**SDM**

System Design and Management

**Keio-SDM - Yokohama**
**22 - 25 February 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 specializing in optimizing project performance. He has over 35 years experience in designing electronic 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. To this effect, Niels developed an approach for effectively teaching Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he taught and coached over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, India, Ireland, Israel, Japan, Romania, South Africa and the US, which led to a wealth of experience in which approaches work better and which work less in practice.

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

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

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

**N R Malotaux**
Consultancy

Niels Malotaux
project coach

*Result Management*

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

## How to get the Right Result at Right Time

Niels Malotaux

**N R Malotaux**
Consultancy

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

---

**Niels Malotaux**

*Result Management*

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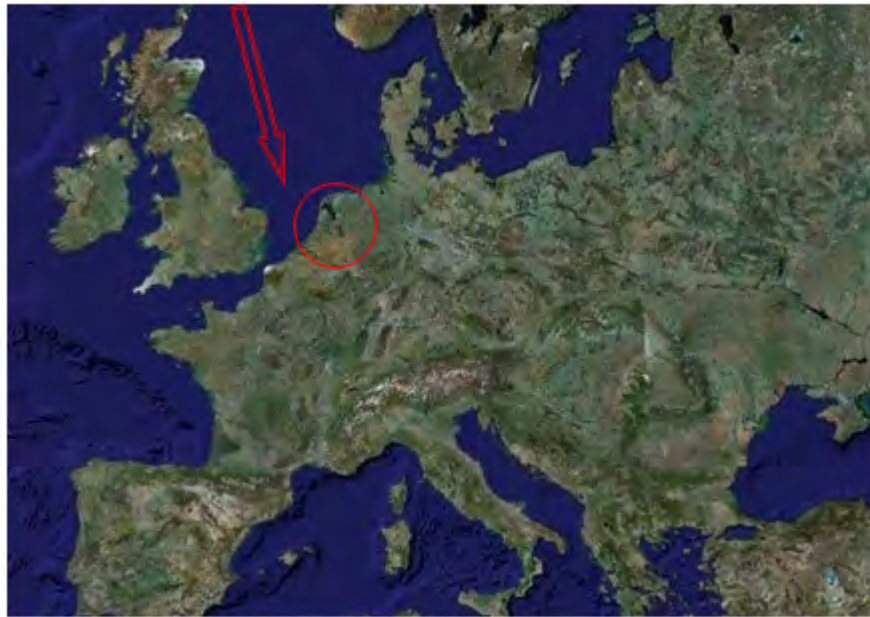  - Requirements Engineering
  - Reviews and Inspections

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

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time



9



10

## Who are you ?

11

## Projects

- Who's working in projects?

- What is a project ?

- Who is a Project Manager ?
- Who is a Systems Engineer ?
- Who is something else ?

- Who has Project Result responsibility ?

12

More information:
www.malotaux.nl

## How good are you in your current work?

- Average?
- Better than average?
- Below average?

13

## Predictable Projects ?

- Any problems with projects ?

14

## Types of project ?

- Process development
- Product development
- Software development
- Systems
- Systems of Systems
- … ?

15

## Time Important ?

- Yes !
- Extremely yes !
- Yes, for our business case; no, customers don't care much
- Yes, it saves cost
- Yes, it's a requirement
- Yes, it's our commitment to the customer

(previous workshop)

16

## Things to Improve

- Time is first priority after product quality
- How to make project management more efficient
- Systems Architecture and Design
- Project Management
- Time and Cost Management
- Documentation
- Predictable Schedule
- Communication
- Quality of requirements and final products
- Project quality

(previous workshop)

17

## Quality On Time

(previous workshop)

| Successful | On Time | Need improvement |
|---|---|---|
| Quality and cost OK | 81% overrun | Time is first priority |
| Normally yes, with good plan and people | Normally yes | More efficient PM |
| Yes | Yes | Improve Design Competence and Planning |
| Yes, customer happy | Yes | Time/Cost Management |
| Yes | No, many changes, component delay (Customers don't mind) | Documentation, Schedule |
| Result OK | Later (req/staff changes) | Predictable schedules |
| Some | Later | Schedule |
| No. Long project, change req → bad product quality | Maybe | Quality of requirements and results |
| Usually | Usually or bit later | Project quality |

18

## How to improve

- CMMI
- Project planning tool
- Skill of team on Project Management
- Improve design competence
- Good planning
- Don't know
- Breaking down to small tasks
- Stable staffing
- Stick with requirements; better prioritizing
- Using PMBOK
- Iterative delivery & customer cooperation

(previous workshop)

19

## 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?

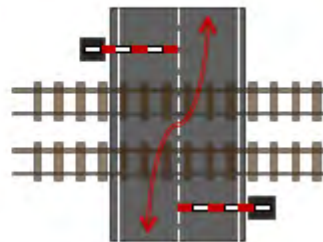
- What is the Right Result?
- What is the Right Time?

20

## Not every project is successful
(at first)

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

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

## AHOB (Automatic Half Barrier Crossing)

**ADOB**
(Automatic Double Barrier Crossing)

1 train every 4 minutes

Few years of trouble
before some stability

At >22°C still trouble

Why it didn't work
is irrelevant

What we deliver
should simply work
Is that so difficult?

## Tunnels in the Netherlands

- 18-02-2008: tunnels open for traffic.
  A73 (42km) is now complete ! (decision 1975)
- V&V said "NO!" → Lots of trouble
- 03-03-2008: tunnels are finally open, after
  safety tests concluded
- 10-03-2008: tunnels are finally open, after
  safety tests concluded
- 06-06-2008: Coming months we're working
  on completion of the A73 tunnels
- 24-09-2008: start of completion in
  January 2009 and/or April 2009
- 24-09-2009: Completion runs as planned.
  Tunnels will be closed from 1 October.
  1 December the tunnels will finally open
- 28-01-2010: Tunnels will have weekend closures
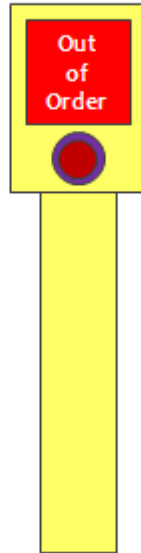  for "regular maintenance"

**Incredible**
**Public Transport**
**Chip-Card**

Out
of
Order

15

**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

16

## Do we mind?

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

27

## Can we afford it?

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

28

## 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**

19

## Predictable Projects                 How to Get Quality On Time

- Introduction
- Is Culture an Issue ?
- Quality On Time
- Human Behavior
- Project Life Cycles
- Evolutionary Principles
- Evolutionary Planning
- Business Case

- Stakeholders & Requirements
- Design & Architecture
- Risk
- V&V / Testing / QA
- Reviews & Inspections
- Metrics
- Some Management Issues
- Introduction Issues

30

## 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

31

## What are you providing your customer ?

- 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
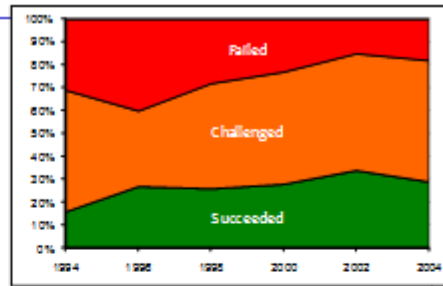
- Who is your customer ?
- What does he need ?
- When does he need it ?
- Will he be happy with it ?
- Will he be more successful ?
- Can the customer afford it ?
- Is it win-win ?

32

More information:
www.malotaux.nl

## What's wrong with projects?

**2/3 of IT projects still fail on Quality On Time**



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

Standish Group International

However
- "Succeeded" projects actually were late from the beginning: Management told that they multiplied "best guess" by 2.5
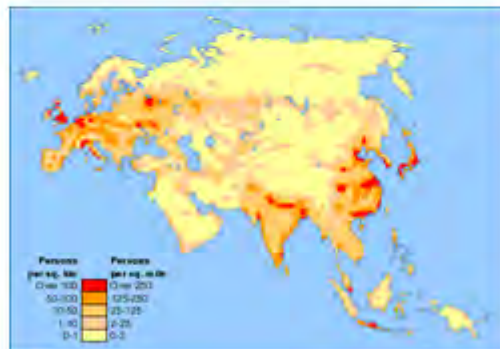- Perhaps the "Failed" projects were rightly killed early

33

## Top 5 success factors:

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

Standish Group International

34

# Is Culture an Issue ?



# 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

## Cultural differences ?

influences on project results ?

| Dutch | Japanese ? |
|---|---|
| • 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 | |

37

## Things I heard

- **Authority**
  - Boss is always right
  - Teacher is always right
  
  They are just doing their best. Lot of experience, but are they perfect ?

- **Group is important**
  - Project team is a group; organization is a group
  - Self ?

- **Group is responsible**
  - No personal responsibility ?
  - Should we hide for responsibility ?

38

## Things I heard (2)

- Losing 'face'
  - We are not perfect, but the customer should never find out
- Cannot say 'No'
  - How do you then say 'no' ?
    (in Holland we say: "Yes, but ...")
- Is that clear? - Yes
  - 'Yes' isn't always 'Yes'
  - If you don't understand:
    - Is the teacher unclear ?
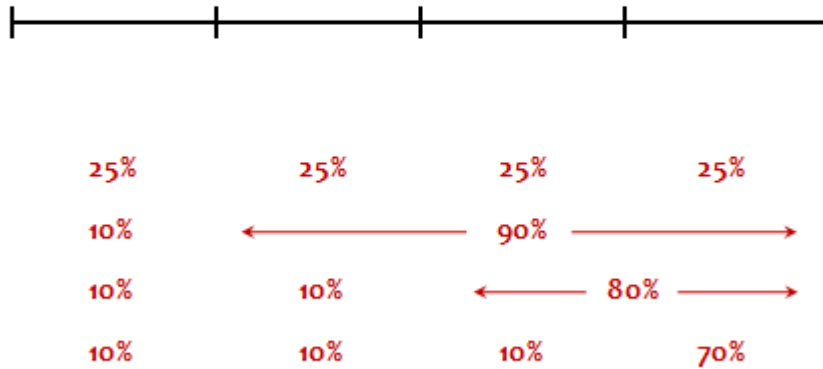    - Am I stupid ?

39

## The boss is always right

- Is he or she ?

- Afraid for losing 'face' ?
- How about losing face *invisibly* ?
  (you don't *say* it, but you *know*)
- Would you like that if you were a boss ?
- How do we tell, without losing 'face' ?
- Should we ?
- Is it *my* culture ?

40

## 4 week project

| 25% | 25% | 25% | 25% |
| 10% | ← | 90% | → |
| 10% | 10% | ← | 80% → |
| 10% | 10% | 10% | 70% |

41

## Is culture a risk for projects?

42

## If we want to be winners

- 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 it
- Let's help each other
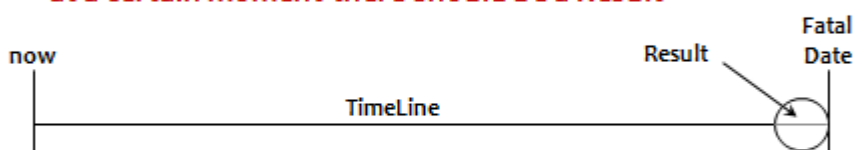- We cannot help each other, if we don't know

43

# Quality on Time
## The Right things at the Right time

44

## Quality On Time

- Whatever we do in a project,
  at a certain moment there should be a Result

now ————————— TimeLine —————————→ Result → (Fatal Date)

- How do we get the Right Result at the Right Time?
- Or shorter: Quality On Time
- What the Customer needs, when he needs it,
  to earn more than we need

45

## Quality On Time

- What is Quality?
- What is On Time?

46

## 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

47

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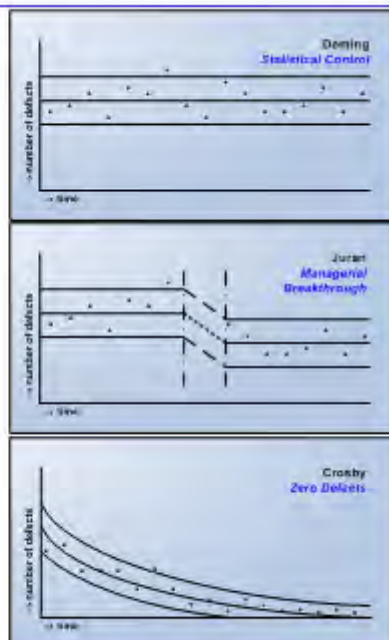
- **Shewhart** - Economic Control of Quality 1930
- **Deming** - Japan 1950, Out of the crisis 1986
- **Juran** - Japan 1954, Quality handbook 1951
- **Crosby** - Zero Defects 1961, Quality is Free 1979
- **Imai** - Kaizen 1986, Gemba Kaizen 1997

48

## Deming - Juran - Crosby



49

## Deming

- Quality comes not from inspection (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*

50

## 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

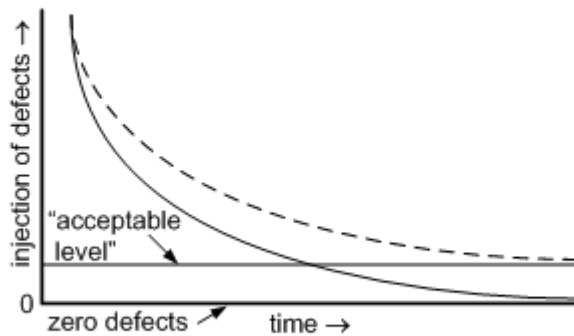- The purpose is customer success (not customer satisfaction)

Added by Philip Crosby Associates, 2004

51

## 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 defect free 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 the failure, we don't learn)
- If we deliver a result, we are sure it is OK and we will be highly surprised when there will be a defect after all
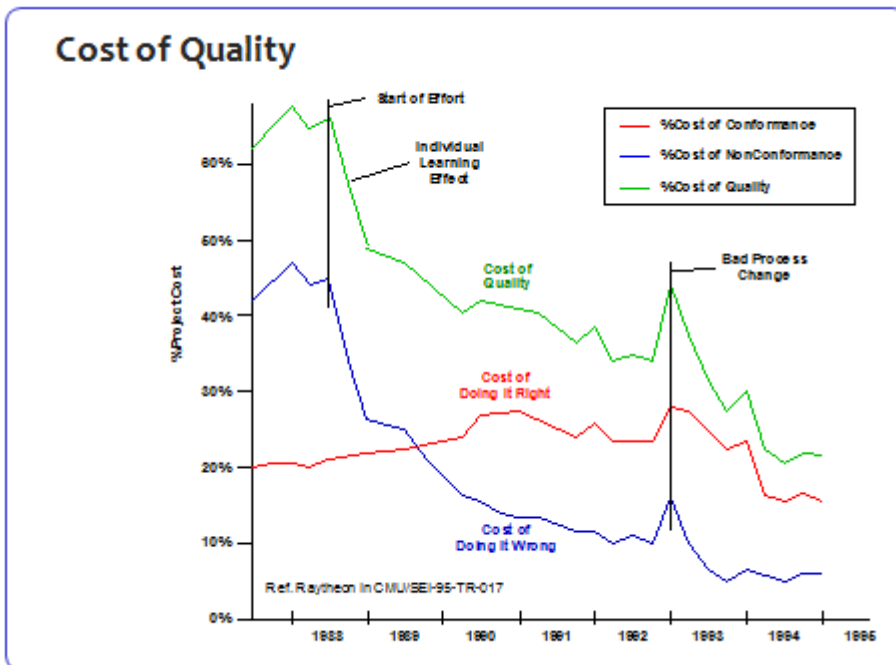- We constantly, actively improve *what* we do and *how* we do it

## Productivity gains



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

56
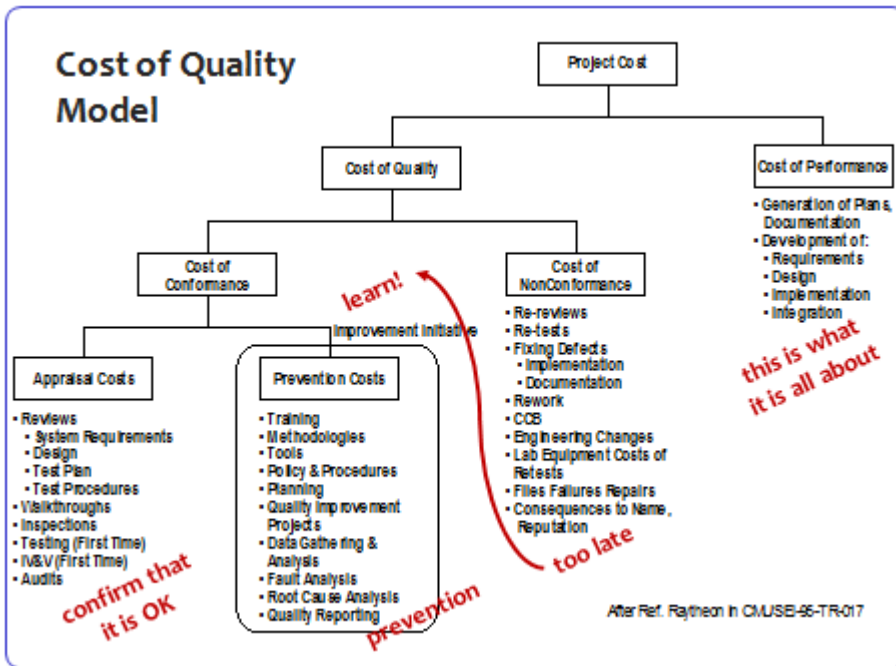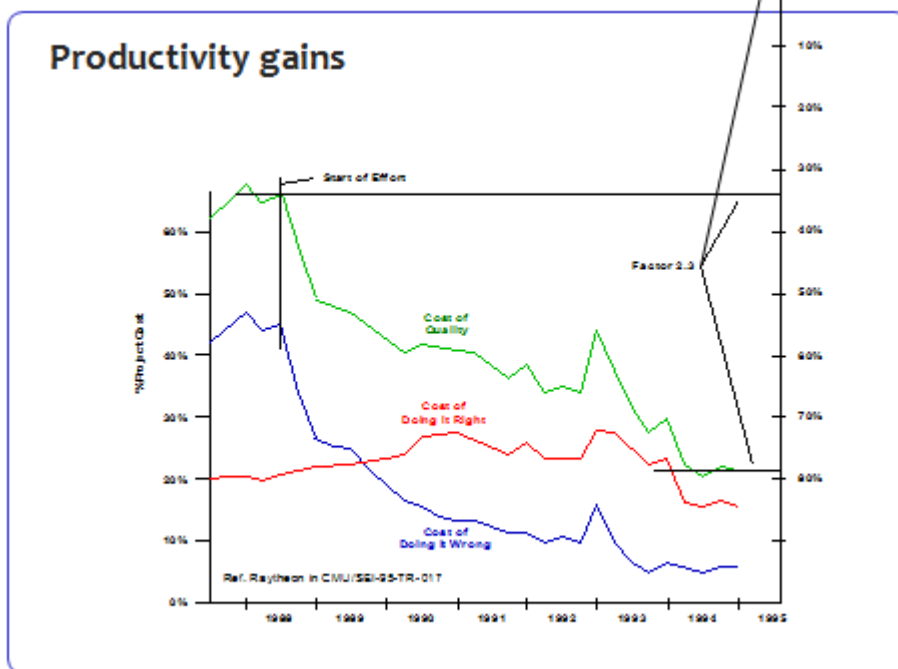
## Peter Drucker

Quality in a service or product is not what you put into it
It is what the client or customer gets out of it

57

## On Time

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

Compromise between what is needed
and what is possible

58

## Are you Serious about Time?

- Is Time Important ?

- Do you mind time ?

- Does your boss mind time ?

- Does your customer mind time ?

- Are you always on time ?

59

## Is it difficult to be on time ?

- Did anyone miss a plane ?

- What did you feel ?

- Why did it happen ?

- Did it happen again ?

60

## 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?

61

## Fallacy of 'all' requirements

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

61

## 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 ?

63

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

64

**But we're not Project Managers !**

- What caused the project being late ?
- Could *we* have prevented the project being late ?
- Was delivery time important ?
- Was delivery time a requirement ?
- Were all other requirements really more important ?

