Predictable Projects

Delivering the Right Result at the Right Time

Niels Malotaux

N R Malotaux Consultancy

+31-6-5575 3604 niels@malotaux.nl www.malotaux.nl

Niels Malotaux

- **Independent Project and Organizational Coach**
- **Expert in helping optimizing performance**

- **Helping projects and organizations very quickly to become**
  - **More effective** – doing the right things better
  - **More efficient** – doing the right things better in less time
  - **Predictable** – delivering as predicted
- **Getting projects on track**

Result Management

Schedule, we'll try to keep ☺

Woensdag	3 februari
9:30~10:45	1:15
break	0:15
11:00~12:30	1:30
lunch	1:00
13:30~14:40	1:10
break	0:10
14:50~15:50	1:00
break	0:10
16:00~17:00	1:00

Who is who ?

- Systems Engineer ?
- Architect ?
- QA ?
- Project Manager ?
- Product Owner ?
- Scrum Master ?
- Team Member ?
- Customer ?
- Manager ?
- Consultant ?
- Coach ?

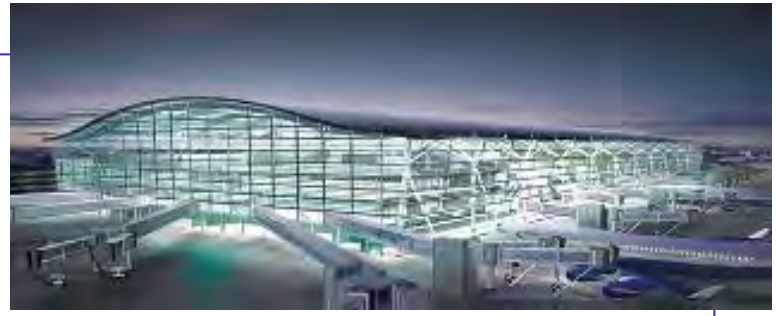
Did you prepare ?

- **The top-3 stakeholders of your work** (*Who* is waiting for it?)
- **The top-3 real requirements for your work** (What are they waiting for?)
- **How much value improvement the stakeholders expect** (3 or 7?)
- **Any deadlines** (No deadlines: it will take longer)
- **What you should and can have achieved in the *coming 10 weeks***
  (Will you succeed? - Failure is not an option!)
- **What you think you should and can do the *coming week* in order to achieve what you're supposed to achieve** (Make sure not to plan what you shouldn't or cannot do - At the end of the week everything you planned will be done)
- **What value you will have delivered by the end of the week and how to prove it**
- **Any issues you expect with the above or otherwise with your work**

Predictable Projects ?

- **Any problems with projects ?**

Not every project is successful
(at first)



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

What is the most important requirement ?

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

- **Are they really ?**
- **How about your current project ?**

Fallacy of '*all*' requirements

- "We're done when *all* requirements are implemented"
- Is delivery time included ?
- Requirements are always *contradictory*
- Design is to find the optimum compromise between the conflicting requirements
- Do we really have focus on the *real* requirements ?
- Did the customers define *real* requirements ?
  - Usually even less trained in defining *real* requirements than we are
- What we think we have to do should fit the available time
- Instead of *letting it happen*, better *decide how it will happen*

Causes of Delay

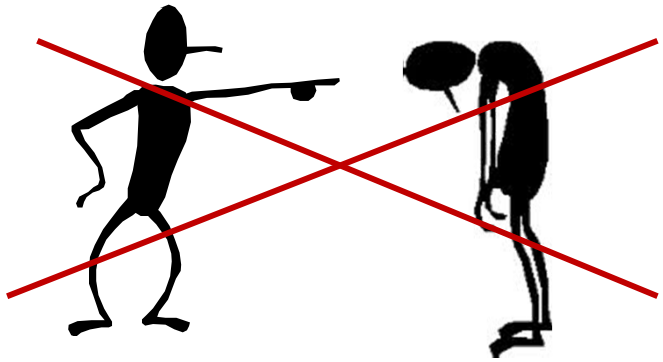
- **Some typical causes of delay are:**
  - Developing the wrong things
  - Unclear requirements
  - Misunderstandings
  - No feedback from stakeholders
  - No adequate planning
  - No adequate communication
  - Doing unnecessary things
  - Doing things less cleverly
  - Waiting (before and during the project)
  - Changing requirements
  - Doing things over
  - Indecisiveness
  - Suppliers
  - Quality of suppliers results
  - No Sense of Urgency
  - Hobbying
  - Political ploys
  - Boss is always right (culture)
- **Excuses, excuses: it's always "*them*". How about "*us*" ?**
- **What are causes of these causes ?** (use 5 times 'Why ?')

Causes of causes

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

Intuition often points us in the wrong direction

Isn't that the Responsibility of the Project Manager ?

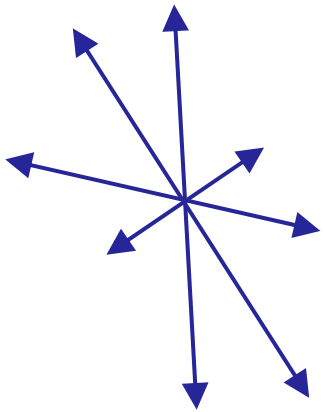


- The Project Manager is *responsible* for *delivering* the right result at the right time
- The Project Worker's work and decisions *determine* the *result* and the *time* it is delivered
- This makes everybody in the project implicitly *as responsible as* Project Management

Systems Engineering

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

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



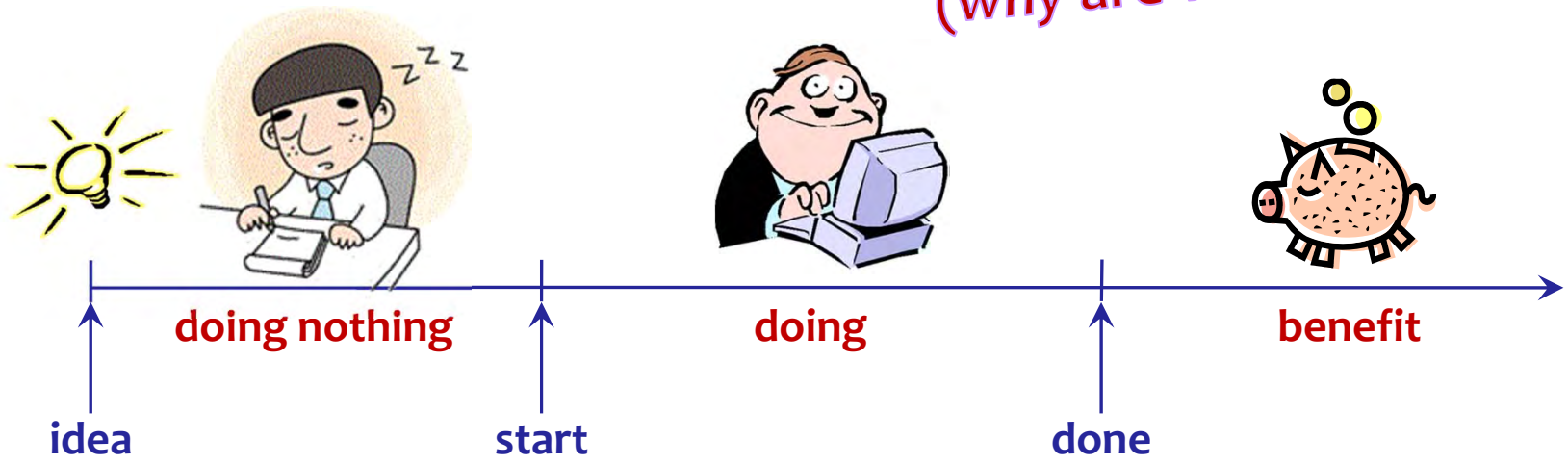
Multidisciplinary ↔ Interdisciplinary

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

The Importance of Time

Business Case

(why are we doing it)



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

Return on Investment (ROI)

- \+ **Benefit of doing** - huge (otherwise we should do an other project)
- – **Cost of doing** - project cost, usually minor compared with other costs
- – **Cost of being late** - lost benefit
- – **Cost of doing nothing yet** - every day we start later, we finish later

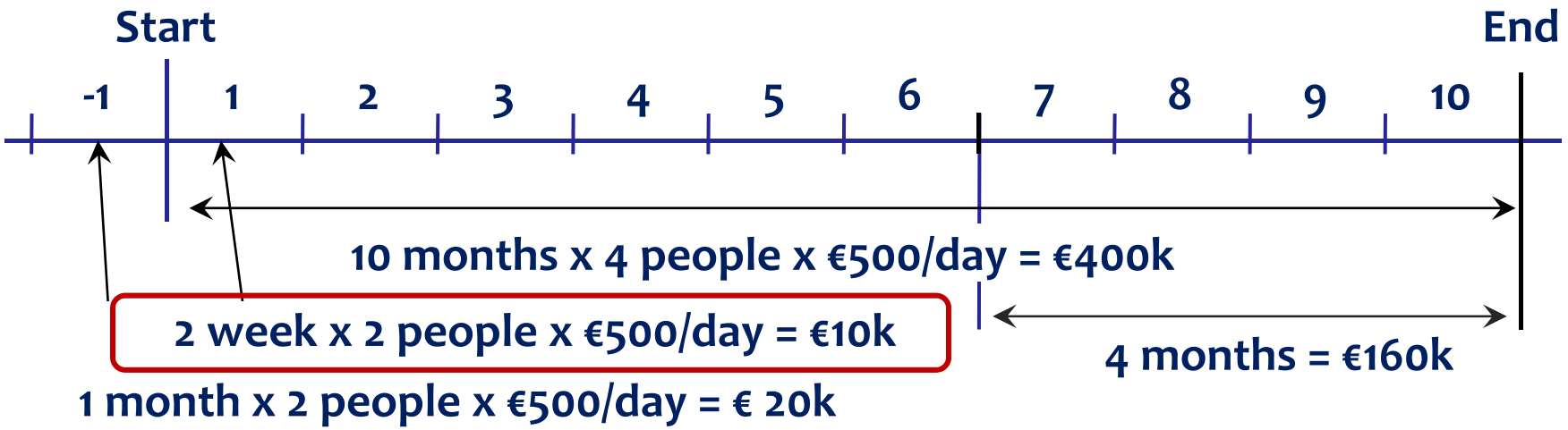
What is the cost of one day of (unnecessary) delay ?

- What is the cost of the project per day ?
- Do you know how much *you* cost per day ?
  Note: that's not what you get !
- If you don't know the benefit, assume 10 times the cost
- How can you make decisions, if you don't know ?

- No need for exact numbers - it'll be a lot anyway

- Do you know the benefit of your projects ?
- Do you know the penalty for delay ?
- Who is paying for the extra time ?

The Cost of Time



- We can save 4 months by investing €200k → "That's too much !"
- It's a *nicer* solution - Let's do 2 weeks more research on the benefits
- What are the expected revenues when all is done? → €16M/yr (€1.3M/mnd)
- So 2 weeks extra doesn't cost €10k. It costs €16M/26 = €620k
- And saving 4 months brings €16M/3 = €5M extra

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

The challenge

Failure is not an option

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

Goal for today

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

Warning:

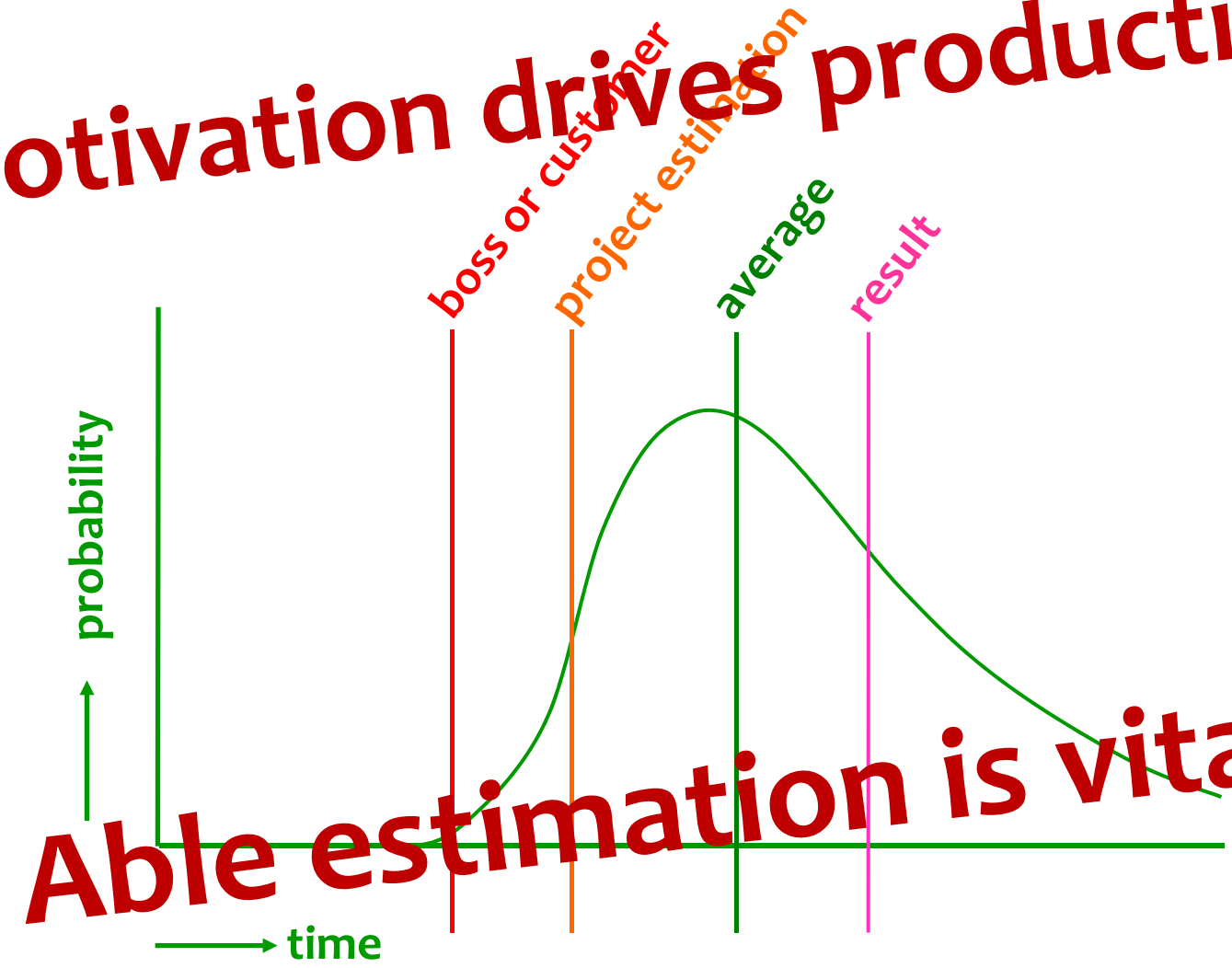
After today you don't have an excuse any more !

But you shouldn't need one either

Estimation Exercise

Lead time

Motivation drives productivity



Able estimation is vital

Estimation Exercise

Are you an optimistic or a realistic estimator?

Let's find out !

Project:

Multiplying two numbers of 4 figures

Example

```
    0000
    0000  x
----------
00000000
```

How many seconds would you need to complete this Project?

Is this what you did?

Defect rate

- **Before verification ?**
- **After verification ?**

Alternative Design (*how* to solve the requirement)

Another alternative design

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

What was the real requirement?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

Elements in the exercise

- Estimation, optimistic / realistic
- Interrupts
- Verification, verification strategy
- Defect-rate
- Design, design options
- Requirements
- Assumptions

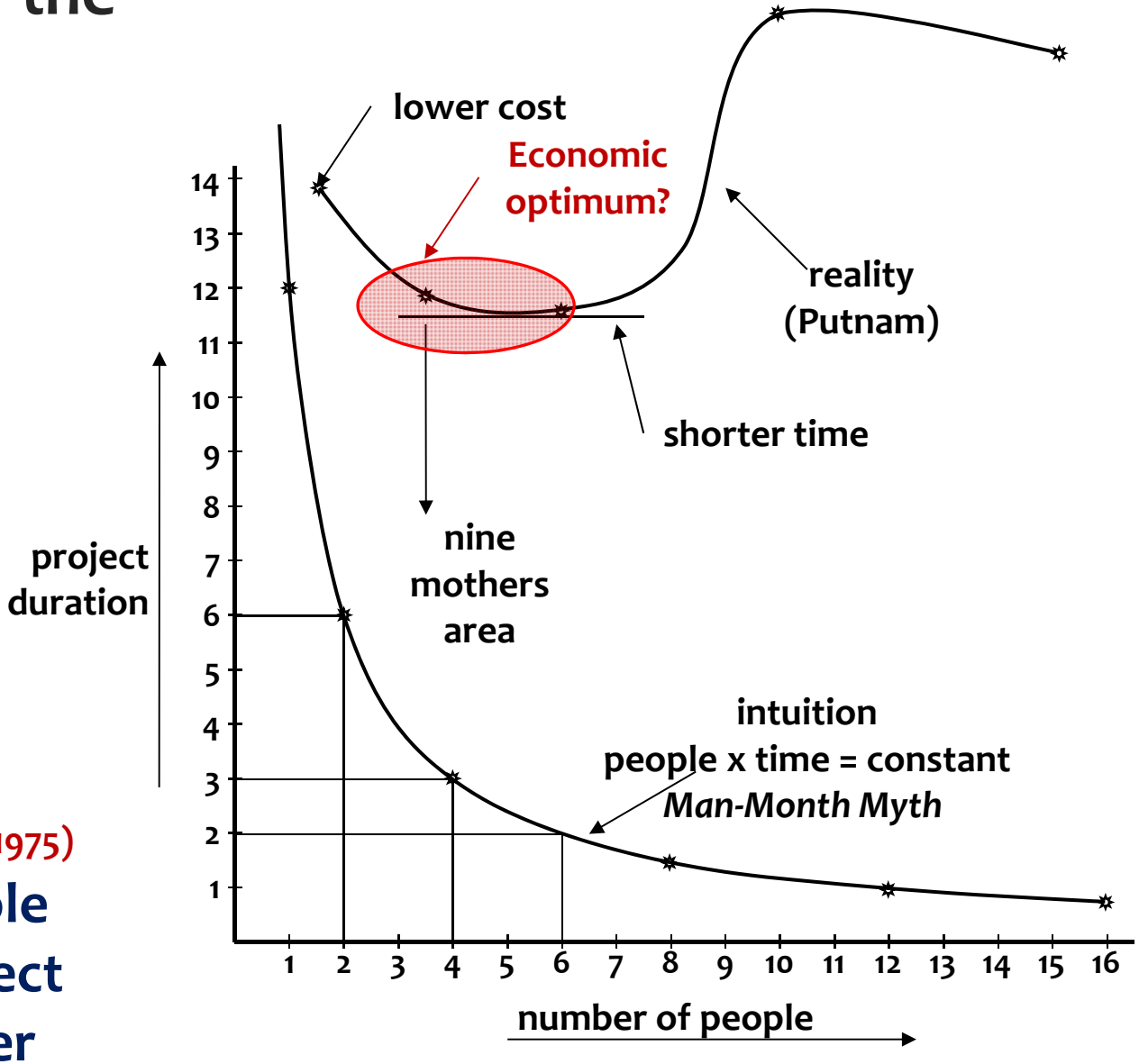
How can we be On Time ?

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

Intuition often guides us into the wrong direction

The Myth of the Man-Month



Brooks' Law (1975)

Adding people to a late project makes it later

Saving time

Continuous elimination of waste

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

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

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 projects
  - 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*

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)

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

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

Perception

- Quick, acute, and intuitive cognition (www.M-W.com)
- Intuitive understanding and insight (www.oxforddictionaries.com)
- What people say and what they do is not always the same
- 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!

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

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

  it's no excuse for failure of your project

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

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

We failed because of politics

- **Good politics:**
  - People decide differently on different values
- **Bad politics: hidden agenda's**
  - Say this, mean that - often even unintentionally
  - Politics thrive by vagueness
  - Facts can make bad politics loose ground

- **If you accepted the responsibility for the project, failure because of "politics" is just an excuse**
- **What did you really do about it ?**

Excuses, excuses, excuses …

- We have been thoroughly trained to make excuses
- We always downplay our failures
- It's always 'them' – How about 'us' ?

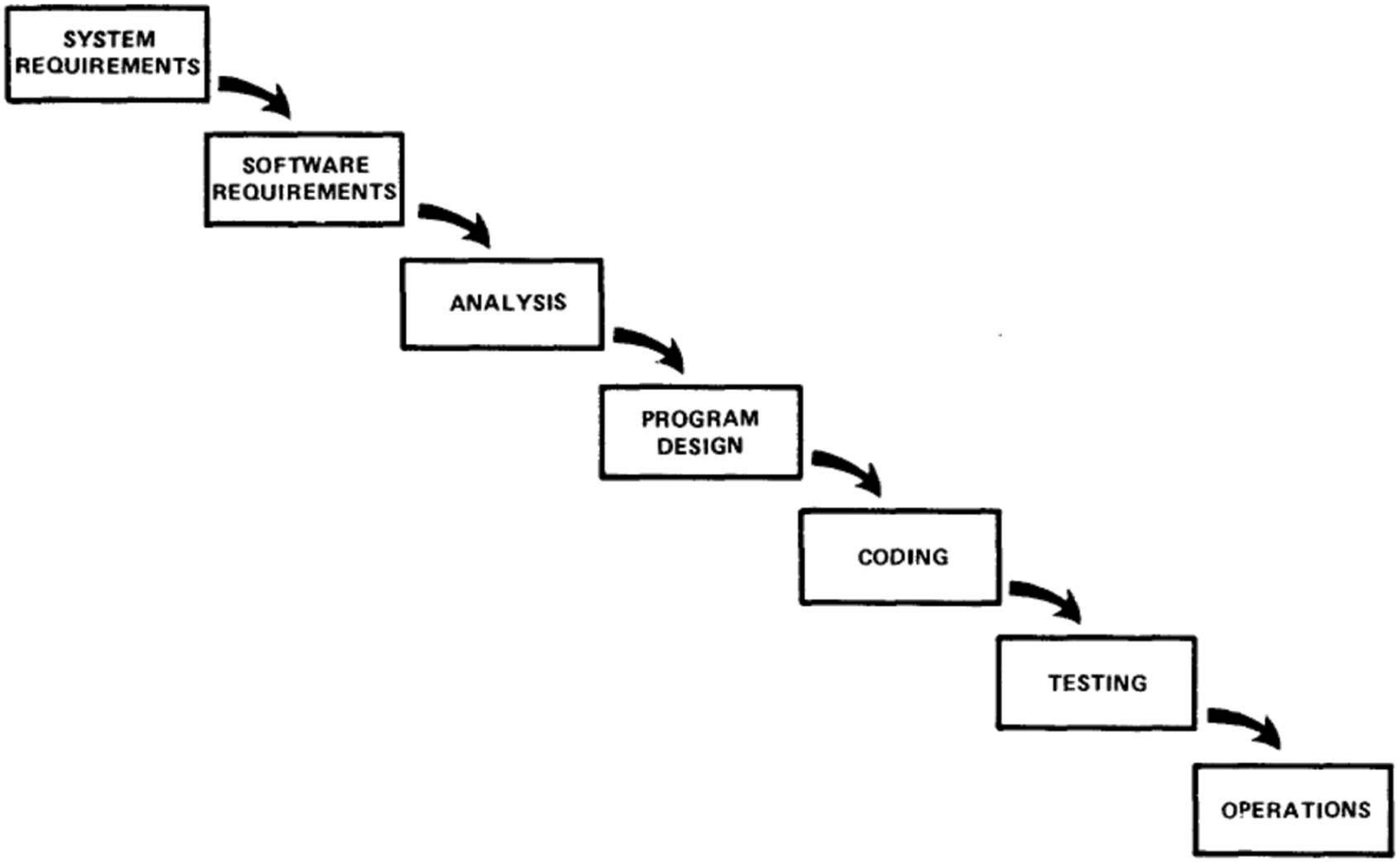
- At a Fatal Day, any excuse is in vain: *we failed*
- Even if we "really couldn't do anything about it"
- Failure is a very hard word. That's why we are using it !
- No pain, no gain
- We never say: "*You* failed" - Use: "*We* failed"
  - After all, *we* didn't help the person not to fail

Project Life Cycles

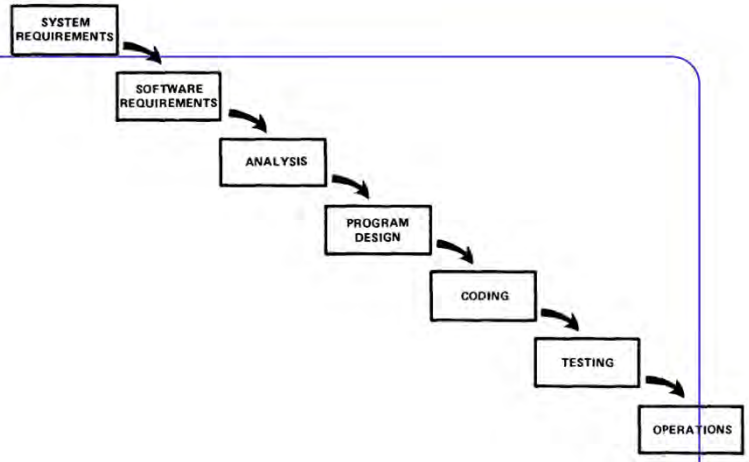
skip

Waterfall ?

