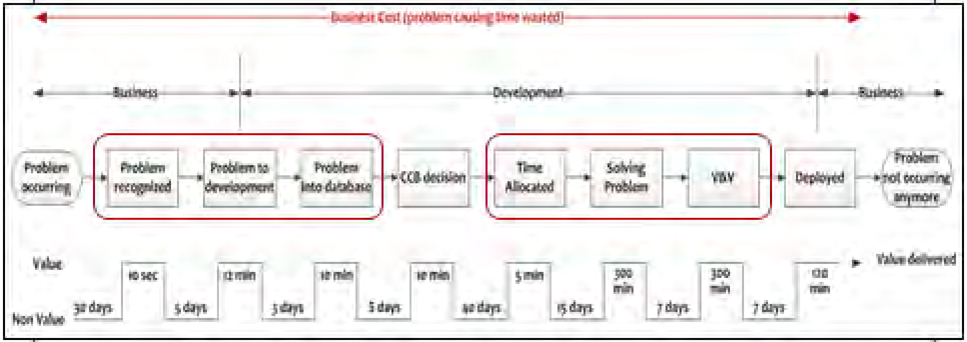
Pillars of the TPS

- 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 (Jidoka 自働化)
  - 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

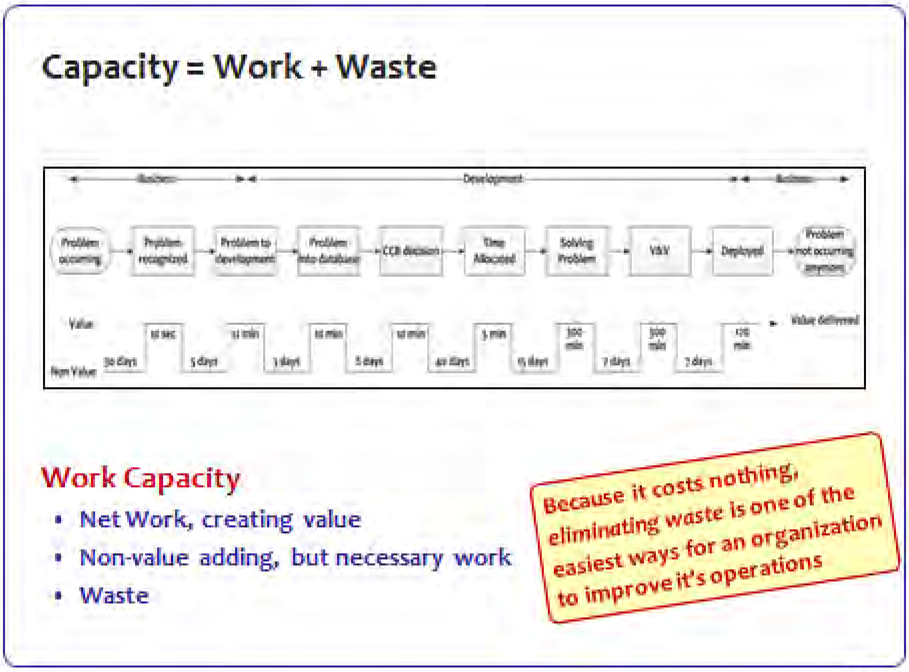
90

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€

91



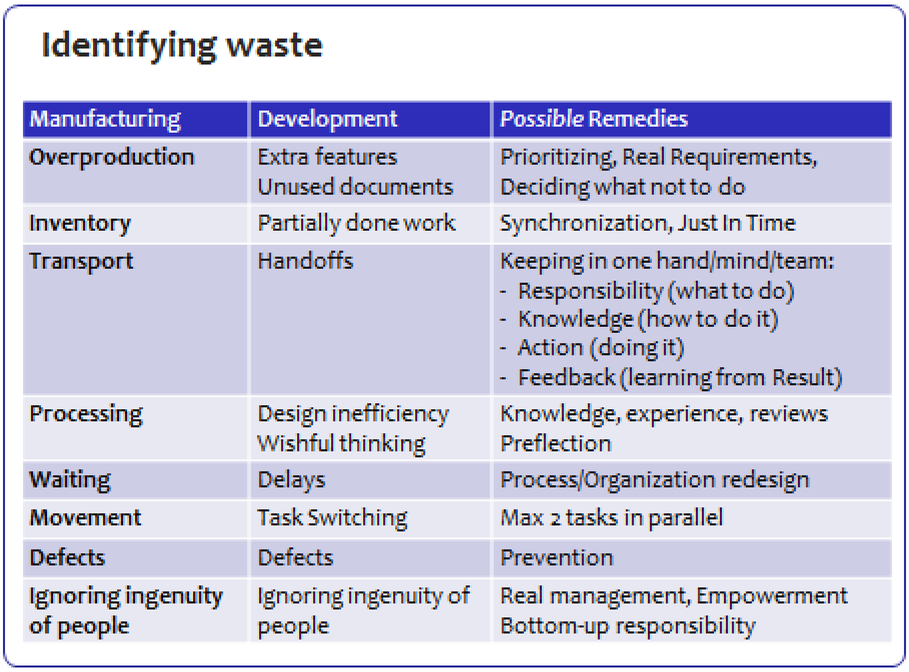
Capacity = Work + Waste

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

92

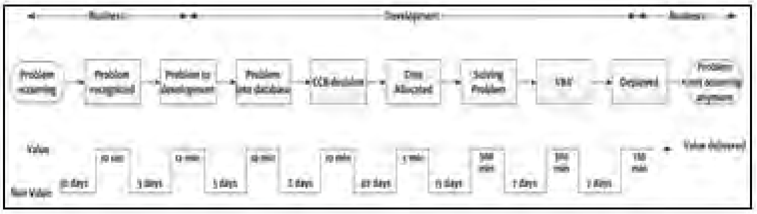


Identifying waste

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/team: - 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

93

5-S



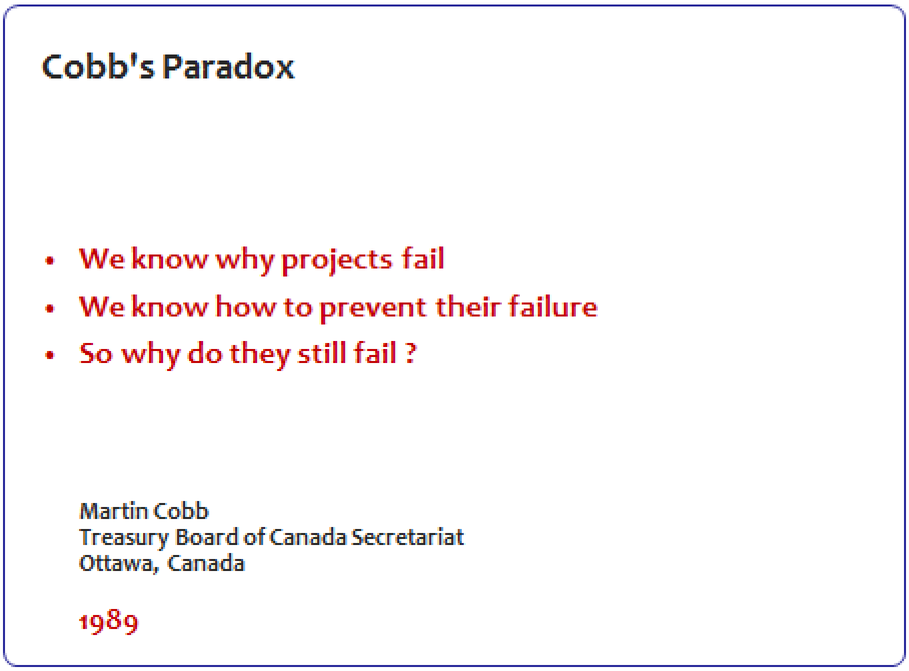
- Seiri 整理 - Remove unnecessary things →waste
- Seiton 整頓 - Arrange remaining things orderly →flow
- Seiso 清掃 - Keep things clean →uncovers hidden problems
- Seiketsu 清潔 - Keep doing it, standardize →know what to improve
- Shitsuke 躾 - Keep training it →fighting entropy

94

The 3 Mu's to remove

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

95



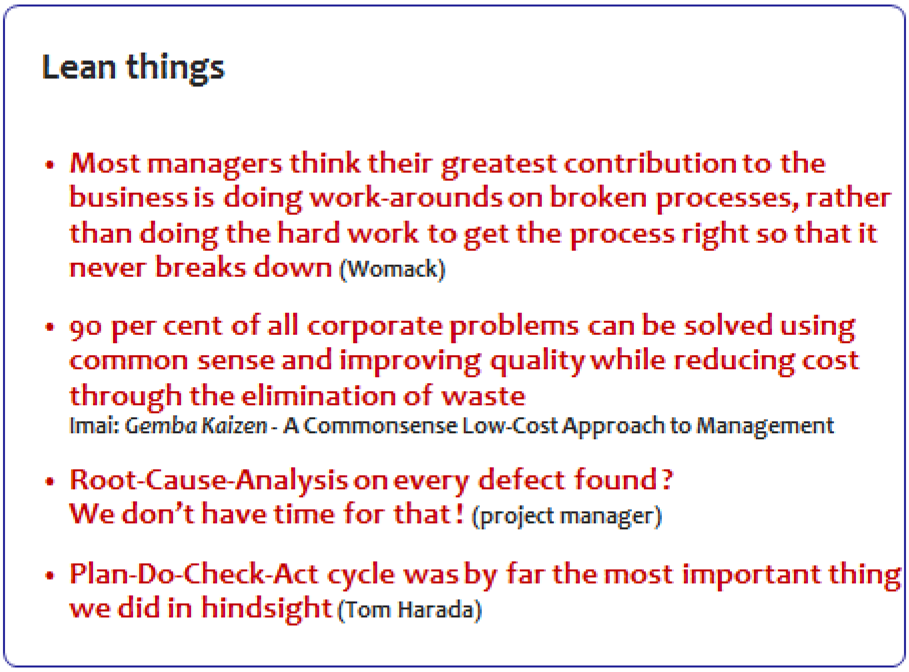
Cobb's Paradox

- 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

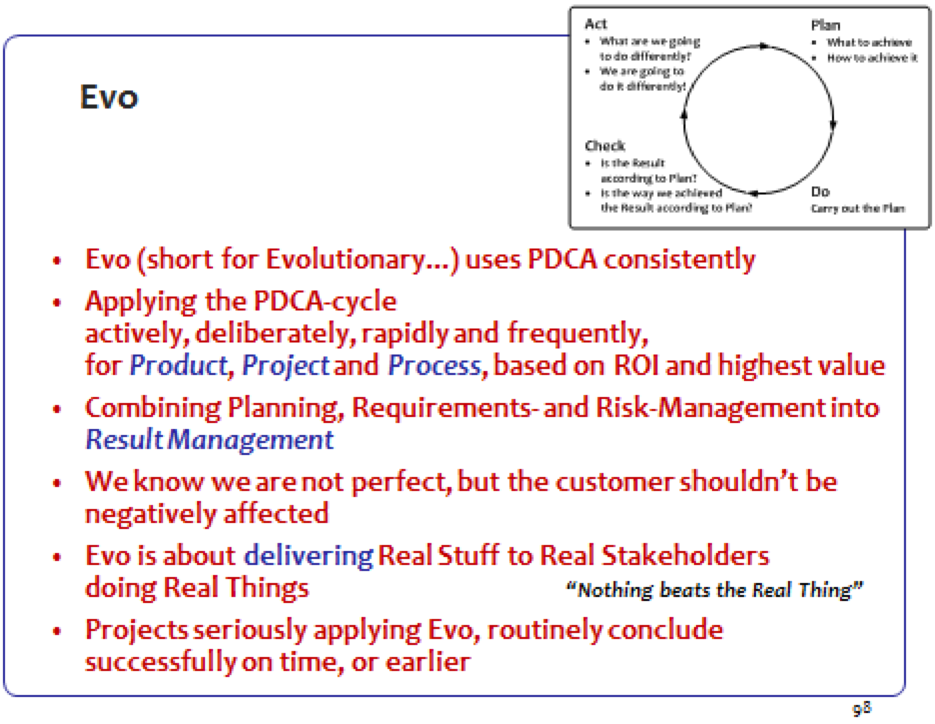
96



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)

97



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?

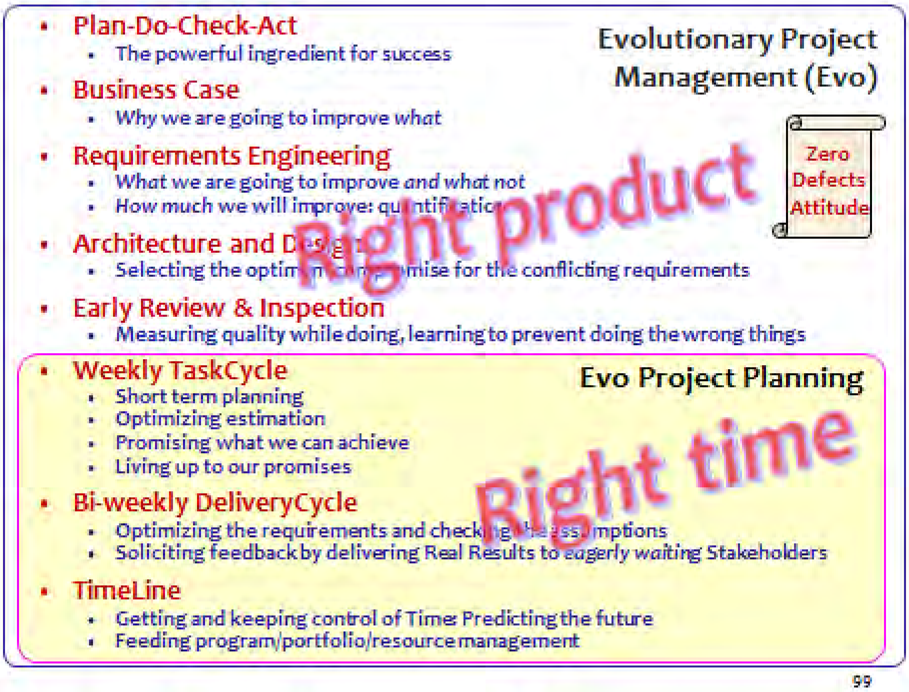
Do
Carry out the Plan

- 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 negatively 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

98



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

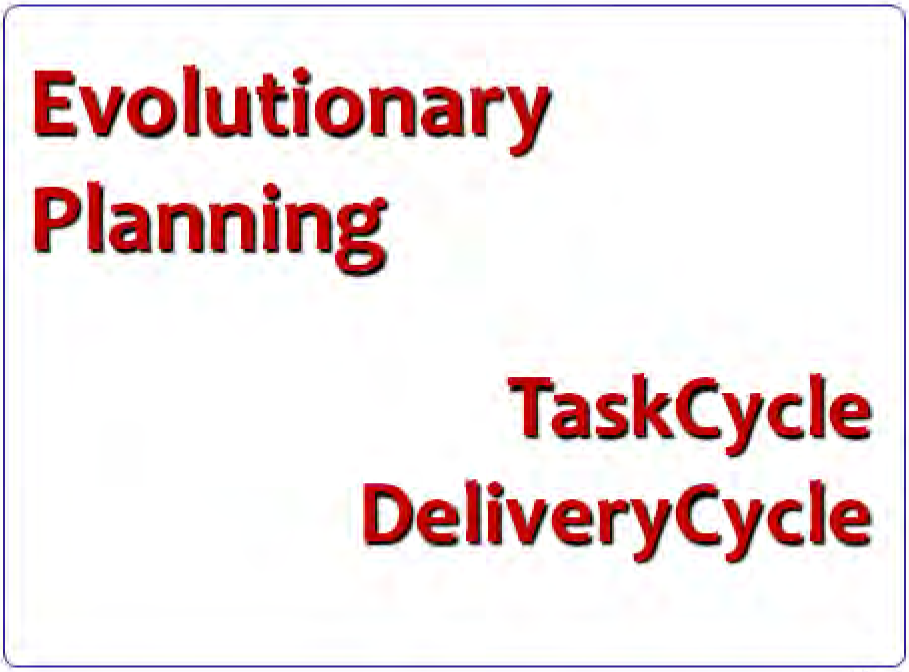
Right product

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

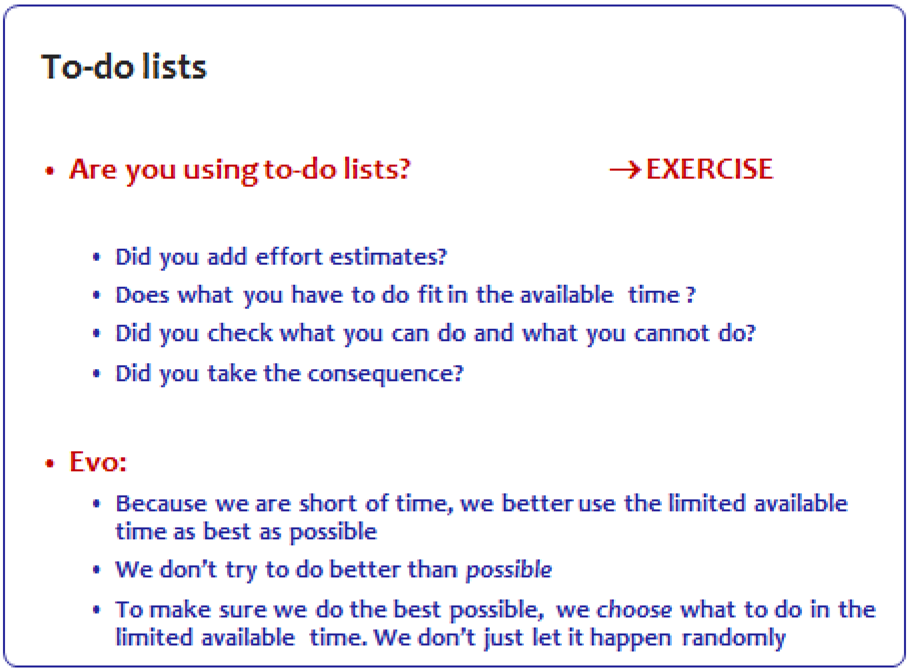
99



Evolutionary Planning

TaskCycle
DeliveryCycle

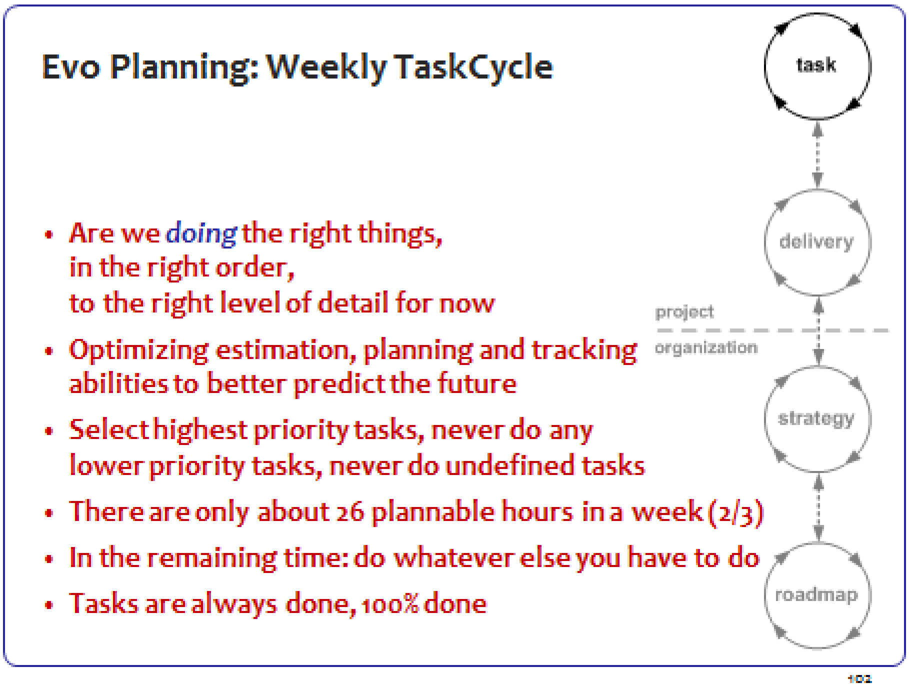
100



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

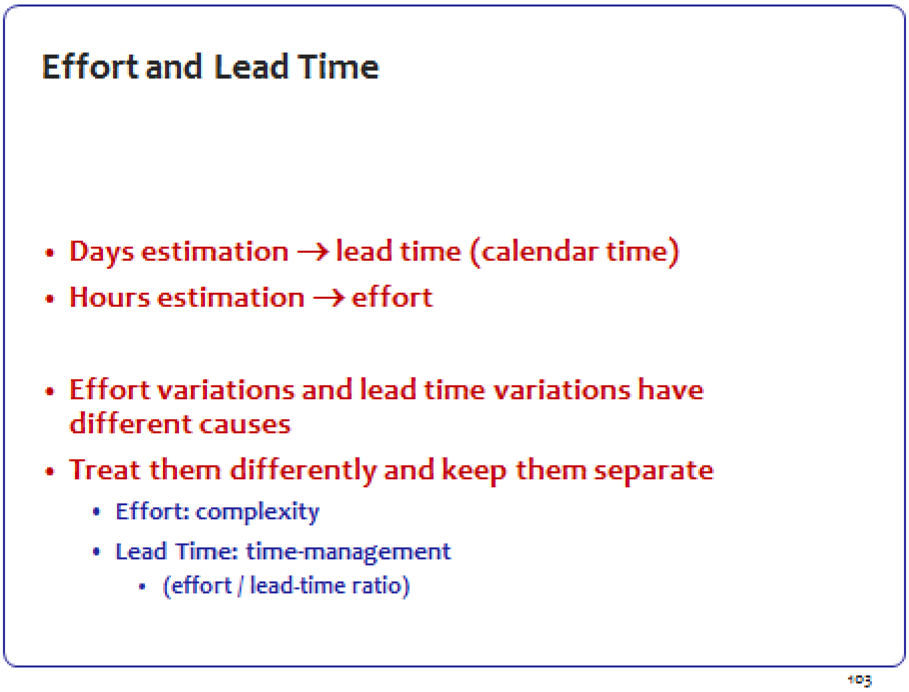
101



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

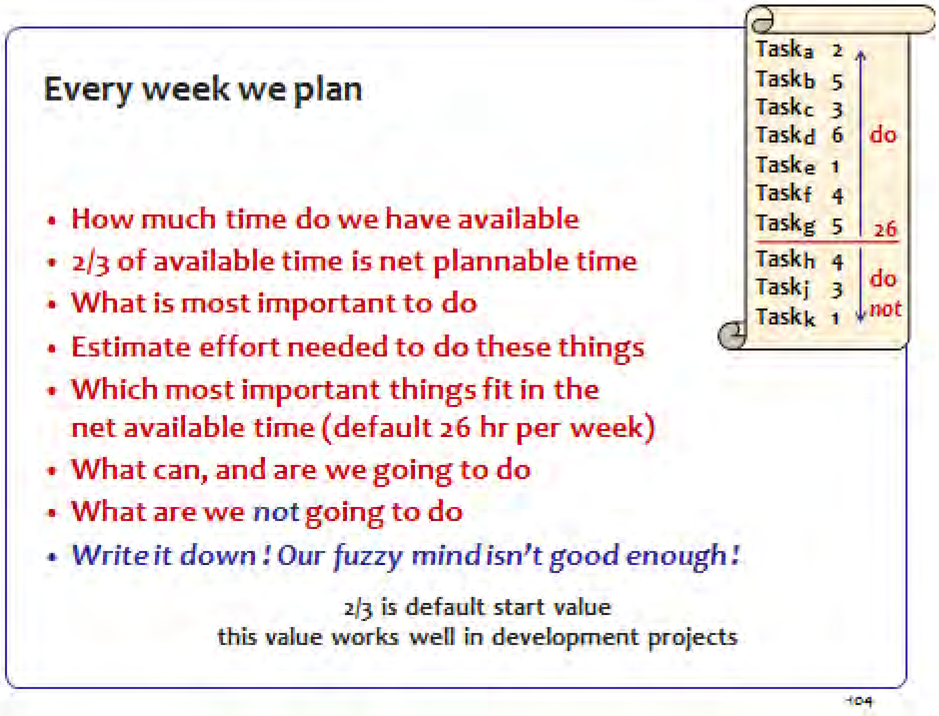
102



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)

103



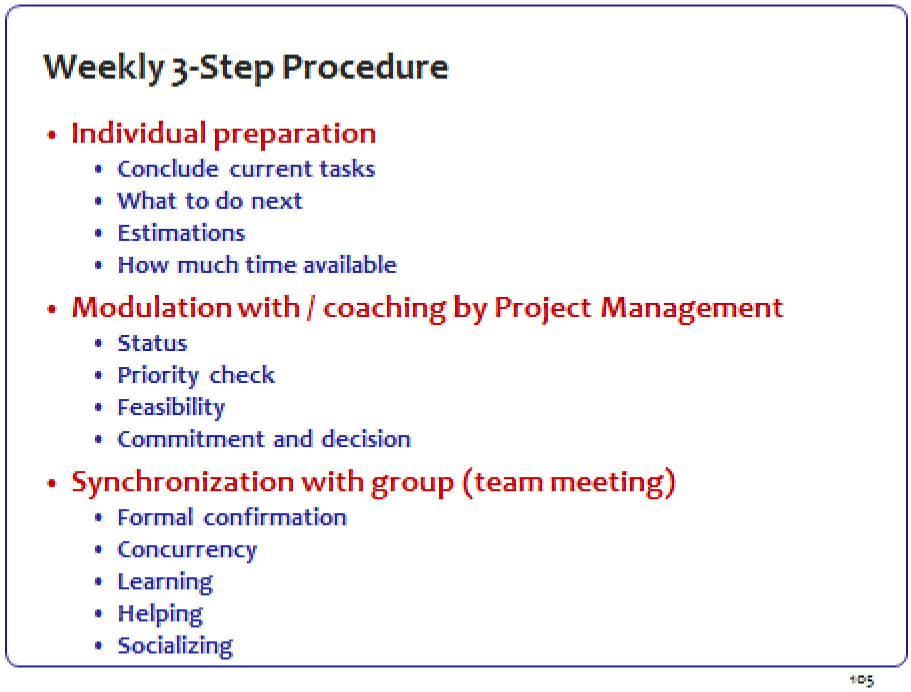
Every week we plan

- 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 !*

Taska	2	
Taskb	5	
Taskc	3	
Taskd	6	do
Taske	1	
Taskf	4	
Taskg	5	26
Taskh	4	
Taskj	3	do
Taskk	1	not

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

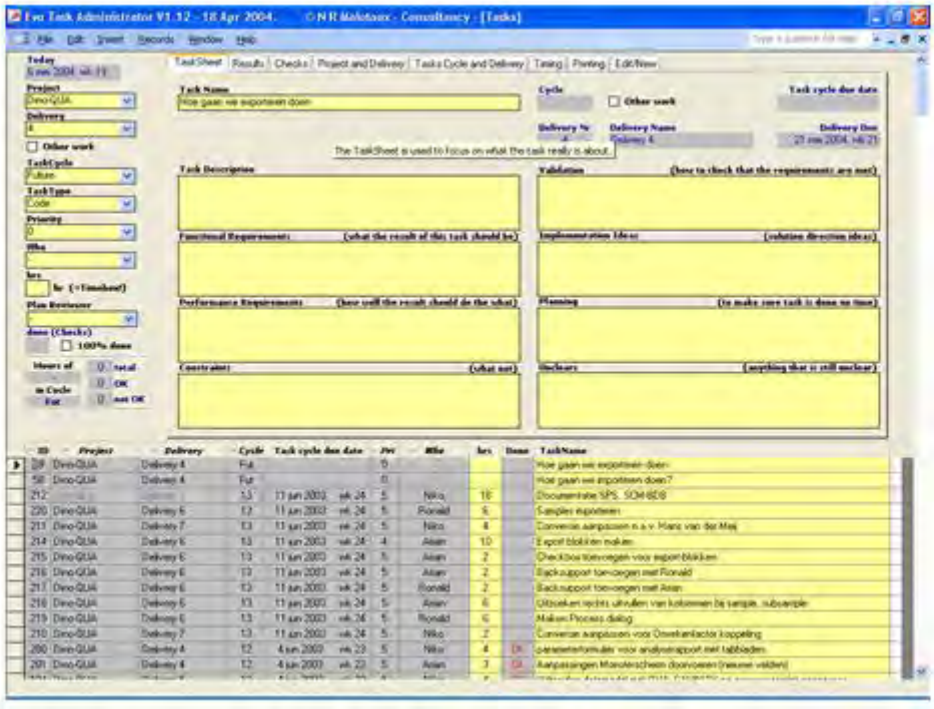
104



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

105



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

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: 0 total, 0 OK, 0 not OK
in Cycle: Fut

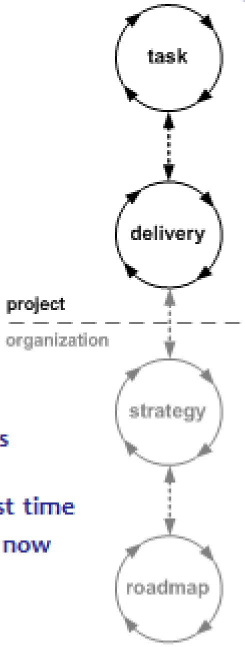
Task Name: Hoe gaan we exporteren doen
Cycle: □ Other work
Task cycle due date:
Delivery №: 4
Delivery Name: Delivery 4
Delivery Due: 21 mei 2004 wk 21

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)
Unclears (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 importeren doen?
212			13	11 jun 2003	wk 24	5	Niko	16		Documentatie SPS SCM-BDB
220	Dino-QUA	Delivery 6	12	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	Arian	10		Export blokken maken
215	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	2		Checkbox toevoegen voor export blokken
216	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	2		Backsupport toevoegen met Ronald
217	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Ronald	2		Backsupport toevoegen met Arian
218	Dino-QUA	Delivery 6	13	11 jun 2003	wk 24	5	Arian	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	4	OK	parameterformules voor analyserapport met tabbladen
201	Dino-QUA	Delivery 4	12	4 jun 2003	wk 23	5	Arian	3	OK	Aanpassingen Monsterscherm doorvoeren (nieuwe velden)

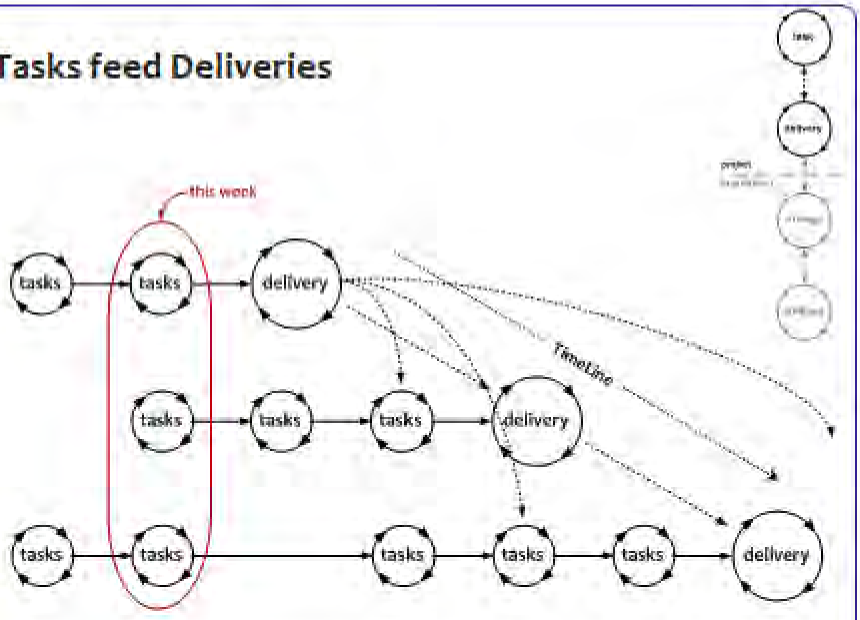
DeliveryCycle



- 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 best 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

107

Tasks feed Deliveries



108

Task Cycle ↔ Delivery Cycle

Doing Delivering
the *right things*, in the *right order* to the *right level of detail*

Optimizing
Estimation, planning, tracking Requirements, assumptions

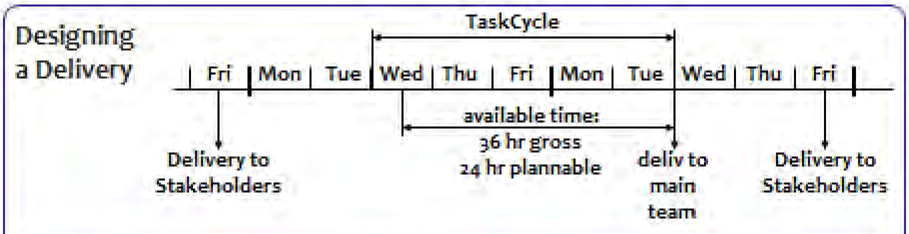
Selecting
Highest priority tasks Most important values

≤ 1 week ≤ 2 weeks

Always done, 100% done

109

Designing a Delivery



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

Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	42

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

Optimizing Value Stream by Prevention

Jerome	
XMLa	3
XMLb	3
...	

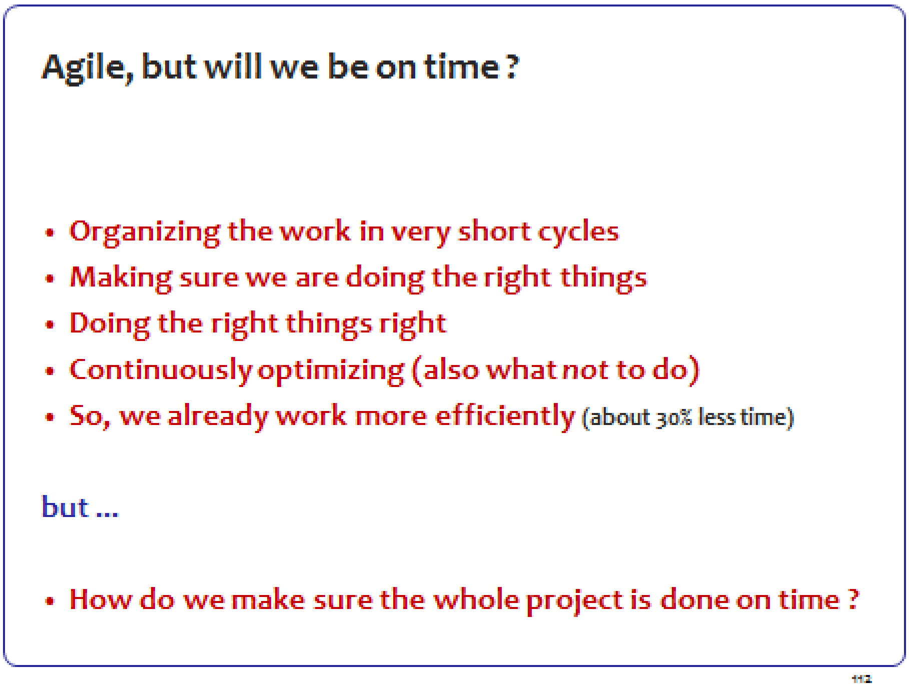
110

TaskCycle Exercise

Task a	2	
Task b	5	
Task c	3	
Task d	6	do
Task e	1	
Task f	4	
Task g	5	26
Task h	4	do
Task j	3	
Task k	1	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 hr)
- What can you do, and what are you going to do
- What are you *not* going to do
- Why ?

111



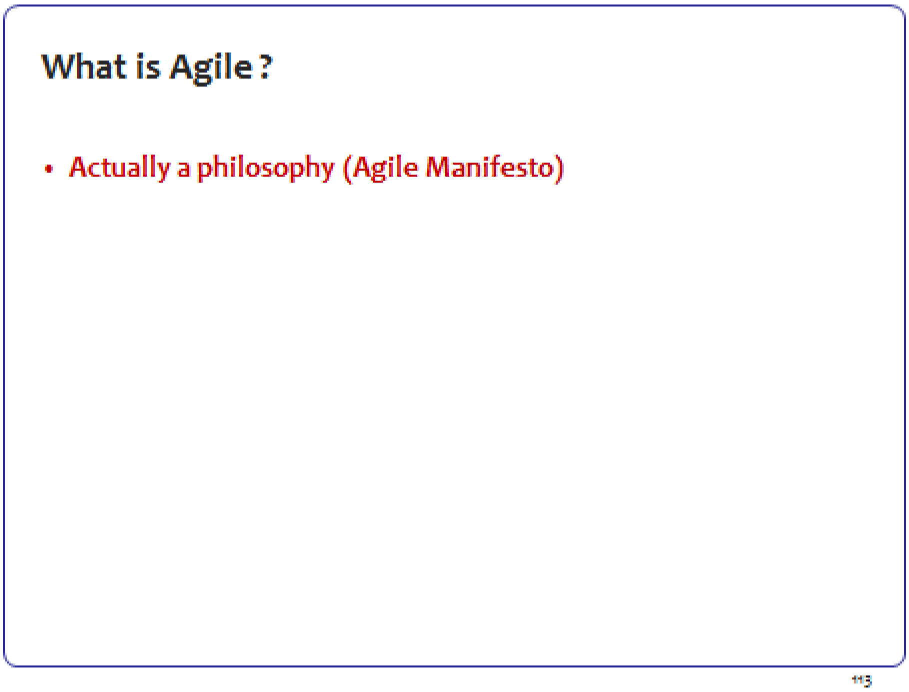
Agile, but will we be on time ?

- Organizing the work in very short cycles
- Making sure we are doing the right things
- Doing the right things right
- Continuously optimizing (also what not to do)
- So, we already work more efficiently (about 30% less time)

but ...

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

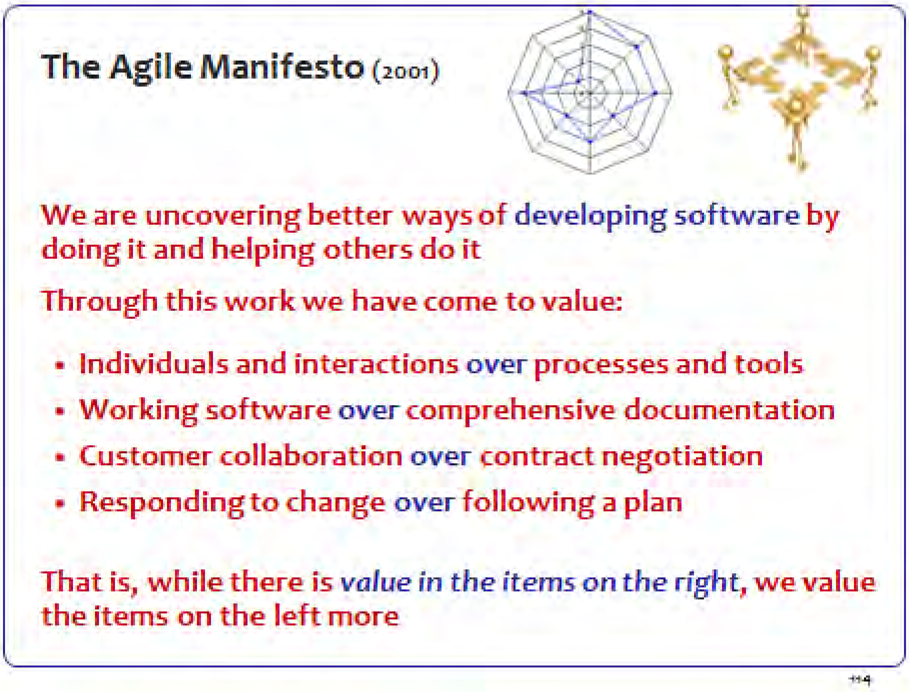
112



What is Agile ?

- Actually a philosophy (Agile Manifesto)

113



The Agile Manifesto (2001)

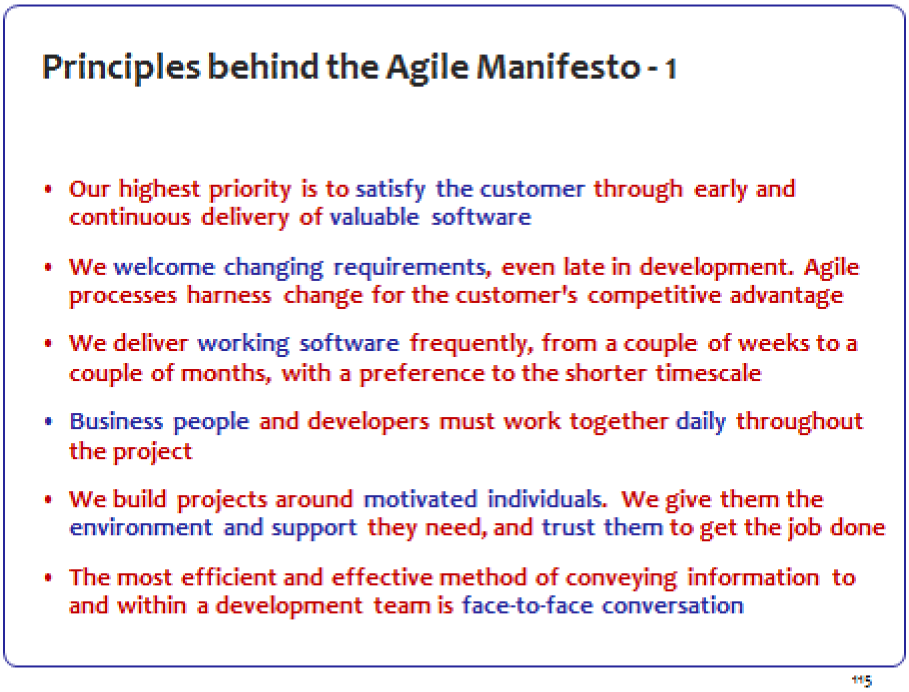
We are uncovering better ways of developing software by doing it and helping others do it

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

114



Principles behind the Agile Manifesto - 1

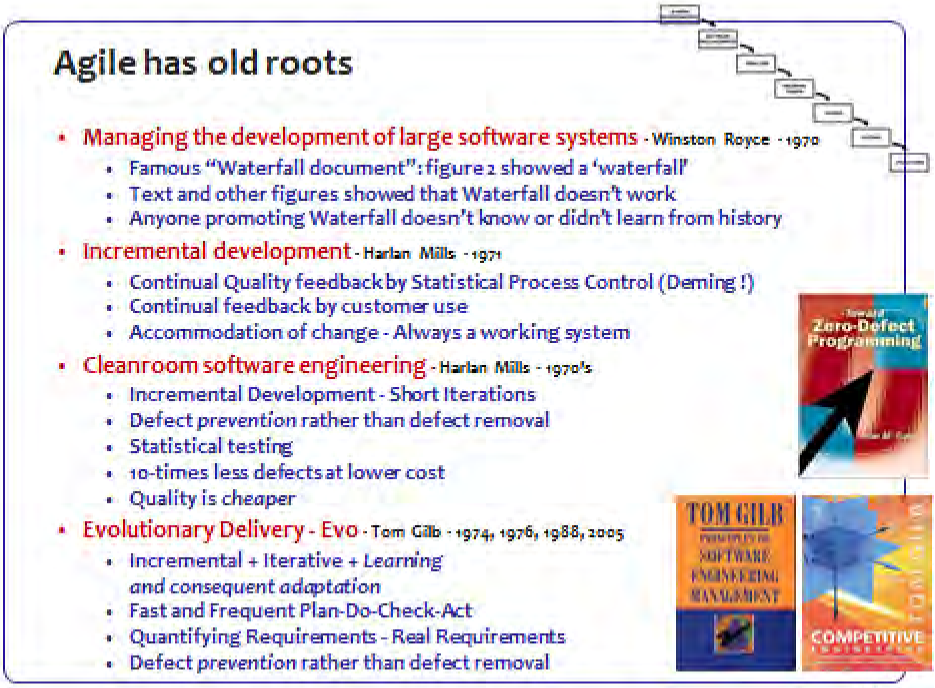
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- We welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
- We deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
- Business people and developers must work together daily throughout the project
- We build projects around motivated individuals. We give them the environment and support they need, and trust them to get the job done
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

115

Principles behind the Agile Manifesto - 2

- Working software is the primary measure of progress
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- Continuous attention to technical excellence and good design enhances agility
- Simplicity - the art of maximizing the amount of work not done - is essential
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

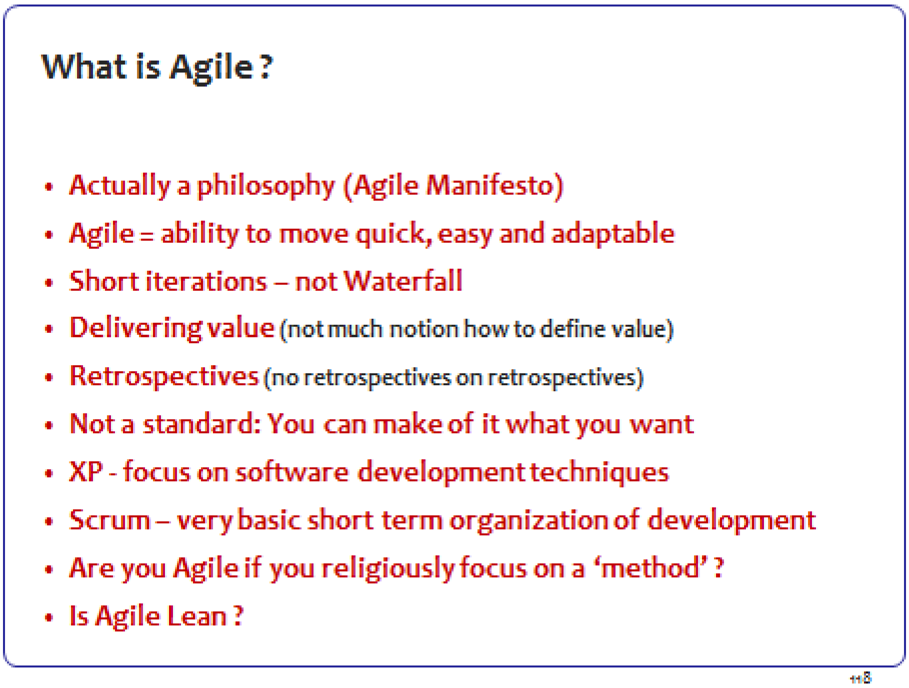
116



Agile has old roots

- 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

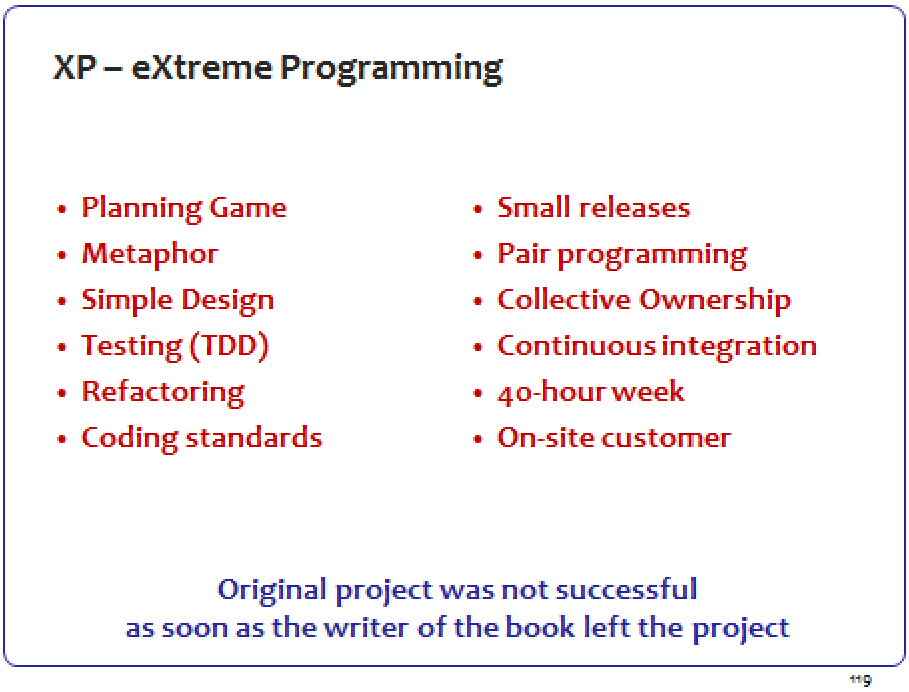
117



What is Agile ?

- Actually a philosophy (Agile Manifesto)
- Agile = ability to move quick, easy and adaptable
- Short iterations – not Waterfall
- Delivering value (not much notion how to define value)
- Retrospectives (no retrospectives on retrospectives)
- Not a standard: You can make of it what 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’ ?
- Is Agile Lean ?

118

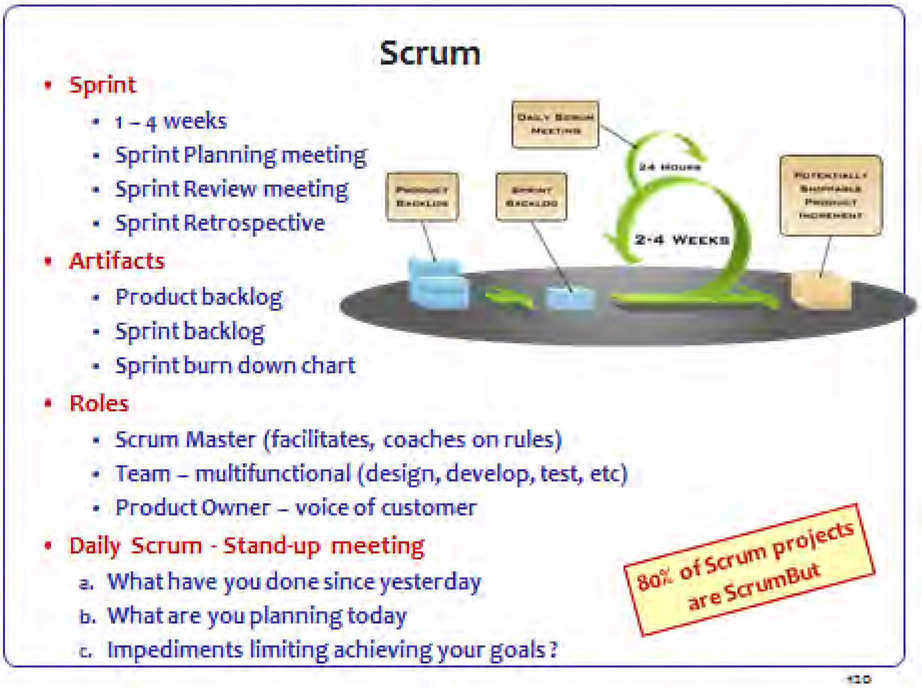


XP – eXtreme Programming

- 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

119

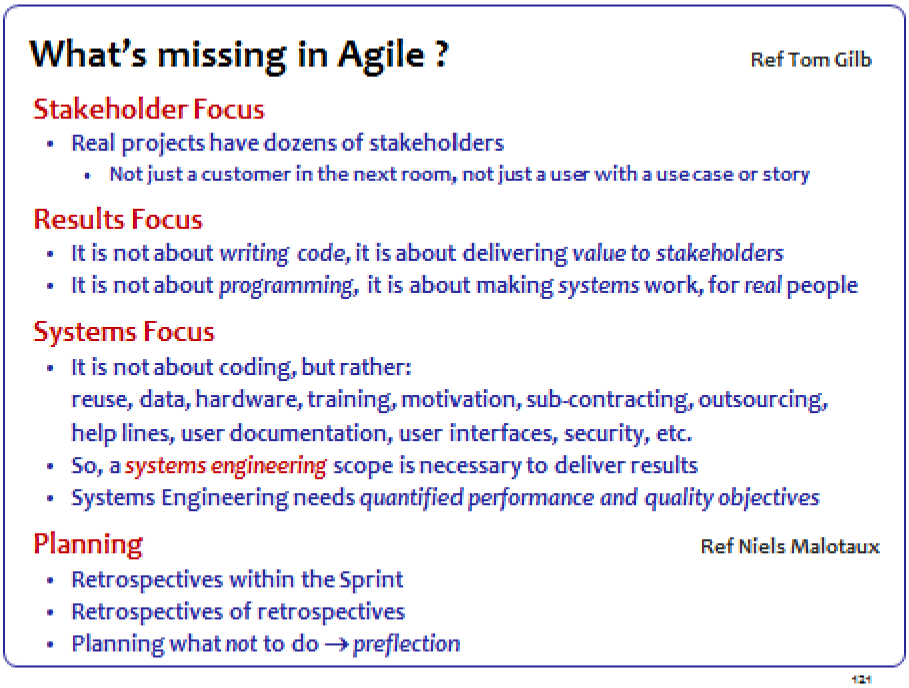


Scrum

- Sprint
  - 1 – 4 weeks
  - Sprint Planning meeting
  - Sprint Review meeting
  - Sprint Retrospective
- Artifacts
  - 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 ?

80% of Scrum projects are ScrumBut

120



What's missing in Agile ?

Ref Tom Gilb

Stakeholder Focus

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

Results Focus

- It is not about *writing code*, it is about delivering *value to stakeholders*
- It is not about *programming*, it is about making *systems* work, for *real* people

Systems Focus

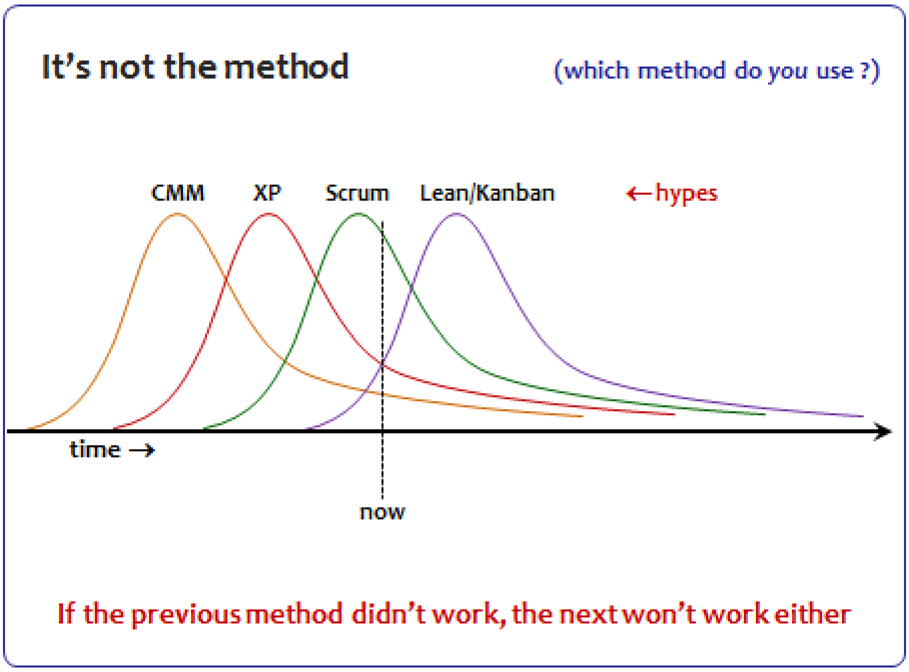
- It is not about coding, but rather:
  reuse, data, hardware, training, motivation, sub-contracting, outsourcing, help lines, user documentation, user interfaces, security, etc.
- So, a *systems engineering* scope is necessary to deliver results
- Systems Engineering needs *quantified performance and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

Planning

Ref Niels Malotaux

- Retrospectives within the Sprint
- Retrospectives of retrospectives
- Planning what *not* to do → *preflection*

121



It's not the method

(which method do you use ?)