65

## What could *we* have done to save time?

66

## Who's Responsible for the Result of the Project?

- The Project Manager is *responsible* for *delivering* the right result at the right time

- The Project Workers work and decisions *determine* the result and the time it is delivered

- This makes everybody in the project implicitly as responsible as Project Management

67

More information:
www.malotaux.nl

## 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 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 !)
Projects with more than one project leader also fail

68

## Architect ↔ Project Manager

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

69

## PM and SE/Architect are like Conductors



- **Project Manager is the conductor of the *Project***
  - Vision and techniques to organize the project
- **Systems Engineer/Architect is the conductor of the *Product***
  - Vision and techniques to realize the product
- **However, there should be only *one captain on the ship***

70

## Systems Engineers

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

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

71

## Multidisciplinary ↔ Interdisciplinary

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

72

## 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**

73

## The challenge

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

74

## FatalDay

- FatalDay is the last moment *it* shall be there
- After the FatalDay, we'll have real trouble if the Result isn't there
- Real Option Theory says that we should do things as late as possible, but not later
  - As late as possible, having the most up-to-date information to decide what to do
  - Not later: the option has expired; it has no value any more
- Count backwards from the FatalDay to know when we should have started
- If that's before now, what are we going to do about it, because *failure is not an option*

75

More information:
www.malotaux.nl

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## 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

76

## Time to market

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

Every day we start later, we'll be done a day later
and miss € 10.000

77

## Cost of one day of delay

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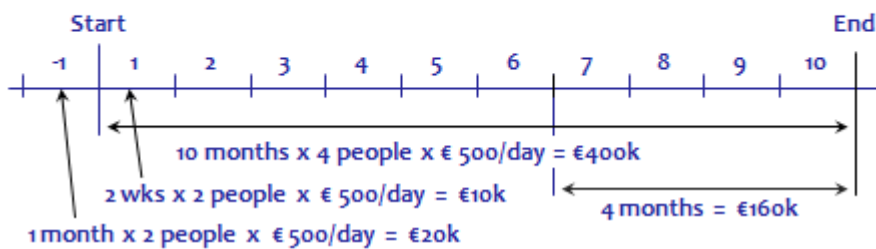- **Plus US$ 30,000 per day for lost benefit**
- **Total: US$ 70,000 per day for every day of** (unnecessary) delay

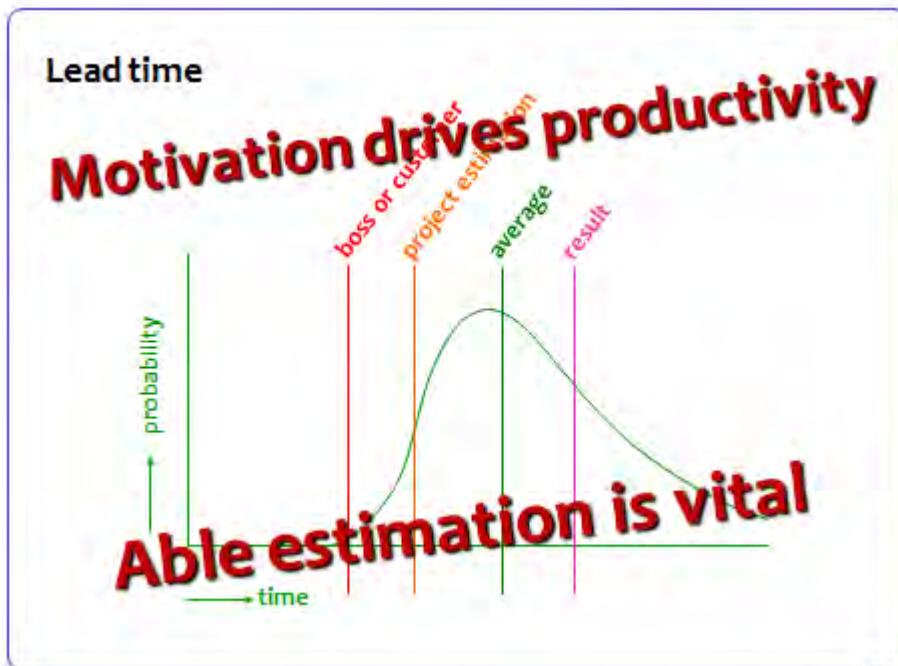- **$0^{th}$ order estimations are good enough**

## 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

4 months = €160k

1 month x 2 people x € 500/day = €20k

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

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

## Lead time

# Motivation drives productivity

boss or customer

project estimation

average

result

probability

# Able estimation is vital

time

80

## 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?

81

## Is this what you did?

81

## Defect rate

- Before test ?

- After test ?

83

More information:
www.malotaux.nl

**Alternative Design** (*how to solve the requirement*)

84

**Another alternative design**

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

85

**Niels Malotaux**
# Predictable Projects
**Delivering the Right Result at the Right Time**

## What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

## Elements in the exercise

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

# Human Behavior

## Human Behavior

- Systems are conceived, designed, implemented, maintained, used, and tolerated (or not) by people

- People react quite predictably

- However, often differently from what we intuitively think

- Most project process approaches (PMI, INCOSE, as well as developers)
  - ignore human behavior,
  - incorrectly assume behavior,
  - or decide how people should behave (ha ha)
- To succeed in projects, we must study and adapt to real behavior rather than assumed behavior

- Even if we don't agree with that behavior

## Is Human Behavior a risk?

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

90

## Human Behavioral Inhibitors

- **No Sense of Urgency**
- **Indifference**
- **(lack of) Discipline**
- **Intuition**
- **Fear of Uncertainty**
- **Fear of Perceived Weakness**

- **Fear of Failure**
- **Perceived lack of time**
- **(lack of) Zero Defects attitude**
- **Ignorance**
- **Incompetence**
- **Politics**

91

## People responsible for success

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

91

## 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)

93

## Intuition

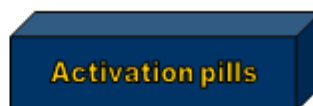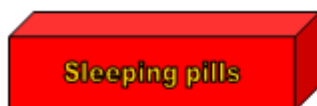- Makes you react on every situation
- Intuition is fed by experience
- It is free, we always carry it with us
- We cannot even turn it off
- Sometimes intuition shows us the wrong direction
- In many cases the head knows, the heart not
- Coaching is about redirecting intuition

94

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

Sleeping pills

Activation pills

95

## Communication

- Talking as near as possible along each other



To each other          Along each other

- Don't *assume* we understand: *check!*

96

## 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

97

## 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!

98

## Responsibility

- Taking responsibility - commitment
- Getting responsibility - empowerment
- Understanding responsibility
- Giving back responsibility

99

## 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

100

## 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
  at the FatalDate any excuse is irrelevant
- You knew much earlier

101

## People *oppose* change !

- 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

102

## Excuses, excuses, excuses ...

- We have been thoroughly trained to make excuses
- We always downplay our failures

- At the Fatal Day, any excuse is in vain: we failed
- Even if we "couldn't do anything about it"
- Failure is a very hard word. That's why we are using it !
- No pain, no gain
- We never say: "You failed", better: "We failed"
  - After all, *we* didn't help the person not to fail
  - "Lose face" is not only typical Asian

103

## 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"

- We have been trained from childhood to make excuses
- Ignore the bla bla
- Wait for the next reaction

104

## Logical thinking is not always better

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

**Real Options**
- Option to make or abandon a decision
- The later you make the decision,
  the more information you can have about it
- Options have value until expiration
- On expiration the value has disappeared
- Just in Time delivery
- Start feeding your sub-consciousness in due time,
  to decide just in time

105

## Accept Human Psychology

- **Why don't they practice what we preach?**
  (Humphrey 1999)

- **They don't practice what we preach !**

  *What now?*

106

## Ready in January

**Sense of Urgency**

- **Stick to your agreement**

    - Can you do that?
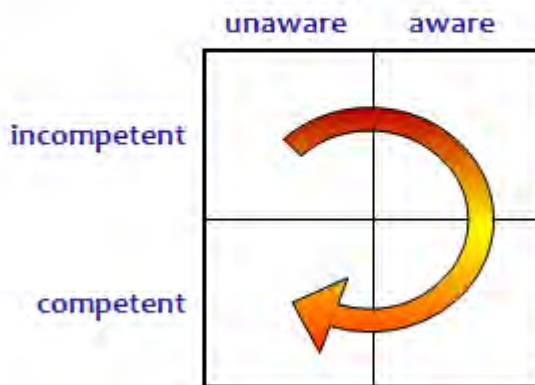    - Yes
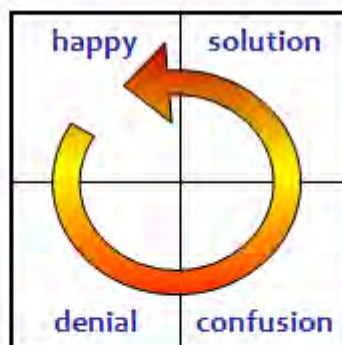
    - When is it done?

    **Be as explicit as needed**

107

## Competence square



## The problem of problem denial

**What do we do first?**

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

Idea — Begin — Product

- Known ...
- Clear ...
- Unknown ...
- Unclear ...

110



**First Time Right**

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

Idea — Lost Time to Market — Begin — Experiments — Begin First Time Right — Immediately right — Product

- Known ...
- Clear ...
- Unknown ...
- Unclear ...

- Known ...
- Clear ...
- Unknowns known
- Unclears clear
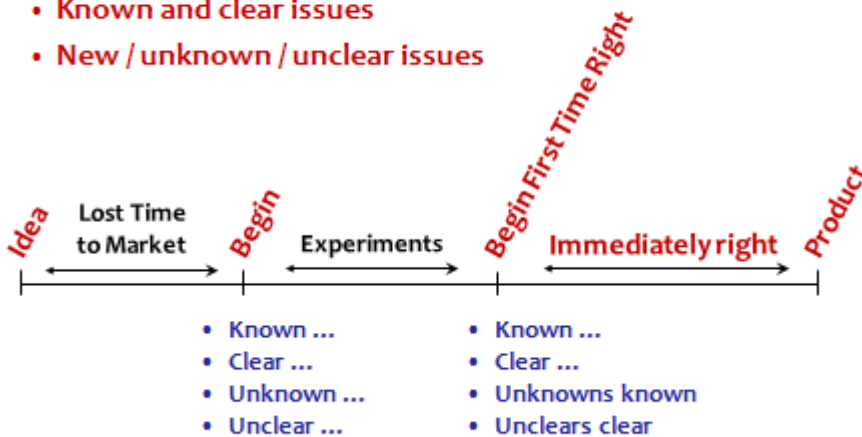
111

# Project
# Life Cycles

112

## Waterfall ?

Winston Royce 1970

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

ANALYSIS

PROGRAM DESIGN

CODING

TESTING

OPERATIONS
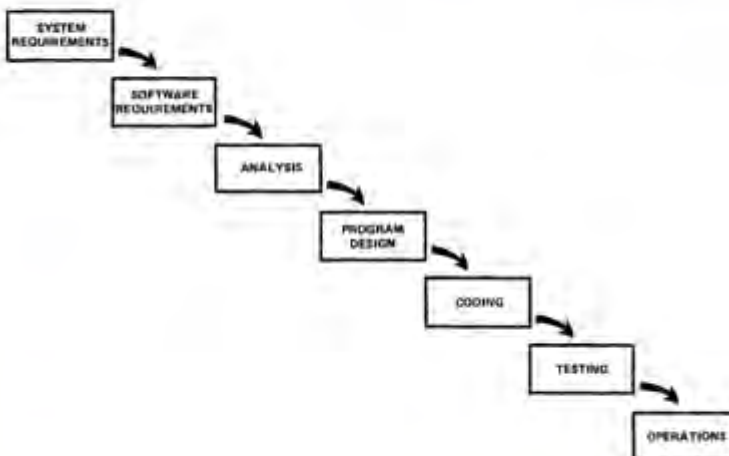
113

## This is what Royce wanted to say
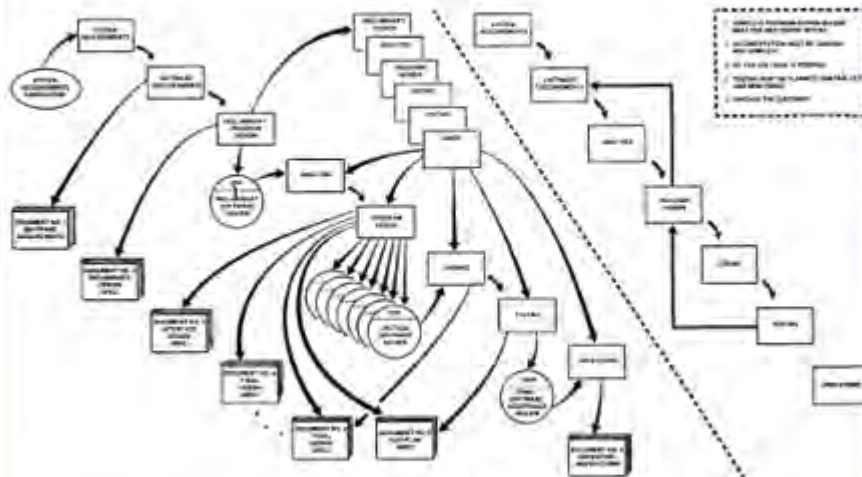


## When can we use waterfall ?

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

- **In your projects:**
  - Is everything completely clear ?
  - Will nothing change ?
  - Does everybody know exactly what to do ?
  - Are you sure ?
- **Even most production doesn't run smoothly the first time**

How management likes it

Start Project — We can do it — We did it

116



V-Model

117
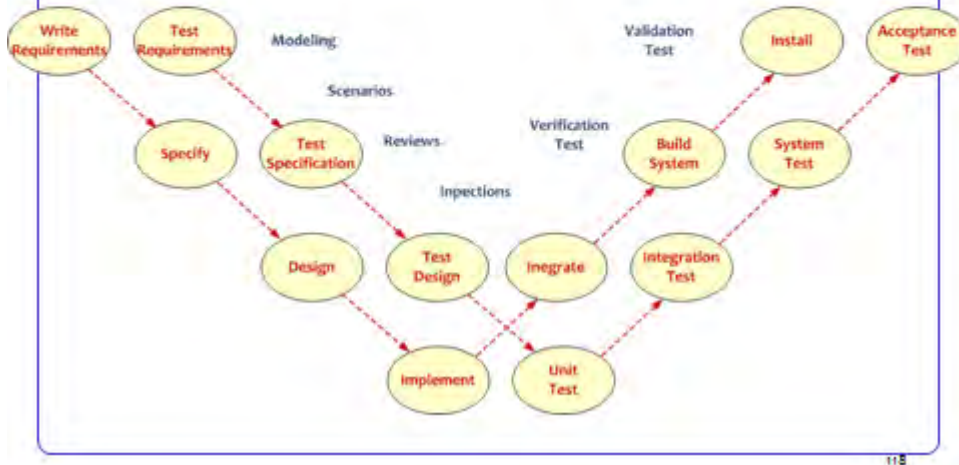
## W-model



## All Models are wrong

Some are useful

# Evolutionary Principles

120

## No cure - no pay

- If what we do doesn't deliver a positive ROI, there is no money to pay our salary
- So, better do not do things that do not deliver ROI

- Do you dare to work on a no-cure-no-pay basis ?

121

## Value

- Value is what makes the customer more successful and happy than before

- We're in the game of
  - Optimizing Value
  - Eliminating Non-Value
  - Not causing problems
  - At the lowest cost

122

## Perceived value

- What we perceive as value
- What the users perceive as value
- What the customer perceives as value
- What the stakeholders perceive as value

- May be different from real value
- Better assume that a lot of our assumptions are wrong

- Still, value means different things to different stakeholders
- If we want to be successful, we may have to find the best compromise

123

## The head and the hart

- There is often a paradox between what the mind tells and what the body does (logic ↔ emotion)

- So, we shouldn't just do what the customer says, but rather find out what he really needs   and still wants

- If we base our perception of the requirements on what the customer says (Waterfall, Agile), we're probably developing a great solution to the wrong problem

124

## 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*

125

## 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
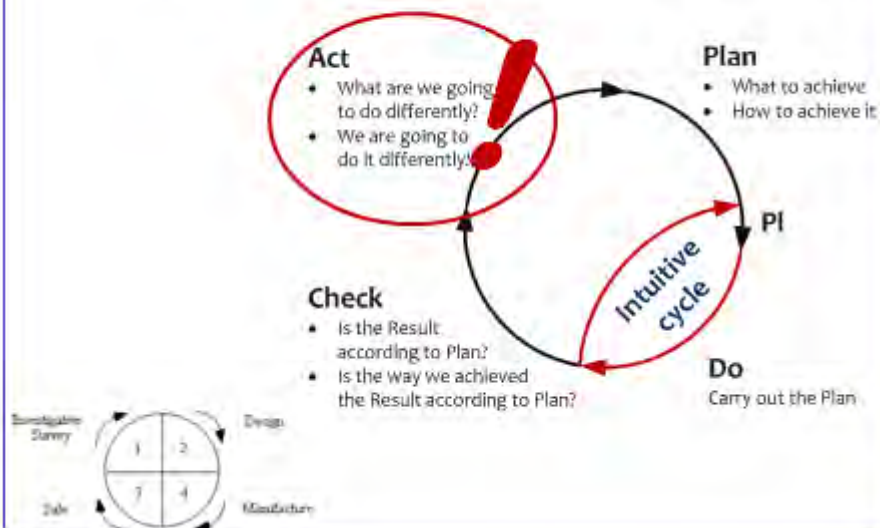- **Preflection** is for foresight and prevention

**Only with *prevention* we can save precious time**
**This is used in the Deming or Plan-Do-Check-Act cycle**

126

## 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

127

More information:
www.malotaux.nl

## It can't be done - it must be done

- **It can't be done**
  - Management doesn't allow it
  - "They" won't do it
  - It's impossible

  → Stuck in the *Check Phase*

- **It must be done**
  - How are we going to do it

  → Moving to the *Act phase*

- **We don't let others make us fail**
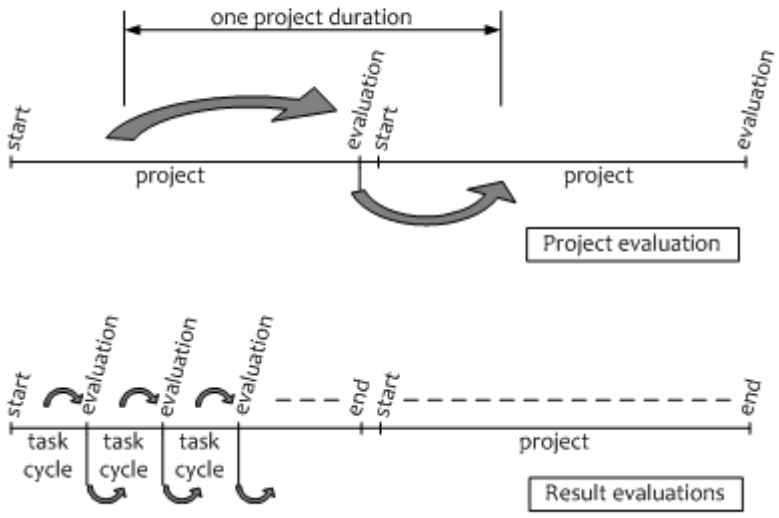
**Act**
What are we going to do differently

**Plan**
What do we want to know or to do

**Check**
Is Result according to plan?

**Do**
Carry out plan

128

## Project evaluations

one project duration

start — project — evaluation/start — project — evaluation

Project evaluation

start — task cycle/evaluation — task cycle/evaluation — task cycle/evaluation — end/start — project — end

Result evaluations

129

## Is waterfall wrong ?

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

ANALYSIS

PROGRAM DESIGN

CODING

TESTING

OPERATIONS

| cycle | 1 | 2 | 3 | 4 | 5 | - - - - - - - | n-1 | n |

prepare  waterfall  waterfall  waterfall  waterfall  waterfall  waterfall  waterfall  waterfall  finalize  finalize

130
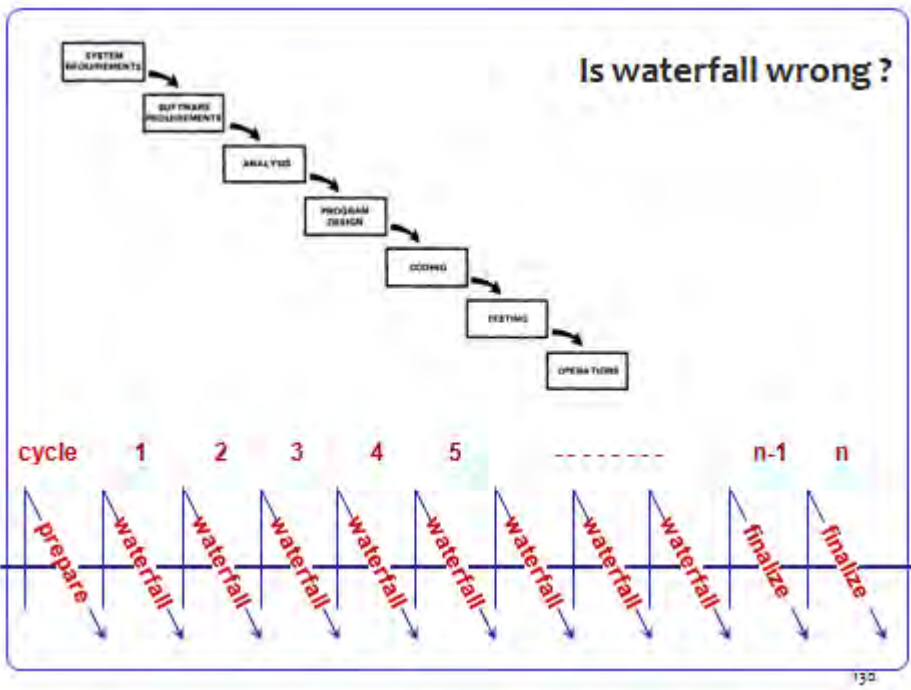
## Development cycles

start — planning

start — smart planning — planning

start — planning

131
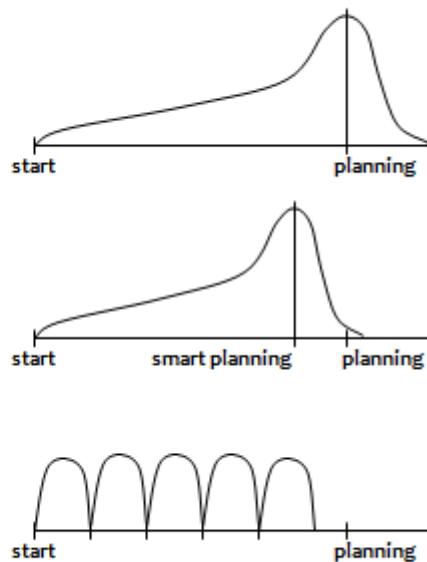
## Knowledge
## how to achieve the goal

**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

*doing the right things*

**then we can**

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

**and we can**

*doing the right things right*

- Spot problems quicker, allowing more time to do something about them
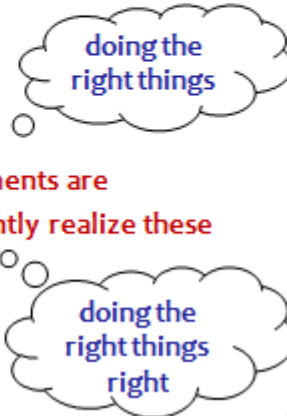
132

## Known for decades

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

*Eliminating Waste*
*Not doing what doesn't yield value*

133

## There is nothing new in software too

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

*Toward Zero-Defect Programming*

134

---

## If we know, why do projects still fail ?

**Cobb's Paradox:**

"We know why projects fail,
we know how to prevent their failure
-- so why do they still fail?"