Winston Royce 1970



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 ?

Problem - Solution

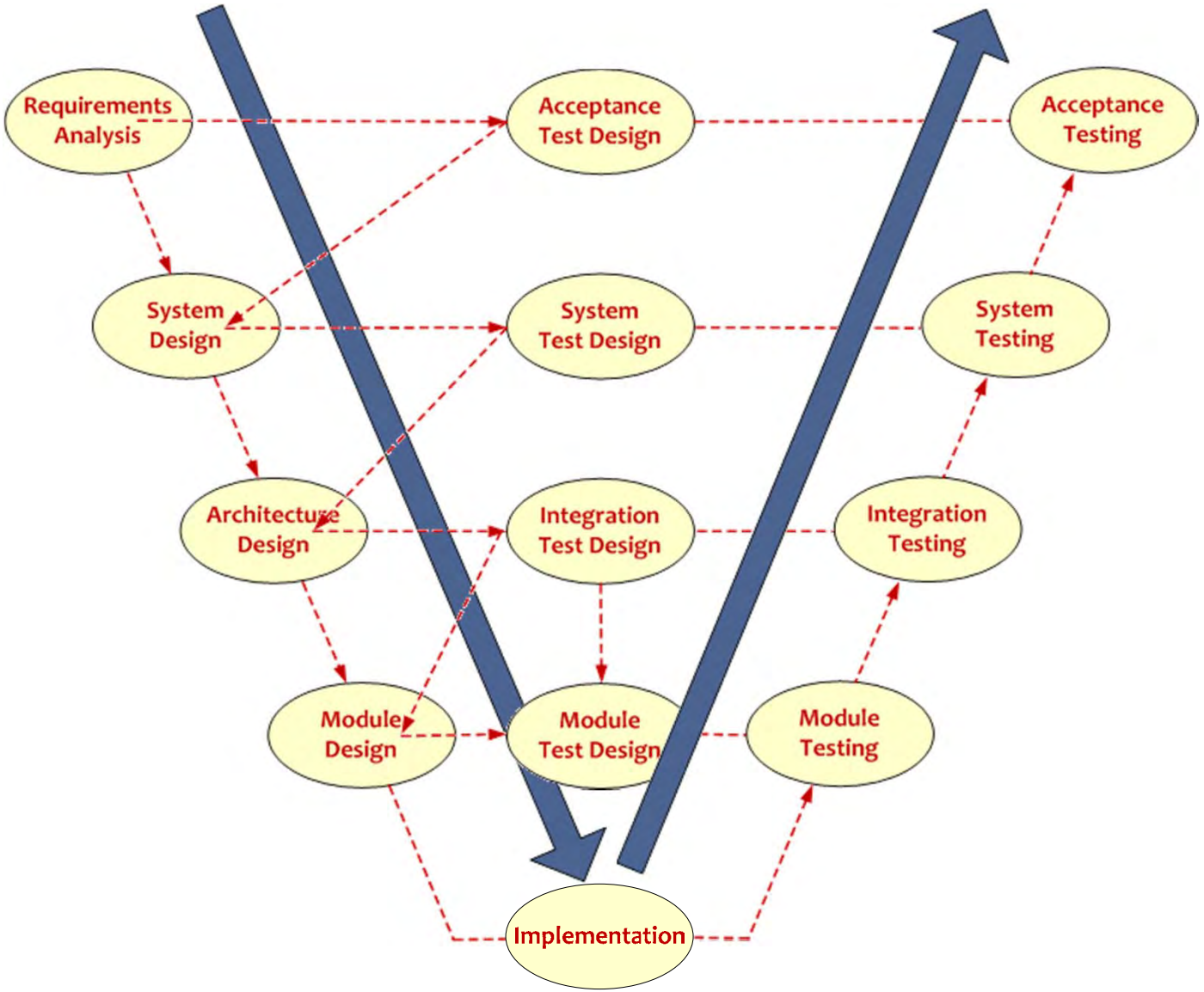
Problem known - Solution known = production

Problem known - Solution unknown = development

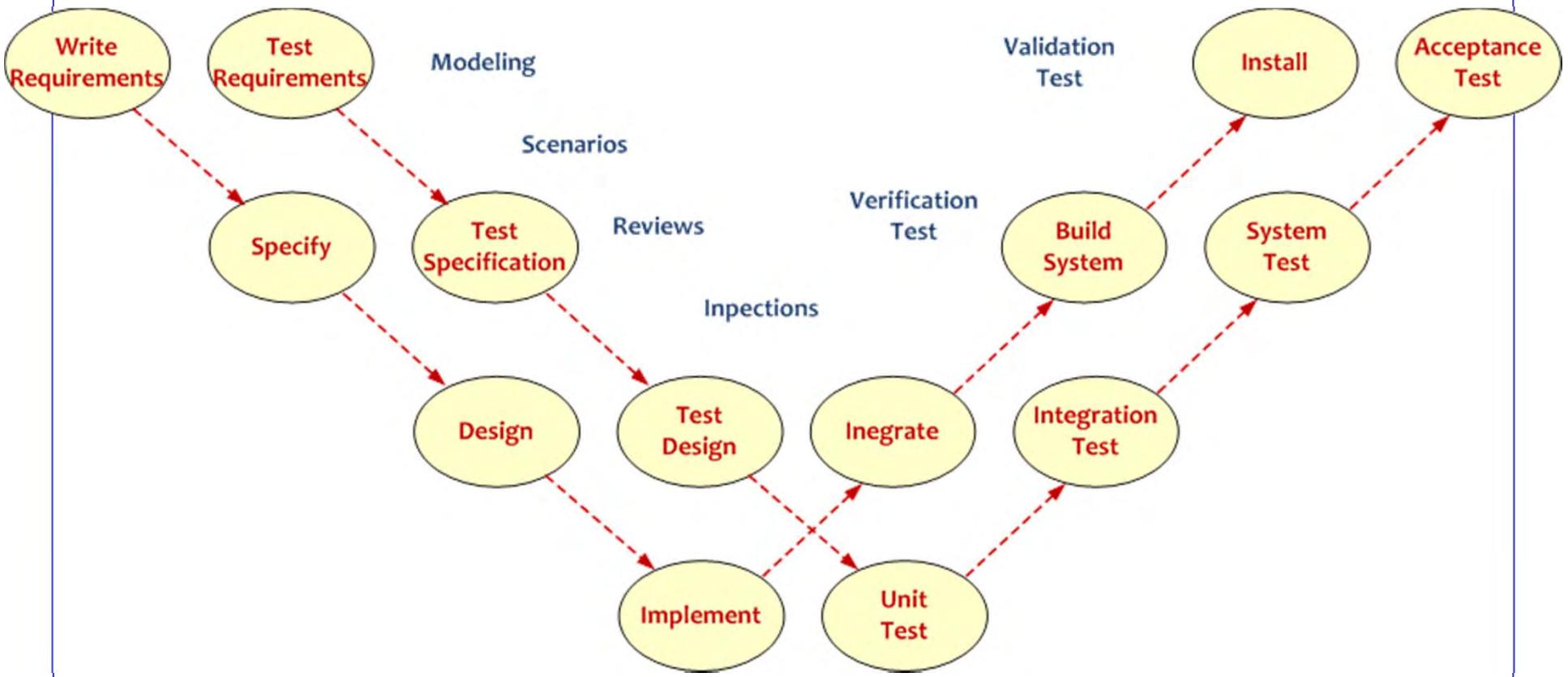
Problem unknown - Solution known = many IT projects

Problem unknown - Solution unknown = no problem ?

V-Model


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Module Test Design
Module Testing
Implementation

W-model


Write Requirements
Test Requirements
Modeling
Validation Test
Install
Acceptance Test
Scenarios
Specify
Test Specification
Reviews
Verification Test
Build System
System Test
Inpections
Design
Test Design
Inegrate
Integration Test
Implement
Unit Test

All Models are wrong

Some are useful

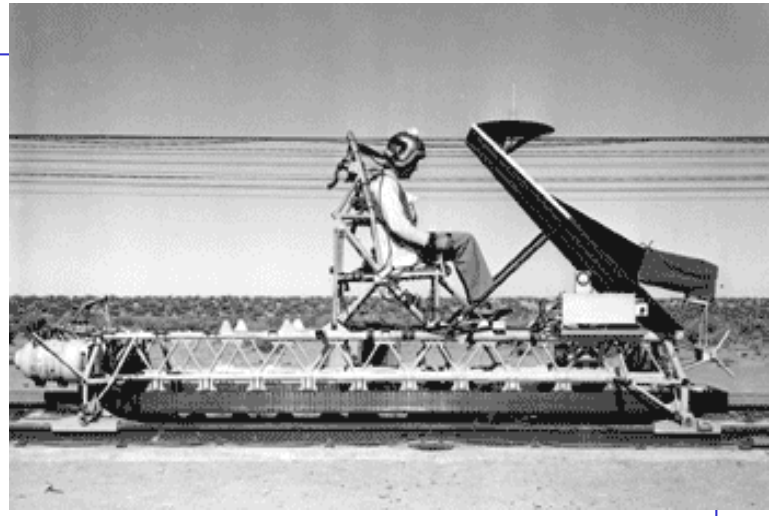
Evolutionary Principles

It's not a method

Just a bunch of add-ins to what you are already doing

Perhaps some alternatives …

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*

Do you use Retrospectives ?

Do we really learn from what happened ?

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

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*

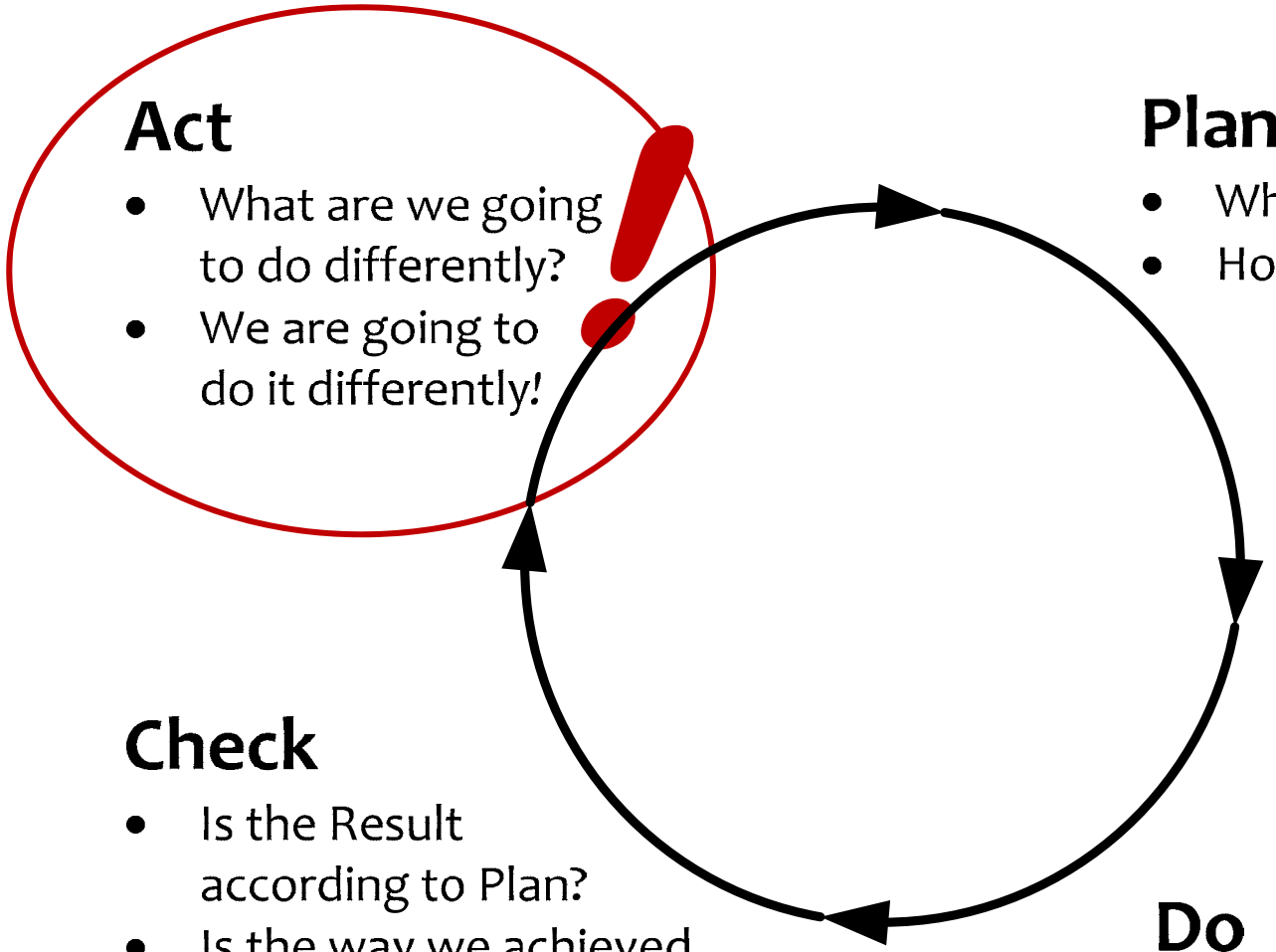
- **Hindsight is easy, but reactive**
- **Foresight is less easy, but proactive**
- **Reflection is for hindsight and learning**
- **Preflection is for foresight and prevention**

Only with *prevention* we can save precious time

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

The essential ingredient: the PDCA Cycle

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



Act

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

Plan

- What to achieve
- How to achieve it

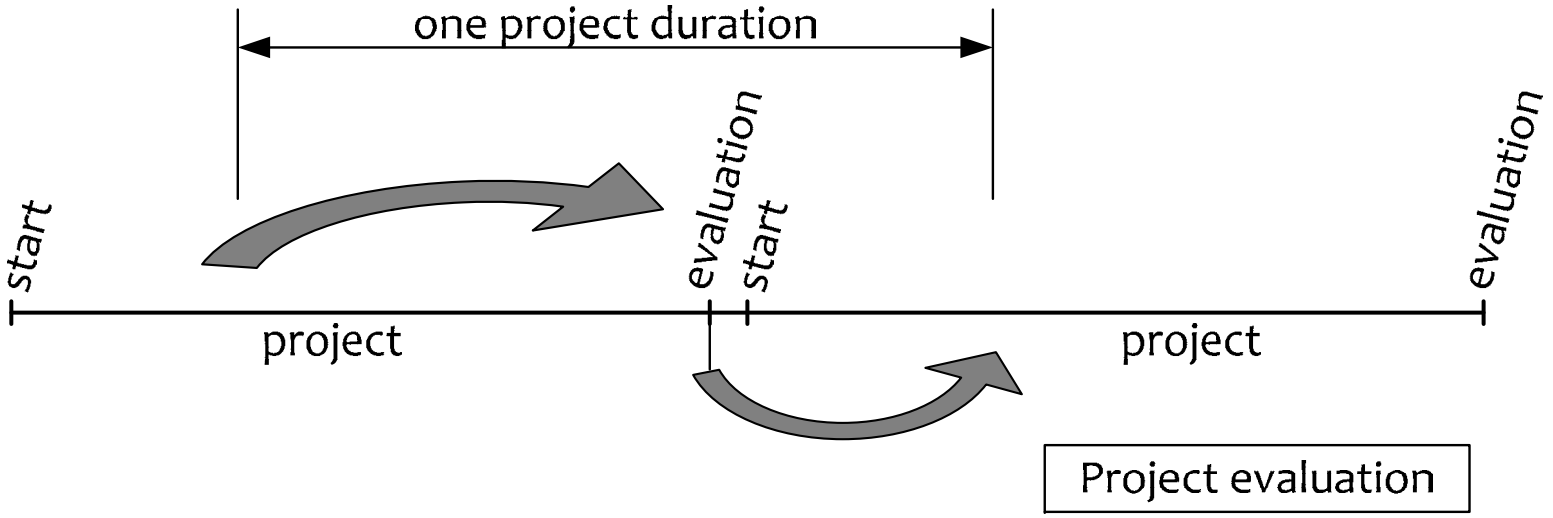
Check

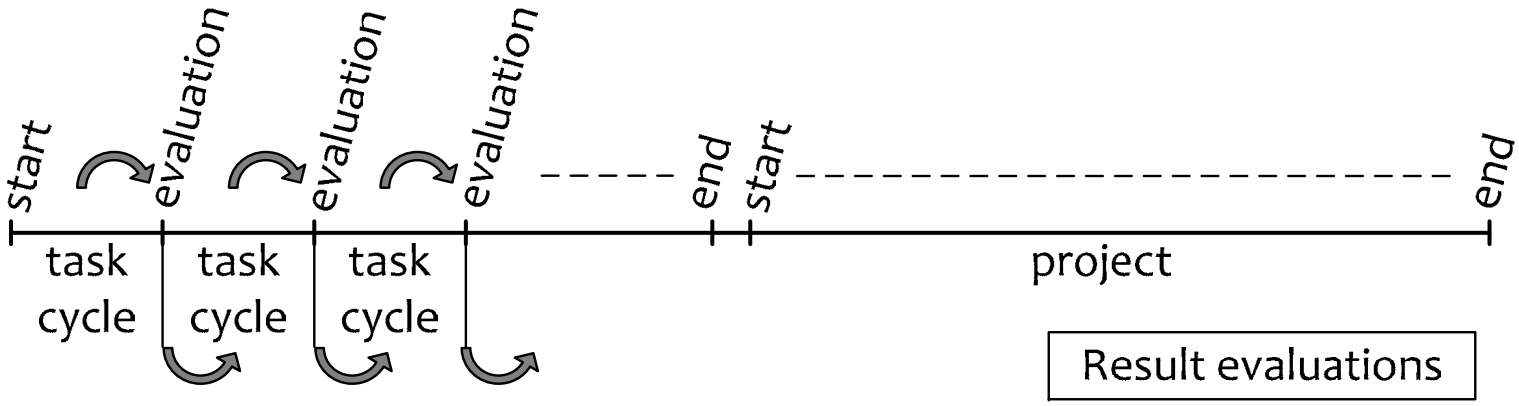
- Is the Result according to Plan?
- Is the way we achieved the Result according to Plan?

Do

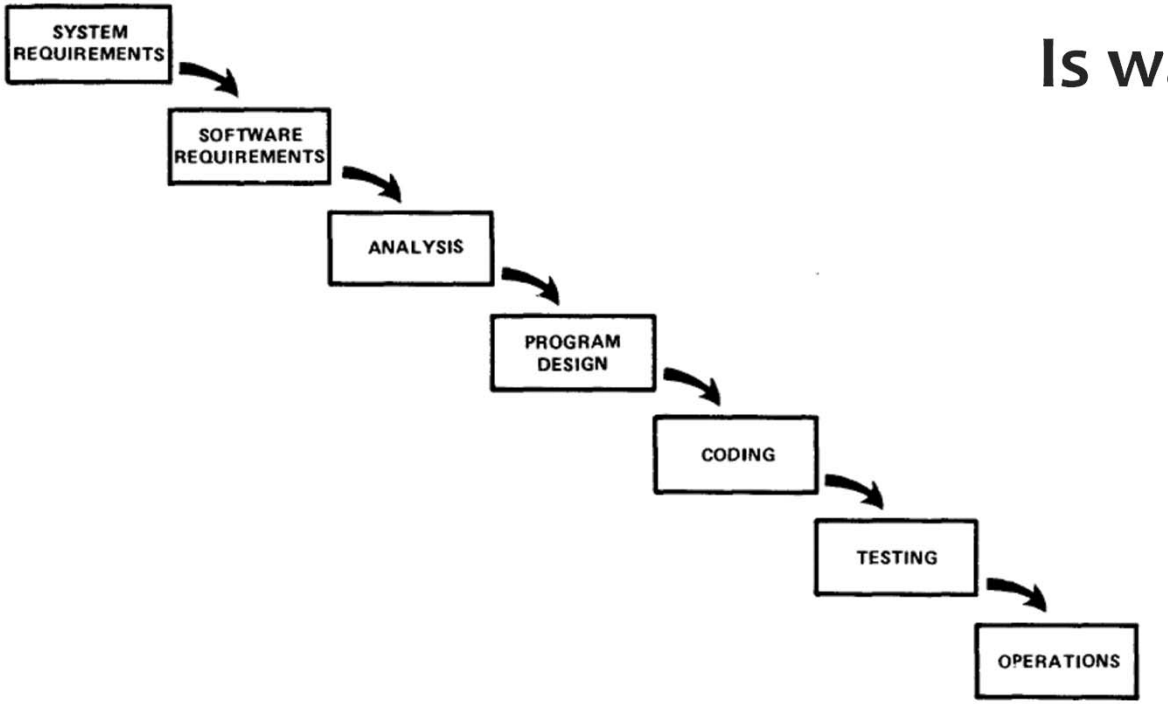
Carry out the Plan

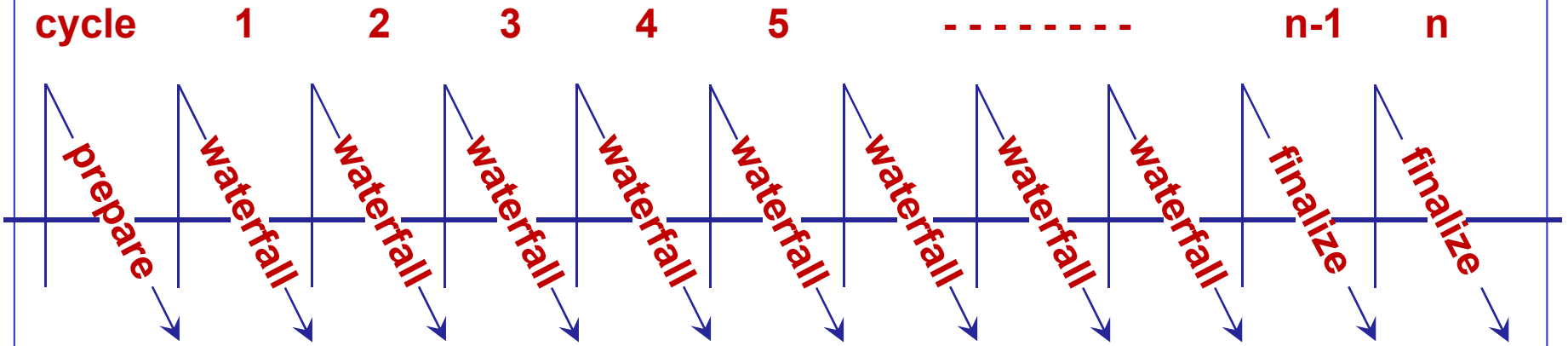
Project evaluations





Is waterfall wrong ?

