Predictable Projects

Getting the Right Results at the Right Time

Niels Malotaux

N R Malotaux
Consultancy

niels@malotaux.nl

www.malotaux.nl

Niels Malotaux

- Project Coach

- 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 back on track

Result Management

Who are you ?

*

Schedule

24-27 September 2013

Tuesday	**17:15 ~ 18:45**	**19:00 ~ 20:30**	
Wednesday	**17:15 ~ 18:45**	**19:00 ~ 20:30**	
Thursday	**13:00 ~ 14:30**	**14:45 ~ 16:15**	**← !!**
Friday*	**17:15 ~ 18:45**	**19:00 ~ 20:30**	

*** Note: Friday time may change: we will decide on Tuesday ← !!**

1 credit if you attend *all* lectures and participate in *all* exercises (Keio students only)

Subjects

1. What's the problem, Universal goal of any project
2. Quality, Human behaviour in projects
3. Business case - stakeholders - real requirements
4. How to specify results - How to select the right solution
5. Estimation:
   - Estimation of time
   - Estimation of what to do
6. Evolutionary Project Planning – prevention is better than cure
   - Optimizing the efficiency of what we do
   - Optimizing the effectiveness of what we do
7. How to make sure that we get the right result at the right time
8. How to check that we wrote the right things

What's the problem ?

Predictable Projects ?

- **What is a project ?**

- **Do you have experience with projects ?**
- **Any problems with projects ?**

*

Types of project ?

- Product improvement
- Process improvement
- Software improvement
- Service improvement
- Systems
- Systems of Systems
- Complex systems
- Student exam
- … ?

改善

Every project should improve something, otherwise it's waste

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 ?

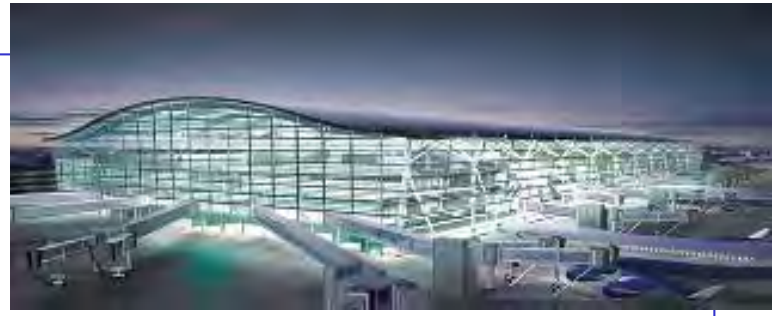
- How about your project ?
  Did you deliver the right result at the right time ?

Delivering the right result

- **What is the right result ?**
- **How do we know ?**
- **Is it really ?**

*

Real Requirements



- **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 ?**

Is being on time important ?

納期って重要？

- **What is on time ?**

*

Delivery time is a Requirement

(納期を守ることも要求です)

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

- **Are they really?**

Fallacy of '*all*' requirements

- "We're done when *all* requirements are implemented"
- Isn't delivery time a requirement ?
- Requirements are always *contradictory* (要求は相反する)
- 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*

Why is time important

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

Causes of Delay

遅延の原因

- **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

原因の原因

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

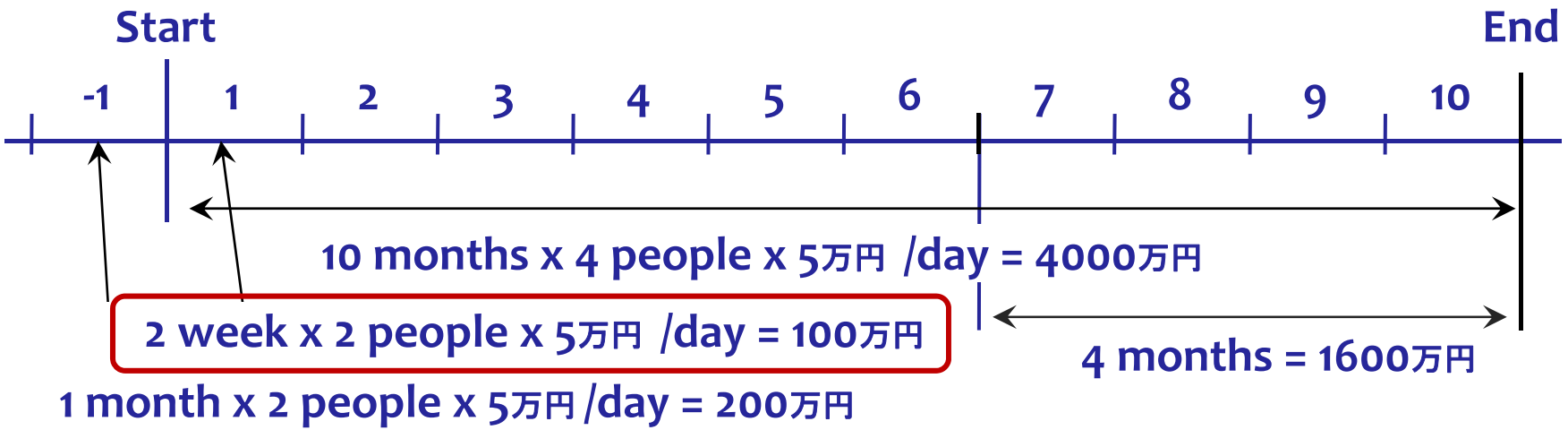
Intuition often points us in the wrong direction

What is the 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**

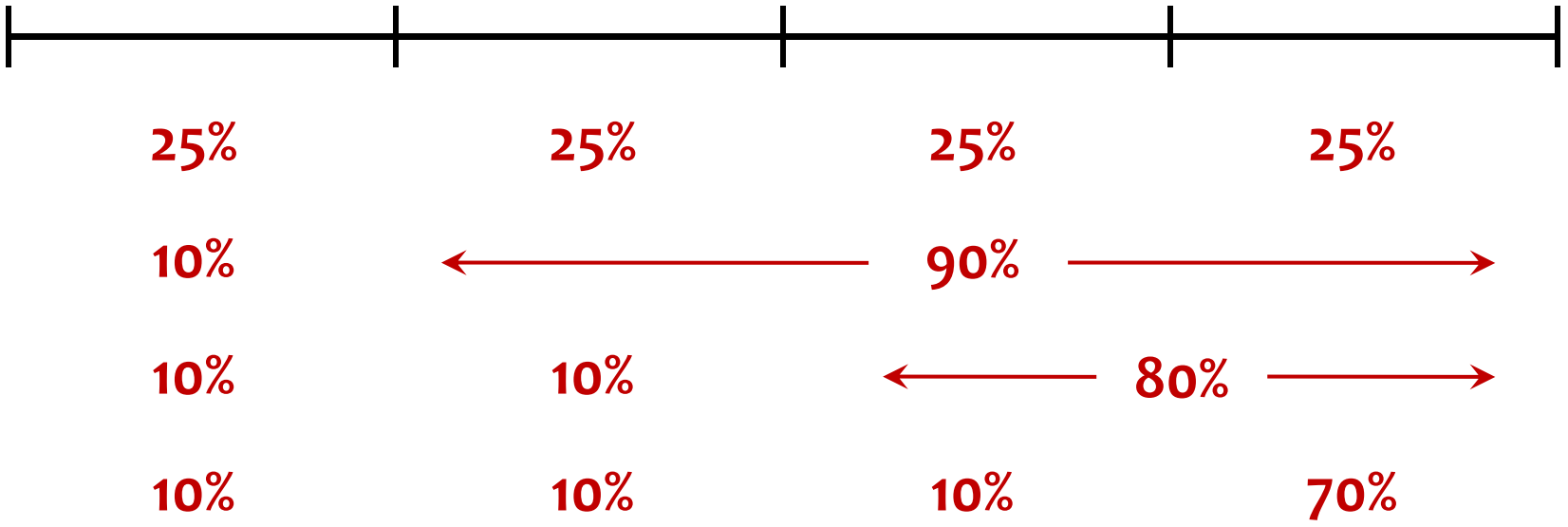
The Cost of Time 時間の値段



- We can save 4 months by investing 2000万円 → "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? → 16億円/yr (1.3 億円 /mnd)
- So 2 weeks extra doesn't cost 100万円. It costs 16億円/26 = 6200万円
- And saving 4 months brings 16億円/3 = 5億円 extra

➔ Invest that 2000万円 *NOW* and *don't waste time*!

4 week project

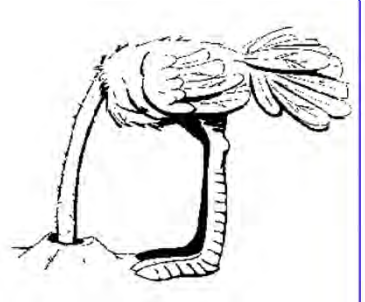


The challenge

- 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 doing the right things at the right time
- The rest is *waste* (むだ)
- Who would like to produce waste ?

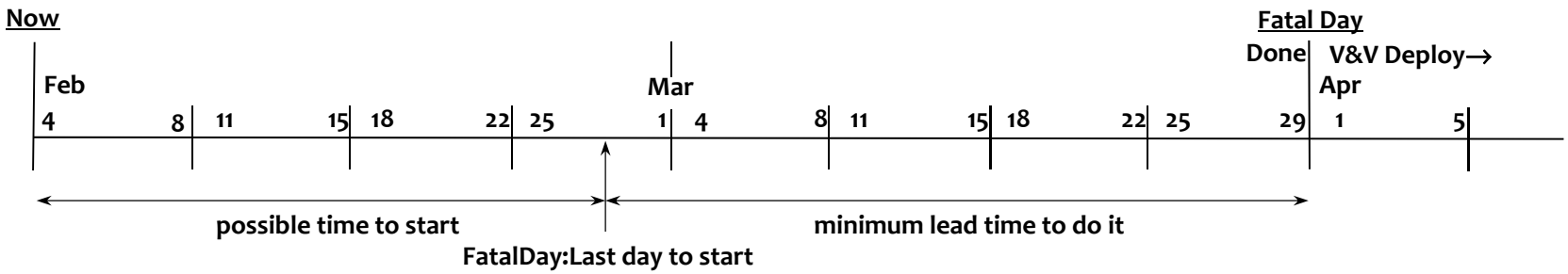
FatalDay

- FatalDay is the last moment *it* shall be there
- After the FatalDay, we'll have real trouble if the Result isn't there
- Count backwards from the FatalDay to know when we should have started (starting deadlines !)
- If that's before now, what are we going to do about it, because *failure is not an option*



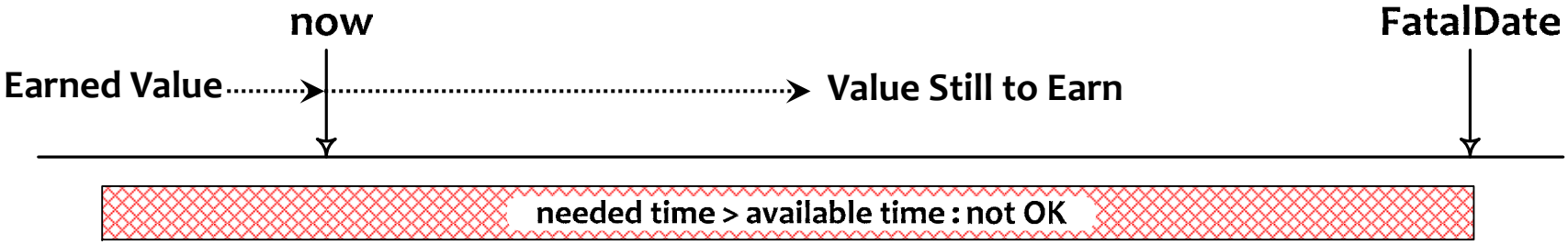
Starting deadlines

Last day of starting not to need an excuse later



Last day of starting some activity not to need an excuse later

If the match is over, we cannot score a goal



- **Value Still to Earn**

versus

- **Time Still Available**

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

Quality on Time

- Delivering the Right Result at the Right Time

Goals for this week

- 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

Warning:

After this lecture, you don't have an excuse any more !

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

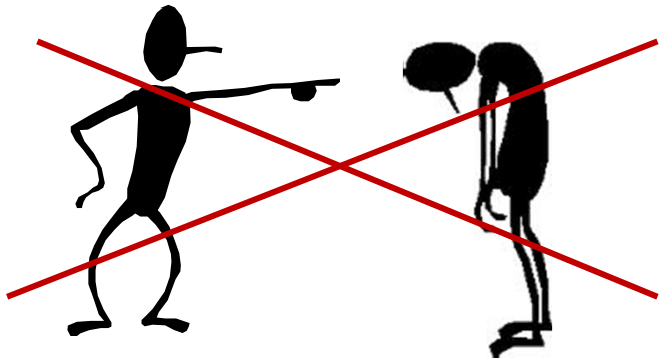
Exercise: How about your current project ?

- 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 ?

- 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

- What did you find out during this exercise ?

Who's Responsible for the Result of the Project ?



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

What could we have done to save time ?