Martin Cobb
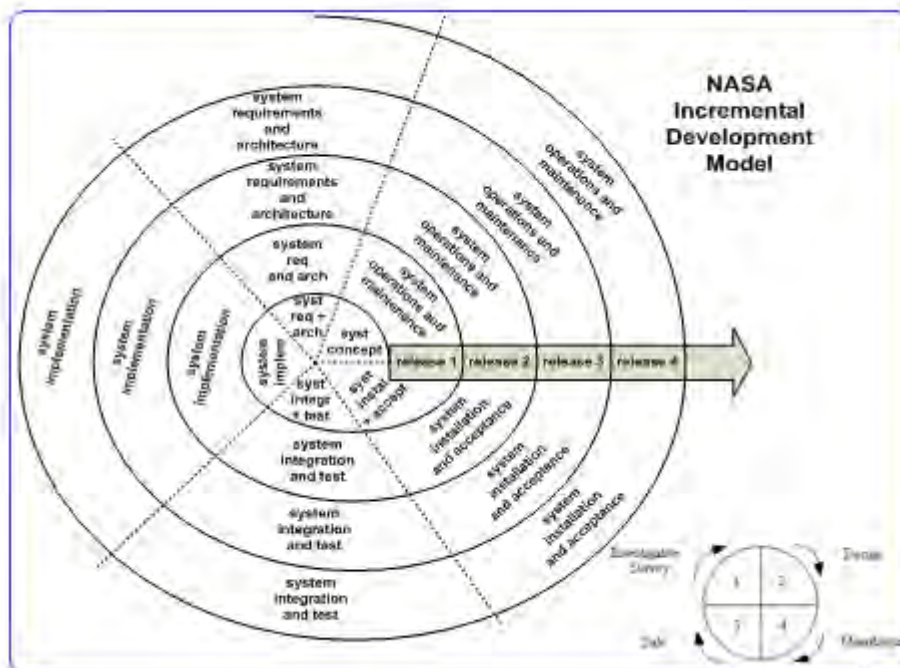Treasury Board of Canada Secretariat
Ottawa, Canada

1989

135

## Evo



- Evo (short for Evolutionary...) uses PDCA consistently
- Applying the PDCA-cycle
  actively, deliberately, rapidly and frequently,
  for Product, Project and Process, based on ROI and highest value
- Combining Planning, Requirements- and Risk-Management into
  Result Management
- We know we are not perfect, but the customer shouldn't be
  affected
- Evo is about delivering Real Stuff to Real Stakeholders
  doing Real Things                    "Nothing beats the Real Thing"
- Projects seriously applying Evo, routinely conclude
  successfully on time, or earlier
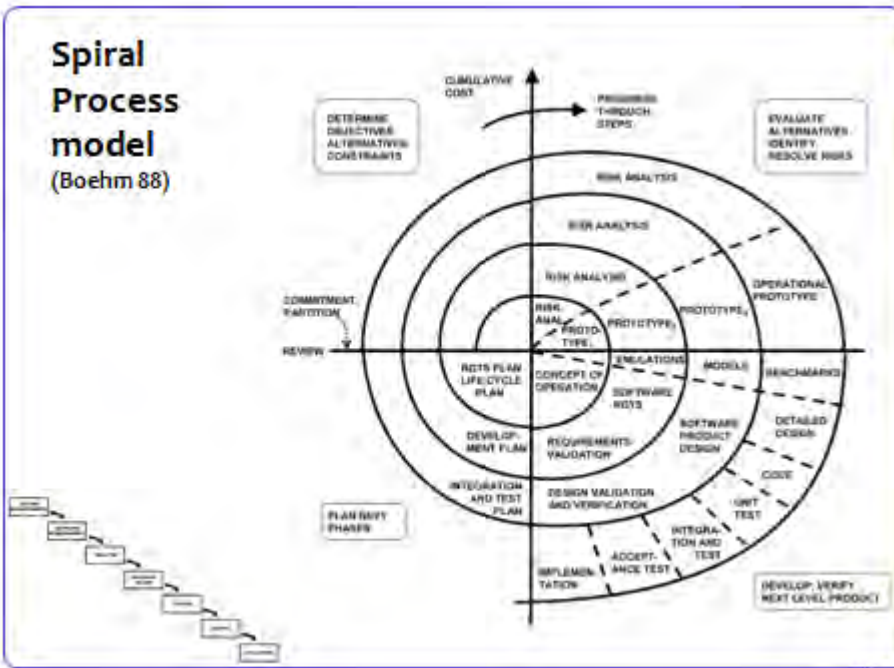
136



NASA Incremental Development Model

137

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

# Evolutionary Planning

# TaskCycle
# DeliveryCycle

141

## 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

142

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## 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

task

delivery

project
organization

strategy

roadmap

143

## 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)

144

## 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

| Task | | |
|---|---|---|
| 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 | not |
| Task$_k$ | 1 | |

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

145

## Estimation

- Changing from Optimistic to Realistic

- Only works if we are Serious about Time

Sense of Urgency

146

## At the end of the week

*Immediate consumption of metrics*

- **Was all planned work really done?**
  **If a Task was not completed, we have to learn:**
  - Time spent but the work not done? → effort estimation problem
    Discuss what the causes may be and decide how to change your estimation habits
  - Time not spent? → time management problem
    - Too much distraction
    - Too much time spent on other (poorly-estimated) Tasks
    - Too much time spent on unplanned Tasks

    Discuss what the causes may be and decide how to improve (Check and Act)

- **Conclude unfinished Tasks after *having dealt with the consequences*** ← immediate metrics consumption !
  - Feed the disappointment of the "failure" into your intuition mechanism
  - Define new Tasks, with estimates, and put on the Candidate Task List
  - Declare the Task finished after having taken the consequences
- **Continue with planning the Tasks for the next week**

147

## Meetings

- **Do you have weekly project meetings ?**

- **Pitfalls**
  - Not reaching set goals
  - One to ones, others waiting
    - Example: status round ("round of excuses")
    - Example: detailed discussion
  - Discussing less important subjects for too long

- **Meetings are very costly** (ROI?)
  - Try the meeting-meter
    number of people * average hourly rate: show $$ ticking

148

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## TaskSheet
(this is the think part of First Think - Then Do)

- Task description
- Requirements for this task to be used as reference for verification:
  - Functions (what should be the result of this task?)
  - Qualities (how well should the results be)
  - Constraints (e.g. what not to do)
- What activities must be done to realize the requirements stated?
- Implementation details (how am I going to implement it)
- Verification approach - test design
- Planning
- Is everything really clear?
- Have this document (and related docs, if any) reviewed
- Clarify any unclears until everything is clear and agreed with the reviewer
- Do the work

151

## TaskSheet: More time needed?

- Some people think they need more time for the Task if they "must fill in" the TaskSheet

- If you feel you "must fill in": *Don't do it !*

- If you think you need more time: add more time

- You will need this information during the Task anyway, so you should want it and

- It should *save* time

152

## Analysis Tasks

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

153

## Management Questions on Tasks



- **Is the Project under Control?**

- **Show me!**
  - No "holes" in OK's
  - All available plannable time planned
  - TaskSheets used
  - Results used
  - Prompt explanation in case of discrepancies

## DeliveryCycle



- **Are we delivering the right things, in the right order to the right level of detail for now**

- **Optimizing requirements and checking assumptions**
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important stakeholder values that can be made in the least time
  - What will make Stakeholders more productive now

- **Not more than 2 weeks**

task

delivery

project
organization

strategy

roadmap

## Tasks feed Deliveries

this week

tasks → tasks → delivery

tasks → tasks → tasks → delivery

tasks → tasks → tasks → tasks → tasks → delivery

TimeLine

## 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**

**Niels Malotaux**
# Predictable Projects
## Delivering the Right Result at the Right Time

---

### Designing a Delivery

TaskCycle

| Fri | Mon | Tue | Wed | Thu | Fri | Mon | Tue | Wed | Thu | Fri |

available time:
36 hr gross
24 hr plannable

Delivery to Stakeholders

deliv to main team

Delivery to Stakeholders

| Serge (ProjLead) | | Gregory | | Gregory (later) | |
|---|---|---|---|---|---|
| MbWA | 3 | Draft design | 6 → | Draft design | 0 |
| Planning nxt wk | 3 | Finish design | 6 → | Finish design | 0 |
| Work for deliv | 4 | Work for deliv | 3 | ... | |
| - | 6 | - | 1 | | |
| - | 2 | - | 2 | Repair deliv | 0 |
| - | 1 | - | 2 | ... | |
| - | 5 | - | 3 | | |
| **Total** | **24** | - | 5 | | |
| | | - | 6 | **Jerome** | |
| | | XMLa | 4 → | XMLa | 3 |
| | | XMLb | 4 → | XMLb | 3 |
| | | **Total** | **42** | ... | |

Zero Defects Attitude

159

---

### TaskCycle Exercise

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

| Task | | |
|---|---|---|
| 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 | not |
| Task_k | 1 | |

160

---

## TaskSheet exercise

- Select one of the Tasks for next week
- Organize the Task with a TaskSheet

- Review the TaskSheet of a colleague
- Let the colleague review your TaskSheet

- Did anything change due to the reviews?

161

## Agile, but will we be on time ?

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

but ...

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

162

# Evolutionary Planning

# TimeLine

163

## TimeLine — What the customer wants, he cannot afford

now — date needed (FatalDate) — "all" done

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

**Standard Projects**

now — date needed (FatalDate)

will be done | might be done | not done

most important things — bells & whistles

**Evo**

164

## Two options

1. **Conventional option**
   At the fatal day we'll tell we didn't succeed

2. **Evo option**
   We already know we won't succeed, so we can tell it now,
   then together we can decide what to do

**Which option do you want?**

**Quality On Time is also being honest as soon as you can**
The challenge is to find out as soon as you can

## Dependencies

## Priorities

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

**Let it be the most important 80%**

167

## If it easily fits ...

now                                                          FatalDate

needed time << available time : OK for now

168

Setting a Horizon



Result to Tasks and back

171



172

Accepting Fate?

Calibrating estimations and options and consequences

-a- originally estimated
-b- actually done after 4 wk

after 4 wk:
-c- extrapolated consequence
-d- taking longer
-e- going faster
-f- more time needed
-g- doing only what is needed
-h- working more cleverly
-g₂- target



Calibration

| Activity | Estimate | Real |
|----------|----------|------|
| Act1 | Ae1 | Ar1 |
| Act2 | Ae2 | Ar2 |
| Act3 | Ae3 | Ar3 |
| Act4 | Ae4 | Ar4 |
| Act5 | Ae5 | Ar5 |
| Act6 | Ae6 | Ar6 |
| Act7 | Ae7 | Ar7 |
| Act8 | Ae8 | Ar8 |
| Act9 | Ae9 | Ar9 |
| Act10 | Ae10 | Ar10 |
| Act11 | Ae11 | |
| Act12 | Ae12 | |
| Act13 | Ae13 | |
| Act14 | Ae14 | |
| Act15 | Ae15 | |
| Act16 | Ae16 | |
| Act17 | Ae17 | |
| Act18 | Ae18 | |
| Act19 | Ae19 | |
| Act20 | Ae20 | |
| Act21 | Ae21 | |
| | | |
| Act... | Ae... | |

ratio ΣAr/ΣAe in the past

now

predicted Value Still To Earn in the future

then

then2

**Calibration Factor**

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

**Value Still To Earn**

$$Calibration\ Factor * \sum_{now}^{then} Ae$$
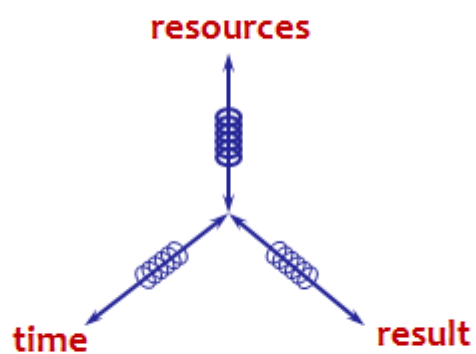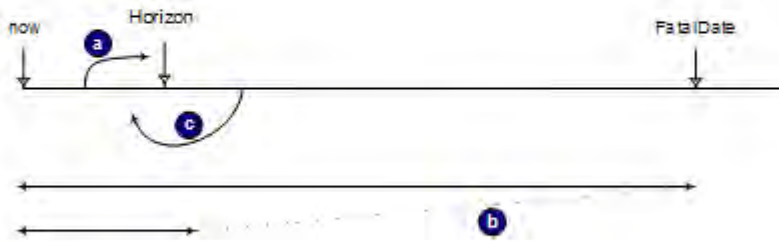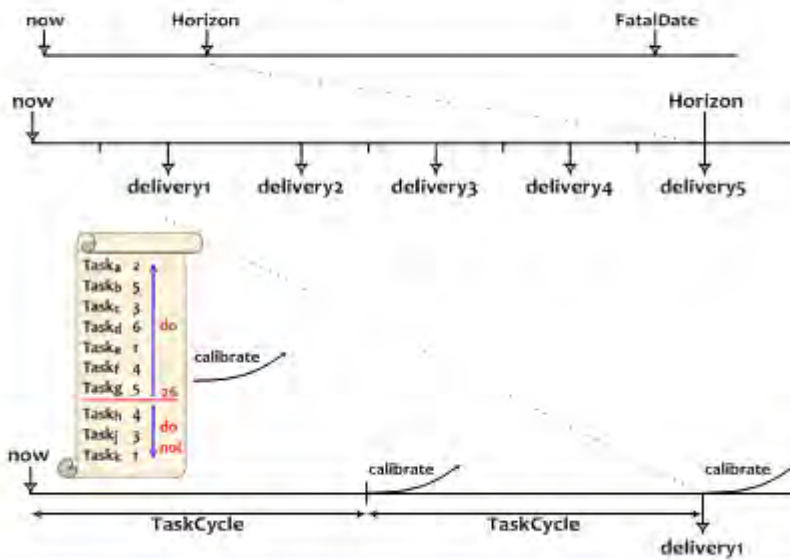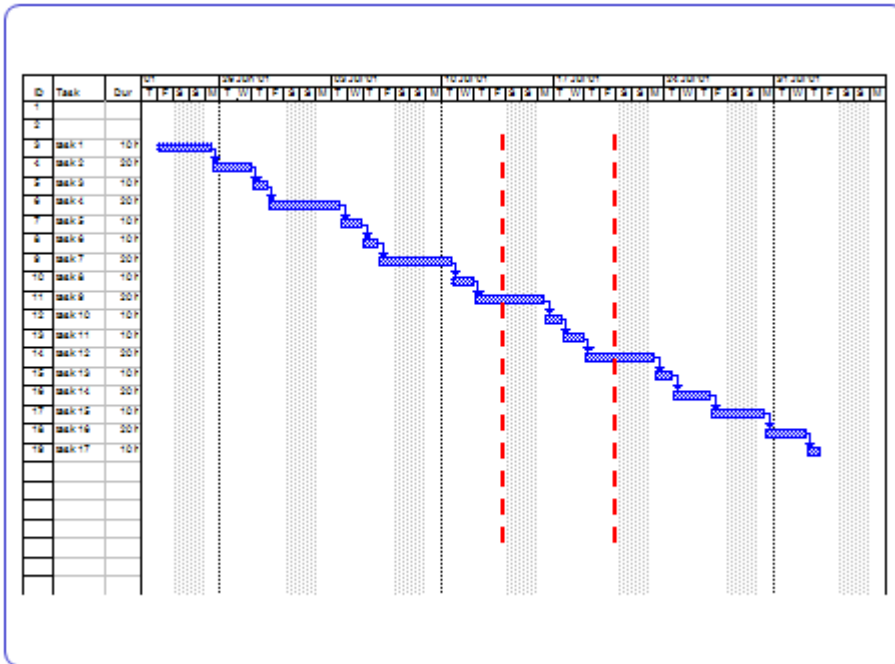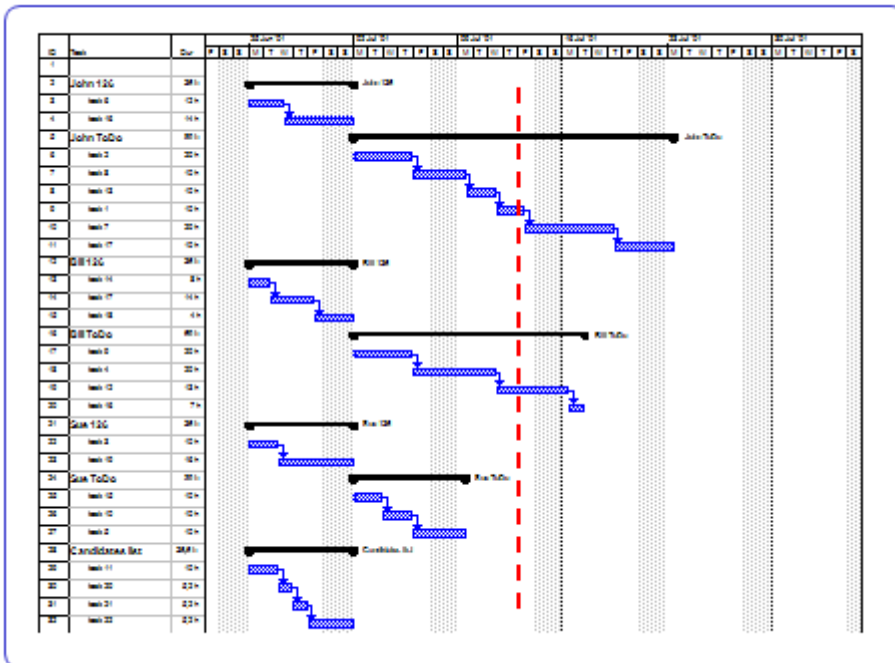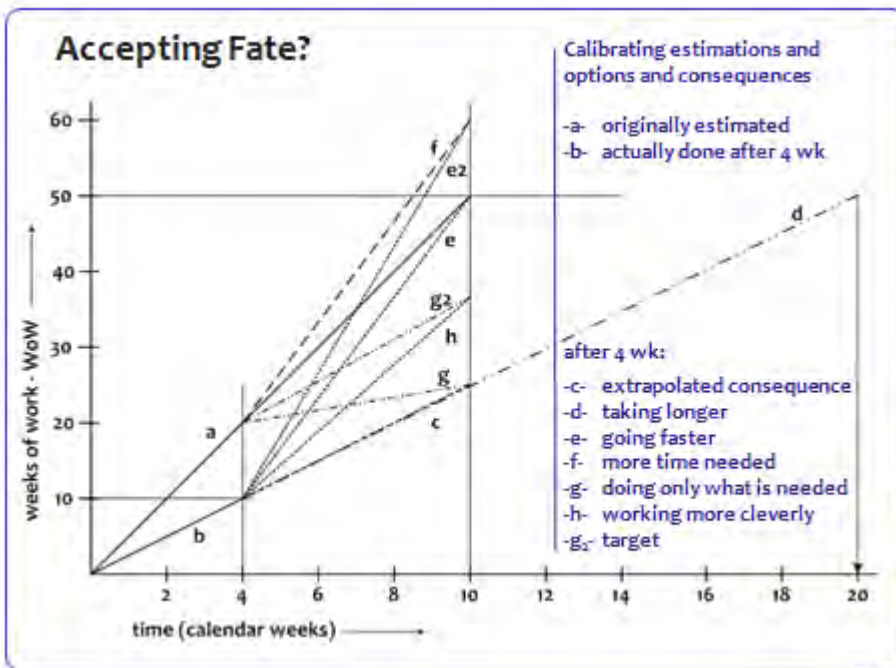
## Predicting *what will be done when*

| 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 | 3.5 | 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 |
| | | | | | | | | |