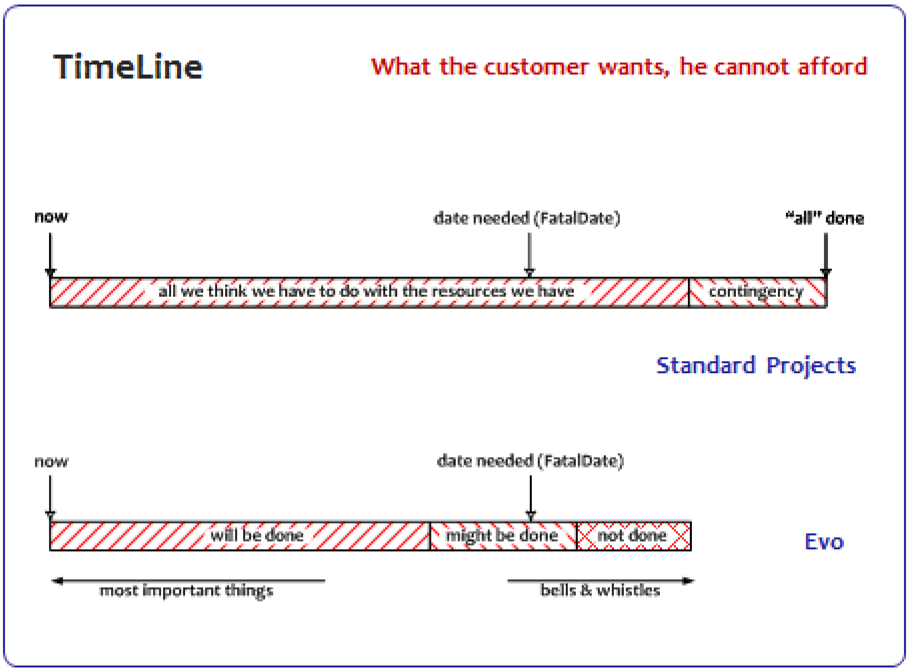
If the previous method didn't work, the next won't work either

122

Evolutionary Planning

TimeLine

123



TimeLine

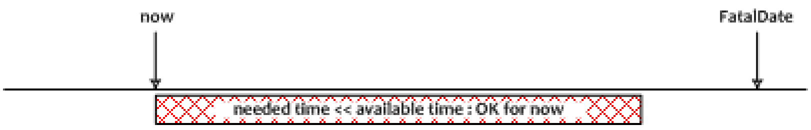
What the customer wants, he cannot afford

Standard Projects

Evo

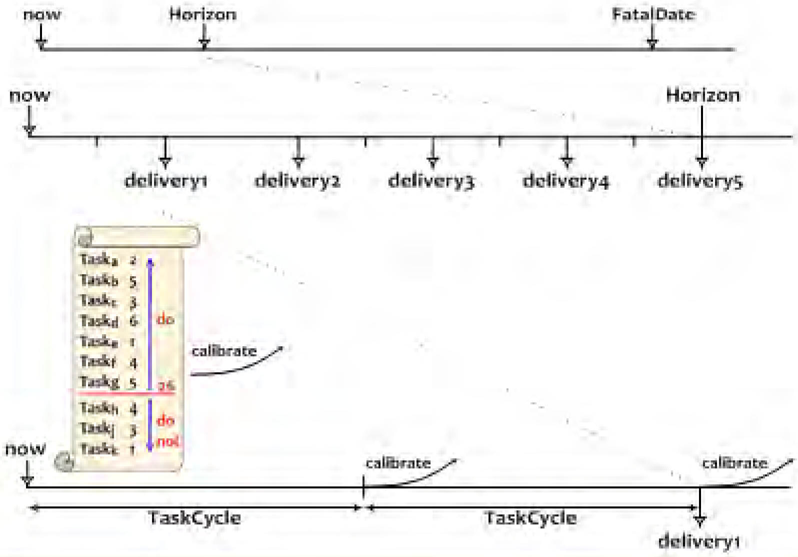
124

If it easily fits ...

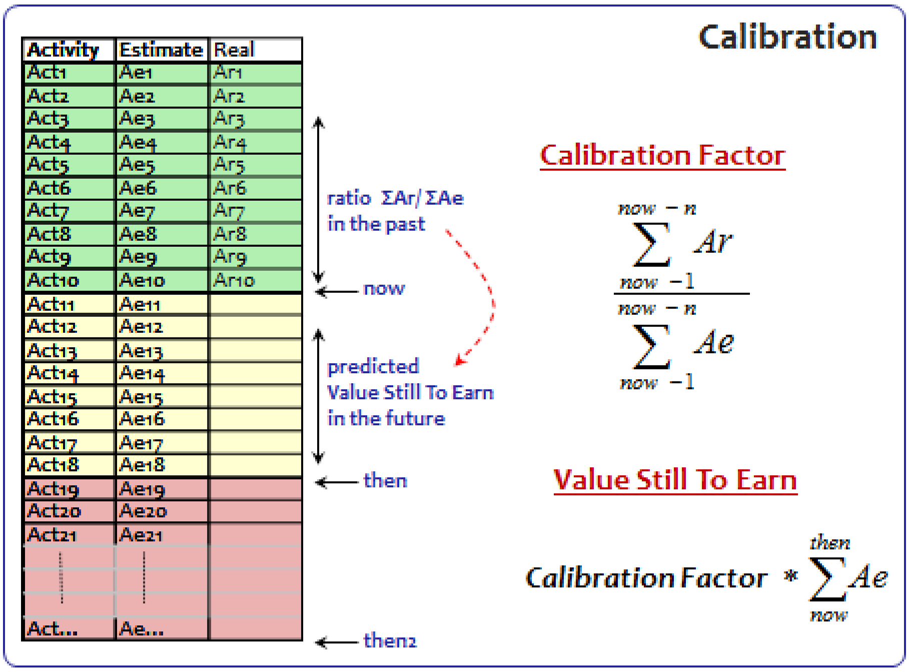


125

Result to Tasks and back



126



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...	

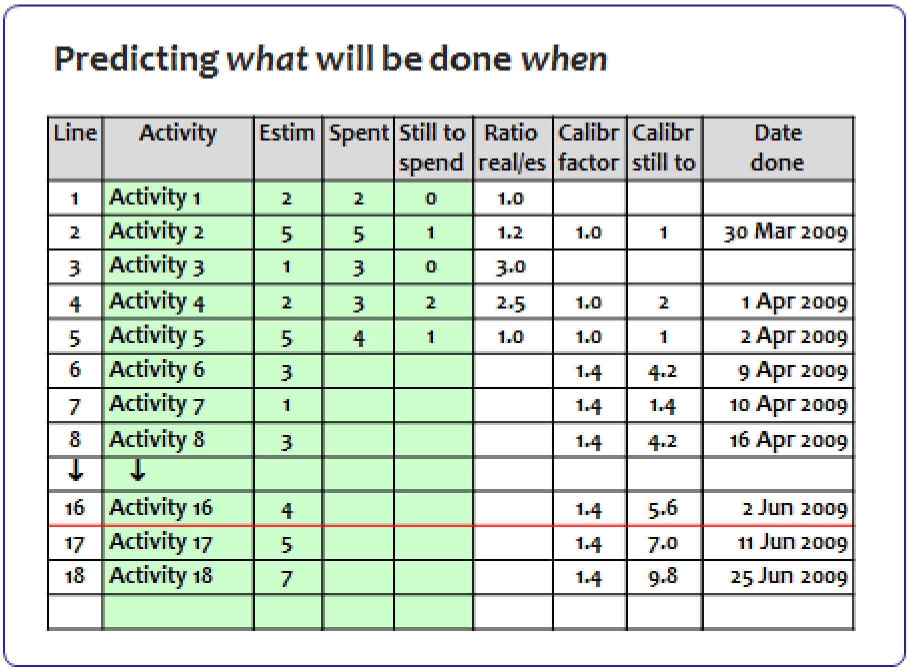
Calibration Factor

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

Value Still To Earn

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

127



Predicting *what* will be done *when*

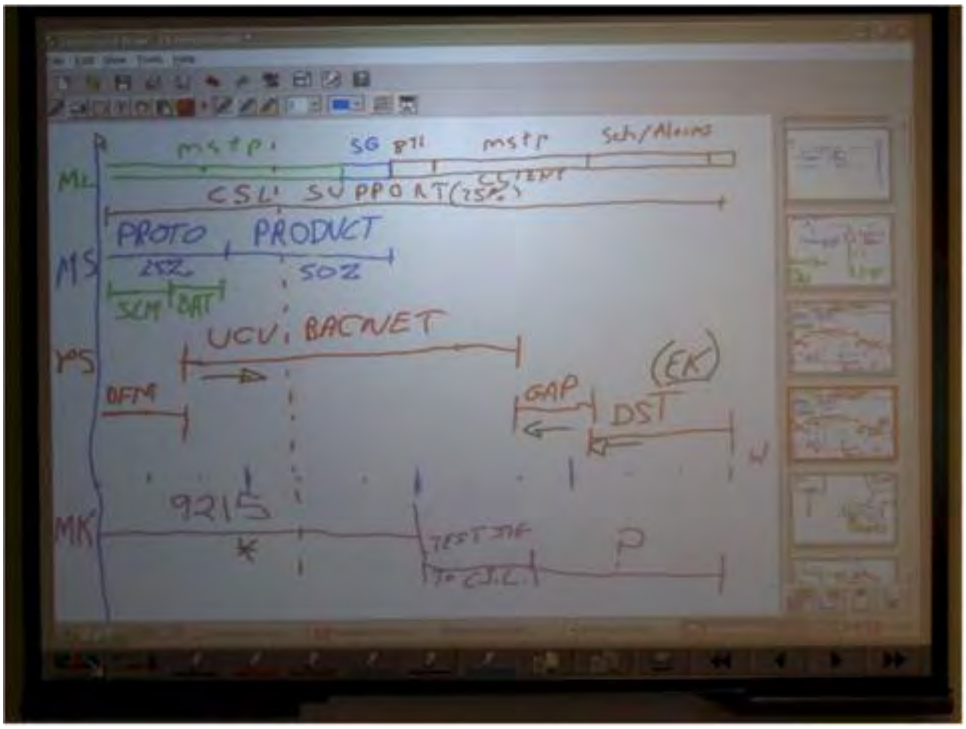
Line	Activity	Estim	Spent	Still to spend	Ratio real/es	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009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

128

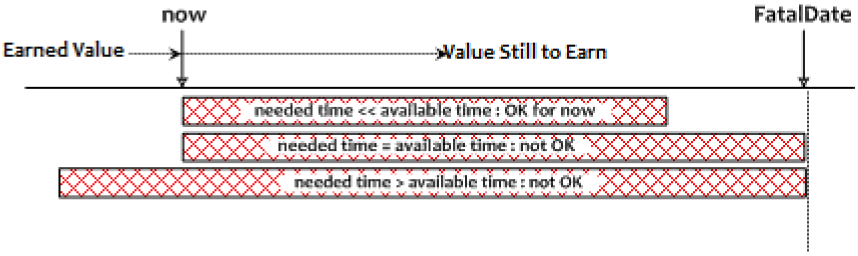
Product/Portfolio/Resource Management

- 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

129



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 (a FatalDate is a FatalDate)

131

Deceptive options

- Hoping for the best (fatalistic)
- Going for it (macho)
- Working Overtime (fooling ourselves – 過労死)
- 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

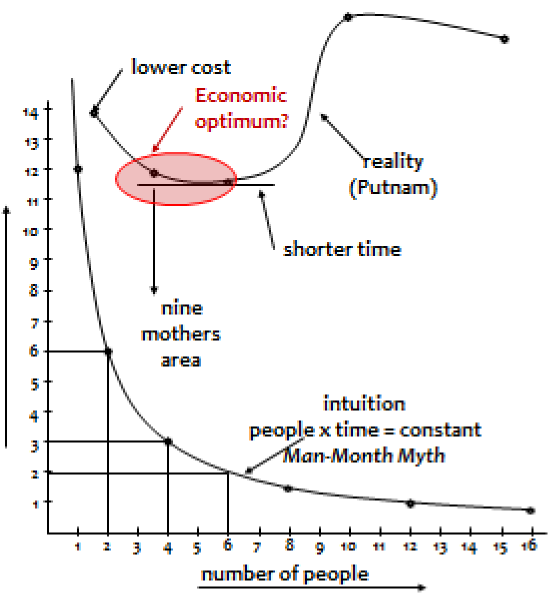
132

Adding people to a late project ...

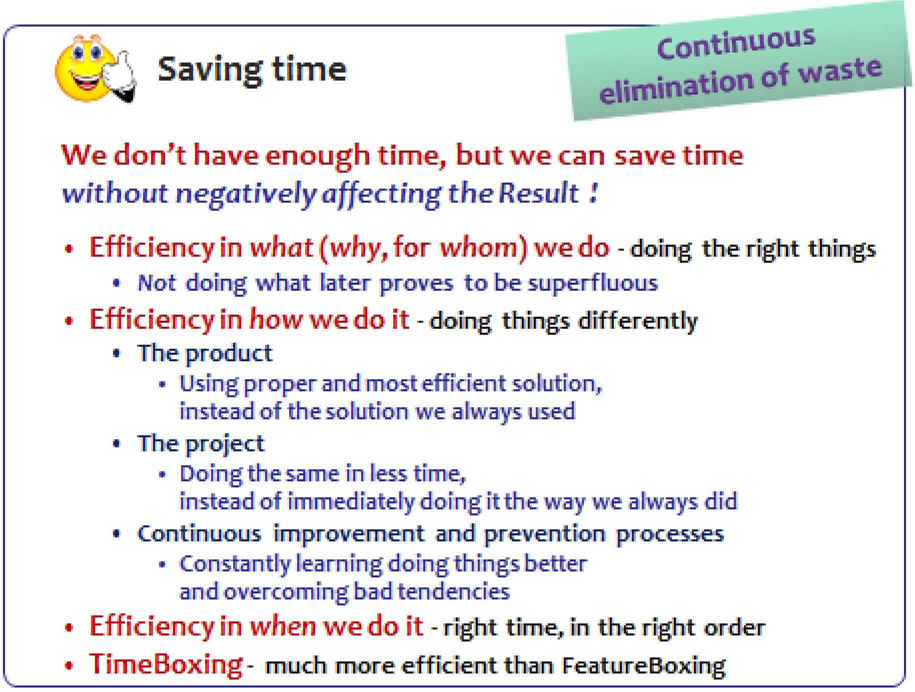
makes it later

(Brooks' Law, 1975)

The Myth of the Man-Month



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



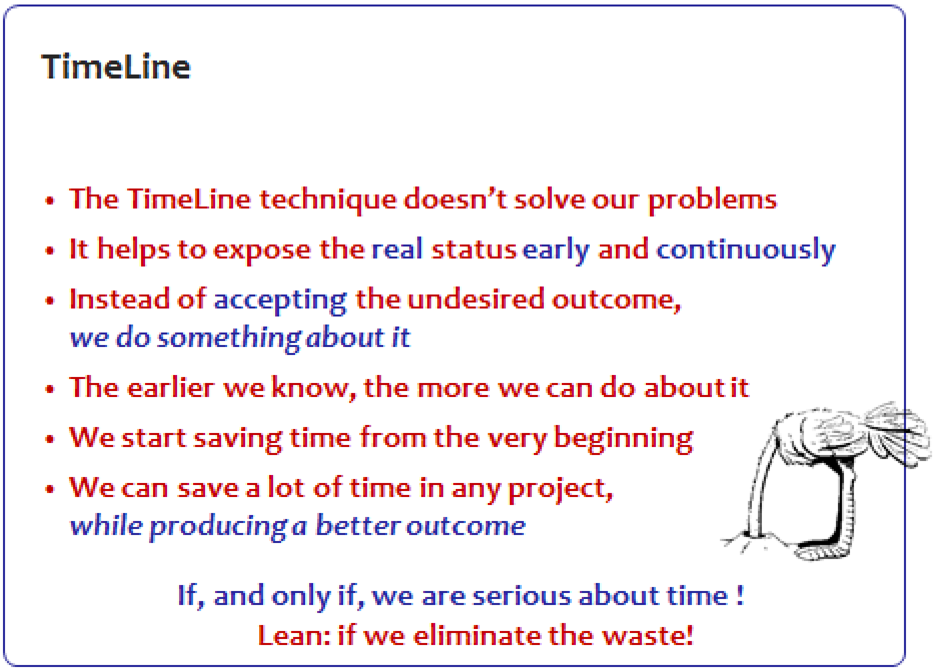
Saving time

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

135

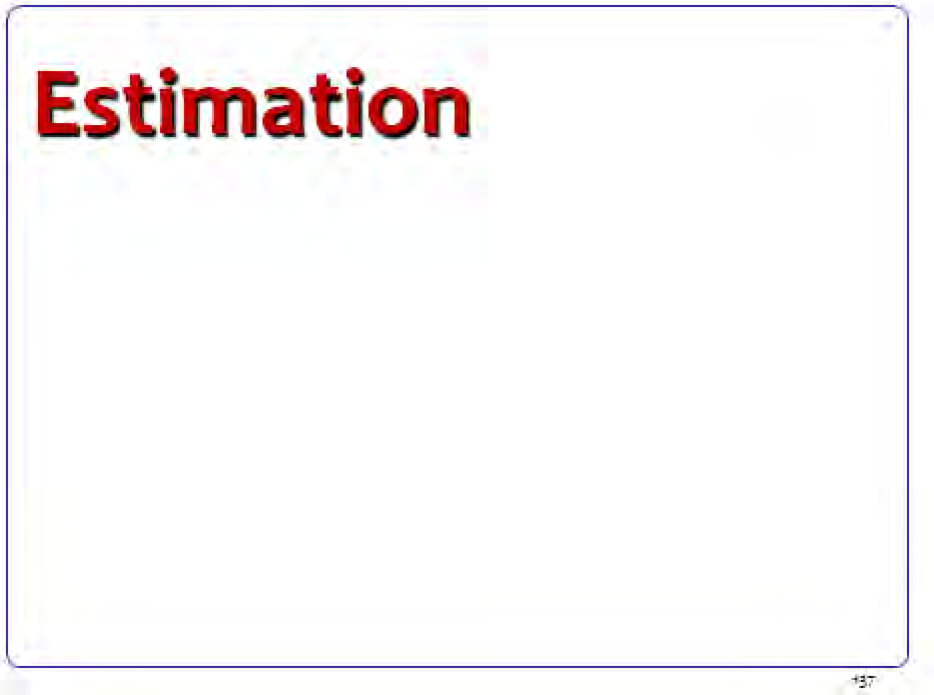


TimeLine

- 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*

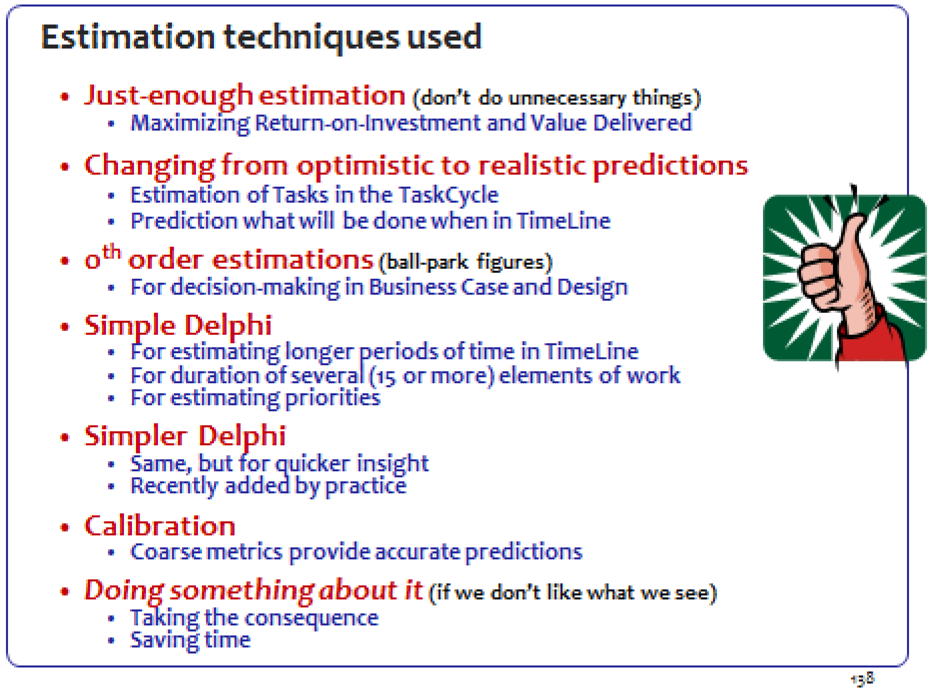
If, and only if, we are serious about time !
Lean: if we eliminate the waste!

136



Estimation

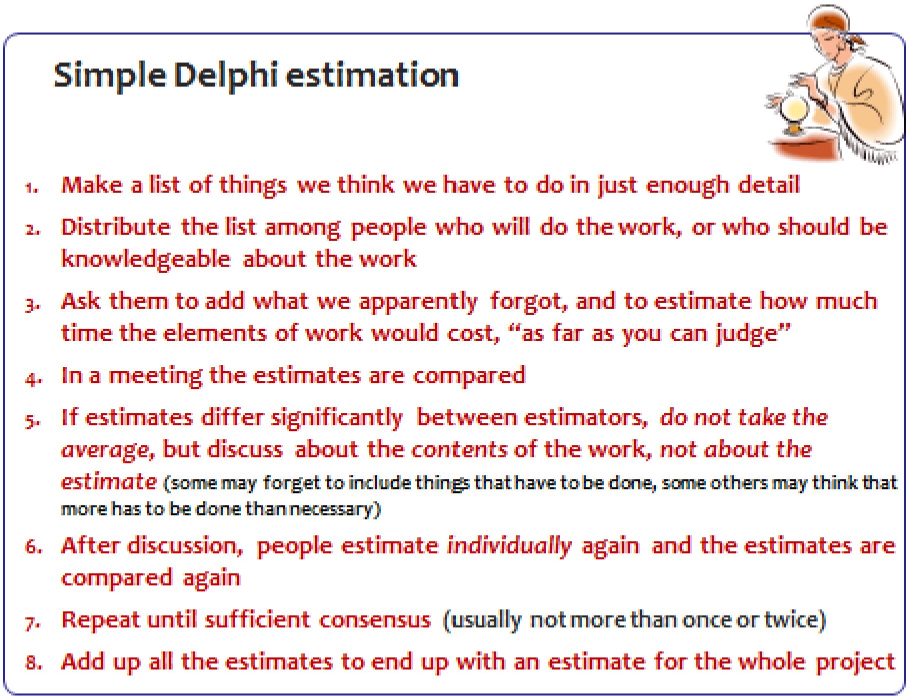
137



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
  - For estimating priorities
- Simpler Delphi
  - 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

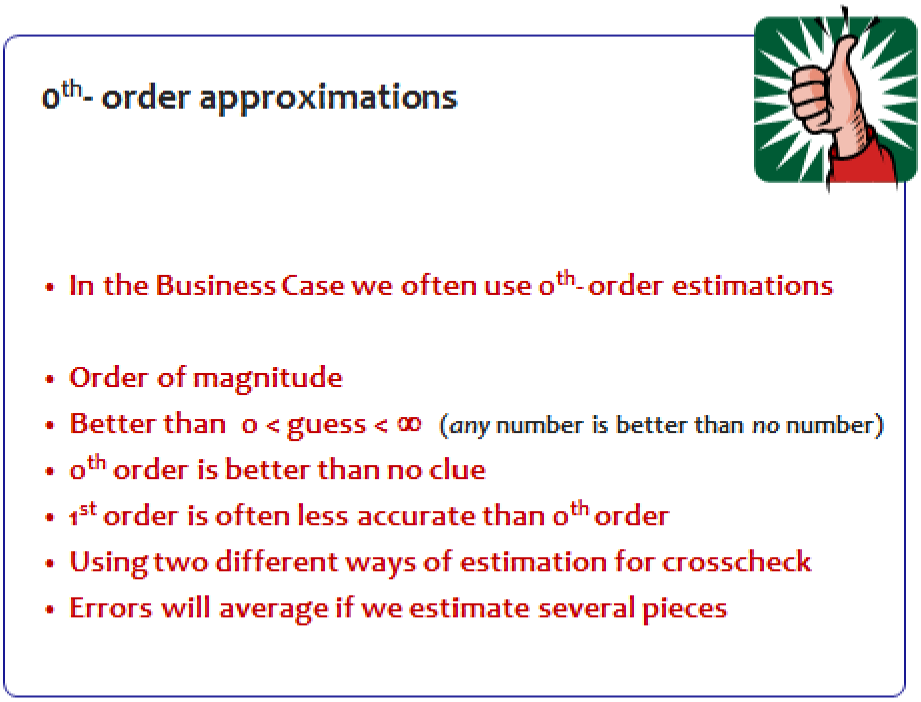
138



Simple Delphi estimation

1. Make a list of things we think we have to do in just enough detail
2. Distribute the list among people who will do the work, or who should be knowledgeable about the work
3. Ask them to add what we apparently forgot, and to estimate how much time the elements of work would cost, "as far as you can judge"
4. In a meeting the estimates are compared
5. If estimates differ significantly between estimators, *do not take the average*, but discuss about the *contents* of the work, *not about the estimate* (some may forget to include things that have to be done, some others may think that more has to be done than necessary)
6. After discussion, people estimate *individually* again and the estimates are compared again
7. Repeat until sufficient consensus (usually not more than once or twice)
8. Add up all the estimates to end up with an estimate for the whole project

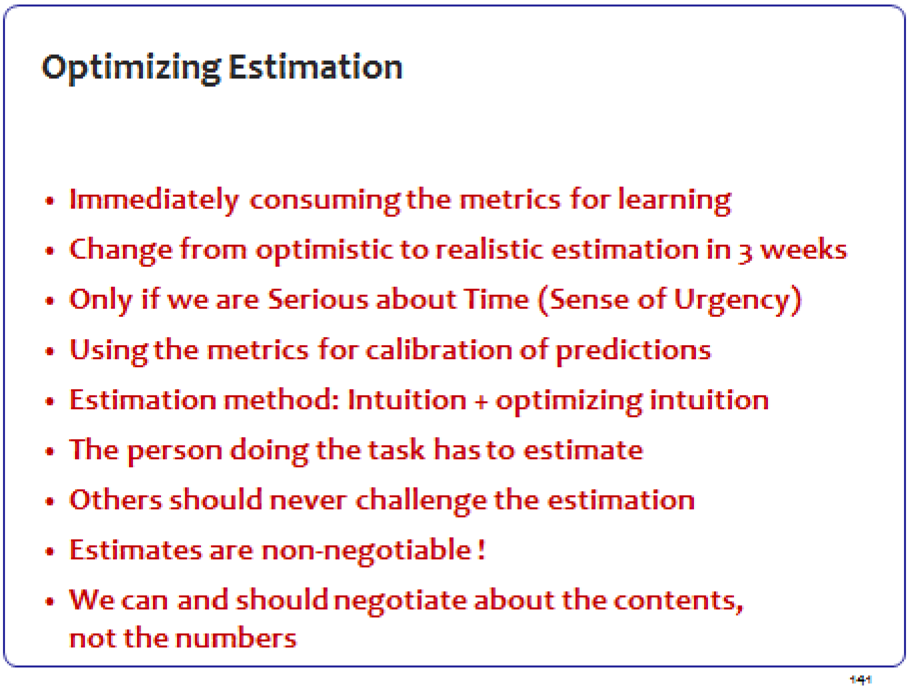
139



0^{th}- order approximations

- In the Business Case we often use 0^{th}- order estimations
- Order of magnitude
- Better than $0 < guess < \infty$ (*any* number is better than *no* number)
- 0^{th} order is better than no clue
- 1^{st} order is often less accurate than 0^{th} order
- Using two different ways of estimation for crosscheck
- Errors will average if we estimate several pieces

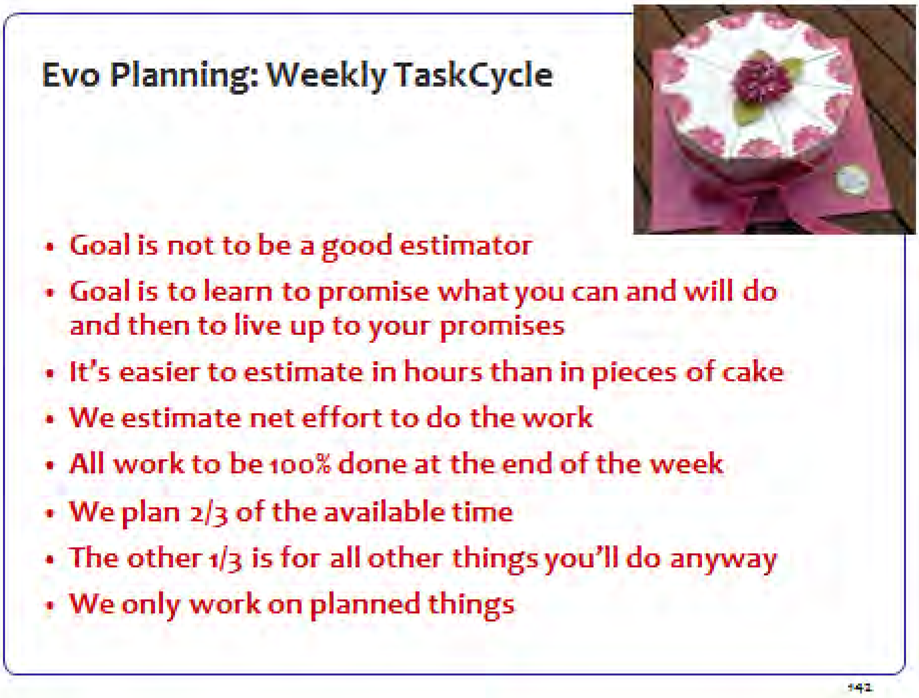
140



Optimizing Estimation

- Immediately consuming the metrics for learning
- Change from optimistic to realistic estimation in 3 weeks
- Only if we are Serious about Time (Sense of Urgency)
- Using the metrics for calibration of predictions
- Estimation method: Intuition + optimizing intuition
- The person doing the task has to estimate
- Others should never challenge the estimation
- Estimates are non-negotiable !
- We can and should negotiate about the contents, not the numbers

141



Evo Planning: Weekly TaskCycle

- Goal is not to be a good estimator
- Goal is to learn to promise what you can and will do and then to live up to your promises
- It's easier to estimate in hours than in pieces of cake
- We estimate net effort to do the work
- All work to be 100% done at the end of the week
- We plan 2/3 of the available time
- The other 1/3 is for all other things you'll do anyway
- We only work on planned things

142

TimeLine examples

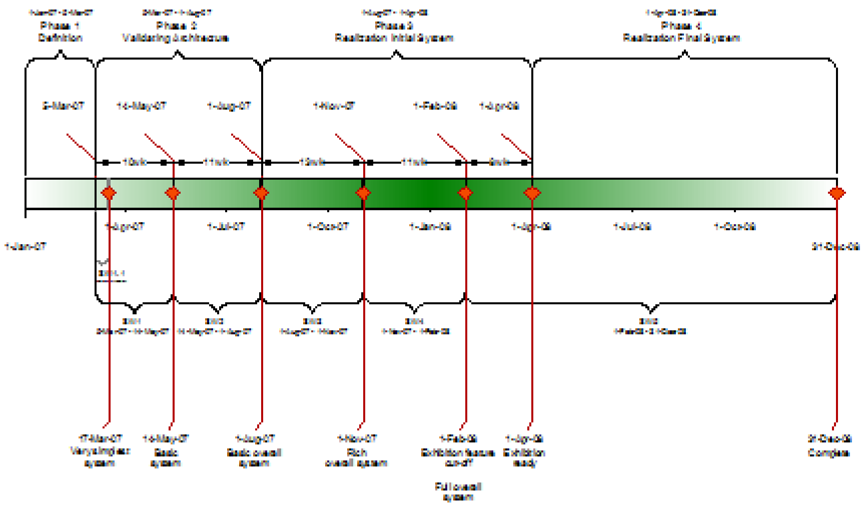
143

Tasks - Deliveries - Projects

- Tasks - Deliveries - Projects actually are similar, except for the *time* and *complexity* scales
  - At the end there is a defined Result, 100% done
  - The journey to the Result should be *designed*
- Use TimeLine on any planning level

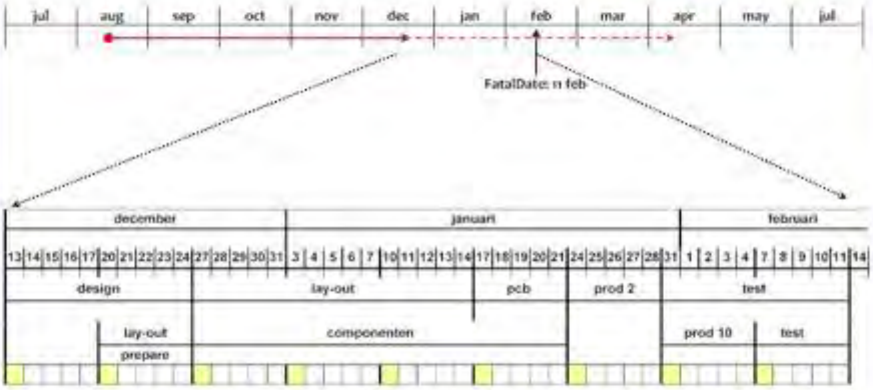
144

TimeLine example



145

TimeLine planning



146

TimeLine exercise example

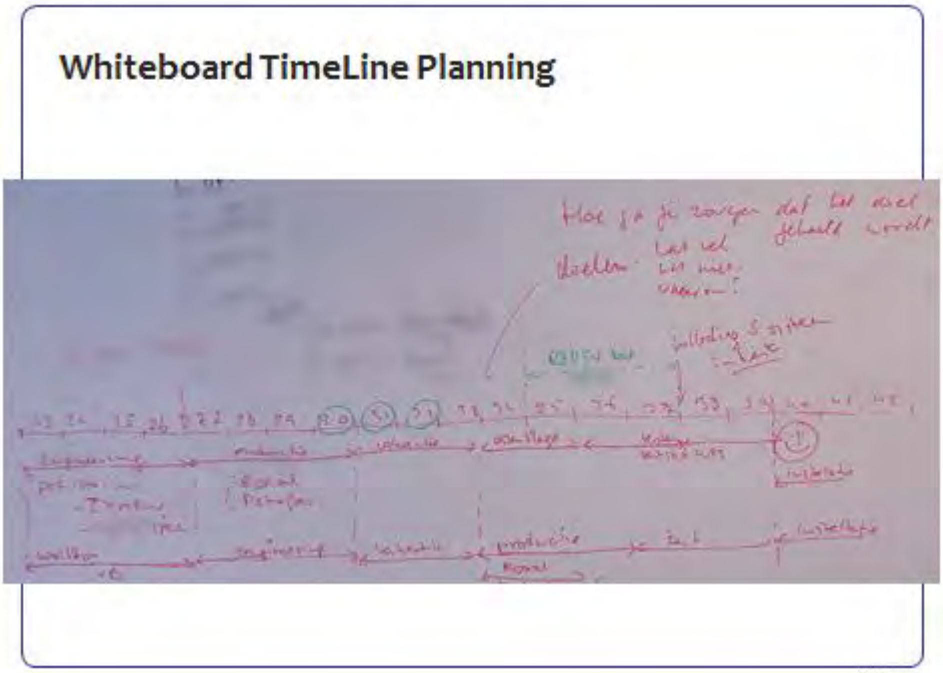
- Preparing for student exams

*

147

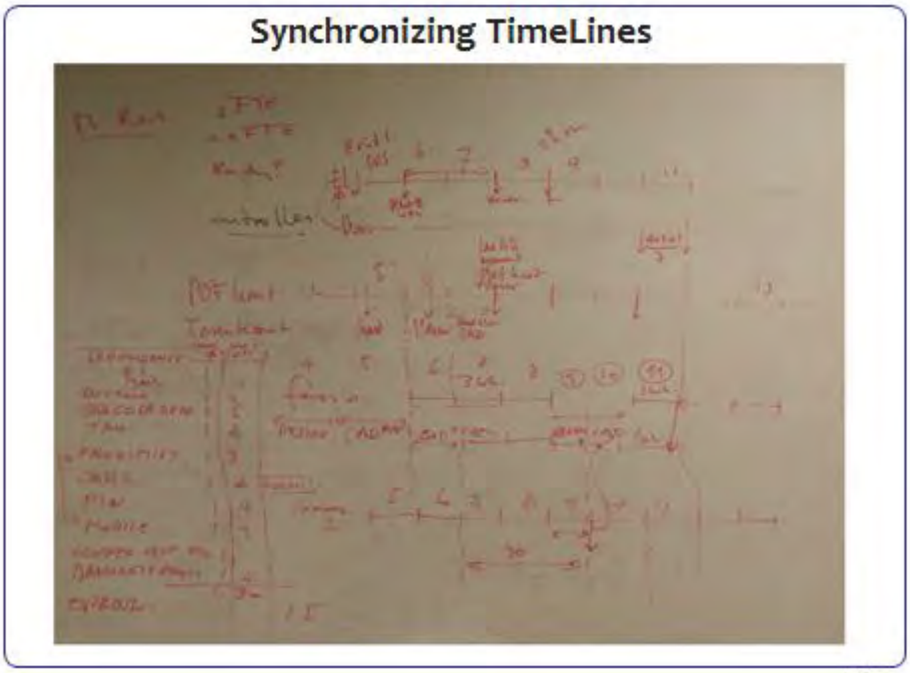
What we did

148



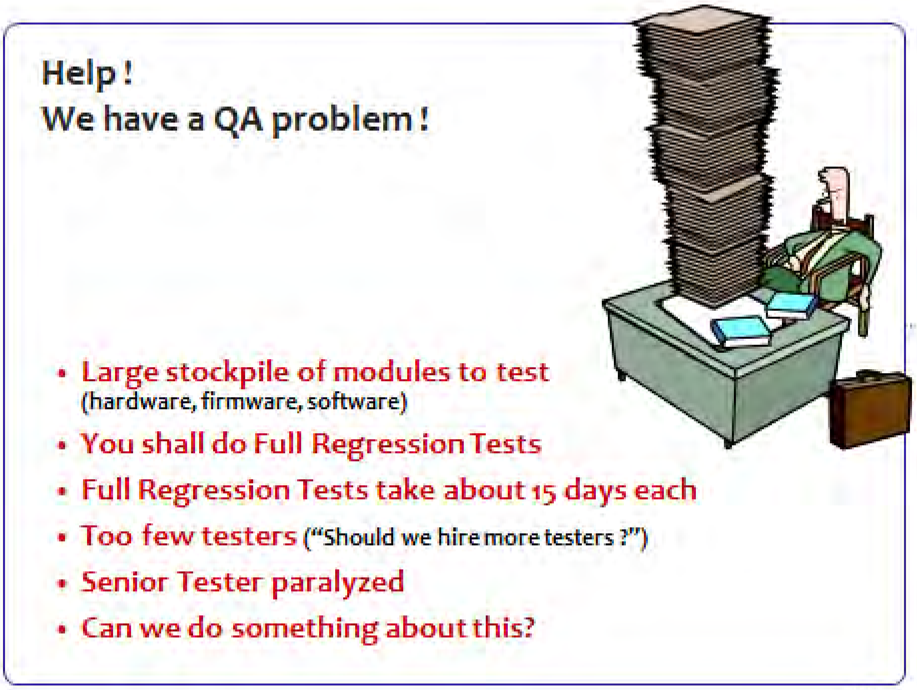
Whiteboard TimeLine Planning

149



Synchronizing TimeLines

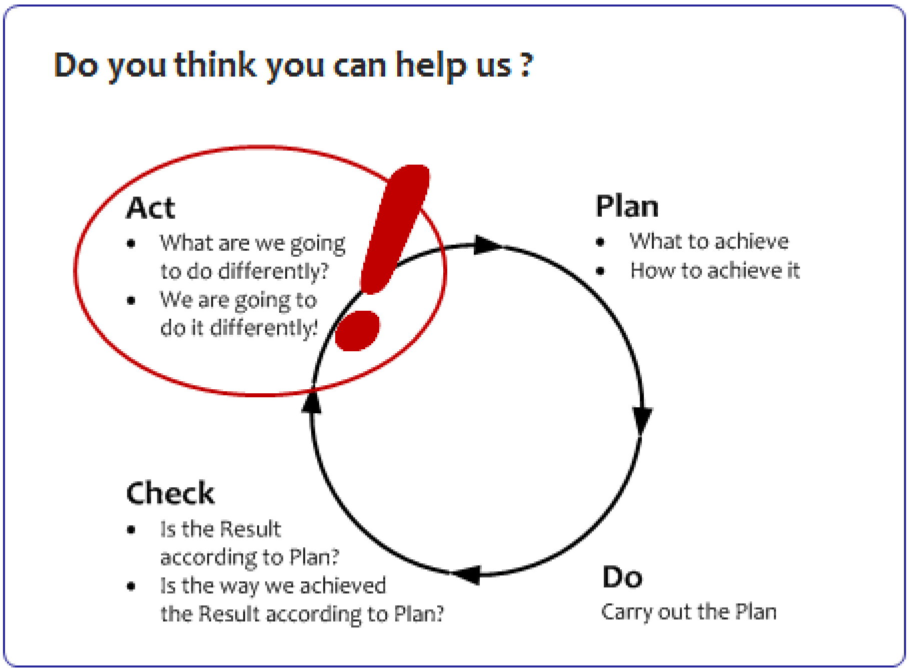
150



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?

151



Do you think you can help us ?

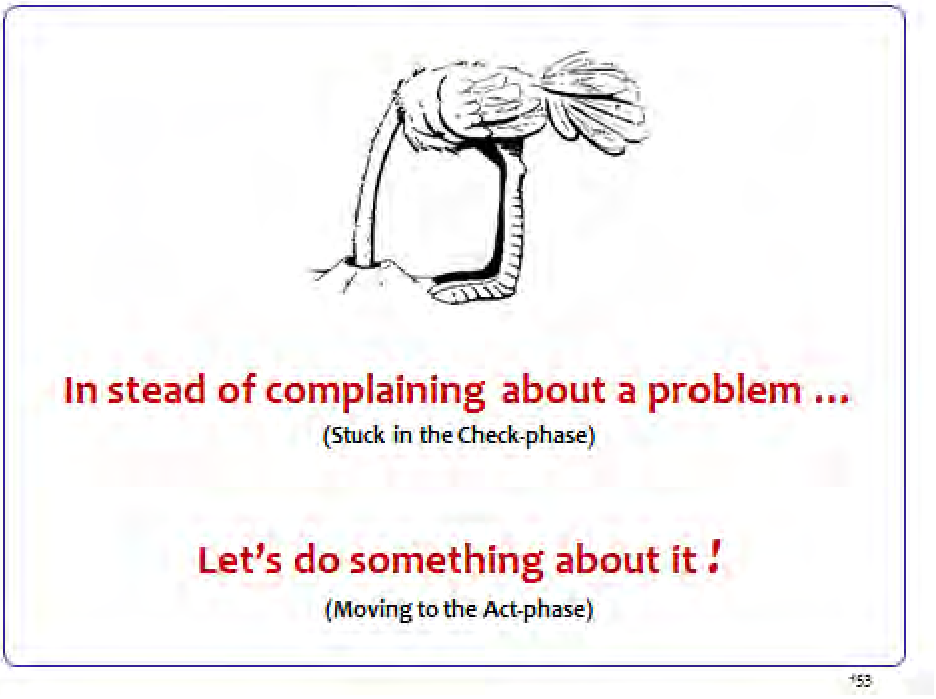
Act
- What are we going to do differently?
- We are going to do it differently!

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

152



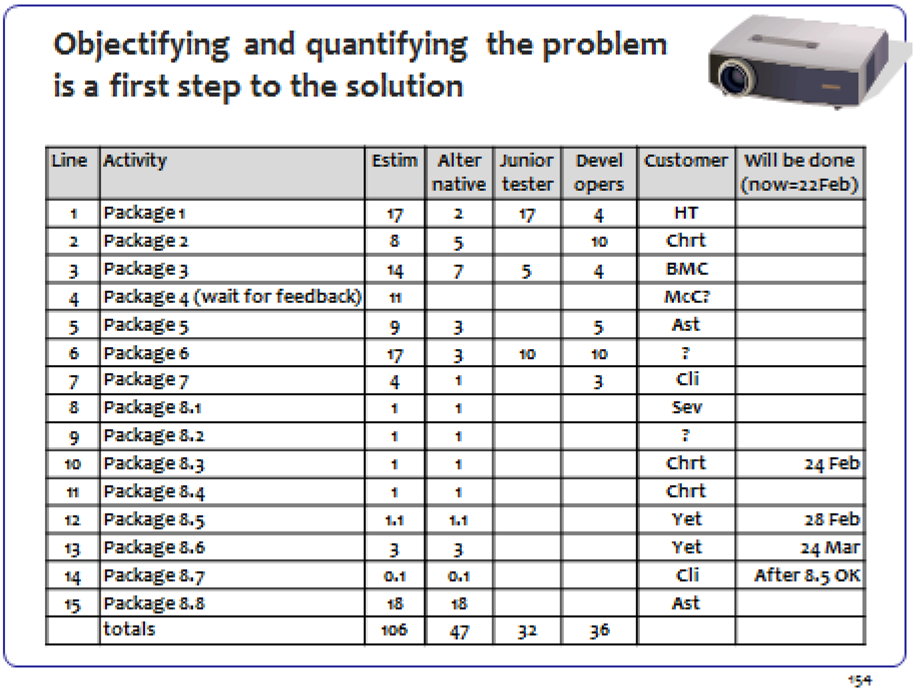
In stead of complaining about a problem ...

(Stuck in the Check-phase)

Let's do something about it !

(Moving to the Act-phase)

153

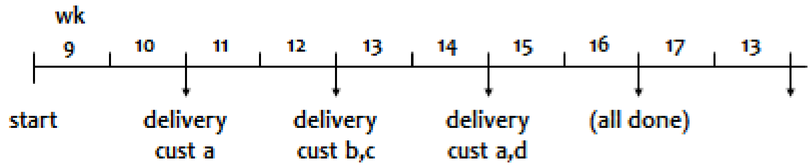


Objectifying and quantifying the problem is a first step to the solution

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		

154

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

155

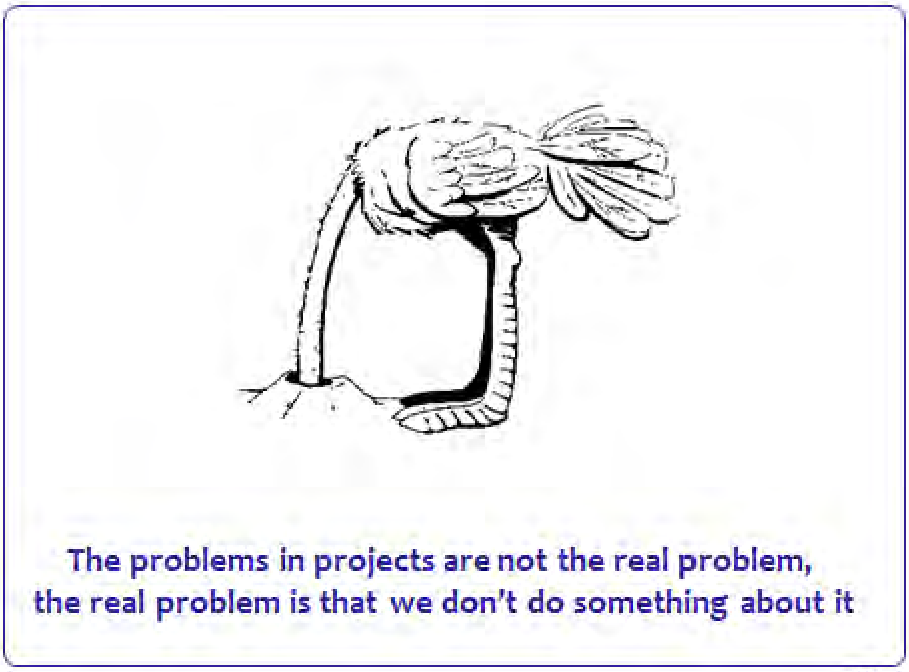
Result

- 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:

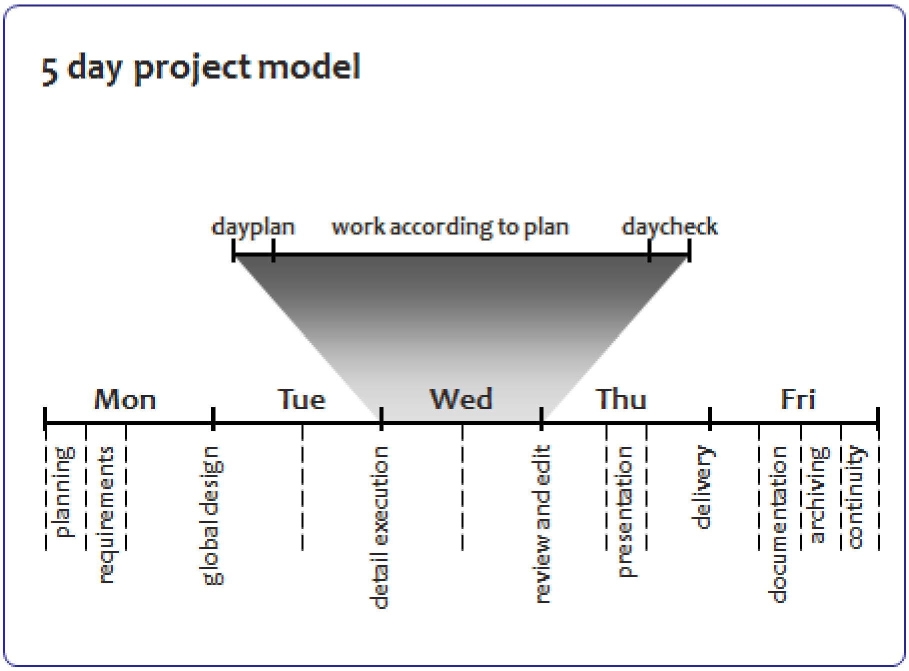
- Tester promoted to product manager
- Still coaching successors how to plan

156



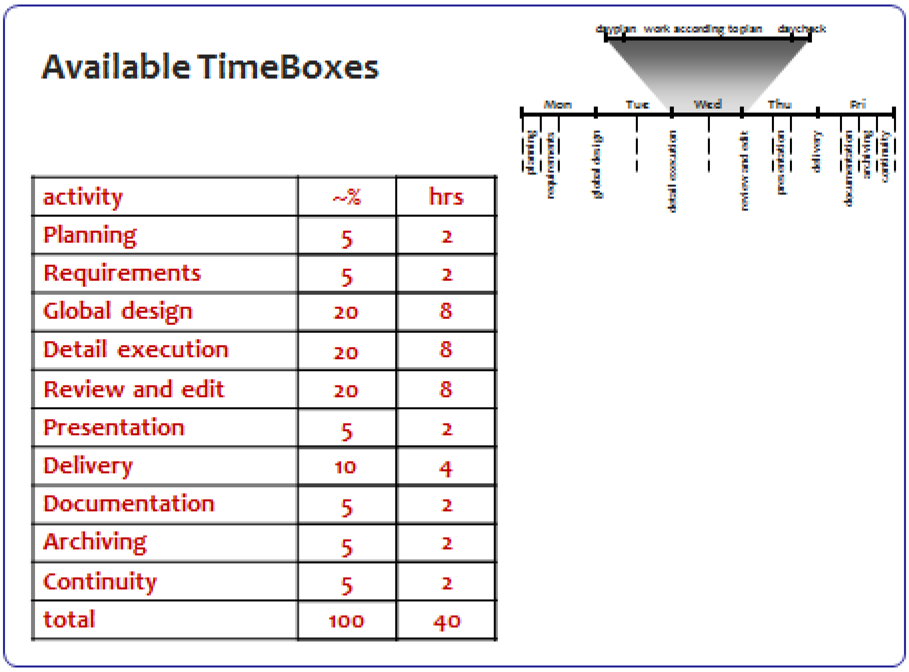
The problems in projects are not the real problem,
the real problem is that we don't do something about it

157



5 day project model

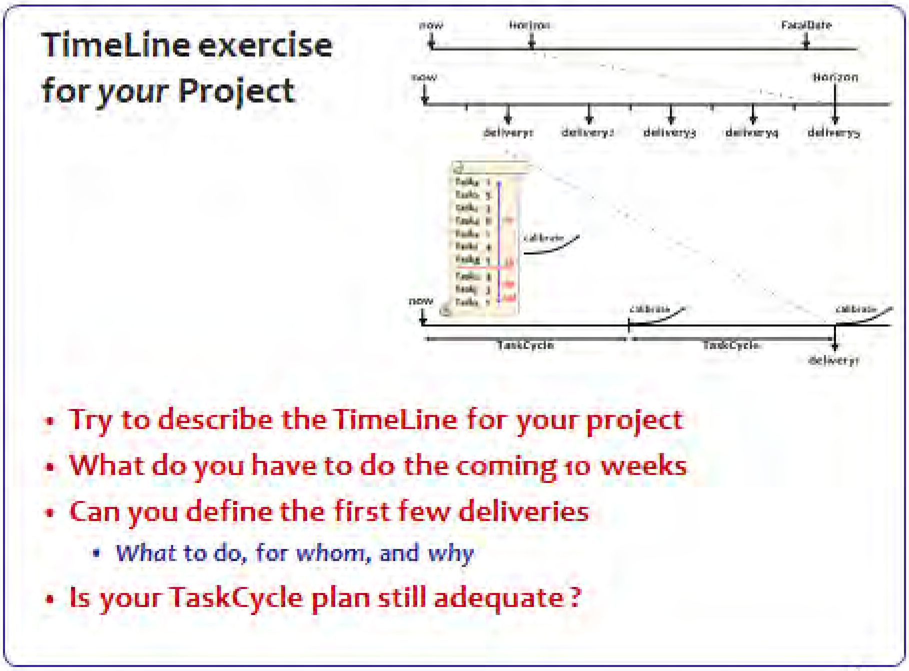
158



Available TimeBoxes

activity	~%	hrs
Planning	5	2
Requirements	5	2
Global design	20	8
Detail execution	20	8
Review and edit	20	8
Presentation	5	2
Delivery	10	4
Documentation	5	2
Archiving	5	2
Continuity	5	2
total	100	40

159



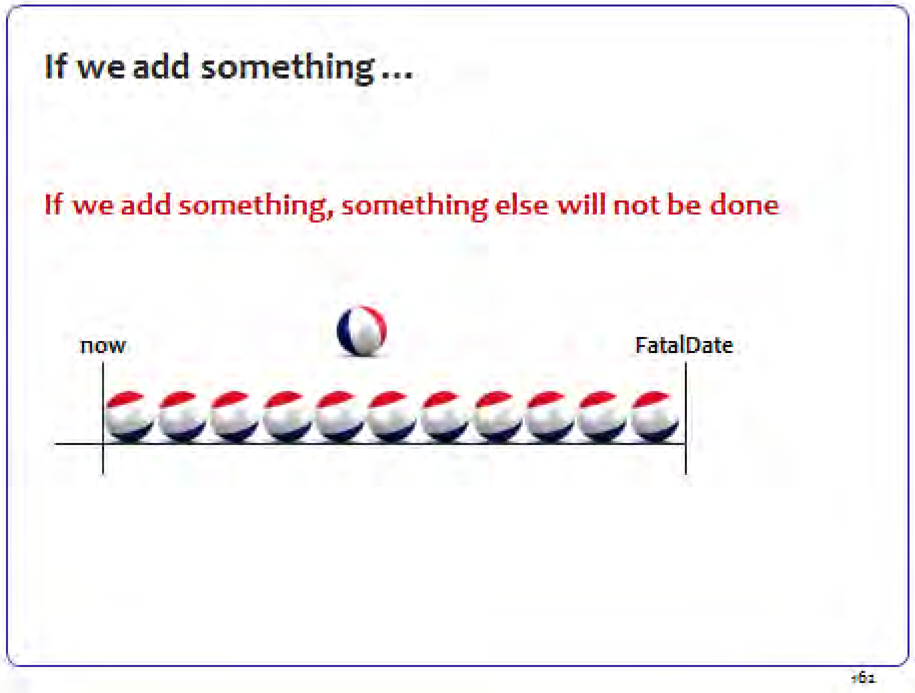
TimeLine exercise for *your* Project

- 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?