- **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 ?**

*

Types of Project Leader

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

Quality

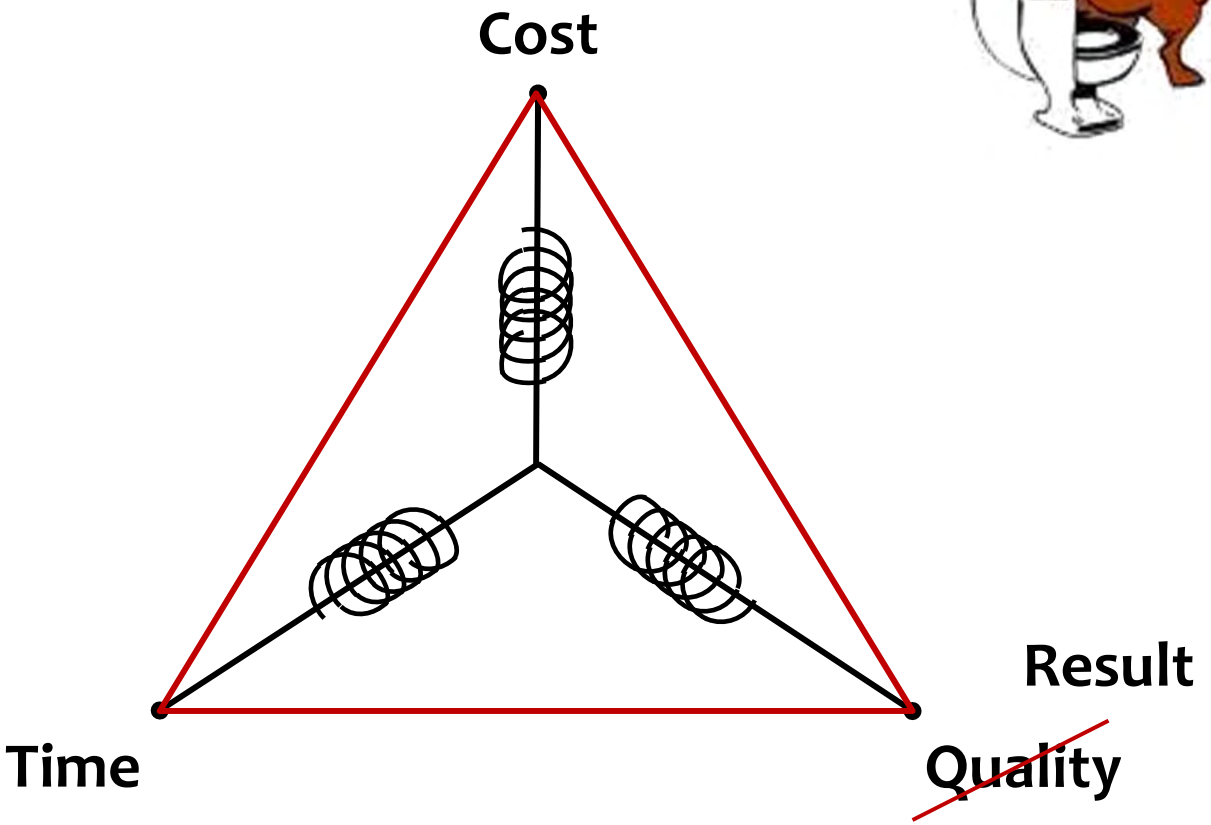
What is Quality ?

- I know it when I see it …?

- Should be *measurable*
- Should be *predictable*

- But …
  ultimately they must like it when they see it
- It must satisfy the goal

So called 'Iron Triangle'

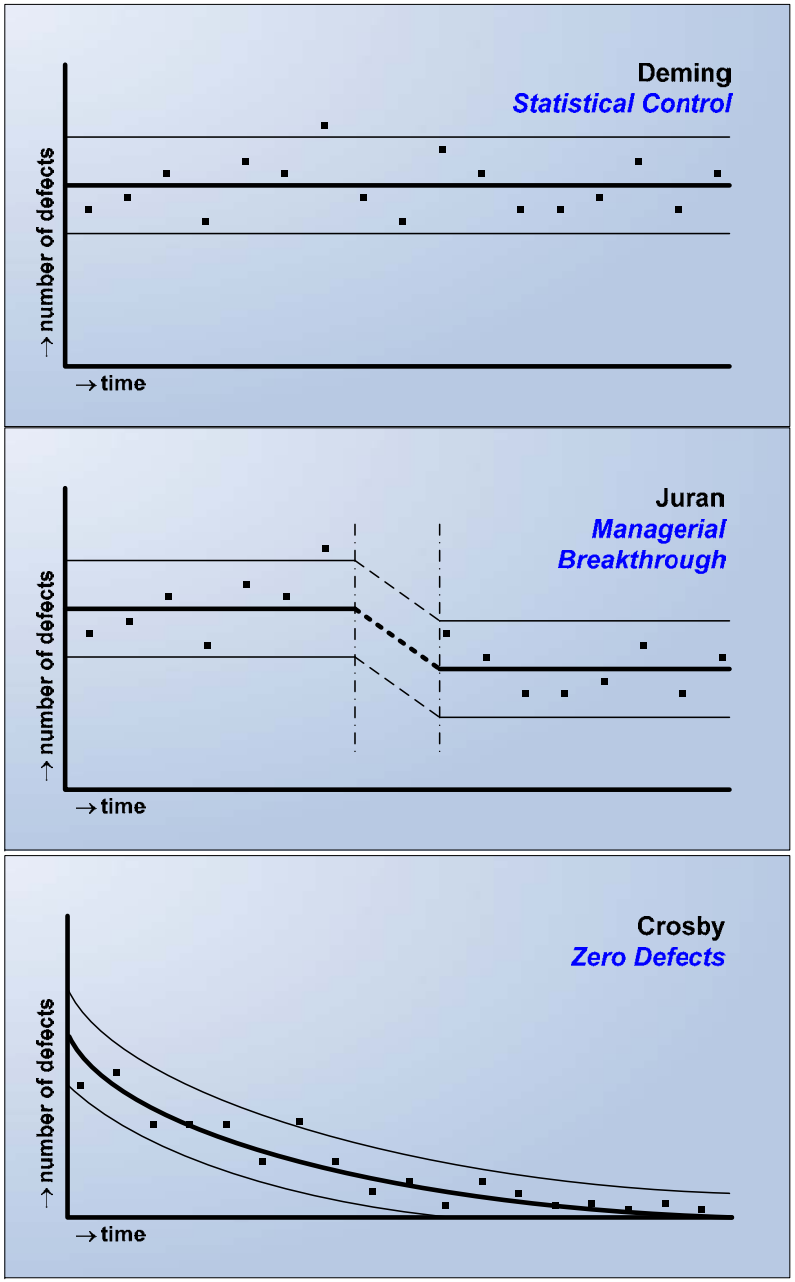


The *right* quality costs *less*

Quality guru's

- **Shewhart** - Economic Control of Quality 1931
- **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

Deming - Juran - Crosby



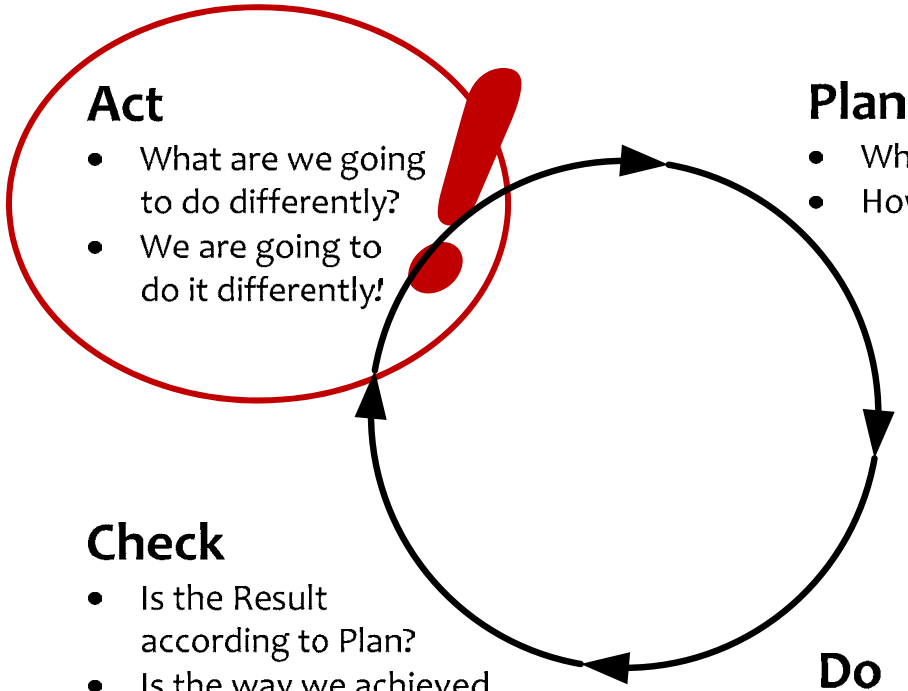
Deming

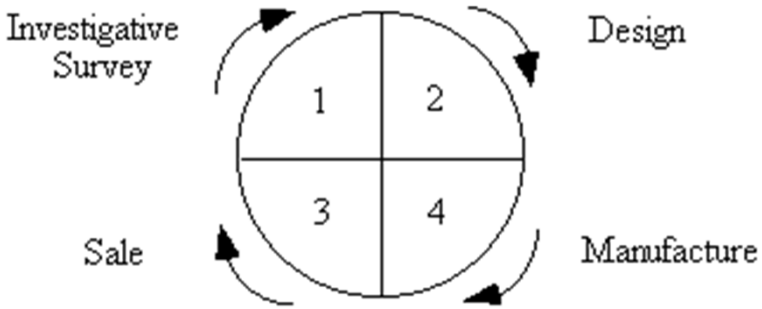
- **Quality comes not from inspection (Verification & Validation), but from *improvement of the production process***
- **Inspection does not improve quality, nor guarantee quality**
- **It's too late**
- **The quality, good or bad, *is already in the product***
- ***You cannot inspect quality into a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

→ ***People who do the work put the quality in, good or bad***

The essential ingredient: the PDCA Cycle

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





Do we deliver quality (value) ?

“We must deliver value !”

A project doesn’t deliver value

A project should create the *conditions*
for the *users* to let the quality emerge

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

Crosby: Absolutes of Quality

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

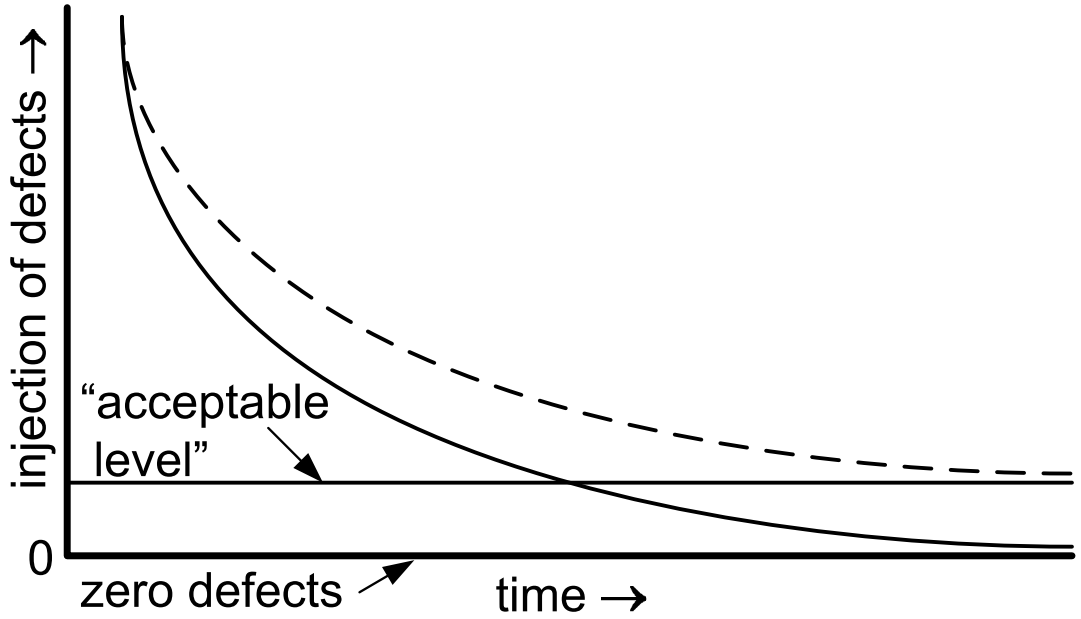
Philip Crosby, 1970

- The purpose is customer success (not customer satisfaction)

Added by Philip Crosby Associates, 2004

Is Zero Defects possible?

- Zero Defects is an asymptote



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

Philip Crosby

[Quality is Still Free]

- **Conventional wisdom says that error is inevitable**
  (一般通念では、エラーは避けられない)
- **As long as the performance standard requires it, then this self-fulfilling prophecy** (自己達成予言) **will come true**
- **Most people will say:**
  People are humans and humans make mistakes
- **And people do make mistakes, particularly those who do not become upset when they happen**
- **Do people have a built-in defect ratio ?**
- **Mistakes are caused by two factors:**
  lack of knowledge and lack of attention
- **Lack of attention is an attitude problem**