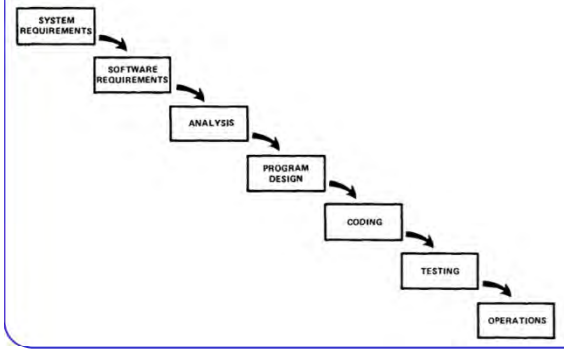


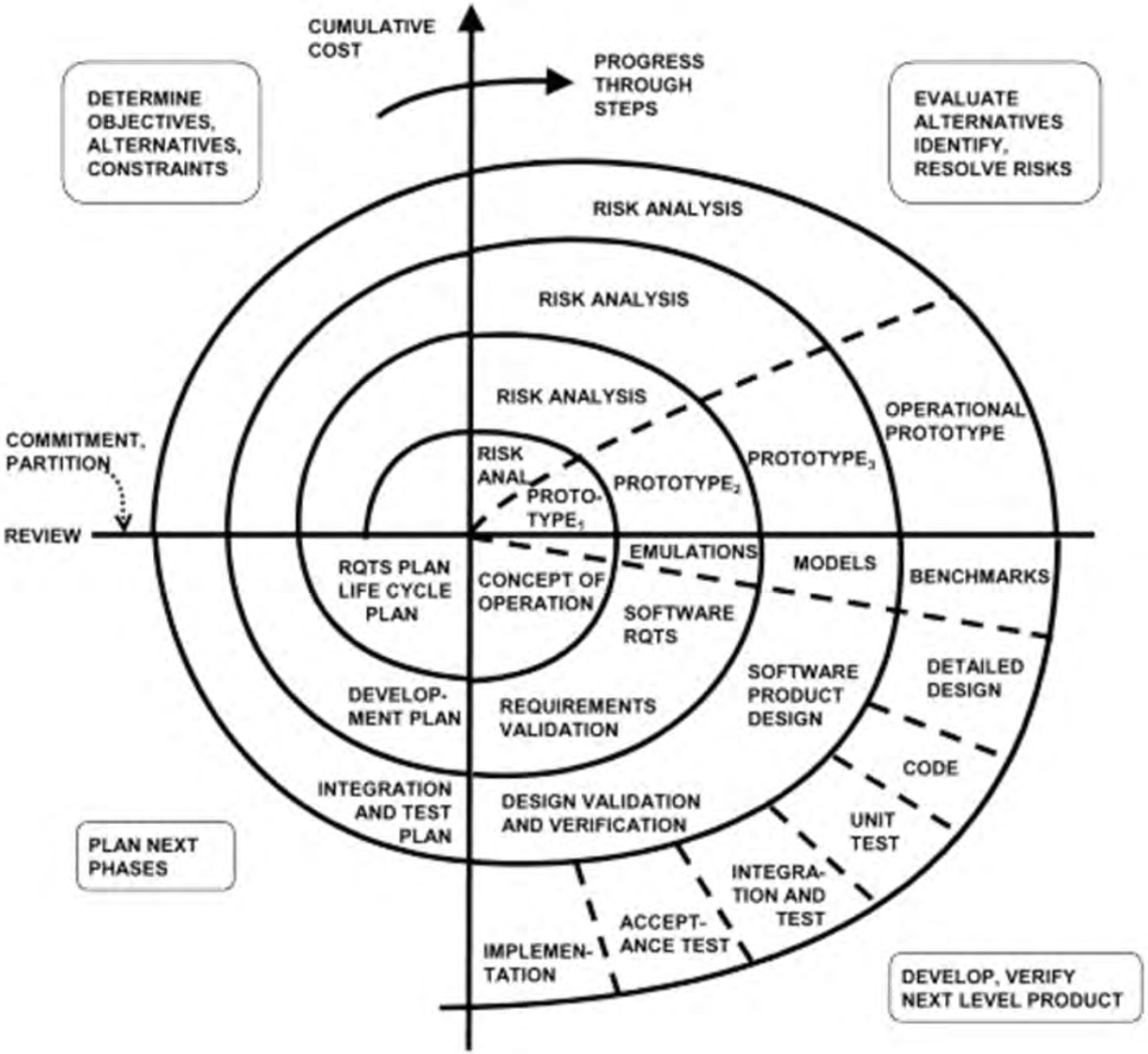


Spiral Process model

(Boehm 88)

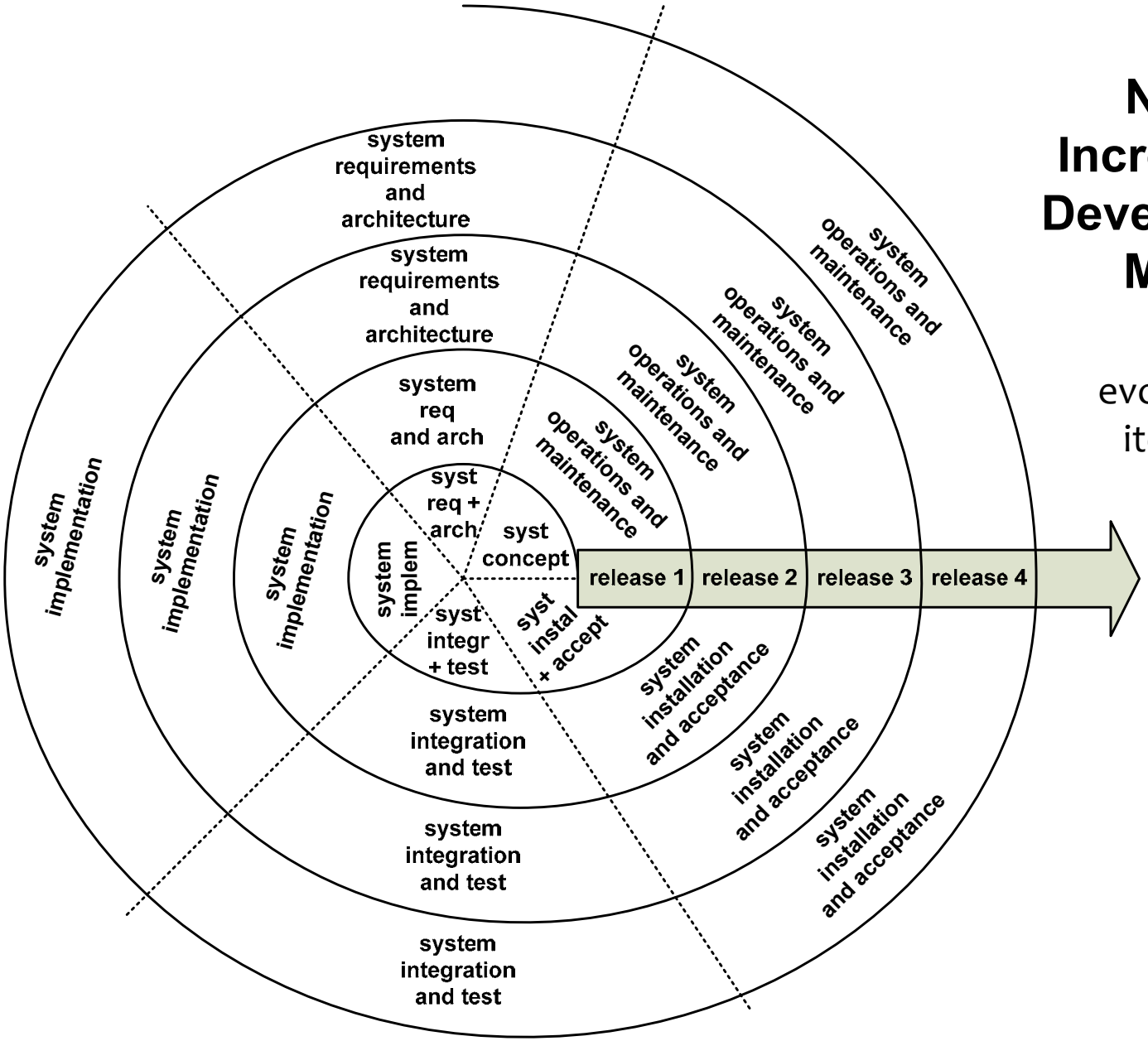
Actually Rolled up waterfall



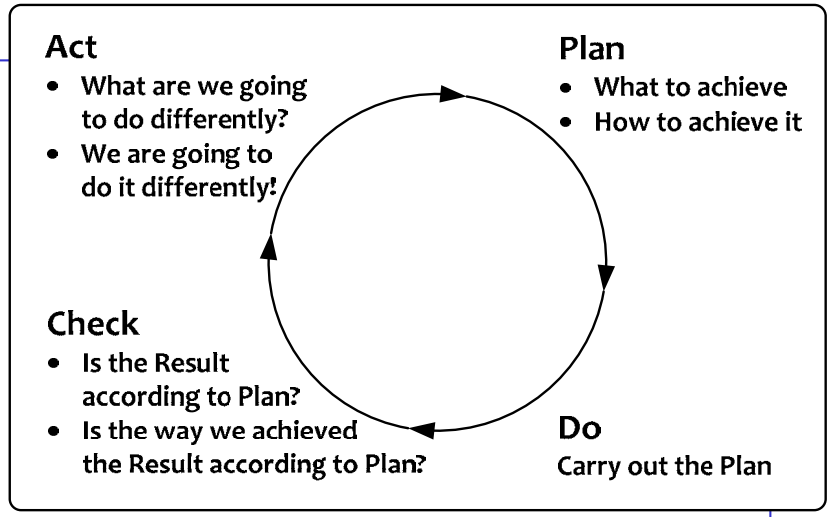


NASA Incremental Development Model

Real evolutionary iterations



Knowledge
how to achieve the goal



If we

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

doing the right things

then we can

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

doing the right things right

and we can

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

Known for decades

Do we still have to talk about this ?









- **Benjamin Franklin** (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- **Henry Ford** (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - Learning from waste, keeping things clean and safe, better treated people produce more
- **Toyoda's (Sakichi, Kiichiro, Eiji)** (1867-1930, 1894-1952, 1913-2013)
  - Jidoka: Zero-Defects, stop the production line (1926)
  - Just-in-time – flow – pull
- **W. Edwards Deming** (1900-1993)
  - Shewart cycle: Design-Produce-Sell-Study-Redesign (Japan – 1950)
  - Becoming totally focused on quality improvement (Japan – 1950) Management to take personal responsibility for *quality of* the product
  - Out of the Crisis (1986) - Reduce waste
- **Joseph M. Juran** (1904-2008)
  - Quality Control Handbook (1951, Japan – 1954)
  - Total Quality Management – TQM
  - Pareto Principe
- **Philip Crosby** (1926-2001)
  - Quality is Free (1980)
    - Zero-defects (1961)
- **Taiichi Ohno** (1912-1990)
  - (Implemented the) Toyota Production System (Beyond Lange-Scale Production) (1988)
  - Absolute elimination of waste - Optimizing the TimeLine from order to cash
- **Masaaki Imai** (1930-)
  - Kaizen: The Key to Japan's Competitive Success (1986)
  - Gemba Kaizen: A Commonsense, Low-Cost Approach to Management (1997)

Eliminating Waste
Not doing what doesn't yield value

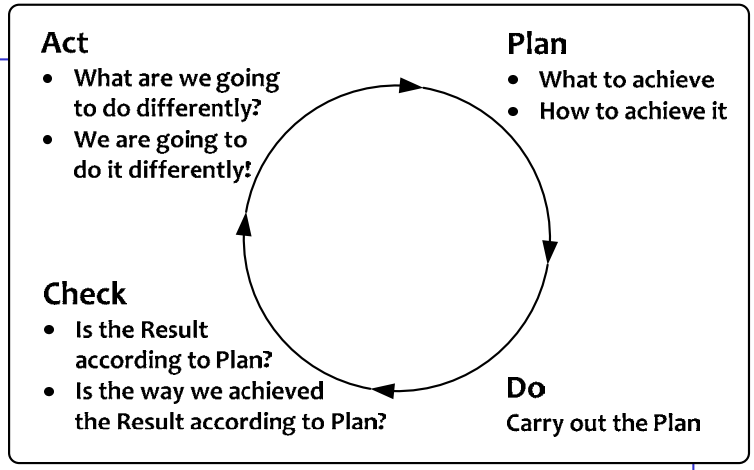
Cobb's Paradox

Martin Cobb - 1989
Treasury Board of Canada Secretariat
Ottawa, Canada

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

- How about your project ?
  Did you deliver the right result at the right time ?

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

Evolutionary Project Management (Evo)

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

Why

What
How much
Are we done

How

Check as early as possible

Right Result

Evo Project Planning

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

Efficiency
of what we do

Right Time

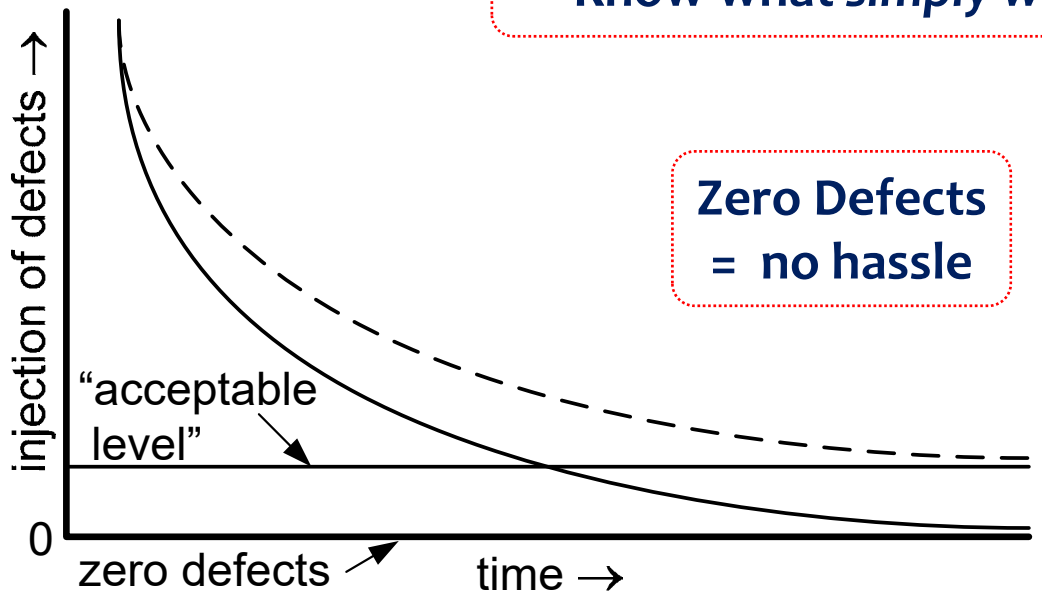
Effectiveness
of what we do

What will happen
and what will we
do about it?

What is Zero Defects

- We aren't perfect,
  but the customer shouldn't find out
- What we deliver *simply works*
- Know what *simply works* means !

- Zero Defects is an *asymptote*



Zero Defects
= no hassle

- When Philip Crosby started with Zero Defects in 1961, errors dropped by 40% almost immediately
- AQL > Zero means that the organization has settled on a level of incompetence
- Causing a hassle other people have to live with

Business Case

Business Case

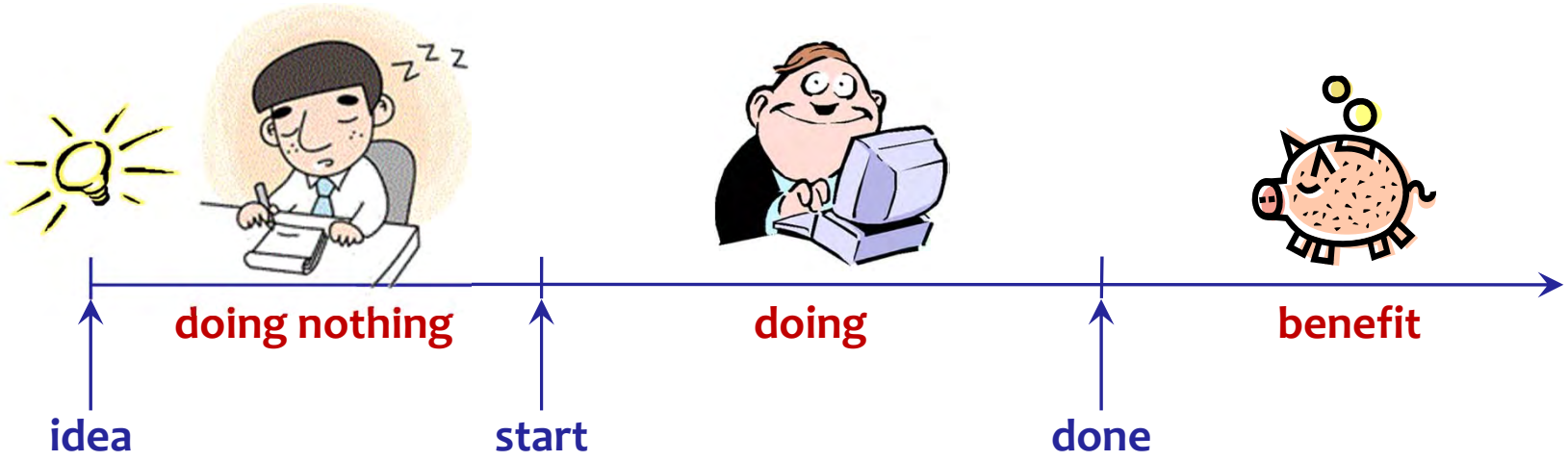
- *Why* are we running a project ?
- Why to improve
- Drives the decision making processes
- To continually align the Projects progress to the dynamic business objectives
- Stakeholders
- Total LifeCycle - cradle to cradle

Higher Productivity

- **All functionality we produce *does already exist***
- **The real reason for running our projects is creating *better performance***

- **Types of improvement:**
  - **Less loss**
  - **More profit**
  - **Doing the same in shorter time**
  - **Doing more in the same time**
  - **Being happier than before**
  - **Travel easier**
- **In short: *Adding Value***

Return on Investment



Return on Investment (RoI)

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

How many Business Cases ?

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

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

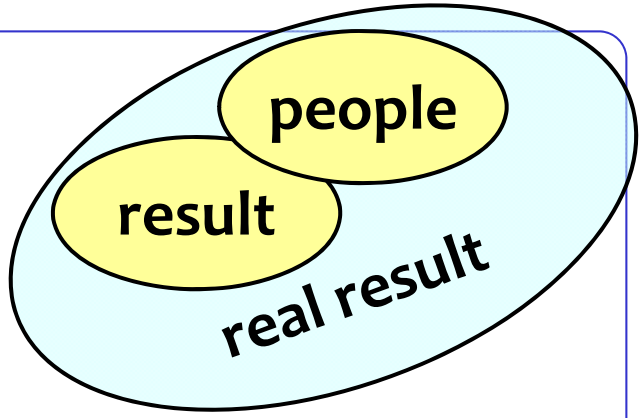
- Actually, every Stakeholder has his own Business Case

Stakeholders
&
Requirements

Did you prepare ?

- **The top-3 stakeholders of your work** (*Who* is waiting for it?)
- **The top-3 real requirements for your work** (What are they waiting for?)
- **How much value improvement the stakeholders expect** (3 or 7?)
- **Any deadlines** (No deadlines: it will take longer)
- What you should and can have achieved in the *coming 10 weeks*
  (Will you succeed? - Failure is not an option!)
- What you think you should and can do the *coming week* in order to achieve what you're supposed to achieve (Make sure not to plan what you shouldn't or cannot do - At the end of the week everything you planned will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work

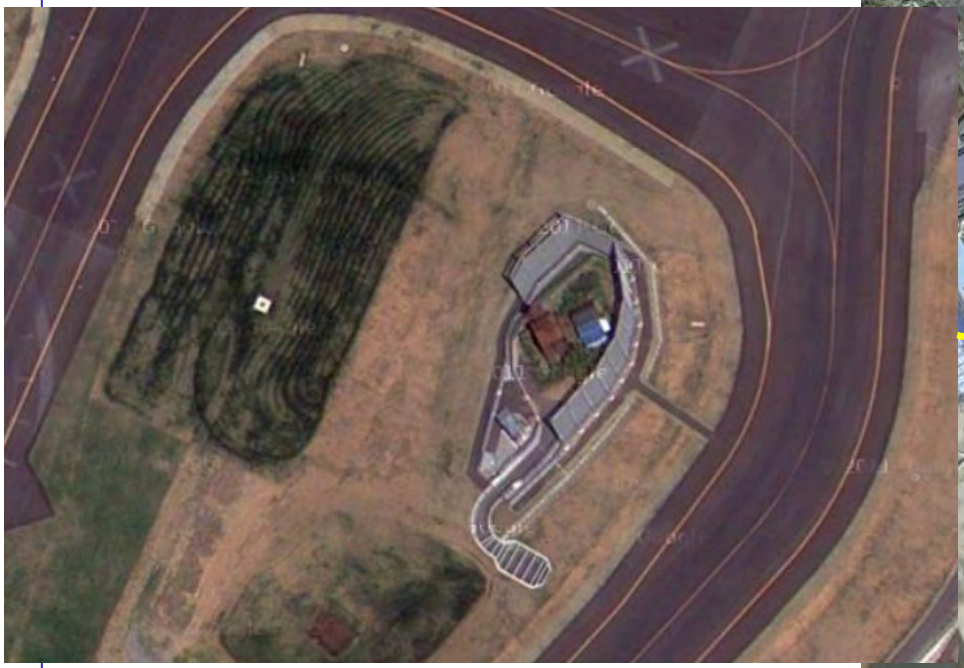
Stakeholders are (not only) people



- 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 !*
- 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 !

Victims can be a big Risk

Victims:
Narita Airport



What are the Requirements for a Project ?

- Requirements are what the Stakeholders require

but for a project ...

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

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

No Stakeholder?

- No Stakeholder: no requirements
- No requirements: nothing to do
- No requirements: nothing to test
- If you find a requirement without a Stakeholder:
  - Either the requirement isn't a requirement
  - Or, you haven't determined the Stakeholder yet
- If you don't know the Stakeholder:
  - Who's going to pay you for your work?
  - How do you know that you are doing the right thing?
  - When are you ready?