175

## 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

176
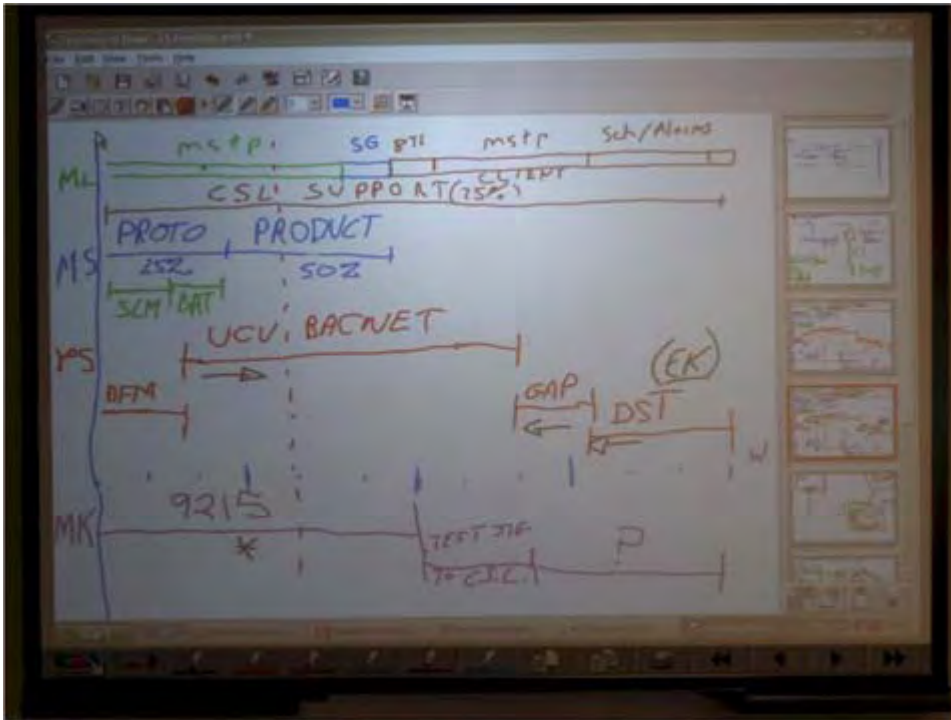
## What do we do if we see we won't make it on time?
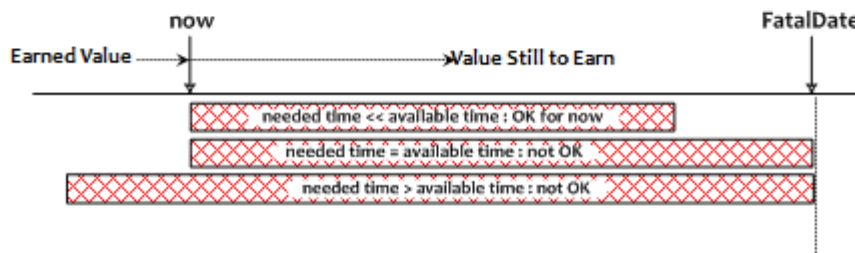
- If it doesn't fit ... count backwards

## Deceptive options

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

179

## Adding people to a late project ...

### makes it later

(Brooks' Law, 1975)

180

## The Myth of the Man-Month



lower cost

Economic optimum?

reality (Putnam)

shorter time

project duration

nine mothers area

intuition
people x time = constant
Man-Month Myth
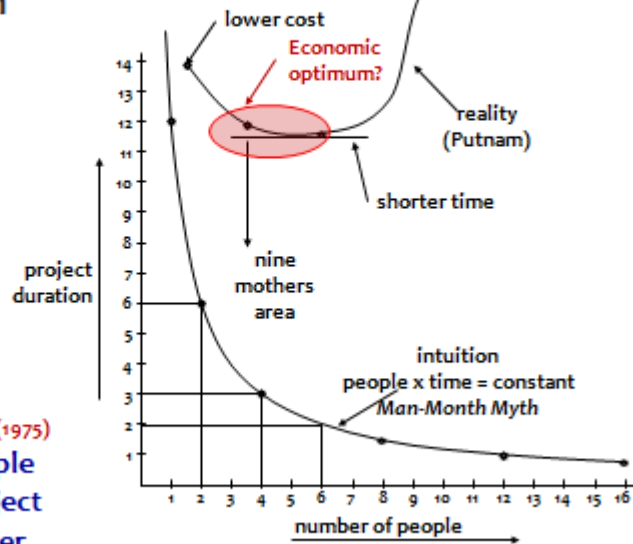
**Brooks' Law** (1975)
**Adding people to a late project makes it later**

number of people

---

## 🙂👍 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

## 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 !

183

# Estimation

## Estimation techniques used

- **Just-enough estimation** (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- **Changing from optimistic to realistic predictions**
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- **0th order estimations** (ball-park figures)
  - For decision-making in Business Case and Design
- **Simple Delphi**
  - For estimating longer periods of time in TimeLine
  - For duration of several (15 or more) elements of work
- **Simpler Delphi**
  - 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

## The Pareto principle (20 - 80 rule)
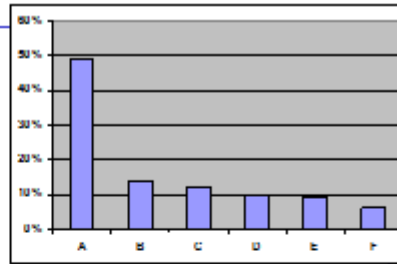
A collection of problems always can be divided into a small number of large problems and a large number of smaller problems

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

Juran, 1960

Example: better have 80% of the Requirements 100% done than 100% 80% done
It may even take only 20% of the resources

---

## Realistic estimation in 3 weeks

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

## Estimation is non-symmetric
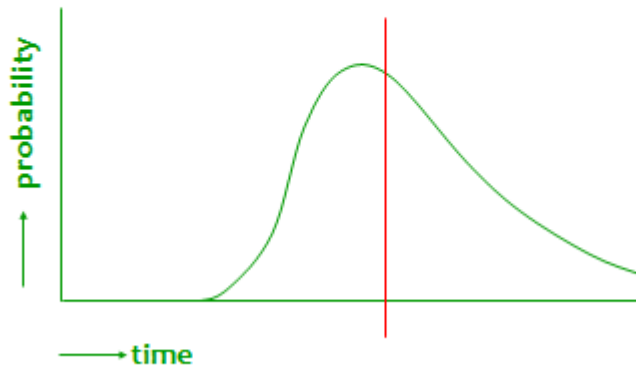


## Simple Delphi estimation

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

## Simple and Simpler Delphi

| | | |
|---|---|---|
| 1. List things to do | 1. List things to do |
| 2. Distribute the list | 2. Distribute the list |
| 3. Add and estimate | 3. Add and estimate |
| 4. List estimates | 4. List estimates: min and max |
| 5. Discuss if differences | 5. Discuss if differences |
| 6. Estimate again | 6. Agree on value between min and max |
| 7. Repeat until consensus | 7. Add up all the estimates |
| 8. Add up all the estimates | |

Even with coarse estimates per element of work,
the sum averages out the variations and can be quite predictive

190

## $0^{th}$- 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

191

## 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 estimates

- Others should never challenge the estimation

- Estimates are non-negotiable!

- We can and should negotiate about the contents

192

## 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 we'll do anyway

- We only work on planned things

193

## Delphi Exercise

- What's the average number of coins we have with us now?
- Write down your estimates

|  | nr of coins |
|---|---|
| 1. Individually (no discussion) estimate the number | |
| 2. Discuss with your neighbor (groups of two) and estimate again | |
| 3. Discuss with your neighbor group (groups of four) and estimate again | |
| 4. Count the actual number of coins | |

194

## TimeLine examples
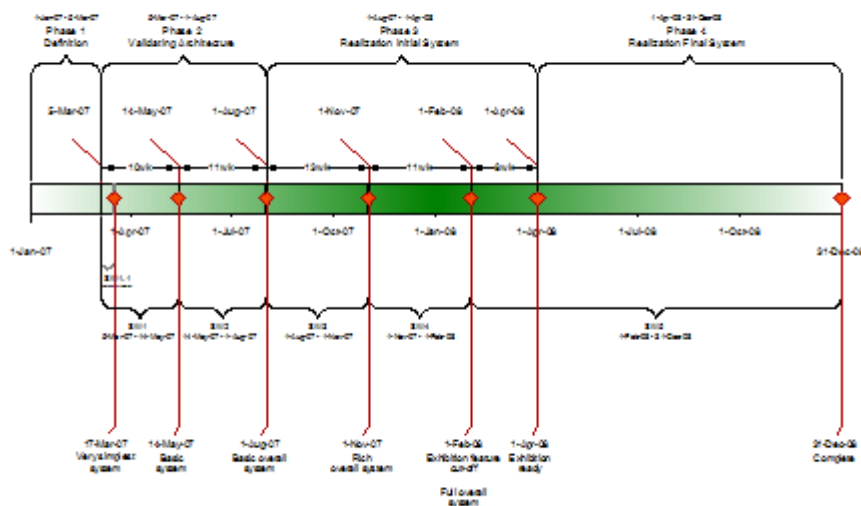


### TimeLine example

More information:
www.malotaux.nl

## TimeLine planning



## 5 day project model
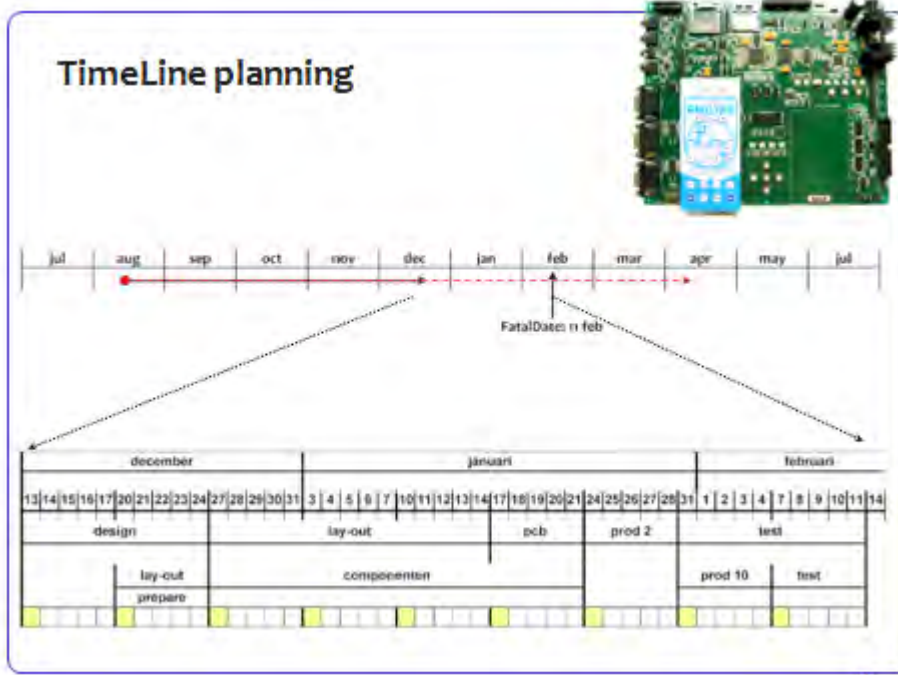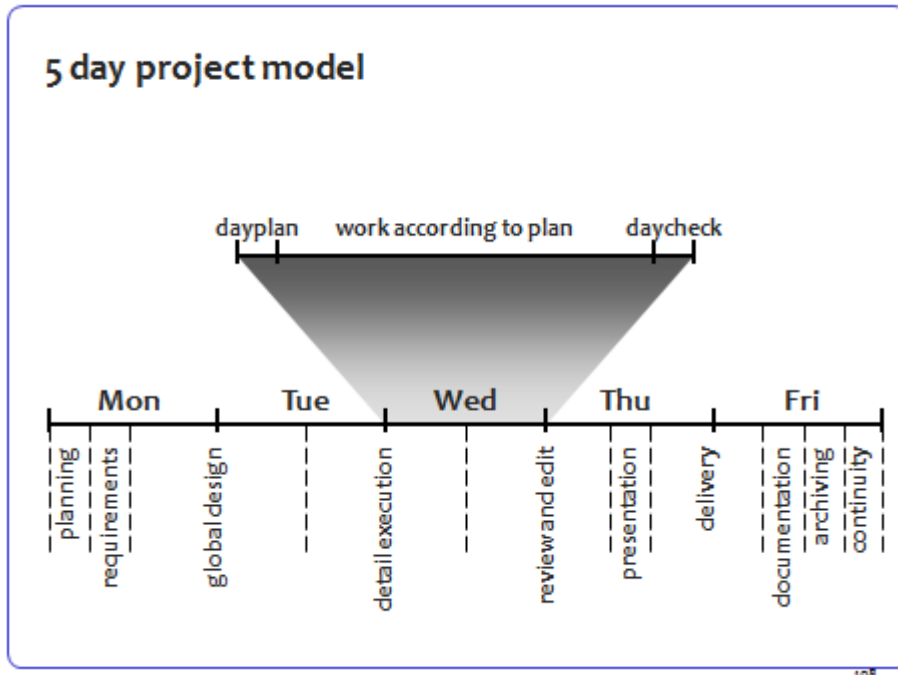
## Available TimeBoxes
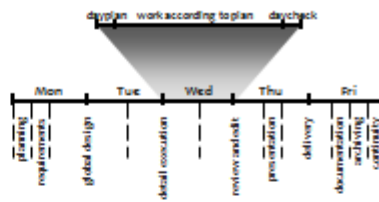


| activity | ~% | hrs |
|----------|-----|-----|
| Planning | 5 | 2 |
| Requirements | 5 | 2 |
| Global design | 20 | 8 |
| Detail execution | 20 | 8 |
| Review and edit | 20 | 8 |
| Presentation | 5 | 2 |
| Delivery | 10 | 4 |
| Documentation | 5 | 2 |
| Archiving | 5 | 2 |
| Continuity | 5 | 2 |
|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

199

## 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?**

200

**Niels Malotaux**
## Predictable Projects
**Delivering the Right Result at the Right Time**

---

### Do you think you can help us ?

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

**Plan**
- What to achieve
- How to achieve it

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

**Do**
Carry out the Plan

---
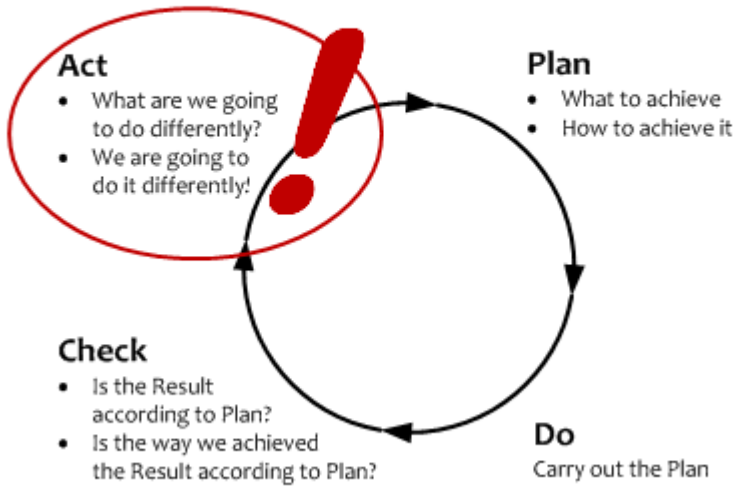
### In stead of complaining about a problem ...
(Stuck in the Check-phase)

### Let's do something about it !
(Moving to the Act-phase)

---

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

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

103

## TimeLine

wk

| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 13 |

| start | delivery cust a | delivery cust b,c | delivery cust a,d | (all done) |
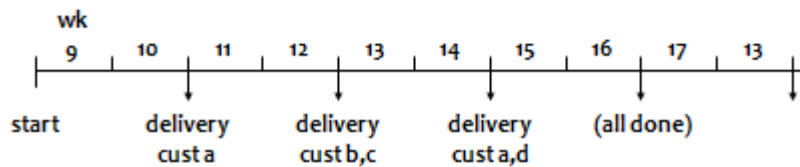
### Selecting the priority order of customers to be served
- "We'll have a solution at that date … Will you be ready for it ?"
  An other customer could be more eagerly waiting
- Most promising customers

104

## Result

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

Recently:

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

105
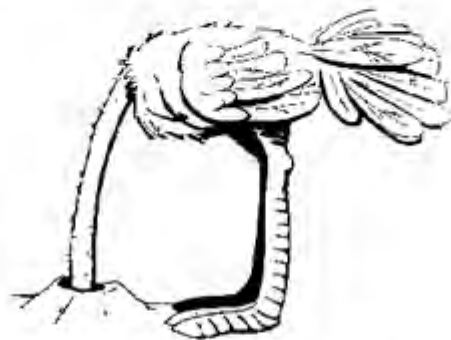


The problems in projects are not the real problem,
the real problem is that we don't do something about it

106
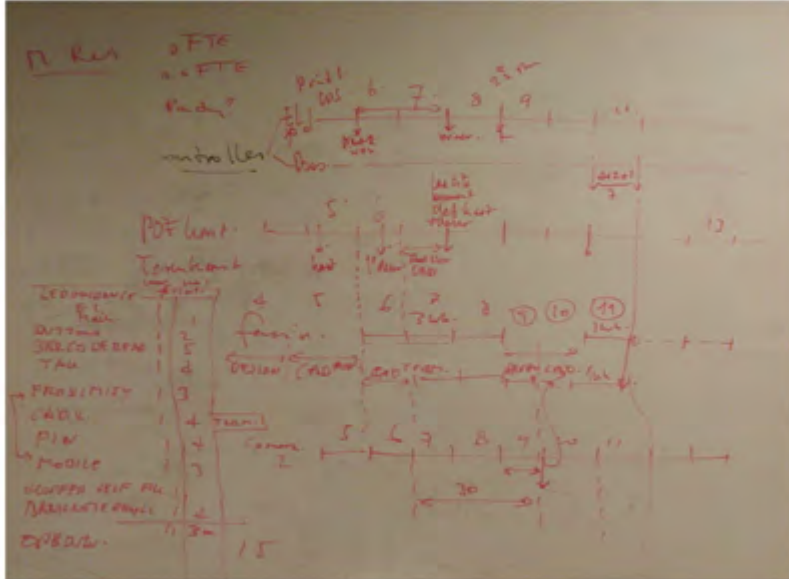
## Whiteboard TimeLine Planning



## Whiteboard TimeLine Planning
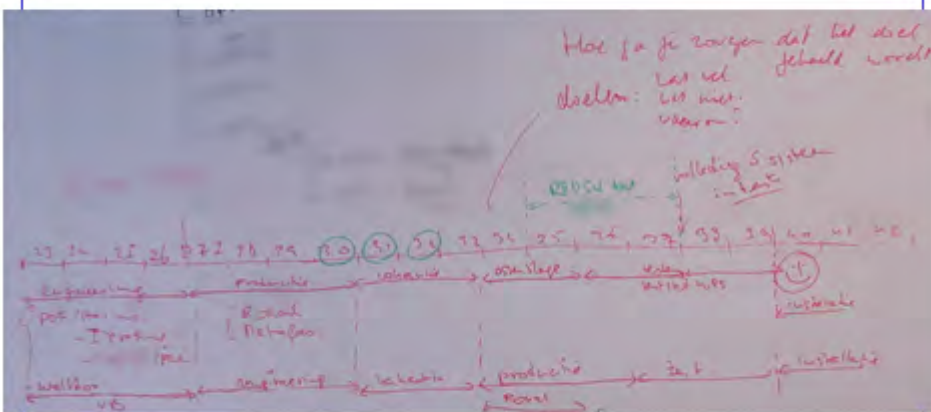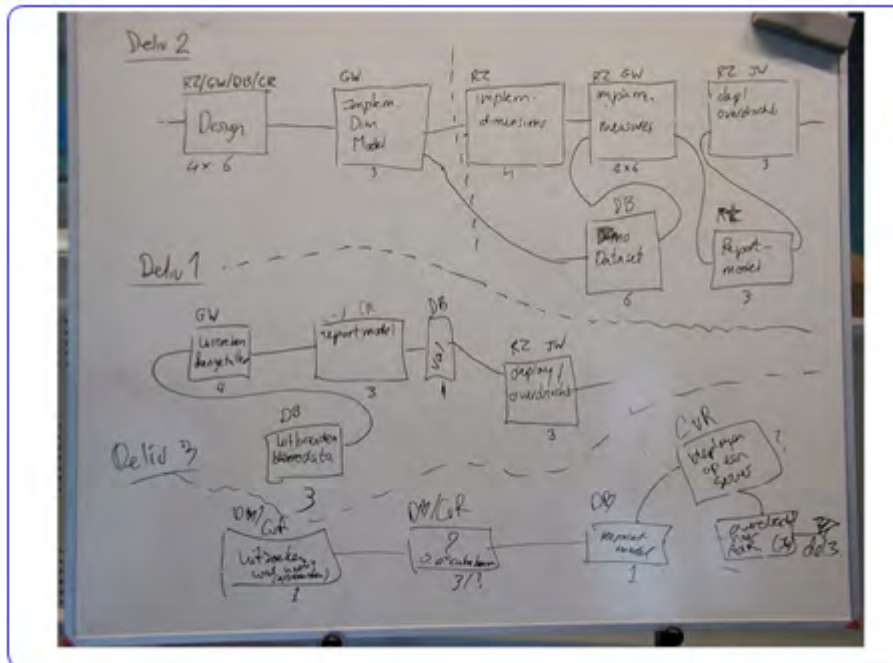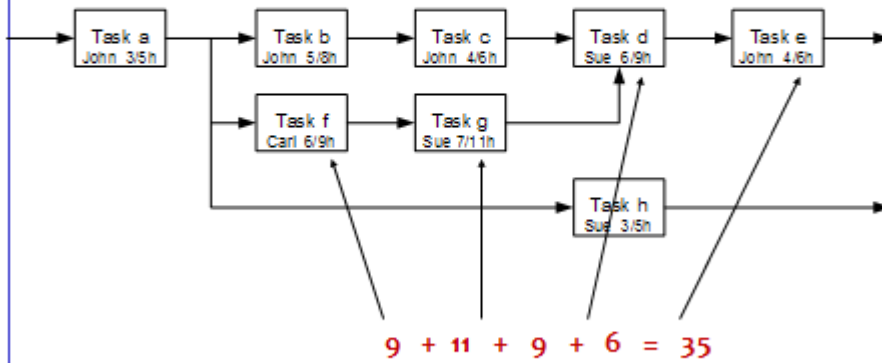
# Some details

212

## 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*
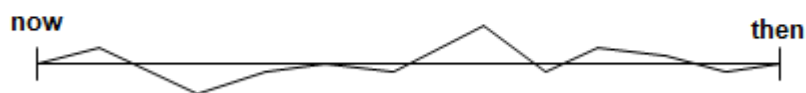
task

delivery

project
——————
organization

strategy

roadmap

now                                    then

213

What if ...

now    task cycles    horizon    end
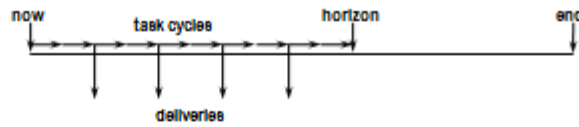
deliveries

- Somebody gets sick
  - Swap deliveries
- Requirement change (functional, performance, constraint)
  - Never say "Is impossible, we have no time", rather:
  - Of course it is possible, tell me what to leave out
  - Where shall we fit it in the TimeLine?