Zero Defects is an attitude

- As long as we think Zero Defects is impossible, we will keep producing defects

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

Conformance to requirements

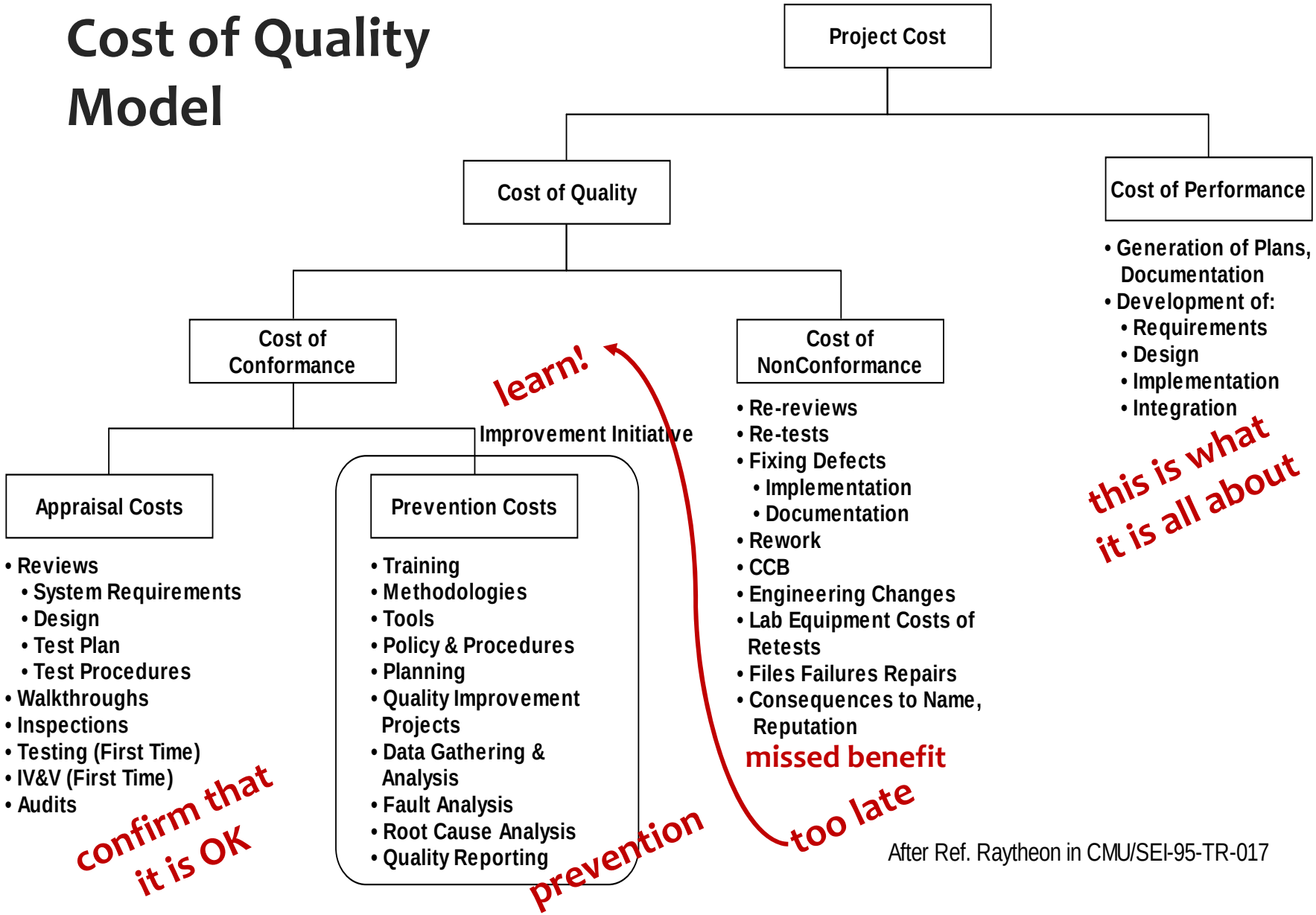
- We meet the agreed requirements

or

- Have the requirements changed to *what we and the customer really need*
- We create requirements with care and we meet them with care
- Does our management take quality seriously ?

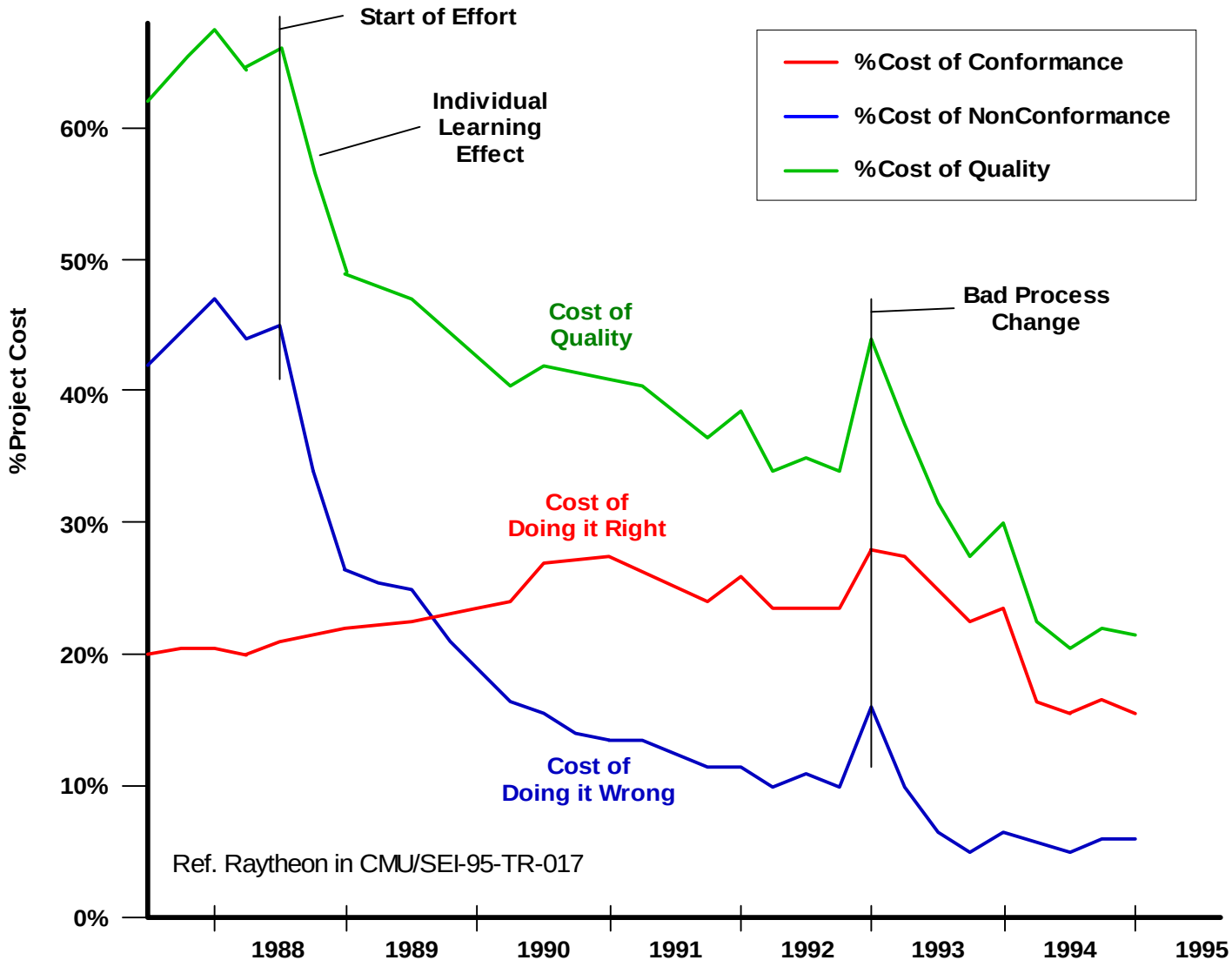
Philip Crosby

Cost of Quality Model

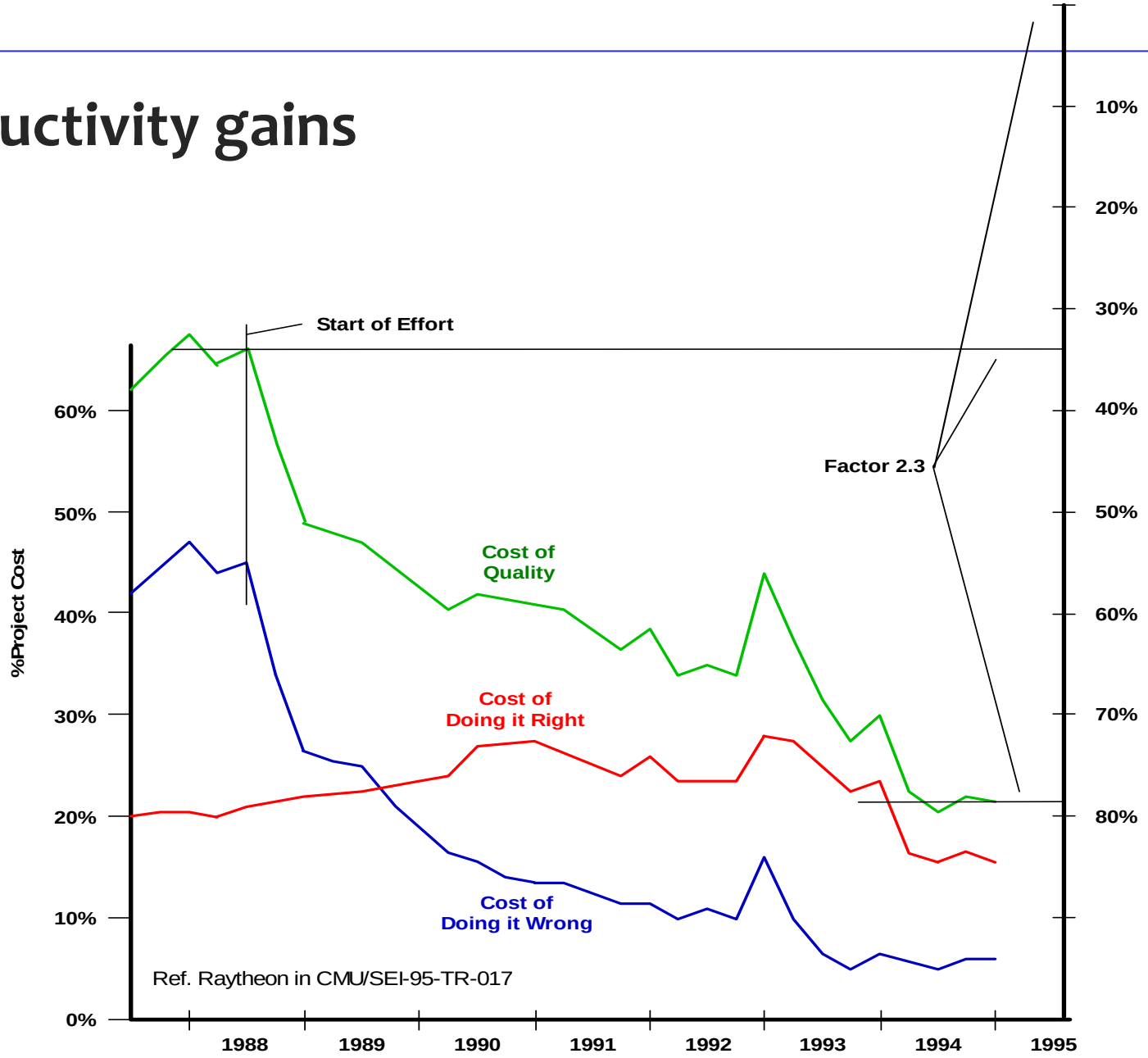


After Ref. Raytheon in CMU/SEI-95-TR-017

Cost of Quality



Productivity gains



Human Behavior

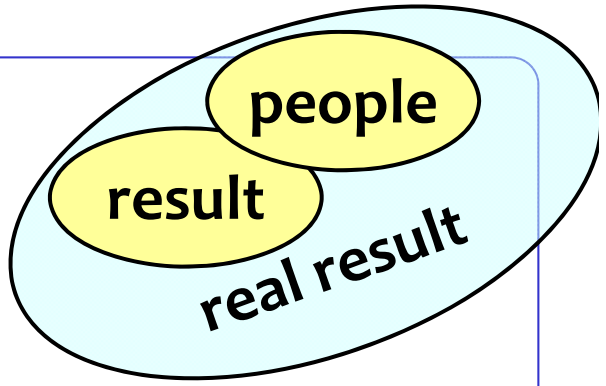
Human Behavior

ふるまい

- 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, Prince-2, as well as people in 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*

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**
  - Human error results from physiological and psychological limitations of humans, most of which are known

Discipline 規律

- **Control of wrong inclinations** （傾向）
- **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…)

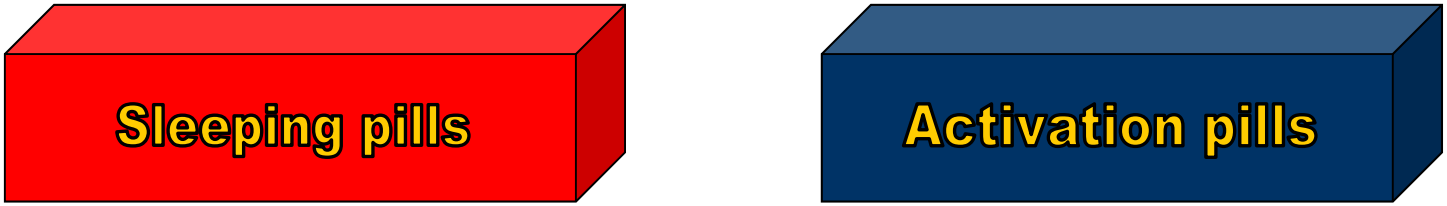
→ **Making mistakes** (provides short window of opportunity)

→ **Openness** (management must learn how to cope)

Intuition 直観

- Makes us 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

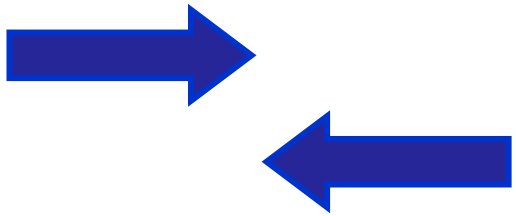
Is intuition wrong, or is the design wrong ?



Communication

- **Talking as near as possible along each other**

To each other



Along each other

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

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)
- 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!