Top level Requirement for any 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

Wish Specification

Nice Input

Wish Specification

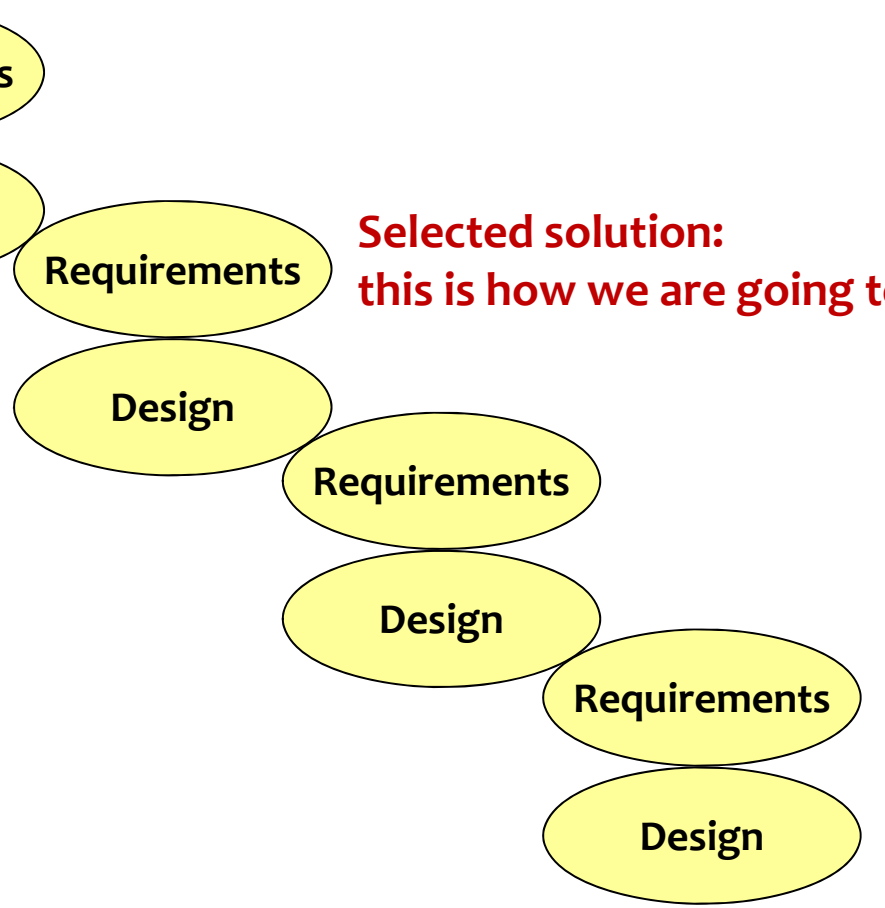
- **What Wish Specification did *you* receive ?**
  - **Write it down**
- **How did you receive it ?**
- **From whom ?**
- **What did you do with it ?**

- **Was it complete ?**
- **Was it clear ?**
- **Did it show the problem to be solved ?** (or was it a *solution* ?)

No Design in the requirements, but ...

Needs:
what do we need

Options:
how can we do it



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

Design provides the
Requirements for the next level

Requirements have Rules

Some examples:

Rule 1: All quality requirements must be expressed *quantitatively*

Rule 2: No design (solutions) in the requirements

Rule 3: Unambiguous

Rule 4: Clear to test

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

Requirements with Planguage

ref Tom Gilb

Definition:

Specific

Measurable

RQ27: Speed of Luggage Handling at Airport

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

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

Benchmarks (Playing Field):

Attainable

Past: 2 min [minimum, 2014], 8 min [average, 2014], 83 min [max, 2014]

Current: < 4 min [competitor y, Jan 2015] ← <who said this?>, <Survey Dec 2014>

Record: 57 sec [competitor x, Jan 2012]

Wish: < 2 min [2017Q3, new system available] ← CEO, 19 Jan 2015, <document ...>

Requirements:

Time

Realizable

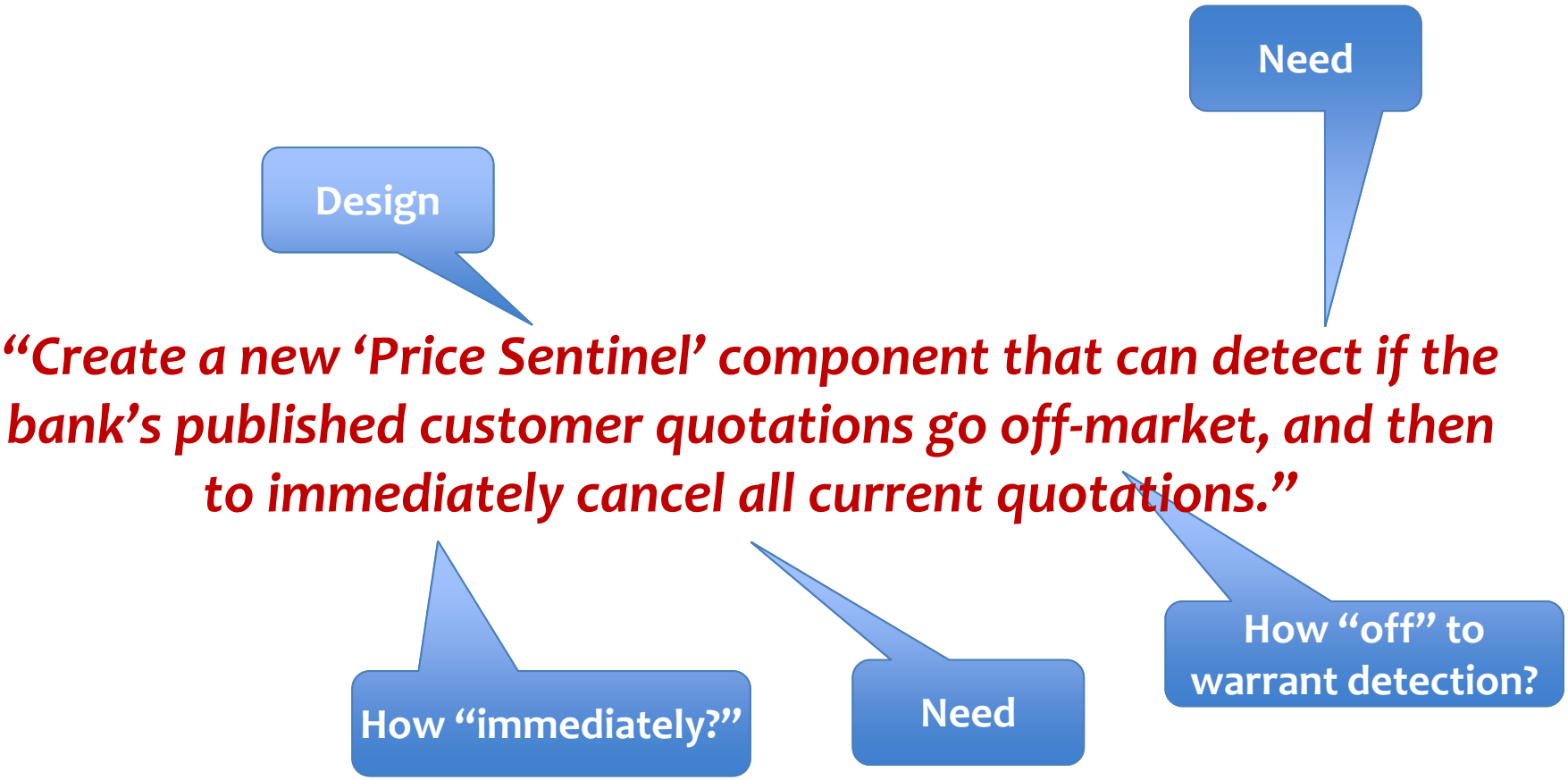
Tolerable: < 10 min [99%, Q4] ← SLA

Tolerable: < 15 min [100%, Q4, Heathrow T4] ← SLA

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

Is this a Requirement ?

or 'nice input', to be taken seriously ?



Ref http://rsbatechnology.co.uk

Using 5 Whys

Why do you need a "Price Sentinel" ?

1. To prevent publishing off-market tradable prices
2. To prevent trading loss
   (having to buy at a higher price than the bank offered to the customer)
3. To demonstrate to senior management that e-trading business can safely (no unexpected loss) manage customer trading
4. To ensure that senior management will agree to expand e-trading business in the future, based on current business performance to other customer segments and business areas
5. To meet business medium / long-term financial targets

Ref http://rsbatechnology.co.uk

First try

New 'Price Sentinel' component:

- **detect if the bank's customer quotations go off-market**
- **then immediately cancel all current quotations**

- **Off-market**
  - **?? – Our margin less than 0.1% ?? – Will have to investigate**
- **Cancelling all current quotations**
  - **Scale: seconds after <detection>**
  - **Current: 600 sec (10 min)**
  - **Goal: 1 sec**

Ref http://rsbatechnology.co.uk

Prioritize solutions by Impact Estimation

	Kill button	Price Sentinel
Cancel 600 → 1 sec	10.5 sec (note) 98%	1 sec 100%
Cost	1 day	30 day (6 sprint)
Note: 10 sec human recognition time, 0.5 sec cancel time		

Ref http://rsbatechnology.co.uk

Tom Gilb quote

- The fact that we can set numeric objectives, and track them, is powerful; *but in fact it is not the main point*
- The main purpose of quantification is to force us to *think deeply*, and *debate exactly*, what we mean
- So that others, later, *cannot fail* to understand us

Examples of Scales

(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>

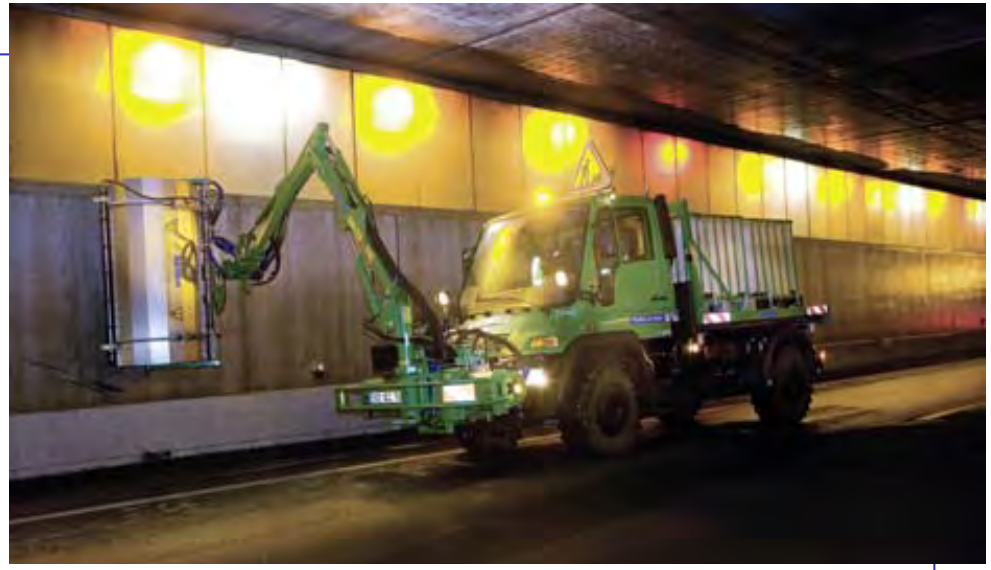
Reliability

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}

Availability



- **Dependability.Availability**
  - Readiness for service
  - Scale: % of <TimePeriod> a <System> is <Available> for its <Tasks>
- **Probability that the system will be functioning correctly when it is needed**
- **Examples**
  - (preventive) maintenance may decrease the availability
  - Snow on the road
  - Telephone exchange (no dial tone) < 5 min per year (99.999%)

Availability

Availability %	Downtime per year	Downtime per month	Downtime per week	Typical usage
90%	36.5 day	72 hr	16.8 hr	
95%	18.25 day	36 hr	8.4 hr	
98%	7.30 day	14.4 hr	3.36 hr	
99%	3.65 day	7.20 hr	1.68 hr	
99.5%	1.83 day	3.60 hr	50.4 min	
99.8%	17.52 hr	86.23 min	20.16 min	
99.9% (three nines)	8.76 hr	43.2 min	10.1 min	Web server
99.95%	4.38 hr	21.56 min	5.04 min	
99.99% (four nines)	52.6 min	4.32 min	1.01 min	Web shop
99.999% (five nines)	5.26 min	25.9 sec	6.05 sec	Phone network
99.9999% (six nines)	31.5 sec	2.59 sec	0.605 sec	Future network

Quantified Requirements

found on Internet

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

How about your requirements ?

- **Expressed quantitatively**
- **No design (solutions)**
- **Unambiguous**
- **Clear to test**

Requirements exercise: (groups of 2 or 3 people)

Specify a quality / performance requirement for your current, previous or future project, using Planguage

Try to use:

Definition:
- Ambition
- Scale
- Meter
- Stakeholders

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

Requirements:
- Must/Fail/Tolerable
- Goal

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

Ambition	
Scale	
Meter	
Stakehldrs	

Past	
Current	
Record	
Wish	

Tolerable	
Goal	

Evolutionary Planning

Producing even more in less time

Did you prepare ?

- The top-3 stakeholders of your work (*Who* is waiting for it?)
- The top-3 real requirements for your work (What are they waiting for?)
- How much value improvement the stakeholders expect (3 or 7?)
- Any deadlines (No deadlines: it will take longer)
- What you should and can have achieved in the *coming 10 weeks*
  (Will you succeed? - Failure is not an option!)
- **What you think you should and can do the *coming week* in order to achieve what you're supposed to achieve** (Make sure not to plan what you shouldn't or cannot do - At the end of the week everything you planned will be done)
- **What value you will have delivered by the end of the week and how to prove it**
- Any issues you expect with the above or otherwise with your work

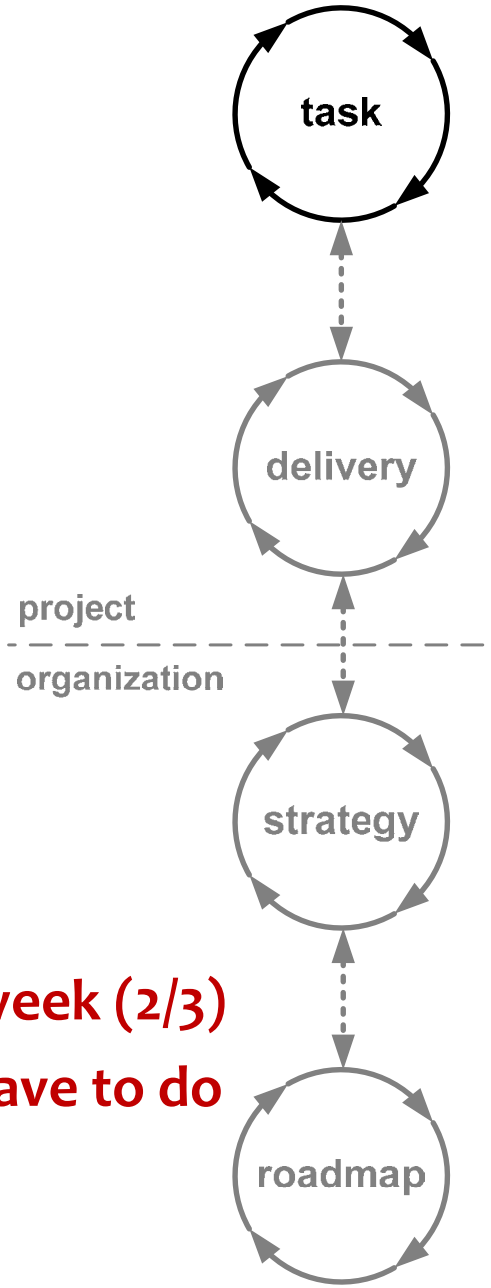
To-do lists

- **Are you using to-do lists ?** **→ EXERCISE**
  - List the things you have to do the coming week
  - Did you add effort estimates?
  - Did you check how much time you have available the coming week ?
  - 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

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

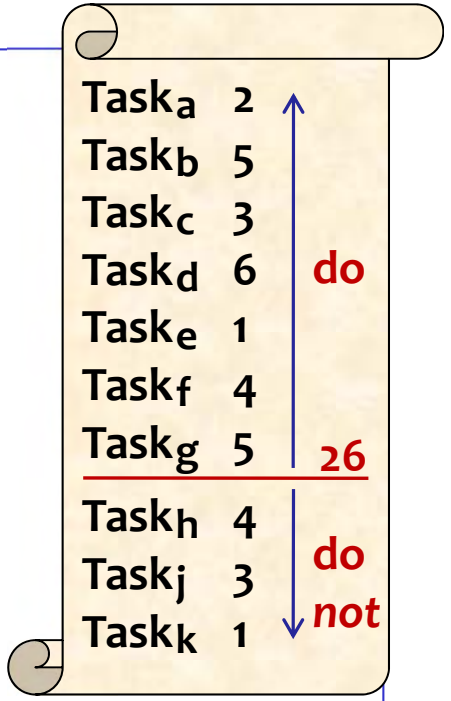


Effort and Lead Time

- 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)

Every week we plan



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

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

Weekly 3-Step Procedure

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

cycle	who	task description	estim	real	done	issues
3	John	*Net time available: 26*				
		aaaaaaaaa	3	3	yes	
		bbbbbbbb [Paul]	1			
		ccccccccc	5	13	yes	
		ddddddddd	2			
		eeeeeeee	3	2		
		fffffffffff	2	1		
		ggggggggg	6	7	yes	
		hhhhhhhh	4			
			26	26		
4	John	*Net time available: 26*				
		jjjjjjjjjjjjj	3			for proj x
		kkkkkkkkk	1			for proj x
		mmmmm	5			for proj x
		nnnnnnnn	2			for proj x
		ppppppppp	3			for proj y
		qqqqqqqq	12			for proj y
		rrrrrrrrrrrr	6			for proj y
		sssssssss	4			for proj y
		ttttttttttt	4			for proj y
			40			

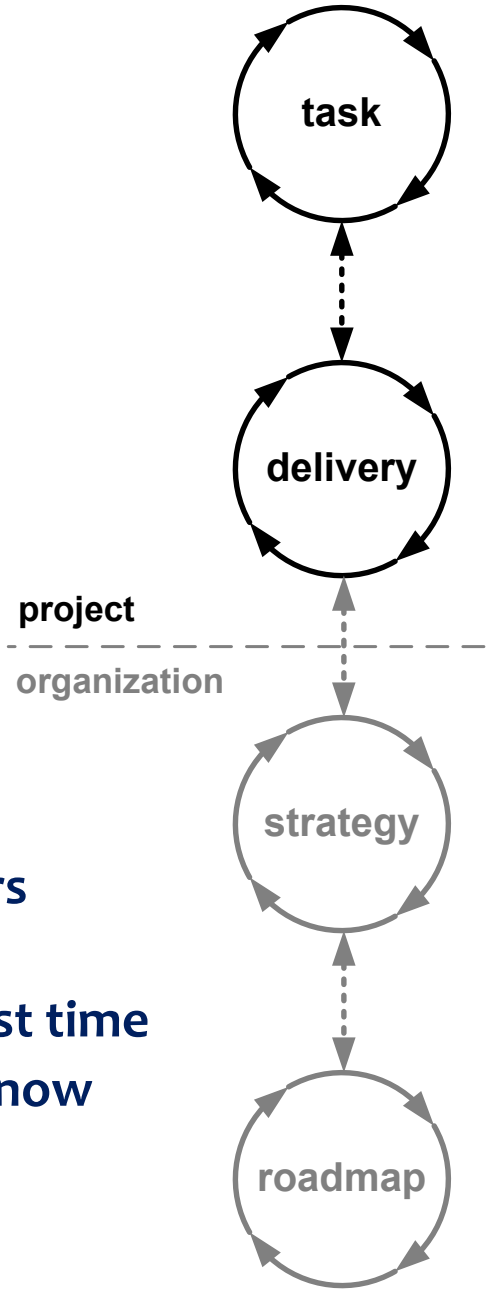
TaskCycle Analysis (retrospective)

learning

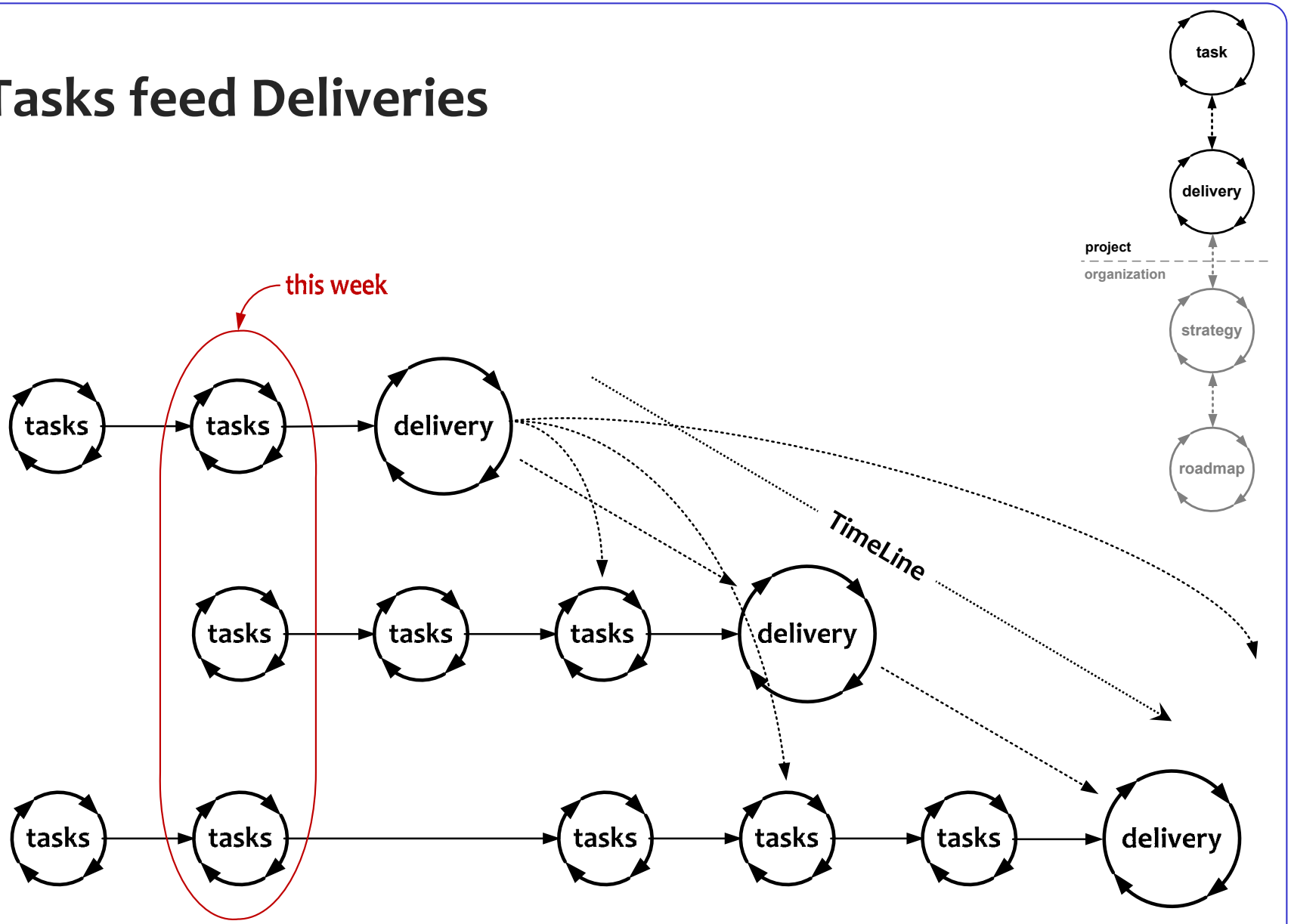
TaskCycle Planning (presepective)