TimeLine provides control over the Project

214

If we add something ...

If we add something, something else will not be done

now    FatalDate

215

## Making best use of limited available time

- **If the work is done, the time is already spent**

- **If we still have to do the work, we can decide**
  - What is really important
  - What is less important
  - What we must do
  - What we can do
  - What we are going to do
  - What we are not going to do

- **Therefore we plan first, in stead of finding out later**

- **We cannot work in history**

216

## TimeBox — taking Time seriously

- **A TimeBox is the maximum time available for a Task**

- **When the time is up, the Task should be completely done: there is no more time !**

- **Because people tend to do more than necessary**
  (especially if the requirements of the Task are unclear)
  - Check halfway whether you're going to succeed on time
  - If not: what can you do less, without doing too little
  - Define the requirements of the Task well
  - If the TimeBox is unrealistic: take the consequences (pdcAct) immediately
    (if a Task suddenly proves to need much more time, is it still worth the investment?)

- **If you really cannot succeed within the TimeBox:**
  - Check what you did
  - Check what you didn't do
  - Check what still has to be done
  - Define new Tasks with estimations (TimeBoxes !)
  - Stop this Task to allow for finishing the other committed Tasks
    (don't let other Tasks randomly be left undone)

217

## Parkinson's Law
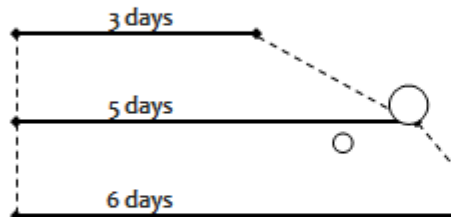
3 days

5 days

6 days

**Evo**
- Do 3 days in 5 days!

- Success
- Unstress
- Energy
- Motivation = Motor of productivity
- Higher productivity!!

**Standard Management**
- Do 6 days in 5 days!

- Never succeed
- Frustration
- De-motivation
- Stress
- Higher productivity??

"Work expands to fill the time available"

218

## 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

219

**Niels Malotaux**
# Predictable Projects
**Delivering the Right Result at the Right Time**

## Interrupts

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



- In case of interrupt, use interrupt procedure

## 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.

## Task selection criteria

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

112

## Delivery selection criteria

1. What will generate the optimum feedback
2. We deliver only to eagerly waiting stakeholders
3. Delivering the juiciest, most important
   stakeholder values that can be made in the least time
- What will make Stakeholders more productive now

- Every delivery must have a useful set of stakeholder values (features,
  qualities), otherwise the stakeholders get stuck
    – Delete ↔ Add
    – Copy ↔ Paste
- Every new delivery must have clear extras,
  otherwise the stakeholders won't keep producing feedback
- Every delivery delivers smallest clear increment,
  to get the most rapid and most frequent feedback
- If a delivery takes more than two weeks, it can usually be shortened:
  try harder

113

## Types of Tasks

1. **Tasks done within estimated time** (= timebox)

2. **Analysis Tasks** (*too short timebox*)
   - What do you know now
   - What do you still not know
   - What do you still have to know
   - Which tasks can you define

3. **Mis-estimated tasks** (we're only human)
   - Feed the disappointment about the failure to your experience/intuition mechanism
   - What did you do
   - What did you not do
   - What do you still have to do
   - Which tasks can you define

114

## Smallest step with highest value

- **Evo tries the smallest possible step**
  - If the result proves to be incorrect, we have to redo as little as possible
  - The earlier we are done, the more time we have in our future
  - Just enough to see we're on the right track

- **Because our (and their !) assumptions may be wrong**

115

## What is the shortest way through the maze?

**How quickly do we know an alley is the main road or a dead-end?**

Result

?

Requirements

116
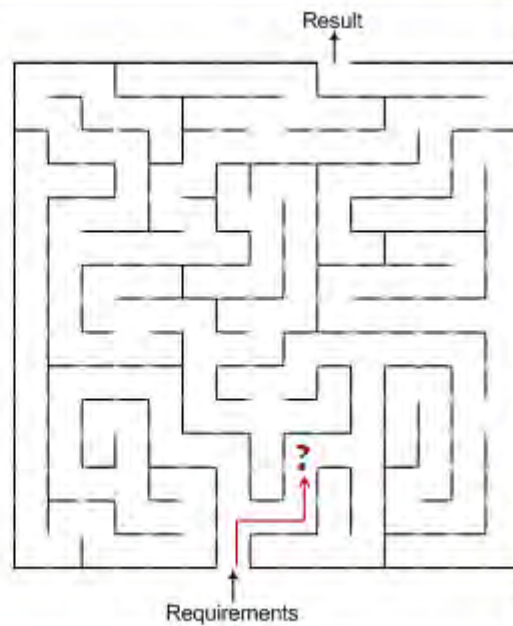
## Accepting a Task

- **Accepting a Task means:**
  - Taking full responsibility for the successful conclusion of the Task within the time agreed

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

117

## What to plan and what not to plan

- We plan tasks that don't get done unless planned
- We do not plan tasks that don't have to be planned to get done. Such planning costs more than it saves
- Account for these tasks as "unplannable tasks"
- Default we allocate 2/3 for plannable tasks and 1/3 for unplannable tasks
- We may include tasks in the planning to show that the hours for these tasks are not available for other work
- Plan all plannable hours

118

## Beware of longer Tasks

- Beware of Tasks longer than about 6 hrs
- Estimation is never exact
- If you have 4 or more Tasks in a week, the variation in the Tasks estimations should average

Only the *average* should be OK:
*Result* is all that counts
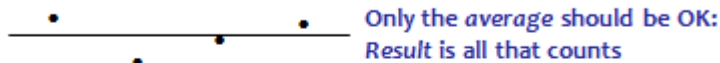
- You have only 2/3 plannable time, so you can cheat a bit to get all the committed tasks done
- May seem contradictory to the TimeBox principle ...

119

## We work on more projects

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

- Vision:

fixed teams - - - - - semi fixed teams - - - - - no teams

230

## Why TaskCycle?

- Reflection and Preflection (PDCA - Quick feedback)
- Not working on anything less important
- Learning to know what to promise
- And then living up to our promises
- Taking responsibility
- Getting the info to be able to carry the responsibility
- Coping with interrupts
- Active Synchronization
- Calibration of estimations on the TimeLine
- Taming Parkinson's Law and Student Syndrome

231

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## What to do with the time gained?

If our original requirements are done in 70% of the time, what do we do with the 30% gained?

1. Choosing the next project
2. Continuing evolutionarily adding extras
3. Beware of Parkinson's Law!
4. Extending the horizon of the project to assure success

131

## Extending the project horizon to success

- Many projects end at: Hurray, it works!
- If customer success is paying our salaries, shouldn't we make sure success is going to happen
- Now a lot of quality requirements suddenly make sense:
  - User friendliness  - Usability
  - Intuitiveness - Learnability
  - Installability
  - Serviceability - Maintainability

133

## TimeLine exercise example

- **Preparing for student exams**

234

## What we did
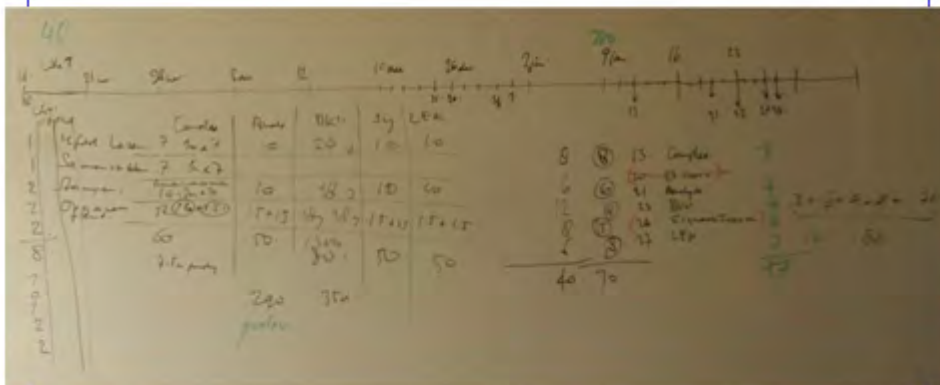


235

## TimeLine exercise for *your* Project

- What is the FatalDate, how many weeks left
- What is the expected result (←Business Case / Reqs)
- What do you have to do to achieve that result
- Cut this into chunks and make a list of chunks of activities
- Estimate the chunks (in weeks or days)
- Calculate number of weeks
- Compensate for estimated incompleteness of the list
- How many people are available for the work
    1. More time needed than available
    2. Exactly fit
    3. Easily fit
- Case 1 and 2: work out the consequence at this level
- Case 3: go ahead (but don't waste time!)

136

# Business Case

337

---

## Business Case

- **Why are we running a project ?**
- **The new project improves previous 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*

338

## Higher Productivity

- All functionality we produce *does already exist*

- The real reason for running our projects is creating *better performance*

- Improvement of value, productivity, success, happiness for our customers through users

139

## 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 |
| | 265 | 25.25 | min |
| **Usability.Intuitiveness:** | | | |
| 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

140

## 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
- So, how many Business Cases will there probably be in your project ?

241

## Nobody's working on the project yet
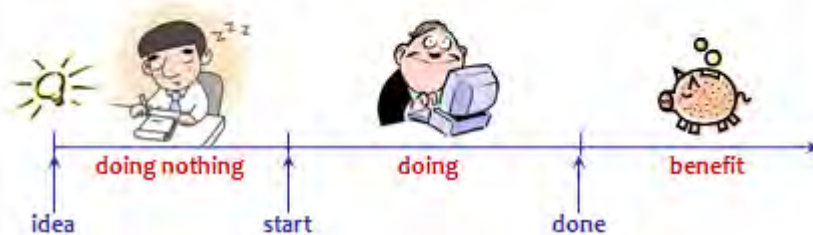


doing nothing          doing          benefit

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

242

## Business Case

*First develop the problem interdisciplinarily, 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 and maintain the Business Cases ??*

- **Stakeholders**
- **Expected Return on Investment (RoI)**
  Benefit of doing – Cost of doing – Cost of delay – Cost of doing nothing
- **Total LifeCycle**

143

## $0^{th}$ 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 may average if we estimate several pieces**

144

## RoI - Return on Investment

doing only what is needed,
and doing it more efficiently

more people?

Ostrich
(letting it
happen)

more people??

start use

return

investment

145
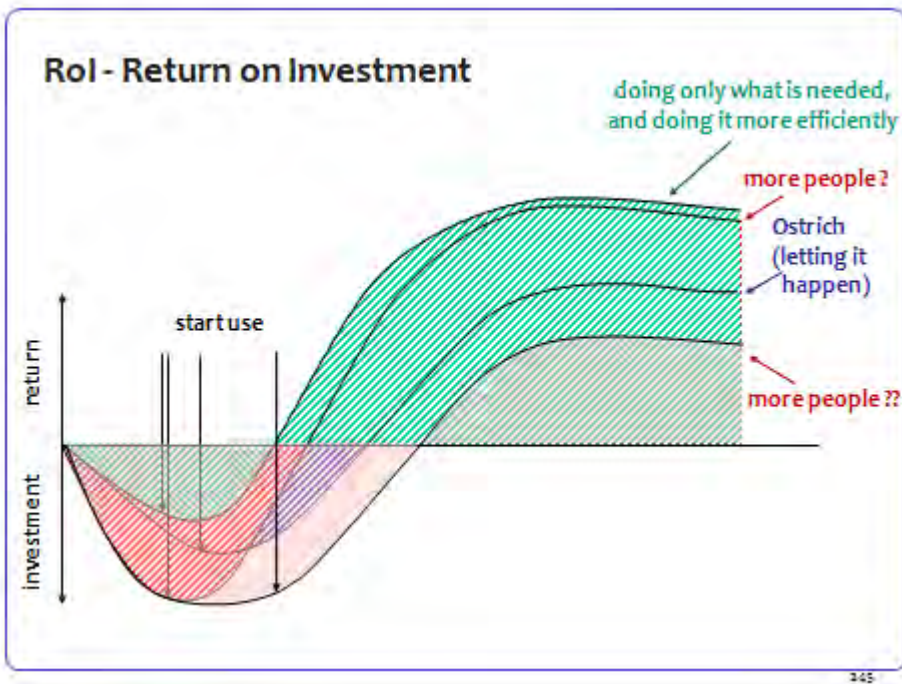
## Business Case exercise

(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 delay – Cost of doing nothing

146

# Stakeholders & Requirements

347

## Stakeholders are people

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

348

Victims can be a big Risk

Narita Airport ...?

149

---



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?

150

## 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?

## What is going to pay you?

Stakeholder gains clear?

|  | no | yes |
|---|---|---|
| **yes** | find out gains | go ahead ! |
| **no** | stop ! | hobby ? |

Stakeholders known?

## Stakeholder exercise

- Write down a list of Stakeholders for your project environment
- Discuss with neighbour

153

## Wish Specification

**Nice Input**

154

## 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 ?

## Requirements

- What Stakeholders need
- What the customer wants, he cannot afford
→ What the project is planning to satisfy:
   This is what you'll get, if you let us continue

## Top-level Requirement  for the Organization

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

- So:

    We should profit from our work

- But:

    Customers provide our income

157

## 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

158

## Customer Success

- **Customer**
  - Orders the system
  - Pays for the system

- **Success and failure**
  - *Through* users of the system
  - More general: *through* Stakeholders

159

## 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

160

## Fallacy of *'all'* requirements

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

## Requirements Compliance ?

The question is not
- to comply with the original Requirements

but
- to make it work according to the top level requirement

Watching over the first line of shoulders

## 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
  - Surely other fields are similar - can you think of examples ?

163

## 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 !

164

## 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 ....

165

### Extended ISO Model

**Reliability**
maturity
fault tolerance
recoverability
*availability*
*degradability*

**Efficiency**
time behavior
resource behavior

**Portability**
adaptability
installability
conformance
replaceability

*safety?*

*real time behavior?*

*dependability*

**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*

*this-ability?*

*that-ability?*

ISO9126 - QUINT

166

## Constraints

*Constraints are 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**

167

## 5 times "Why?"

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

- **Freud and Jung:**
  - Problems are in our sub-consciousness
  - Solutions pop up
  - Solutions are how people tell their problems

- **What's your problem ?**
  - If there's no problem, we don't have to do something

- **Within 5 times "Why?"**
  **we usually come down to the real problem to solve**
  - Otherwise we will be perfectly solving the wrong problem

168

## Requirements exercise

• What are the Requirements of your current project?

Exercise:

• Write down the 3 most important requirements
• With their Stakeholders (Who's waiting for it?)

269

## 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

270

## The 2nd requirements paradox



finish

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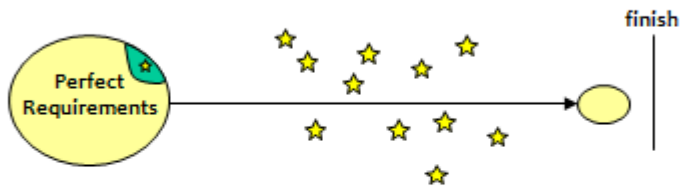
- We don't want requirements to change, however,
- Because requirements change now is a *known risk*:
  We must *provoke* requirements change *as early as possible*

271

## Requirements should be at one place only



Company Standards

+

ProductRange Requirements

+

Product Specific Requirements
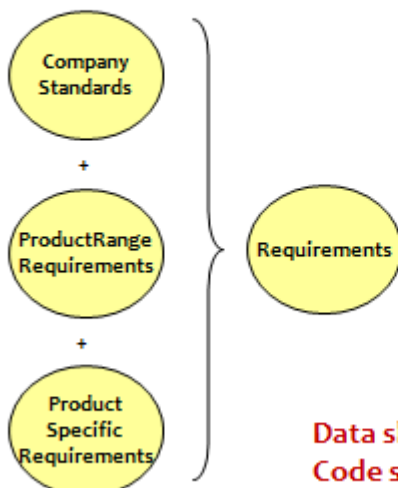
Requirements

Data should be at one place only
Code should be at one place only

272

## Attributes of a Good Requirement

**A Good Requirement is:**

| | | |
|---|---|---|
| Relevant | Clear | Unique |
| Complete | Elementary | Verifiable |
| Consistent | Concise | Traceable |
| Unambiguous | Correct | No solution |
| Feasible | Has no *weak words* | |

**Does your project have Good Requirements?**

273

## Rule

**Rule: 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

274

## Lord Kelvin (1824 - 1907)

When you cannot measure it,
when you cannot express it in numbers,
your knowledge is of a meagre and unsatisfactory kind

...

It may be the beginning of knowledge,
but you have scarcely in your thoughts advanced
to the stage of science

275

## Can we measure everything ?

- How beautiful is music, a painting, a man/woman ?
- Beaty of a car, spacecraft, piece of electronics ?

- What's your weight ?
- How clever are you ?
- Can you quantify 'Love' ?

Not everything that counts can be measured,
and not everything that can be measured counts
Einstein

276

## Why quantifying ?

- **The most important things in life cannot be measured, the more important they are, the less you can measure them**
  Ron Baker

- **Still**
  - The measurement isn't a goal in itself
  - Trying to define the measurement provides us with *better understanding* what the problem really is about

277

## Requirements with Planguage
ref Tom Gilb

**Definition:**
**RQ27:** Maximum Response Time
**Scale:** Seconds between <asking> for information and <appearance> of it.
**Meter:** Add a function to the software to measure the maximum response time value and the <range of values> per <working day>.