160

Some details

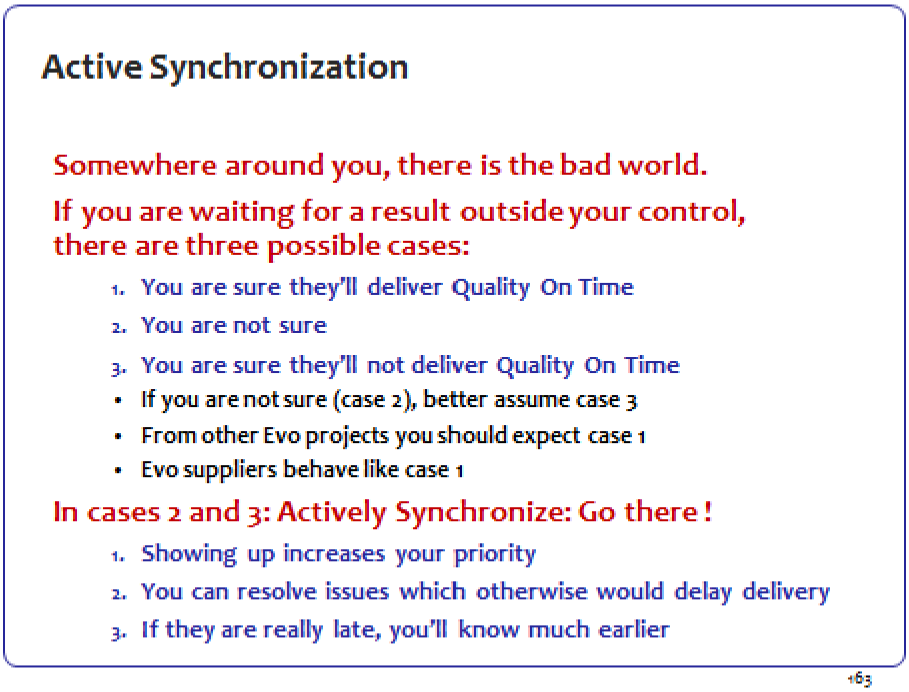
161



If we add something …

If we add something, something else will not be done

162



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:

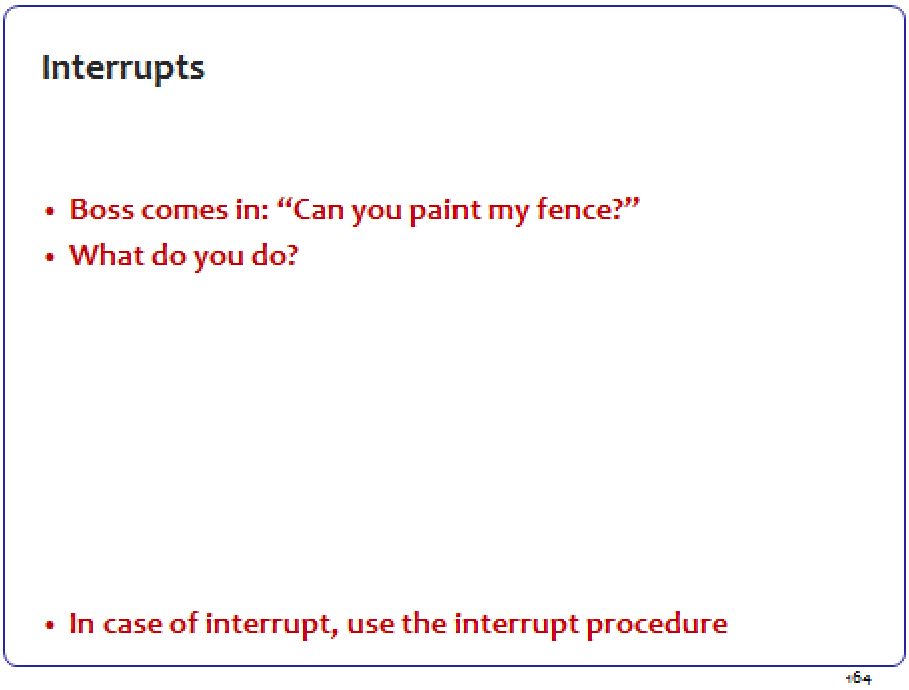
1. You are sure they'll deliver Quality On Time
2. You are not sure
3. You are sure they'll not deliver Quality On Time

- 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

163

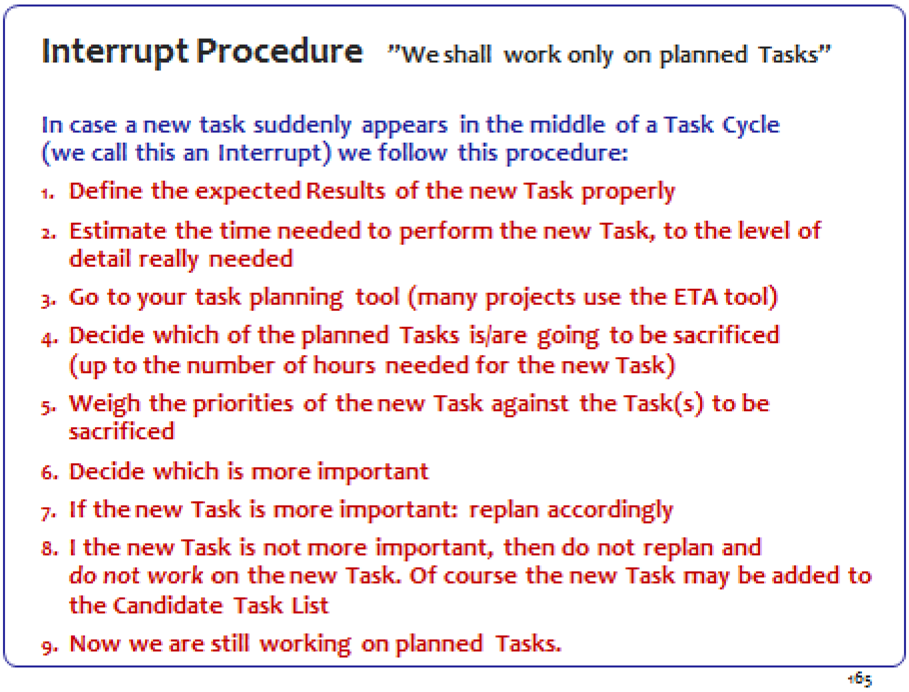


Interrupts

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

- In case of interrupt, use the interrupt procedure

164

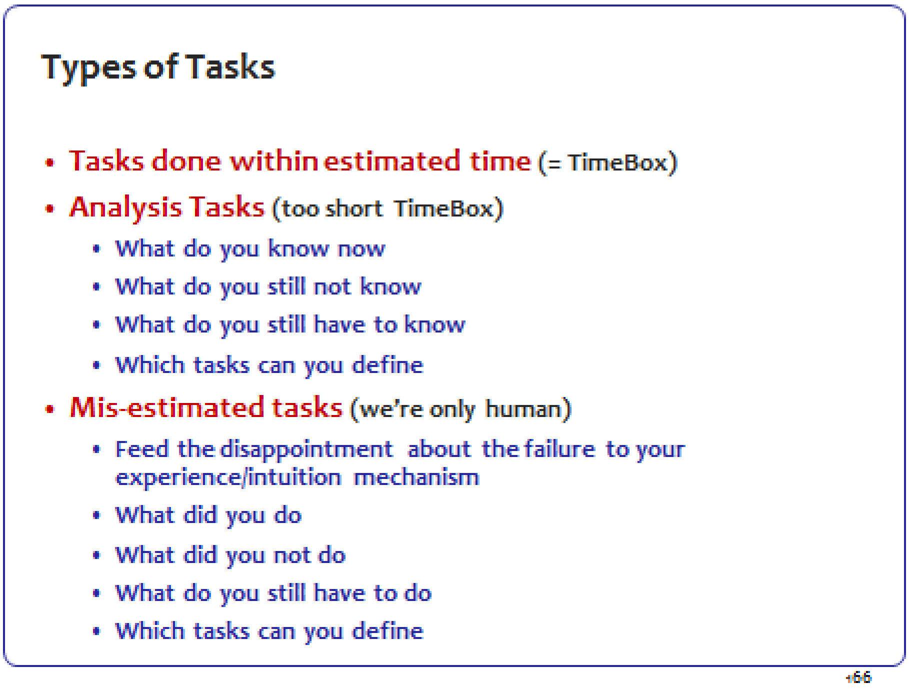


Interrupt Procedure "We shall work only on planned Tasks"

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.

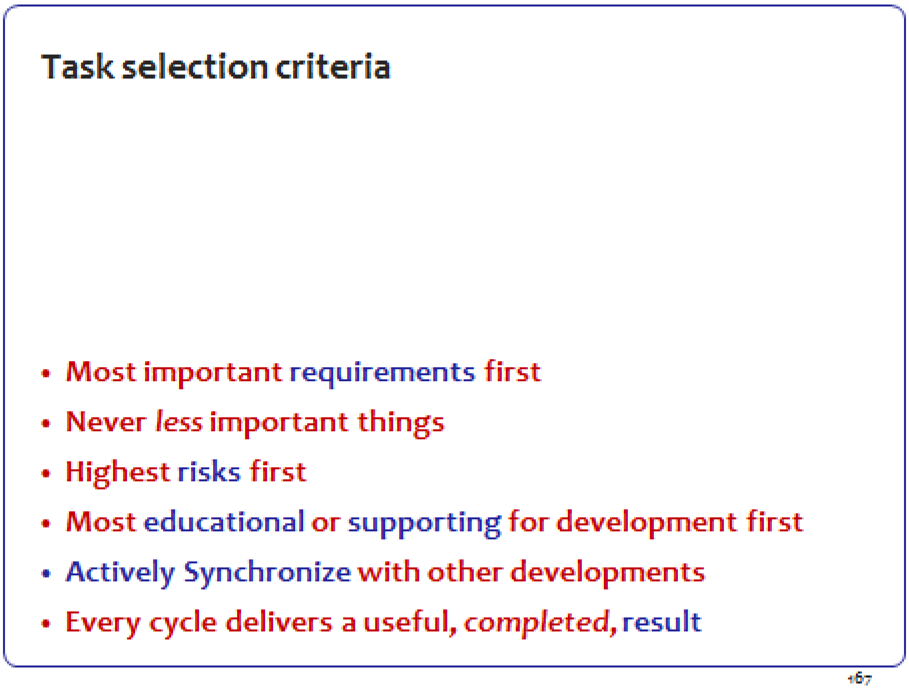
165



Types of Tasks

- Tasks done within estimated time (= TimeBox)
- Analysis Tasks (too short TimeBox)
  - What do you know now
  - What do you still not know
  - What do you still have to know
  - Which tasks can you define
- Mis-estimated tasks (we're only human)
  - Feed the disappointment about the failure to your experience/intuition mechanism
  - What did you do
  - What did you not do
  - What do you still have to do
  - Which tasks can you define

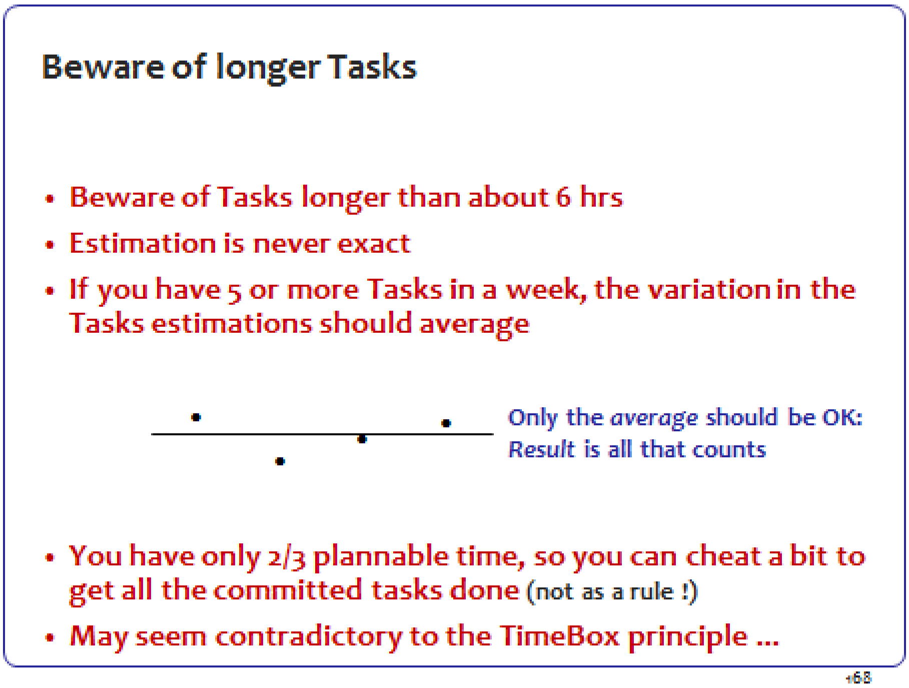
166



Task selection criteria

- Most important requirements first
- Never *less* important things
- Highest risks first
- Most educational or supporting for development first
- Actively Synchronize with other developments
- Every cycle delivers a useful, *completed*, result

167



Beware of longer Tasks

- Beware of Tasks longer than about 6 hrs
- Estimation is never exact
- If you have 5 or more Tasks in a week, the variation in the Tasks estimations should average

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

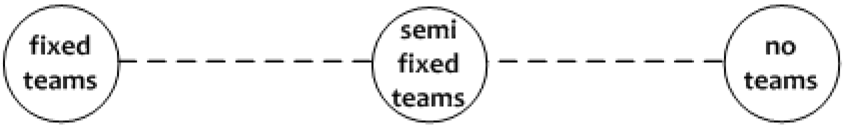
- You have only 2/3 plannable time, so you can cheat a bit to get all the committed tasks done (not as a rule !)
- May seem contradictory to the TimeBox principle ...

168

We work on more projects

- Define how many hours available for this project
- Deliver these hours

Vision:

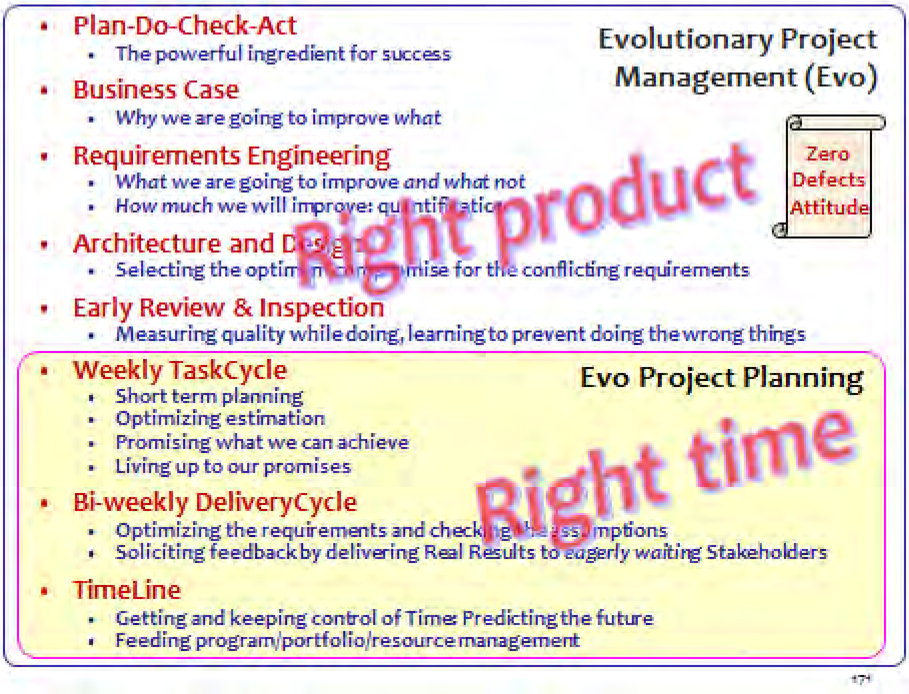


169

My project is different

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

170



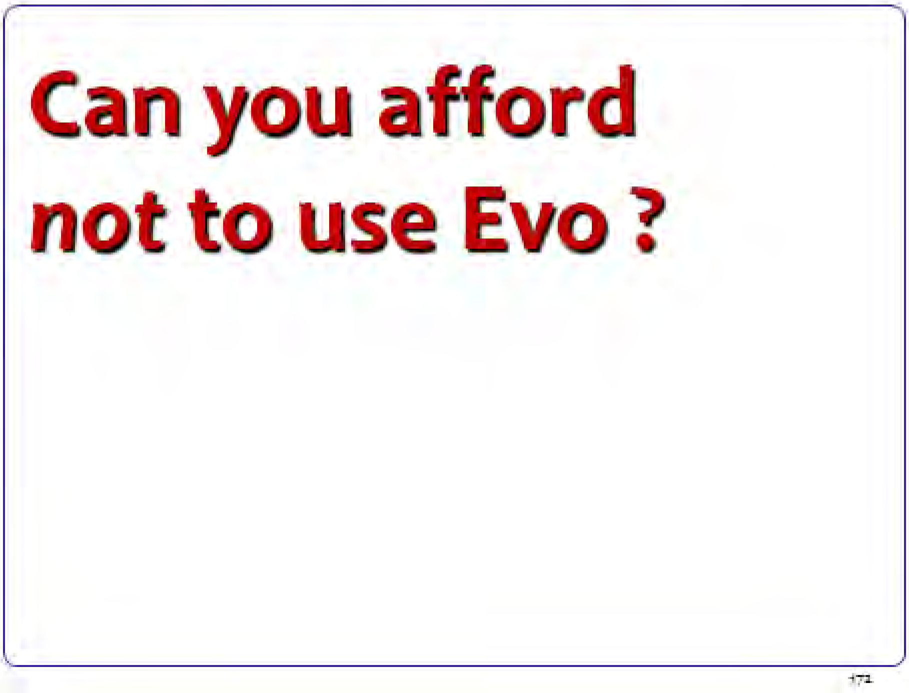
Evolutionary Project Management (Evo)

- 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

Zero Defects Attitude
Right product
Right time
171



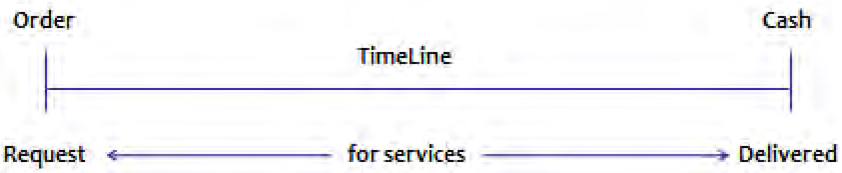
Can you afford not to use Evo ?

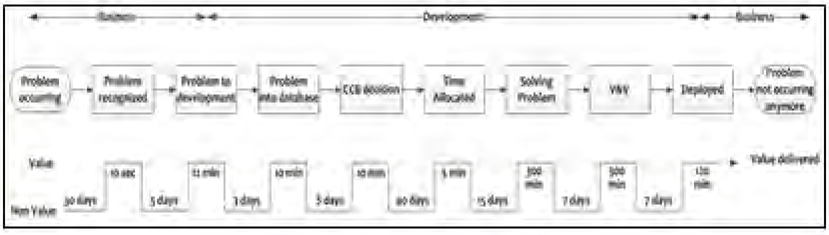
172

Business Case

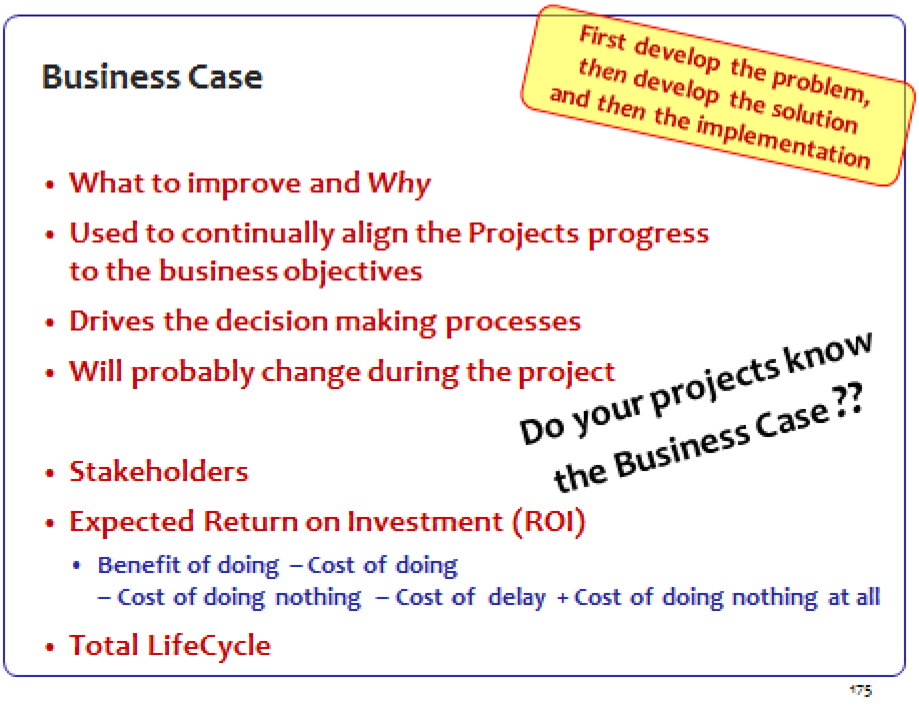
173

Business Case according to Taiichi Ohno (TPS)





174



Business Case

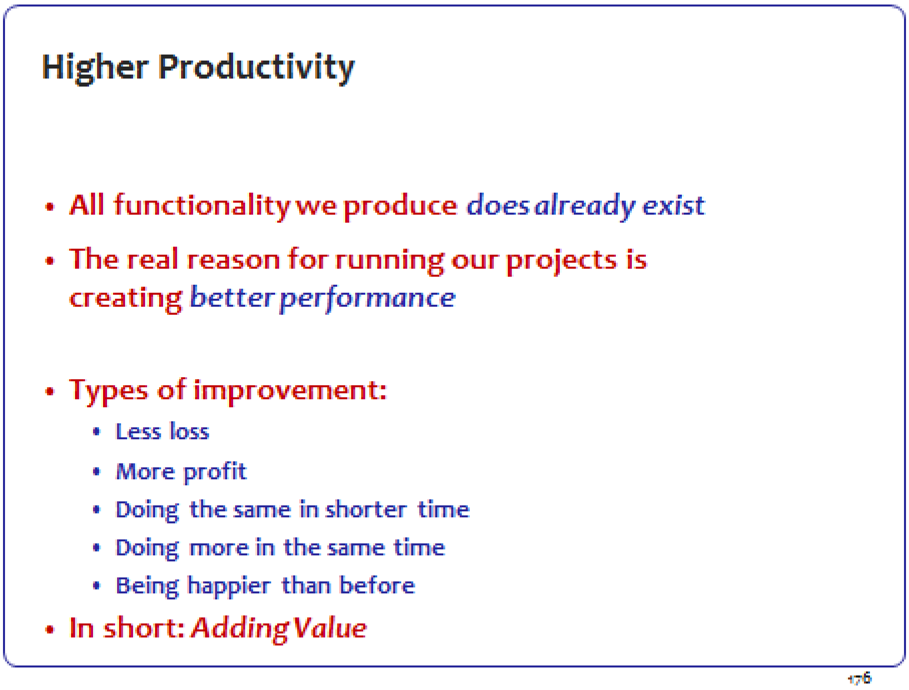
First develop the problem, then develop the solution and then the implementation

- What to improve and Why
- Used to continually align the Projects progress to the business objectives
- Drives the decision making processes
- Will probably change during the project

Do your projects know the Business Case ??

- Stakeholders
- Expected Return on Investment (ROI)
  - Benefit of doing – Cost of doing – Cost of doing nothing – Cost of delay + Cost of doing nothing at all
- Total LifeCycle

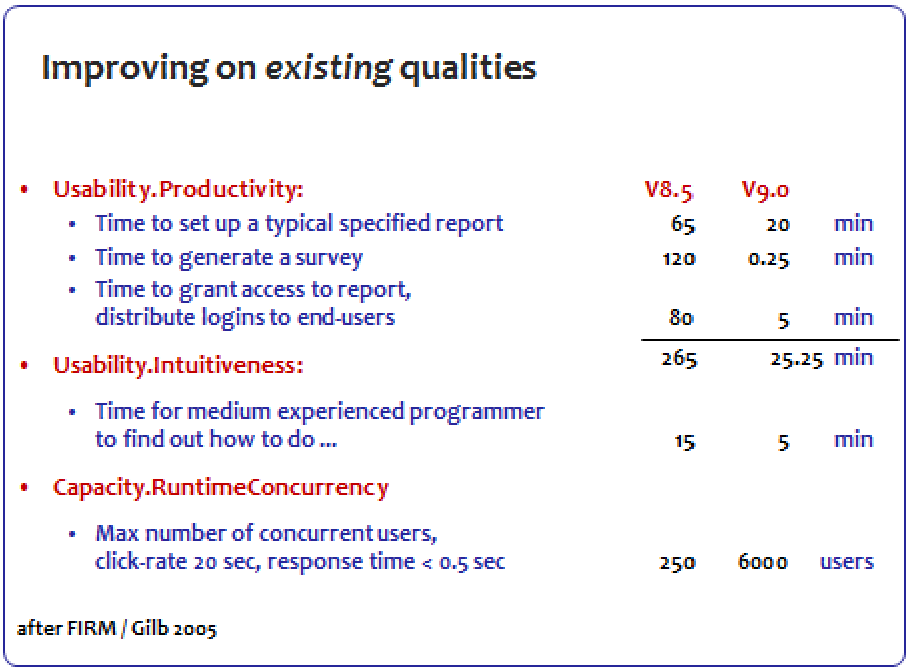
175



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*

176

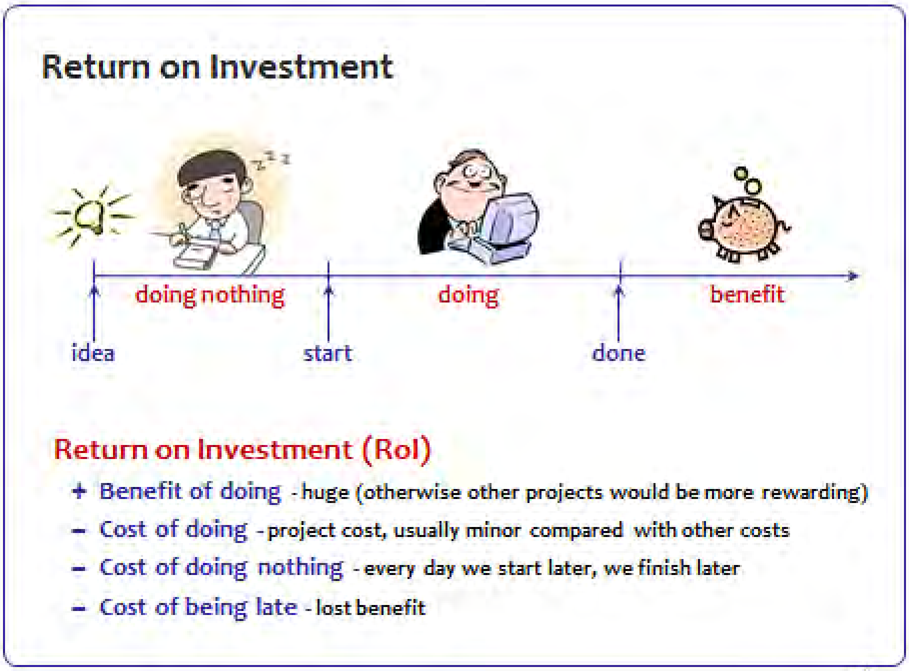


Improving on *existing* qualities

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

after FIRM / Gilb 2005

177

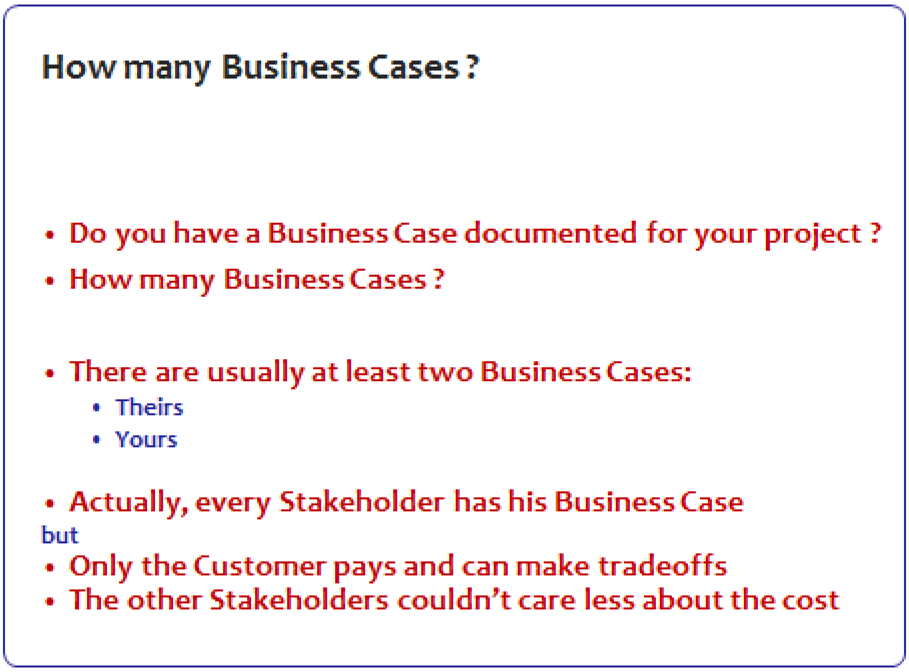


Return on Investment

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

178



How many Business Cases ?

- Do you have a Business Case documented for your project ?
- How many Business Cases ?

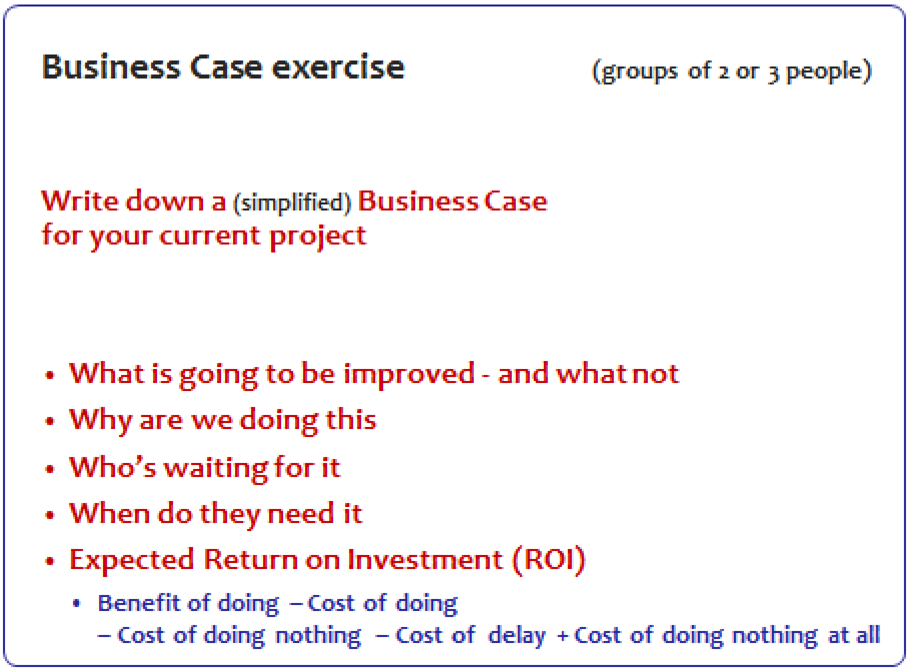
- There are usually at least two Business Cases:
  - Theirs
  - Yours

- 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

179



Business Case exercise (groups of 2 or 3 people)

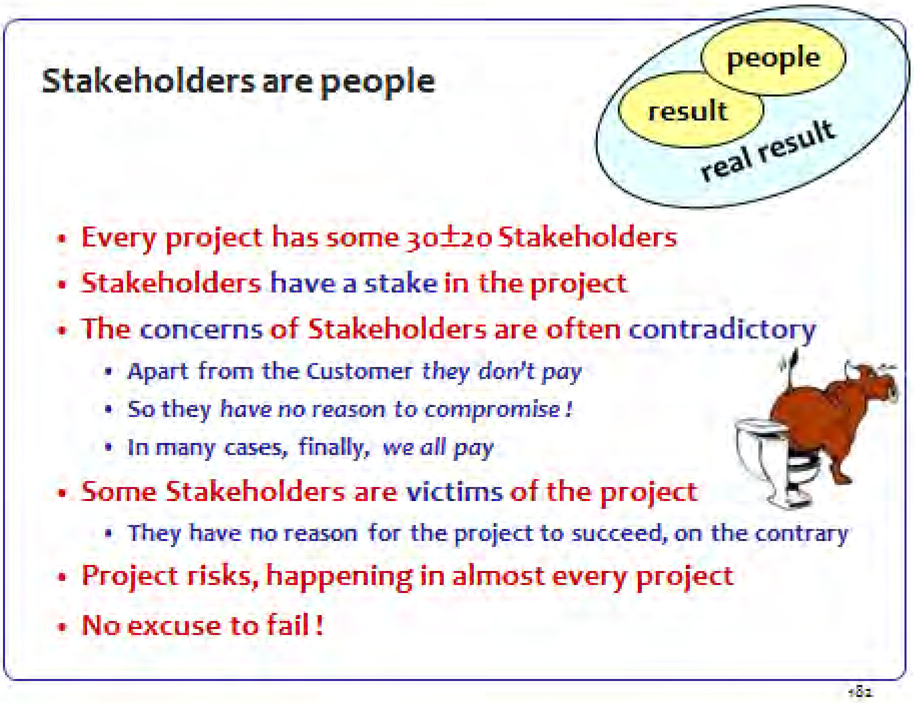
Write down a (simplified) Business Case
for your current project

- What is going to be improved - and what not
- Why are we doing this
- Who's waiting for it
- When do they need it
- Expected Return on Investment (ROI)
  - Benefit of doing – Cost of doing
    – Cost of doing nothing – Cost of delay + Cost of doing nothing at all

180

Stakeholders & Requirements

81



Stakeholders are people

people
result
real result

- 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 !

82

Victims can be a big Risk

183

What are the Requirements for a Project ?

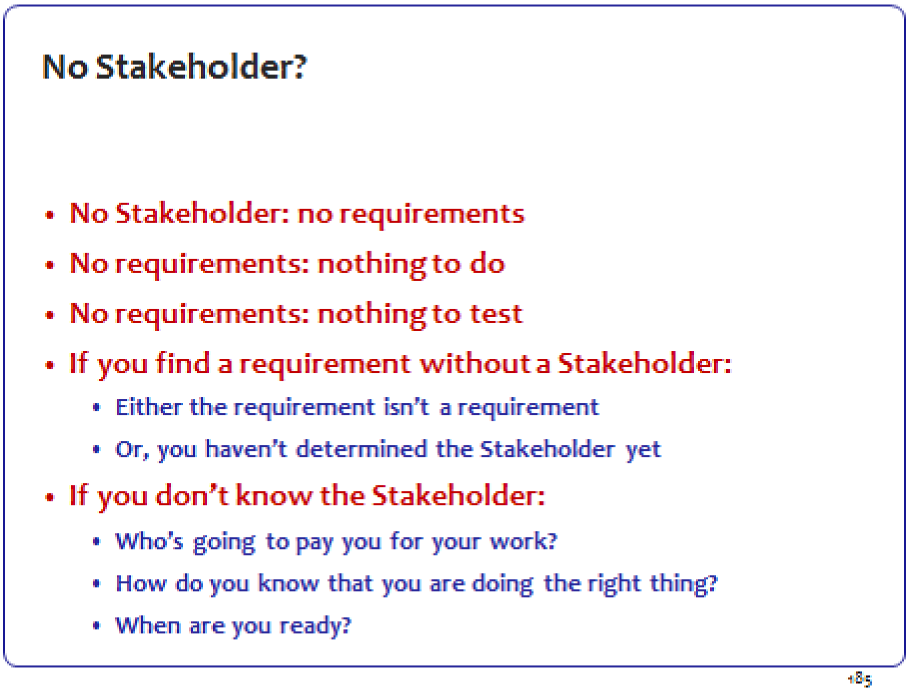
- Requirements are what the Stakeholders require

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 ?

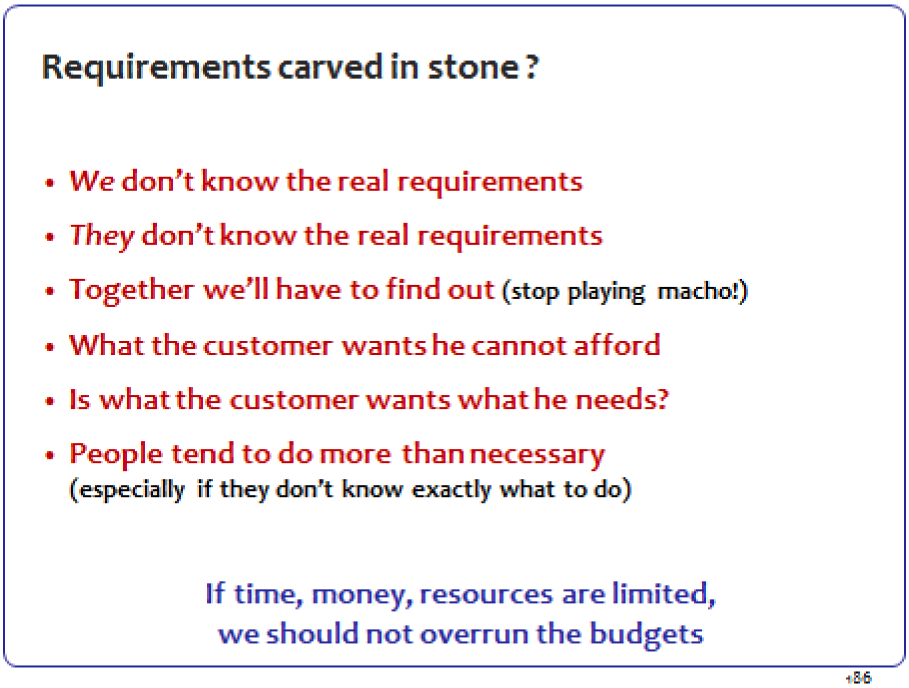
184



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?

185

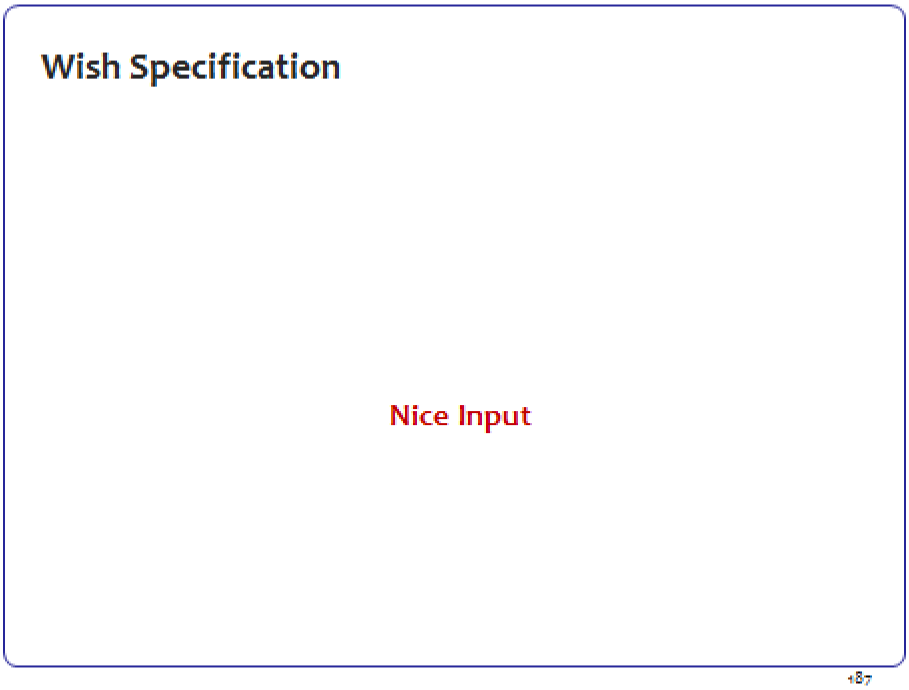


Requirements carved in stone ?

- *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

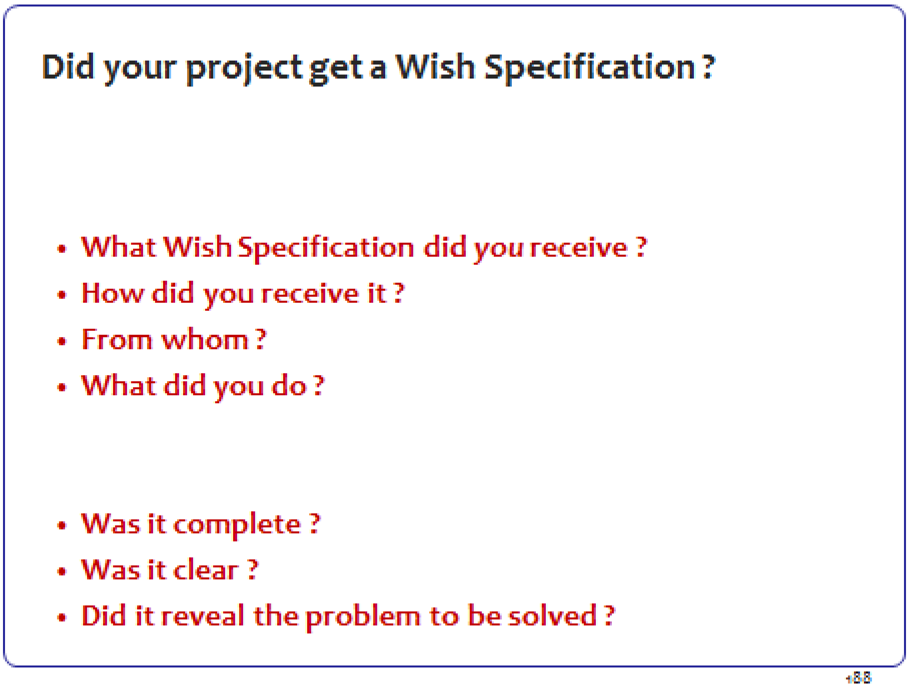
186



Wish Specification

Nice Input

187

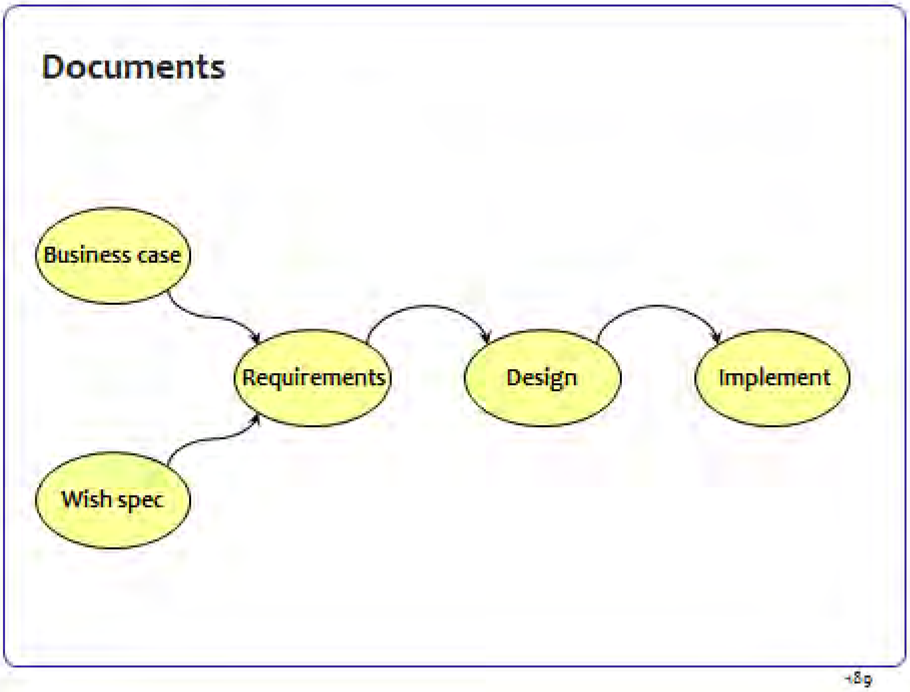


Did your project get a Wish Specification ?

- What Wish Specification did you receive ?
- How did you receive it ?
- From whom ?
- What did you do ?

- Was it complete ?
- Was it clear ?
- Did it reveal the problem to be solved ?

188



Documents

Business case

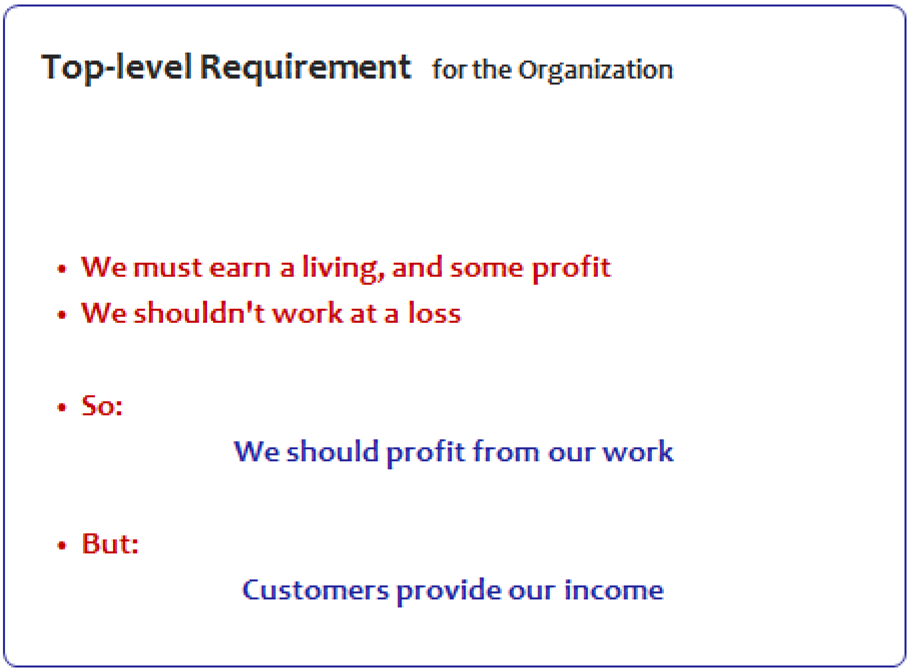
Wish spec

Requirements

Design

Implement

189



Top-level Requirement for the Organization

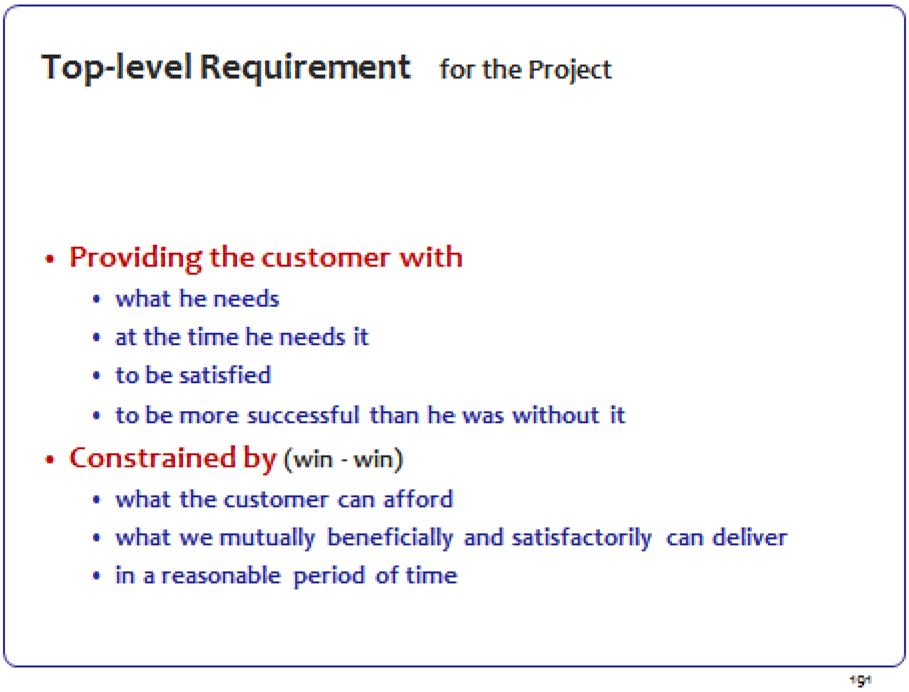
- We must earn a living, and some profit
- We shouldn't work at a loss
- So:

  We should profit from our work

- But:

  Customers provide our income

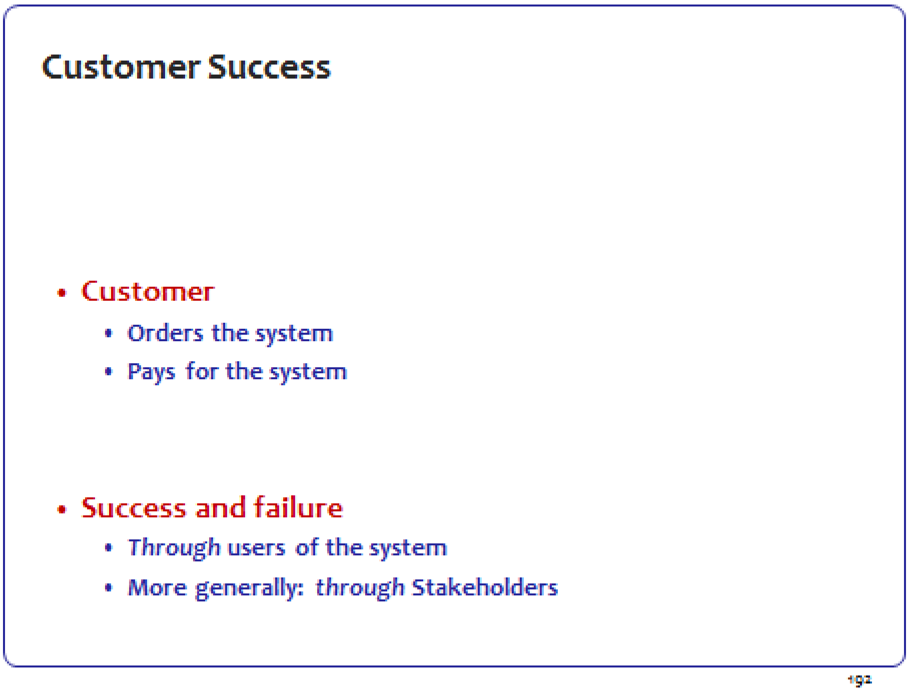
190



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

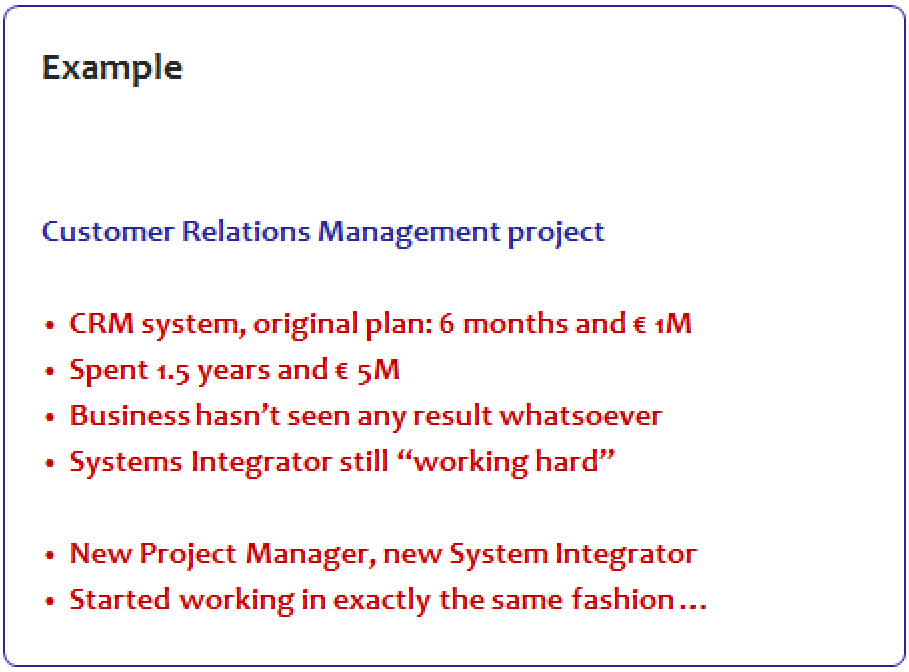
191



Customer Success

- Customer
  - Orders the system
  - Pays for the system
- Success and failure
  - *Through* users of the system
  - More generally: *through* Stakeholders

192



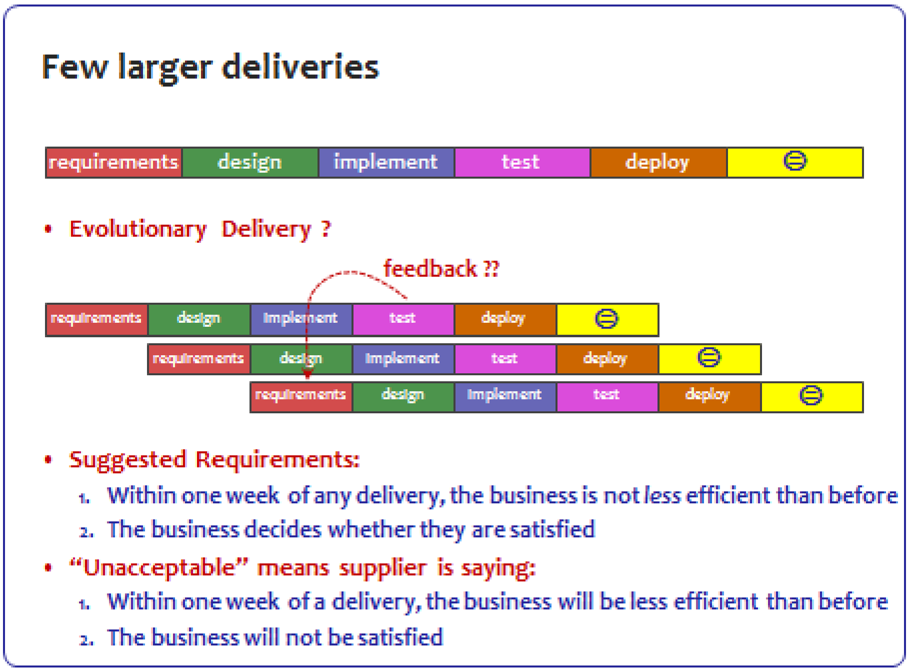
Example

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 …

193



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

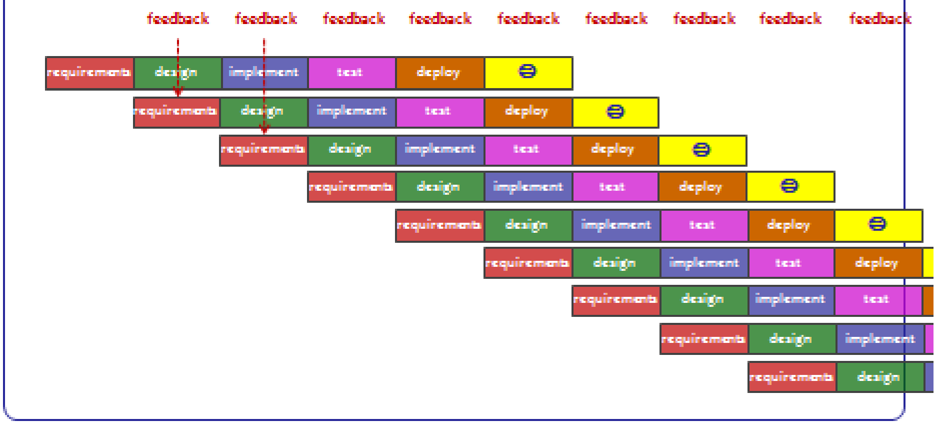
194

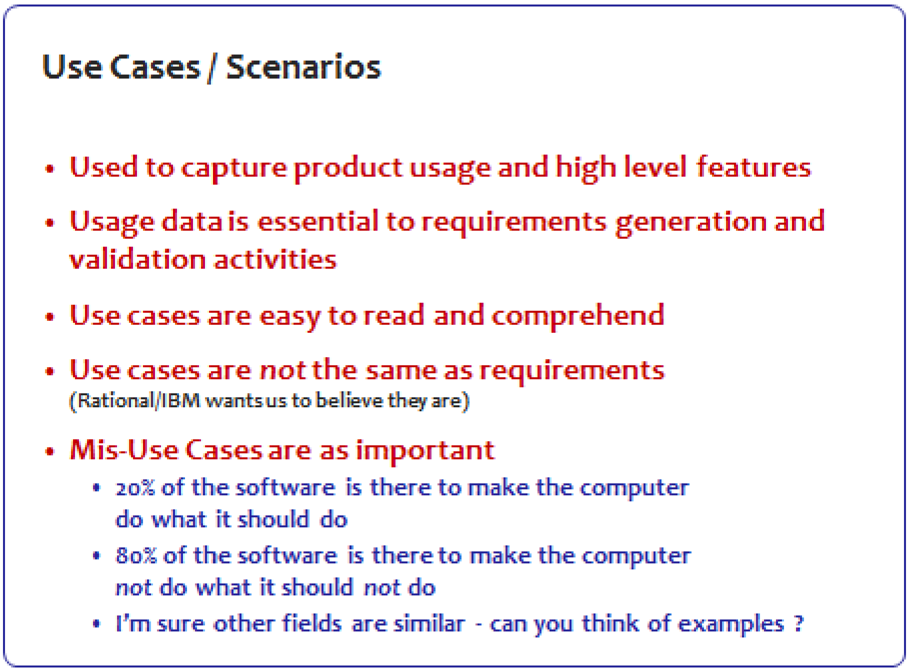
Peter Drucker

Quality in a service or product is not what you put into it

It is what the client or customer gets out of it

How about many small deliveries

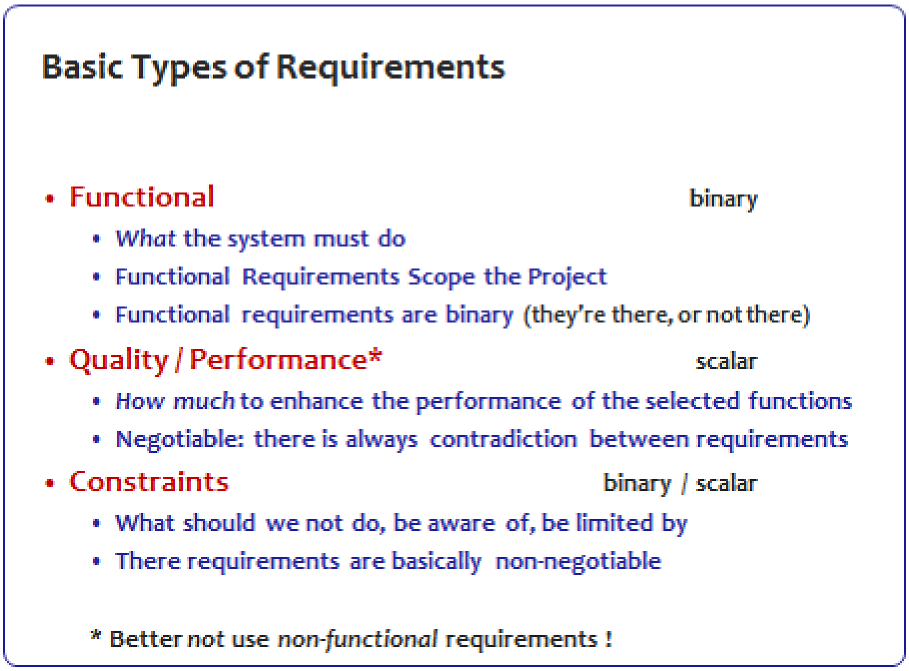




Use Cases / Scenarios

- Used to capture product usage and high level features
- Usage data is essential to requirements generation and validation activities
- Use cases are easy to read and comprehend
- Use cases are *not* the same as requirements
  (Rational/IBM wants us to believe they are)
- Mis-Use Cases are as important
  - 20% of the software is there to make the computer do what it should do
  - 80% of the software is there to make the computer not do what it should *not* do
  - I'm sure other fields are similar - can you think of examples ?