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

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

Excuses, excuses, excuses …　言い訳言い訳

- **We have been thoroughly trained to make excuses**
- **We always downplay our failures**
  私たちは失敗を軽視してしまいがちです
- **At the Fatal Day, any excuse is in vain** (無駄に)**: we failed** (失敗)
- **Even if we "couldn't do anything about it"**
- **Failure** (失敗) **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**

痛みなくして得るものなし

Mistakes, unnecessary things

- **What was the last time you made a mistake ?**
- **What was the last time you did something unnecessary ?**

(むだ)

- **Did you talk with others about it ?**
- **Did you learn from it ?**
- **What did you do about it ?**

Ignore the first reaction

- If you show something is wrong
- Even if the person agrees, first you'll get:

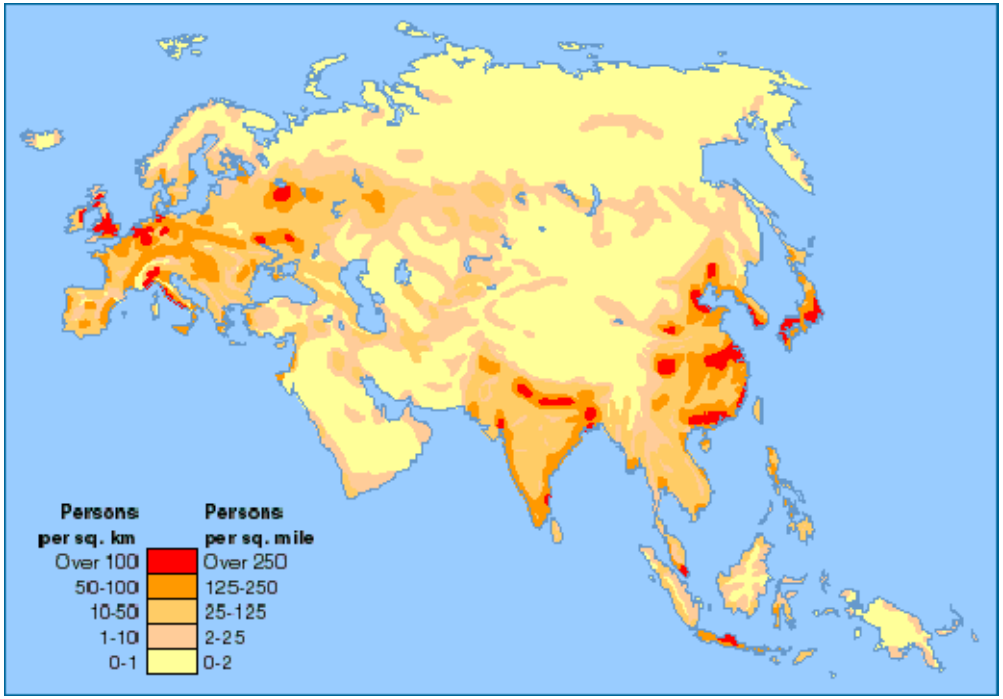
  "Yes, but ... bla bla" or,
  "That's because ... bla bla"

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

Logical thinking is not always better

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

Is Culture an Issue ?



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

Culture

- **Culture: Ingrained**（染み付いた）**customs** (習慣)
  - 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 know 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 is not obvious at all; neither is theirs
- **Still we judge others through our own cultural spectacles,**
  whether we like it or not

Cultural elements

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	• … • … • … ?

Things I heard

- **Authority**
  - Boss is always right
  - Teacher is always right

  They are just doing their best. Lot of experience. Are they perfect ?
- **Group is responsible**
  - No personal responsibility ?
  - Should we hide for responsibility ?

Things I heard (2)

- Losing ‘face’
  - Why not challenge? As long as you don’t get personal
- Cannot say ‘No’
  - How do you say ‘no’ ?

    (in Holland we say: “Yes, but …”)
- Is that clear? - Yes
  - ‘Yes’ isn’t always ‘Yes’, but can you say ‘No’ ?
  - If you don’t understand:
    - Is the teacher unclear ?
    - Am I stupid ?

The boss is always right

- **Is the boss always right ?**

- **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 ?**

Is culture a risk for projects ?

- **Let's talk about it**

*

Preparation for tomorrow

- **Make a list of the top-3 requirements of your current or last project**
- **Think also about:**
  - **Did you achieve these requirements ?**
  - **On time ?**
  - **Was the project successful ?**
  - **Was the customer happy ?**

Day 2

Did you prepare ?

- **The top-3 requirements of your current or last project**
  - **Did you achieve these requirements ?**
  - **On time ?**
  - **Was the project successful ?**
  - **Was the customer happy ?**

Business Case

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
  - Better environment
- In short: *Adding Value*
- *Return on Investment*

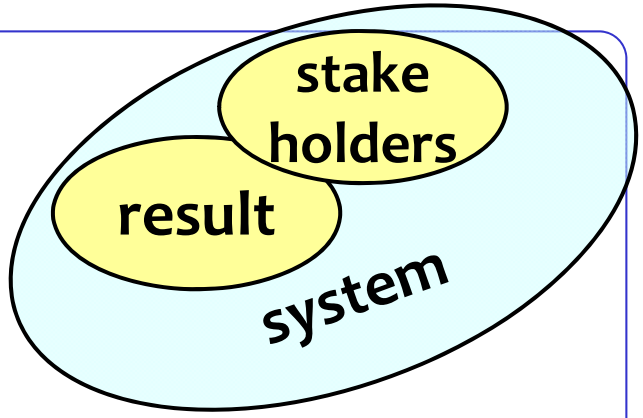
How many Business Cases ?

- There are usually at least two Business Cases:
  - Theirs
  - Yours
- How many Business Cases are there in your project ?

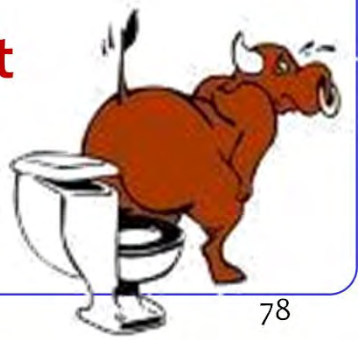
- Every Stakeholder has his own business case

Stakeholders

Stakeholders

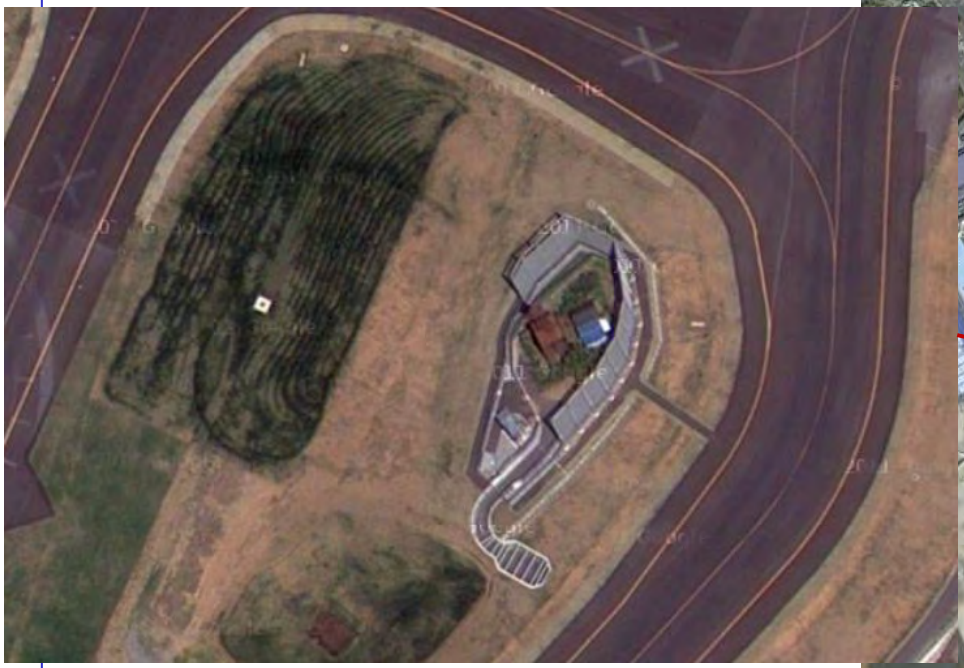


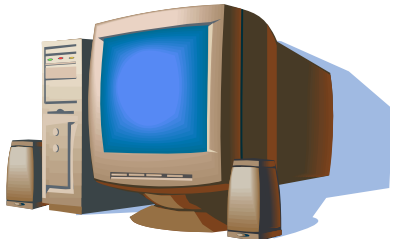
- Every project has some 30±20 Stakeholders
- Stakeholders have a stake (interest - 利害関係) in the project
- The concerns of Stakeholders are often contradictory
  - Apart from the Customer *they don't pay*
  - So they *have no reason to compromise !*
  - Finally, *we all pay*
- Some Stakeholders are victims of the project
  - They want the project to fail
- Project risks, happening in almost every project
- No excuse to fail !



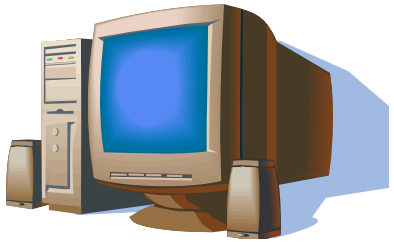
Victims can be a big Risk

Victims:
Narita Airport





Their old system (cash cow)



Our new system

We need the test-system of the previous supplier

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*
  - This is what you'll get, if you let us continue

- The set of Stakeholders doesn't change much
- Do you have a checklist of possible Stakeholders ?
- What will happen if you forget an important Stakeholder ?

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?

Requirements

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

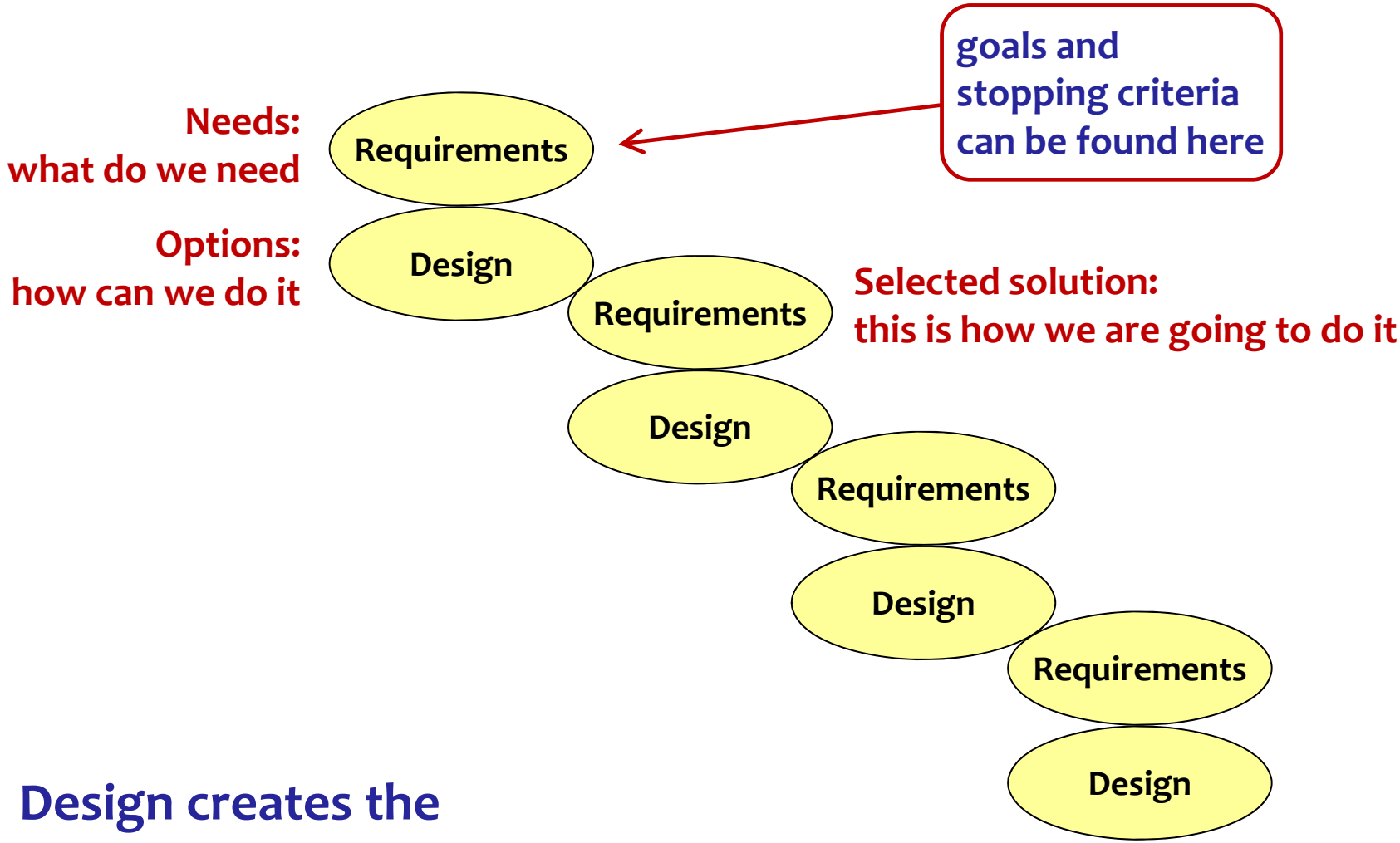
- What Wish Specification did *you* receive ?
  - Write it down
- How did you receive it ?
- From whom ?
- What did you do with it ?