**Benchmarks (Playing Field):**
**Past:** 3 sec (our previous product)
**Current:** 0.6 sec [competitor y, product x, 2009] ← Marketing Survey Jan 2010
**Record:** 0.2 sec [competitor x, product y]
**Wish:** 0.2 sec [2012] ← customer's head of R&D, 19 Feb 2010, <document ...>
**Note:** Less than 0.2 sec is not noticed by the user, so there is no use in trying to be better than 0.2 sec

**Requirements:**
**Must:** 1 sec [99%] ← project-contract
**Must:** 1.5 sec [100%] ← project-contract
**Goal:** 0.5 sec ← project-contract

278

**Design to a Quality Requirement**
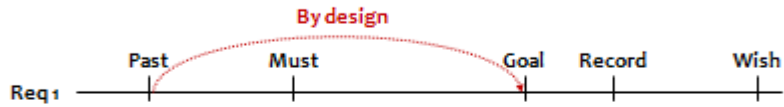
179



**Step-by-step example**

Gradually reaching required response time

180
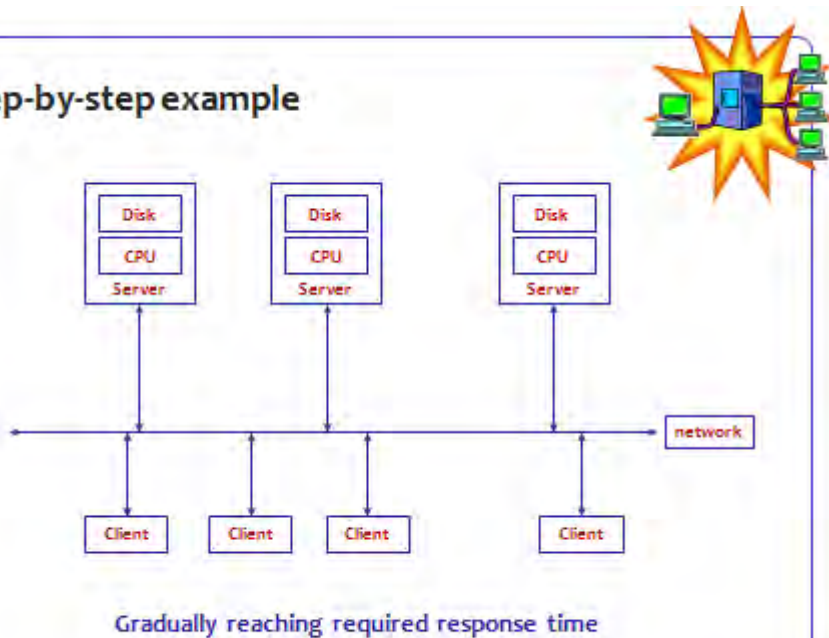
## Design to a Quality Requirement — one step at the time
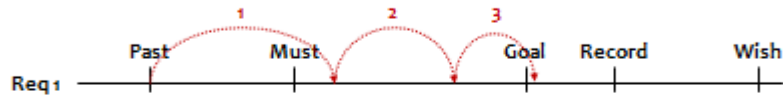
If the Quality Requirement is composed of several elements, start with the best ROI



## Design to Multidimensional Quality Requirements
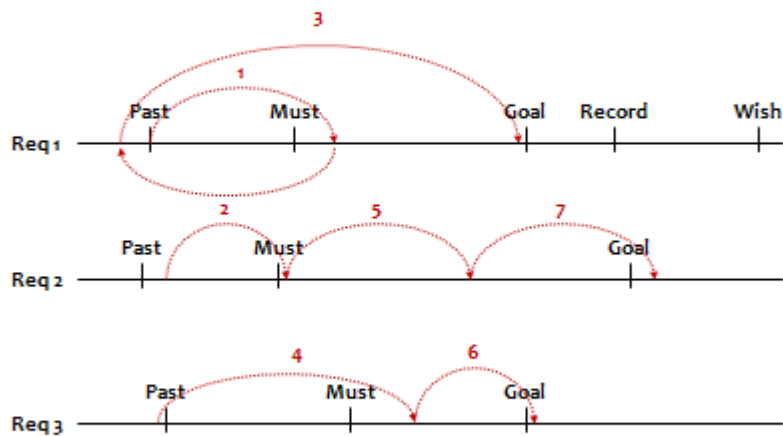
## Requirements Case

- **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

183

## Objective: Monetary Donations



Monetary Donations

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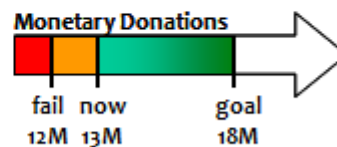
| | |
|---|---|
| **Name** | Monetary Donations |
| **Scale** | Euro's donated to non-profits through our website |
| **Meter** | Monthly Donations Report |
| **Fail** | 12M |
| **Now** | 13M [2008] ← Annual Report 2008 |
| **Goal** | 18M [2009] |

Ref Ryan Shriver
ACCU Overload Feb 2008

184

## Objective: Volunteer Time (Natura) Donations

Volunteer Time Donations

fail    now          goal
2700hr  2800hr       3600hr
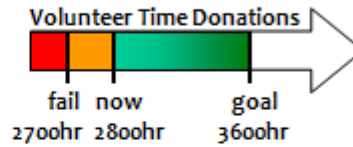
| Name | Volunteer Time Donations |
|------|--------------------------|
| Scale | Hours donated to non-profits through our website |
| Meter | Monthly Donations Report |

| Fail | 2700 hr |
|------|---------|
| Now | 2800 hr [2008] ← Annual Report 2008 |
| Goal | 3600 hr [2009] |

Ref Ryan Shriver
ACCU Overload Feb 2008

185

## Goal: Market Share

Market Share

fail  now       goal
5%    6%        10%
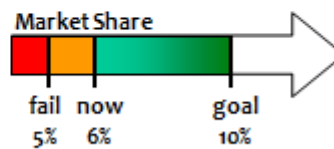
| Name | Market Share |
|------|--------------|
| Scale | Market Share %% online giving |
| Meter | Quarterly Industry Report |

| Fail | 5% |
|------|-----|
| Now | 6% [Q1-2009] ← Quarterly Industry Report |
| Goal | 10% [Q1-2010] |

Ref Ryan Shriver
ACCU Overload Feb 2008

186

## 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

187

## 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 | Reliability | months | | 12 | 12 |
| MTTR | Mean time to repair | Reliability | 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 |

188

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## Examples of Scales

**Environmental Noise**
dBA at 1.0 meter

**System Security**
Time required to <break into the system>

**Software Maintainability**
Time needed from <acceptance of 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**
Number of FTE's (Full Time Equivalent)

189

## 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}

190

## 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

191

## Dependability is a Complex Concept

**Dependability.Availability**
Readiness for correct service
Scale: % of <TimePeriod> a <System> is <Available> for its <Tasks>

**Dependability.Reliability**
Continuity of correct service
Scale: Mean time for a <System> to experience <Failure Type> under <Conditions>

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

**Dependability.Security**
Free from intrusions (theft, alteration)
Scale: Time required to <break into the system>

192

## Portability Goals



Must

Goal ●

Goal ●

Goal ●

Better

Wish

Release:  1st   2nd   3rd   4th   5th

293
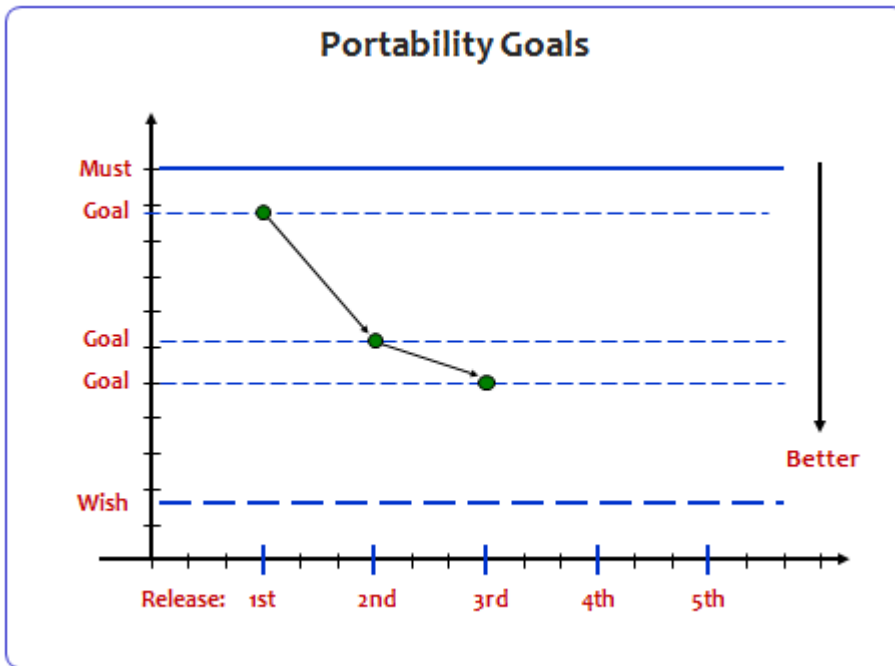
## Nice things

- **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 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

294

More information:
www.malotaux.nl

## Example: Road-Pricing in the Nederlands

**Realize a road-pricing system in four years**

1. Fitting an electronic system in 8 million cars
2. Camera's for number plate recognition
3. Central system for data processing and invoicing
4. Law changes by politicians (tax law, traffic law)
5. Price differentiation for time, place, emissions

**Will this succeed?**

195

## Requirements exercise: (groups of 2 or 3 people)

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

**Try to use:**

| Definition: | Benchmarks: | Requirements: |
|---|---|---|
| • Ambition | • Past | • Must/Fail |
| • Scale | • Current | • Goal |
| • Meter | • Record | |
| • Stakeholders | • (Wish) | |

Note: you may end up with a different requirement
than you started with ... (what is it really about ?)

196

| Ambition | |
|---|---|
| Scale | |
| Meter | |
| Stakehldrs | |
| Past | |
| Current | |
| Record | |
| Wish | |
| Must/Fail | |
| Goal | |

197

# Architecture and Design

198

---

## 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) ⬇

199

## 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

300

## Design Process

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

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

301

Think Process for Problem Solving

## Impact Estimation example

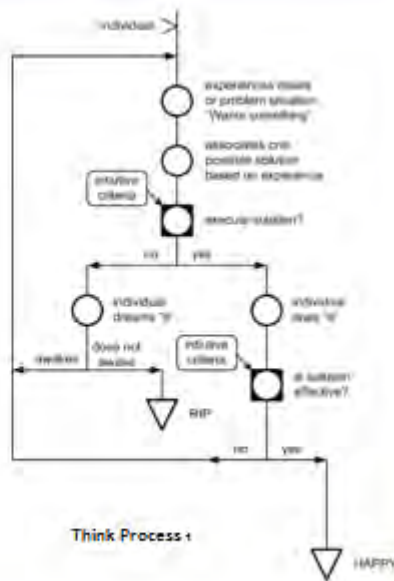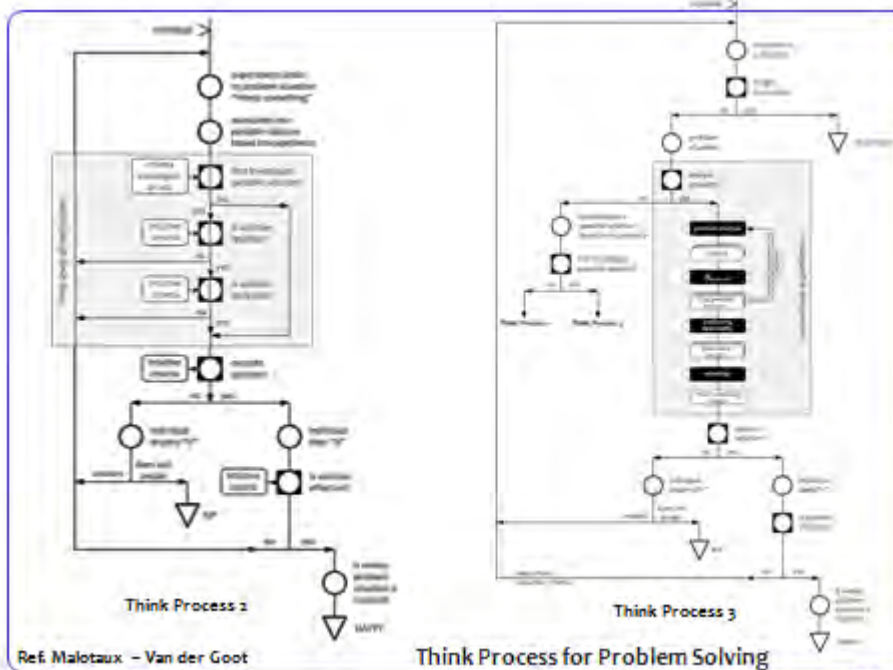| Impact Estimation | Monthly Donations | Facebook integration | Image & video uploads | Total effect for requirement |
|---|---|---|---|---|
| €€ donations 13M€ → 18M€ | 80% ±30% | 30% ±30% | 50% ±20% | 160% ±80% |
| Time donations 2800hr → 3600hr | 10% ±10% | 50% ±20% | 80% ±20% | 140% ±50% |
| Market 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

304

## Impact Estimation Example

| | 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

305

## 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 | Benefits / Cost | Benefits / Cost | Benefits / Cost | |

306

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

Needs: what do we need — Requirements

Options: how can we do it — Design

Requirements

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

Design

Requirements

Design

Requirements

Design

**Design provides the Requirements for the next level**

307
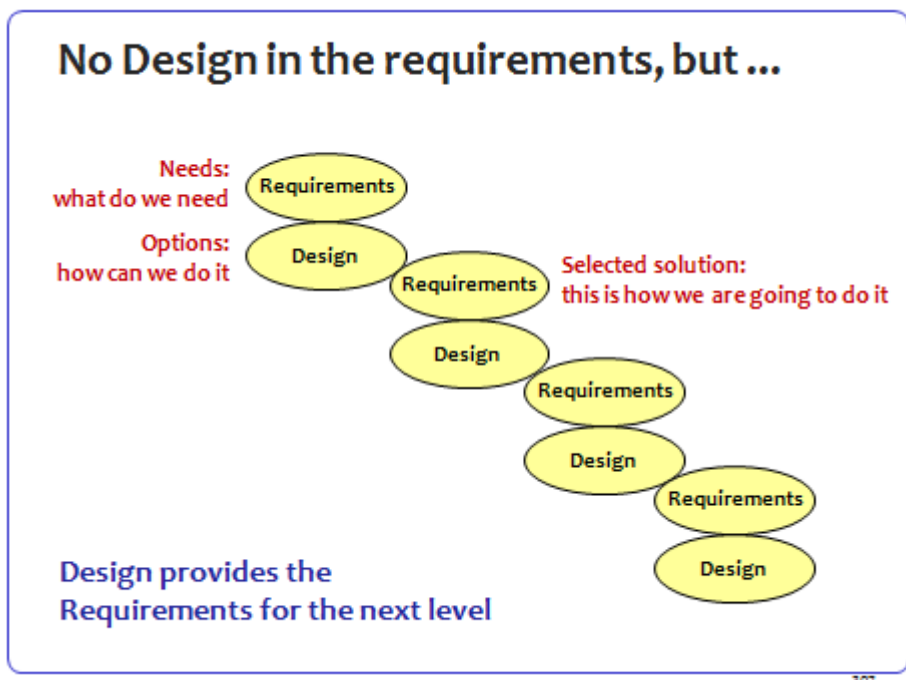
## Priorities are essential

- 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

308

## 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
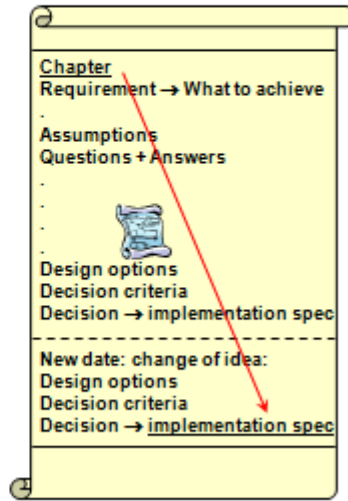  into implementation

309

## 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
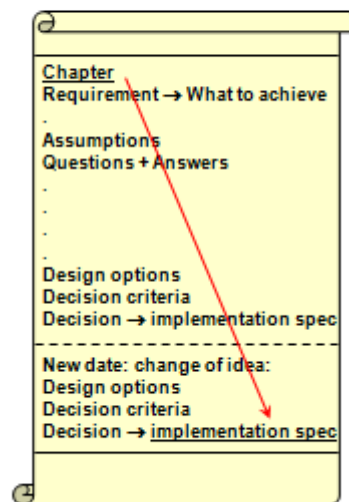Decision → implementation spec

310

## 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

311

**Four processes**

Target process: what you should do

Actual process: what you do

Perceived process: what you think you do

Official process: what you are supposed to do

312
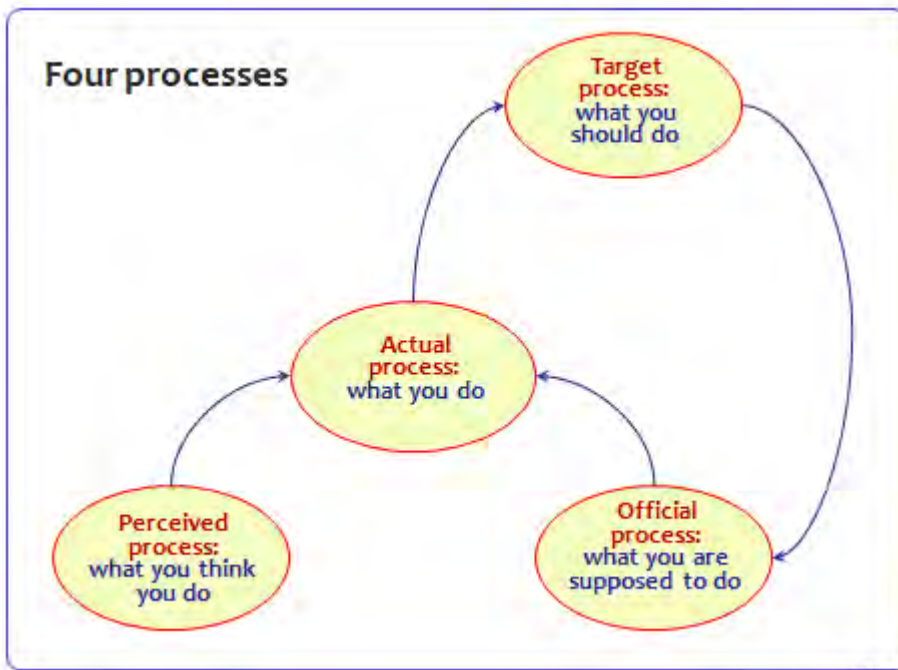
# Risk

313

## 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

314

## 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

315

## Our own Risk

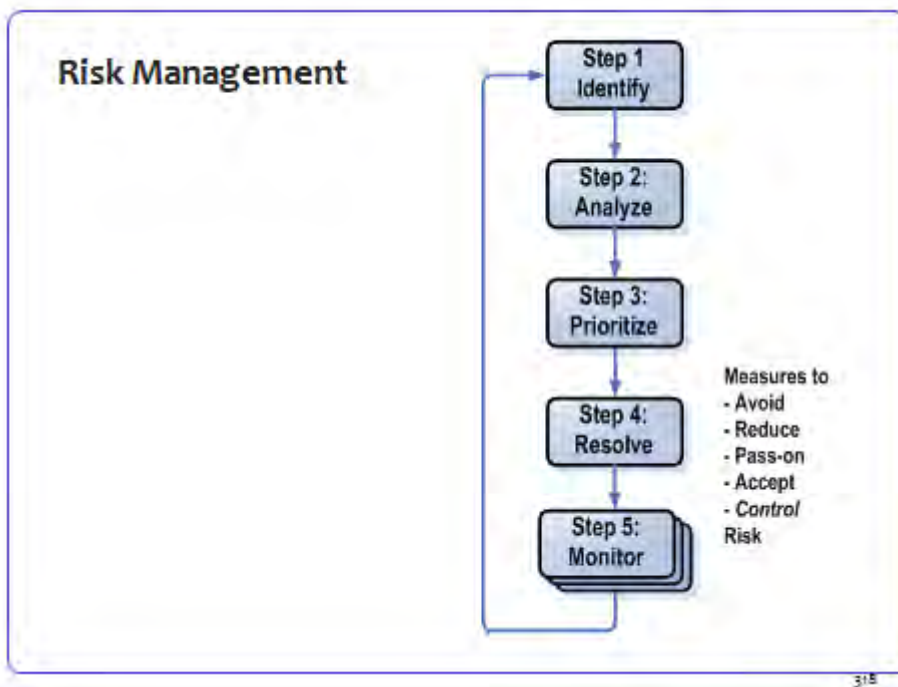**Getting less profit than expected**

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

316

**Risk Model**

Confidence of Estimate — Confidence of Estimate — Confidence of Estimate

worst case ?