197

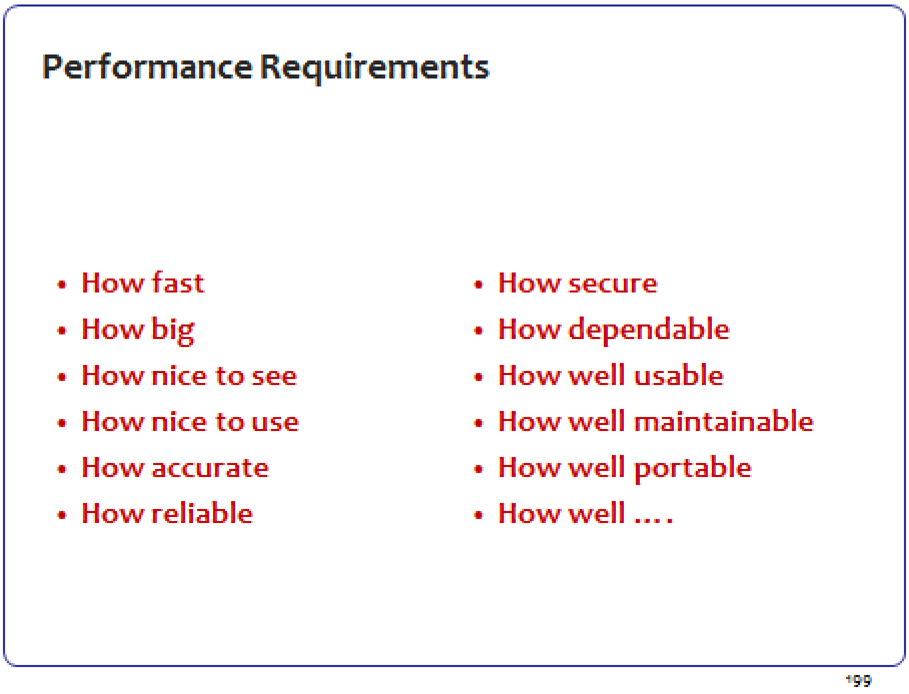


Basic Types of Requirements

- Functional binary
  - *What* the system must do
  - Functional Requirements Scope the Project
  - Functional requirements are binary (they're there, or not there)
- Quality / Performance* scalar
  - *How much* to enhance the performance of the selected functions
  - Negotiable: there is always contradiction between requirements
- Constraints binary / scalar
  - What should we not do, be aware of, be limited by
  - There requirements are basically non-negotiable

* Better *not* use *non-functional* requirements !

198

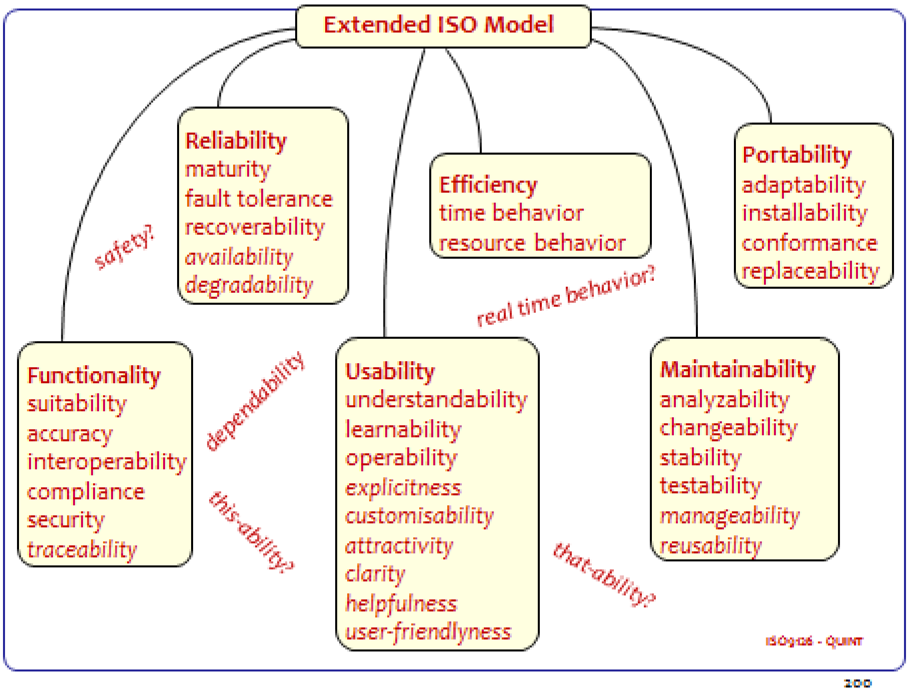


Performance Requirements

- How fast
- How big
- How nice to see
- How nice to use
- How accurate
- How reliable

- How secure
- How dependable
- How well usable
- How well maintainable
- How well portable
- How well ....

199



Extended ISO Model

Reliability
maturity
fault tolerance
recoverability
availability
degradability

Efficiency
time behavior
resource behavior

Portability
adaptability
installability
conformance
replaceability

Functionality
suitability
accuracy
interoperability
compliance
security
traceability

Usability
understandability
learnability
operability
explicitness
customisability
attractivity
clarity
helpfulness
user-friendlyness

Maintainability
analyzability
changeability
stability
testability
manageability
reusability

safety?

real time behavior?

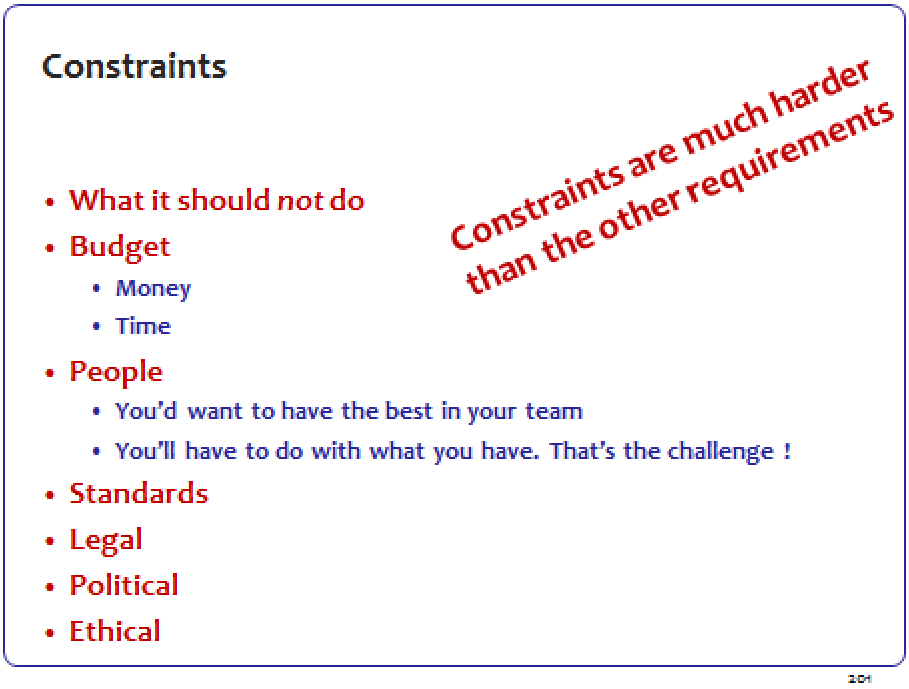
dependability

this-ability?

that-ability?

ISO9126 - QUINT

200

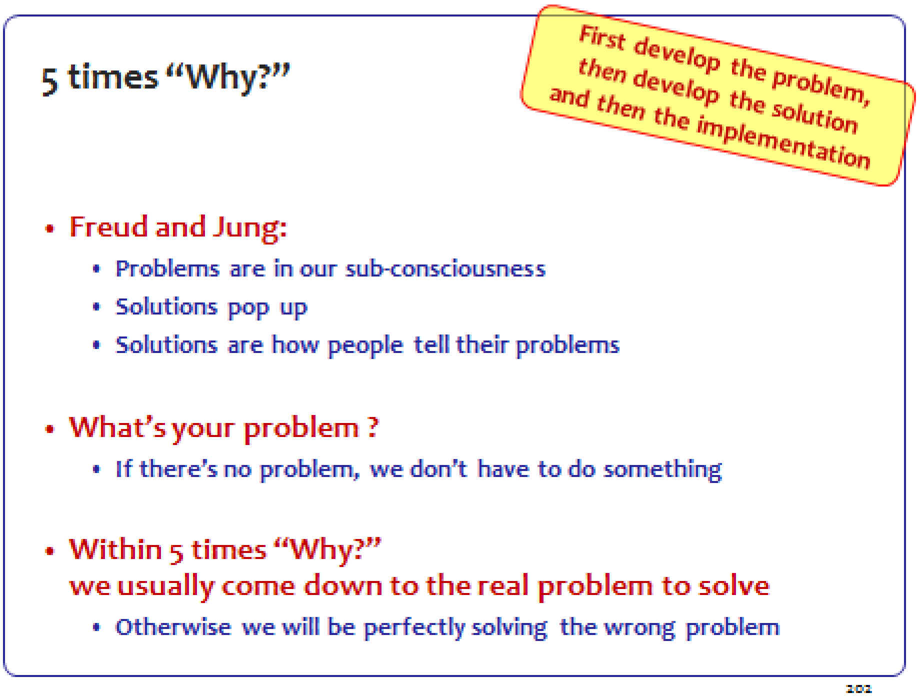


Constraints

Constraints are much harder than the other requirements

- What it should not do
- Budget
  - Money
  - Time
- People
  - You'd want to have the best in your team
  - You'll have to do with what you have. That's the challenge !
- Standards
- Legal
- Political
- Ethical

201

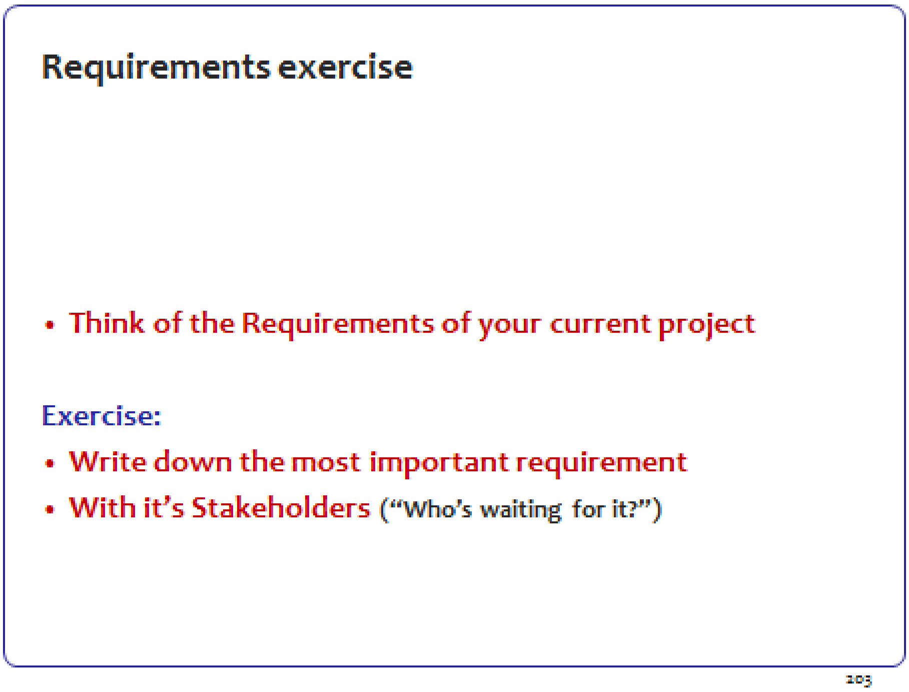


5 times "Why?"

First develop the problem,
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

202



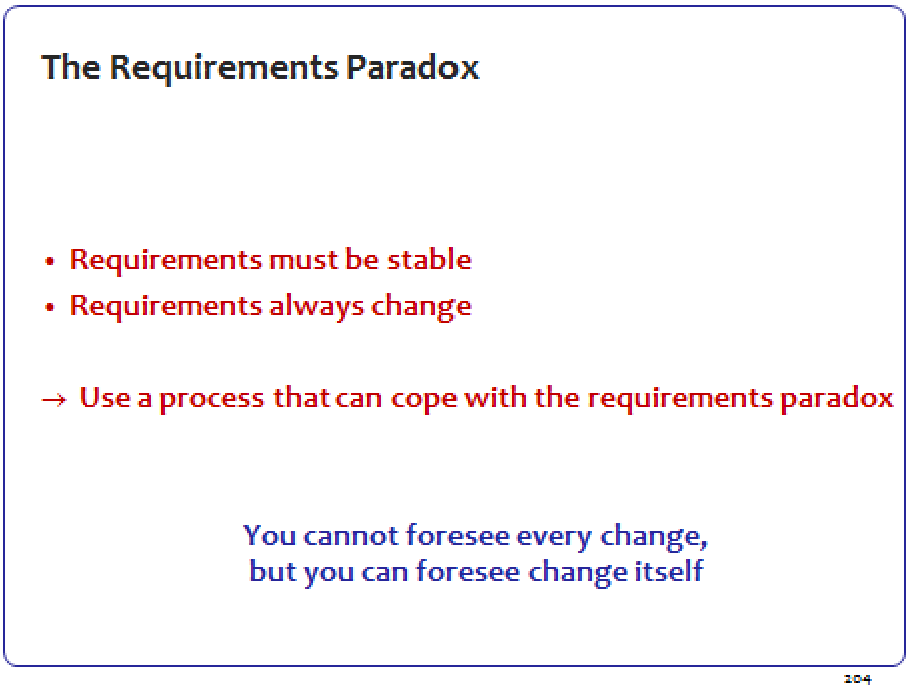
Requirements exercise

- Think of the Requirements of your current project

Exercise:

- Write down the most important requirement
- With it's Stakeholders ("Who's waiting for it?")

103



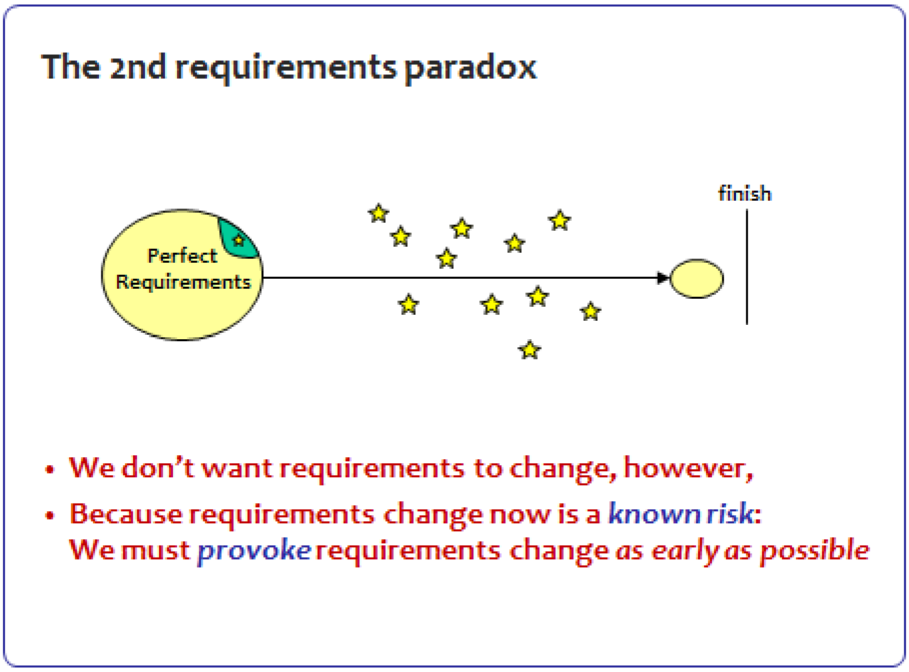
The Requirements Paradox

- Requirements must be stable
- Requirements always change

→ Use a process that can cope with the requirements paradox

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

104



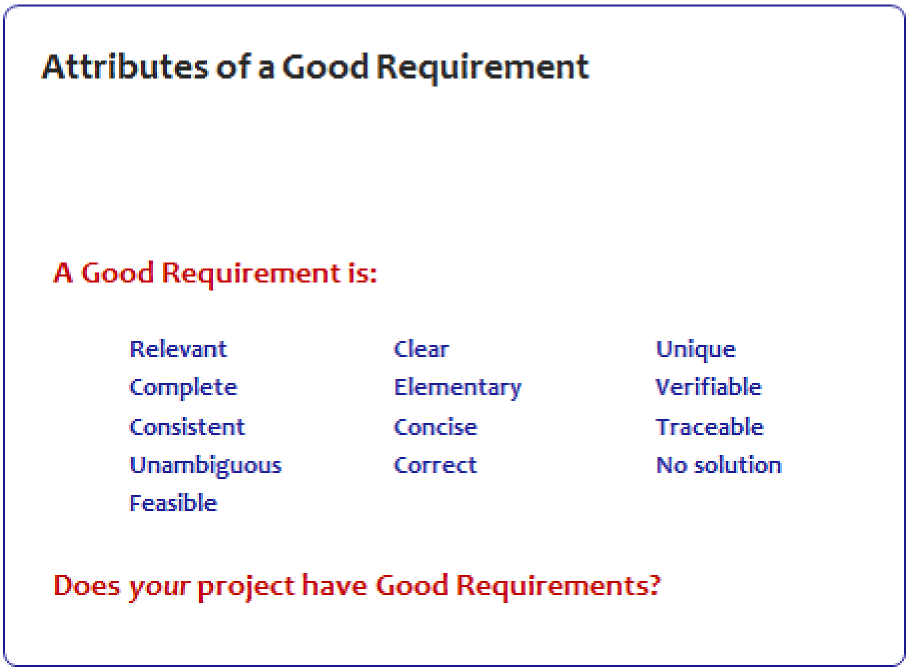
The 2nd requirements paradox

Perfect Requirements

finish

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

205



Attributes of a Good Requirement

A Good Requirement is:

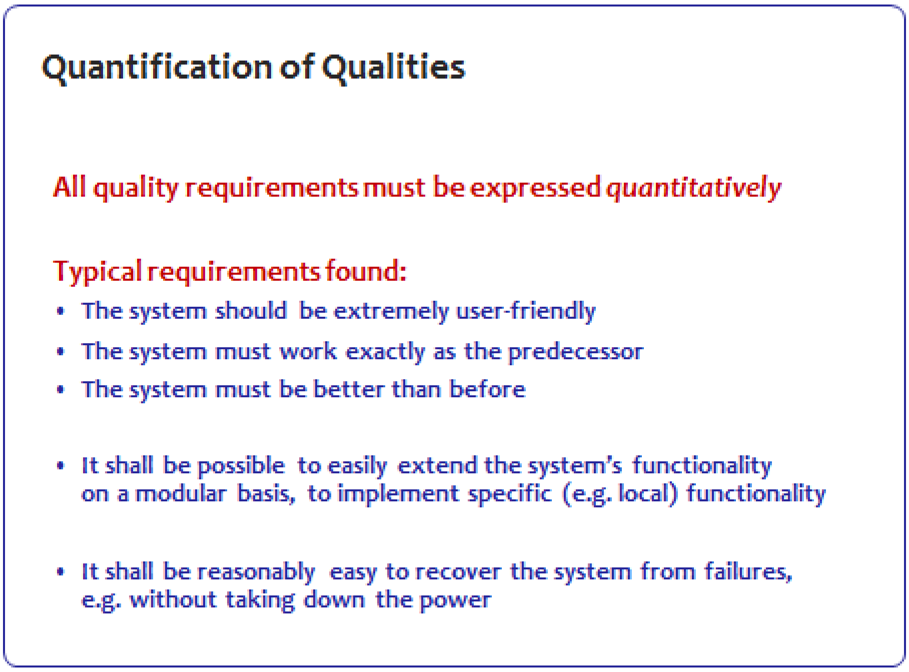
Relevant
Complete
Consistent
Unambiguous
Feasible

Clear
Elementary
Concise
Correct

Unique
Verifiable
Traceable
No solution

Does *your* project have Good Requirements?

206



Quantification of Qualities

All quality requirements must be expressed *quantitatively*

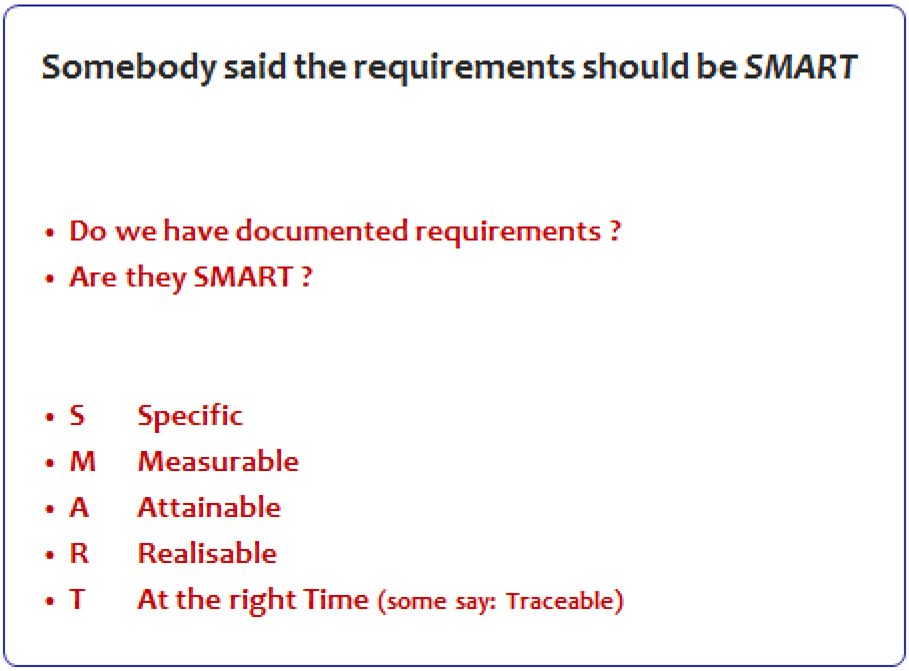
Typical requirements found:

- The system should be extremely user-friendly
- The system must work exactly as the predecessor
- The system must be better than before

- 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

207

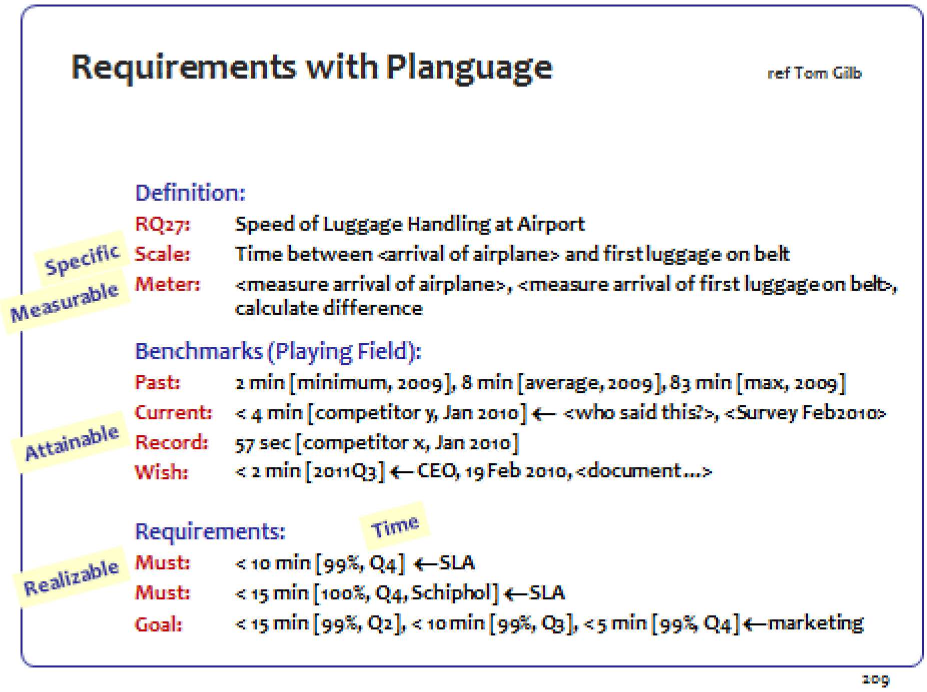


Somebody said the requirements should be *SMART*

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

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

208



Requirements with Planguage

ref Tom Gilb

Definition:

RQ27: Speed of Luggage Handling at Airport

Specific

Scale: Time between <arrival of airplane> and first luggage on belt

Measurable

Meter: <measure arrival of airplane>, <measure arrival of first luggage on belt>, calculate difference

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>

Attainable

Record: 57 sec [competitor x, Jan 2010]

Wish: < 2 min [2011Q3] ← CEO, 19 Feb 2010, <document…>

Requirements: Time

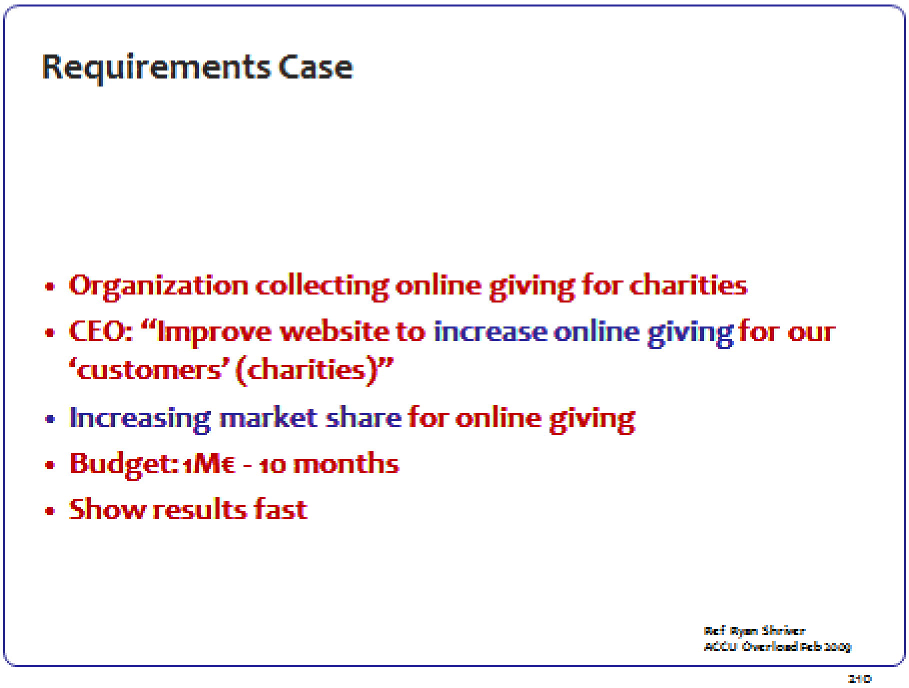
Realizable

Must: < 10 min [99%, Q4] ← SLA

Must: < 15 min [100%, Q4, Schiphol] ← SLA

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

209

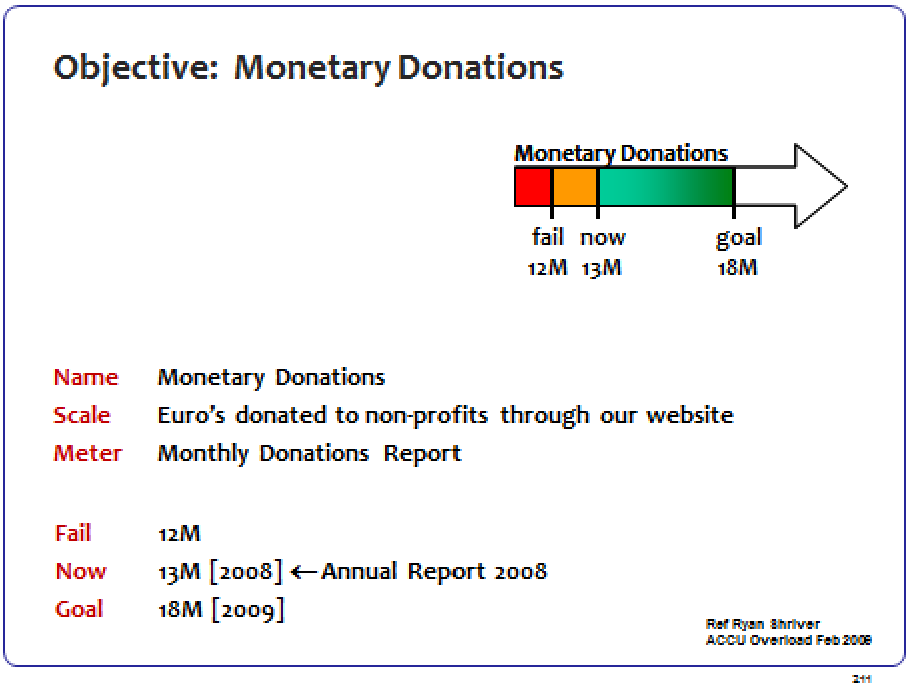


Requirements Case

- 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

210



Objective: Monetary Donations

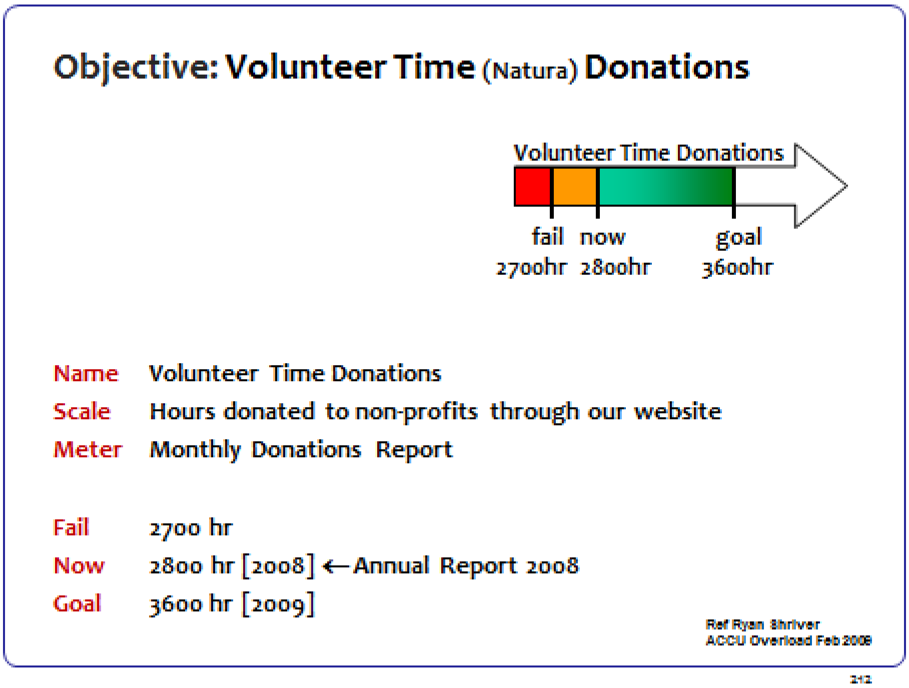
Monetary Donations
fail
now
goal
12M
13M
18M

Name Monetary Donations
Scale Euro’s donated to non-profits through our website
Meter Monthly Donations Report

Fail 12M
Now 13M [2008] ← Annual Report 2008
Goal 18M [2009]

Ref Ryan Shriver
ACCU Overload Feb 2009

211



Objective: Volunteer Time (Natura) Donations

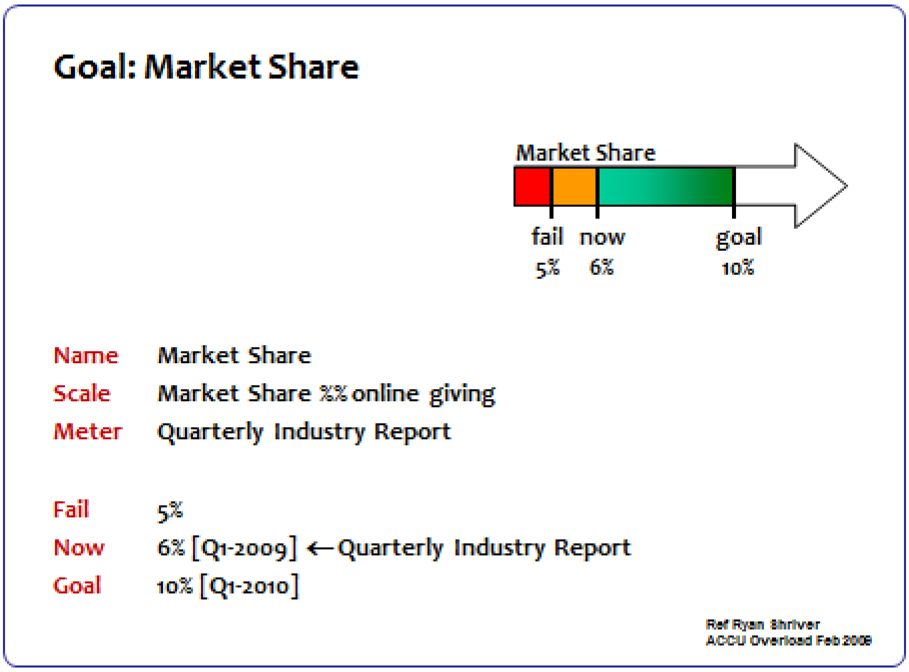
Volunteer Time Donations
fail
now
goal
2700hr
2800hr
3600hr

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 2009

212



Goal: Market Share

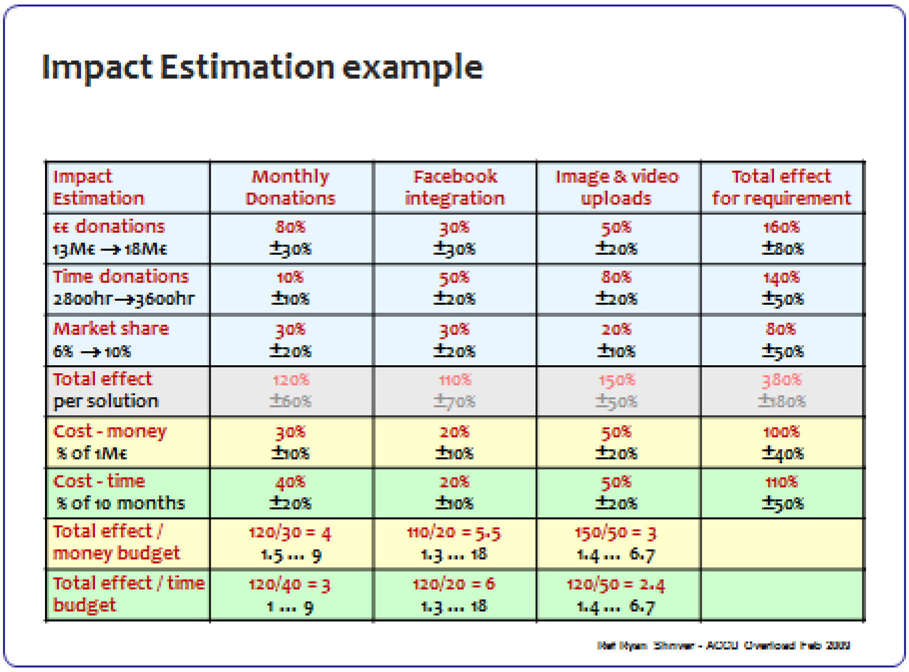
Market Share
fail
5%
now
6%
goal
10%

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 2009

213

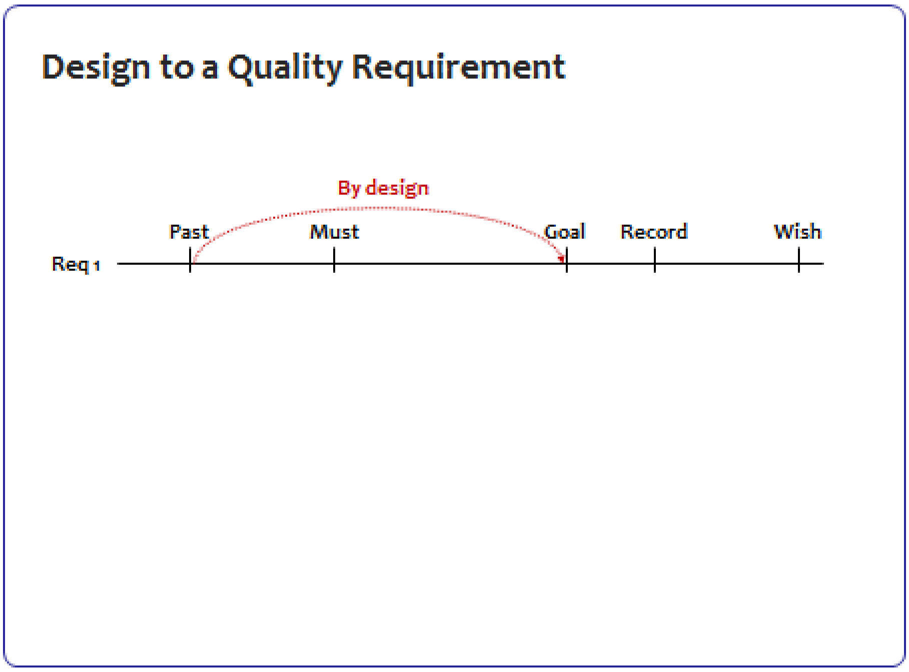


Impact Estimation example

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	

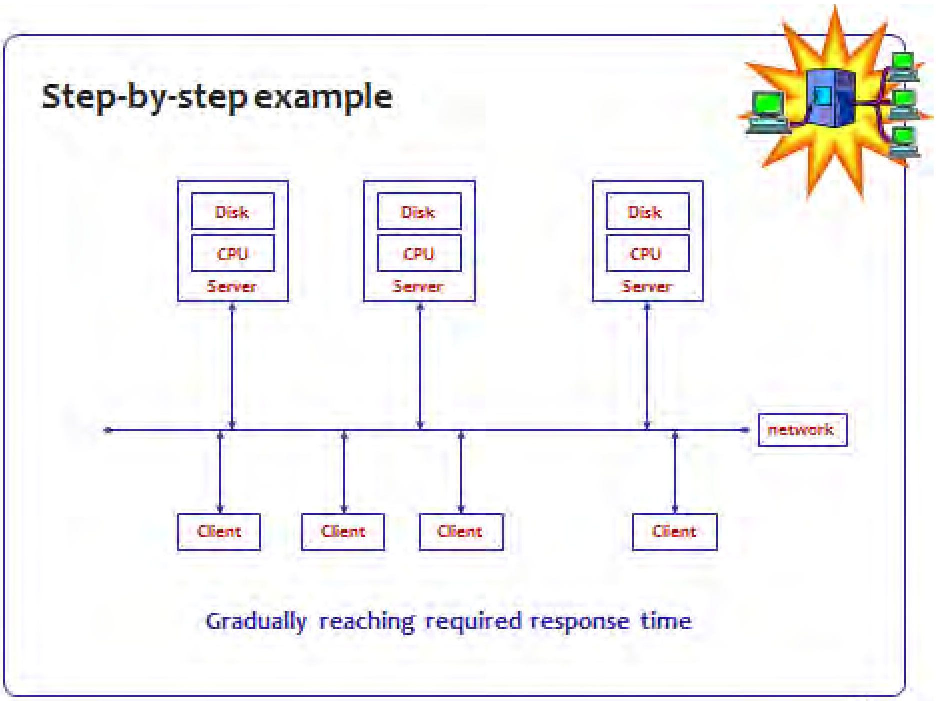
Ref Ryan Shriver - ACCU Overload Feb 2009

214



Design to a Quality Requirement

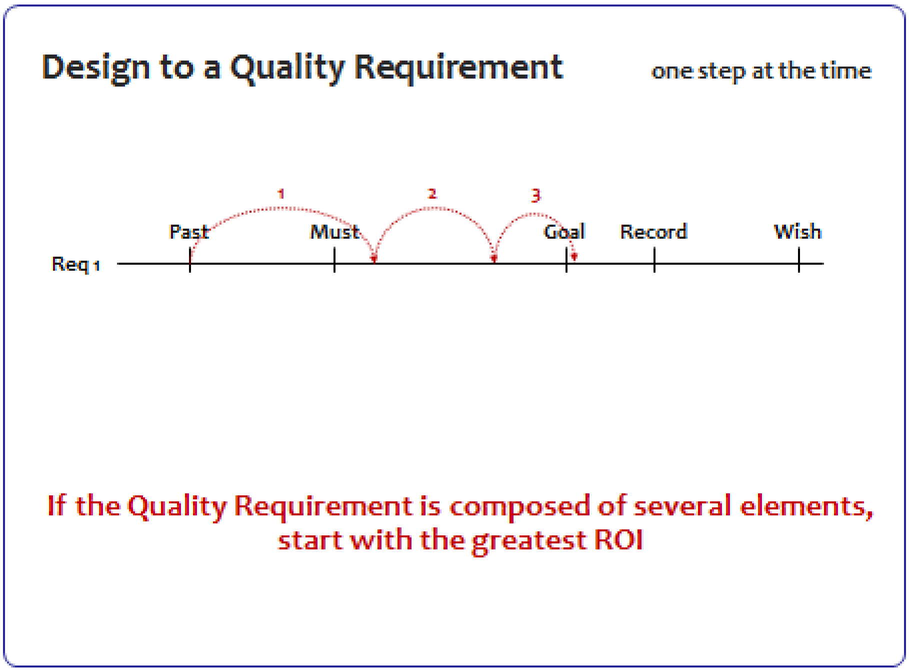
215



Step-by-step example

Gradually reaching required response time

216

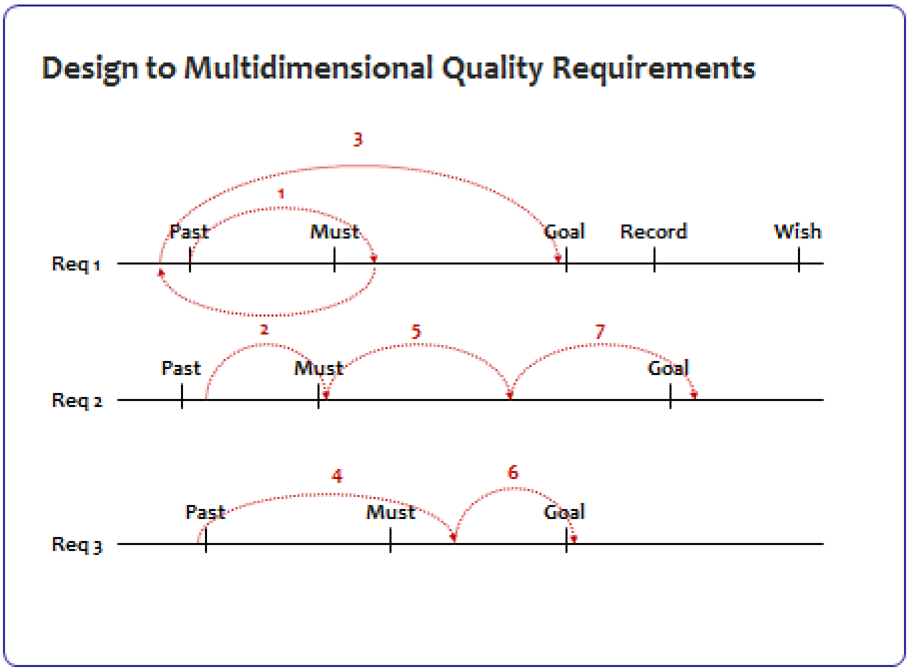


Design to a Quality Requirement

one step at the time

If the Quality Requirement is composed of several elements, start with the greatest ROI

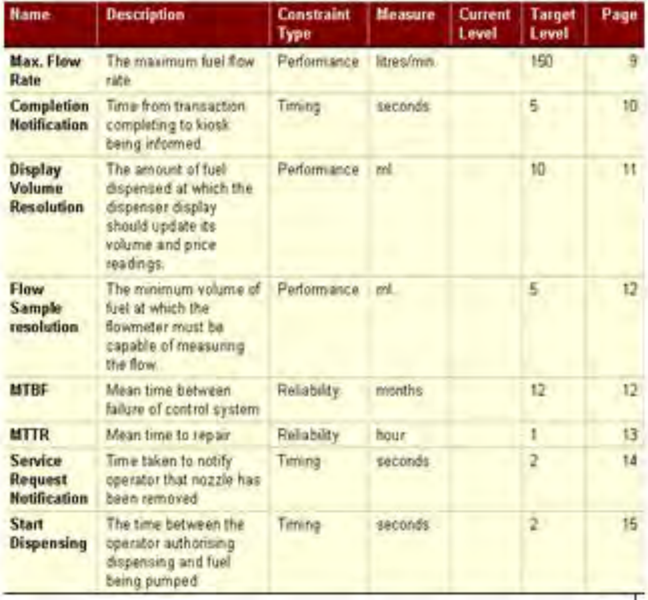
217



Design to Multidimensional Quality Requirements

218

Quantified Requirements



Name	Description	Constraint Type	Measure	Current Level	Target Level	Page
Max. Flow Rate	The maximum fuel flow rate	Performance	litres/min		150	9
Completion Notification	Time from transaction completing to kiosk being informed	Timing	seconds		5	10
Display Volume Resolution	The amount of fuel dispensed at which the dispenser display should update its volume and price readings.	Performance	ml		10	11
Flow Sample resolution	The minimum volume of fuel at which the flowmeter must be capable of measuring the flow	Performance	ml.		5	12
MTBF	Mean time between failure of control system	Reliabilty	months		12	12
MTTR	Mean time to repair	Reliabilty	hour		1	13
Service Request Notification	Time taken to notify operator that nozzle has been removed	Timing	seconds		2	14
Start Dispensing	The time between the operator authorising dispensing and fuel being pumped	Timing	seconds		2	15

219

Examples of Scales

Environmental Noise
dBA at 1.0 meter

System Security
Time required to <break into the system> by <type of people>

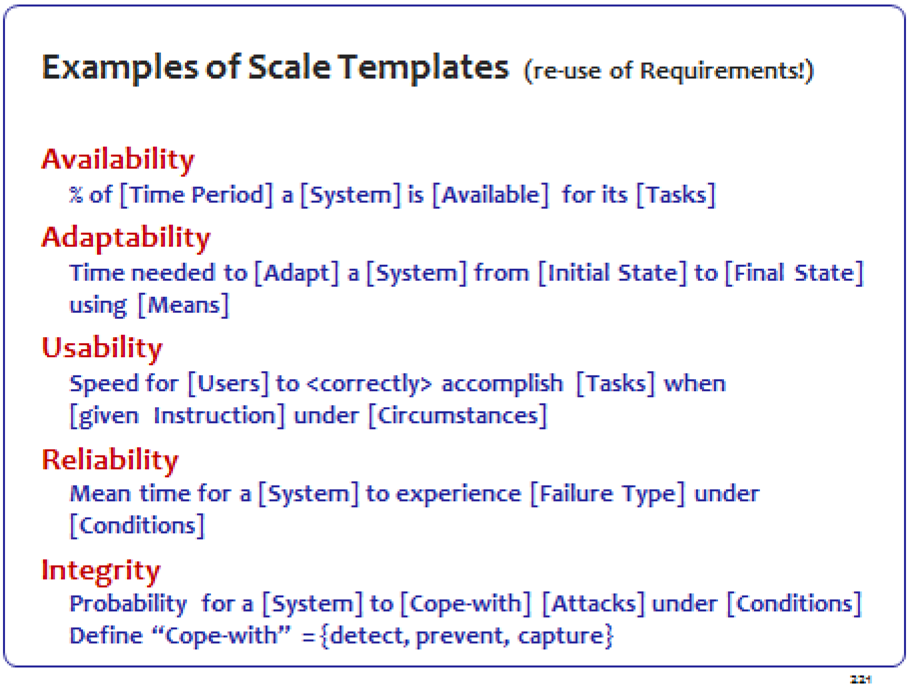
Software Maintainability
Time needed from <request for change> to <availability of change>

System Reliability
The Mean Time Before Failure (MTBF) of the system

System Learnability
Average time for <Novices> to become <Proficient> at a defined set of tasks

Productivity
Time needed by <type of people> to perform a <task>

220



Examples of Scale Templates (re-use of Requirements!)