DeliveryCycle

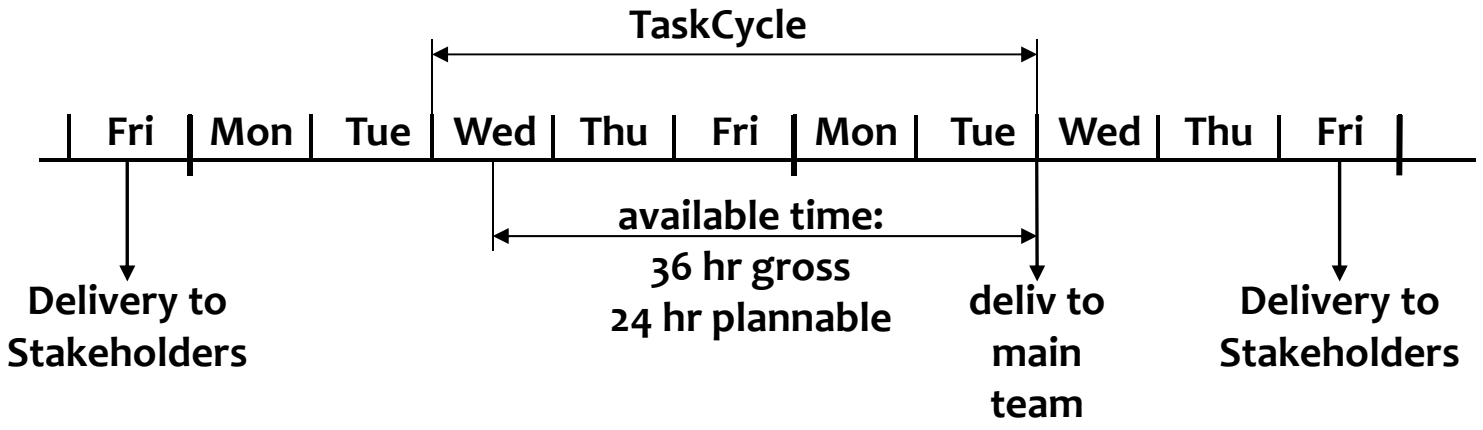
- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
    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



Tasks feed Deliveries



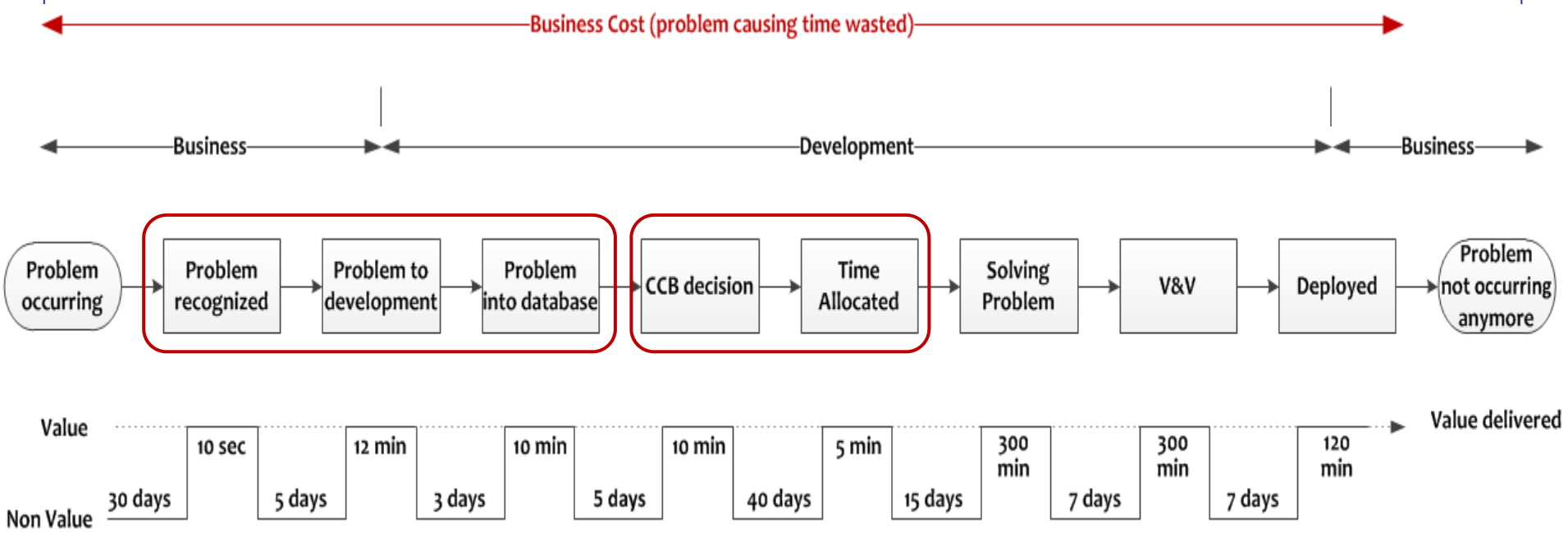
Designing a Delivery



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

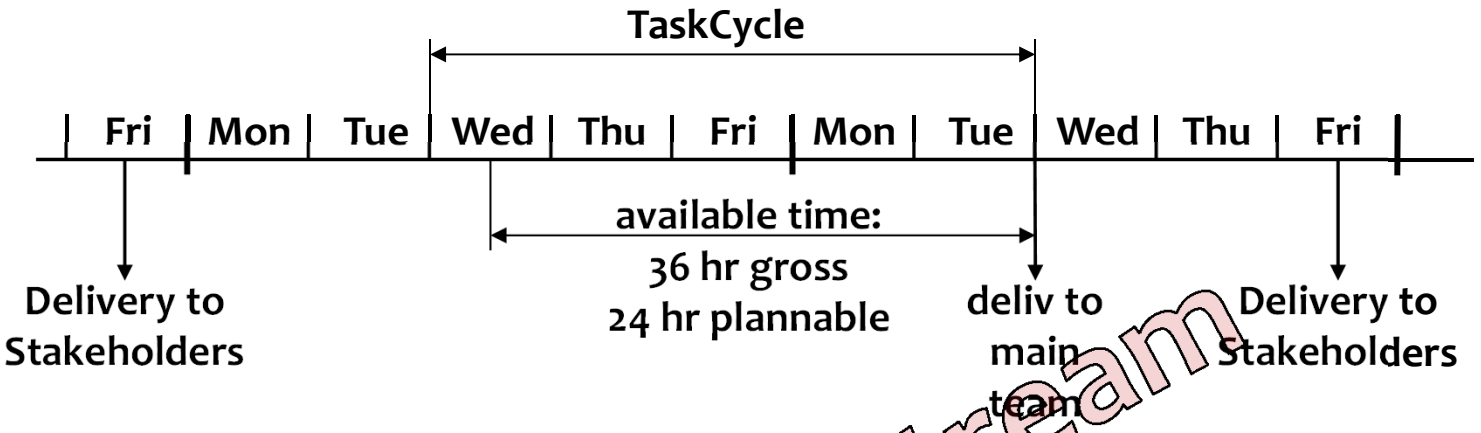
Gregory		Gregory (later)	
Draft design	6 →	Draft design	0
Finish design	6 →	Finish design	0
Work for deliv	3	...	
-	1		
-	2		
-	2		
-	3		
-	5		
-	6		
XMLa	4 →	XMLa	3
XMLb	4 →	XMLb	3
Total	42	...	

Value stream mapping



- **Total Business Cost 114 days, Cost of Non Value: 112 days**
- **Occurrence: 2 x per day, delay per occurrence: 10 min**
- **Number of business people affected: 100**
- **Business Cost of Non Value:** 2 x 10 min x 112 days x 100 people x 400 €/day = **187 k€**
- **Net Cost of Value:** 1.6 days → ~3 people x 1.6 days x 800 €/day = **5 k€**

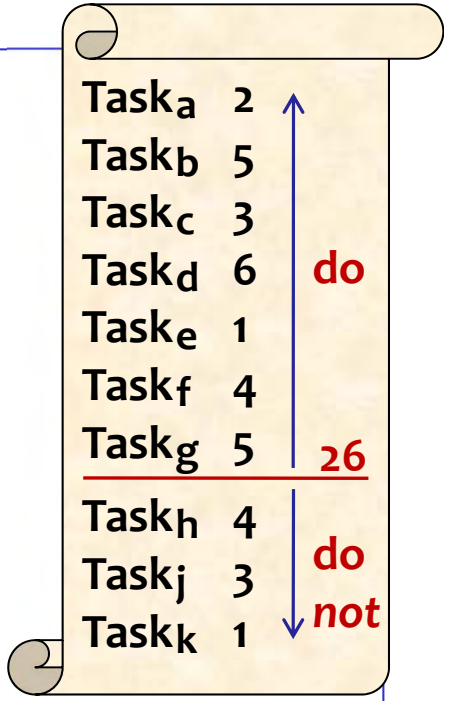
Designing a Delivery



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

Gregory		Gregory (later)	
Draft design	0 →	Draft design	0
Finish design	0 →	Finish design	0
Work for deliv	3	...	
-	1		
-	2		
-	2		
-	3		
-	5		
-	6		
XMLa	1 →	Jerome	
XMLb	1 →	XMLa	3
Total	24	XMLb	3
		...	

TaskCycle Exercise

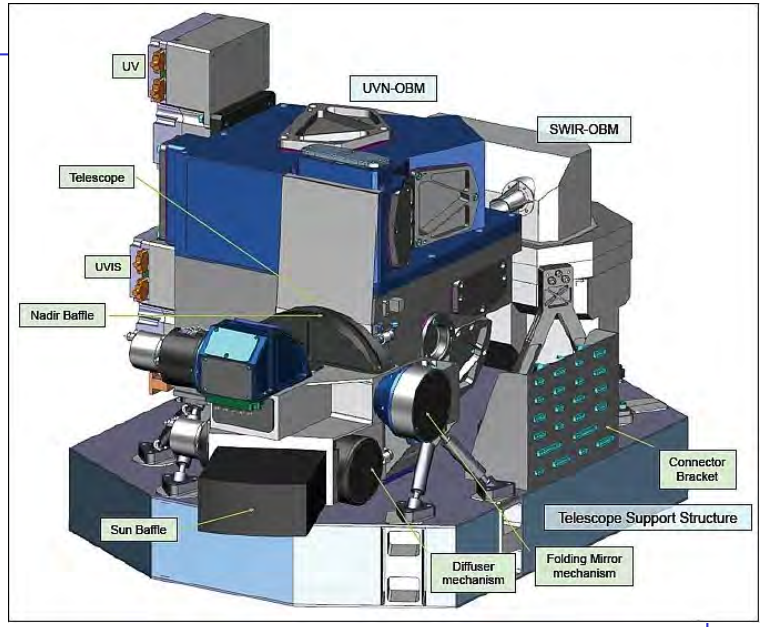


- How much time do you have available
- 2/3 of available time is net plannable time
- What is most important to do (make list)
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr)
- What can you do, and what are you going to do
- What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

Why is this important ?

- TaskCycle Planning is *not* just planning the work for the coming week
- It exposes issues immediately
- Half of what people do in projects later proves not to have been necessary
- During the TaskCycle planning we can very efficiently see
  - What our colleagues think they're going to do
  - Make sure they're going to work on the most important things
  - Not on unnecessary things
  - In line with the architecture and design
  - Leading most efficiently to the goal of the delivery
  - Everyone knows exactly *what*'s going to happen, *what not*, and *why*

Earth Observation Satellite



- Very experienced Systems Engineers
- They use quantified requirements routinely
- They don't know exactly where they'll end up
- 10 year pure waterfall project (imposed by ESA)
- Only problem: They missed all deadlines
- 9 weeks later: They haven't missed any deadline since
- Recently: delivered 1 day early (instead of 1 year late)
- Savings: some 40 man-year
- How did they do that ?

Requirements weren't the problem

- Requirements for tropospheric O3
  - Ground-pixel size : 20 × 20 km2 (threshold); 5 × 5 km2 (target)
  - Uncertainty in column : altitude-dependent
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)
- Requirements for stratospheric O3
  - Ground-pixel size : 40 × 40 km2 (threshold); 20 × 20 km2 (target)
  - Uncertainty in column : altitude-dependent
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)
- Requirements for total O3
  - Ground-pixel size : 10 × 10 km2 (threshold); 5 × 5 km2 (target)
  - Uncertainty in column : 2%
  - Coverage : global
  - Frequency of observation :
    daily (threshold); multiple observations per day (target)

Awful schedule pressure !

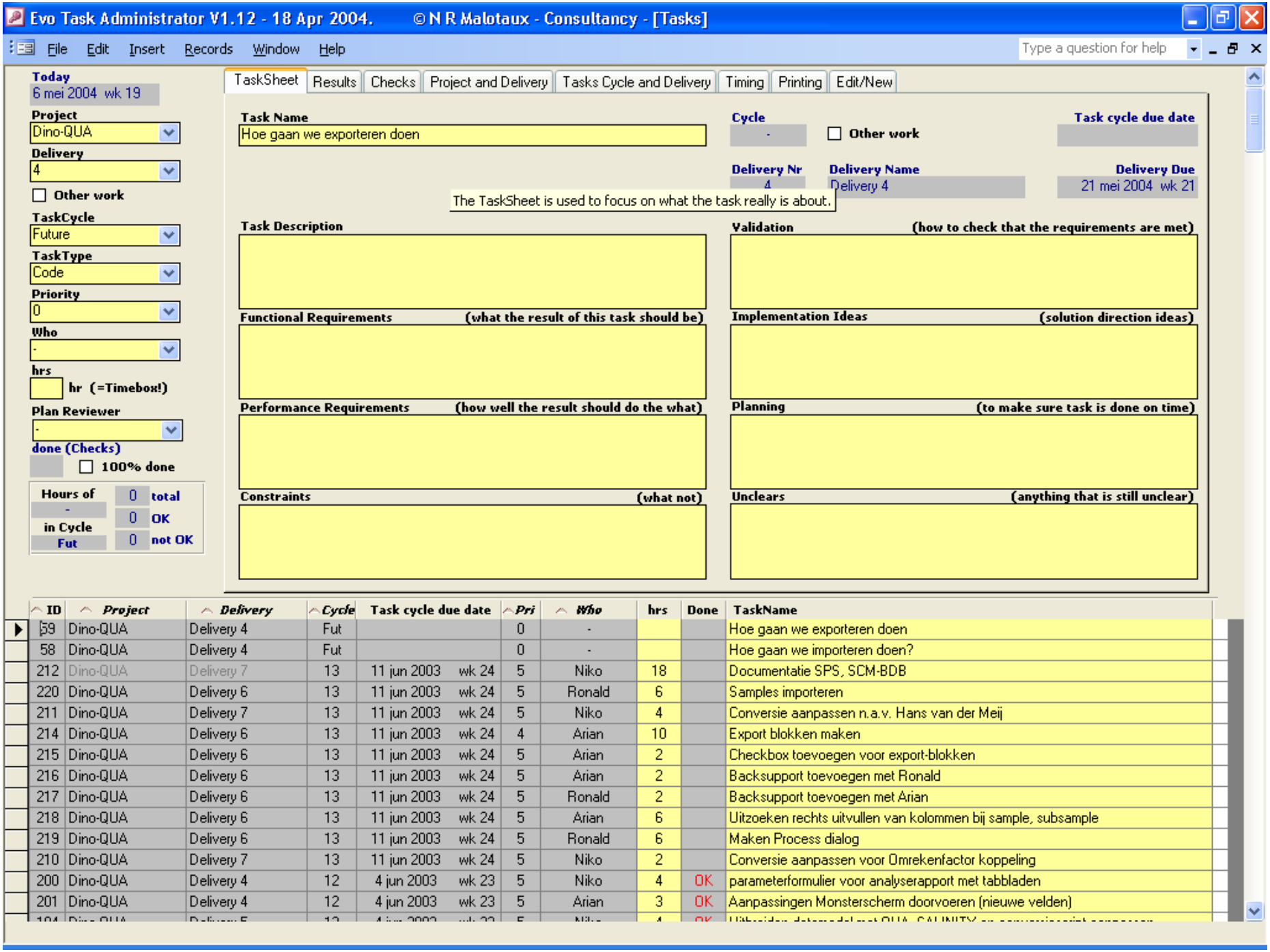
- Meeting with sub-contractors in three weeks
- Many documents to review
- Impossible deadline

	per doc	hr
4 heavy	15	60
3 easy	2	6
	total	66
other work		33
	total	99

available	2 x 26	52

- How many documents to review ?
- How much time per document ?

- Some suggestions …
- Result: well reviewed, great meeting, everyone satisfied


Evo Task Administrator V1.12 - 18 Apr 2004. © N R Malotaux - Consultancy - [Tasks]
File Edit Insert Records Window Help
Type a question for help
Today
6 mei 2004 wk 19
Project
Dino-QUA
Delivery
4
Other work
TaskCycle
Future
TaskType
Code
Priority
0
Who
-
hrs
hr (=Timebox!)
Plan Reviewer
-
done (Checks)
100% done
Hours of
-
in Cycle
Fut
0 total
0 OK
0 not OK
TaskSheet
Results
Checks
Project and Delivery
Tasks Cycle and Delivery
Timing
Printing
Edit/New
Task Name
Hoe gaan we exporteren doen
Cycle
-
Other work
Task cycle due date
Delivery Nr
4
Delivery Name
Delivery 4
Delivery Due
21 mei 2004 wk 21
The TaskSheet is used to focus on what the task really is about.
Task Description
Validation (how to check that the requirements are met)
Functional Requirements (what the result of this task should be)
Implementation Ideas (solution direction ideas)
Performance Requirements (how well the result should do the what)
Planning (to make sure task is done on time)
Constraints (what not)
Unclears (anything that is still unclear)
ID Project Delivery Cycle Task cycle due date Pri Who hrs Done TaskName
59 Dino-QUA Delivery 4 Fut 0 - Hoe gaan we exporteren doen
58 Dino-QUA Delivery 4 Fut 0 - Hoe gaan we importeren doen?
212 Dino-QUA Delivery 7 13 11 jun 2003 wk 24 5 Niko 18 Documentatie SPS, SCM-BDB
220 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Ronald 6 Samples importeren
211 Dino-QUA Delivery 7 13 11 jun 2003 wk 24 5 Niko 4 Conversie aanpassen n.a.v. Hans van der Meij
214 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 4 Arian 10 Export blokken maken
215 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Arian 2 Checkbox toevoegen voor export-blokken
216 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Arian 2 Backsupport toevoegen met Ronald
217 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Ronald 2 Backsupport toevoegen met Arian
218 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Arian 6 Uitzoeken rechts uitvullen van kolommen bij sample, subsample
219 Dino-QUA Delivery 6 13 11 jun 2003 wk 24 5 Ronald 6 Maken Process dialog
210 Dino-QUA Delivery 7 13 11 jun 2003 wk 24 5 Niko 2 Conversie aanpassen voor Omrekenfactor koppeling
200 Dino-QUA Delivery 4 12 4 jun 2003 wk 23 5 Niko 4 OK parameterformulier voor analyserapport met tabbladen
201 Dino-QUA Delivery 4 12 4 jun 2003 wk 23 5 Arian 3 OK Aanpassingen Monsterscherm doorvoeren (nieuwe velden)

Developing a new oscilloscope

- 4 teams of 10 people, 8 more people in Bangalore
- Introduced first in one team
- Other teams followed once convinced
- One team lagged because fear of 'micro-management'

- Even if we would drop all you suggested, the 1-on-1's will be kept, because so powerful:
  - We used to do something and afterwards found out it wasn't what it should be
  - Now we find out before, allowing us to do it more right the first time

Results

- Schedule accuracy for this platform development was 50% better than the program average (as measured by program schedule overrun) over the last 5 years
- This product was the fastest time-to-market with the highest quality at introduction of any platform in our group in more than 10 years
- The team also won a prestigious Team Award as part of the company's Technical Excellence recognition program

www.malotaux.nl/doc.php?id=19 chapter 4.7.1, page 70

Software project in Poland

- **'Mission Impossible': Delivery deadline in 6 weeks**
- **Will you succeed ?**
- **No !**
- **Failure is not an Option !**
- **Changed their way of working** (some coaching)
- **Delivered to amazed customer in 5 weeks**
- **Proudly confided: "Not working overtime !"**

Now we are already much more efficient

- 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 ?

Evolutionary Planning

TimeLine

If we add something …

If we add something, something else will not be done

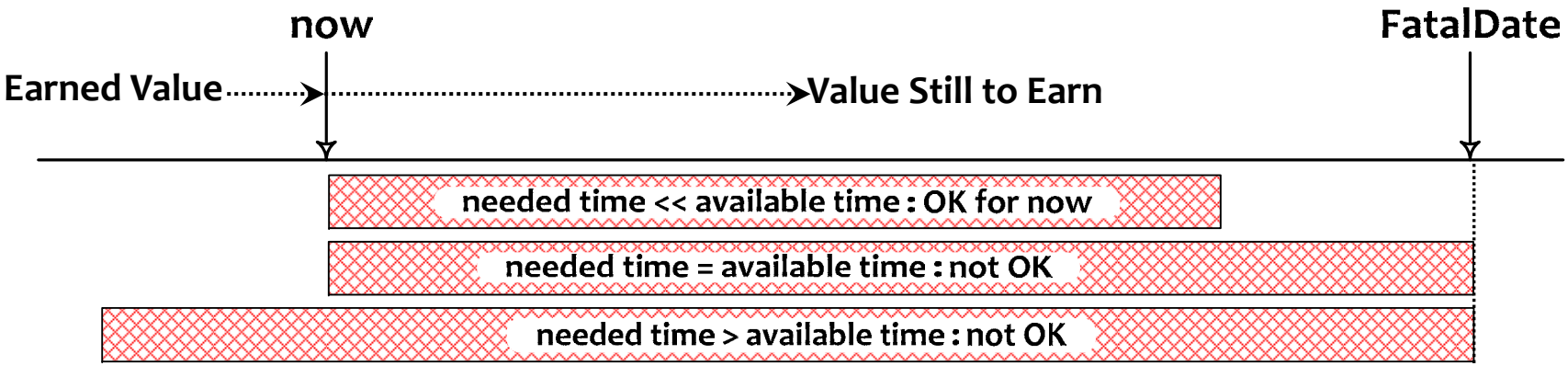
Rather than letting it happen randomly

We better decide what will happen

Did you prepare ?