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

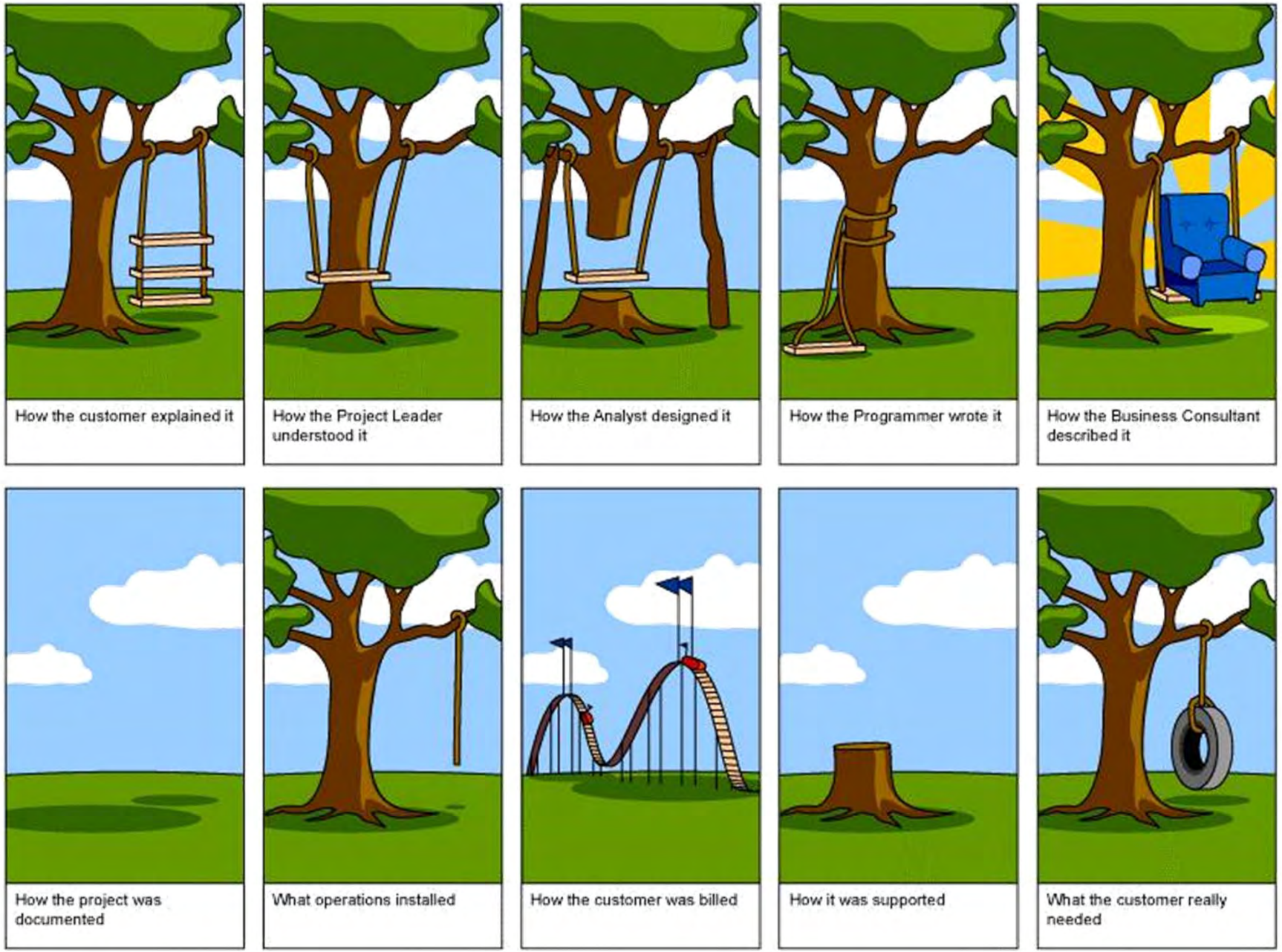
Recent project

- 1600 requirements ‘big design up front’: just deliver
- ‘1600 requirements’ were solutions to an undefined problem
- No clear problem definition
- No clear goals
- No stopping criteria
- Customer hasn’t got anything useful yet (after 2 years)
- Will they be successful by the end of the year ?

No Design in the Requirements, but ...



Design creates the Requirements for the next level


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

We're Agile and we're using Scrum

- Oh dear !
- Dances and rituals
- Demo's
- IT people think the're doing a great job …
- Customer has nothing

Wasting time everywhere

Delivery Strategy Suggestions (Requirements)

- **What we deliver will be used by the appropriate users immediately, within one week not making them less efficient than before**
- **If a delivery isn't used immediately, we analyse and close the gap so that it will start being used** (otherwise we don't get feedback)
- **The proof of the pudding is when it's eaten and found tasty, by them, not by us**
- **The users determine success and whether they want to pay**
  (we don't have to tell them this, but it should be our attitude)

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)

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
  - What is not allowed
  - These requirements are basically non-negotiable

* Better not use *non-functional* requirements !

Performance Requirements

- How secure
- How dependable
- How well usable
- How well maintainable
- How well portable
- How well ….
- How fast
- How big
- How nice to see
- How nice to use
- How accurate
- How reliable

Remember: Projects are about improving something !

Improving something

- All functionality we produce *does already exist*
- The real reason for running our projects is creating *something better*
- Improvement of value, productivity, success, happiness for our customers through users

Improving on *existing* qualities

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

after FIRM / Gilb 2005

Constraints

Constraints are harder than the other requirements

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

5 times "Why?"

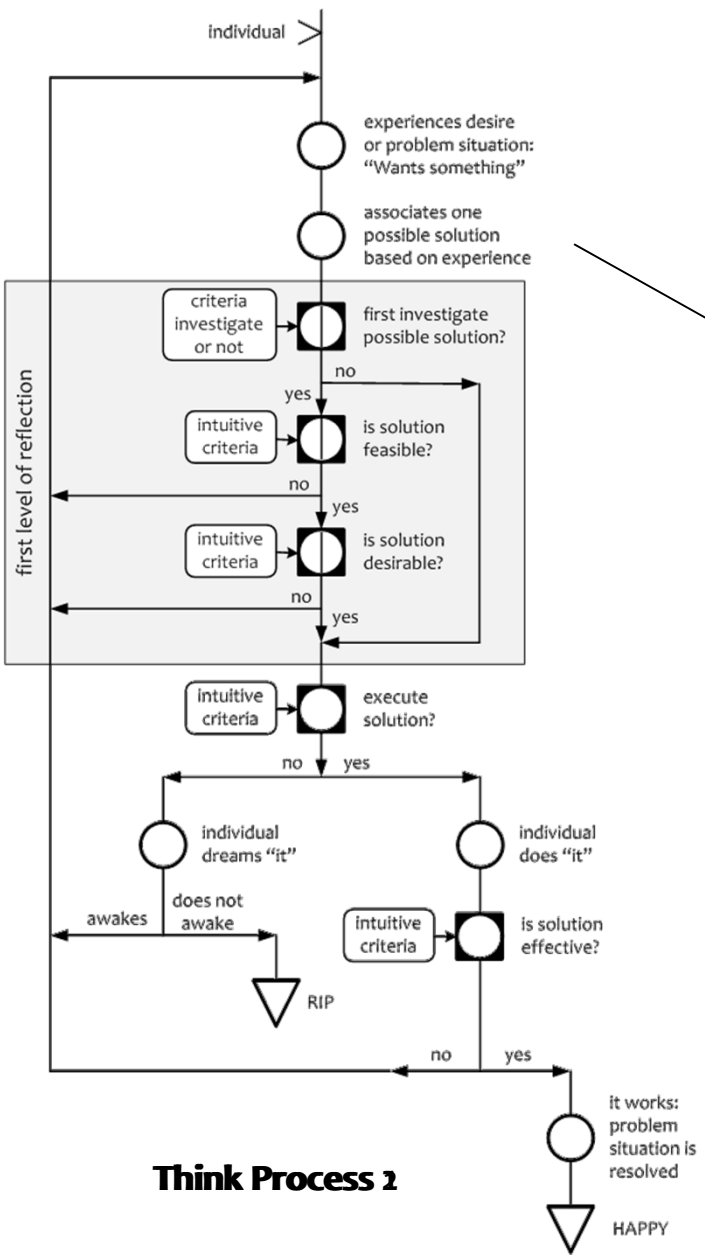
First develop the problem
interdisciplinarily,
then develop the solution
and then the implementation

- **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

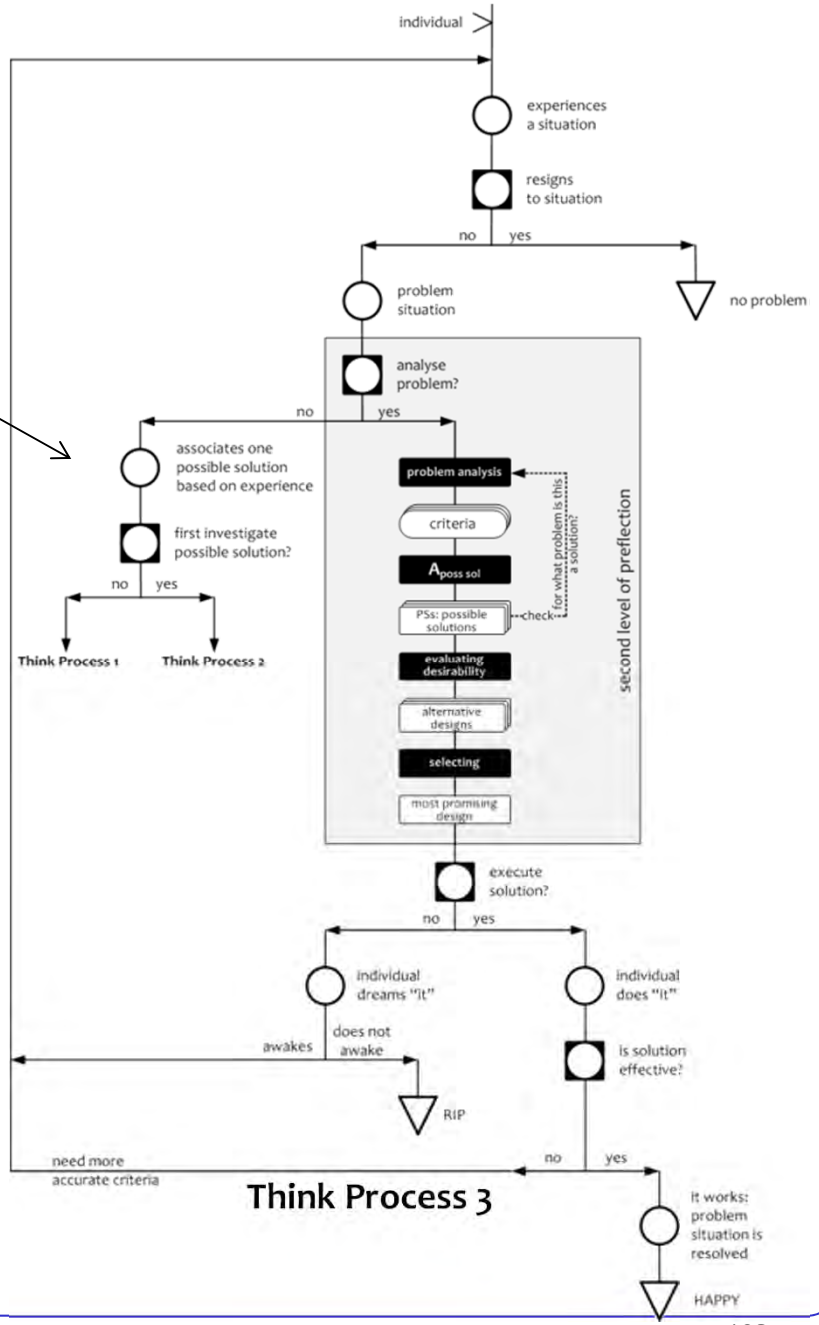
I would like to have a lot of money

- I would like to have a lot of money
- Why ?
- Then I could have and do everything I want
- Why ?
- Well, then I would be happy
- Ah: You're not after money; you want happiness !

Think Process for Problem Solving



Think Process 2



Think Process 3

Ref. Malotaux – Van der Goot

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?

Do we have a Good Requirements checklist for Japanese ?

Weak words

Examples:

accommodate	can	minimize
adequate	capability to	normal
and/or	easy	not limited to
as a minimum	effective	provide for
as applicable	etc.	robust
as appropriate	if practical	sufficient
be able to	maximize	support
be capable of	may	when necessary

How about Japanese requirements ?

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

Why quantifying ?

- **The most important things in life cannot be measured, the more important they are, the less you can measure them**
  Ron Baker
- **Book:**
  How to measure Anything - *Finding the Value of Intangibles in Business*
  Douglas Hubbard

Tom Gilb:

The fact that we can set numeric objectives and track them is powerful, but in fact 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

Somebody said the requirements should be *SMART*

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

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

Requirements with Planguage

ref Tom Gilb

Specific

Measurable

Definition:

RQ27: Speed of Luggage Handling at Airport

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

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

Attainable

Benchmarks (Playing Field):

Past: 2 min [minimum, 2009], 8 min [average, 2009], 83 min [max, 2009]

Current: < 4 min [competitor y, Jan 2010] ← <who said this?>, <Survey Feb2010>

Record: 57 sec [competitor x, Jan 2010]

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

Realizable

Time

Requirements:

Must: < 10 min [99%, Q4] ← SLA

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

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

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>

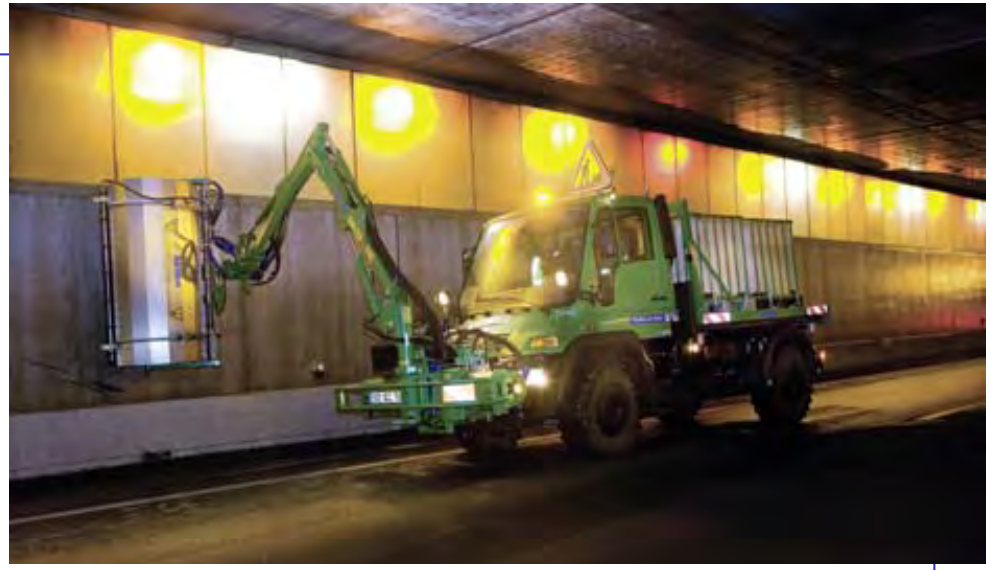
Reliability

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
  - Telephone exchange (no dial tone) < 5 min per year (99.999%)
  - Snow on the road

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

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

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
- Goal

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

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

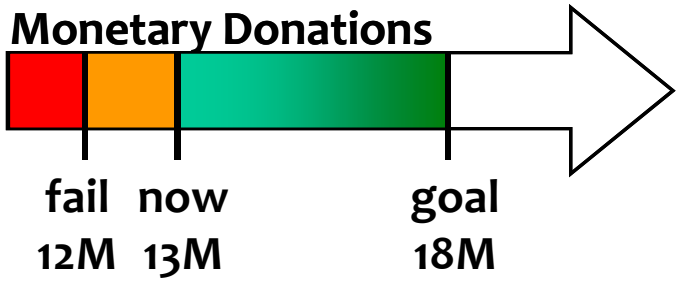
How to specify results
How to select
the right solution ?