Availability
% of [Time Period] a [System] is [Available] for its [Tasks]

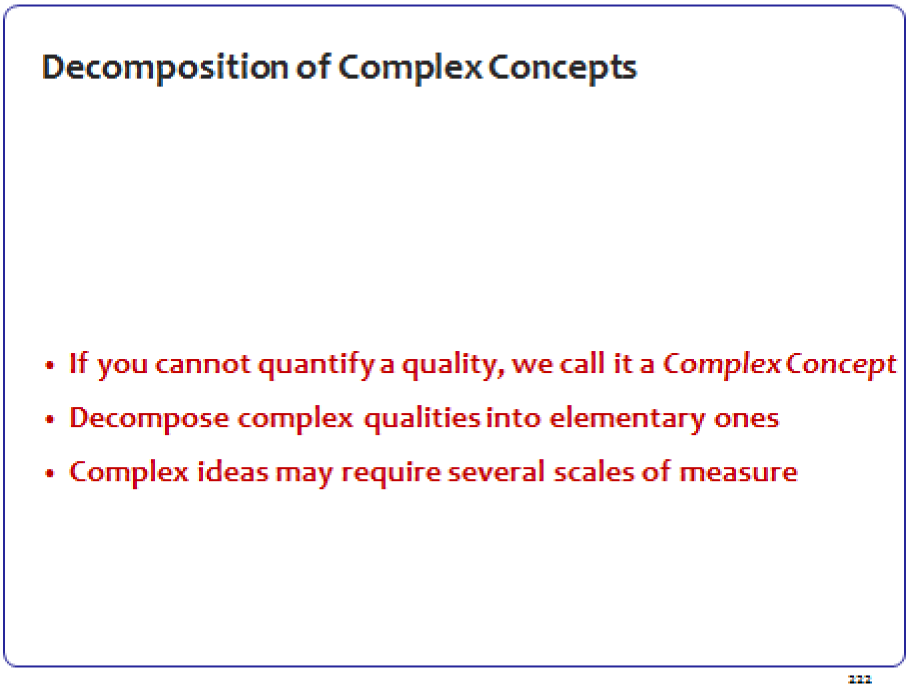
Adaptability
Time needed to [Adapt] a [System] from [Initial State] to [Final State] using [Means]

Usability
Speed for [Users] to <correctly> accomplish [Tasks] when [given Instruction] under [Circumstances]

Reliability
Mean time for a [System] to experience [Failure Type] under [Conditions]

Integrity
Probability for a [System] to [Cope-with] [Attacks] under [Conditions]
Define “Cope-with” = {detect, prevent, capture}

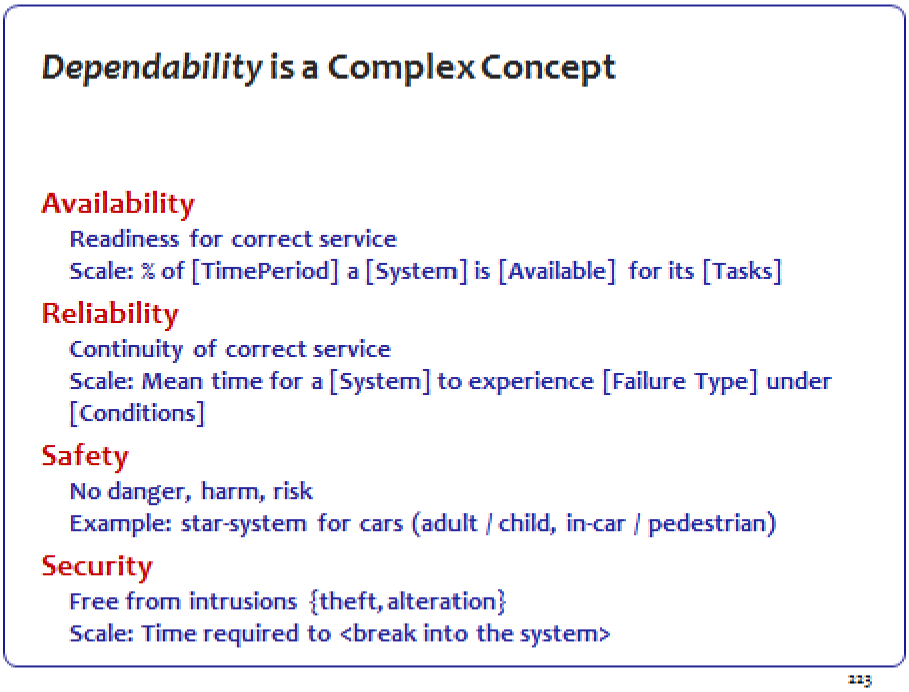
221



Decomposition of Complex Concepts

- If you cannot quantify a quality, we call it a *Complex Concept*
- Decompose complex qualities into elementary ones
- Complex ideas may require several scales of measure

222



Dependability is a Complex Concept

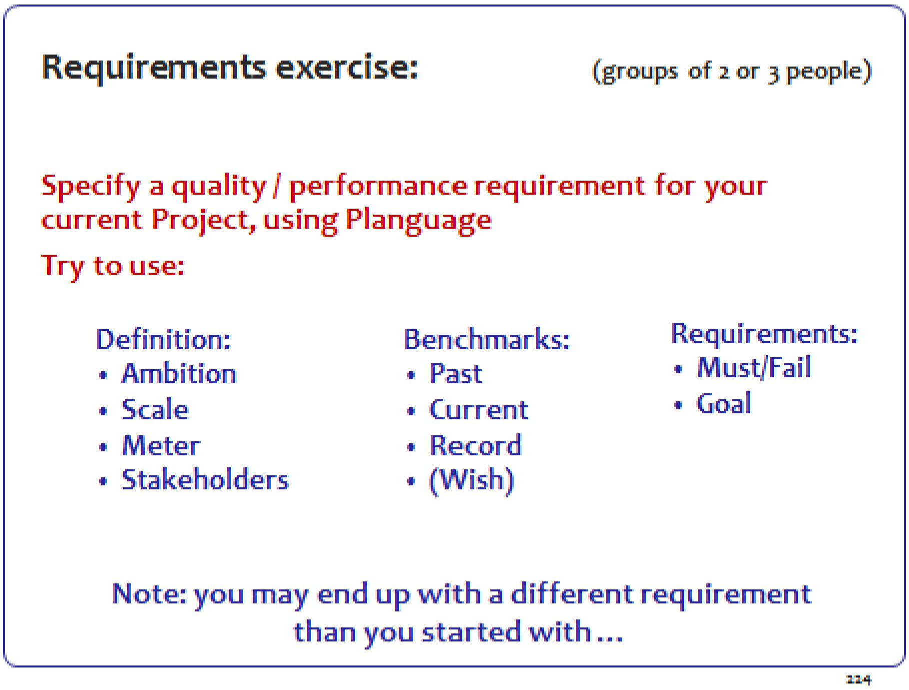
Availability
Readiness for correct service
Scale: % of [TimePeriod] a [System] is [Available] for its [Tasks]

Reliability
Continuity of correct service
Scale: Mean time for a [System] to experience [Failure Type] under [Conditions]

Safety
No danger, harm, risk
Example: star-system for cars (adult / child, in-car / pedestrian)

Security
Free from intrusions {theft, alteration}
Scale: Time required to <break into the system>

223



Requirements exercise:

(groups of 2 or 3 people)

Specify a quality / performance requirement for your current Project, using Planguage

Try to use:

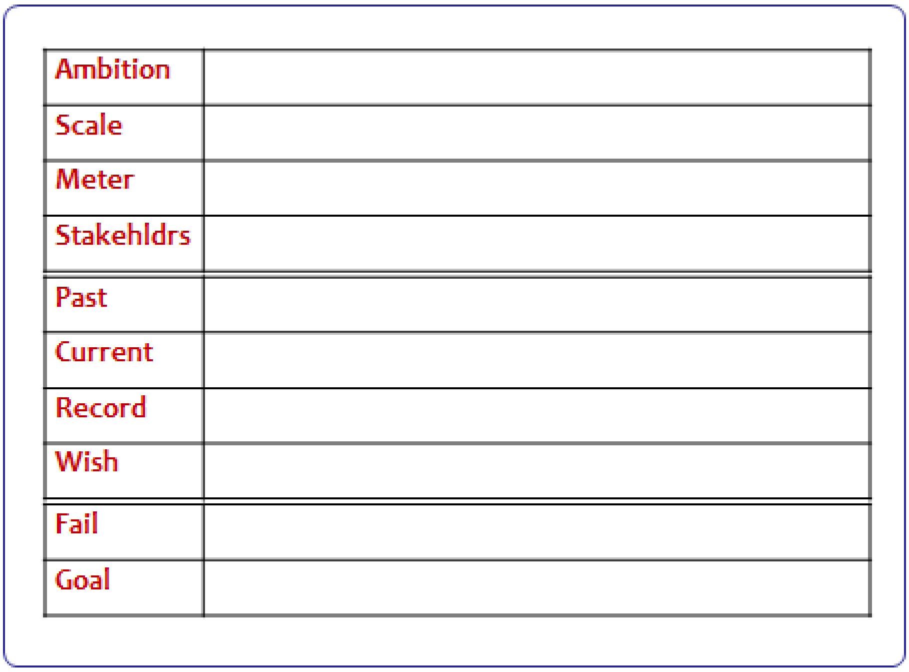
Definition:
- Ambition
- Scale
- Meter
- Stakeholders

Benchmarks:
- Past
- Current
- Record
- (Wish)

Requirements:
- Must/Fail
- Goal

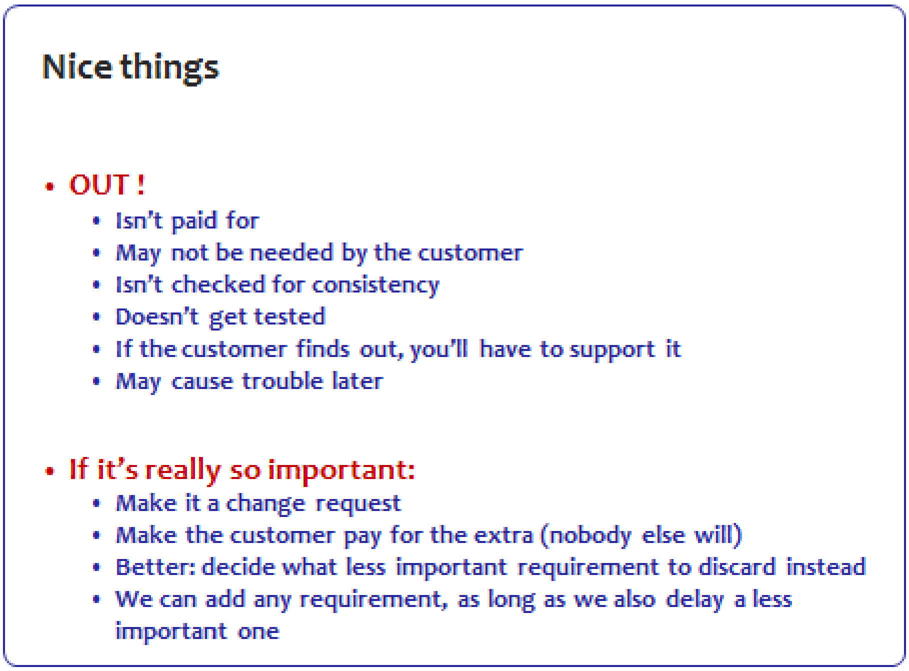
Note: you may end up with a different requirement than you started with ...

224



Ambition	
Scale	
Meter	
Stakehldrs	
Past	
Current	
Record	
Wish	
Fail	
Goal	

225



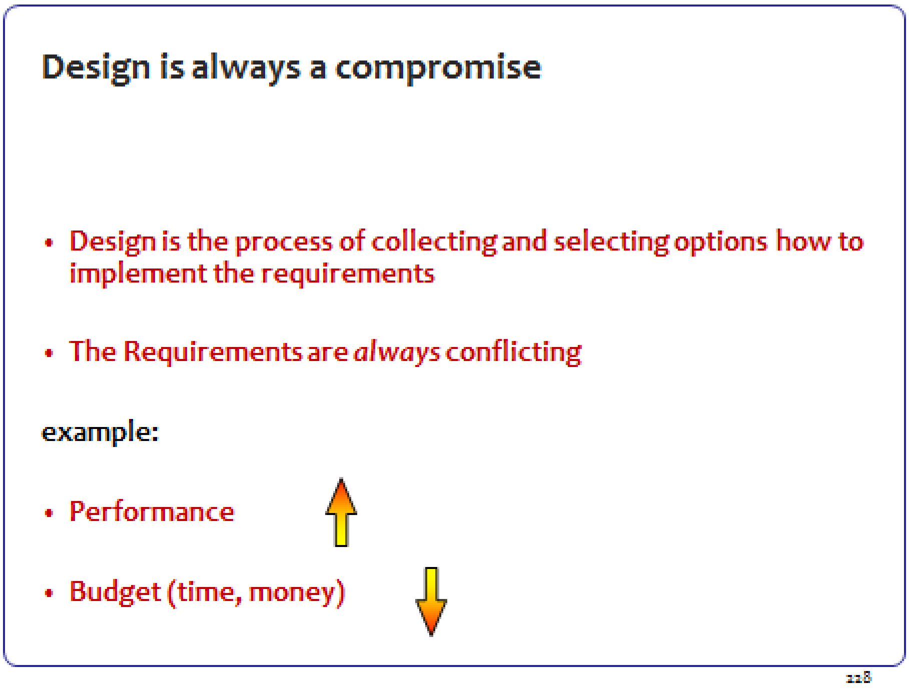
Nice things

- OUT !
  - Isn't paid for
  - May not be needed by the customer
  - Isn't checked for consistency
  - Doesn't get tested
  - If the customer finds out, you'll have to support it
  - May cause trouble later
- If it's really so important:
  - Make it a change request
  - Make the customer pay for the extra (nobody else will)
  - Better: decide what less important requirement to discard instead
  - We can add any requirement, as long as we also delay a less important one

226

Architecture & Design

227



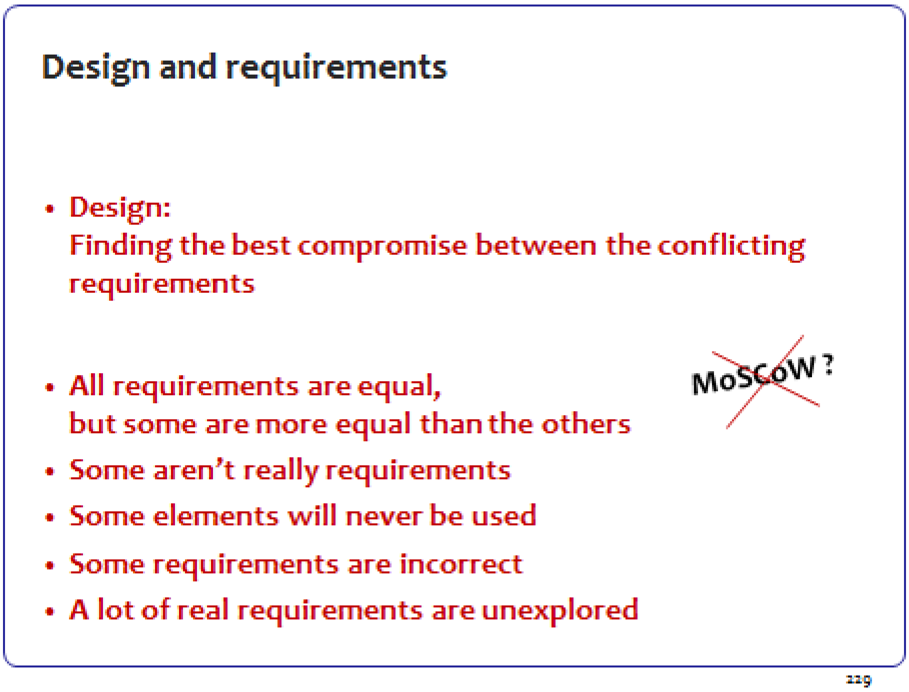
Design is always a compromise

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

example:

- Performance ↑
- Budget (time, money) ↓

228

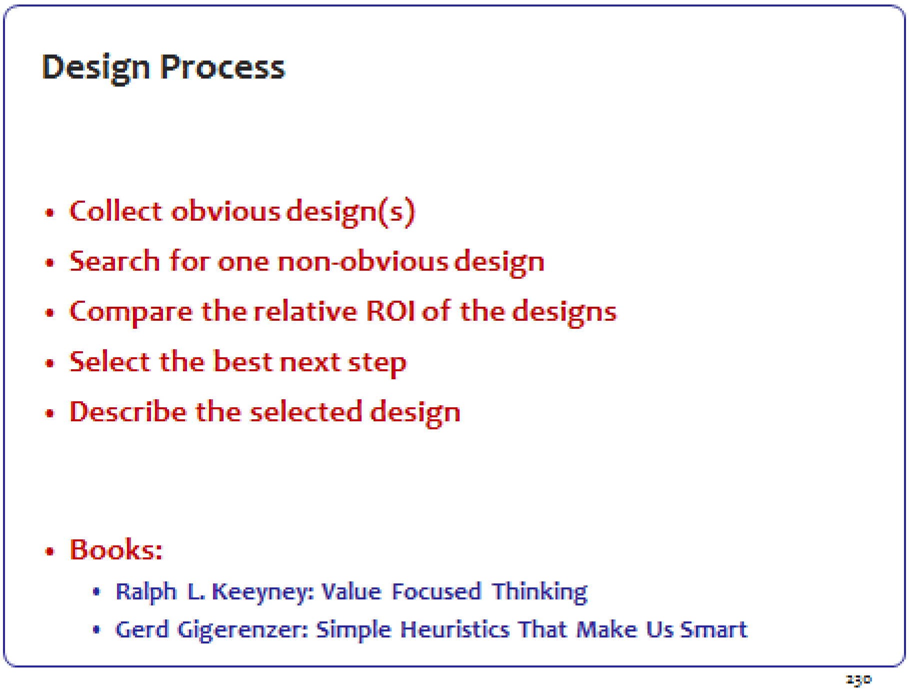


Design and requirements

- 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~~ ?

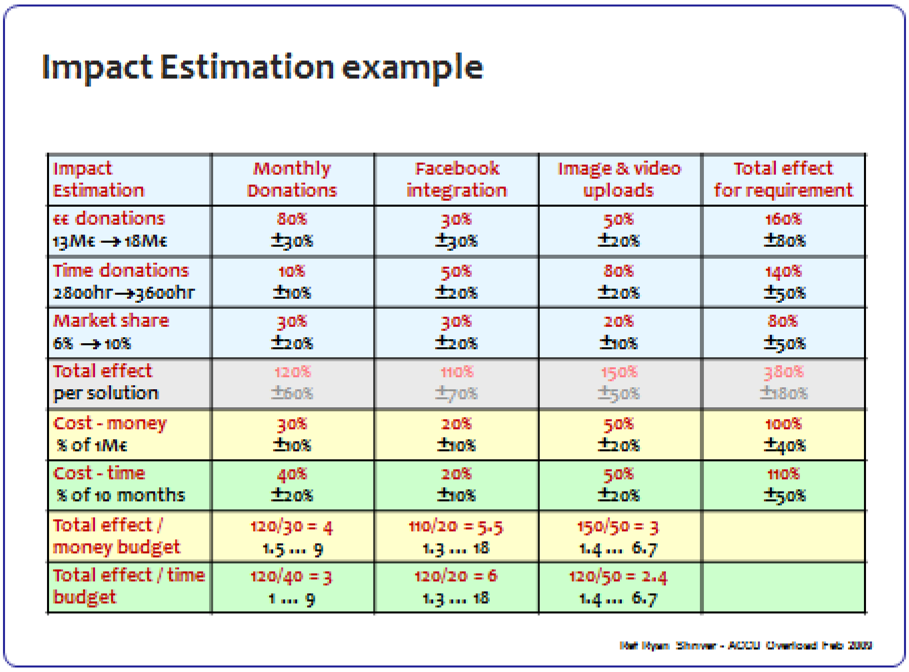
229



Design Process

- Collect obvious design(s)
- Search for one non-obvious design
- Compare the relative ROI of the designs
- Select the best next step
- Describe the selected design
- Books:
  - Ralph L. Keeyney: Value Focused Thinking
  - Gerd Gigerenzer: Simple Heuristics That Make Us Smart

230

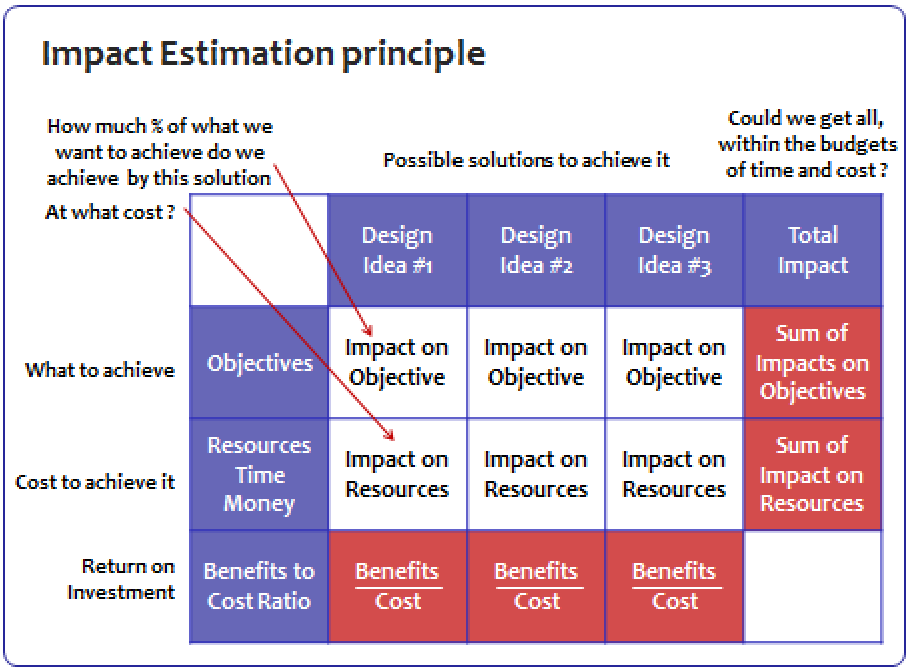


Impact Estimation example

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

231



Impact Estimation principle

How much % of what we want to achieve do we achieve by this solution

At what cost ?

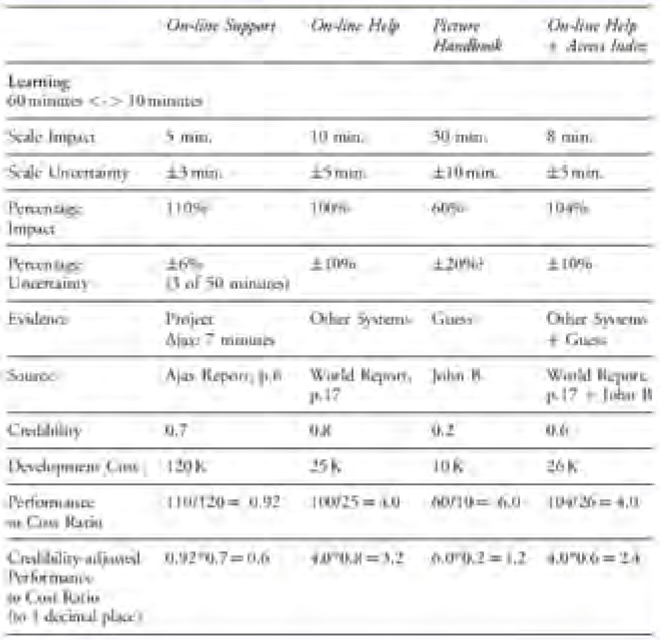
Possible solutions to achieve it

Could we get all, within the budgets of time and cost ?

		Design Idea #1	Design Idea #2	Design Idea #3	Total Impact
What to achieve	Objectives	Impact on Objective	Impact on Objective	Impact on Objective	Sum of Impacts on Objectives
Cost to achieve it	Resources Time Money	Impact on Resources	Impact on Resources	Impact on Resources	Sum of Impact on Resources
Return on Investment	Benefits to Cost Ratio	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	

232

Impact Estimation



	On-line Support	*On-line Help*	*Picture Handbook*	*On-line Help + Access Index*
Learning 60 minutes <-> 10 minutes				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3 min.	±5 min.	±10 min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%?	±10%
Evidence	Project Ajax: 7 minutes	Other Systems	Guess	Other Systems + Guess
Source	Ajax Report, p.6	World Report, p.17	John B	World Report, p.17 + John B
Credibility	0.7	0.8	0.2	0.6
Development Cost	120 K	25 K	10 K	26 K
Performance to Cost Ratio	110/120 = 0.92	100/25 = 4.0	60/10 = 6.0	104/26 = 4.0
Credibility-adjusted Performance to Cost Ratio (to 1 decimal place)	0.92*0.7 = 0.6	4.0*0.8 = 3.2	6.0*0.2 = 1.2	4.0*0.6 = 2.4

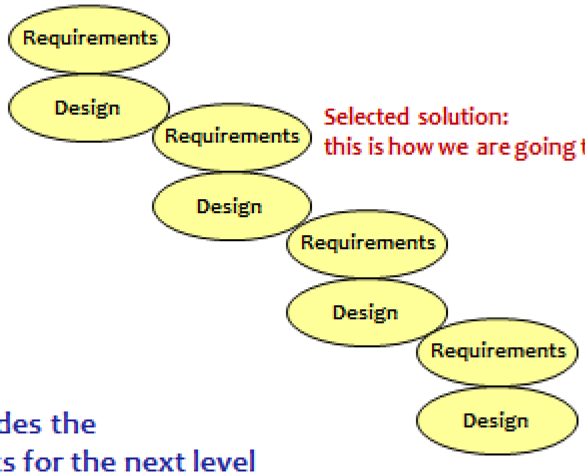
ref
Tom Gilb
Competitive Engineering

233

No Design in the requirements, but ...

Needs:
what do we need

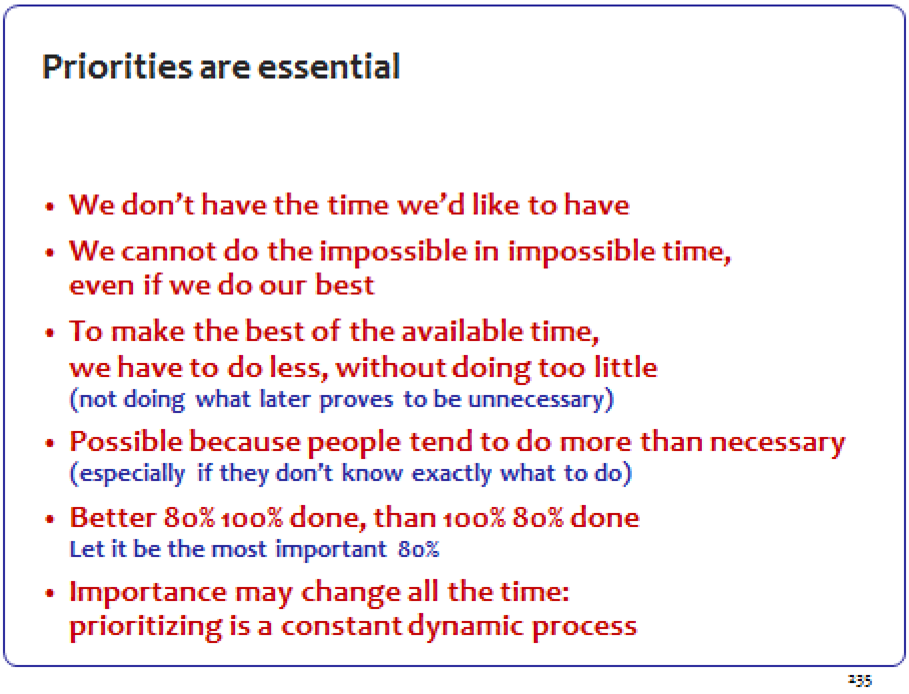
Options:
how can we do it



Selected solution:
this is how we are going to do it

Design provides the
Requirements for the next level

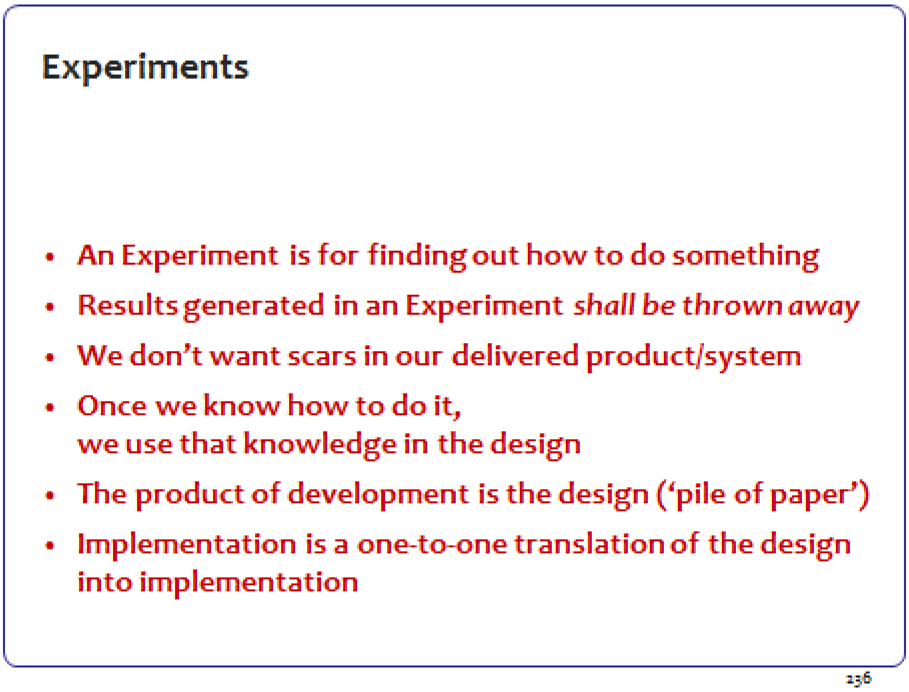
234



Priorities are essential

- We don't have the time we'd like to have
- We cannot do the impossible in impossible time, even if we do our best
- To make the best of the available time, we have to do less, without doing too little
  (not doing what later proves to be unnecessary)
- Possible because people tend to do more than necessary
  (especially if they don't know exactly what to do)
- Better 80% 100% done, than 100% 80% done
  Let it be the most important 80%
- Importance may change all the time: prioritizing is a constant dynamic process

235



Experiments

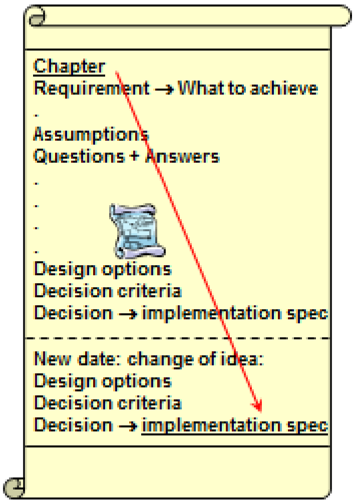
- An Experiment is for finding out how to do something
- Results generated in an Experiment *shall be thrown away*
- We don't want scars in our delivered product/system
- Once we know how to do it, we use that knowledge in the design
- The product of development is the design ('pile of paper')
- Implementation is a one-to-one translation of the design into implementation

236

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

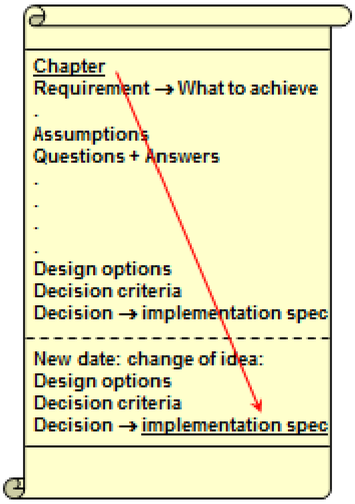
New date: change of idea:
Design options
Decision criteria
Decision → implementation spec

237

ProcessLog

(department / organization level)

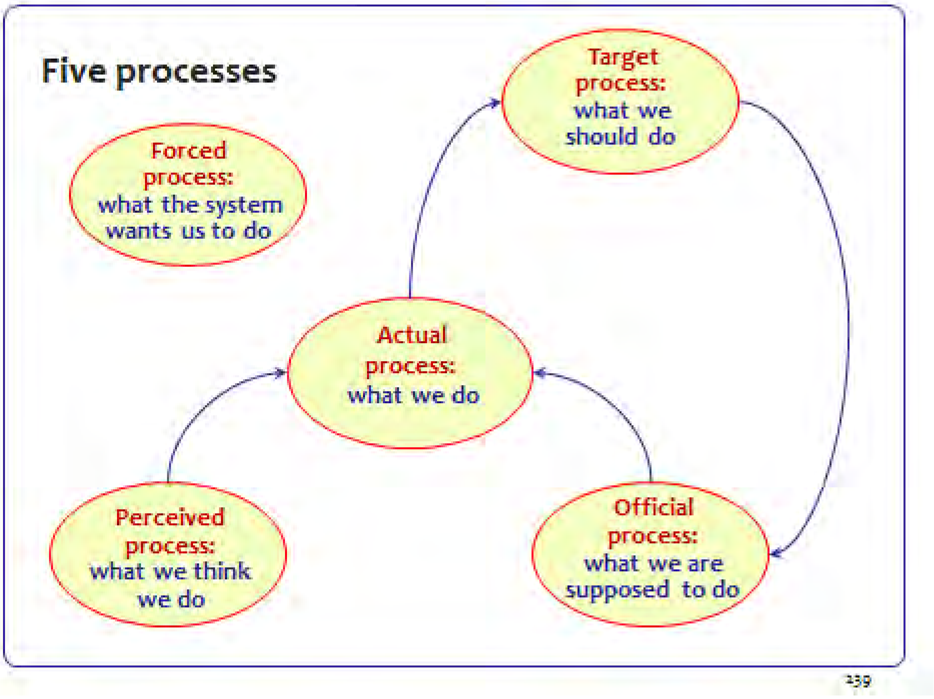
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Graphics (drawings)
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Known techniques
  - Choices + reasoning :
    - If rejected: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

238



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

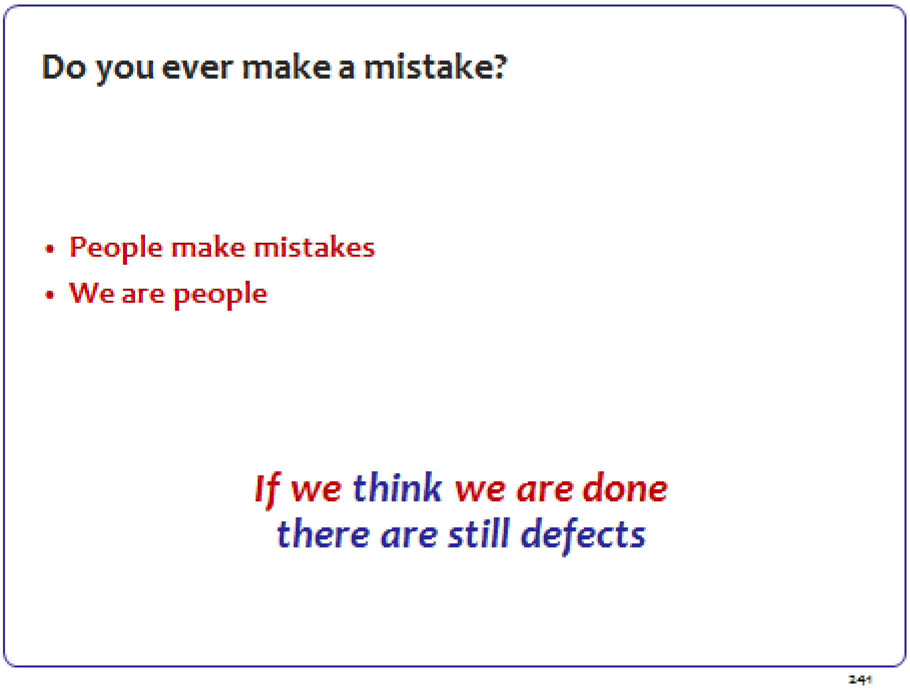
139

Verification & Validation

Testing

QA

240



Do you ever make a mistake?

- People make mistakes
- We are people

If we think we are done
there are still defects

241

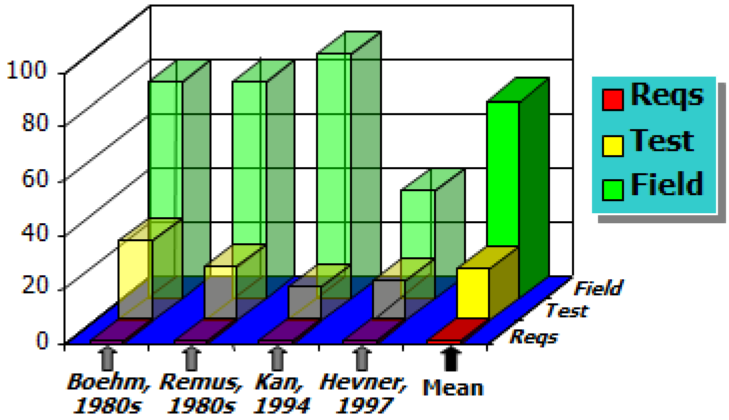
Costs of defects

The longer a defect stays in the system, the more it costs to find and repair

242

Cost of Requirements Defects

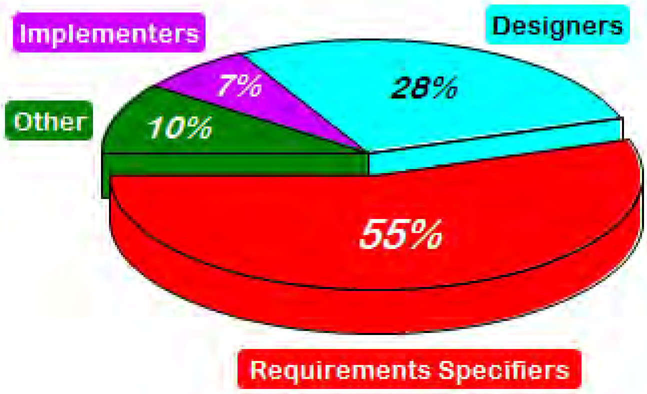
The longer a defect stays in the system, the more it costs to repair



DM

243

Typical Defect Injectors (cost breakdown)

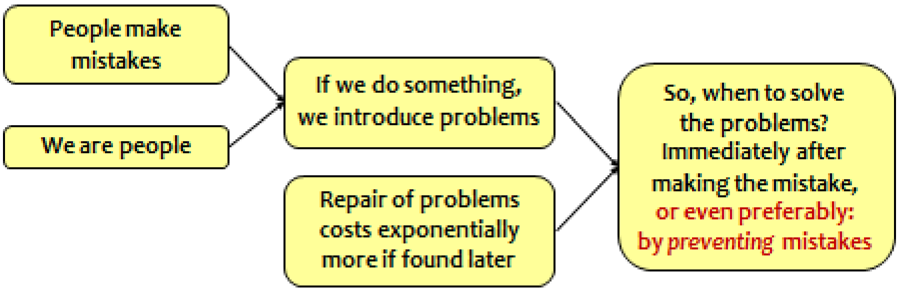


After Bender Associates, *1996*

DM

244

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

245

Do we deliver Zero Defect products ?

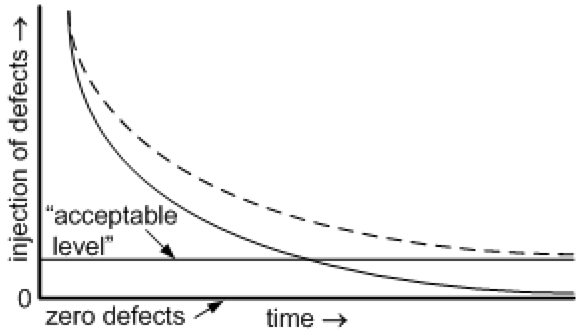
- What do you think is acceptable ?

*

246

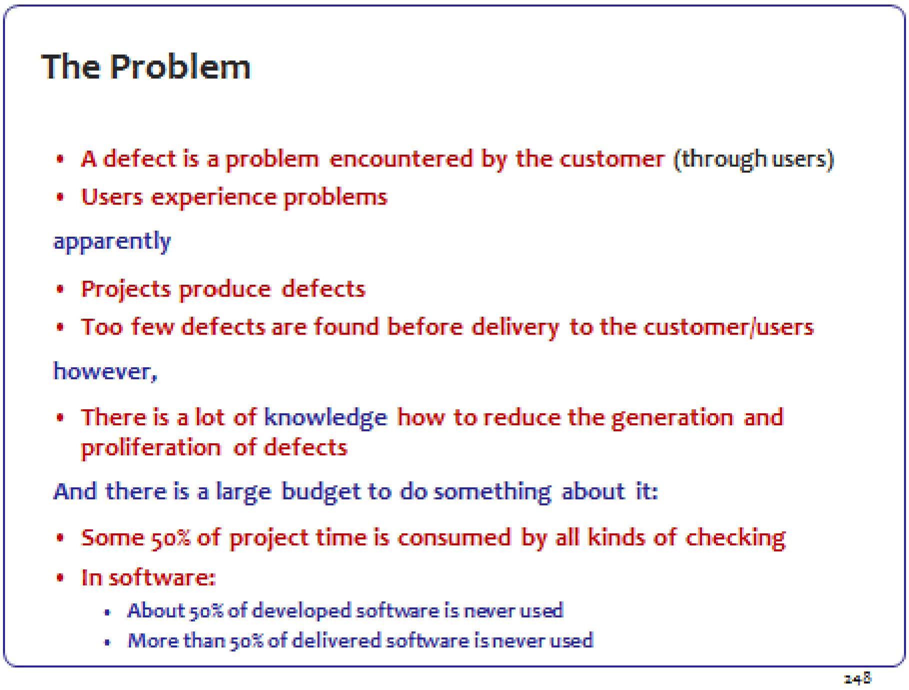
Is Zero Defects possible ?

- Zero Defects is an asymptote



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

247



The Problem

- A defect is a problem encountered by the customer (through users)
- Users experience problems

apparently

- Projects produce defects
- Too few defects are found before delivery to the customer/users

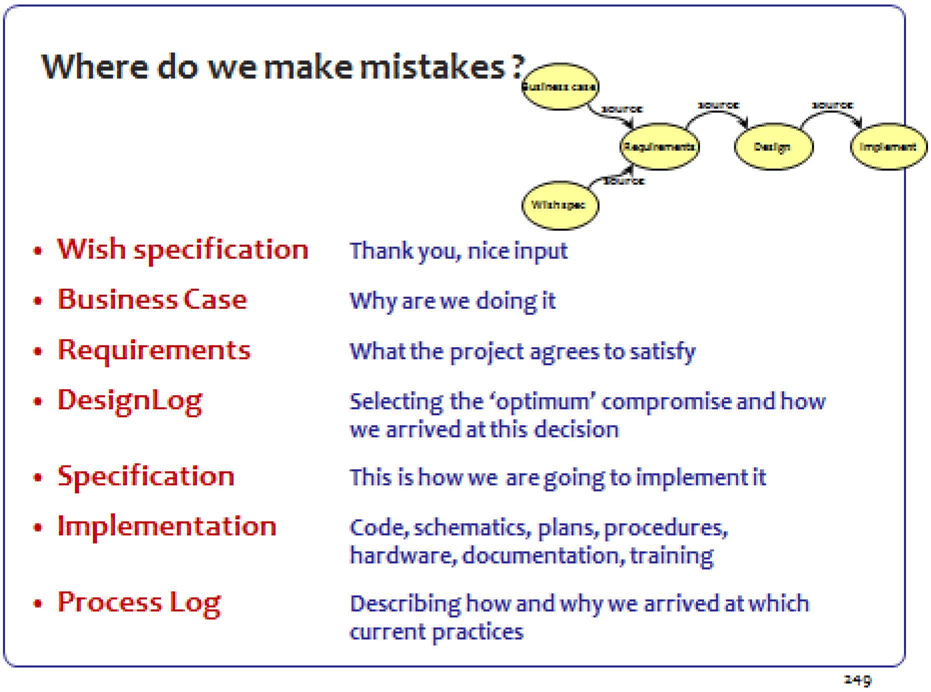
however,

- There is a lot of knowledge how to reduce the generation and proliferation of defects

And there is a large budget to do something about it:

- Some 50% of project time is consumed by all kinds of checking
- In software:
  - About 50% of developed software is never used
  - More than 50% of delivered software is never used

248

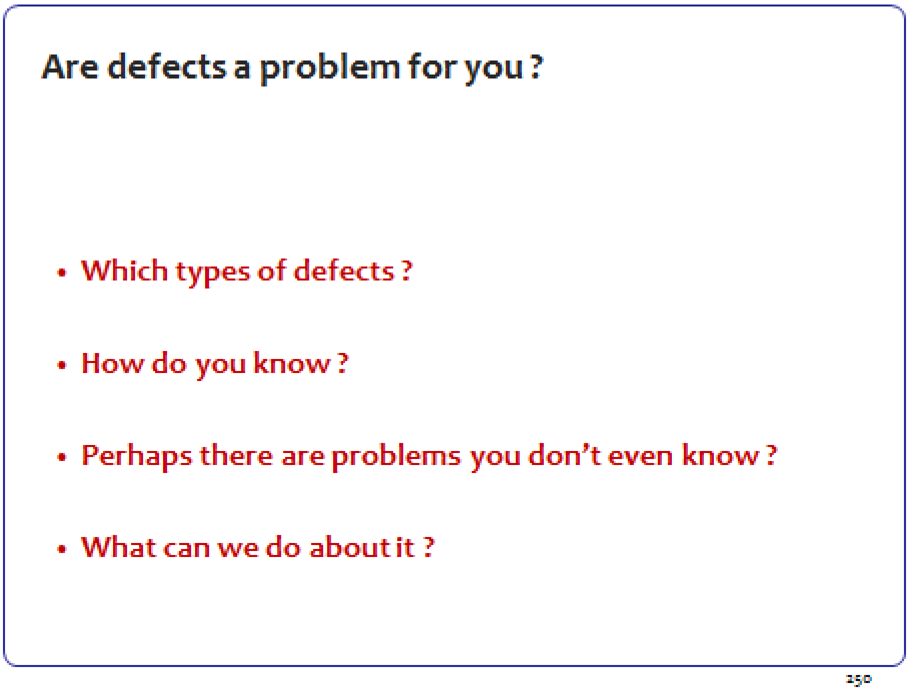


Where do we make mistakes ?

Business case
source
Requirements
source
Design
source
Implement
source
Wishspec

• 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

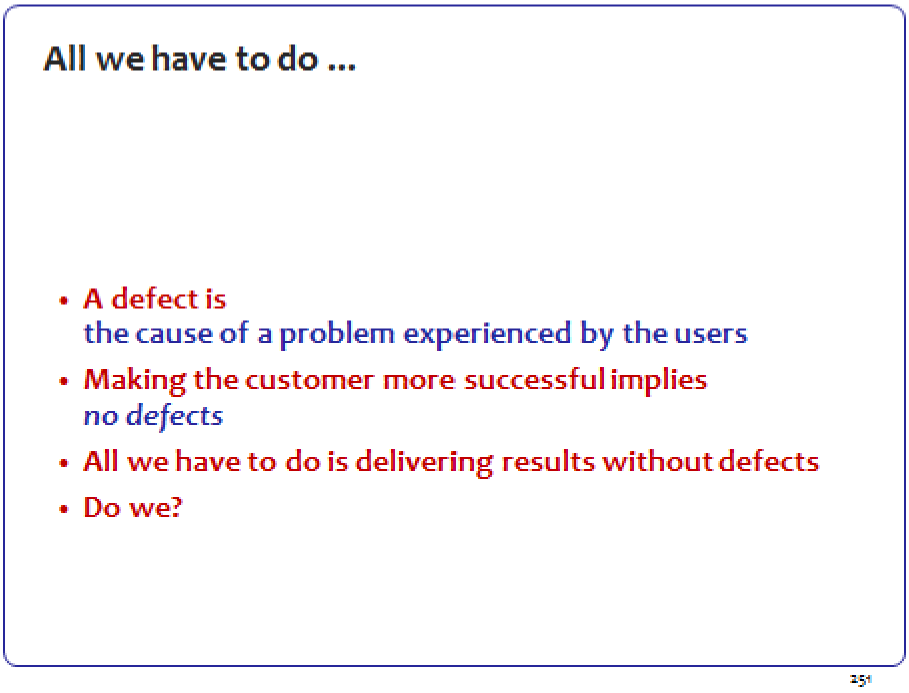
249



Are defects a problem for you ?

- Which types of defects ?
- How do you know ?
- Perhaps there are problems you don't even know ?
- What can we do about it ?

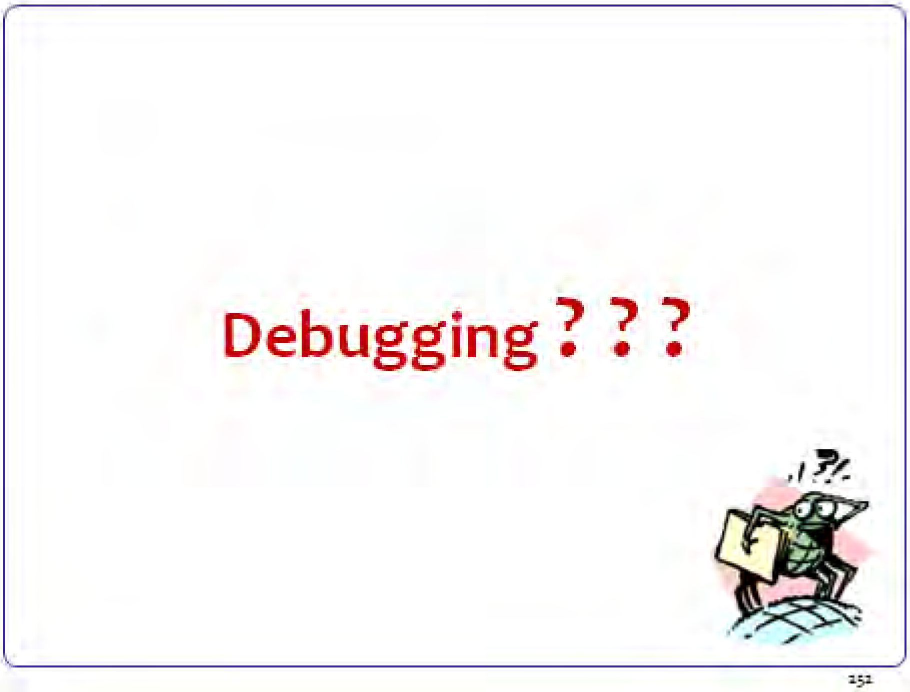
250



All we have to do ...

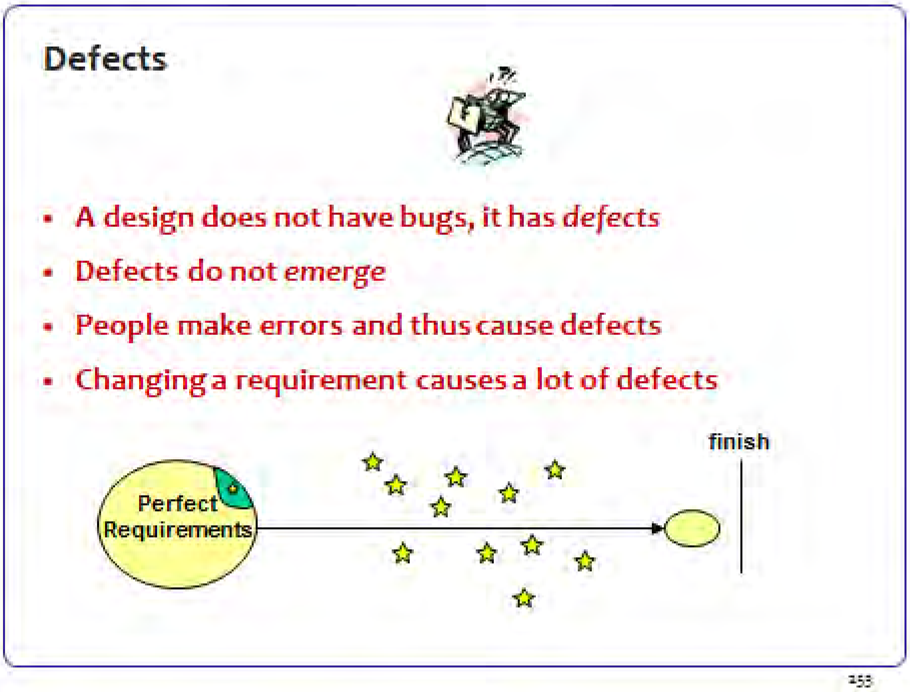
- A defect is
  the cause of a problem experienced by the users
- Making the customer more successful implies
  no defects
- All we have to do is delivering results without defects
- Do we?

251



Debugging ? ? ?