- The top-3 stakeholders of your work (*Who* is waiting for it?)
- The top-3 real requirements for your work (What are they waiting for?)
- How much value improvement the stakeholders expect (3 or 7?)
- **Any deadlines** (No deadlines: it will take longer)
- What you should and can have achieved in the *coming 10 weeks*
  (Will you succeed? - Failure is not an option!)
- What you think you should and can do the *coming week* in order to achieve what you're supposed to achieve (Make sure not to plan what you shouldn't or cannot do - At the end of the week everything you planned will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work

Deadlines



- **Value Still to Earn**

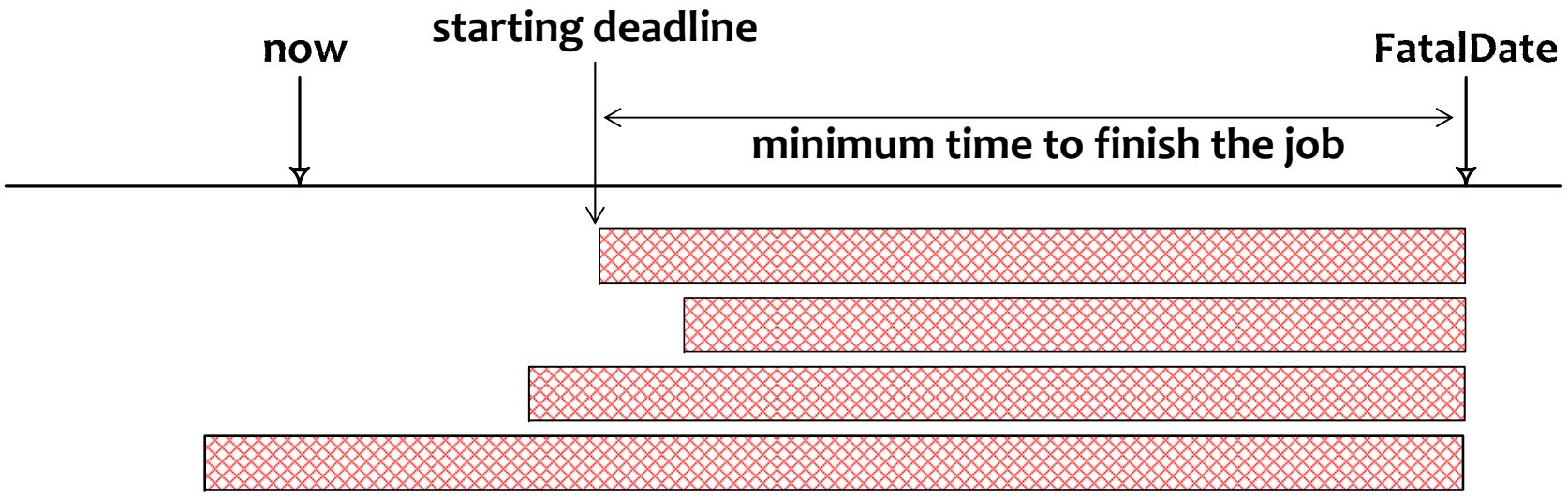
versus

- **Time Still Available**

If the match is over, you cannot score a goal

Even more important: *Starting Deadlines*

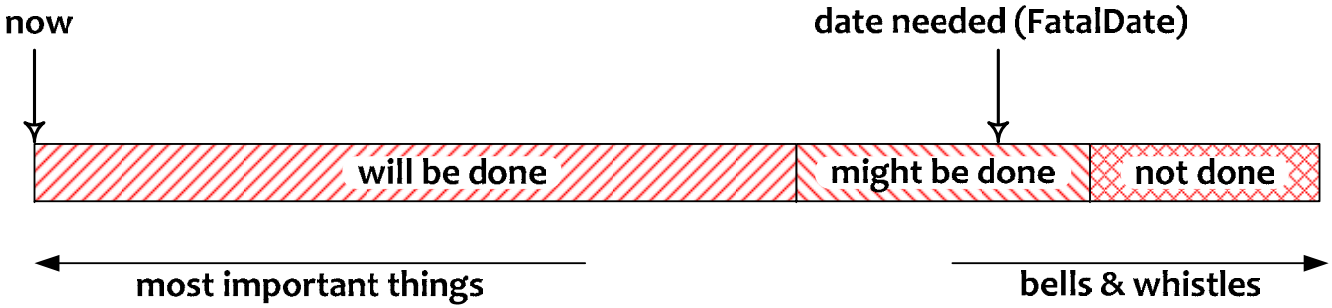
- **Starting deadline**
  - Last day to start to make the finish deadline
  - Every day we start later, we will end later



TimeLine

What the customer wants, he cannot afford

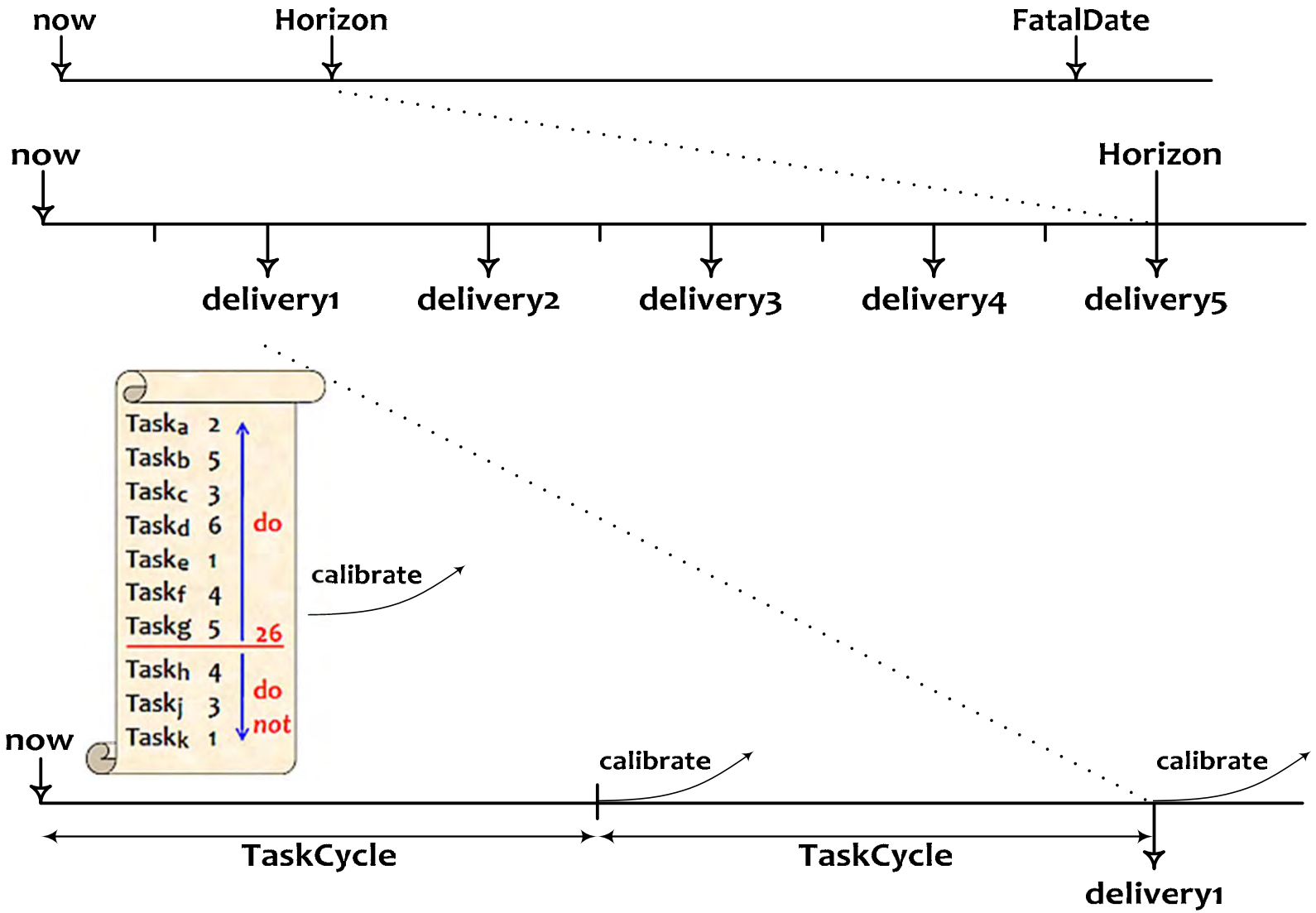
Standard Projects



Evo

If it easily fits ...

Result to Tasks and back



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

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

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	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009
7	Activity 7	1				1.4	1.4	10 Apr 2009
8	Activity 8	3				1.4	4.2	16 Apr 2009
↓	↓							
16	Activity 16	4				1.4	5.6	2 Jun 2009
17	Activity 17	5				1.4	7.0	11 Jun 2009
18	Activity 18	7				1.4	9.8	25 Jun 2009

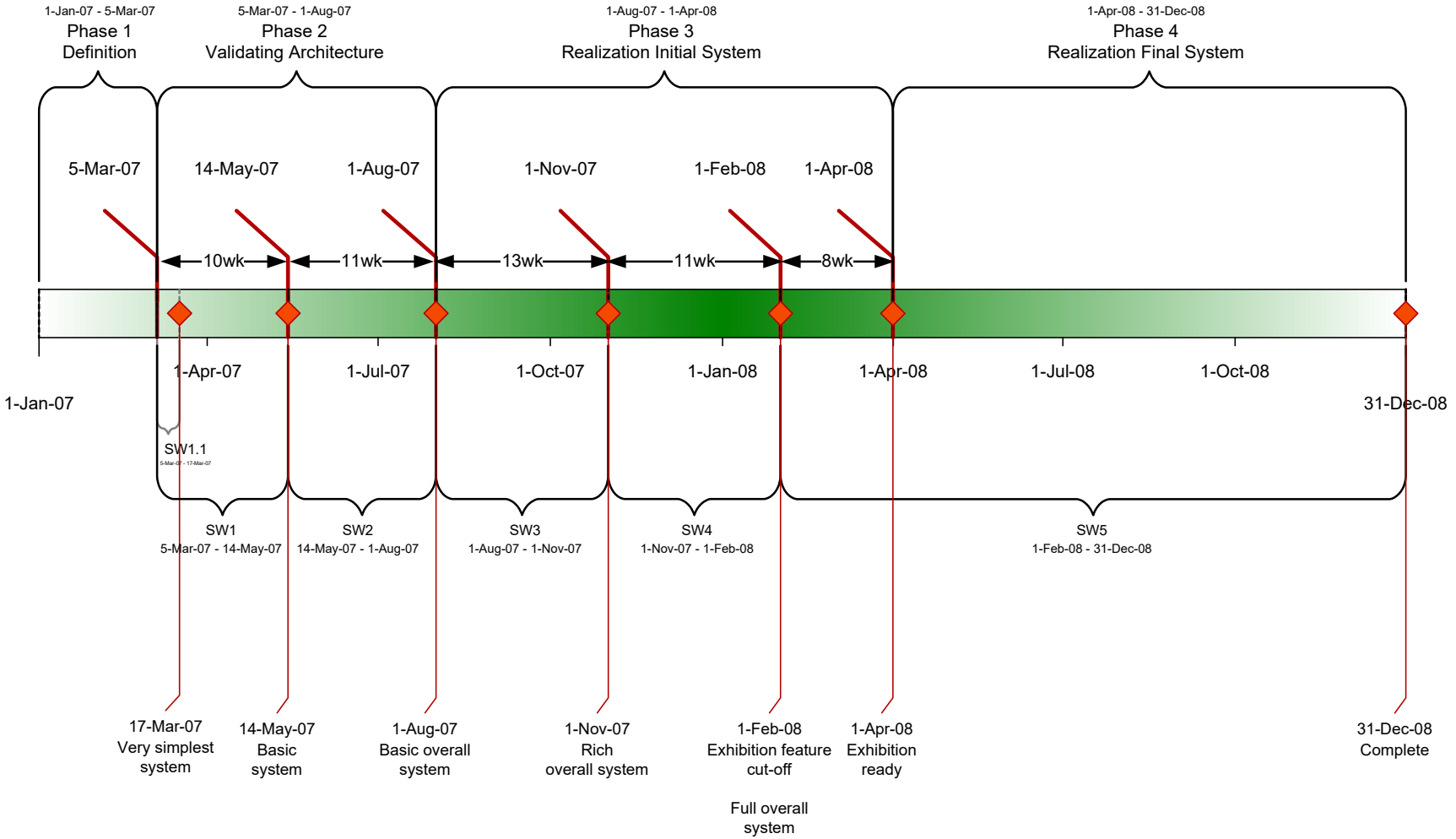
Product/Portfolio/Resource Management

- 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

TimeLine examples

TimeLine example



Sorry

Picture removed for confidentiality

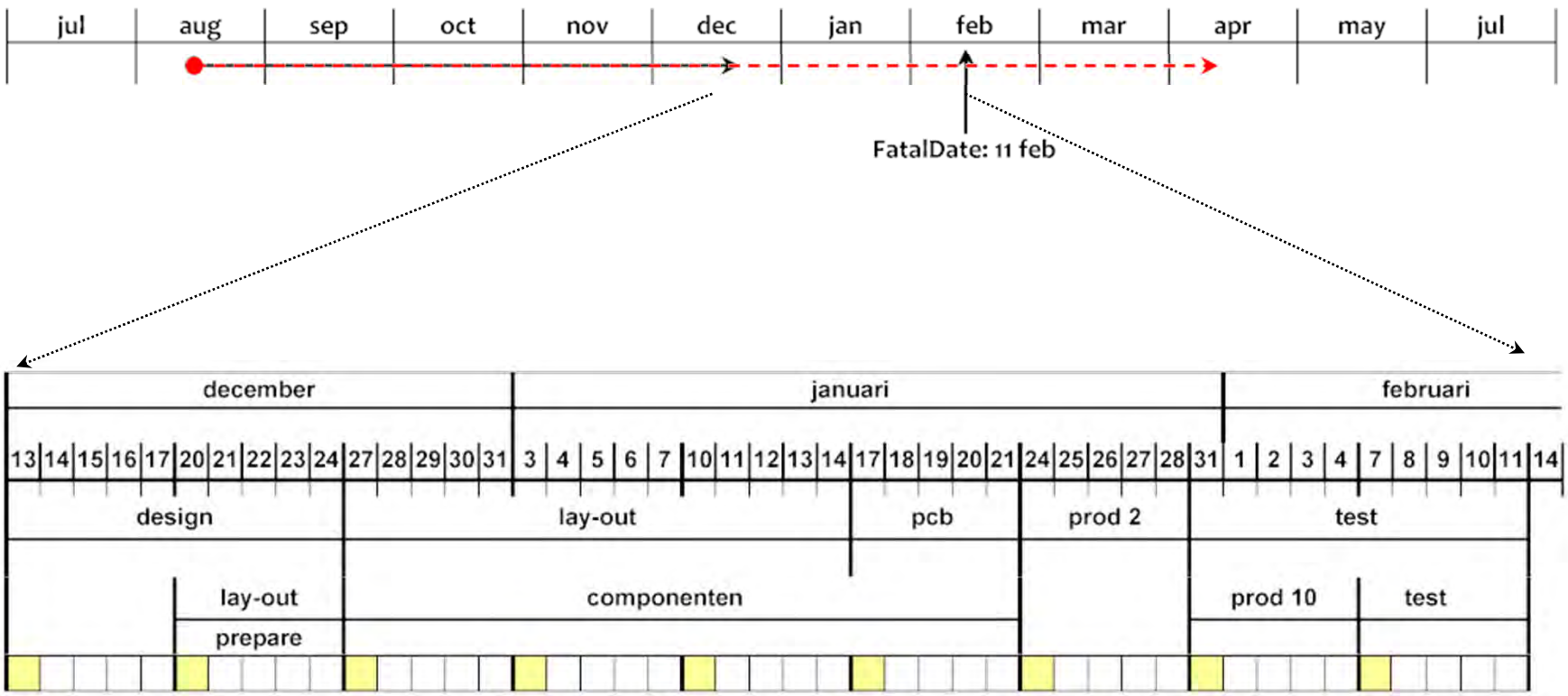
Sorry

Picture removed for confidentiality

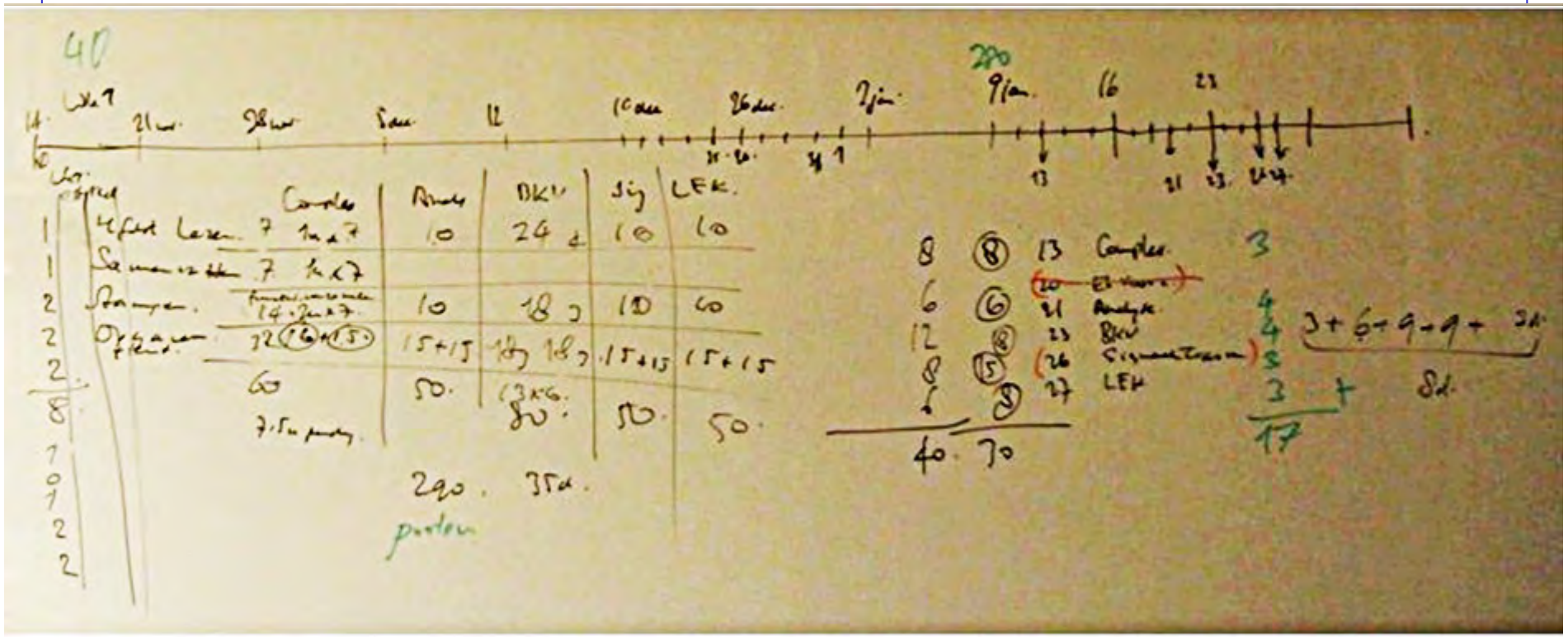
Sorry

Picture removed for confidentiality

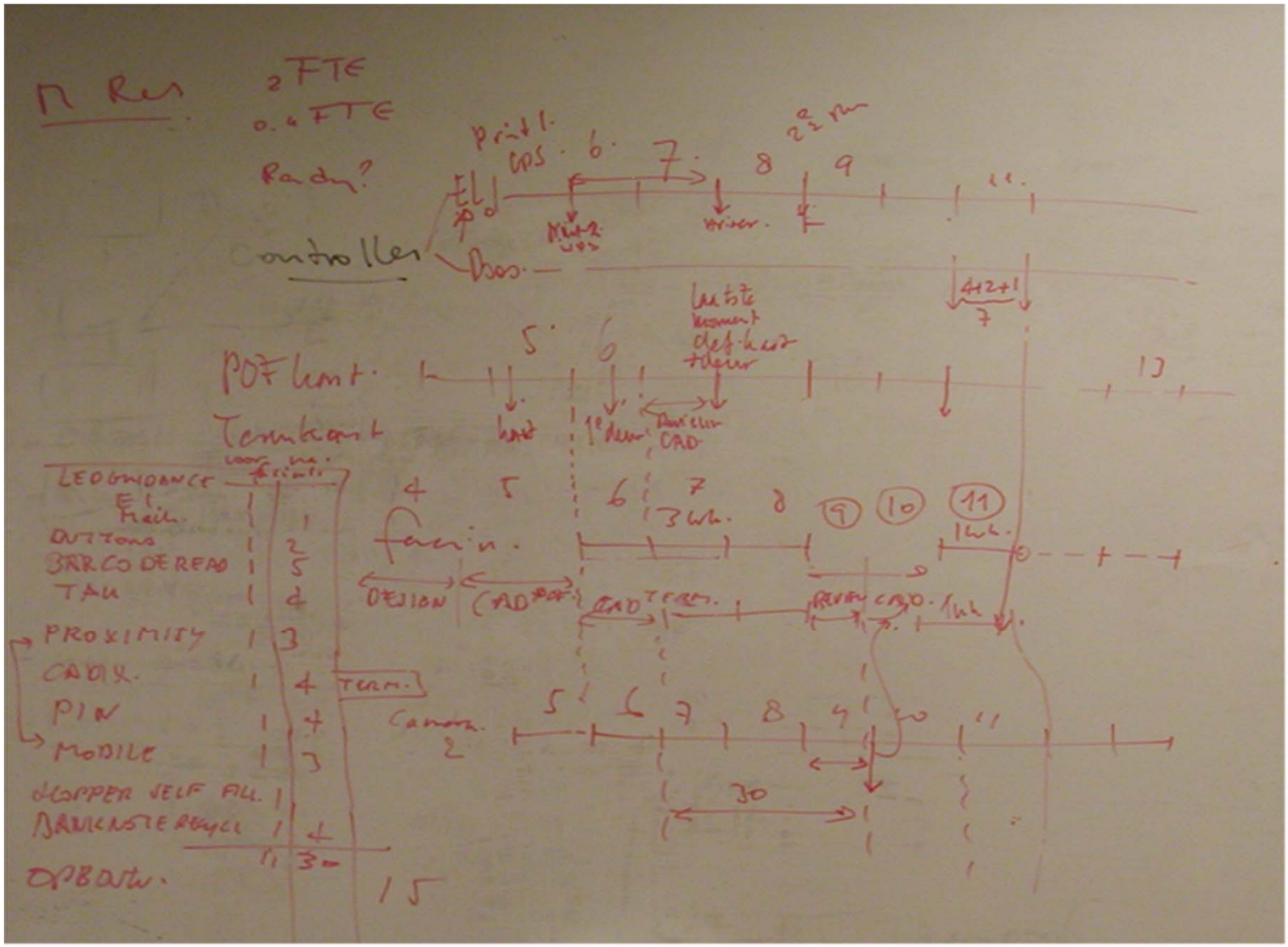
TimeLine planning



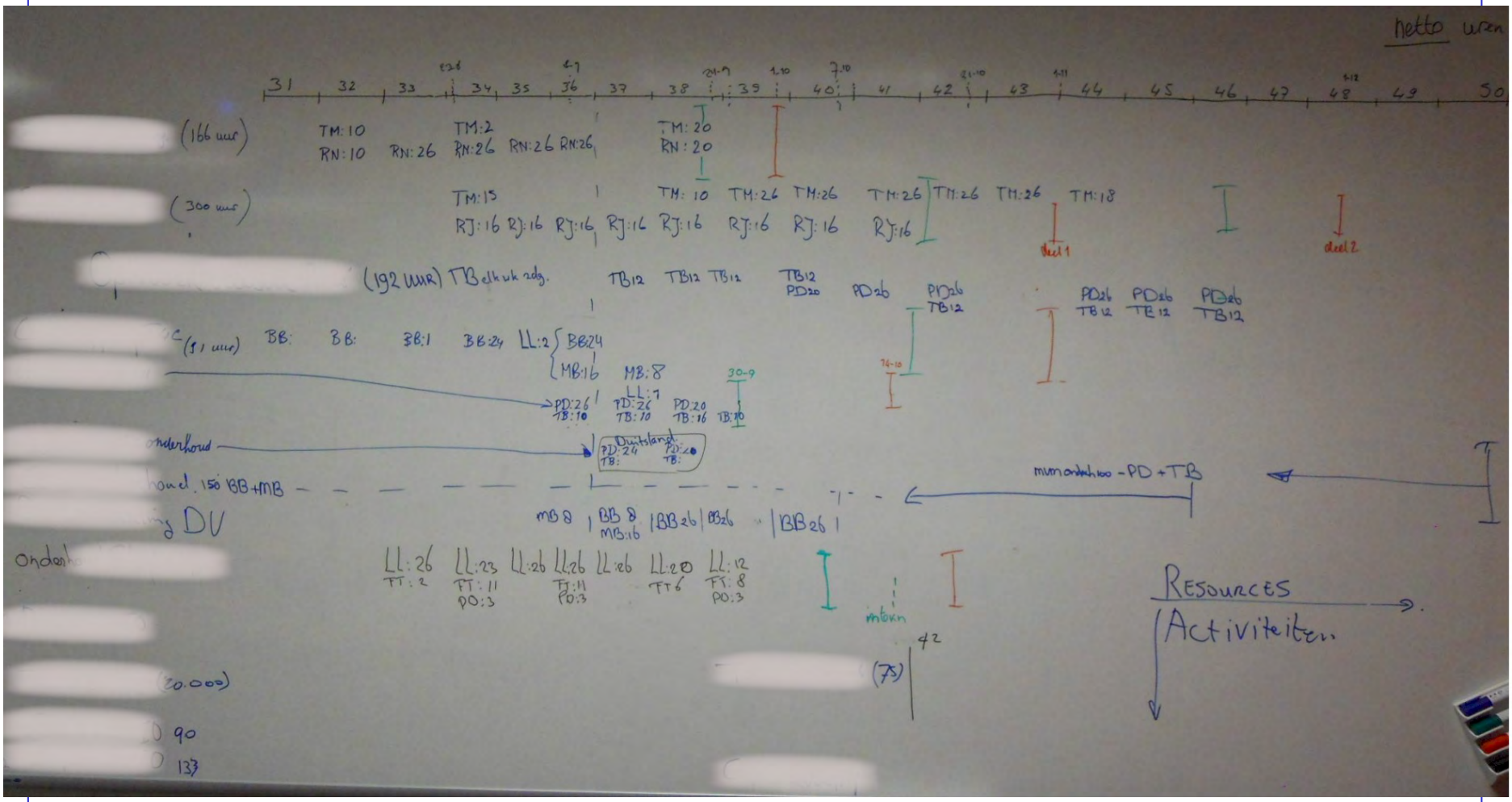
Preparing for student exams



Synchronizing TimeLines



Whiteboard TimeLine Resource Planning



Did you prepare ?