Probability of Event **X** Probability of Impact **X**

Event — Impact — Cost

Event Driver(s) — Impact Driver(s) — =

Prevention Plans — Contingency Plans — Risk Value

$$V_R = P_e * P_i * C$$

317



**Risk Management**

Step 1: Identify

Step 2: Analyze

Step 3: Prioritize

Step 4: Resolve

Step 5: Monitor

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

315

## Prioritize Risk?
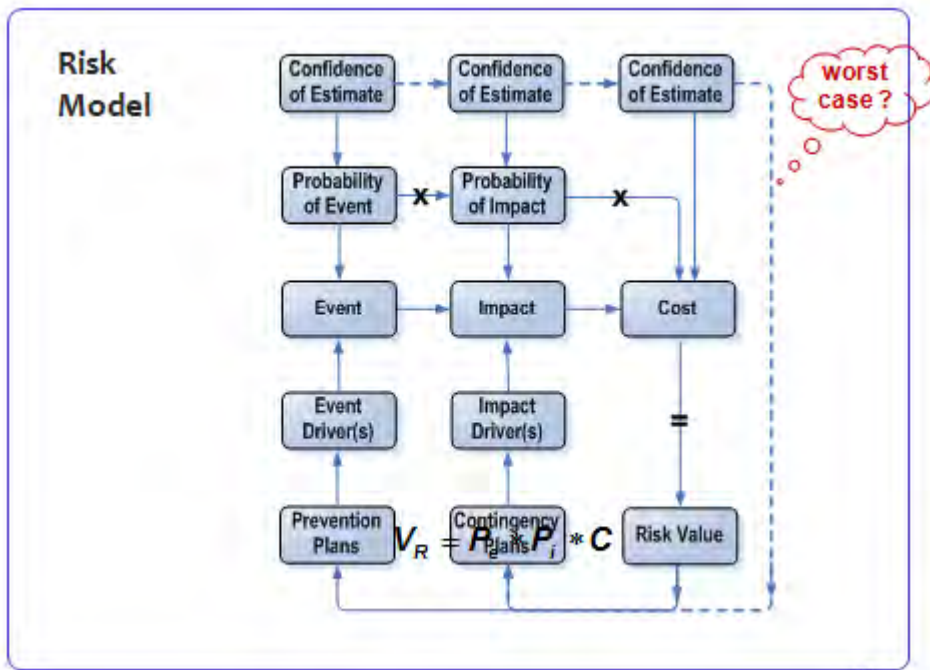
ref. INCOSE SE Handbook
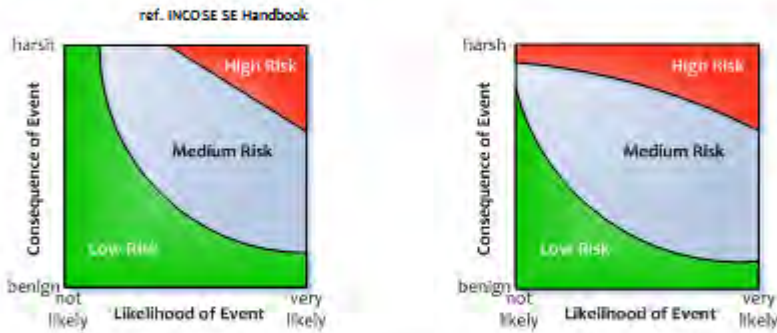


**Risk Priority = Likelihood x Consequence ??**

319

## 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

320

## Mathematical Risk Management can be risky

ref
Carlo Rafele,
David Hillson,
Sabrina Grimaldi



## 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

## 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

313

## What are Risks in your Projects?

- …
- …
- …

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

314

## Controlling Risk by design
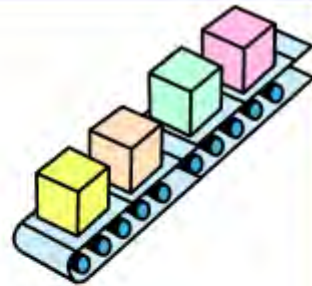
- **Every project is unique**
  (otherwise it's production)

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)

315

## Many known risks are hardly risks

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

$$V_{Risk} = P_{event} * P_{impact} * C \qquad P_{event} = 1$$
$$P_{impact} \to 0$$

- 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

316
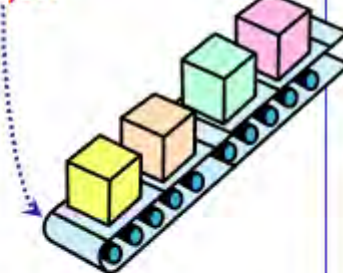
**Risk mitigation**

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

| control | avoid | reduce | pass-on | accept |
|---------|-------|--------|---------|--------|
| $P_I\downarrow$ | $P_I = 0$ | $\downarrow\downarrow\downarrow$ | subcontract ?? insure | self-insure |
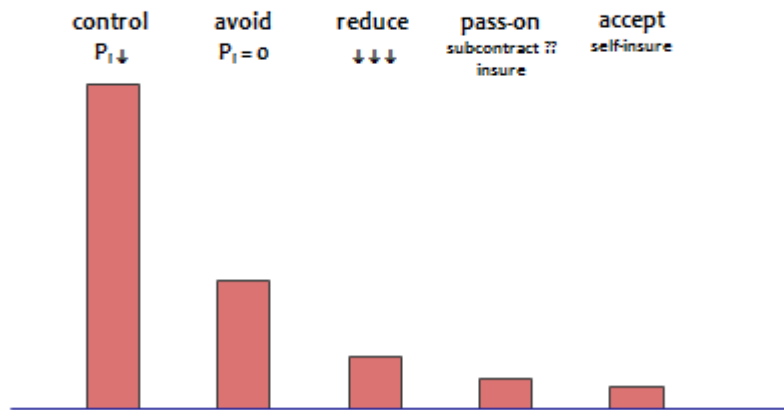
317

**Swiss Cheese model**

ref James Reason

Hazards

Losses
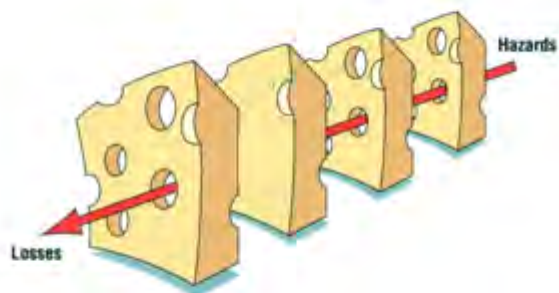
**Can we add some cheese from Holland?**

318

## Product Risks

Root-cause of risk

- **Development**
  - Requirements errors
  - Incorrect Assumptions
  - Design errors
  - Calculation errors
  - Implementation errors
- **Maintenance**
  - Incorrect or insufficient maintenance
- **Use**
  - Operator errors
  - User errors
  - Victims
  - Technical errors

*All these risks are introduced by humans*

*People like you and me*

329

## 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

330

## 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

331

## Development time is limited

- Having not enough development time is a safety risk
- Working overtime is a safety risk
- We must use the limited available time well
- Therefore we should plan well
  - What do we have to do?
  - What can we do?
- Taking the consequence
  - (People tend to do more than necessary, especially if they don't know exactly what to do)
  - What are we going to do?
  - What are we *not* going to do? (saving time!)
- Carefully select the most important work
- Finding out as quickly as possible whether we are doing the right things

332

## What do we do to stay on time ?

- **Short term:**
  - Many short cycles: continual result and opportunity to adjust
  - Changing from optimistic to realistic estimation
  - Estimating hours, not days
  - Estimation is a TimeBox
  - Focus on Value Delivery
- **Longer term**
  - TimeLine
  - Continuous stakeholder consequences / agreement

  - TaskSheet (first think, then do)

333

## What do we do to stay on budget ?

- **Budget is constraint in time, money, resources**
- **Clearly define how much time, money and resources are available: Hard limit !**
- **If we don't get a limit, we'll create our own** (BudgetBox)
- **As quickly as possible finding out what is really needed**
- **If it doesn't fit the budgets, it won't be done** (unless … )

334

## How to deliver Quality

- **Early and frequent deliveries to check requirements and assumptions: only doing the right things**
- **Stakeholder identification en regular contact to find out together what is really needed**

335

## Mastering unclearness

- **Analysis tasks:**
  - What do I know now
  - What do I still not know
  - What do I still have to find out
  - Defining and estimating Tasks
  - Tight TimeBox

- **Result of analysis task:**
  - What's the benefit ?
  - What's the cost ?
  - How important is it (where does it fit in) ?

336

## Mastering supplier risks

- Active synchronisation
- Showing the importance of FatalDates
- Proactively solving problems
- Foreseeing delays and doing something about it, before it's too late

337

## Mastering client risks

- Are they capable to receive the result ?
- Are they ready to do something with the result ?
- Is the client system ready ?

Frequent deliveries solve these problems

338

## When is the project done ?

- **Using exit criteria**
- **Goal levels met**
- **Making sure the users will become more successful**
  - If the users won't be more successful, they're not going to generate our salaries

- **Risk management:**
  - Making sure our salaries <u>can</u> and <u>will</u> be paid

339

## Remaining Risks

- **Risks beyond our control**
  - Within or beyond control of client
  - Within or beyond control of suppliers

  - Example: Satellite launching date moving
    - Extra costs for maintaining project support
    - Extra costs of users waiting, having prepared using the satellite
    - Meanwhile new requirements may emerge
  - Example: Delays caused by compulsory purchase procedures
  - Weather
  - Traffic Jams ?

340

More information:
www.malotaux.nl

### 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

341

### 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)

342

## 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

343

## 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

344

More information:
www.malotaux.nl

### 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*

345

### 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

346

## If we add something ...

If we add something, something else will not be done

now                                    FatalDate

347

## Problems with externally furnished components
Boehm 1991

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

348

### 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
  (using the right architecture !)

349

### 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

350

## 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%**

351

## 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 ?**

352

# Verification & Validation

# Testing

# QA

393

---

Do we deliver Zero Defect products?

• What do you think is acceptable?

394

## Is Zero Defects possible ?
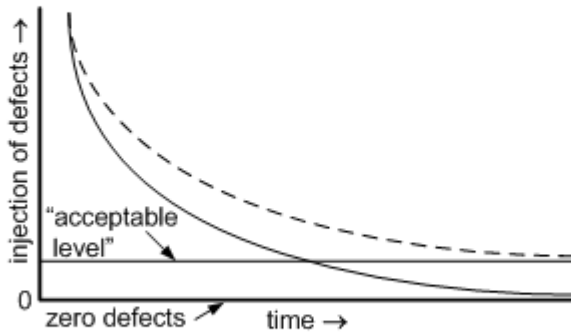
- **Zero Defects is an asymptote**
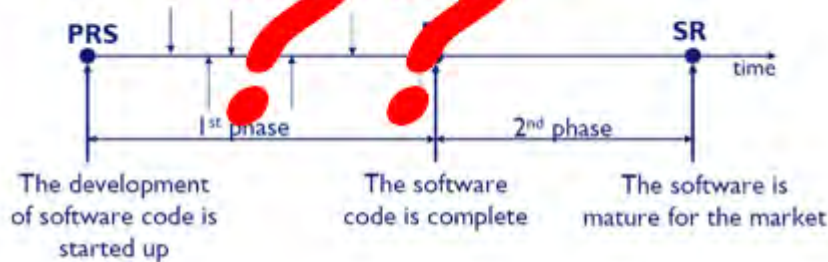


- **When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately**

355

**PHILIPS**                                      **TU/e** technische universiteit eindhoven

## Software development process



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

## 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)

357

## All we have to do ...

- **A defect is the cause of a problem experienced by any of the stakeholders while relying on our results, ultimately affecting the customer**
- **Making the customer more successful implies no defects**
- **All we have to do is delivering results without defects**
- **Do we?**

- **Is being late a defect?**

358

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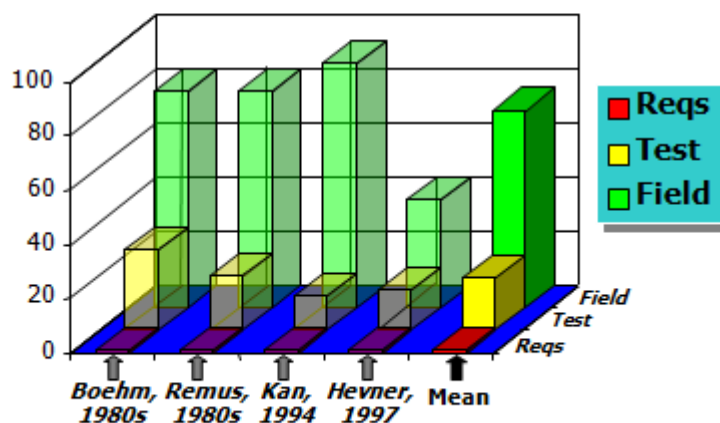
And there is a large budget to do something about it:

- 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

359

## Cost of Requirements Defects

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



360

## Typical Defect Injectors (cost breakdown)

Implementers

Designers

Other

7%

28%

10%

55%

Requirements Specifiers
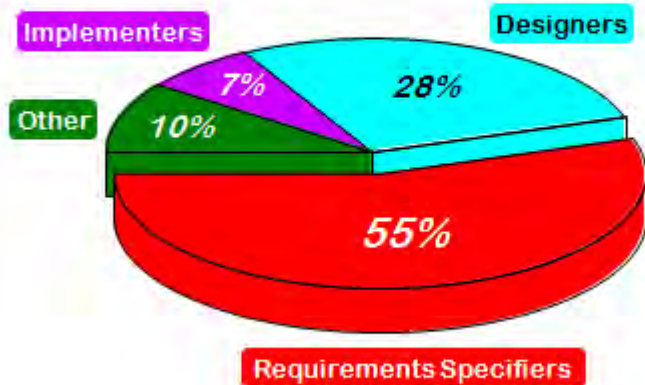
After Bender Associates, *1996*

DM
361

## 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

361
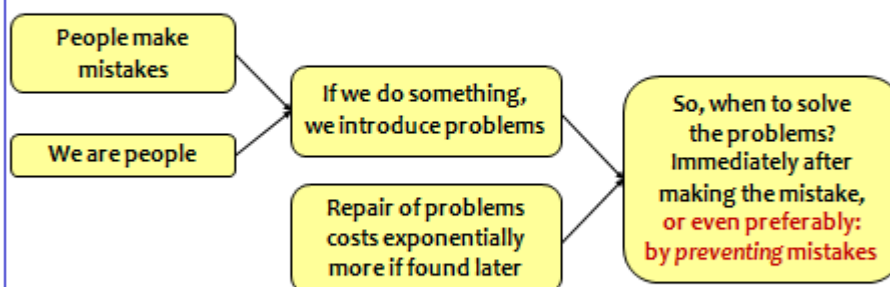
## Where do we make mistakes ?

- **Wish specification**   Thank you, nice input

- **Business Case**   Why are we doing it

- **Requirements**   What the project agrees to satisfy

- **DesignLog**   Selecting the 'optimum' compromise and how we arrived at this decision

- **Specification**   This is how we are going to implement it

- **Implementation**   Code, schematics, plans, procedures, hardware, documentation, training

- **Process Log**   Describing how and why we arrived at which current practices

363

## Documents and Sources



363
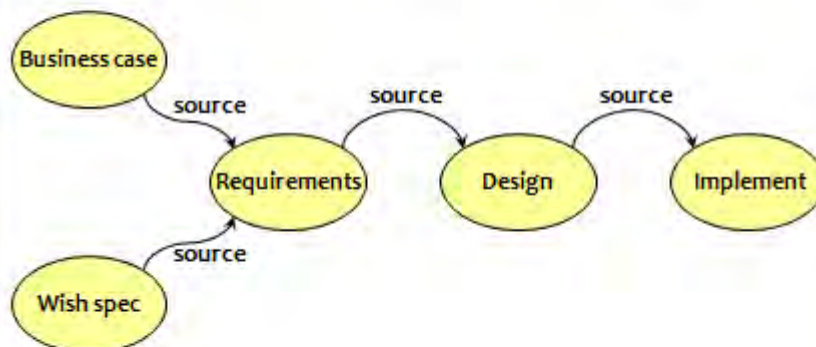
## Defects

- 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



## 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 ?

Debugging ? ? ?

367

---

## Bugs are so important, are they really?

- People make mistakes; we are people
- "Software without bugs is impossible"
- Testers try to find as may bugs as possible
- 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

368

## 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**

369

## 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

370

## 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

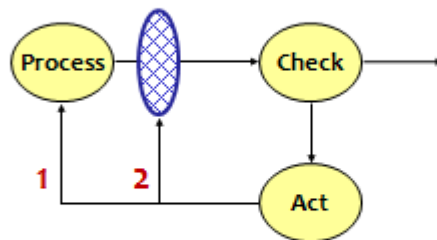371

## 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

372

More information:
www.malotaux.nl

## Testing is checking correctness



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

373

## 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

374

## Who is the customer 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.

    → **Developers are the customer**

- **Testers help 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**

375

## Developers are continuously optimizing

- **The product**
  how to arrive at the most effective product (goal !)
- **The project**
  how to arrive at the most effective product effectively and efficiently
- **The process**
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

376

## Testers are continuously optimizing

- **The product**
  how to arrive at the most effective product (goal !)
- **The project**
  how to arrive at the most effective product effectively and efficiently
- **The process**
  - Finding ways to do better
  - Learning from other methods
  - Absorbing those methods that work better
  - Shelving those methods that currently work less

377

## Evo Testing
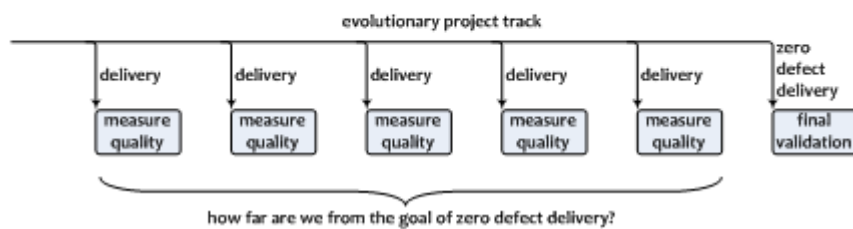


evolutionary project track

delivery · measure quality
delivery · measure quality
delivery · measure quality
delivery · measure quality
delivery · measure quality
zero defect delivery · 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**

378

## Further Improvement
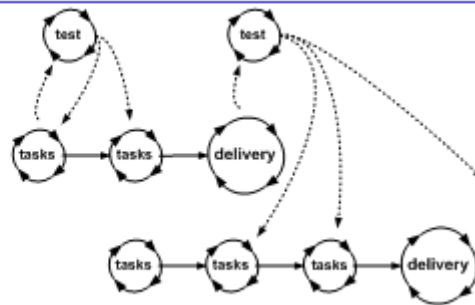
- Tester's customer is "the developers"

- 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

379

## 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

380

## 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

381

## 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

382

## When are we done with testing?
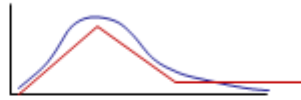
- **Conventional:**
  - Number of defects found per day less than n
  - Defect backlog decreased to zero
  - Prediction by curve fitting based on early found defect numbers
  - Using historical data
  - Other?

- **Evo:**
  - The project is ready at the agreed date, or earlier
  - That includes testing

383

## 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)
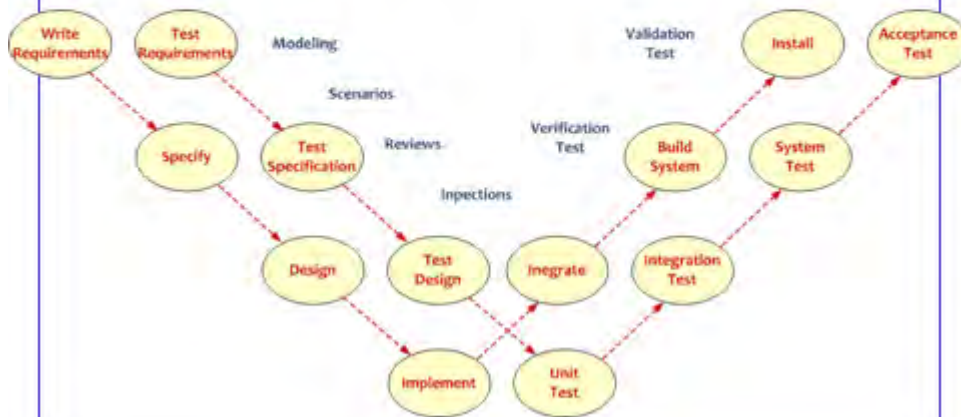  What would testing say about these?

384

# Predictable Projects
### Delivering the Right Result at the Right Time

## Remember the W-model
### but also remember: all models are wrong ...



## Ways to achieve better quality?

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

**Prevention !!**

## CR/PR/RI Database

Focus on Prevention

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

387

## Anything we think must be done goes through the *Candidate Task Mechanism*
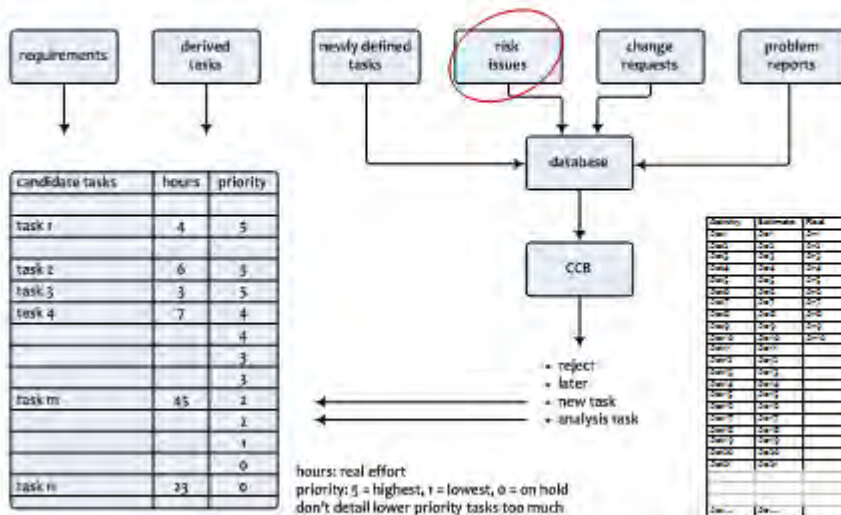
# Reviews
# &
# Inspections

389

## Are you reviewing?

390

## 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

## 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*

**Niels Malotaux**
# Predictable Projects
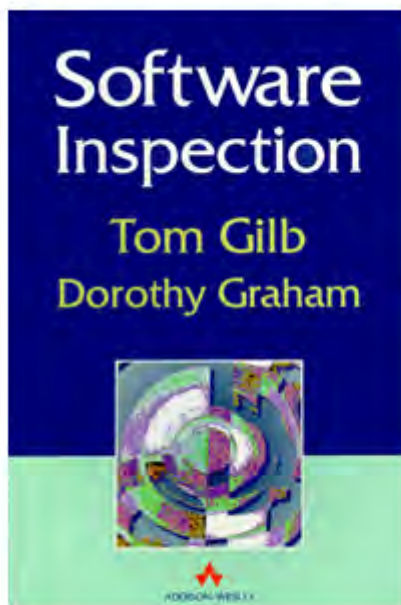**Delivering the Right Result at the Right Time**

## Inspection

- Most rigorous form of review
- Pioneered by Fagan (IBM) (paper 1976)
    - Locating all the defects in a work product
- 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?**

393

Software
Inspection
Tom Gilb
Dorothy Graham

ADDISON-WESLEY

A ready to use recipe ...