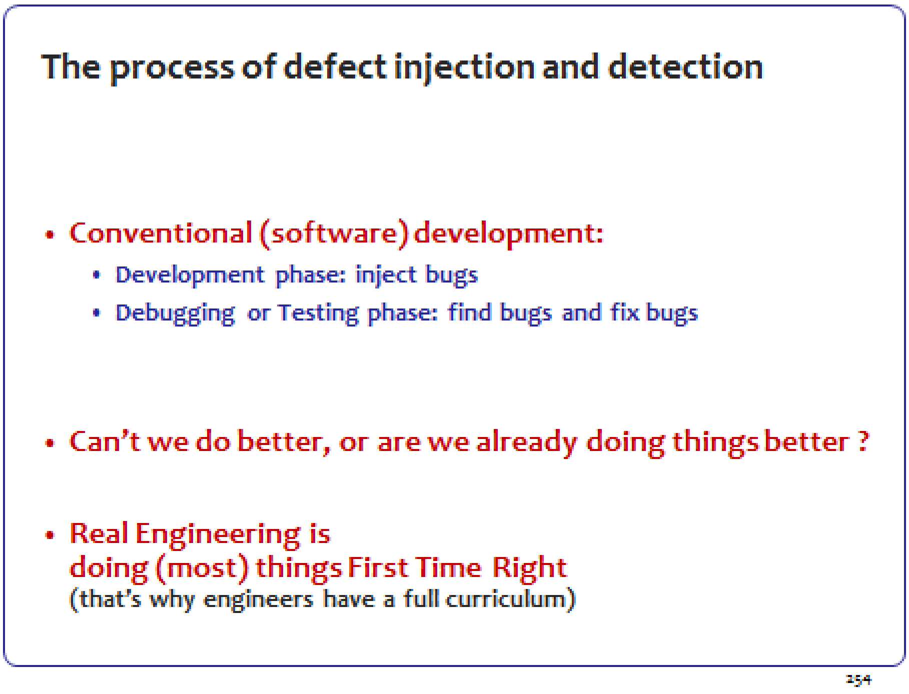
252



Defects

- A design does not have bugs, it has *defects*
- Defects do not *emerge*
- People make errors and thus cause defects
- Changing a requirement causes a lot of defects

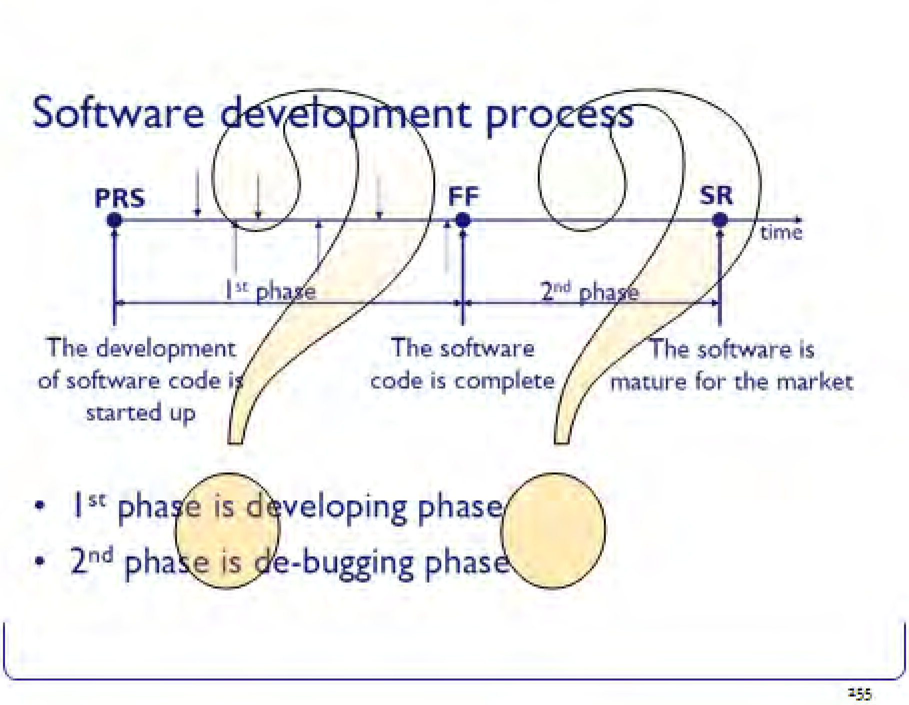
253



The process of defect injection and detection

- Conventional (software) development:
  - Development phase: inject bugs
  - Debugging or Testing phase: find bugs and fix bugs
- Can't we do better, or are we already doing things better ?
- Real Engineering is doing (most) things First Time Right
  (that's why engineers have a full curriculum)

254

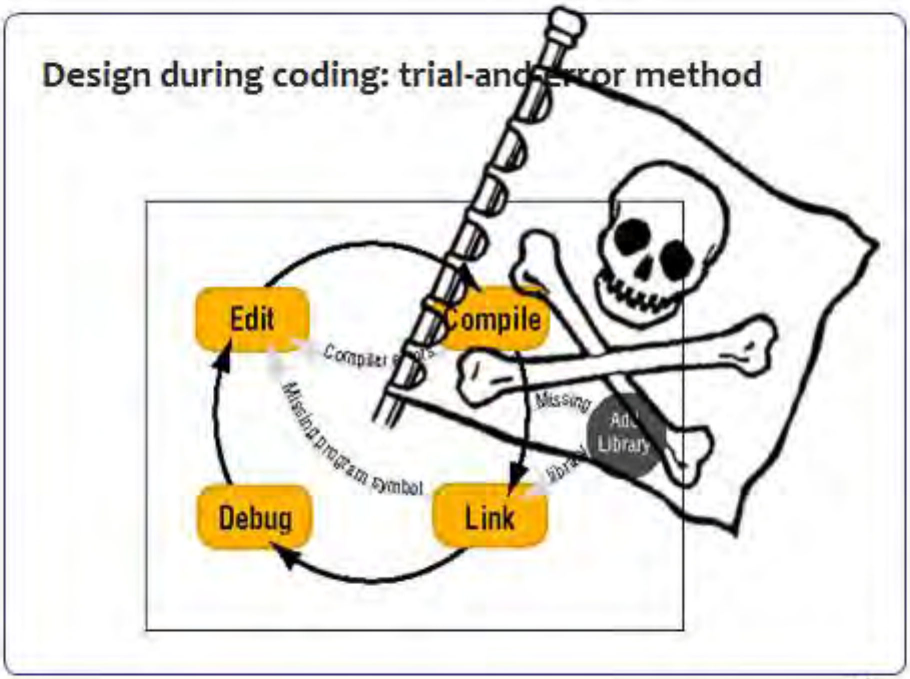


Software development process

PRS
FF
SR
time
1st phase
2nd phase
The development of software code is started up
The software code is complete
The software is mature for the market

- 1st phase is developing phase
- 2nd phase is de-bugging phase

255



Design during coding: trial-and-error method

Edit

Compile

Link

Debug

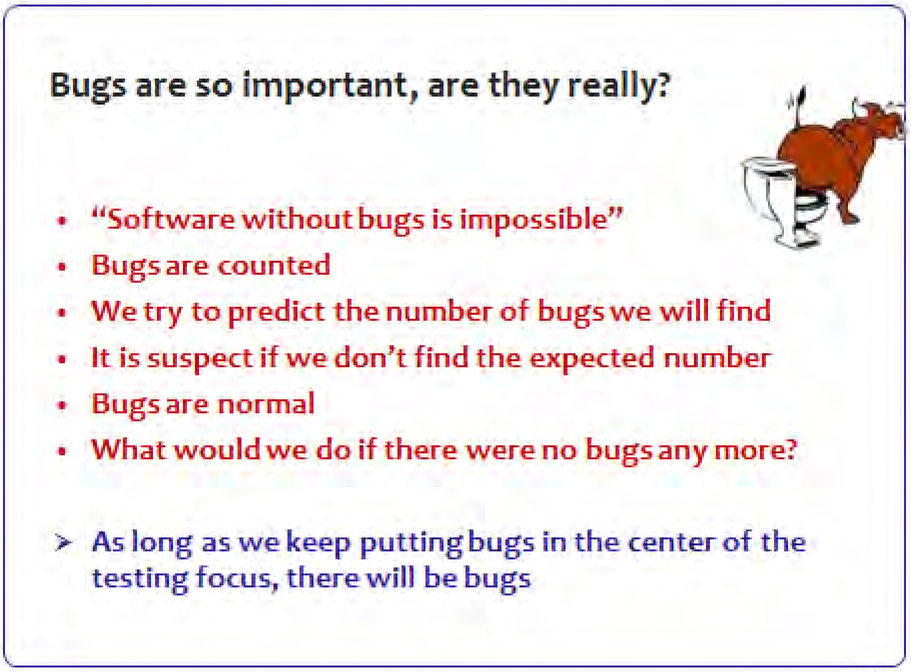
Compiler errors

Missing program symbol

Missing library

Add Library

256

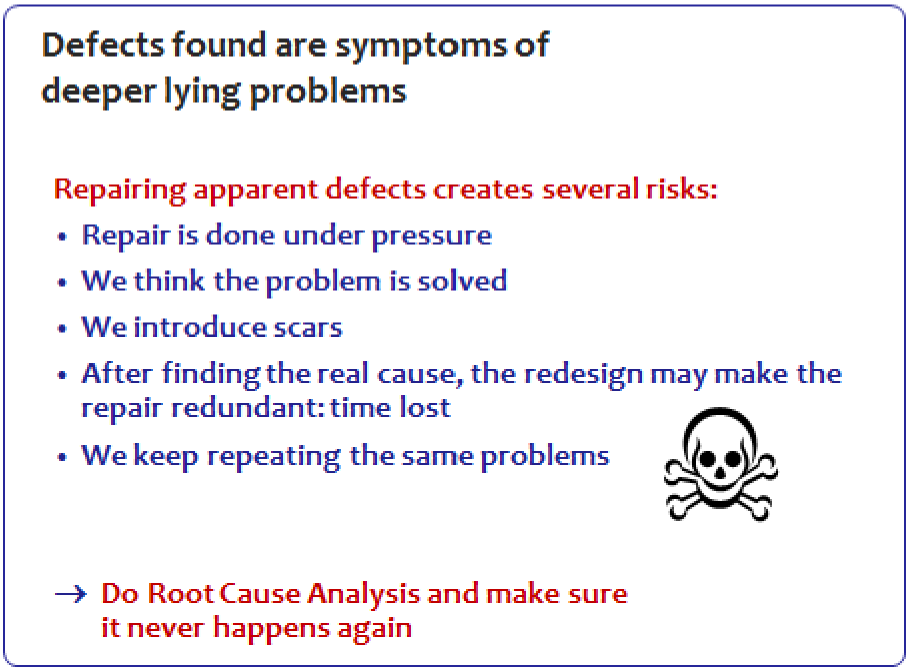


Bugs are so important, are they really?

- “Software without bugs is impossible”
- Bugs are counted
- We try to predict the number of bugs we will find
- It is suspect if we don’t find the expected number
- Bugs are normal
- What would we do if there were no bugs any more?

➢ As long as we keep putting bugs in the center of the testing focus, there will be bugs

257



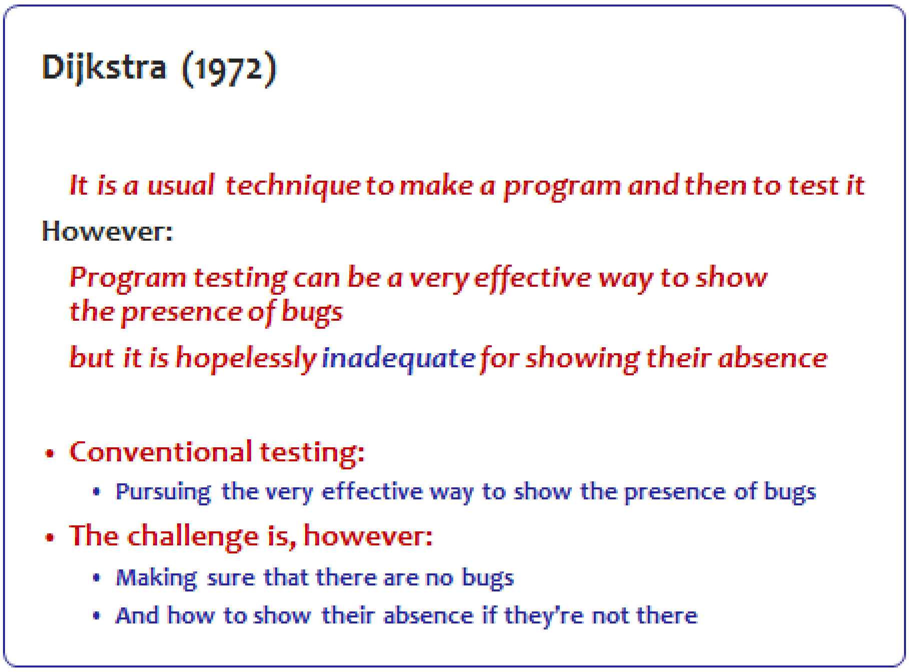
Defects found are symptoms of deeper lying problems

Repairing apparent defects creates several risks:

- Repair is done under pressure
- We think the problem is solved
- We introduce scars
- After finding the real cause, the redesign may make the repair redundant: time lost
- We keep repeating the same problems

→ Do Root Cause Analysis and make sure it never happens again

258



Dijkstra (1972)

It is a usual technique to make a program and then to test it

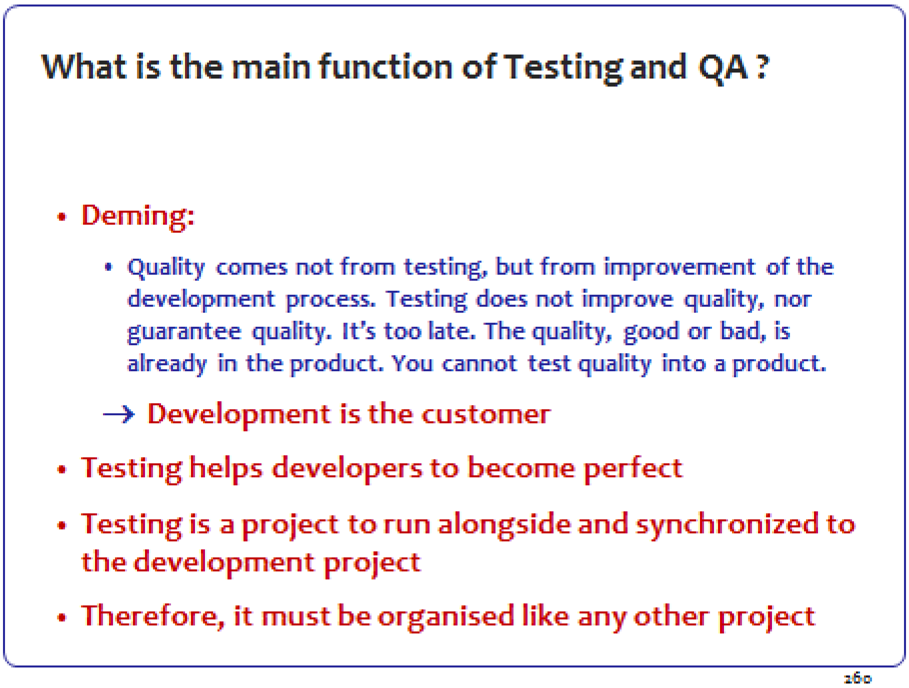
However:

Program testing can be a very effective way to show the presence of bugs

but it is hopelessly inadequate for showing their absence

- Conventional testing:
  - Pursuing the very effective way to show the presence of bugs
- The challenge is, however:
  - Making sure that there are no bugs
  - And how to show their absence if they're not there

259

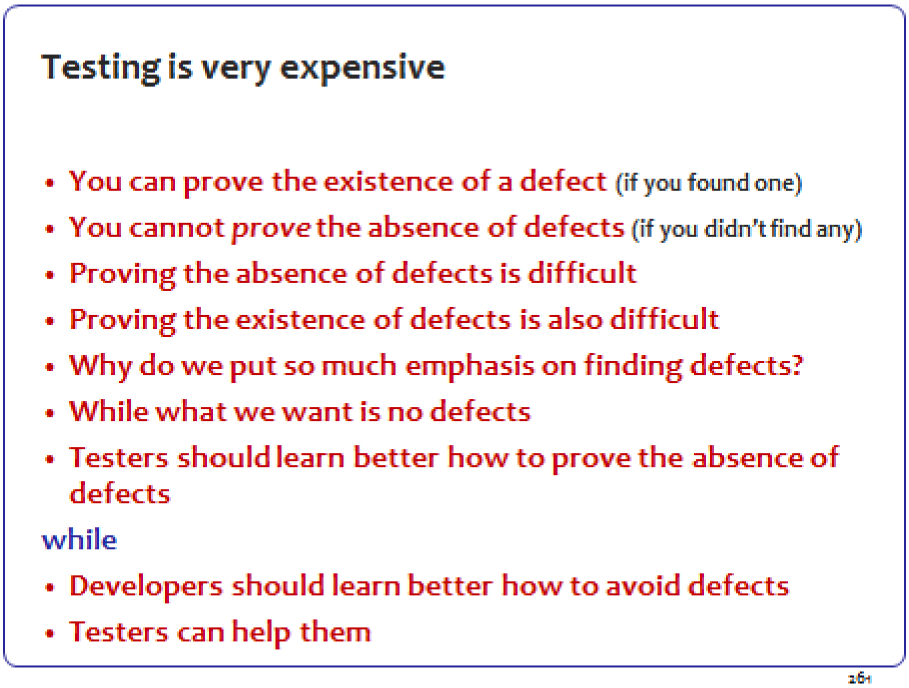


What is the main function of Testing and QA ?

- Deming:
  - Quality comes not from testing, but from improvement of the development process. Testing does not improve quality, nor guarantee quality. It's too late. The quality, good or bad, is already in the product. You cannot test quality into a product.

  → Development is the customer
- Testing helps developers to become perfect
- Testing is a project to run alongside and synchronized to the development project
- Therefore, it must be organised like any other project

260



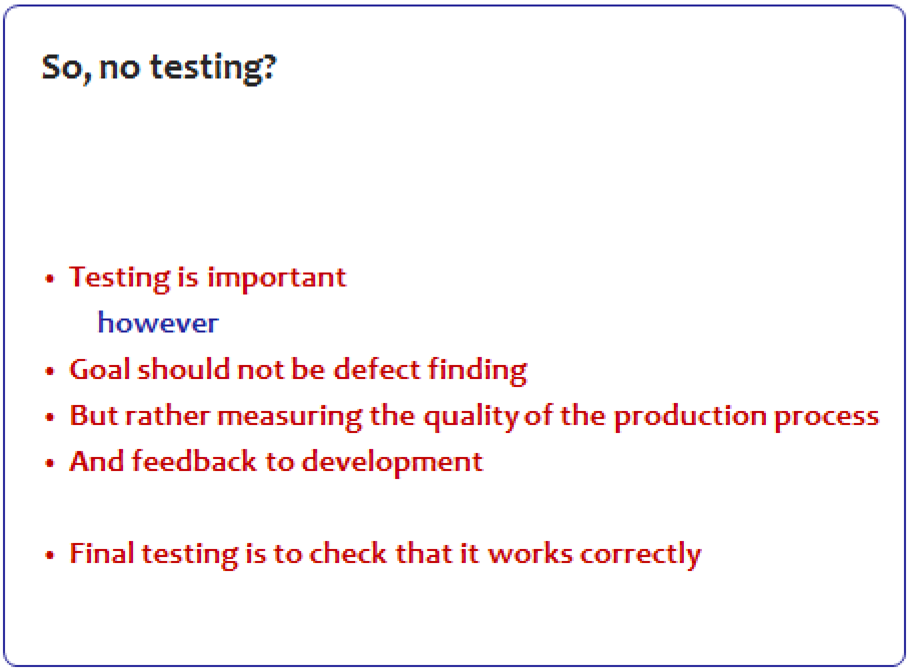
Testing is very expensive

- You can prove the existence of a defect (if you found one)
- You cannot *prove* the absence of defects (if you didn't find any)
- Proving the absence of defects is difficult
- Proving the existence of defects is also difficult
- Why do we put so much emphasis on finding defects?
- While what we want is no defects
- Testers should learn better how to prove the absence of defects

while

- Developers should learn better how to avoid defects
- Testers can help them

261



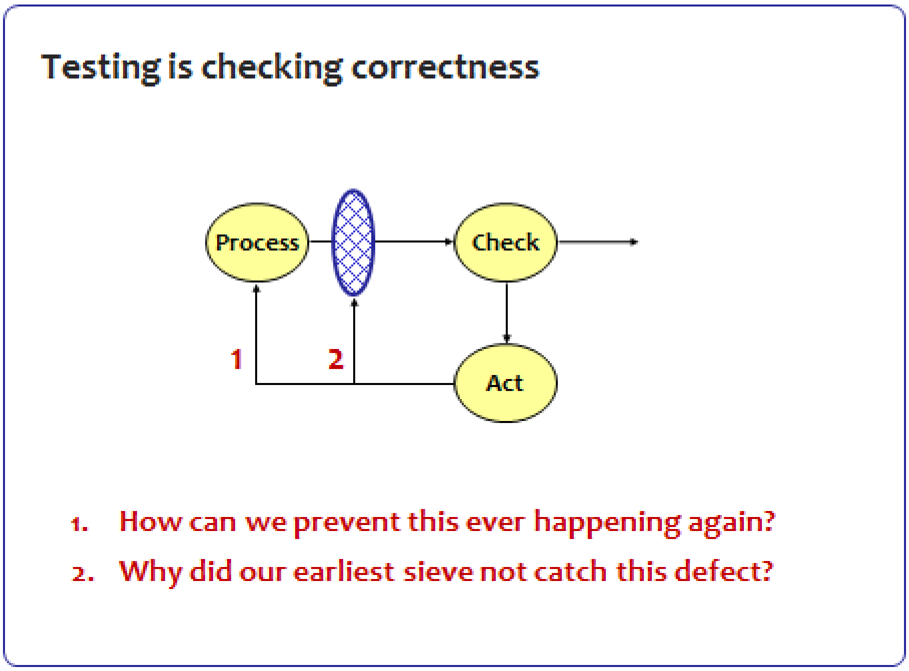
So, no testing?

- Testing is important

  however
- Goal should not be defect finding
- But rather measuring the quality of the production process
- And feedback to development

- Final testing is to check that it works correctly

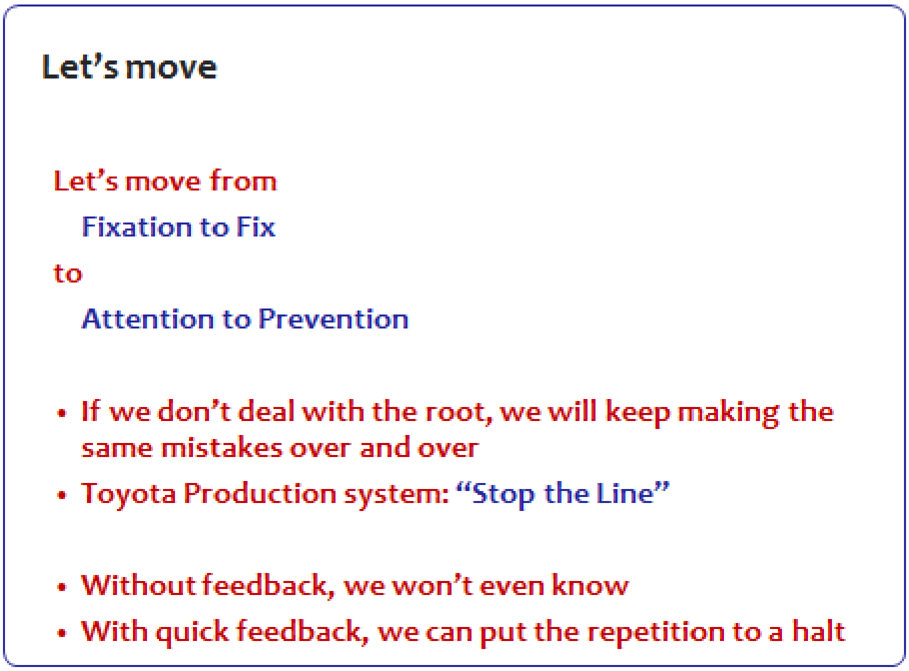
262



Testing is checking correctness

1. How can we prevent this ever happening again?
2. Why did our earliest sieve not catch this defect?

263



Let's move

Let's move from

Fixation to Fix

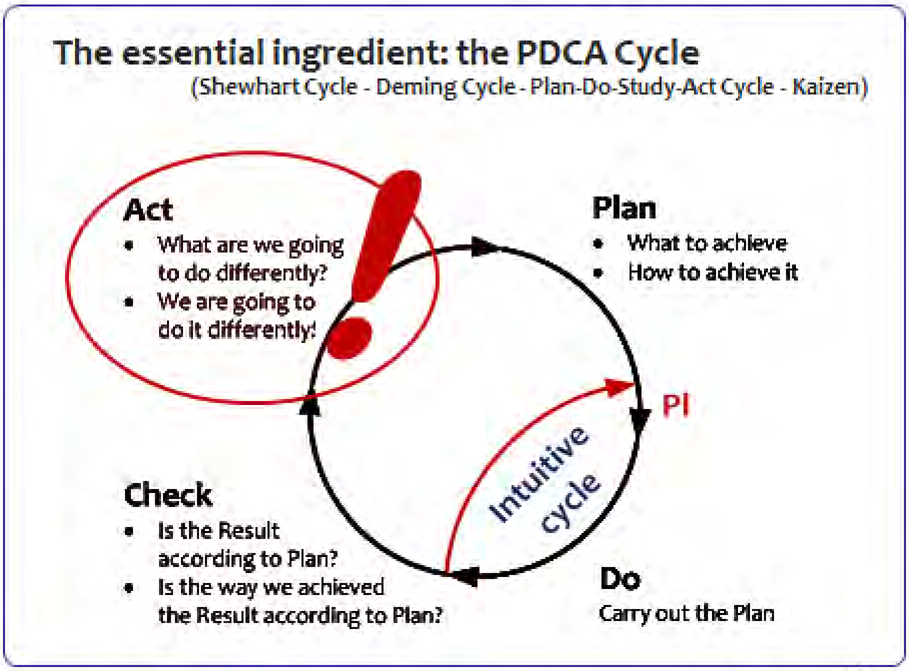
to

Attention to Prevention

- If we don't deal with the root, we will keep making the same mistakes over and over
- Toyota Production system: "Stop the Line"

- Without feedback, we won't even know
- With quick feedback, we can put the repetition to a halt

264



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!

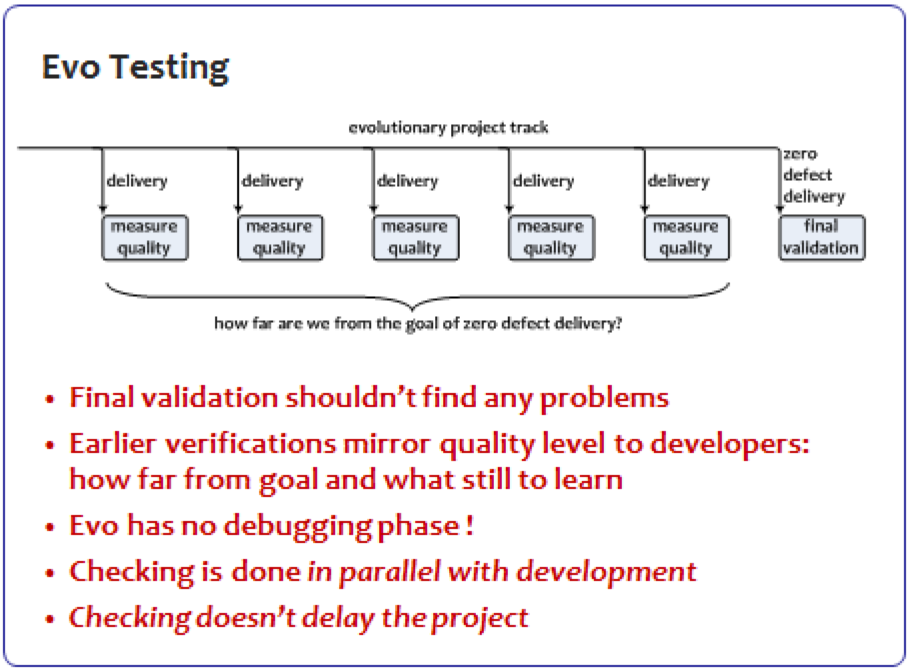
Plan
- What to achieve
- How to achieve it

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

Do

Carry out the Plan

265

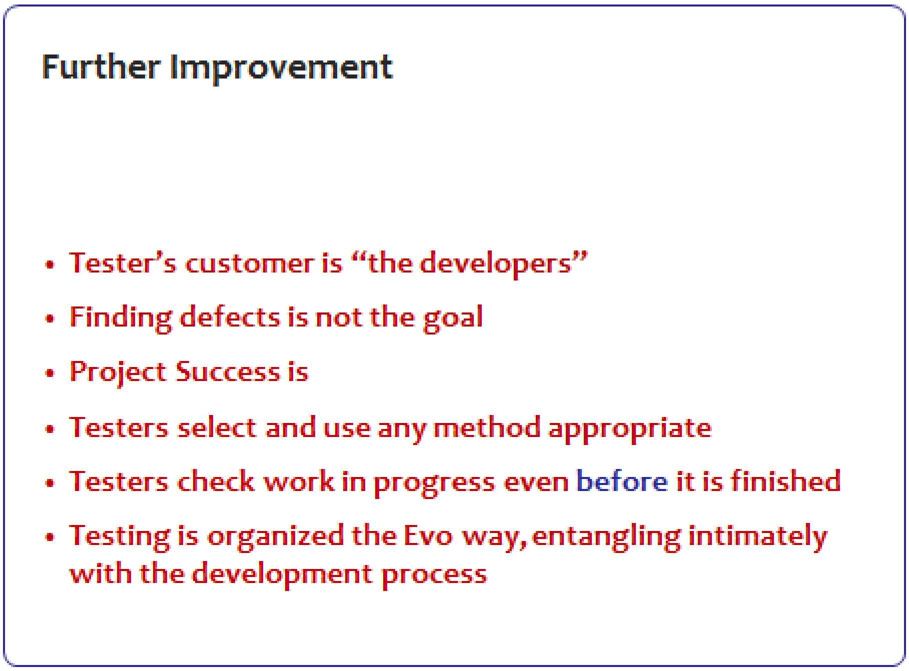


Evo Testing

evolutionary project track
delivery
delivery
delivery
delivery
delivery
zero defect delivery
measure quality
measure quality
measure quality
measure quality
measure quality
final validation
how far are we from the goal of zero defect delivery?

- Final validation shouldn't find any problems
- Earlier verifications mirror quality level to developers: how far from goal and what still to learn
- Evo has no debugging phase !
- Checking is done *in parallel with development*
- *Checking doesn't delay the project*

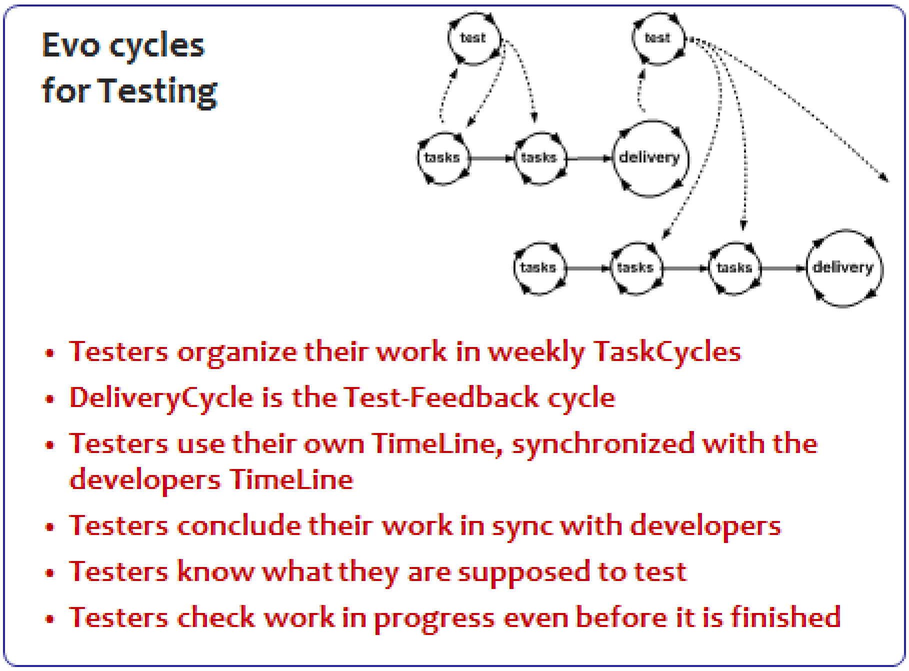
266



Further Improvement

- Tester's customer is "the developers"
- Finding defects is not the goal
- Project Success is
- Testers select and use any method appropriate
- Testers check work in progress even before it is finished
- Testing is organized the Evo way, entangling intimately with the development process

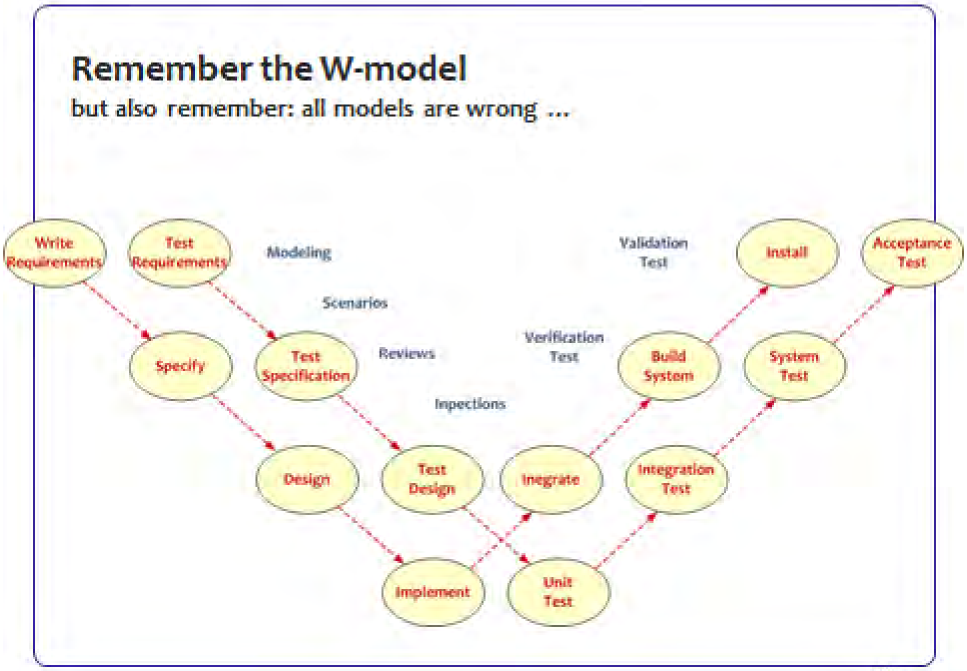
267



Evo cycles for Testing

- Testers organize their work in weekly TaskCycles
- DeliveryCycle is the Test-Feedback cycle
- Testers use their own TimeLine, synchronized with the developers TimeLine
- Testers conclude their work in sync with developers
- Testers know what they are supposed to test
- Testers check work in progress even before it is finished

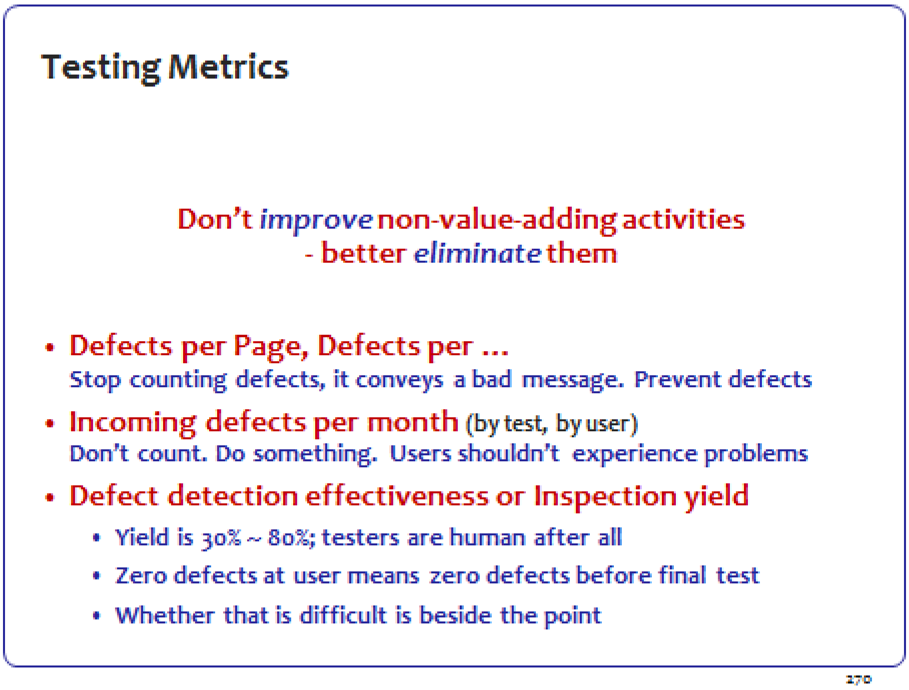
268



Remember the W-model

but also remember: all models are wrong ...

269

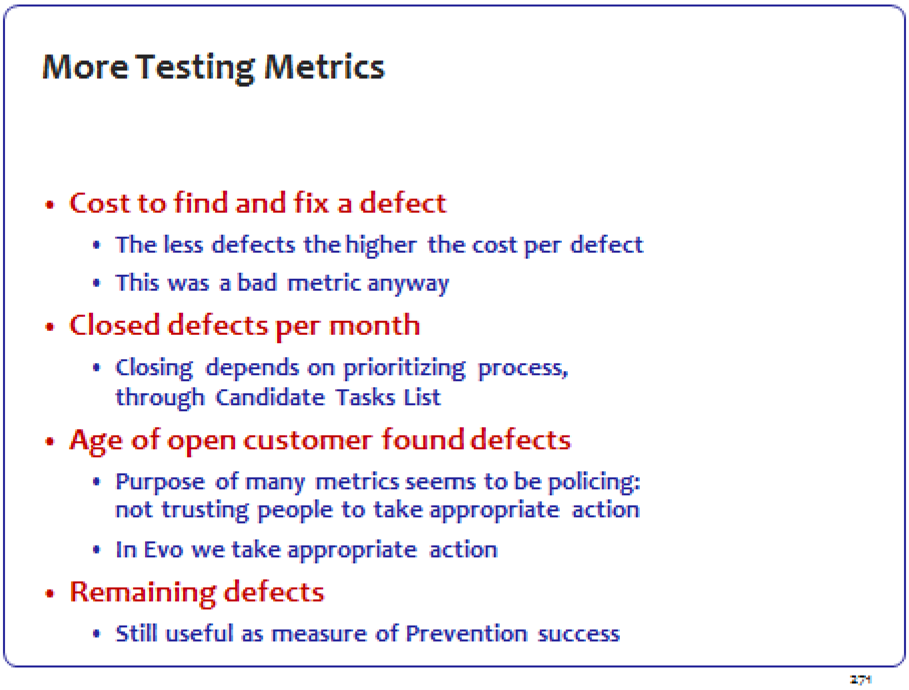


Testing Metrics

Don't *improve* non-value-adding activities
- better *eliminate* them

- Defects per Page, Defects per ...
  Stop counting defects, it conveys a bad message. Prevent defects
- Incoming defects per month (by test, by user)
  Don't count. Do something. Users shouldn't experience problems
- Defect detection effectiveness or Inspection yield
  - Yield is 30% ~ 80%; testers are human after all
  - Zero defects at user means zero defects before final test
  - Whether that is difficult is beside the point

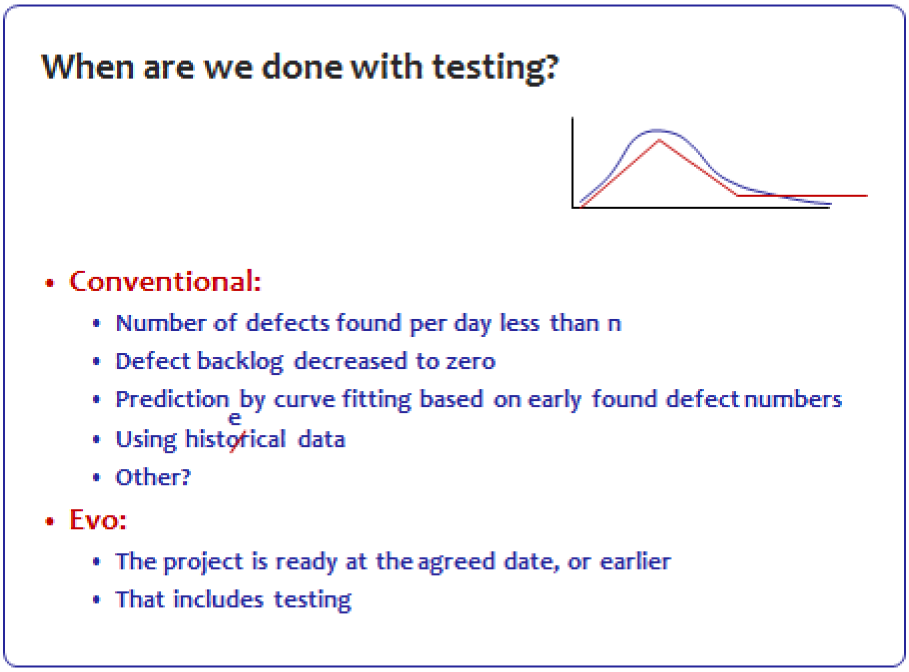
270



More Testing Metrics

- Cost to find and fix a defect
  - The less defects the higher the cost per defect
  - This was a bad metric anyway
- Closed defects per month
  - Closing depends on prioritizing process, through Candidate Tasks List
- Age of open customer found defects
  - Purpose of many metrics seems to be policing: not trusting people to take appropriate action
  - In Evo we take appropriate action
- Remaining defects
  - Still useful as measure of Prevention success

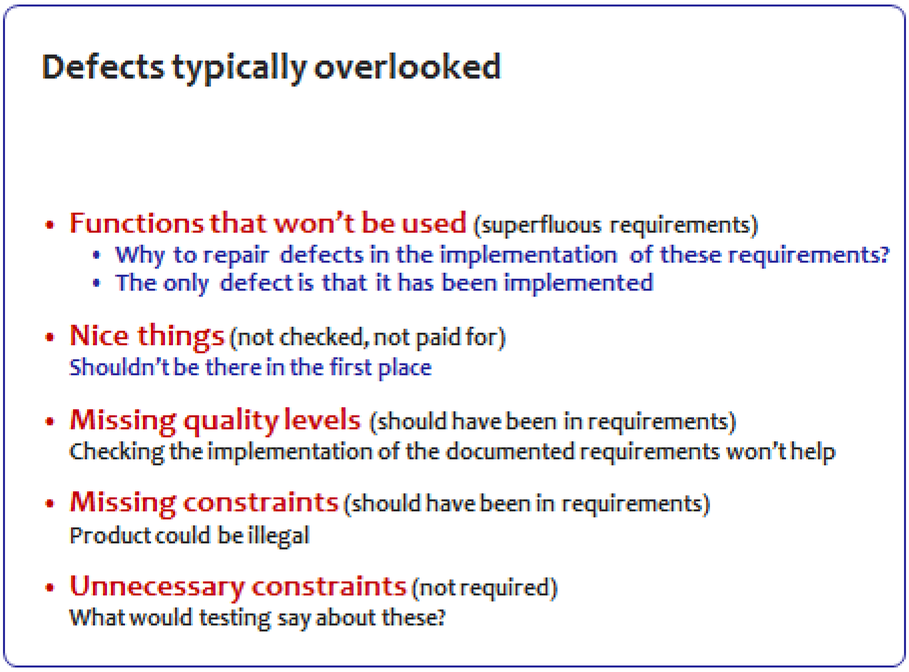
271



When are we done with testing?

- Conventional:
  - Number of defects found per day less than n
  - Defect backlog decreased to zero
  - Prediction by curve fitting based on early found defect numbers
  - Using hist$\overset{e}{\cancel{o}}$rical data
  - Other?
- Evo:
  - The project is ready at the agreed date, or earlier
  - That includes testing

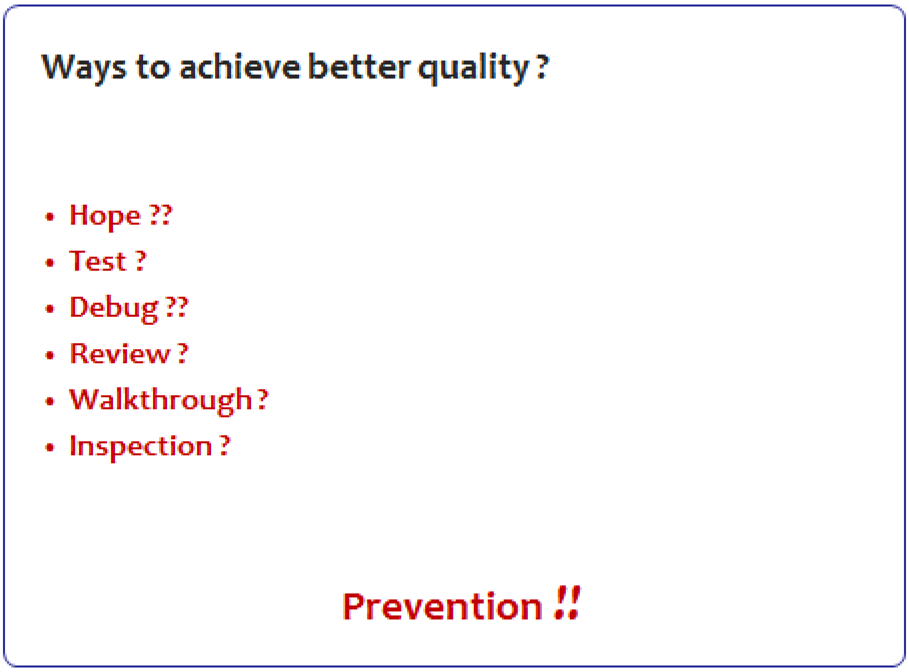
272



Defects typically overlooked

- Functions that won't be used (superfluous requirements)
  - Why to repair defects in the implementation of these requirements?
  - The only defect is that it has been implemented
- Nice things (not checked, not paid for)
  Shouldn't be there in the first place
- Missing quality levels (should have been in requirements)
  Checking the implementation of the documented requirements won't help
- Missing constraints (should have been in requirements)
  Product could be illegal
- Unnecessary constraints (not required)
  What would testing say about these?

273

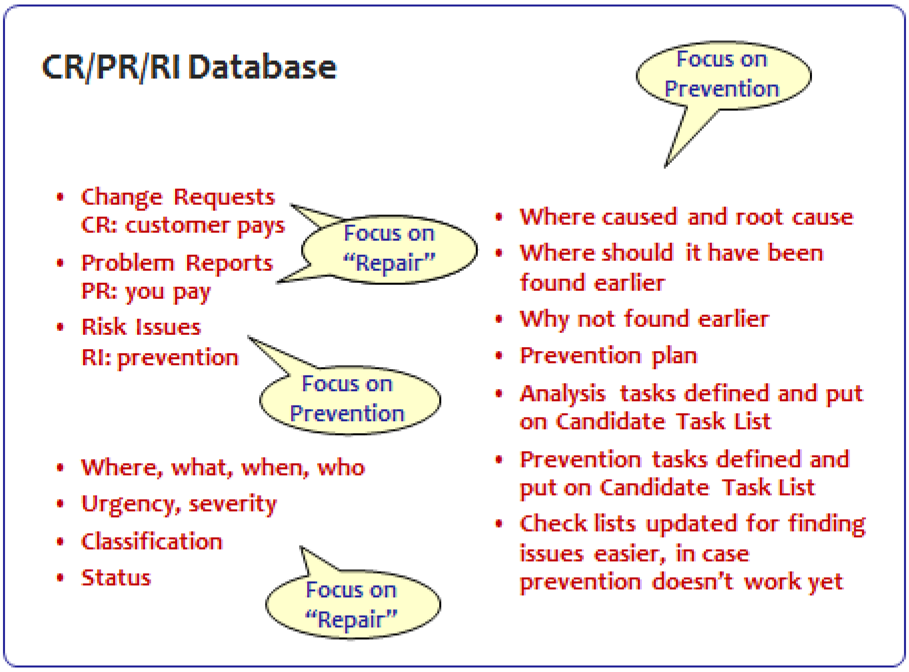


Ways to achieve better quality ?

- Hope ??
- Test ?
- Debug ??
- Review ?
- Walkthrough ?
- Inspection ?

Prevention !!

274



CR/PR/RI Database

Focus on Prevention

- Change Requests
  CR: customer pays
- Problem Reports
  PR: you pay
- Risk Issues
  RI: prevention

Focus on "Repair"

Focus on Prevention

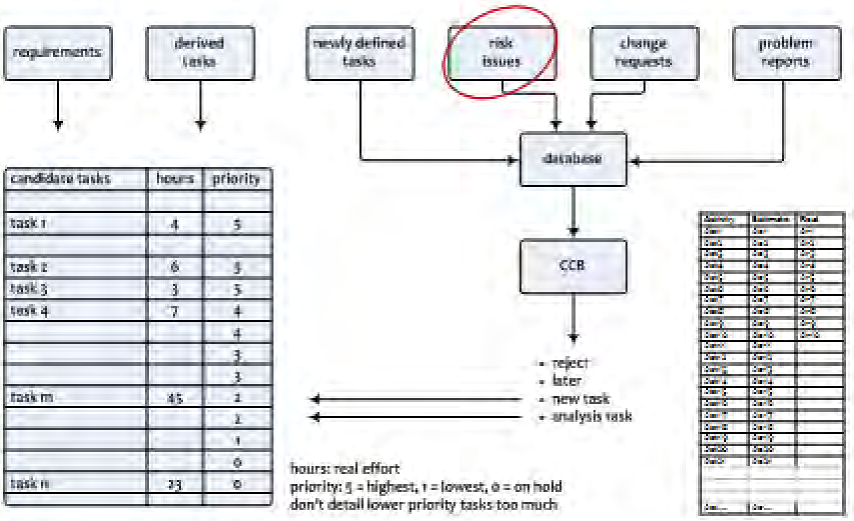
- Where, what, when, who
- Urgency, severity
- Classification
- Status

Focus on "Repair"

- Where caused and root cause
- Where should it have been found earlier
- Why not found earlier
- Prevention plan
- Analysis tasks defined and put on Candidate Task List
- Prevention tasks defined and put on Candidate Task List
- Check lists updated for finding issues easier, in case prevention doesn't work yet

275

Anything we think must be done goes through the *Candidate Task* Mechanism



candidate tasks	hours	priority
task 1	4	5
task 2	6	5
task 3	3	5
task 4	7	4
		4
		3
		3
task m	45	2
		2
		1
		0
task n	23	0

hours: real effort
priority: 5 = highest, 1 = lowest, 0 = on hold
don't detail lower priority tasks too much

276

Testing vs Reviews & Inspections

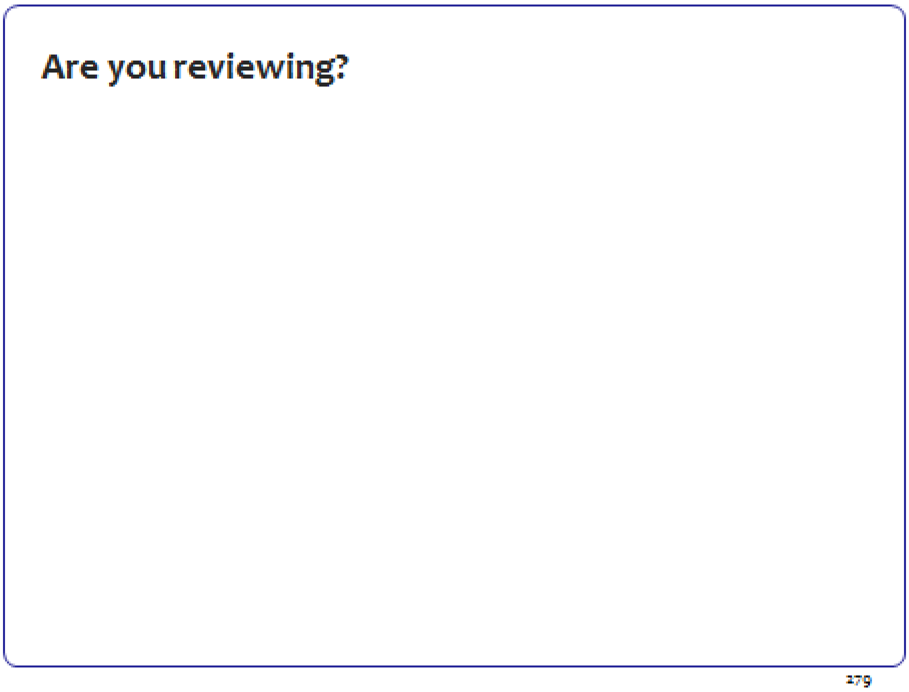
- If you find an issue during Test, you still have to find the origin
- If you find an issue during Review or Inspection, you're on top of it

- If Testing means running the system
- And Review / Inspection means Verifying and/or Validation of a document

277

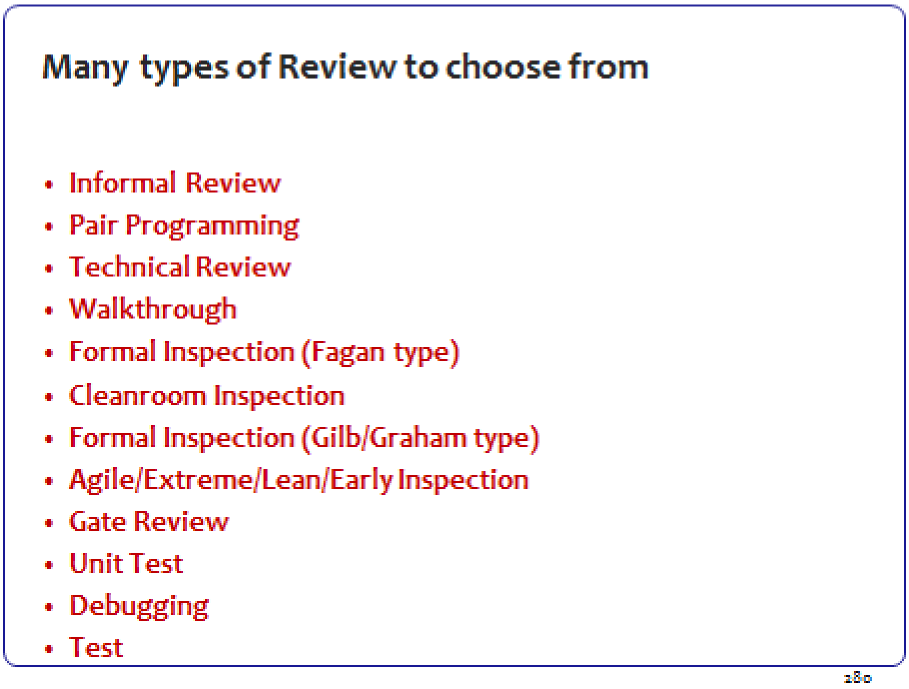
Early Reviews & Inspections

278



Are you reviewing?

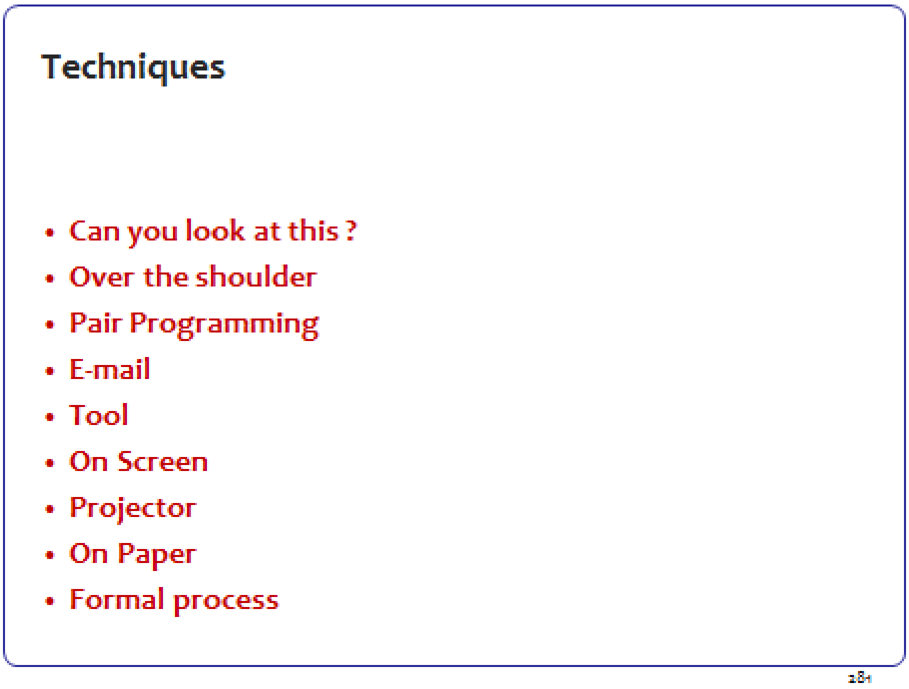
279



Many types of Review to choose from

- Informal Review
- Pair Programming
- Technical Review
- Walkthrough
- Formal Inspection (Fagan type)
- Cleanroom Inspection
- Formal Inspection (Gilb/Graham type)
- Agile/Extreme/Lean/Early Inspection
- Gate Review
- Unit Test
- Debugging
- Test

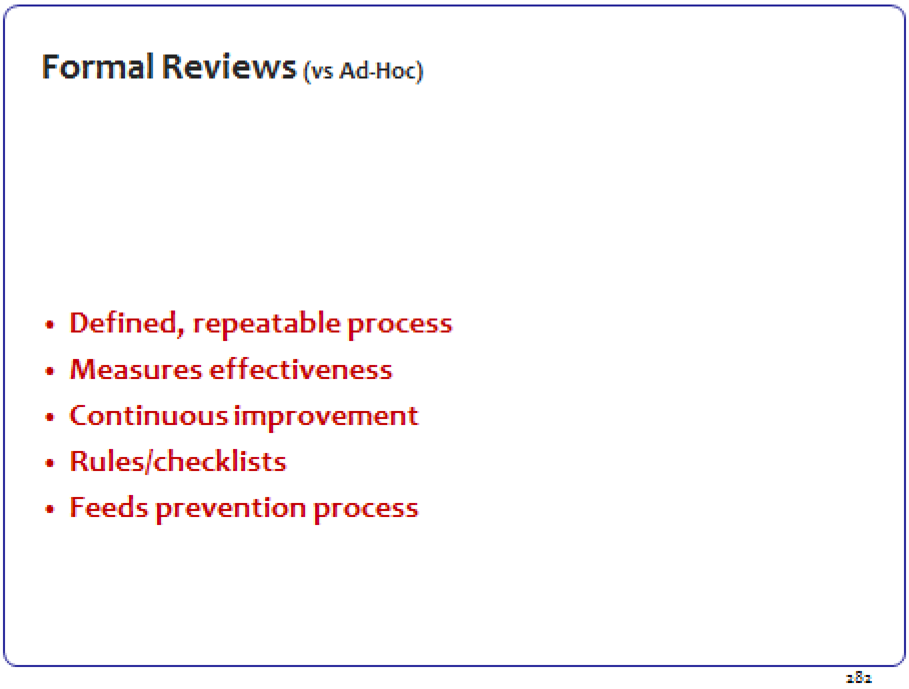
280



Techniques

- Can you look at this ?
- Over the shoulder
- Pair Programming
- E-mail
- Tool
- On Screen
- Projector
- On Paper
- Formal process

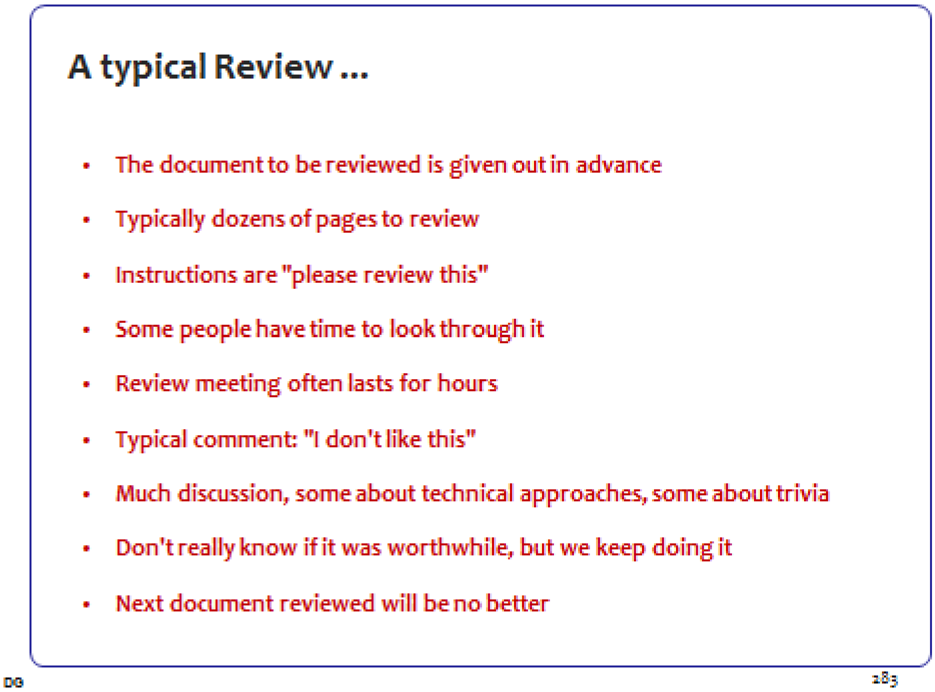
281



Formal Reviews (vs Ad-Hoc)

- Defined, repeatable process
- Measures effectiveness
- Continuous improvement
- Rules/checklists
- Feeds prevention process

282

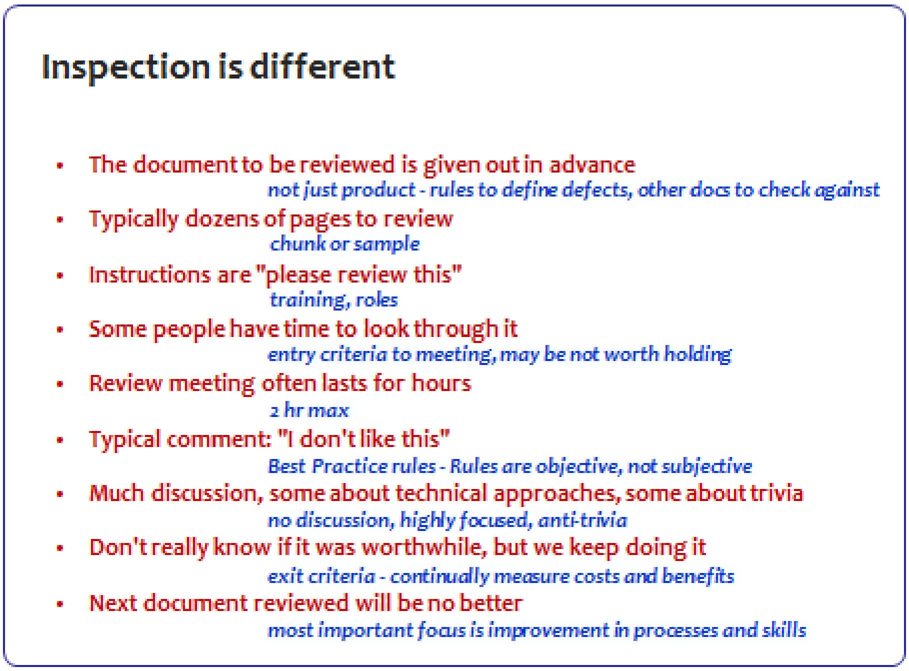


A typical Review ...