- The top-3 stakeholders of your work (*Who* is waiting for it?)
- The top-3 real requirements for your work (What are they waiting for?)
- How much value improvement the stakeholders expect (3 or 7?)
- **Any deadlines** (No deadlines: it will take longer)
- **What you should and can have achieved in the *coming 10 weeks***
  (Will you succeed? - Failure is not an option!)
- What you think you should and can do the *coming week* in order to achieve what you're supposed to achieve (Make sure not to plan what you shouldn't or cannot do - At the end of the week everything you planned will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work

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!)

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 !

More

www.malotaux.nl/booklets

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

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

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

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

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

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

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

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

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

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

www.malotaux.nl/inspections

Inspection pages

What now ?

Predictable Projects

Delivering the Right Result at the Right Time

Niels Malotaux

N R Malotaux Consultancy

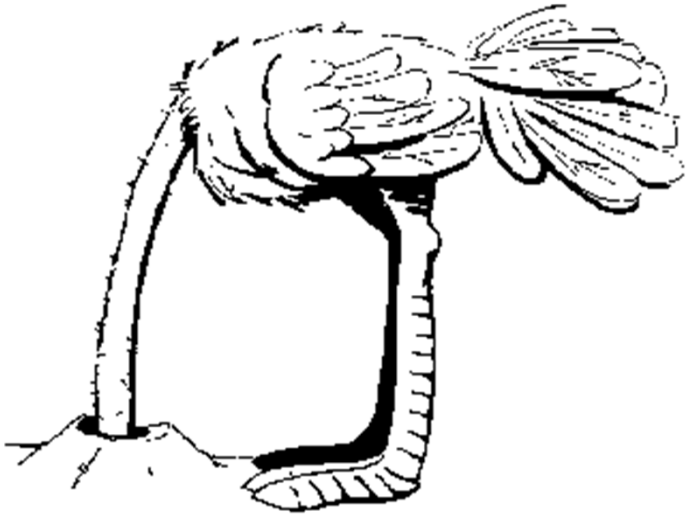
+31-6-5575 3604 niels@malotaux.nl www.malotaux.nl

Help !
We have a
QA problem !

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?**

Do you think you can help us ?



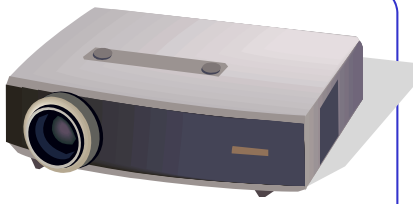
In stead of complaining about a problem …

(Stuck in the Check-phase)

Let's do something about it !

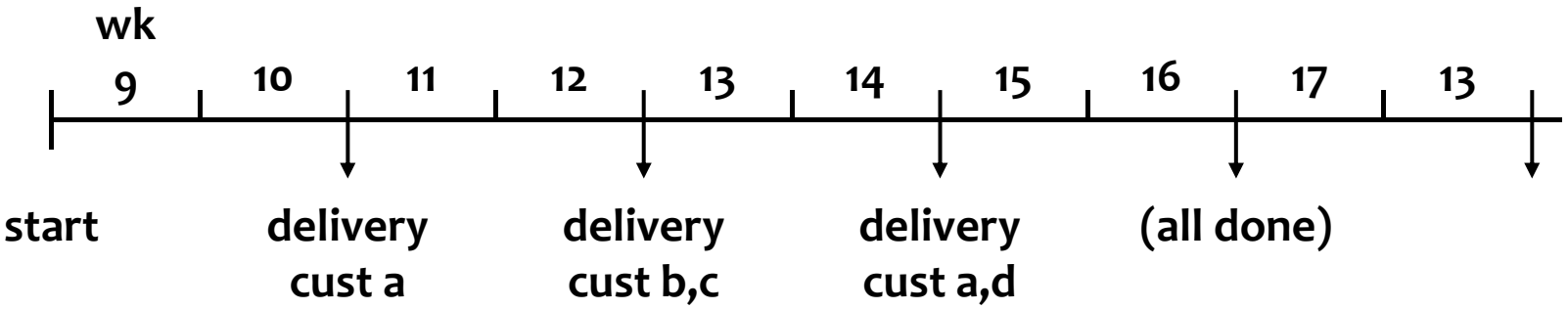
(Moving to the Act-phase)

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



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

TimeLine



Selecting the priority order of customers to be served

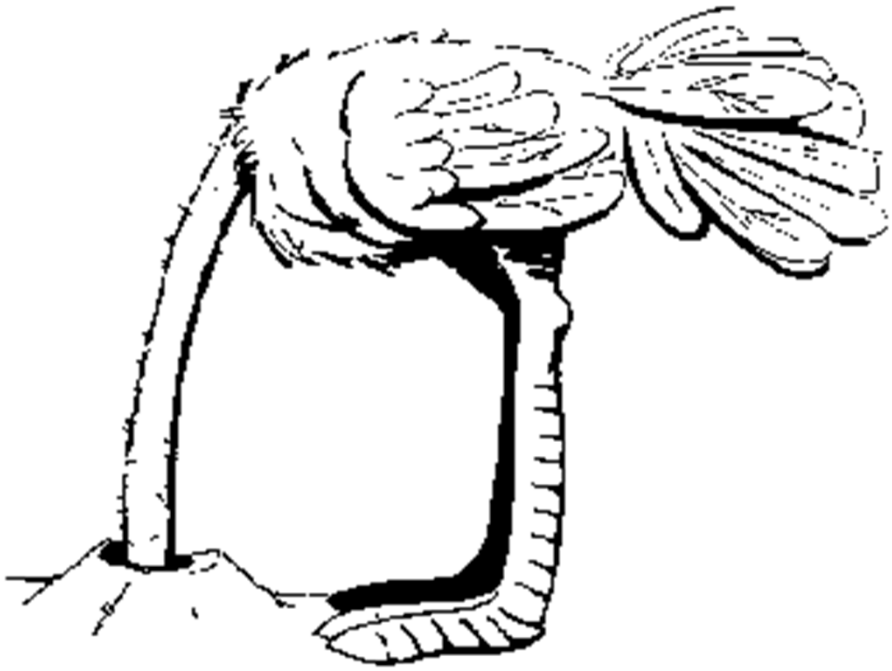
- **"We'll have a solution at that date … Will you be ready for it ?"**
  An other customer could be more eagerly waiting
- **Most promising customers**

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



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

Doing retrospectives does not solve the problem !

Prespectives save a lot of time

Some extra

Active Synchronization

Somewhere around you, there is the bad world.

If you are waiting for a result outside your control, there are three possible cases:

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

- **If you are not sure (case 2), better assume case 3**
- **From other Evo projects you should expect case 1**
- **Evo suppliers behave like case 1**

In cases 2 and 3: Actively Synchronize: Go there !

1. **Showing up increases your priority**
2. **You can resolve issues which otherwise would delay delivery**
3. **If they are really late, you'll know much earlier**

Interrupts

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

- **In case of interrupt, use interrupt procedure**

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.**

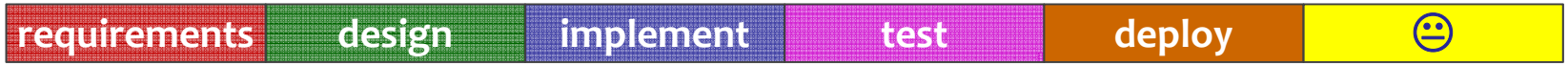
Example

Customer Relations Management project

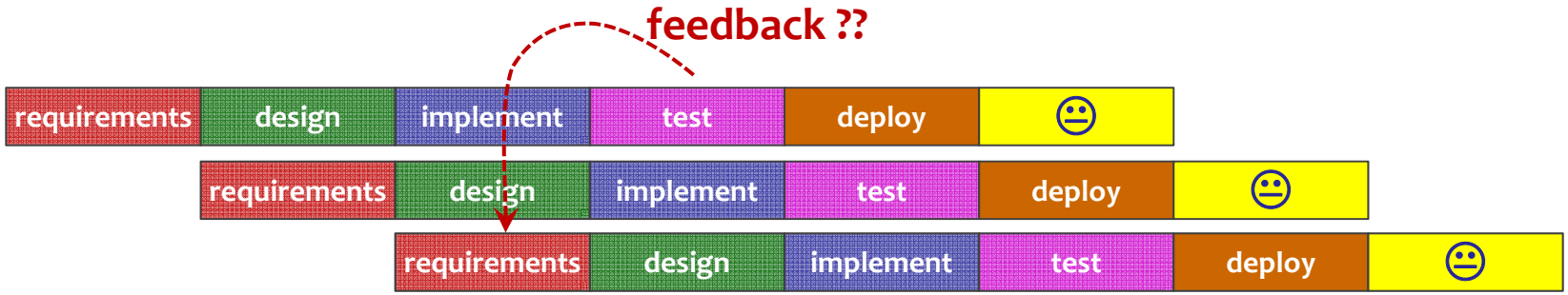
- CRM system, original plan: 6 months and € 1M
- Spent 1.5 years and € 5M
- Business hasn't seen any result whatsoever
- Systems Integrator still "working hard"

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

Few larger deliveries

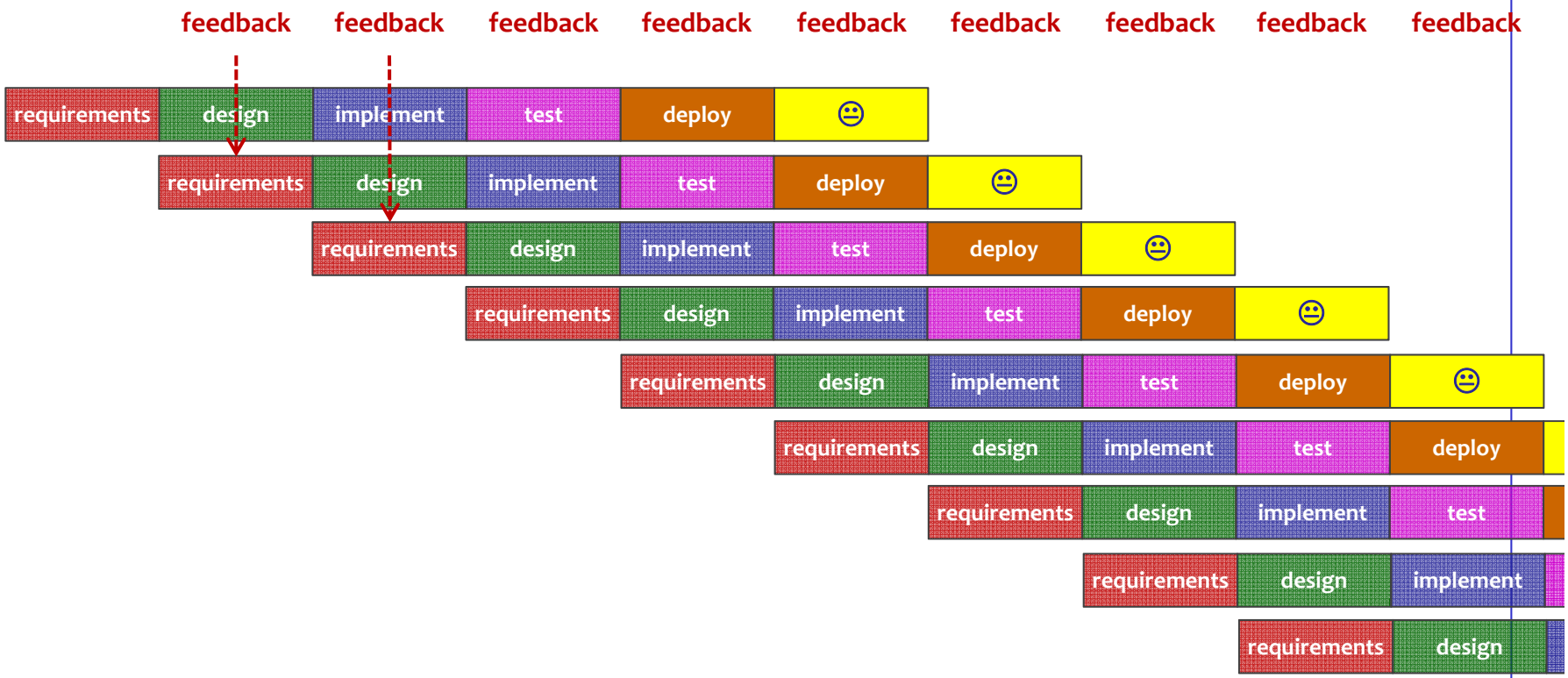


- Evolutionary Delivery ?



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

Suggestion to start a conversation



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)

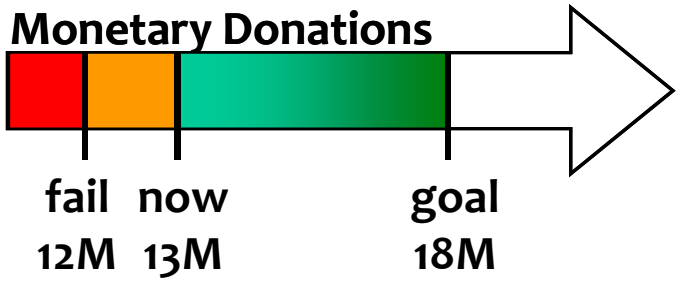
If time, money, resources are limited,
we should not overrun the budgets

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

Objective: Monetary Donations

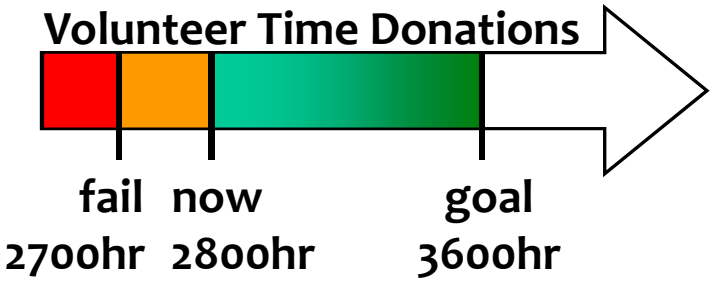


Name	Monetary Donations
Scale	Euro's donated to non-profits through our website
Meter	Monthly Donations Report

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

Ref Ryan Shriver
ACCU Overload Feb 2009

Objective: Volunteer Time (Natura) Donations



Name Volunteer Time Donations

Scale Hours donated to non-profits through our website

Meter Monthly Donations Report

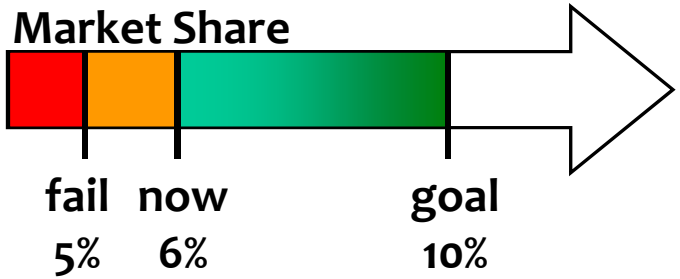
Fail 2700 hr

Now 2800 hr [2008] ← Annual Report 2008

Goal 3600 hr [2009]

Ref Ryan Shriver
ACCU Overload Feb 2009

Goal: Market Share



Name Market Share

Scale Market Share %% online giving

Meter Quarterly Industry Report

Fail 5%

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

Goal 10% [Q1-2010]

Ref Ryan Shriver
ACCU Overload Feb 2009

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

Architecture and Design

Design is always a compromise

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

example:

- Performance 
- Budget (time, money) 

Design and requirements

- **Design:**
  Finding the best compromise between the conflicting requirements

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

Design Process

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

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

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

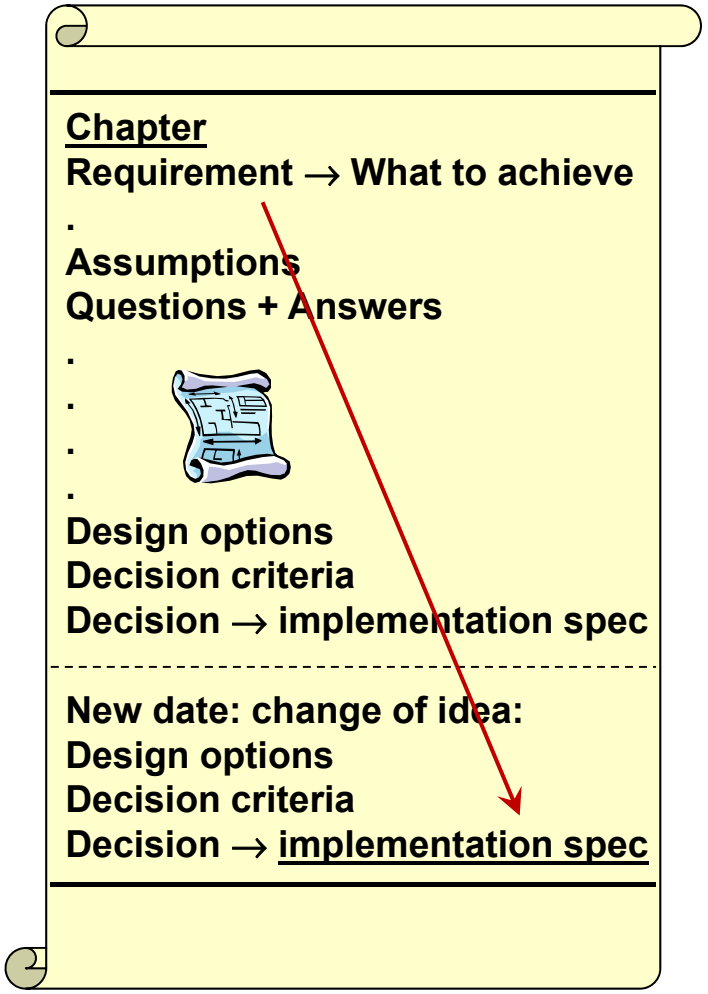
Could we get all, within the budgets of time and cost ?

	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	

DesignLog

(project level)

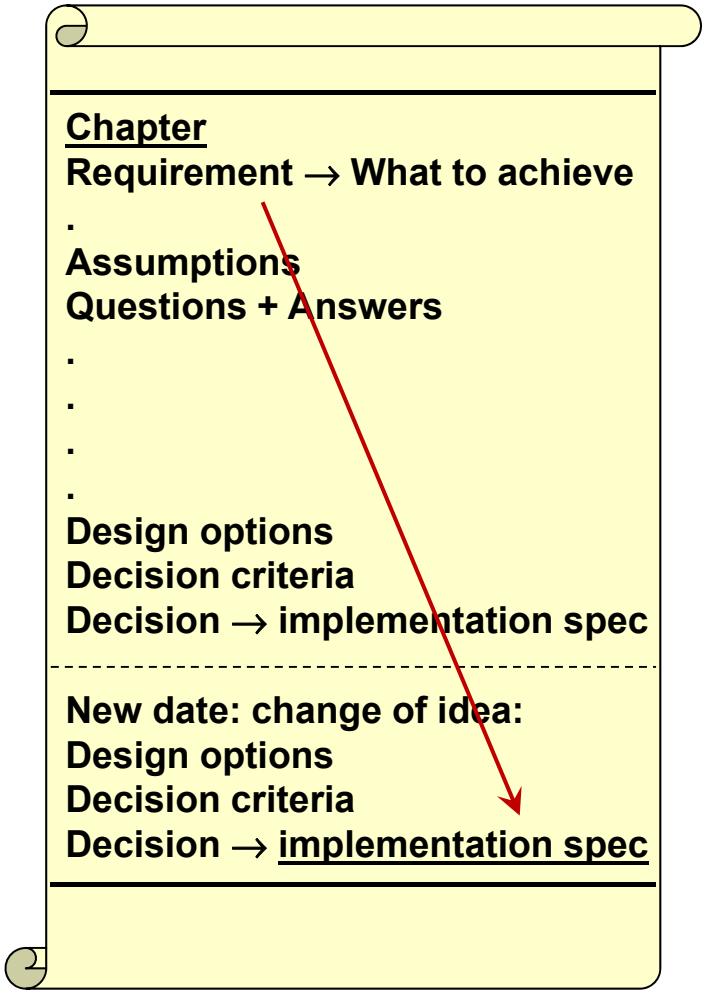
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + reasoning:
    - If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



ProcessLog

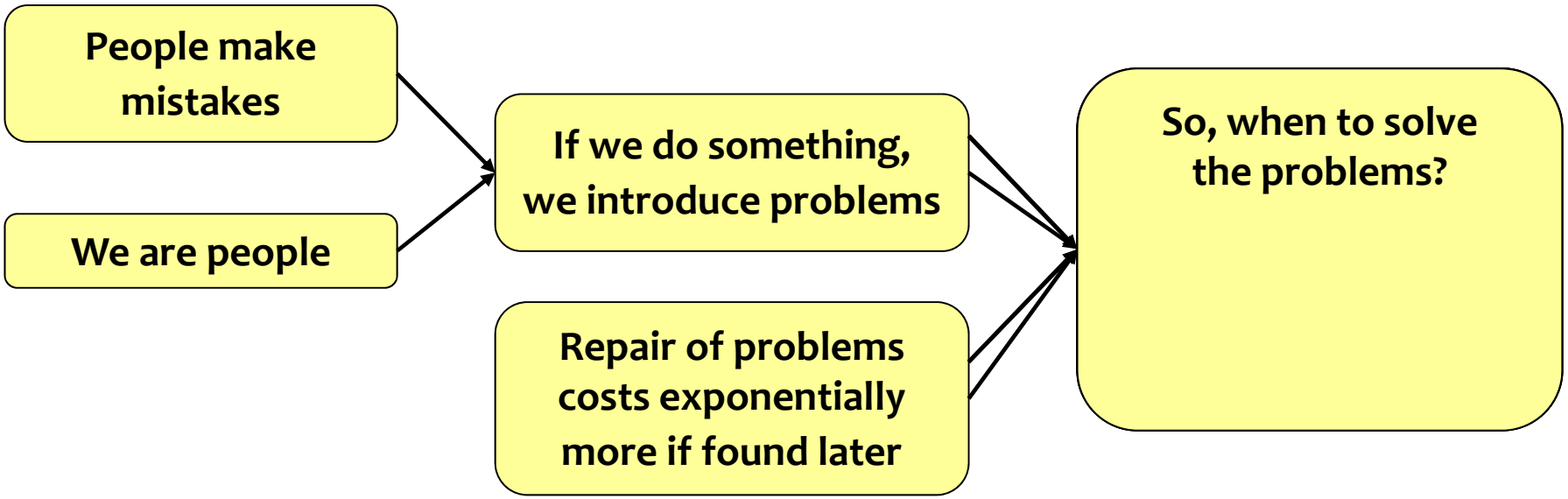
(department / organization level)