394

## 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?**
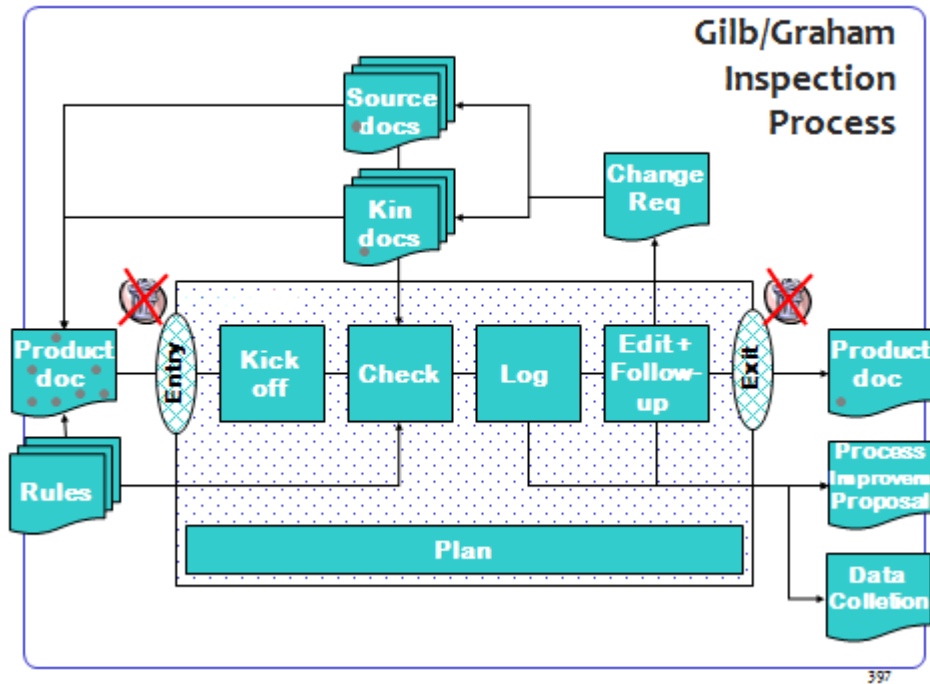
395

## 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 generate documents in the first place?
- **Interesting side-effect:**
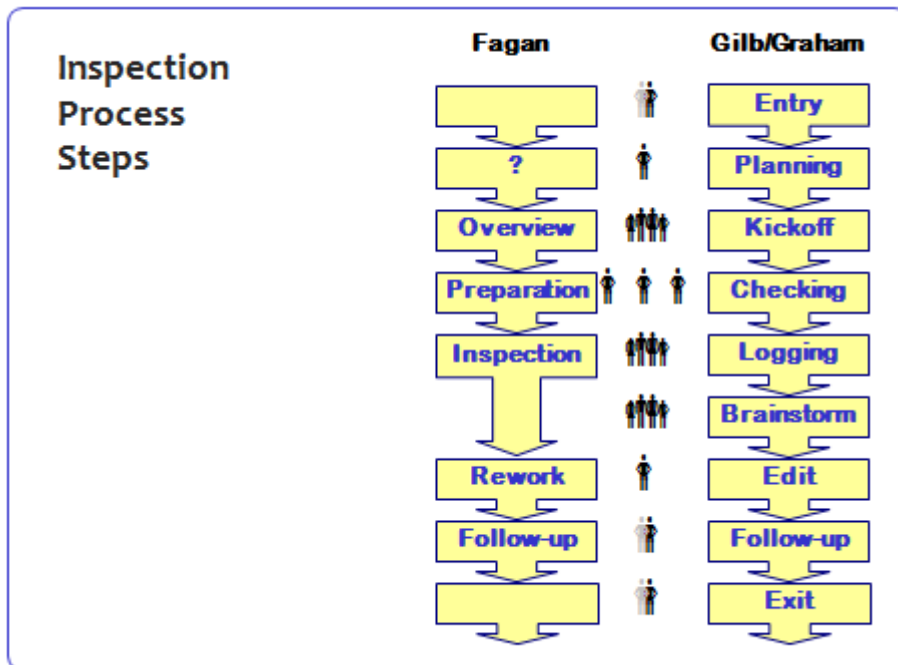  People get to know each others documents efficiently

396

**Niels Malotaux**
# Predictable Projects
## Delivering the Right Result at the Right Time



Gilb/Graham Inspection Process



Inspection Process Steps

## Gilb/Graham Concepts
## Entry and Exit Criteria

**Once the quality level of a specification is known, there are three possible paths forward:**



Well above exit criteria: Process failure! Recreate after training or process improvement

Somewhat above exit criteria: Rework or enlarge inspection sample

Meets exit criteria: Success! Exit
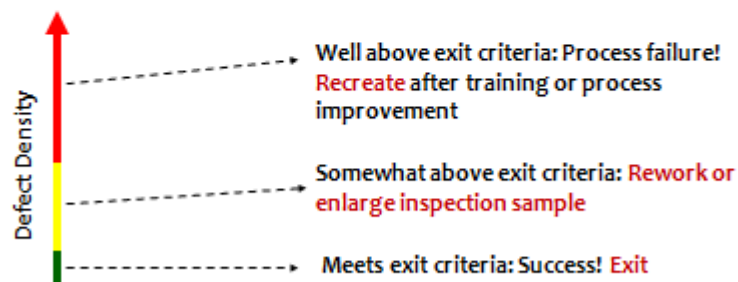
399

## Optimum Checking Rate

- 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")

400

More information:
www.malotaux.nl

## Inspection Thoroughness

*Ref. Dorothy Graham*
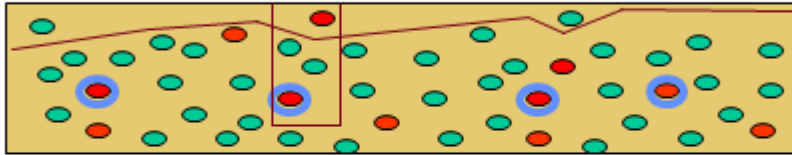


- Inspection can find deep-seated defects
- <u>All of that type</u> can be corrected
- Needs optimum checking rate

- 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                                                                                               403

## Gilb/Graham Inspection



Gilb/Graham inspection differs from other types of inspection in some or all of these ways:

- **Purpose:**
  Quantifying quality, not searching for all defects
- **Controlled reading rate:**
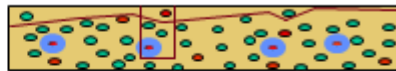  The material being inspected is read very slowly in order to identify as many defects as possible (deep vs shallow sample)
- **Sampling:**
  Only samples are inspected to optimize time and effort investment while maintaining the reading rate
- **Entry/Exit Criteria:**
  Quantified entry and exit criteria used to guide the inspection effort
- **Rules:**
  Written rule sets used to locate and classify defects

E8                                                                                               404

More information:
www.malotaux.nl

## Gilb/Graham Concepts
## Reading Rate
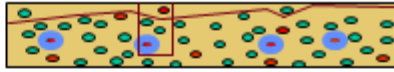
- Default recommended reading rate is one logical page per hour, lower than in many other inspection methods

- This ensures adequate time to locate the vast majority of latent defects in the specification

- Supporting documents, rules, etc. can be read at any speed
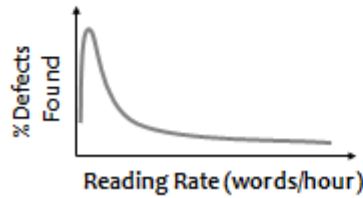
**Read too fast and you will miss most of the defects!**

E 8                                                                                405



16 page
Inspection
Manual

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

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

## 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)
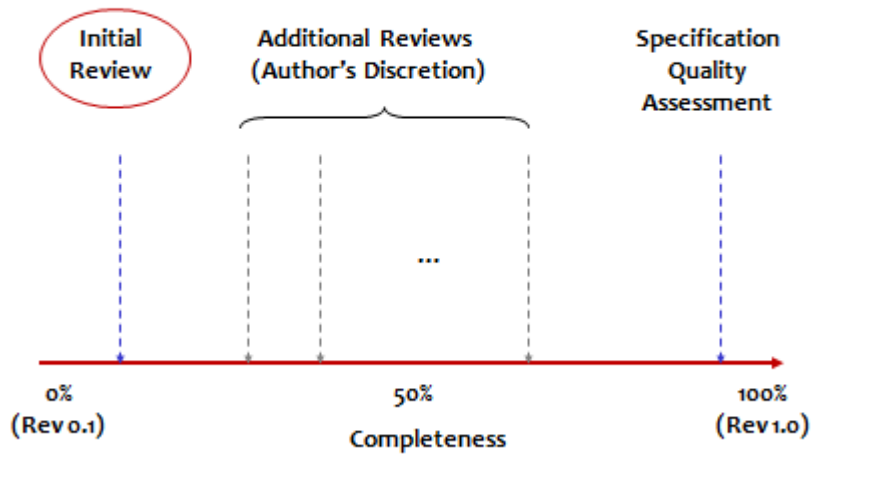
E 8

407

## 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***

E 8

408

## 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**

E 8                                                                                      409

## 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

E 8                                                                                      410

## 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**

E 8
411

## 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

E 8
412

More information:
www.malotaux.nl

## 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                                                                                             413

## 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                                                                                             414

## 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

E 8                                                                                                415

## Cleanroom Software Development

- **Design** (Mathematical proof)
- **Verification** (by others)
- **Implementation**
- **Verification** (by others)
- **No unit test**
- **Only Integration Test** (by others)
  (Test is Running Code)

- **Verification is for finding defects**
- **Testing is for *not* finding defects**

416

**Cleanroom** (ref Allan M. Stavely: Toward Zero Defect Programming)

- The purpose of Inspection is to eliminate defects
- Exit criterion for design:
  - One design statement materializes as 3 to 10 code statements
- Checklists of typical errors we make
- No Unit Test - Developer does not run software!
- Testing:
  - Finding as many of the remaining defects as possible
  - Too many errors discovered
    - → previous steps are not being done properly
    - → redo previous steps (do not "repair")

417

## Testing in Cleanroom

- Testing is an important part of the process, but it is done only after verification is successfully completed
- Testing is done:
  - Primarily to measure quality
  - Secondarily to find defects that escaped detection during verification
- Number of bugs per thousand lines of code <10 after verification, compilation and syntax checking
- Very good teams produce 2.3 bugs per kLoC and reject code with 4 or 5 bugs per kLoC
- No attempt is done to try to salvage rejected code by debugging
  - The code is sent back to the developers to be rewritten and reverified
  - Then it is tested as a completely new product
- Usage based testing ⎫
- Risk based testing  ⎬ Statistical testing

418

# Metrics

419

## Useful Evo metric

### Size of the smile on the customers face

- In many cases, the Evo attitude and techniques replace the need for metrics
- I did not say always

- In Evo, we consume metrics immediately for learning, rather than collecting them
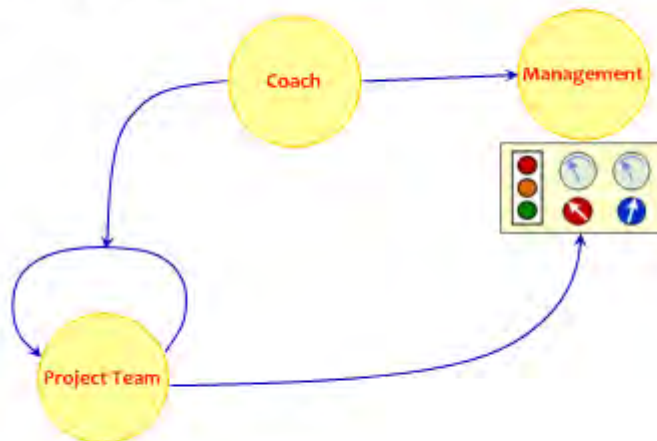
420

**Niels Malotaux**
# Predictable Projects
**Delivering the Right Result at the Right Time**

## Why would we measure ?

- Goal
- Question
- Metric
- Consequence
- Does it help ?

## Local Loop Principle

## Metrics used

- **Metrics for the project**
  - Ratio Real time used / Estimated time
  - Calibration factor
    $\Sigma$ Realized / $\Sigma$ Estimations
  - Predicted date of what will be done when
    Today plus Sum of Calibrated Estimations
  - Ratio plannable / unplannable hours (default: 2/1)
  - Available time, available budget (less is better)
  - Cost of one day of delay
    - Cost-Of-Doing-Nothing
    - Project Cost + Lost Benefit



- **Metrics for the product**
  - Quantified requirements (ref Planguage - Tom Gilb)
  - Rate of improvement on quantified requirements (Impact Estimation)

413

## Metrics techniques used

- **Just-enough metrics** (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- **Consuming the metrics immediately**
  - Not putting them in Databases
  - Using immediately for learning and improving
  - Feeding intuition to come up with better estimations
  - Preventing failure
- **Time-boxing** (not Feature-boxing)
  - Minimizes the need for tracking
- **Calibration**
  - Coarse metrics provide accurate predictions (Law of Large Numbers)
  - Moving Sense of Urgency from the end towards now

414

## Estimation techniques used

- **Just-enough estimation** (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- **Changing from optimistic to realistic predictions**
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- **0th order estimations** (ball-park figures)
  - For decision-making in Business Case and Design
- **Simple Delphi**
  - For estimating longer periods of time in TimeLine
  - For duration of several (15 or more) elements of work
- **Simpler Delphi**
  - Same, but for quicker insight
  - Recently added by practice
- **Doing something about it** (if we don't like what we see)
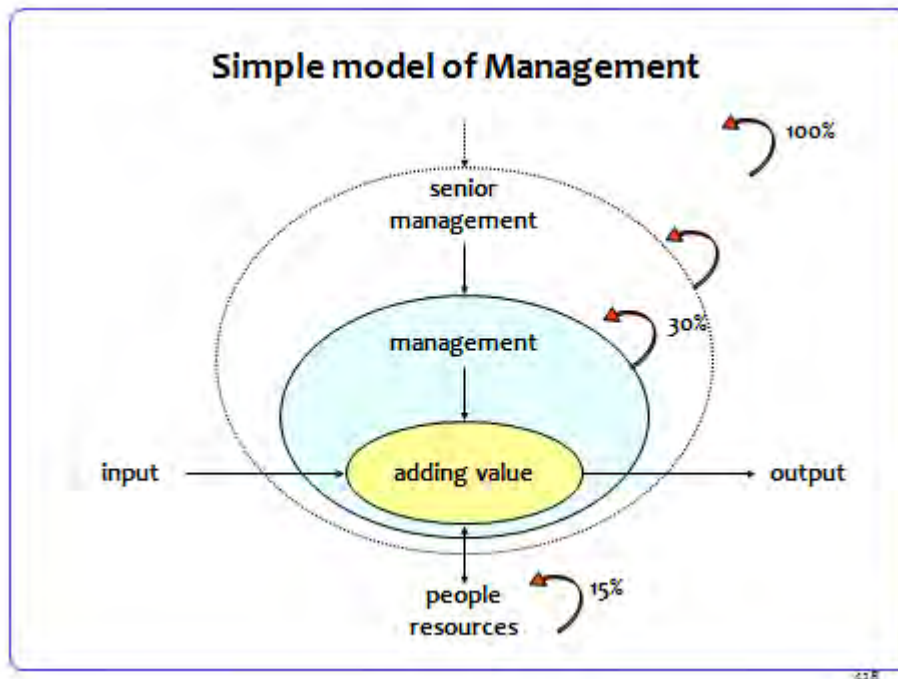  - Taking the consequence
  - Saving time

415

# Management Issues

## Managers have to learn

- Managers should be coaches
- Not police
- Managers have to learn to understand the Evo approach

## Simple model of Management



senior management

management

input → adding value → output

people resources

100%

30%

15%

## Management Questions on Tasks

- **Is the Project under Control?**

- **Show me !**
  - No "holes" in OK's
  - All available, plannable time planned
  - TaskSheets used
  - Results used
  - Prompt explanation in case of discrepancies

# Introduction Issues

430

## How Lean is Evo ?

- Kaizen - PDCA, preflection
- Waste - Only do the most important things, constantly seeking to do less, without doing too little. Preflection
- Value - Only produce Value, quantifying Value, increasing Value every Evo step
- Pull - Who's waiting for it? Defer to last moment
- Poka Yoke - Murphy
- Etc

- Evo fills in the Lean principles with pragmatic action
- Lean ← Toyota ← Ford ← Benjamin Franklin

431

## Order of Introduction

- **Evo-day**
- **Tasks**
- **Deliveries**
- **TimeLine**
  - Sense of urgency
  - Value delivered
  - Better performance pays salaries
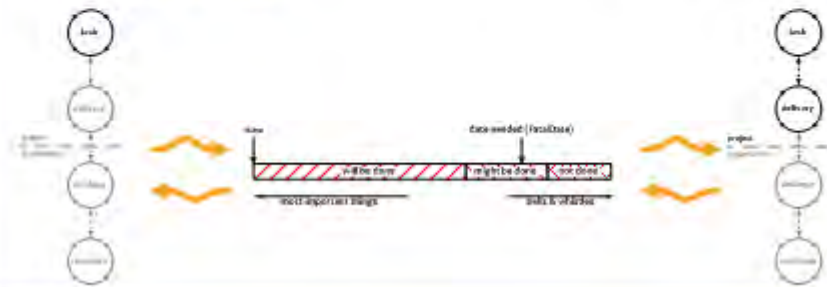- **Requirements engineering details**
- **Reviews & Inspections**

431

## 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, what *not*, and why

- **Weekly Evo day**
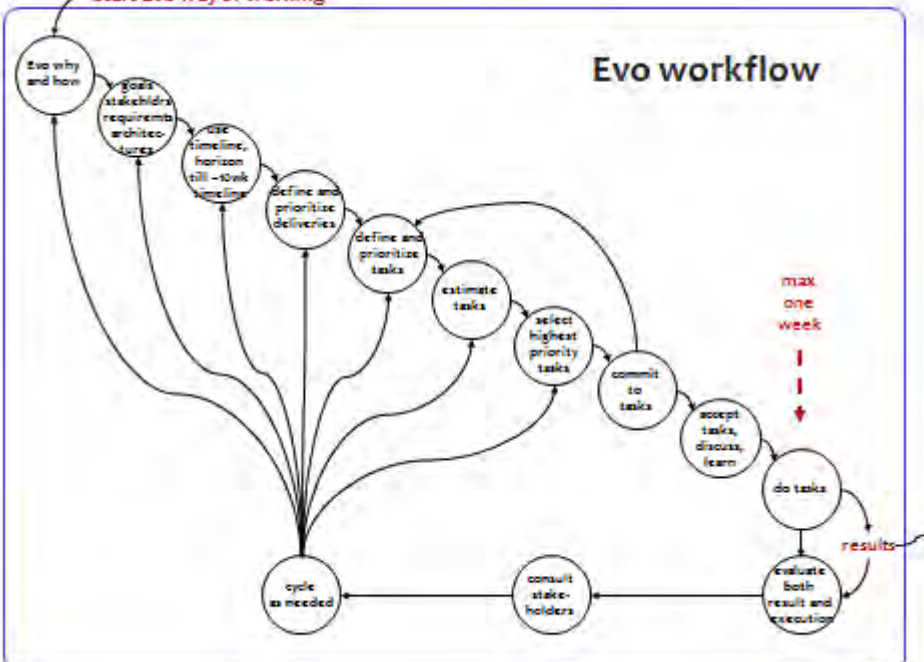  - Execution of the 3-step procedure

433

**Niels Malotaux**
## Predictable Projects
### Delivering the Right Result at the Right Time

**Cases**

436

## Case 7: A "failure"

- **Seasoned project manager: "Good idea, but…"**
- **No emphasis on TimeBoxing**
- **Didn't try to understand Delivery and TimeLine concepts**
- **Many "hero's" in the team**
  - I can do whatever I want. I know so much, they won't fire me anyway
- **No Sense of Urgency both in team and from management**
  - Management by fear
  - Management asks different things every week
  - Management asks impossible results

If you don't apply Evo, Evo does not fail, the project does

437

## Case 9: US company

- Started with 10 people of a 40 people project (don't over-eat)
- Carefully *designed* the Evolutionary Introduction of Evo
- Now the whole team is routinely working the Evo way
- Including 8 people in India
- Didn't miss a milestone since
  (Average time overrun before Evo was 20%)
- They still hardly believe this is possible

Evo works with larger and distributed projects

438

## Case 10: Managers

- Managers asked
  "Can I use this for my own busy schedule?"
  - Write down what you have to do
  - Add effort hours
  - List in order of priority
  - Check how much time available this week
  - Draw line at 2/3 of the available time
  - Decide what to do and what not to do
- Manager Reports:
  "This made me 40% more productive immediately!"

439

**Niels Malotaux**
# Predictable Projects
### Delivering the Right Result at the Right Time

## Finally

440

## Magic words

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

441

## Magic Sentences

- Customer may never find out about our problems
- Evo metric: Size of the smile of the customer
- Delivery Commitments are always met
- People tend to do more than necessary
- Can we do less, without doing too little
- What the customer wants, he cannot afford
- Who is waiting for that?

## Why would the *product* need Evo ?

- *We* don't know the real requirements
- *They* don't know the real requirements
- *Together* we have to find out (stop playing macho!)
- What the customer wants he cannot afford
- Is what the customer wants what he needs?
- People tend to do more than necessary
  especially if they don't know exactly what to do

If time, money, resources are limited,
we should not overrun the budgets

## Why would the *project* need Evo?

- **Are we effective?** (producing Results)
- **Are we efficient?** (optimally using the available time)
- **Are we actively learning from our mistakes?** (PDCA)
- **How do we estimate, plan and track progress?**
- **How do we handle interruptions?**
- **Did we learn from feedback per project?** (project evaluation)

444

## When would we *not* need Evo

- **Requirements are completely clear, nothing will change: use waterfall** (= production)
- **Requirements can be easily met with the available resources, within the available time** (Still, Evo can make it faster)
- **Everybody knows exactly what to do**
- **Customer can wait until you are ready**
- **Management doesn't know what to do with the time saved**
- **No Sense of Urgency**

    Use Evo only on projects you want to succeed

445

## 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

446

### www.malotaux.nl/Booklets                    More

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

447