- The document to be reviewed is given out in advance
- Typically dozens of pages to review
- Instructions are "please review this"
- Some people have time to look through it
- Review meeting often lasts for hours
- Typical comment: "I don't like this"
- Much discussion, some about technical approaches, some about trivia
- Don't really know if it was worthwhile, but we keep doing it
- Next document reviewed will be no better

DG

283

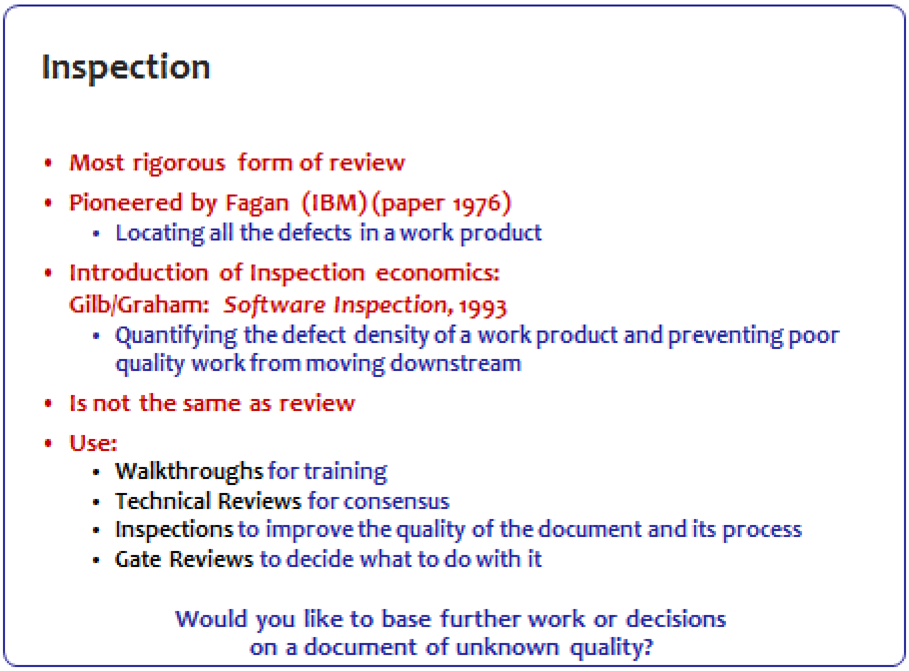


Inspection is different

- The document to be reviewed is given out in advance
  not just product - rules to define defects, other docs to check against
- Typically dozens of pages to review
  chunk or sample
- Instructions are "please review this"
  training, roles
- Some people have time to look through it
  entry criteria to meeting, may be not worth holding
- Review meeting often lasts for hours
  2 hr max
- Typical comment: "I don't like this"
  Best Practice rules - Rules are objective, not subjective
- Much discussion, some about technical approaches, some about trivia
  no discussion, highly focused, anti-trivia
- Don't really know if it was worthwhile, but we keep doing it
  exit criteria - continually measure costs and benefits
- Next document reviewed will be no better
  most important focus is improvement in processes and skills

DC

284

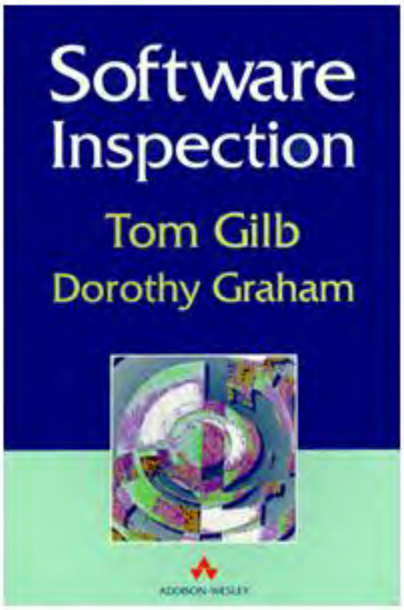


Inspection

- Most rigorous form of review
- Pioneered by Fagan (IBM) (paper 1976)
  - Locating all the defects in a work product
- Introduction of Inspection economics:
  Gilb/Graham: *Software Inspection*, 1993
  - Quantifying the defect density of a work product and preventing poor quality work from moving downstream
- Is not the same as review
- Use:
  - Walkthroughs for training
  - Technical Reviews for consensus
  - Inspections to improve the quality of the document and its process
  - Gate Reviews to decide what to do with it

Would you like to base further work or decisions
on a document of unknown quality?

285



A ready to use recipe ...

286

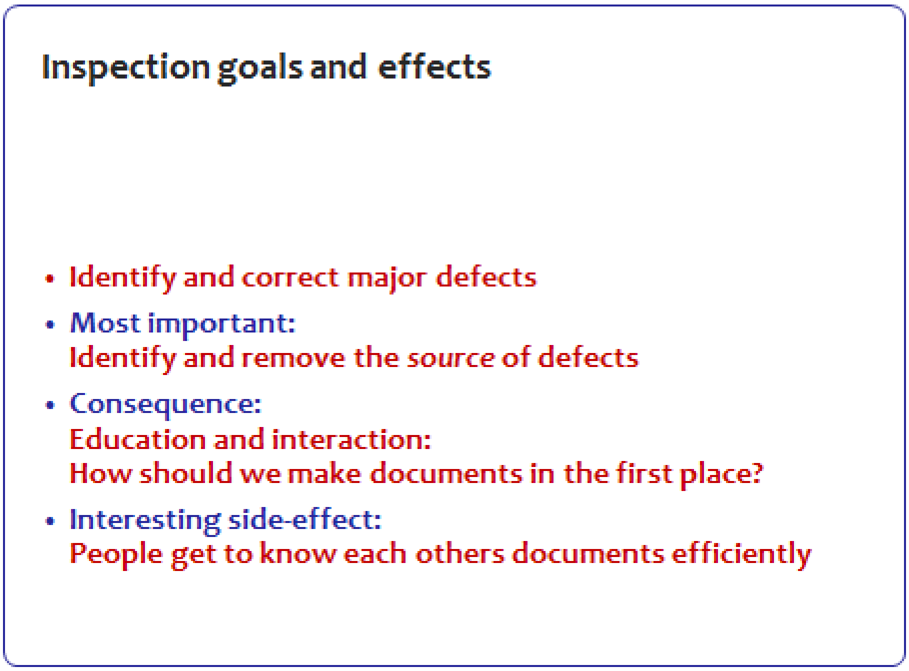
16 page Inspection Manual

Inspection Manual
Procedures, rules, checklists and other texts for use in Inspections

Version 0.40 (Changed from previous)
Date: Oct 13, 2007
Owner: Niels Malotaux
Status: not inspected
Intended readership: anybody interested in or busy with Inspections

Note: Most of these texts are originally taken from the book "Software Inspection" by Tom Gilb and Dorothy Graham, Addison Wesley, 1993, ISBN 0-201-63181-4, and from websites, such as www.result-planning.com (Tom Gilb's website). This is a starting point from which the procedures, rules etc. may be adapted to the local culture.

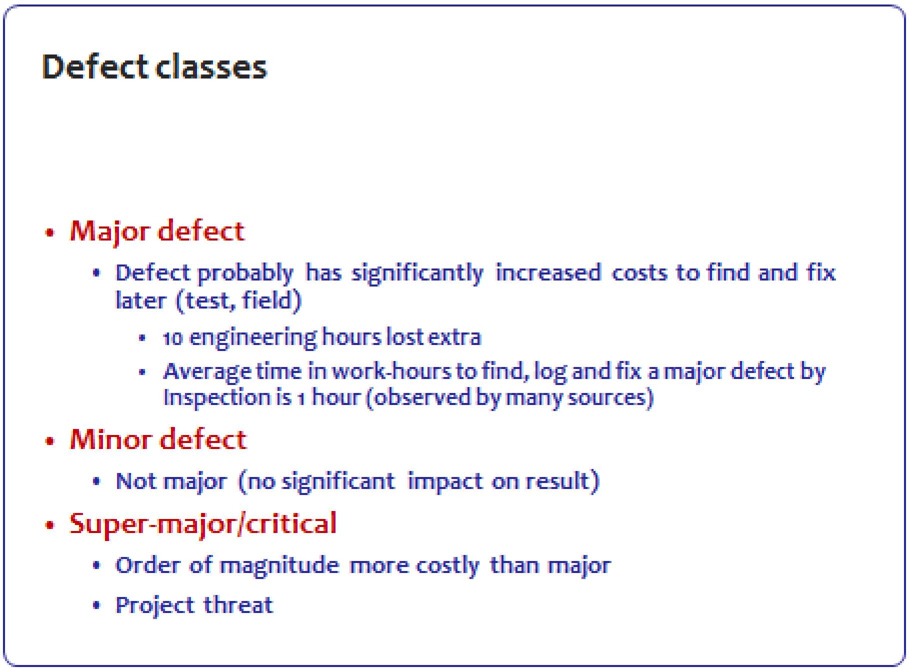
287



Inspection goals and effects

- Identify and correct major defects
- Most important:
  Identify and remove the source of defects
- Consequence:
  Education and interaction:
  How should we make documents in the first place?
- Interesting side-effect:
  People get to know each others documents efficiently

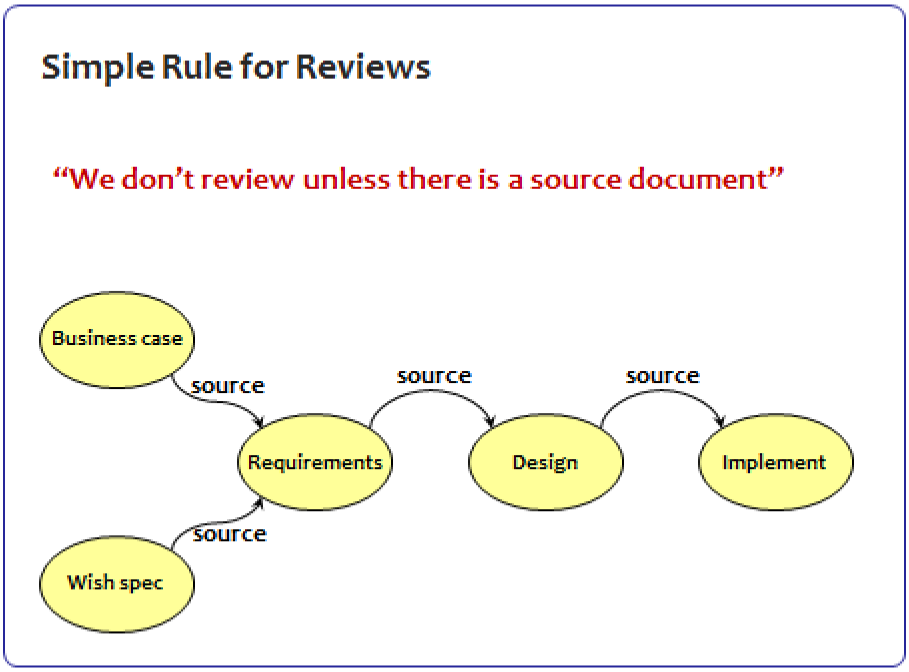
288



Defect classes

- Major defect
  - Defect probably has significantly increased costs to find and fix later (test, field)
    - 10 engineering hours lost extra
    - Average time in work-hours to find, log and fix a major defect by Inspection is 1 hour (observed by many sources)
- Minor defect
  - Not major (no significant impact on result)
- Super-major/critical
  - Order of magnitude more costly than major
  - Project threat

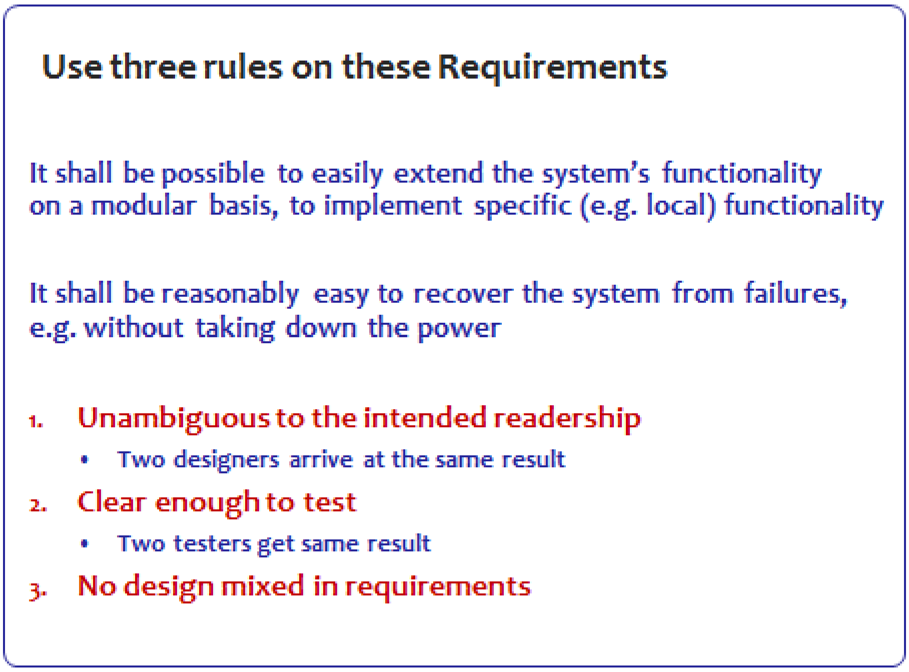
289



Simple Rule for Reviews

"We don't review unless there is a source document"

290



Use three rules on these Requirements

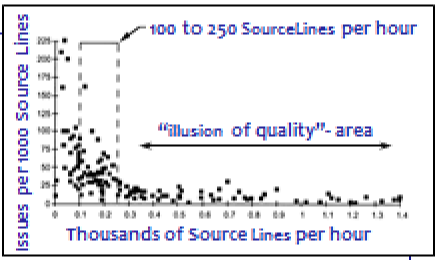
It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality

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

291

Optimum checking rate



Raytheon, CMU/SEI-95-TR017

- The most effective individual speed for 'checking a document against all related documents' in page/hr
- Not 'reading' speed, but rather correlation speed
- Failure to use it, gives 'bad estimate' for 'Remaining defects'

- 100~250 SLoC per hour
- 1 page of 300 words per hour ("logical page")

TG

292

Optimum checking rate

Ref. Dorothy Graham

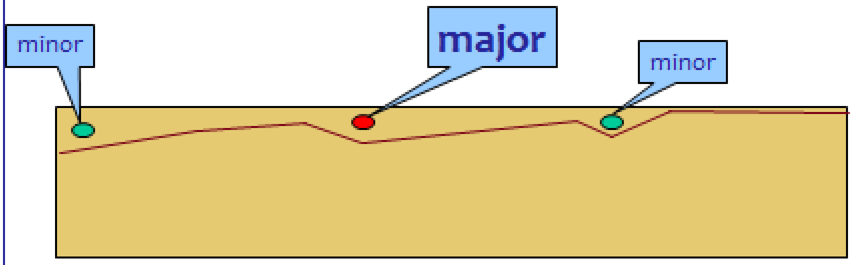
Here's a document: review this (or Inspect it)

DG

293

Review "Thoroughness"?

Ref. Dorothy Graham

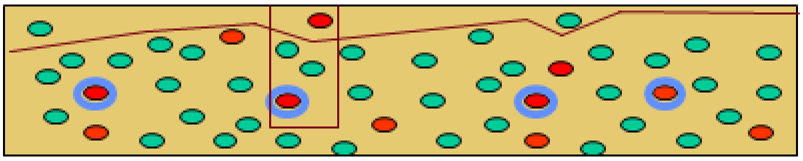


- **Ordinary review**
  - Find some defects, one Major
  - Fix them
  - Consider the document now corrected and OK ...

DG 294

Inspection Thoroughness

Ref. Dorothy Graham



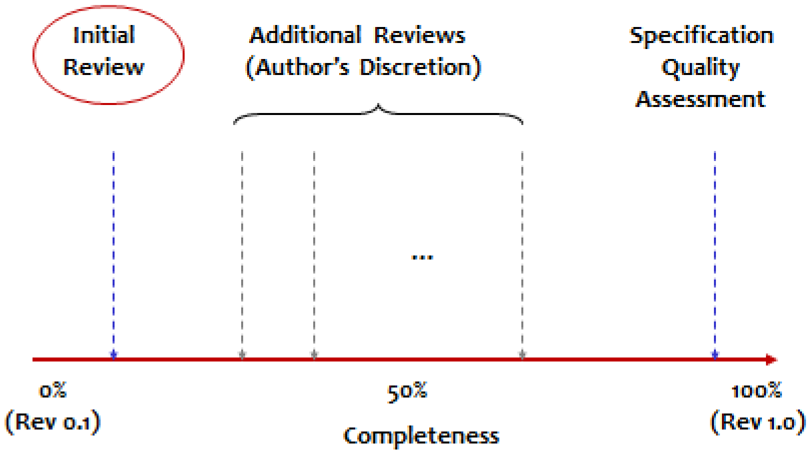
- Inspection can find deep-seated defects
- <u>All of that type</u> can be corrected
- Needs optimum checking rate

- In the above case we are clearly taking a sample
- In the "shallow" case we we're also taking a sample, however, we didn't realize it !

DG 295

Early Inspection

Prevention costs less than Repair



Initial Review

Additional Reviews
(Author's Discretion)

Specification
Quality
Assessment

...

0%
(Rev 0.1)

50%
Completeness

100%
(Rev 1.0)

E8 296

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)

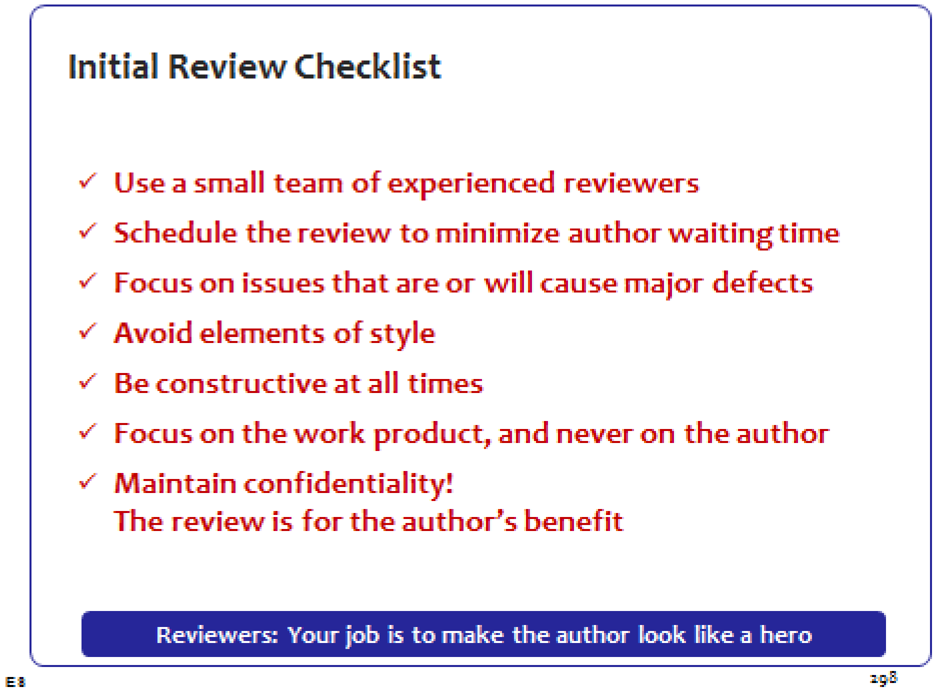
Who: Individual or small team (1 or 2)
- Expertise in the subject matter
- Expertise in generic principles (such as requirements engineering, design, specific language)

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 297



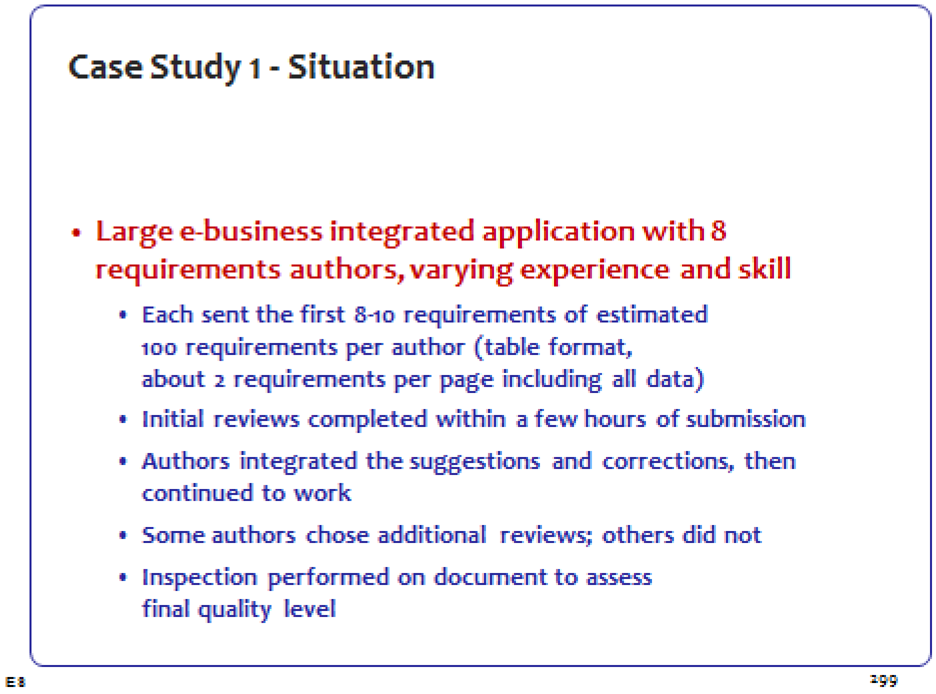
Initial Review Checklist

- ✓ 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

298

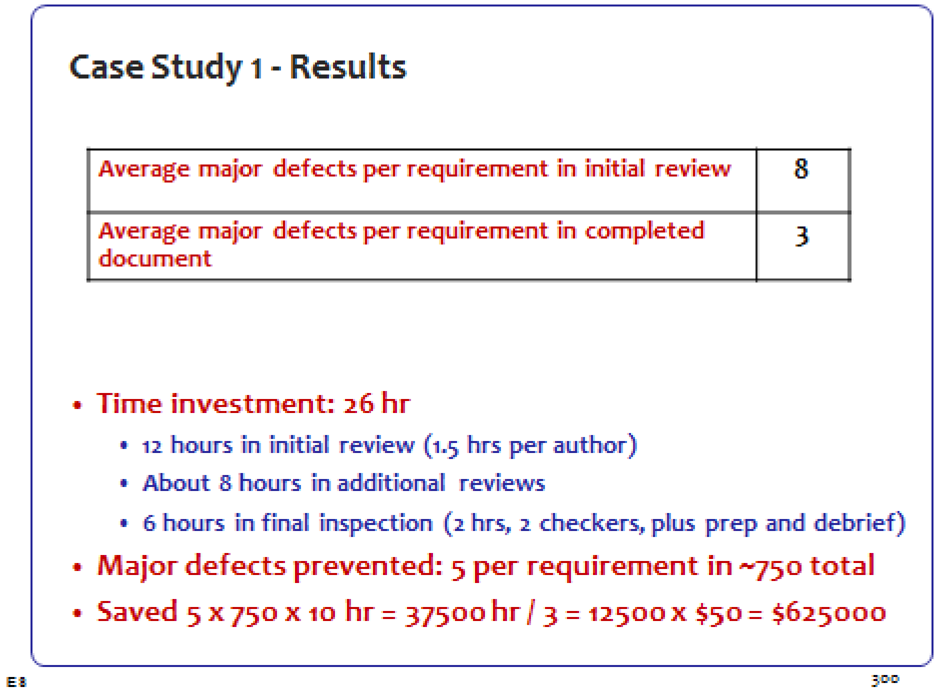


Case Study 1 - Situation

- 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

299



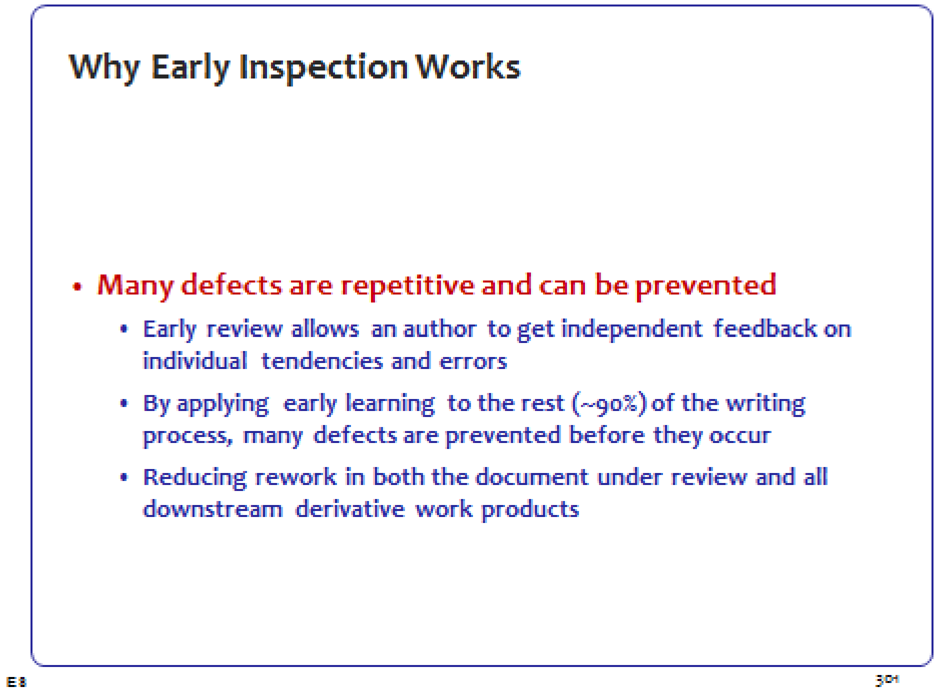
Case Study 1 - Results

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

300

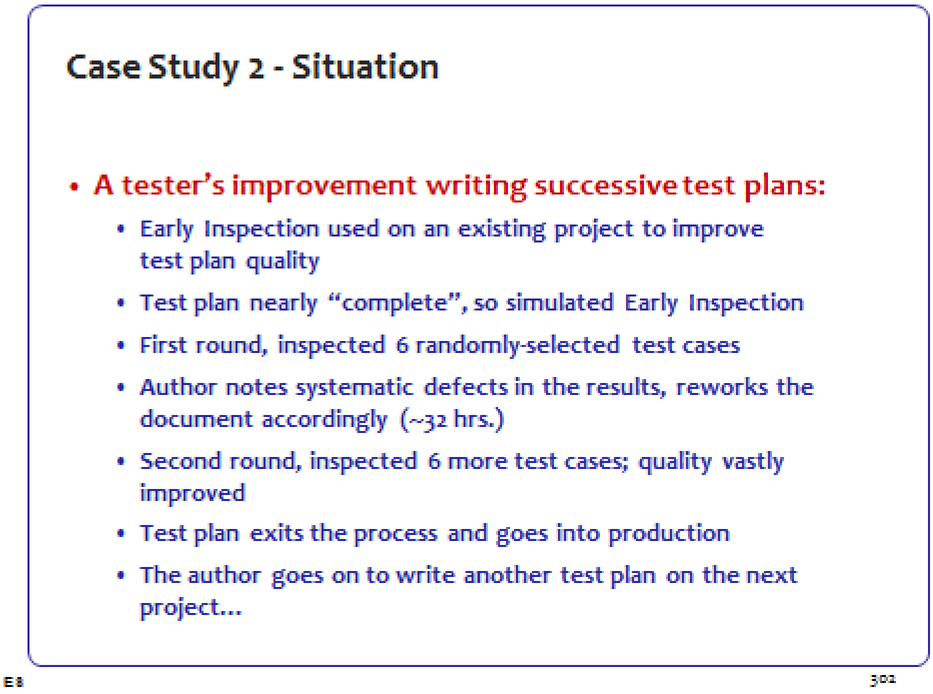


Why Early Inspection Works

- 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

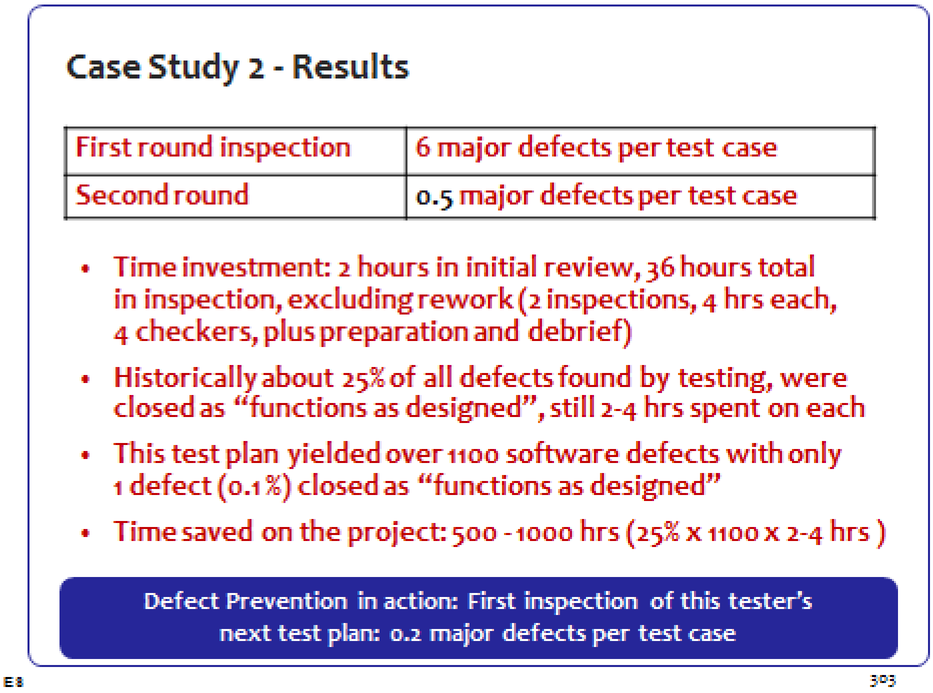
301



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 302



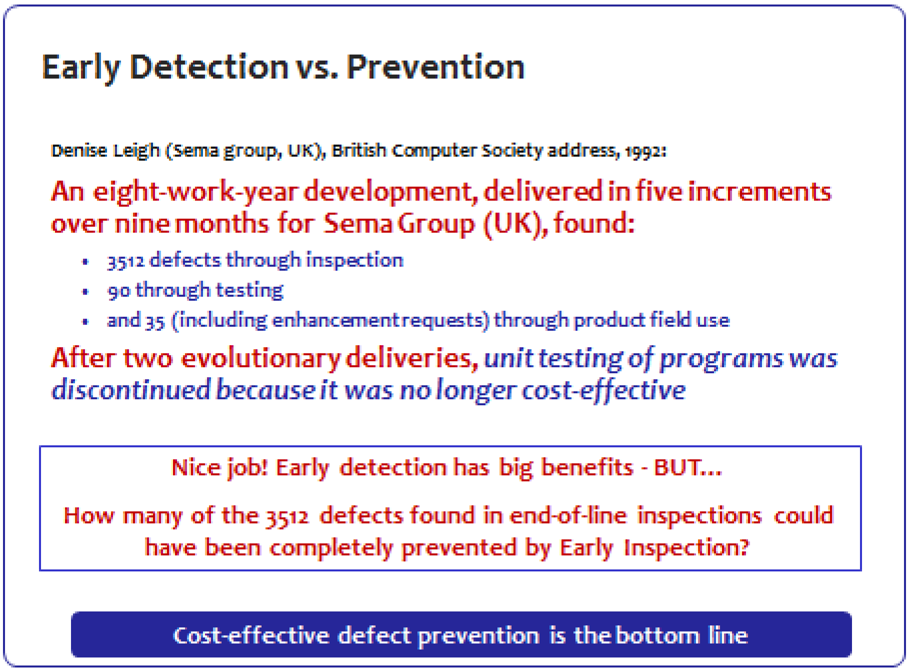
Case Study 2 - Results

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 303



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*

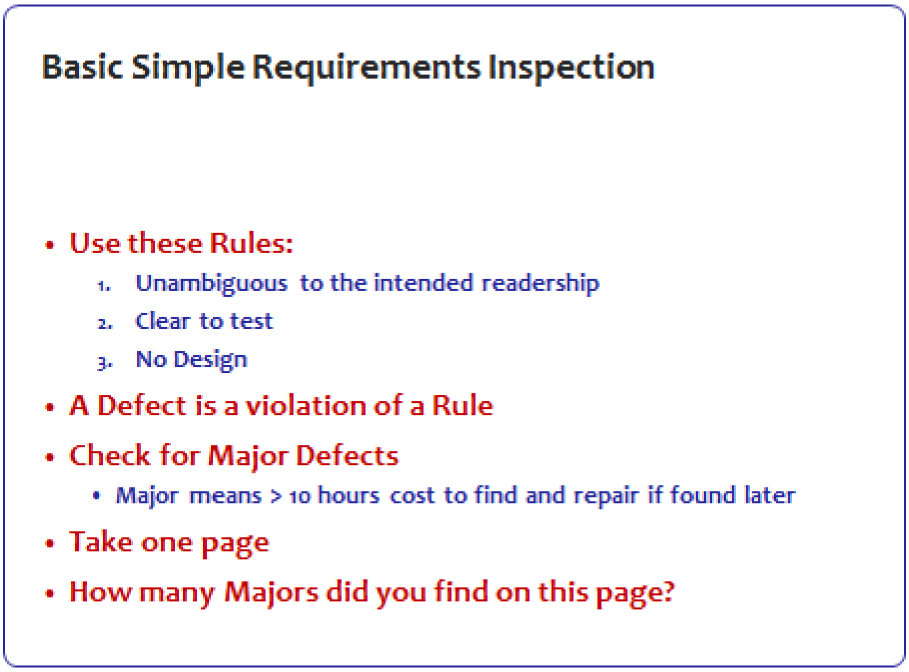
Nice job! Early detection has big benefits - BUT...

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

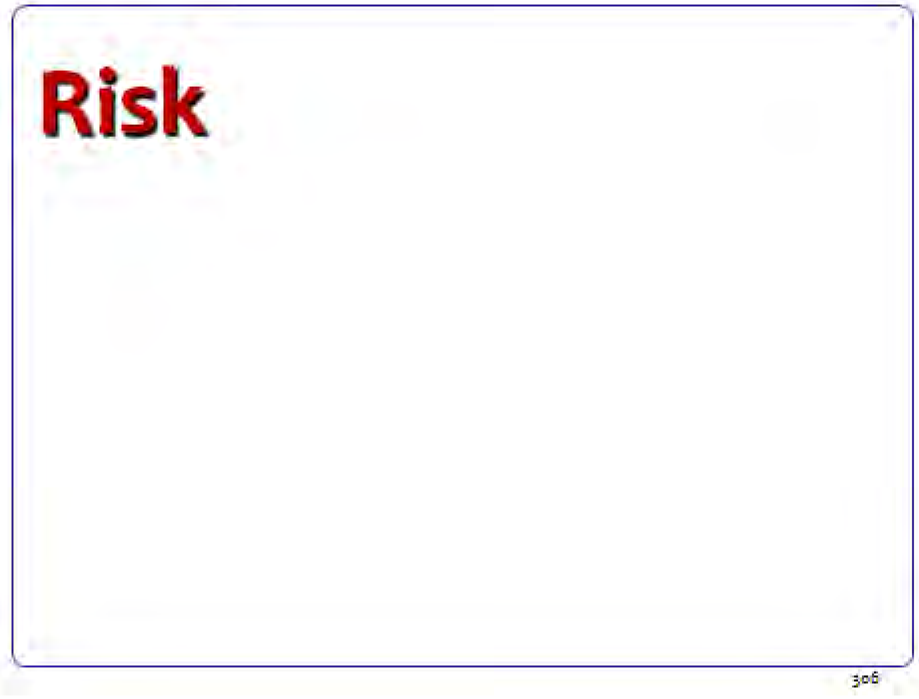
304



Basic Simple Requirements Inspection

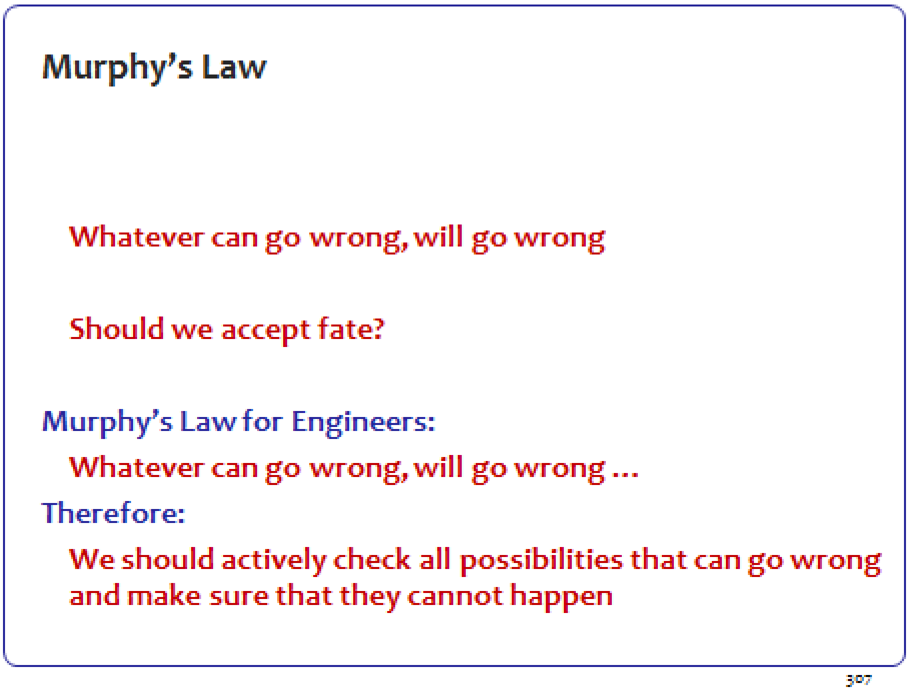
- Use these Rules:
  1. Unambiguous to the intended readership
  2. Clear to test
  3. No Design
- A Defect is a violation of a Rule
- Check for Major Defects
  - Major means > 10 hours cost to find and repair if found later
- Take one page
- How many Majors did you find on this page?

305



Risk

306



Murphy's Law

Whatever can go wrong, will go wrong

Should we accept fate?

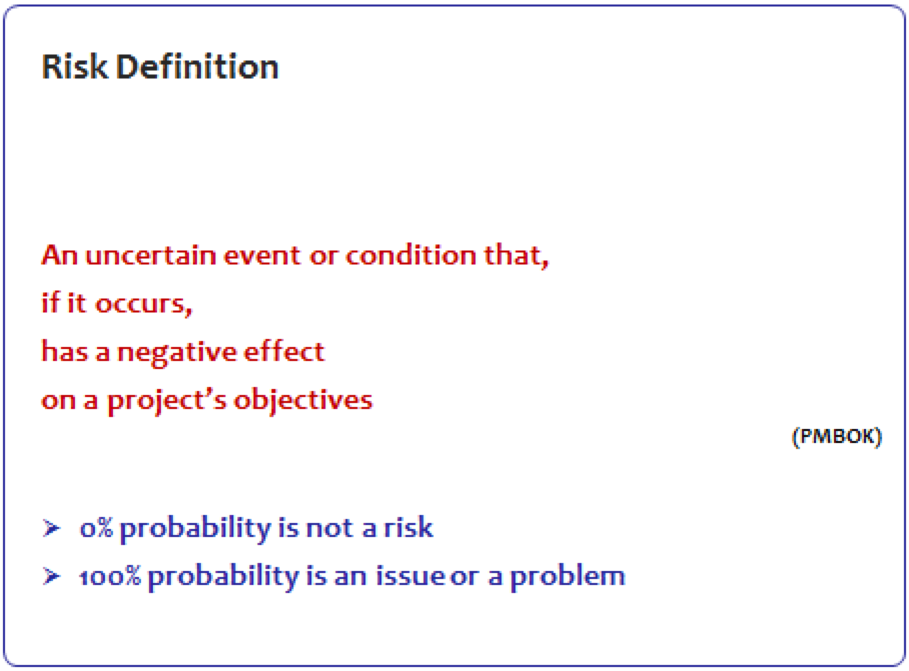
Murphy's Law for Engineers:

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

307



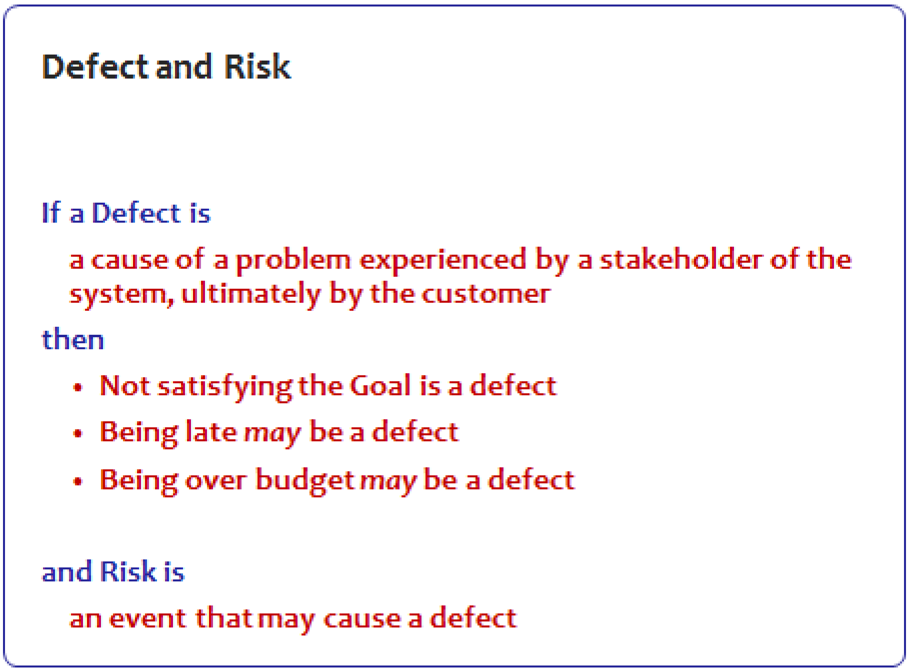
Risk Definition

An uncertain event or condition that,
if it occurs,
has a negative effect
on a project's objectives

(PMBOK)

- 0% probability is not a risk
- 100% probability is an issue or a problem

308



Defect and Risk

If a Defect is

a cause of a problem experienced by a stakeholder of the system, ultimately by the customer

then

- Not satisfying the Goal is a defect
- Being late *may* be a defect
- Being over budget *may* be a defect

and Risk is

an event that may cause a defect

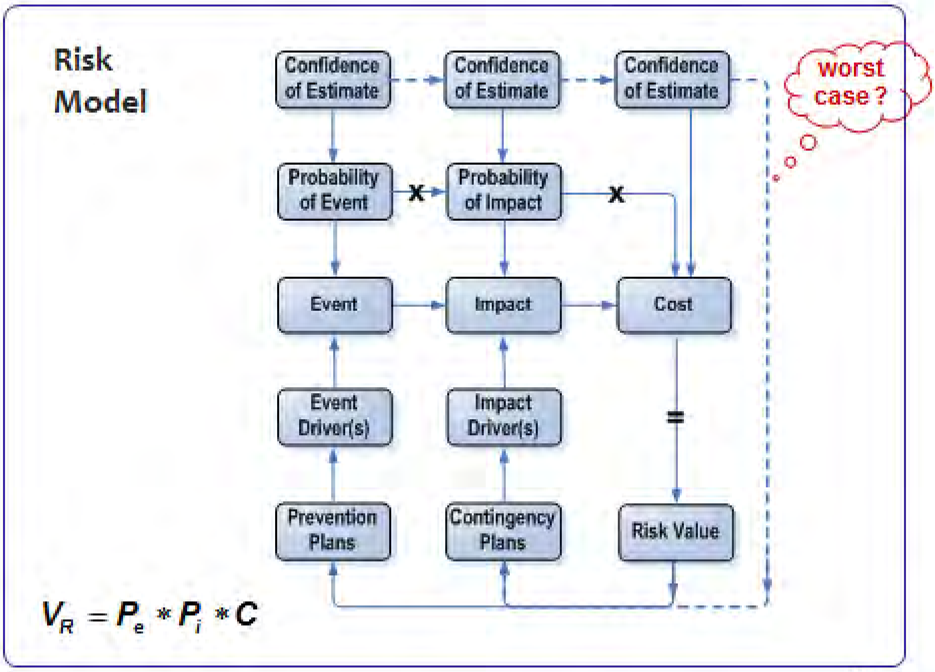
309

Our own Risk

Getting less profit than expected

- Takes more time to develop
- Costs more to develop
- Operating cost more than expected
- Performance less than expected
- Guarantee
- Contract liability
- Legal liability
- Claims

310

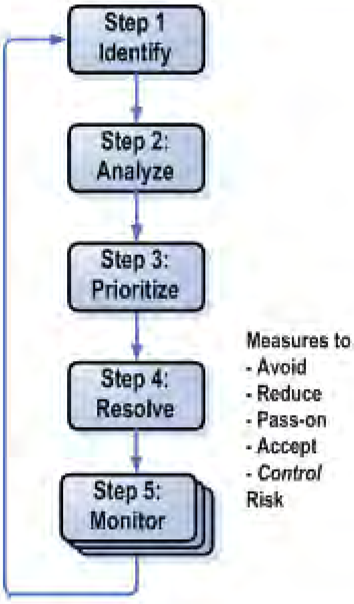


Risk Model

$$V_R = P_e * P_i * C$$

311

Risk Management

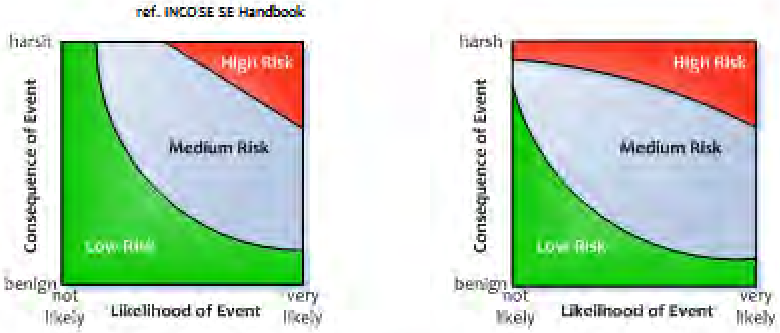


Measures to
- Avoid
- Reduce
- Pass-on
- Accept
- *Control*

Risk

312

Prioritize Risk?



Risk Priority = Likelihood x Consequence ??

313

Mathematical Risk Management can be risky

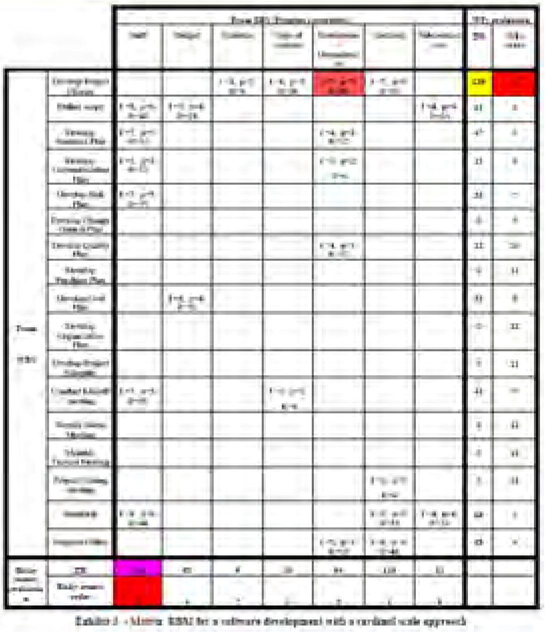


Exhibit 3 - Matrix RBM for a software development with a cardinal scale approach

ref
Carlo Rafele,
David Hillson,
Sabrina Grimaldi

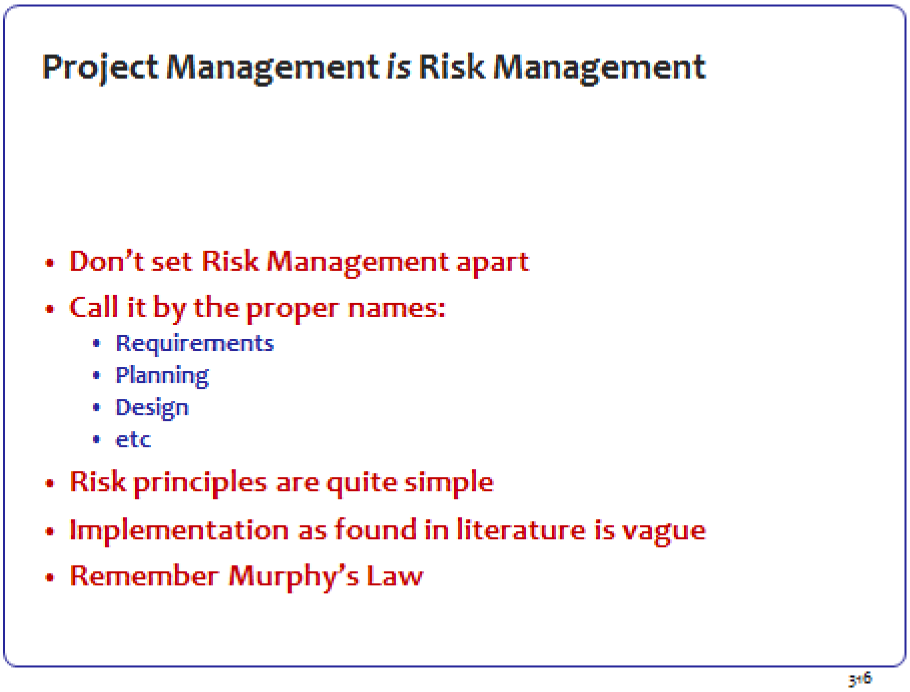
314

Checklists for brainstorm

- Human risk
  - In the project
  - After the project
- Technical risk
  - Can we make it
  - Will it survive
- Environmental risk
  - Example: CE
- Regulatory risk
  - Example: CE
- Consequential risk
- …

Each of these can have
it's own checklist
to trigger the recognition
of real risks

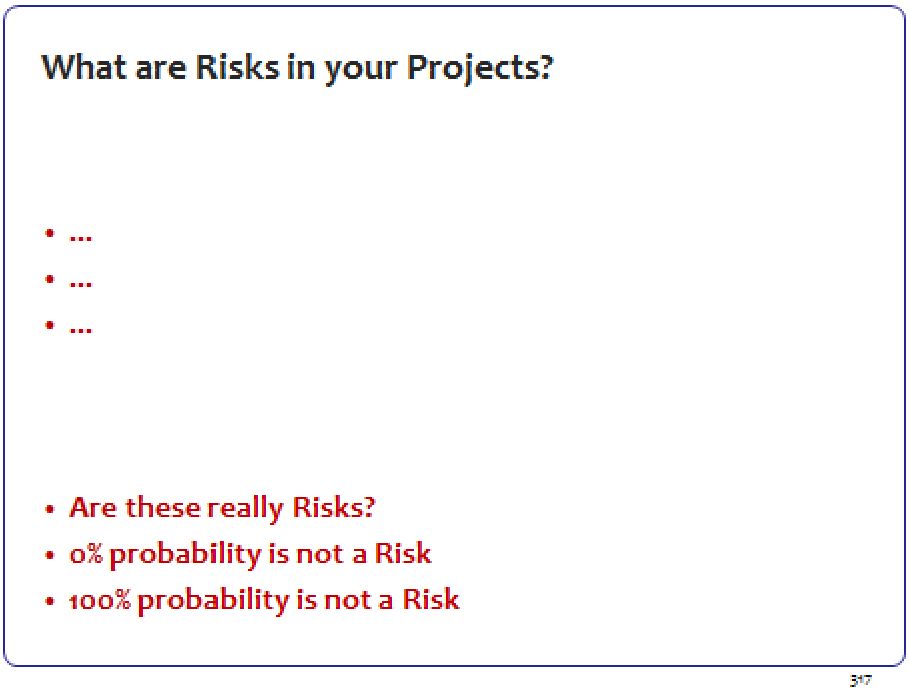
315



Project Management *is* Risk Management

- Don’t set Risk Management apart
- Call it by the proper names:
  - Requirements
  - Planning
  - Design
  - etc
- Risk principles are quite simple
- Implementation as found in literature is vague
- Remember Murphy’s Law

316

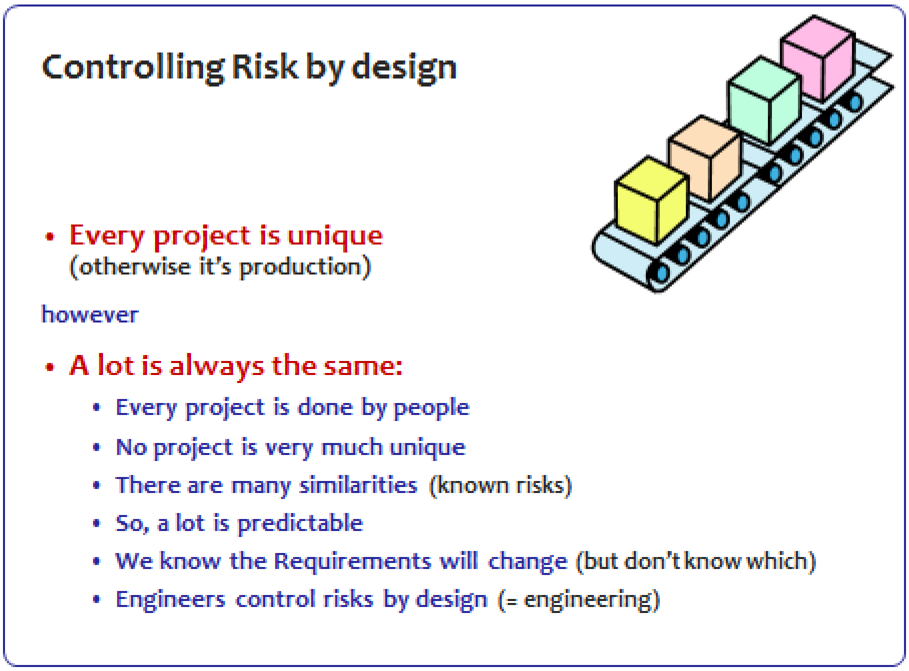


What are Risks in your Projects?