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

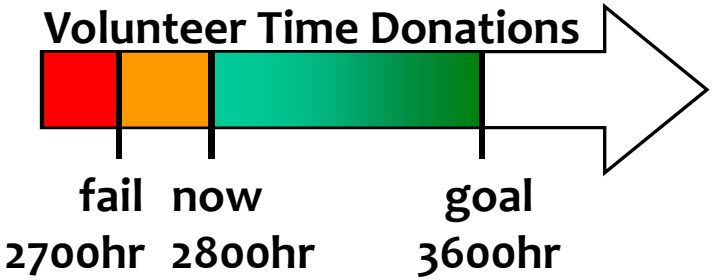
Fail 12M

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

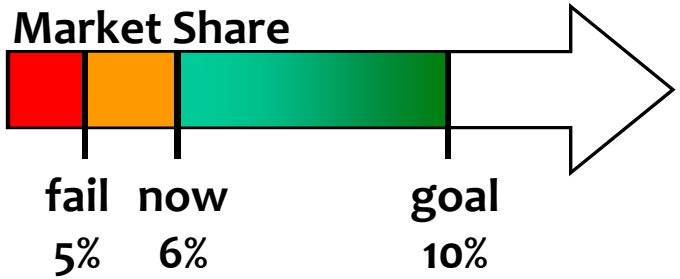
Fail 2700 hr

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

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

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

Impact Estimation principle

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

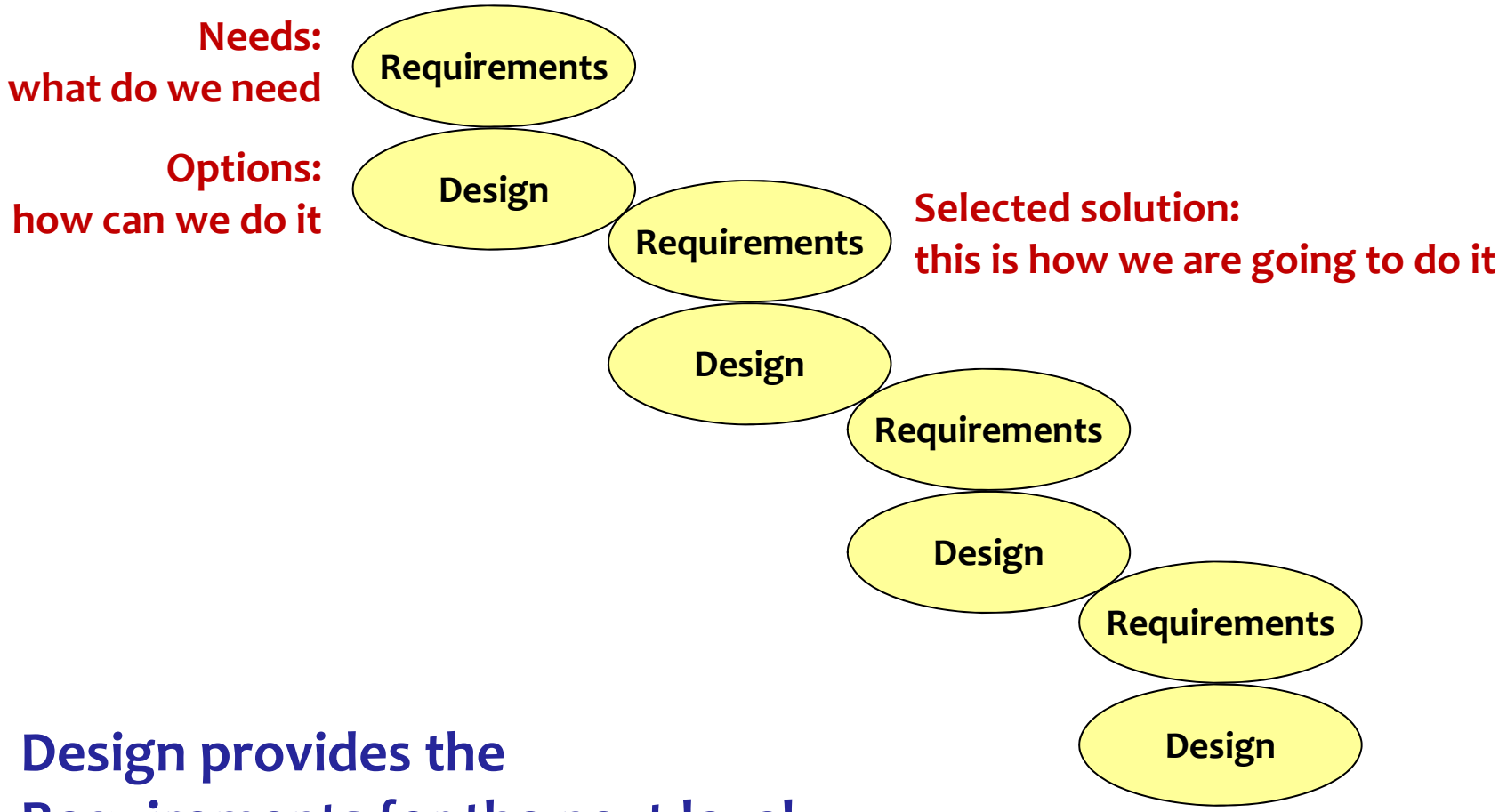
At what cost ?

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	

No Design in the requirements, but ...

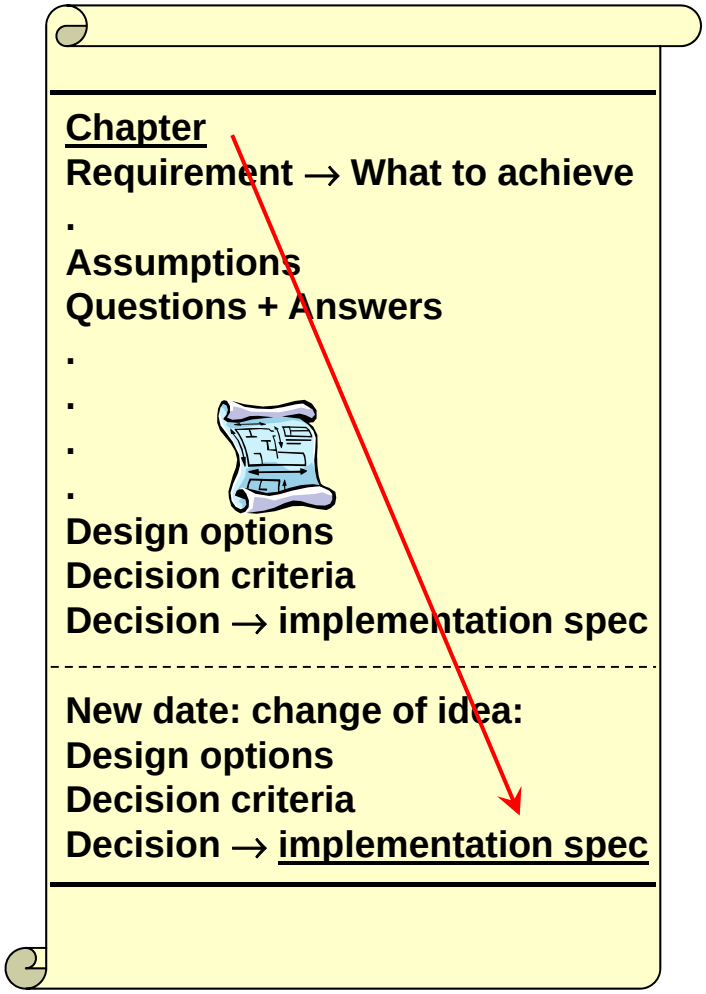


Design provides the Requirements for the next level

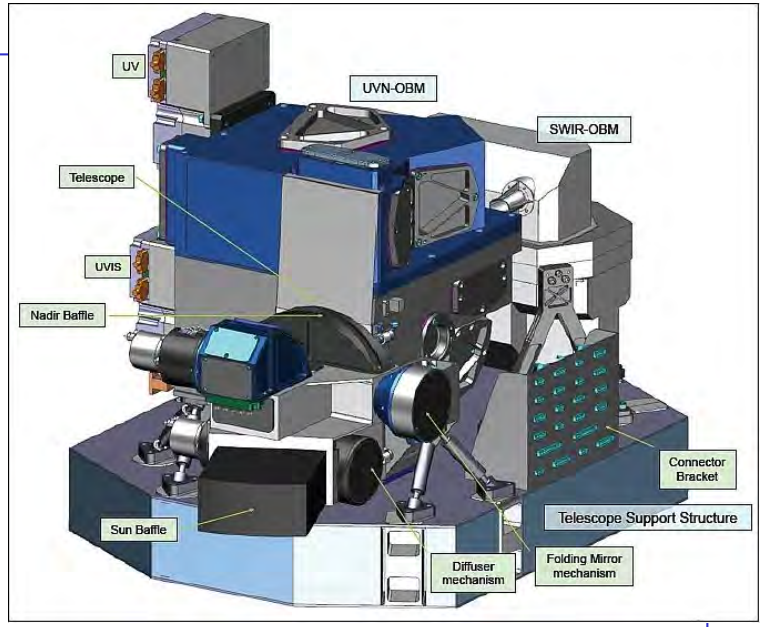
DesignLog

(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



Earth Observation Satellite
地球観測衛星



- Earth observation instrument
- Launch 2014
- ESA imposed project model - super waterfall: 8 years project
- Some 40 people, several sub-groups
- A lot of sub-contractors in different countries, lot of politics
- Very experienced Systems Engineers
- Using quantified requirements routinely

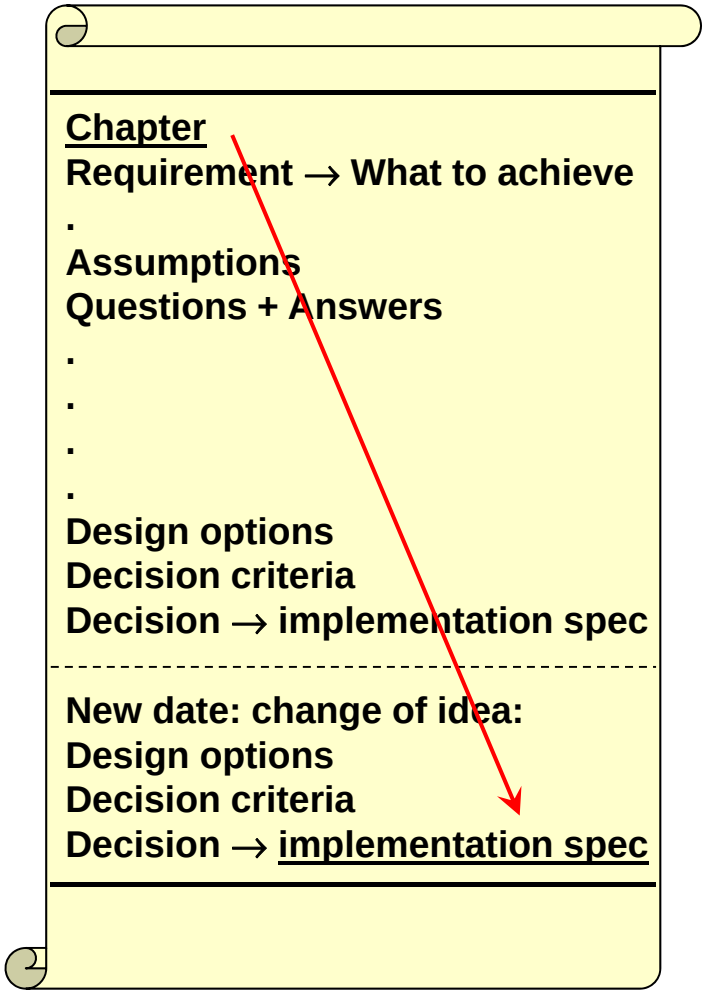
Summary of requirements for the ozone products:

- **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)