- 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



Early Reviews & Inspections

Inevitable consequence



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**

Use the three rules on these Requirements

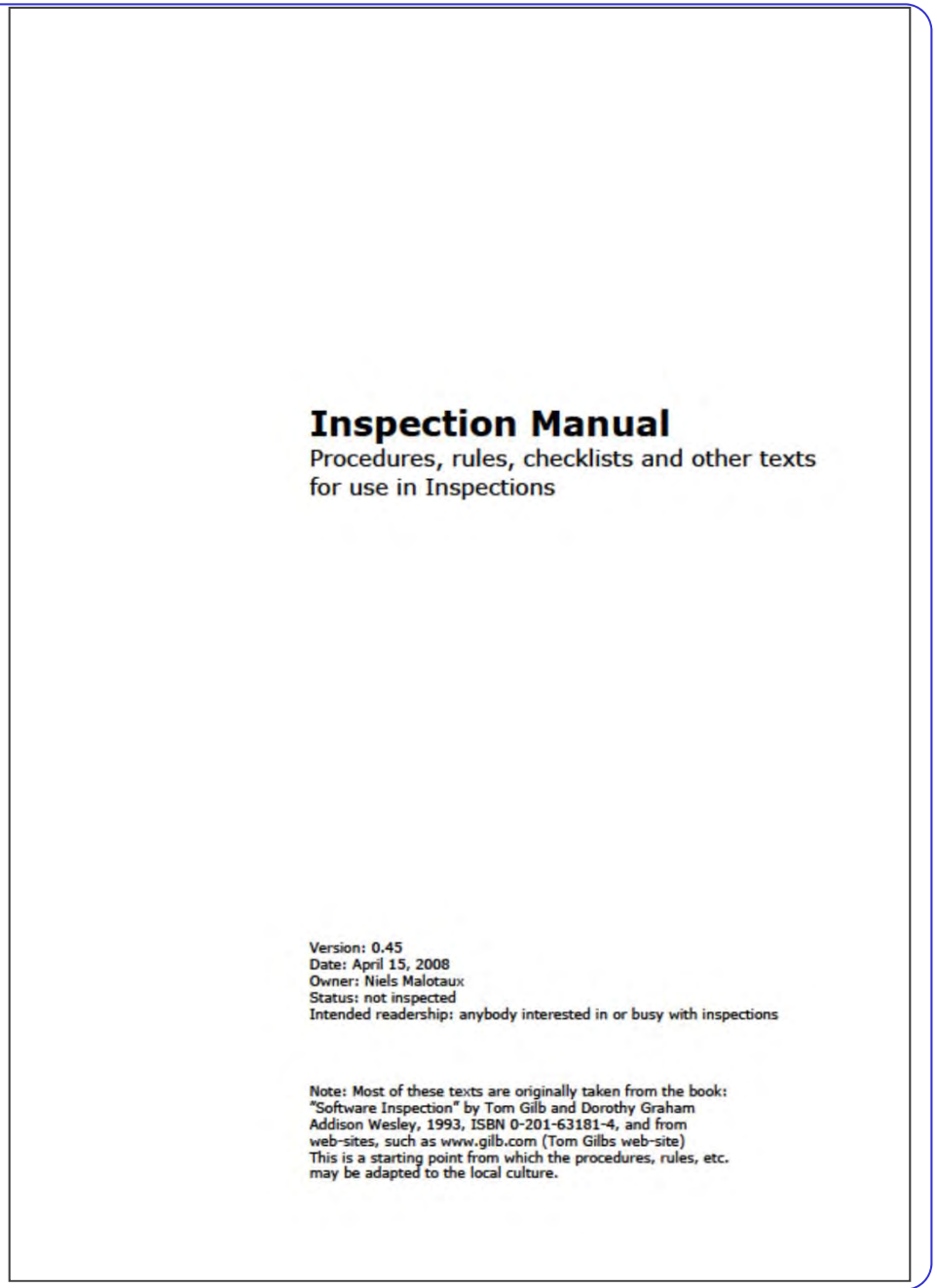
It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality

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

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

16 page Inspection Manual

www.malotaux.nl/doc.php?id=61



Inspection Manual

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

Version: 0.45
Date: April 15, 2008
Owner: Niels Malotaux
Status: not inspected
Intended readership: anybody interested in or busy with inspections

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

Early Inspection

Prevention costs less than Repair

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*

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

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

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

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

Case Study 2 - Situation

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

Case Study 2 - Results

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

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

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

Early Detection vs. Prevention

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

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

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

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

Nice job! Early detection has big benefits - BUT…

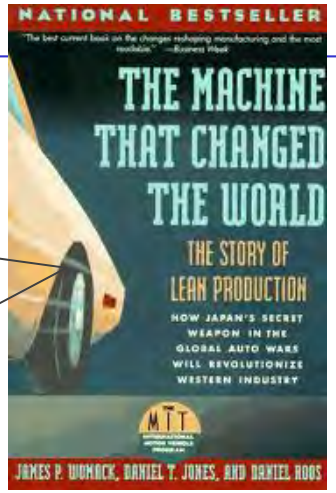
How many of the 3512 defects found in end-of-line inspections could have been completely prevented by Early Inspection?

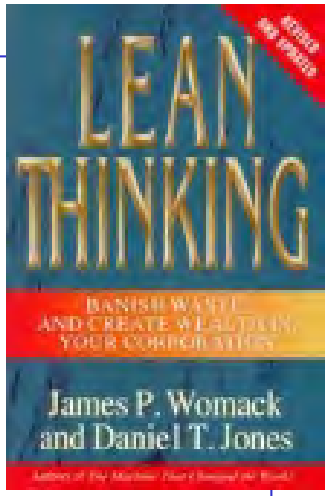
Cost-effective defect prevention is the bottom line

Lean ?

Lean

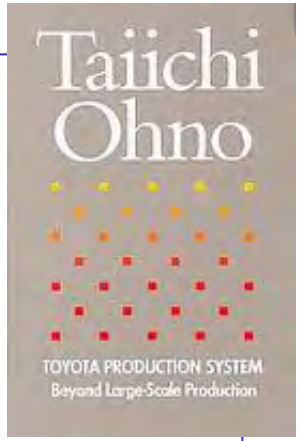
A lot of the cost of vehicles is based on:
- bad design
- poor management
- an attitude that problems, no matter how small, can be overlooked





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

Toyota Production System (TPS)



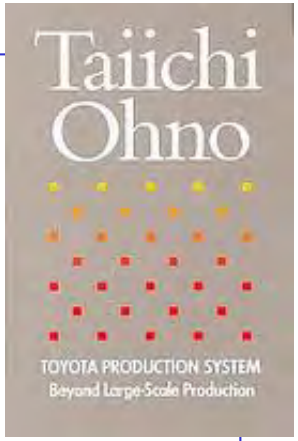
1950

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

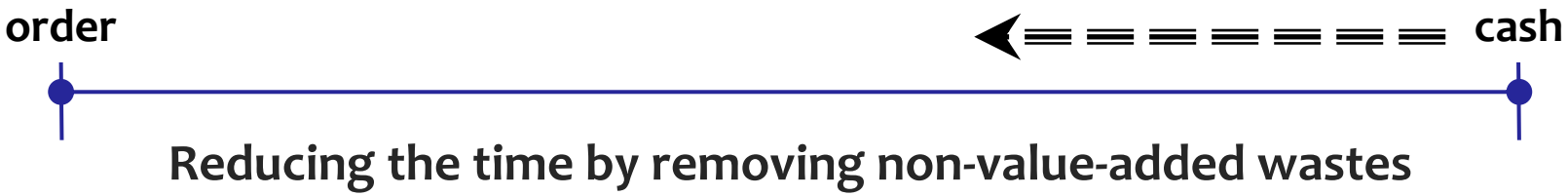
Four specific aims:

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

Taiichi Ohno - The Toyota Production System



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



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

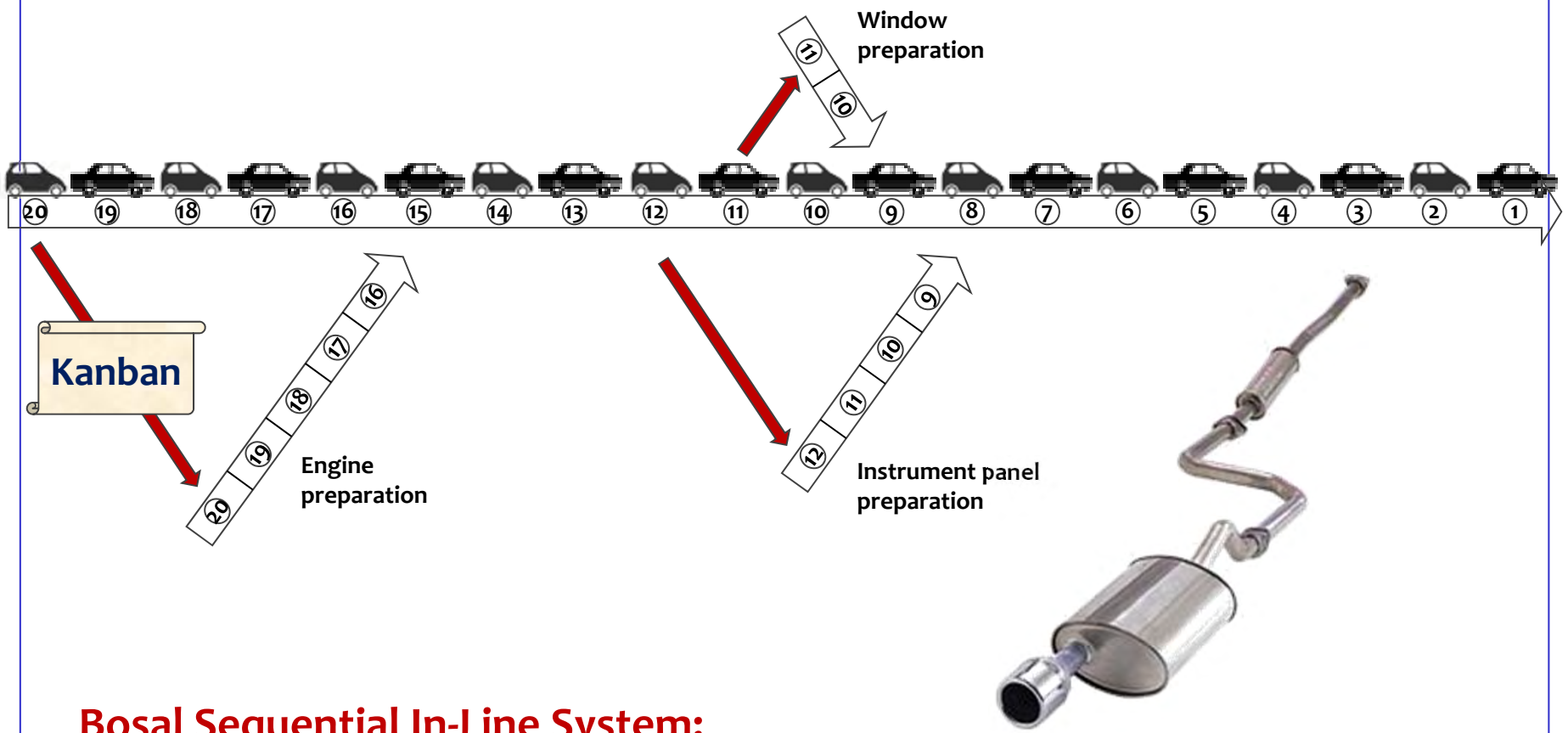
Pillars of the TPS

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

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

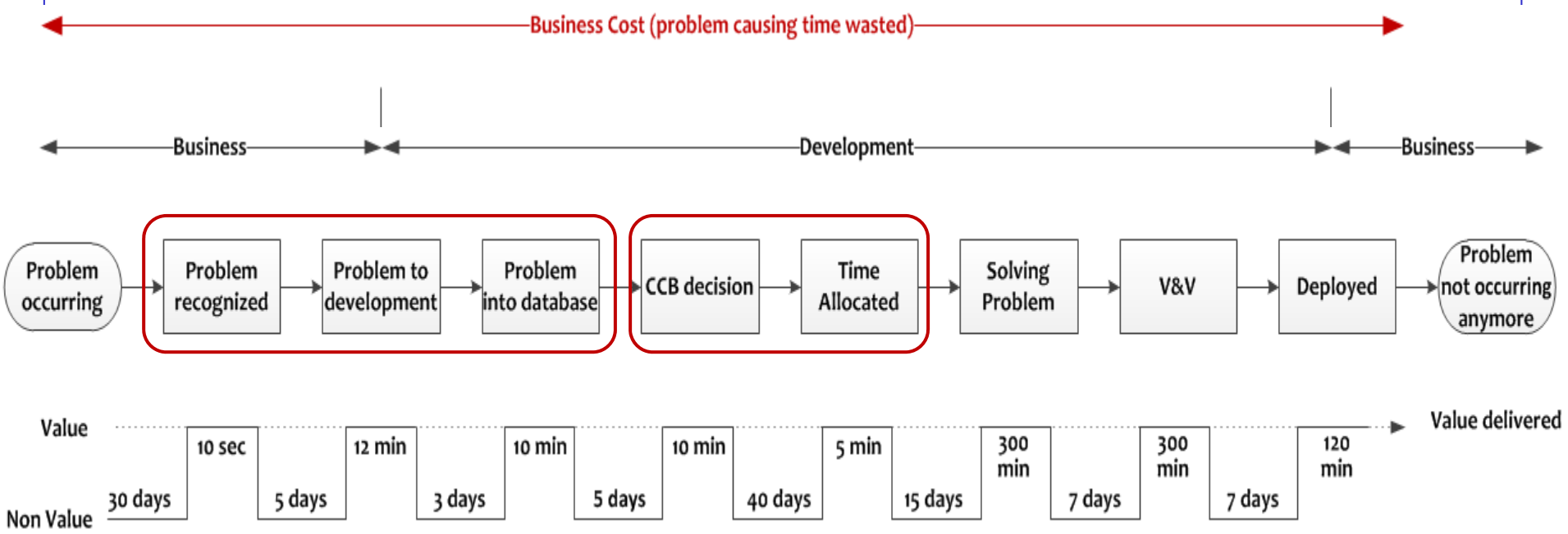
Just In Time delivery – *no inventory*

(after Ohno)



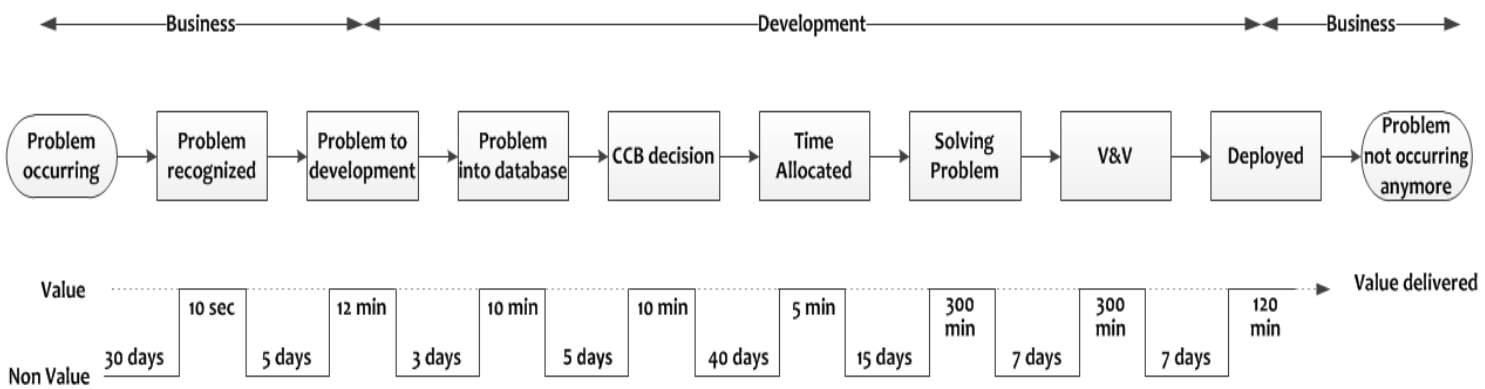
Bosal Sequential In-Line System:
We pioneered just-in-time delivery of exhaust systems - supplying systems to the assembly line within 80 minutes of receiving the order

Value stream example



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

5-S

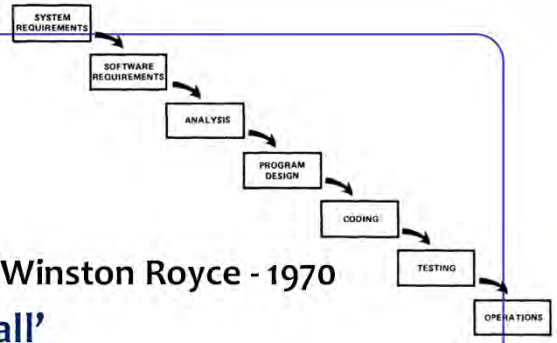


- **Seiri - Remove unnecessary things → waste**
- **Seiton - Arrange remaining things orderly → flow**
- **Seiso - Keep things clean → uncovers hidden problems**
- **Seiketsu - Keep doing it, standardize → know what to improve**
- **Shitsuke - Keep training it → fighting entropy**

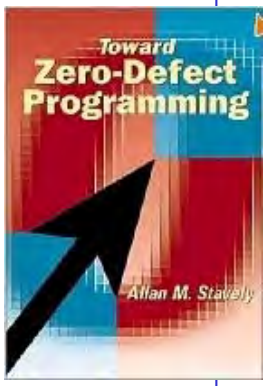
The 3 Mu's to remove

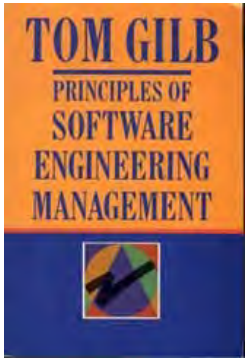
- **Muda - Waste** → **minimize waste**
- **Mura - Irregularities** → **optimize flow**
- **Muri - Stress** → **sustainable pace**

There is nothing new in software too



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





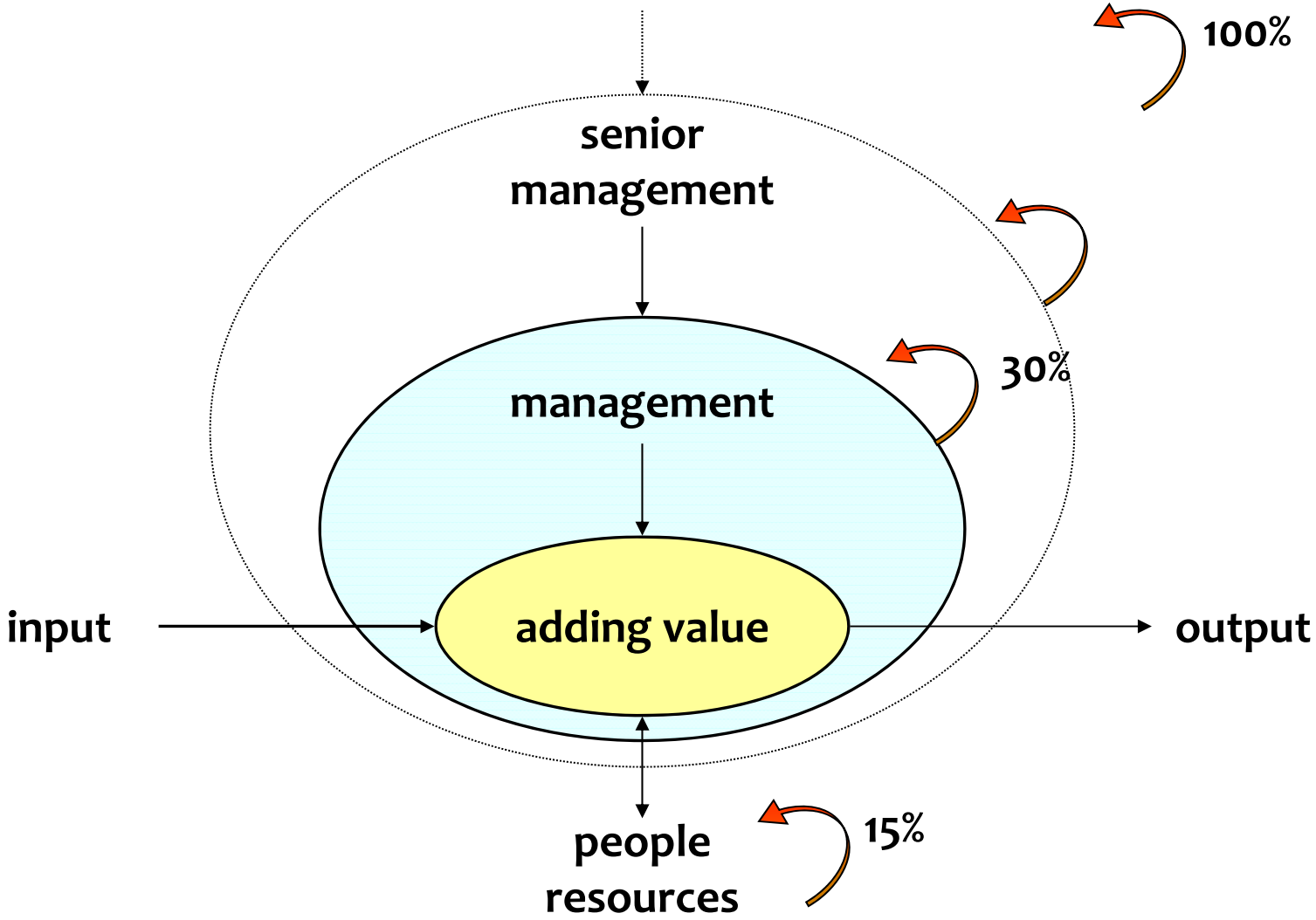
Lean things

- **Most managers think their greatest contribution to the business is doing work-arounds on broken processes, rather than doing the hard work to get the process right so that it never breaks down** (Womack)
- **90 per cent of all corporate problems can be solved using common sense and improving quality while reducing cost through the elimination of waste**
  Imai: *Gemba Kaizen* - A Commonsense Low-Cost Approach to Management
- **Root-Cause-Analysis on every defect found ?**
  We don't have time for that ! (project manager)
- **Plan-Do-Check-Act cycle was by far the most important thing we did in hindsight** (Tom Harada)

Four processes

Managers Tasks

The managers task



Managers have to learn

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

Local Loop Principle