- …
- …
- …

- Are these really Risks?
- 0% probability is not a Risk
- 100% probability is not a Risk

317



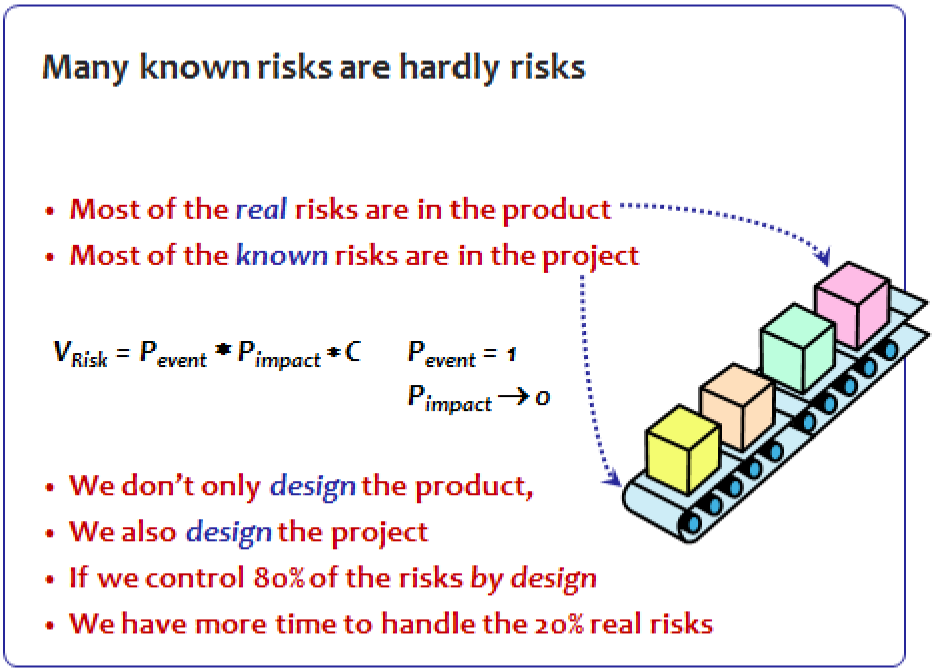
Controlling Risk by design

- Every project is unique
  (otherwise it's production)

however

- A lot is always the same:
  - Every project is done by people
  - No project is very much unique
  - There are many similarities (known risks)
  - So, a lot is predictable
  - We know the Requirements will change (but don't know which)
  - Engineers control risks by design (= engineering)

318



Many known risks are hardly risks

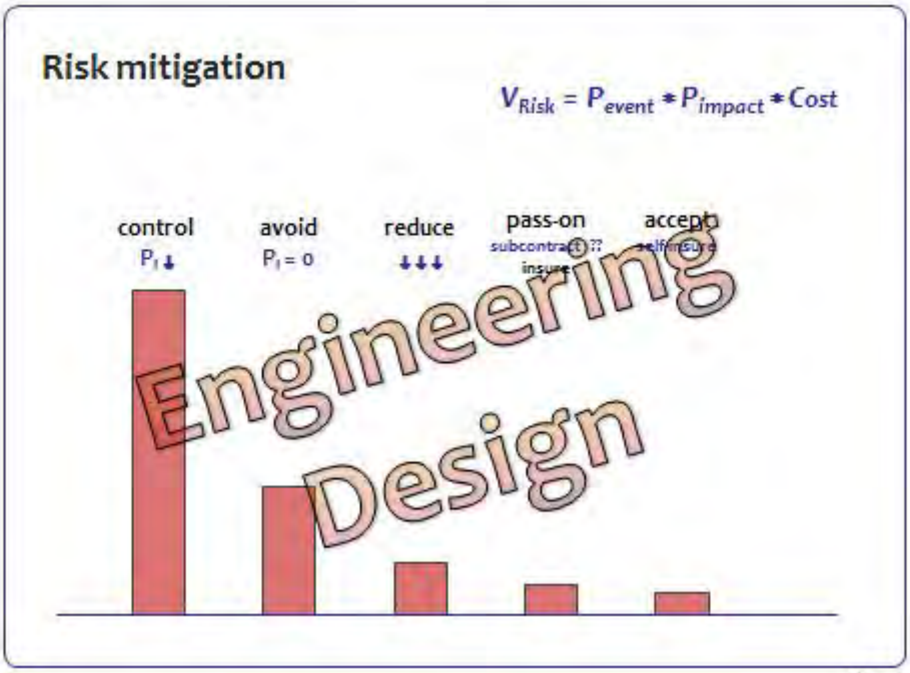
- Most of the *real* risks are in the product
- Most of the *known* risks are in the project

$V_{Risk} = P_{event} * P_{impact} * C$ $\quad P_{event} = 1$

$P_{impact} \to 0$

- We don't only *design* the product,
- We also *design* the project
- If we control 80% of the risks *by design*
- We have more time to handle the 20% real risks

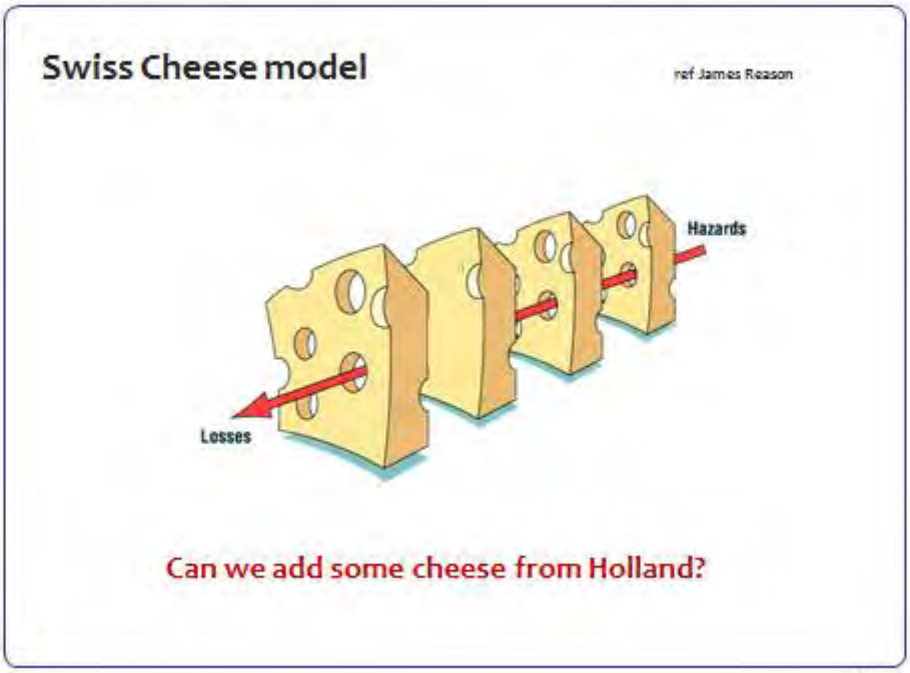
319



Risk mitigation

$V_{Risk} = P_{event} * P_{impact} * Cost$

320

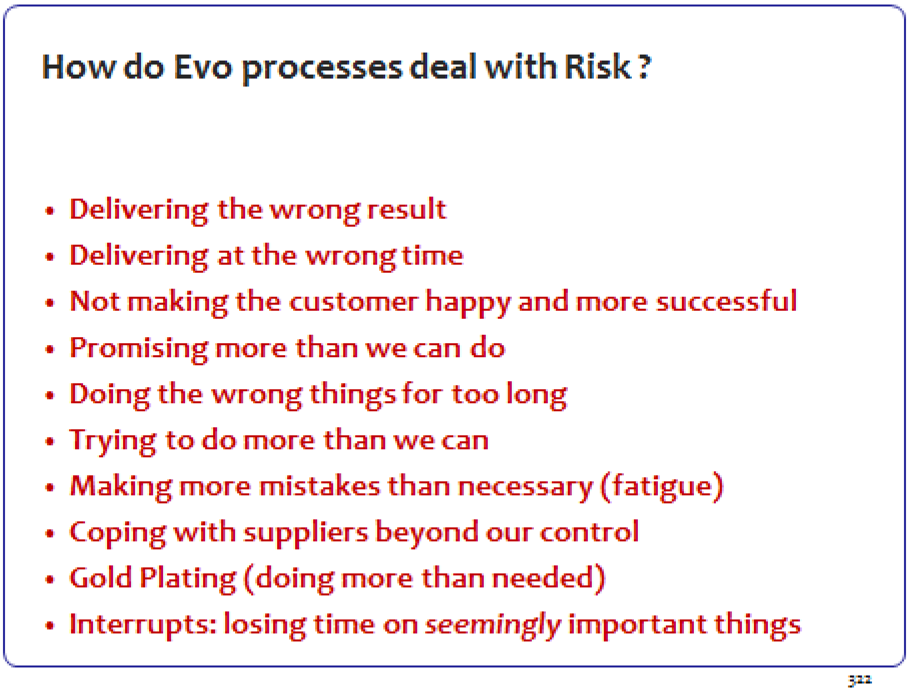


Swiss Cheese model

ref James Reason

Can we add some cheese from Holland?

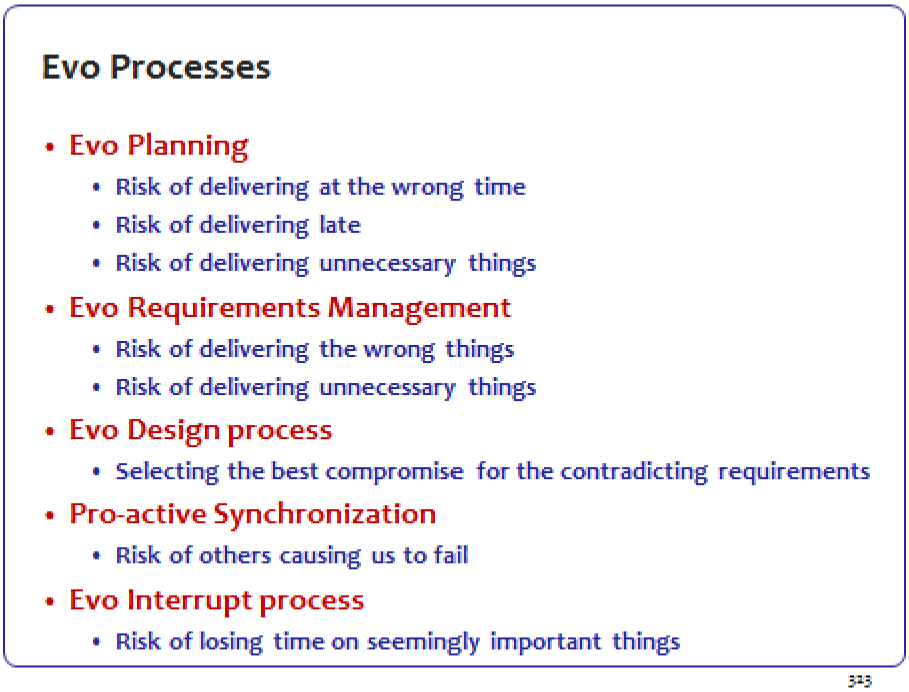
321



How do Evo processes deal with Risk ?

- Delivering the wrong result
- Delivering at the wrong time
- Not making the customer happy and more successful
- Promising more than we can do
- Doing the wrong things for too long
- Trying to do more than we can
- Making more mistakes than necessary (fatigue)
- Coping with suppliers beyond our control
- Gold Plating (doing more than needed)
- Interrupts: losing time on *seemingly* important things

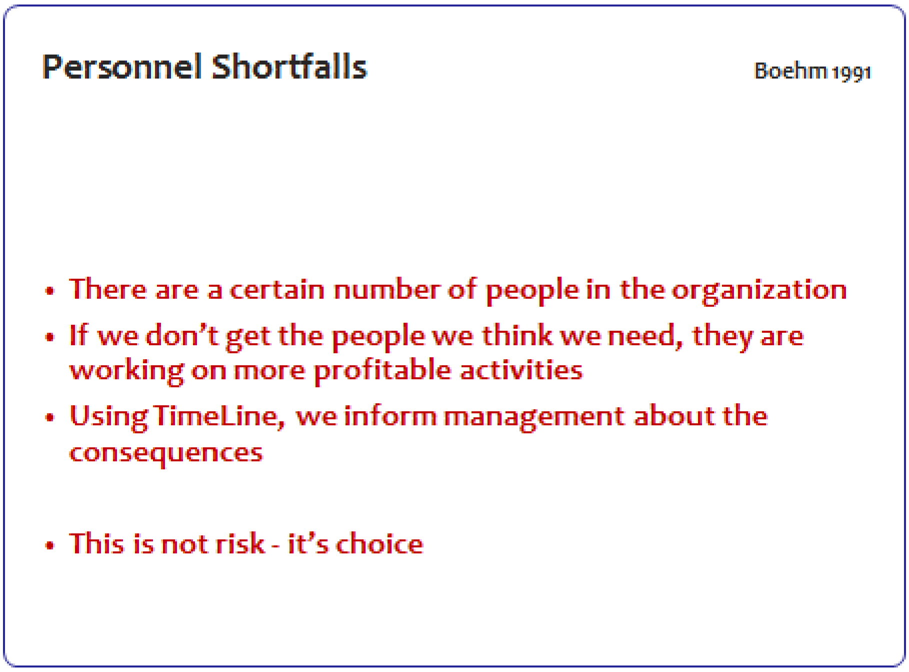
322



Evo Processes

- Evo Planning
  - Risk of delivering at the wrong time
  - Risk of delivering late
  - Risk of delivering unnecessary things
- Evo Requirements Management
  - Risk of delivering the wrong things
  - Risk of delivering unnecessary things
- Evo Design process
  - Selecting the best compromise for the contradicting requirements
- Pro-active Synchronization
  - Risk of others causing us to fail
- Evo Interrupt process
  - Risk of losing time on seemingly important things

323

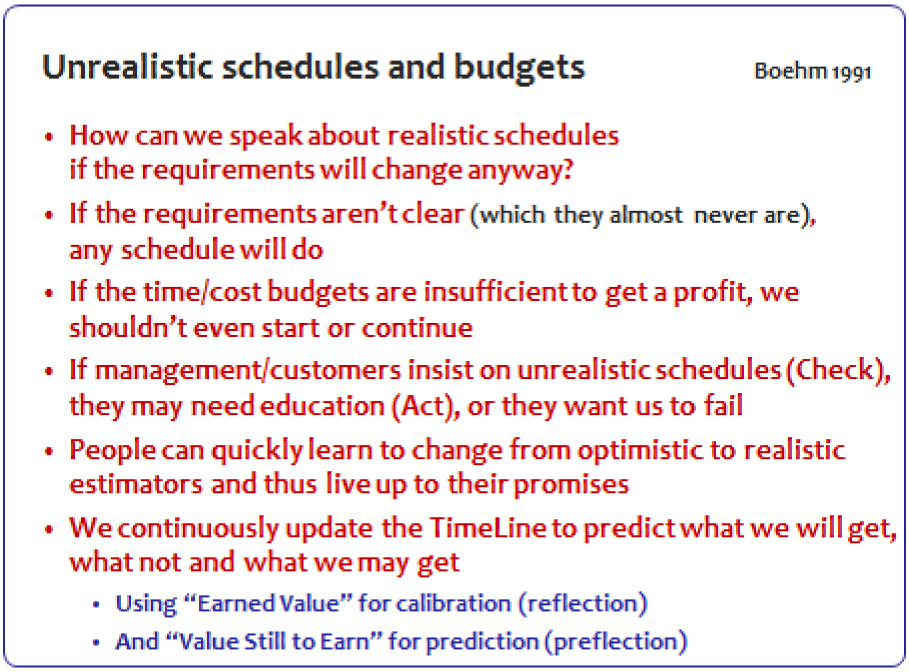


Personnel Shortfalls

Boehm 1991

- There are a certain number of people in the organization
- If we don't get the people we think we need, they are working on more profitable activities
- Using TimeLine, we inform management about the consequences
- This is not risk - it's choice

324

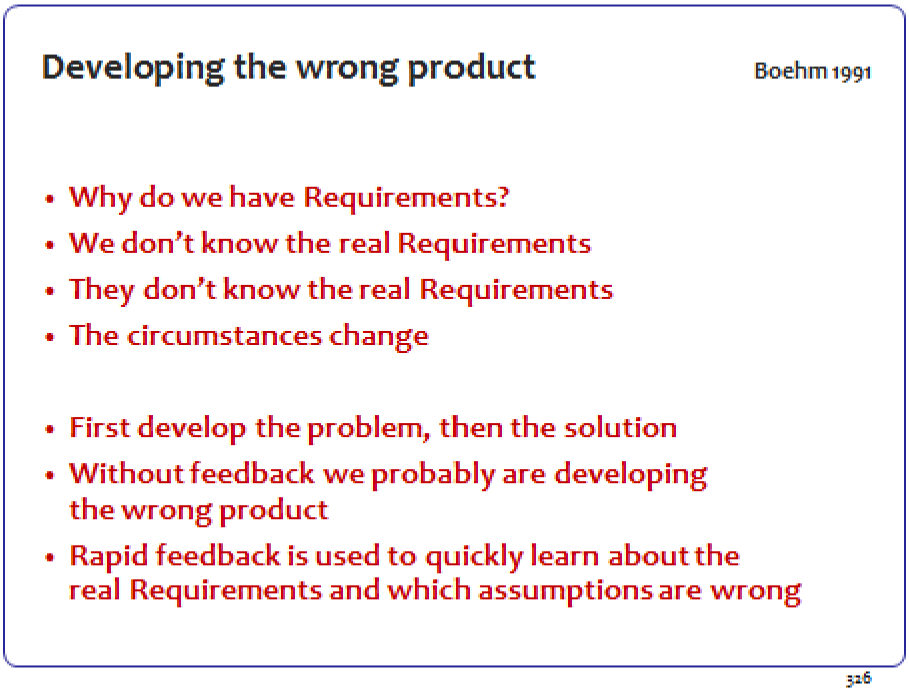


Unrealistic schedules and budgets

Boehm 1991

- How can we speak about realistic schedules if the requirements will change anyway?
- If the requirements aren't clear (which they almost never are), any schedule will do
- If the time/cost budgets are insufficient to get a profit, we shouldn't even start or continue
- If management/customers insist on unrealistic schedules (Check), they may need education (Act), or they want us to fail
- People can quickly learn to change from optimistic to realistic estimators and thus live up to their promises
- We continuously update the TimeLine to predict what we will get, what not and what we may get
    - Using "Earned Value" for calibration (reflection)
    - And "Value Still to Earn" for prediction (preflection)

325



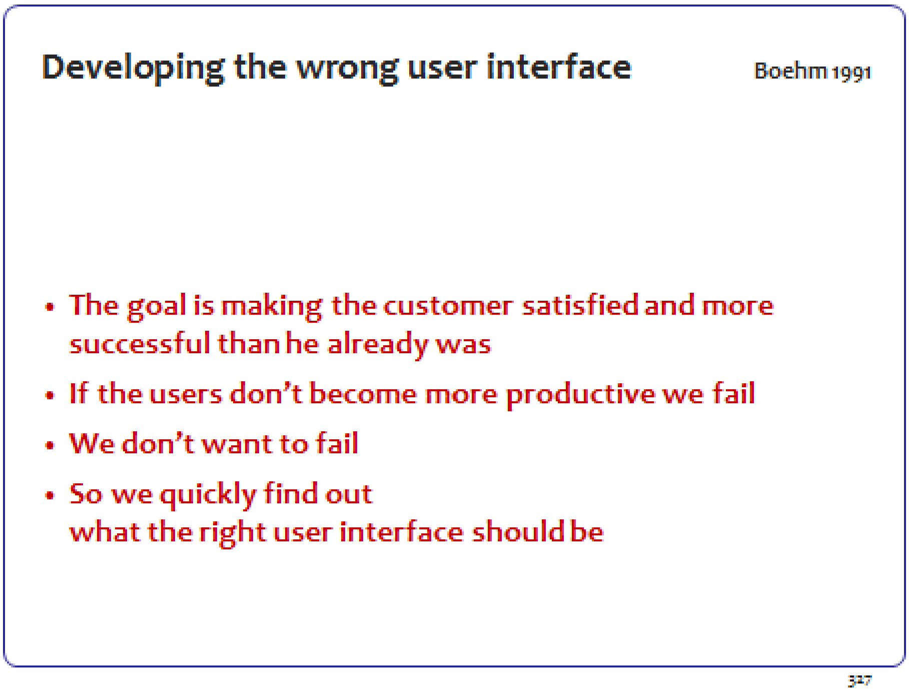
Developing the wrong product

Boehm 1991

- Why do we have Requirements?
- We don’t know the real Requirements
- They don’t know the real Requirements
- The circumstances change

- First develop the problem, then the solution
- Without feedback we probably are developing the wrong product
- Rapid feedback is used to quickly learn about the real Requirements and which assumptions are wrong

326

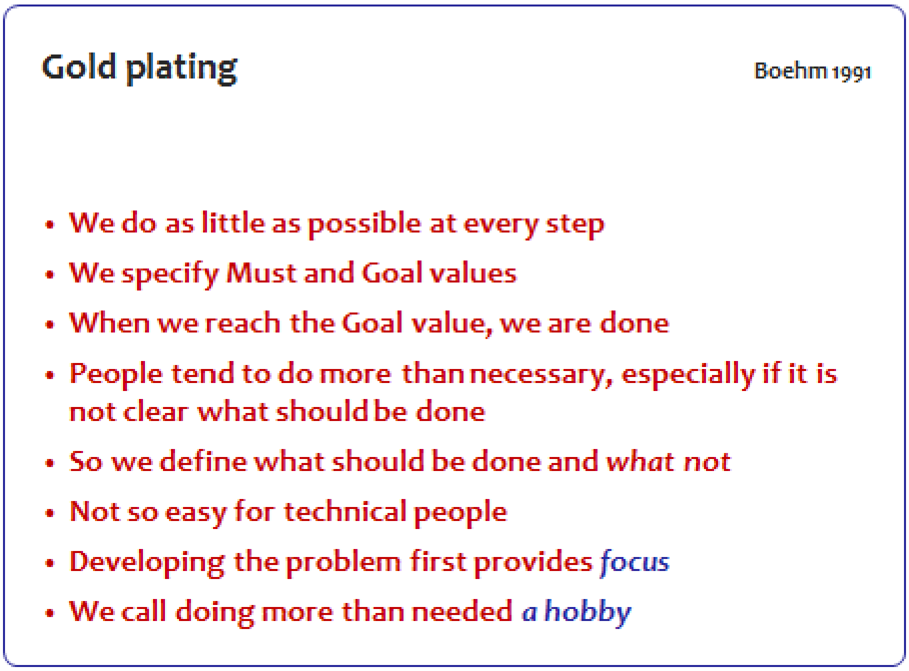


Developing the wrong user interface

Boehm 1991

- The goal is making the customer satisfied and more successful than he already was
- If the users don’t become more productive we fail
- We don’t want to fail
- So we quickly find out what the right user interface should be

327

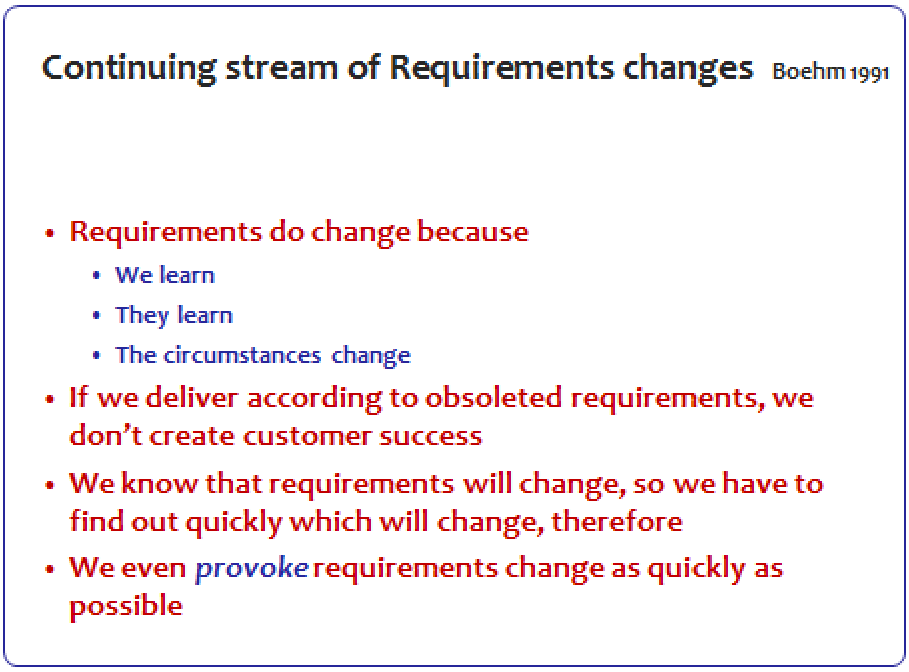


Gold plating

Boehm 1991

- We do as little as possible at every step
- We specify Must and Goal values
- When we reach the Goal value, we are done
- People tend to do more than necessary, especially if it is not clear what should be done
- So we define what should be done and *what not*
- Not so easy for technical people
- Developing the problem first provides *focus*
- We call doing more than needed *a hobby*

328



Continuing stream of Requirements changes

Boehm 1991

- Requirements do change because
  - We learn
  - They learn
  - The circumstances change
- If we deliver according to obsoleted requirements, we don't create customer success
- We know that requirements will change, so we have to find out quickly which will change, therefore
- We even *provoke* requirements change as quickly as possible

329

Problems with externally furnished components

Boehm 1991

- If our FatalDate has come, we have no excuse
- We use Active Synchronization to stay on top

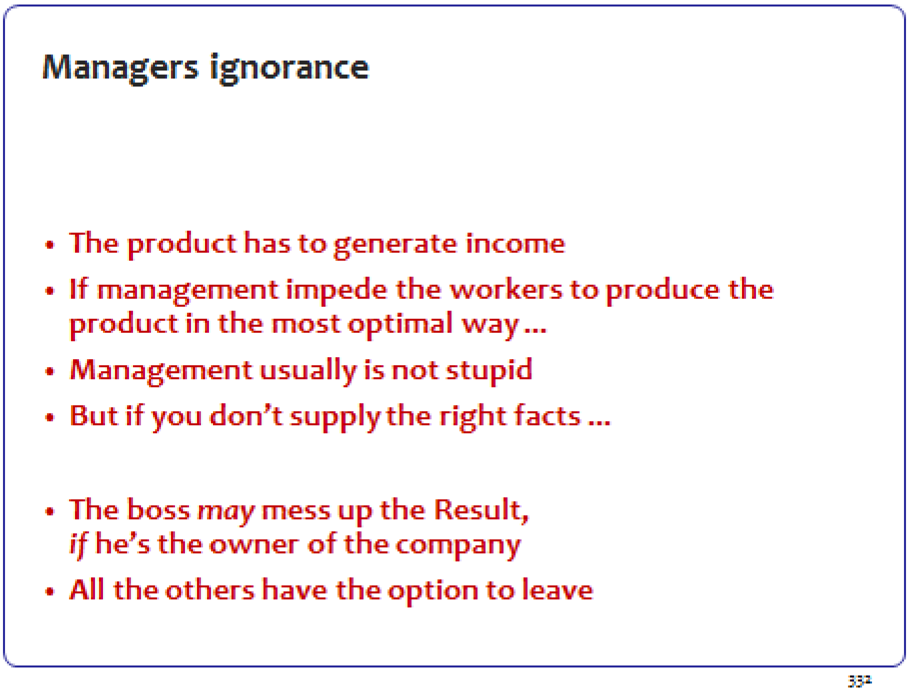
330

Real time performance shortfalls

Boehm 1991

- This is why we have Performance Requirements
- Then we use engineering techniques to make sure the system is according to the requirements
- Real time *should* and *can* be predictable

331

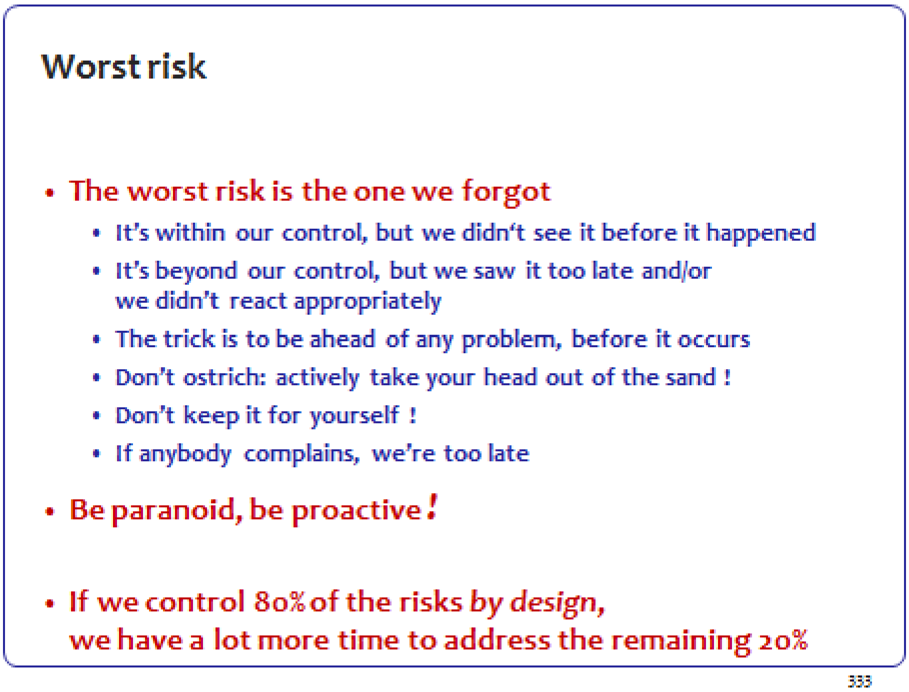


Managers ignorance

- The product has to generate income
- If management impede the workers to produce the product in the most optimal way ...
- Management usually is not stupid
- But if you don't supply the right facts ...

- The boss *may* mess up the Result, *if* he's the owner of the company
- All the others have the option to leave

332

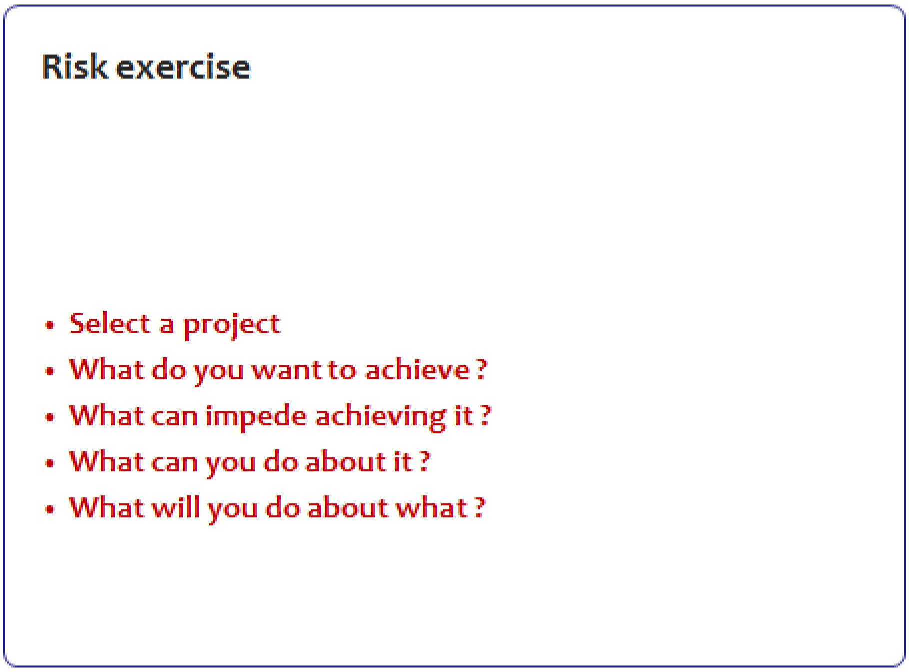


Worst risk

- The worst risk is the one we forgot
  - It's within our control, but we didn't see it before it happened
  - It's beyond our control, but we saw it too late and/or we didn't react appropriately
  - The trick is to be ahead of any problem, before it occurs
  - Don't ostrich: actively take your head out of the sand !
  - Don't keep it for yourself !
  - If anybody complains, we're too late
- Be paranoid, be proactive !

- If we control 80% of the risks *by design*, we have a lot more time to address the remaining 20%

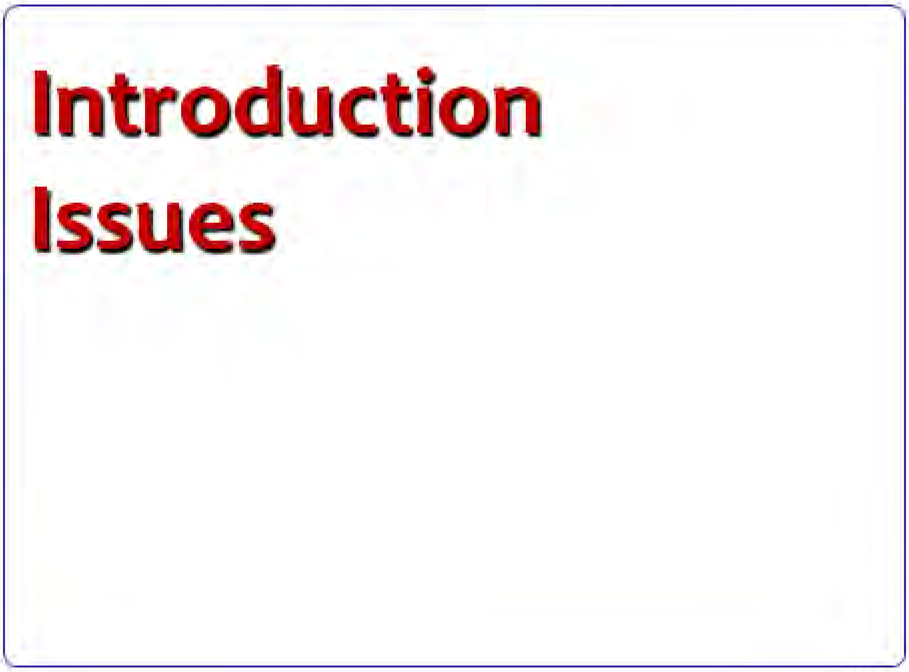
333



Risk exercise

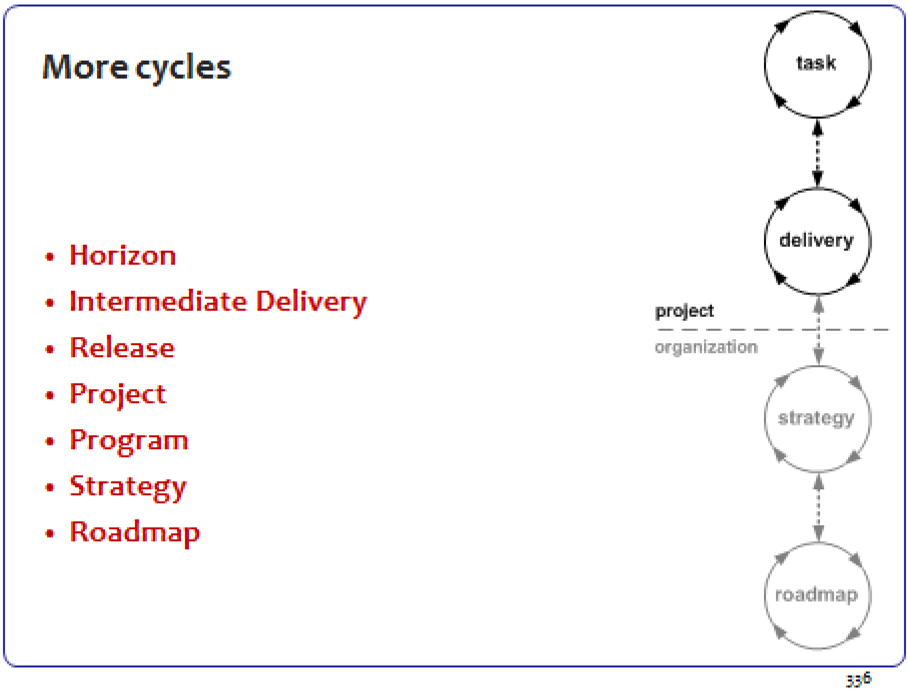
- Select a project
- What do you want to achieve ?
- What can impede achieving it ?
- What can you do about it ?
- What will you do about what ?

334



Introduction Issues

335

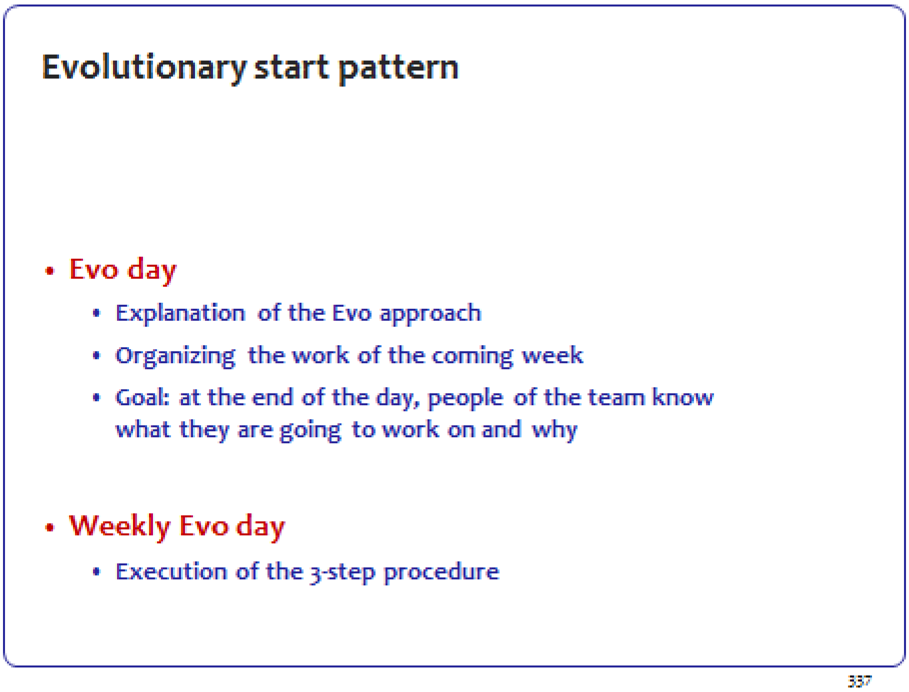


More cycles

- Horizon
- Intermediate Delivery
- Release
- Project
- Program
- Strategy
- Roadmap

task
delivery
project
organization
strategy
roadmap

336



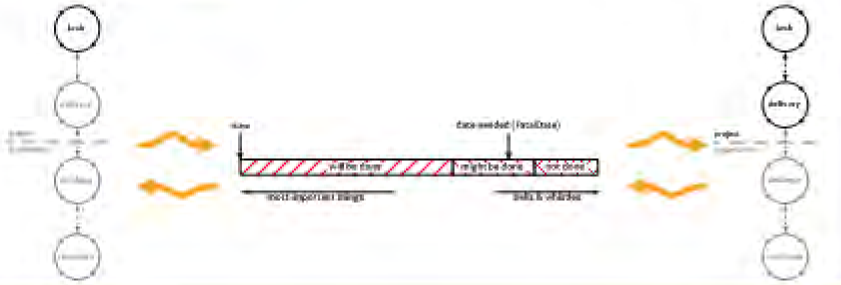
Evolutionary start pattern

- Evo day
  - Explanation of the Evo approach
  - Organizing the work of the coming week
  - Goal: at the end of the day, people of the team know what they are going to work on and why
- Weekly Evo day
  - Execution of the 3-step procedure

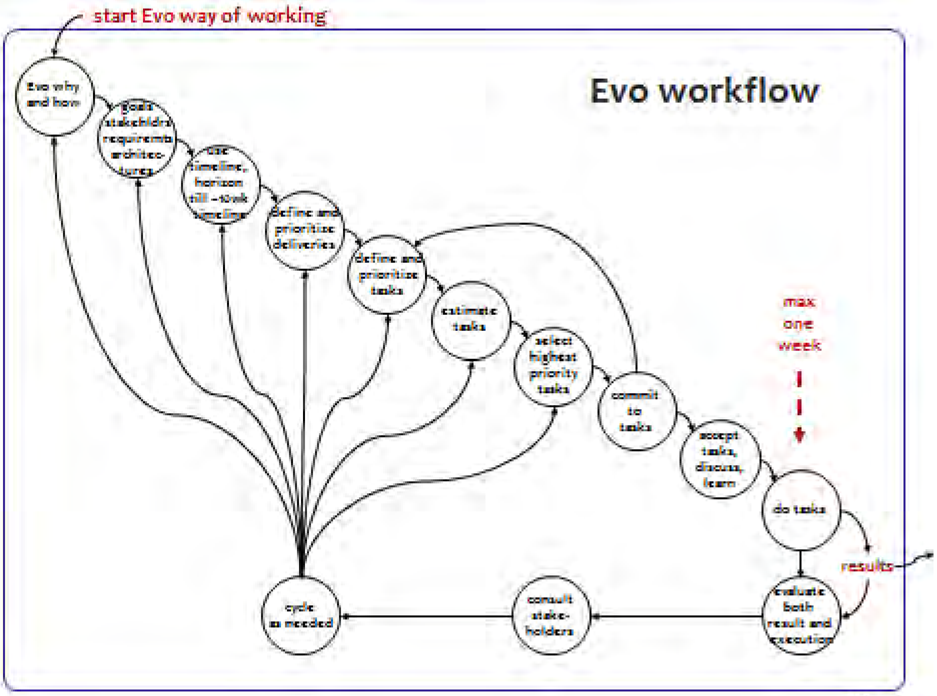
337

Evolutionary introduction pattern

1. Introducing *Tasks* → Short term view
   How to organize the work
2. Introducing *TimeLine* → Longer term view
   The *design* of the project
3. Introducing *Deliveries* → Connecting long and short
   Focusing on Results



338

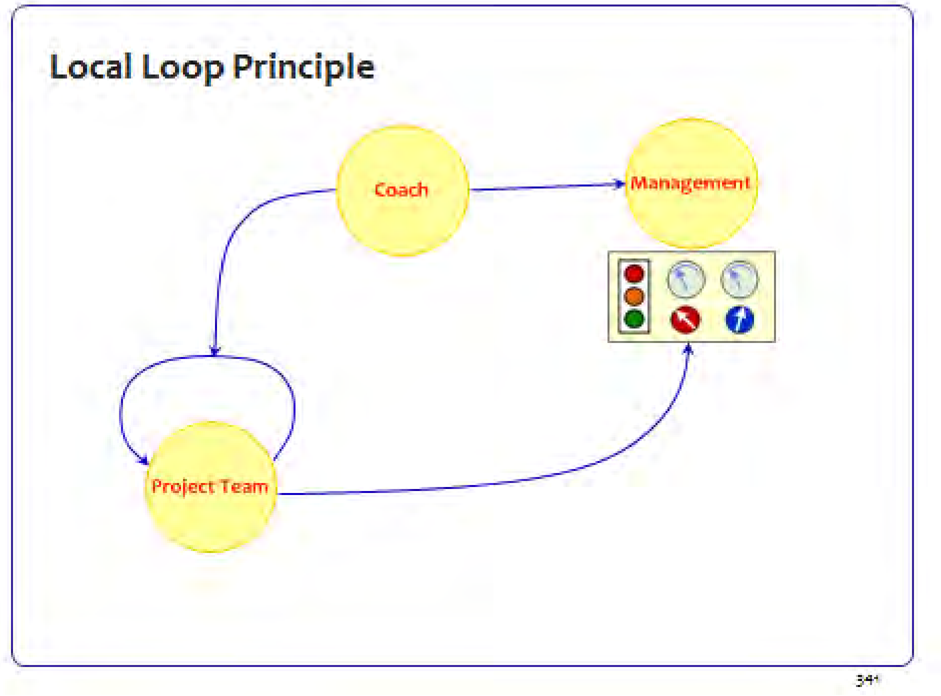

start Evo way of working
Evo why and how
goals stakehldrs requiremts architec-tures
use timeline, horizon till ~10wk timeline
define and prioritize deliveries
define and prioritize tasks
estimate tasks
select highest priority tasks
commit to tasks
accept tasks, discuss, learn
do tasks
max one week
results
evaluate both result and execution
consult stake-holders
cycle as needed

Evo workflow

339

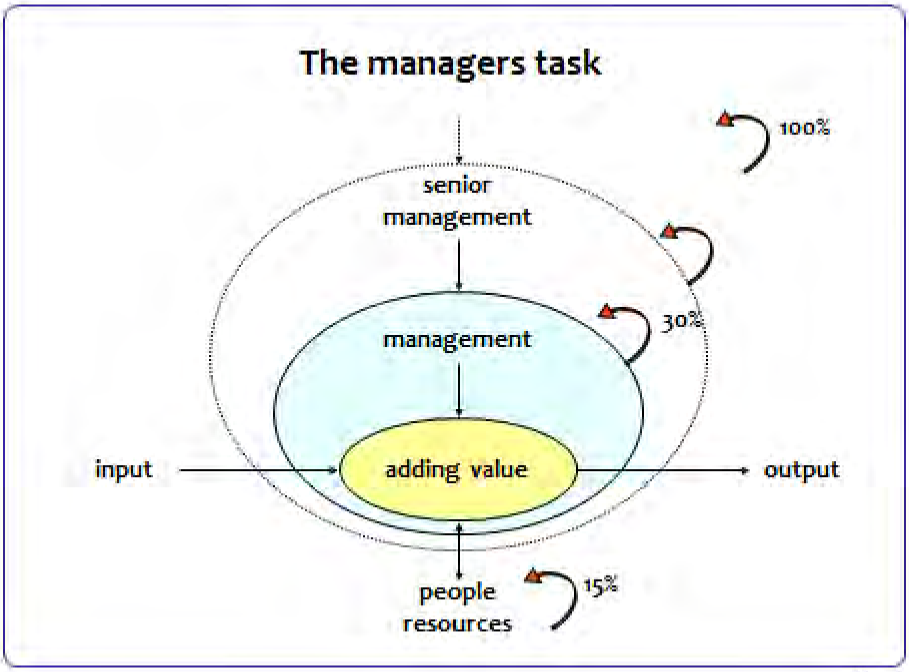
Finally

340



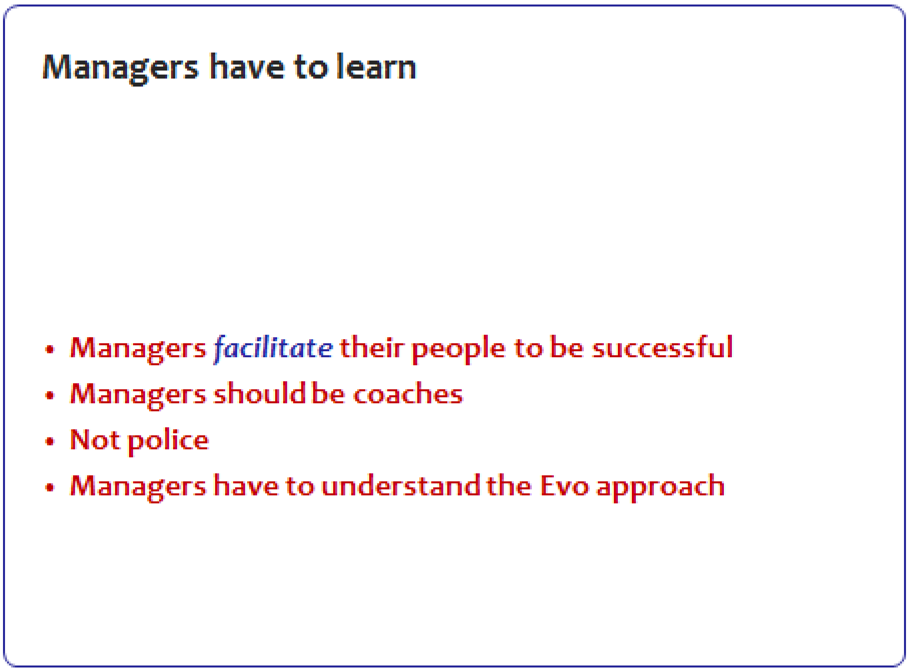
Local Loop Principle

341



The managers task

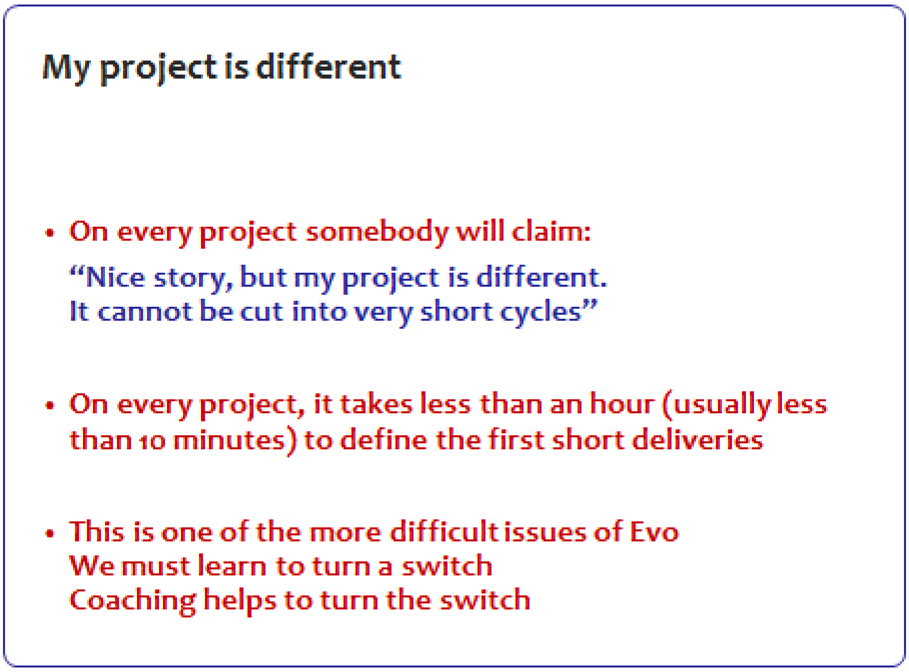
342



Managers have to learn

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

343



My project is different

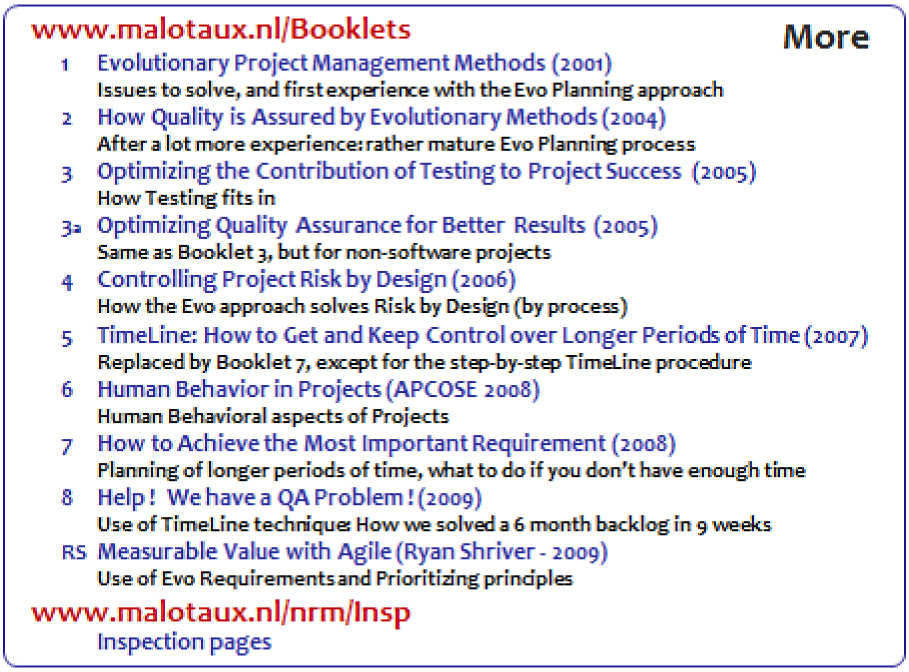
- On every project somebody will claim:

  "Nice story, but my project is different.
  It cannot be cut into very short cycles"

- On every project, it takes less than an hour (usually less than 10 minutes) to define the first short deliveries

- This is one of the more difficult issues of Evo
  We must learn to turn a switch
  Coaching helps to turn the switch

344



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

345

What now ?

346