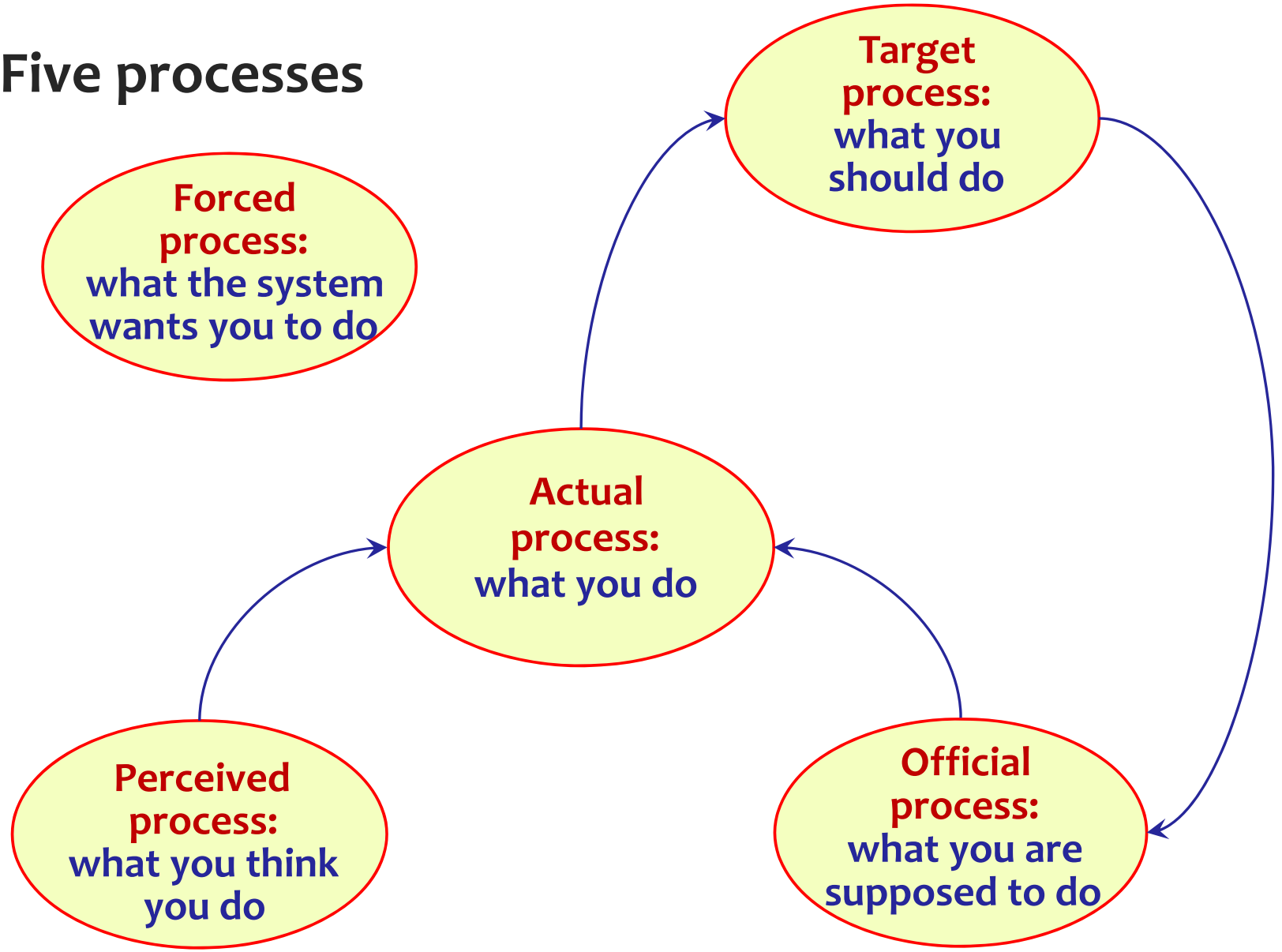
ProcessLog

(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



Five processes



Preparation for tomorrow

- Make a list of the work you have to do the coming weeks
- Make a list of what else you will do
- Write down how much time you have available for the work
- Mark on the list which parts of the work you will have to do the first week
- Check how much time you have available for this work
- Will you be able to complete all the work you think you have to do the coming week ?
- If yes: How do you know ?
- If no: Why not ?

Day 3

Did you prepare ?

- List of the work you have to do the coming weeks
- List of what else you will do
- How much time you have available for the work
- Marked the parts of the work you will have to do the first week
- How much time you have available for this work
- Will you be able to complete all the work you think you have to do the coming week ?
- If yes: How do you know ?
- If no: Why not ?

Estimation

Is it difficult to be on time ?

- Did anyone miss a plane, a train, an appointment ?
- What did you feel ?
- Why did it happen ?
- Did it happen again ?
- Was your recent project on time ?

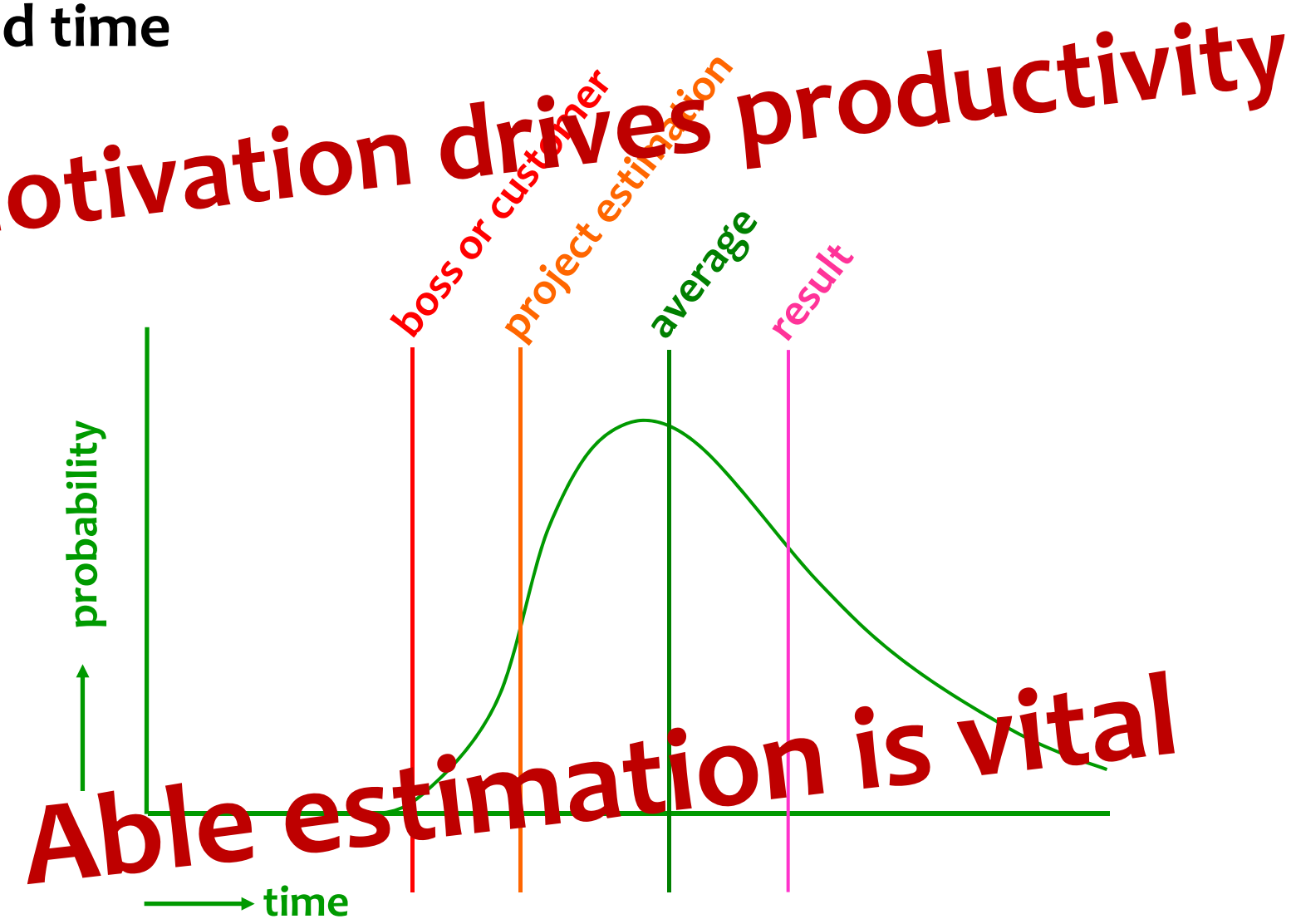
If our previous project was late,
our current project will also be late

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

Lead time


Motivation drives productivity
boss or customer
project estimation
average
result
probability
time
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

How many seconds would you need to complete this Project?

Is this what you did?

Defect rate

- **Before test ?**
- **After test ?**

Alternative Design (*how* to solve the requirement)

Another alternative design

What was the real requirement ?

Assumptions, assumptions ...

Better assume that many assumptions are wrong.

Check !

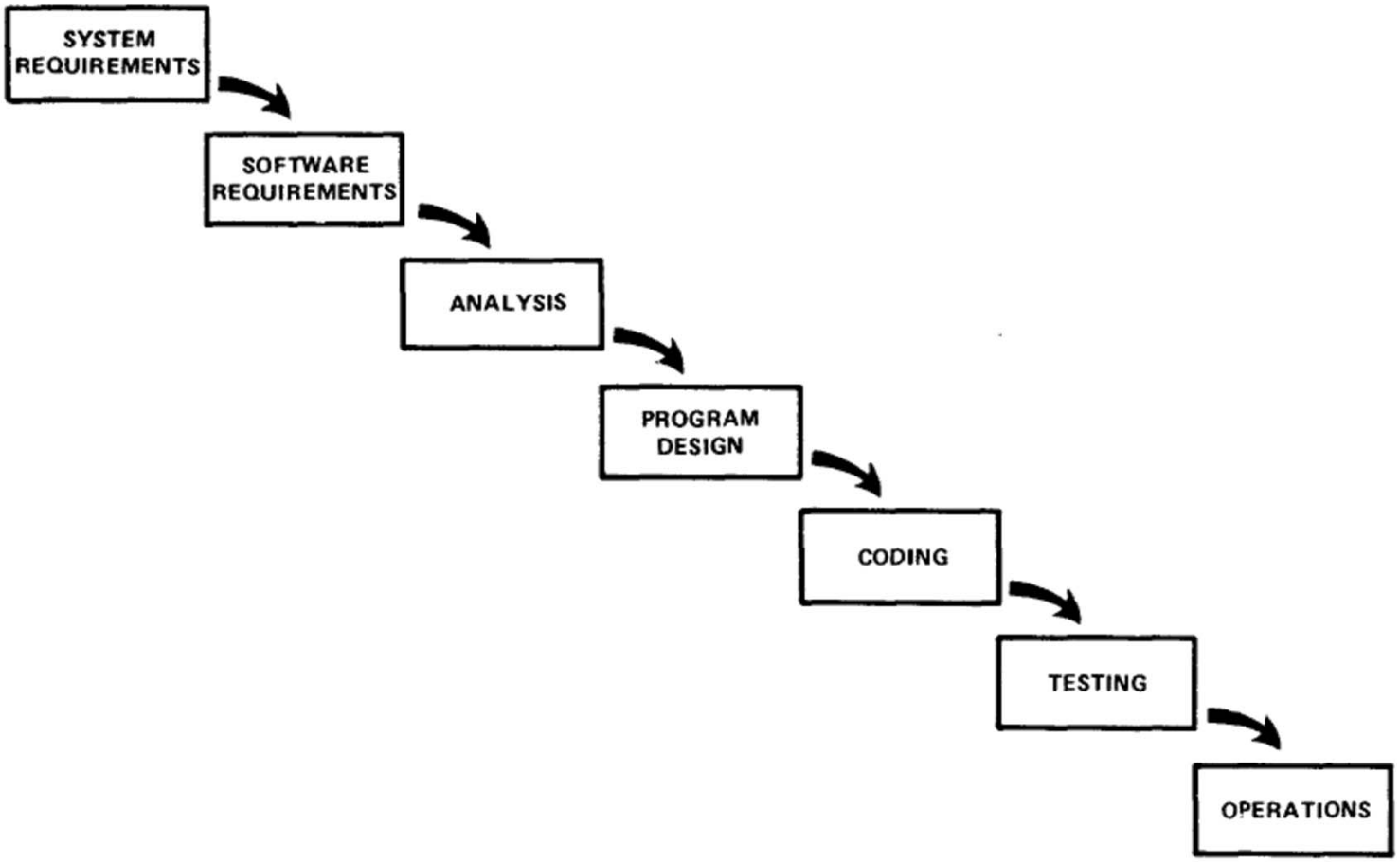
Elements in the exercise

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

Project Life Cycles

Waterfall ?

Winston Royce 1970

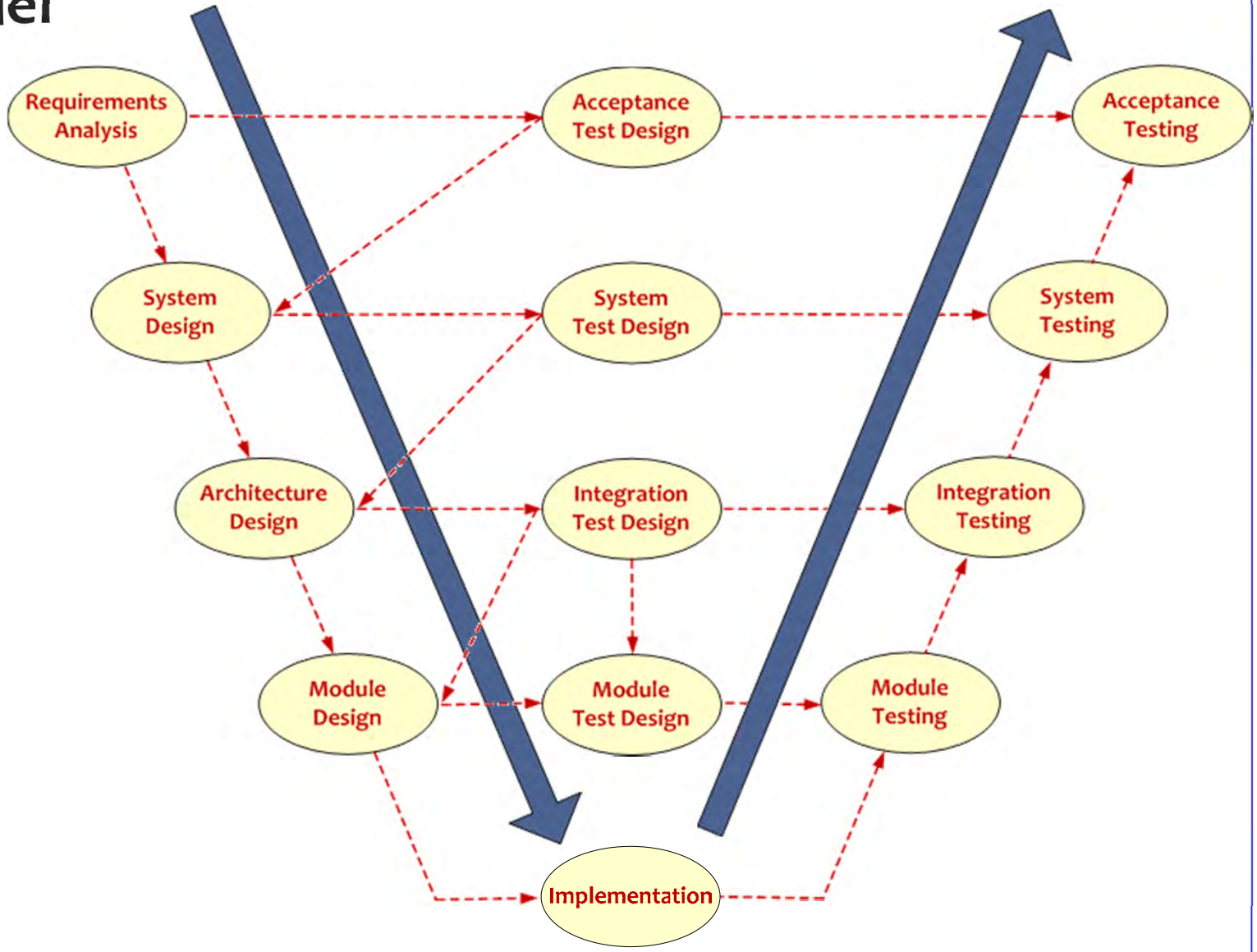


When can we use waterfall ?

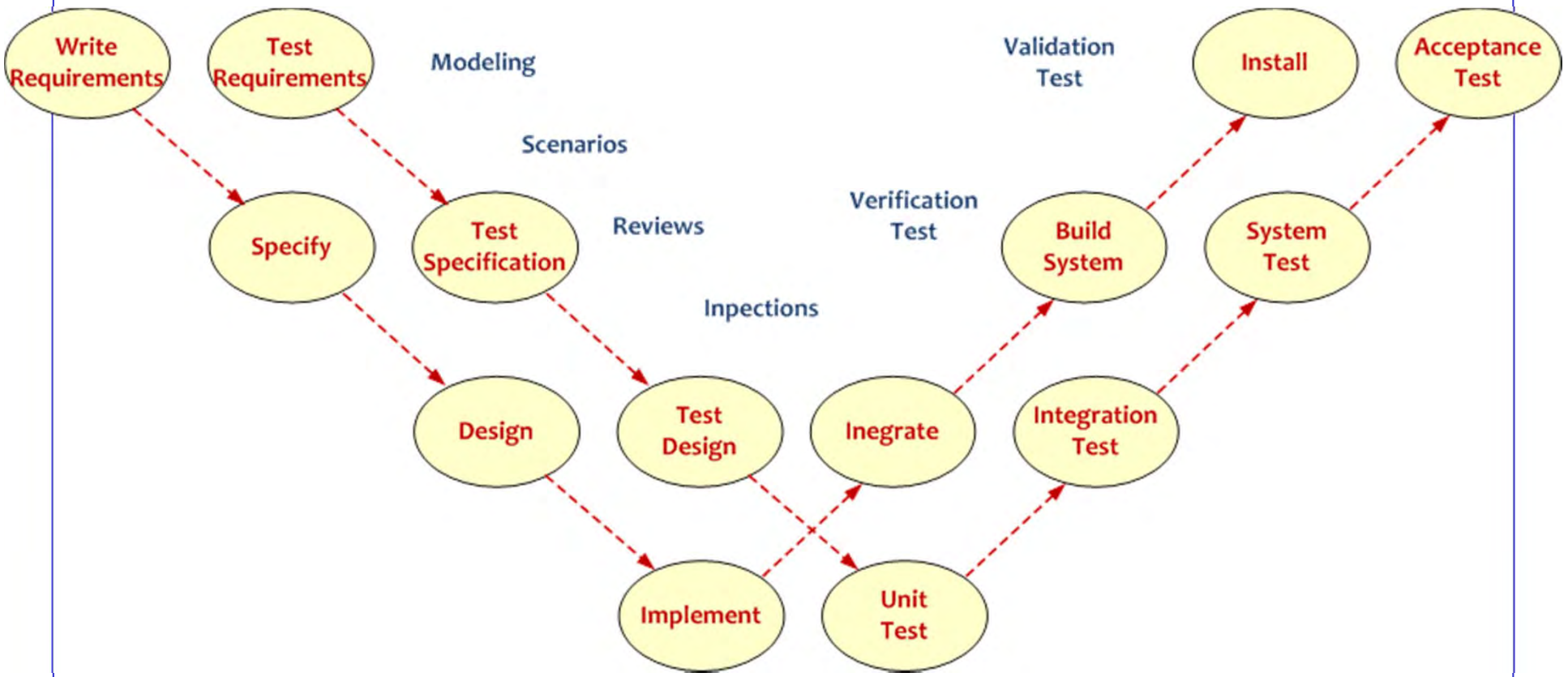
- Requirements are completely clear, nothing will change
- We've done it may times before
- Everybody knows exactly what to do
- We call this *production*

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

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

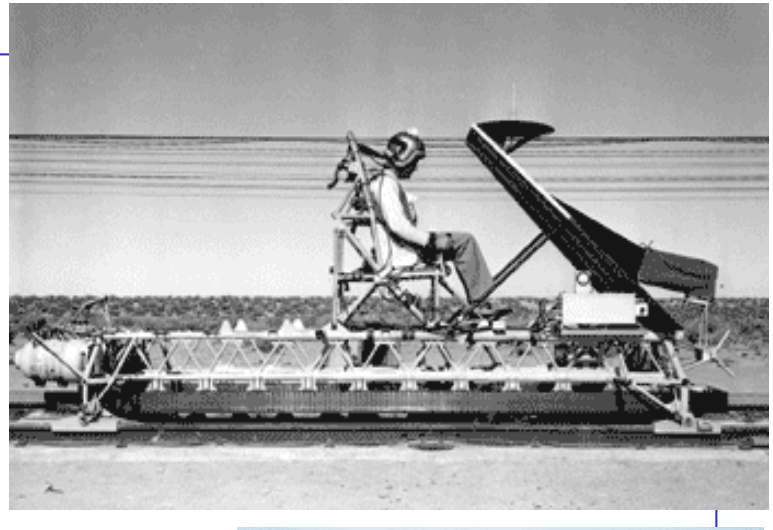

Write Requirements
Test Requirements
Modeling
Validation Test
Install
Acceptance Test
Scenarios
Specify
Test Specification
Reviews
Verification Test
Build System
System Test
Inpections
Design
Test Design
Inegrate
Integration Test
Implement
Unit Test

All Models are wrong

Some are useful

Evolutionary Principles

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* (fool-proof - ポカヨケ)

First Think, Then Do

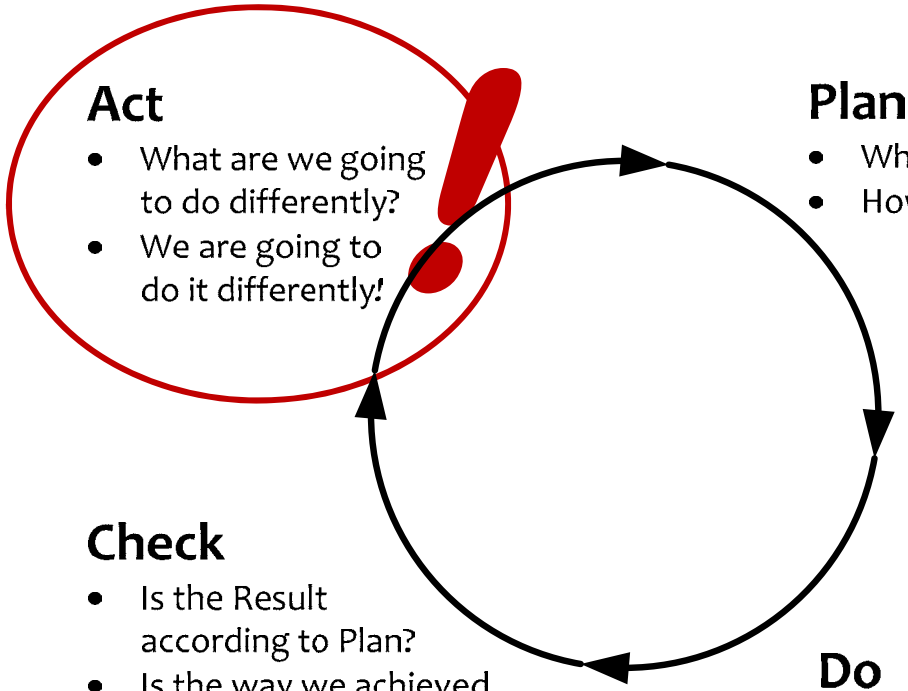
Insanity (狂気) **is doing the same things over and over again and hoping the outcome to be different** ***(let alone better)***

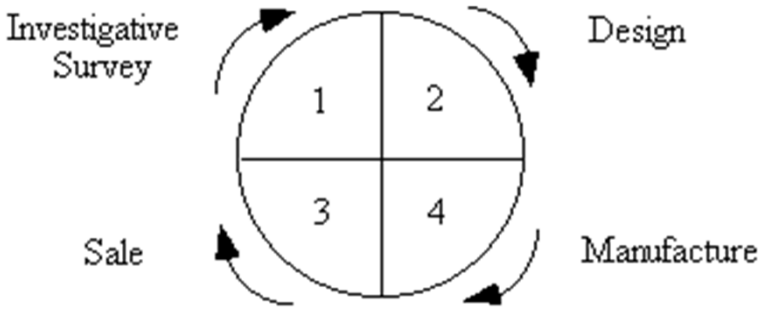
Albert Einstein 1879-1955, Benjamin Franklin 1706-1790, it seems Franklin was first

- **Only if we change our way of working, the result may be different**
  - **Hindsight** (事後考察) **is easy, but *reactive*** (事後の反応)
  - **Foresight** (事前考察) **is less easy, but *proactive*** (事前の対策)
  - **Reflection** (起きた後で考える) **is for hindsight and learning**
  - **Preflection** (起きる前に考える) **is for foresight and prevention** (予防)
- **Only with prevention we can save precious time**
- **This is used in the Deming/Plan-Do-Check-Act cycle**

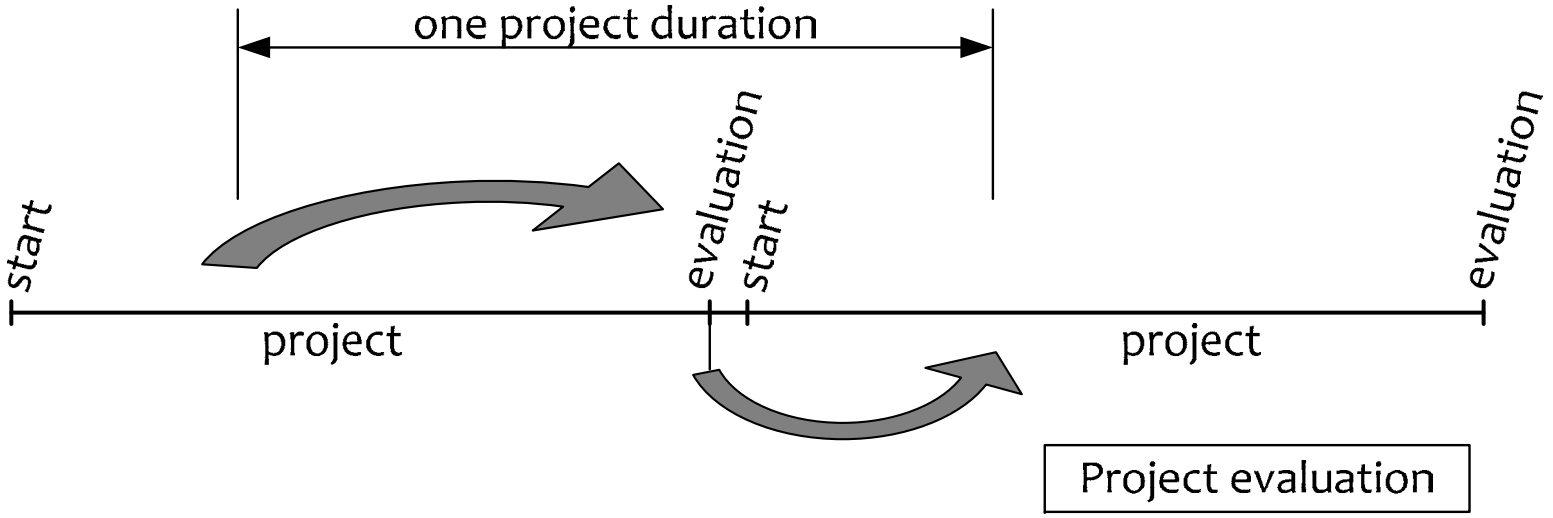
The essential ingredient: the PDCA Cycle

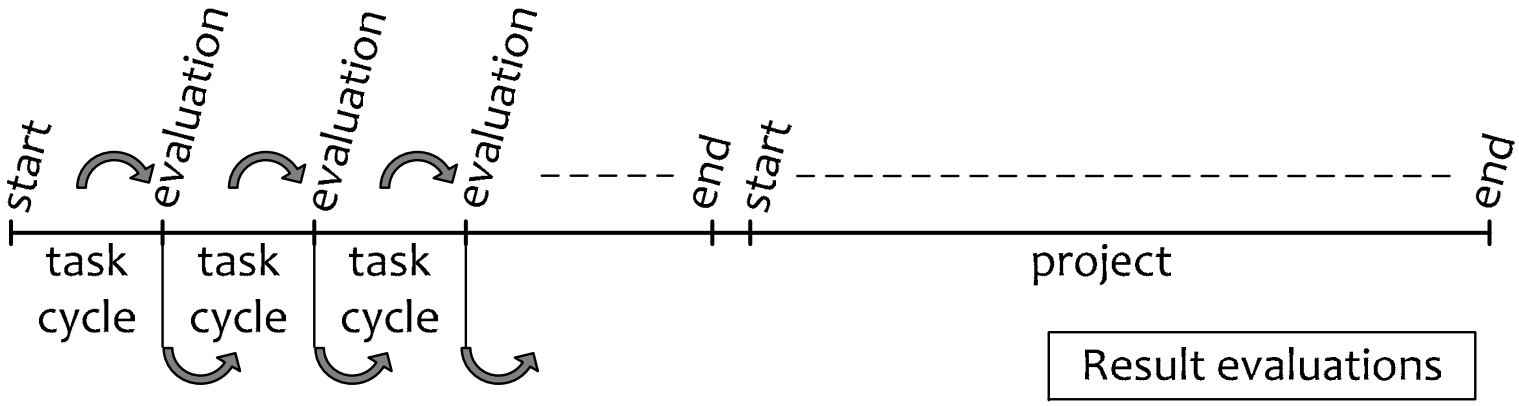
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)



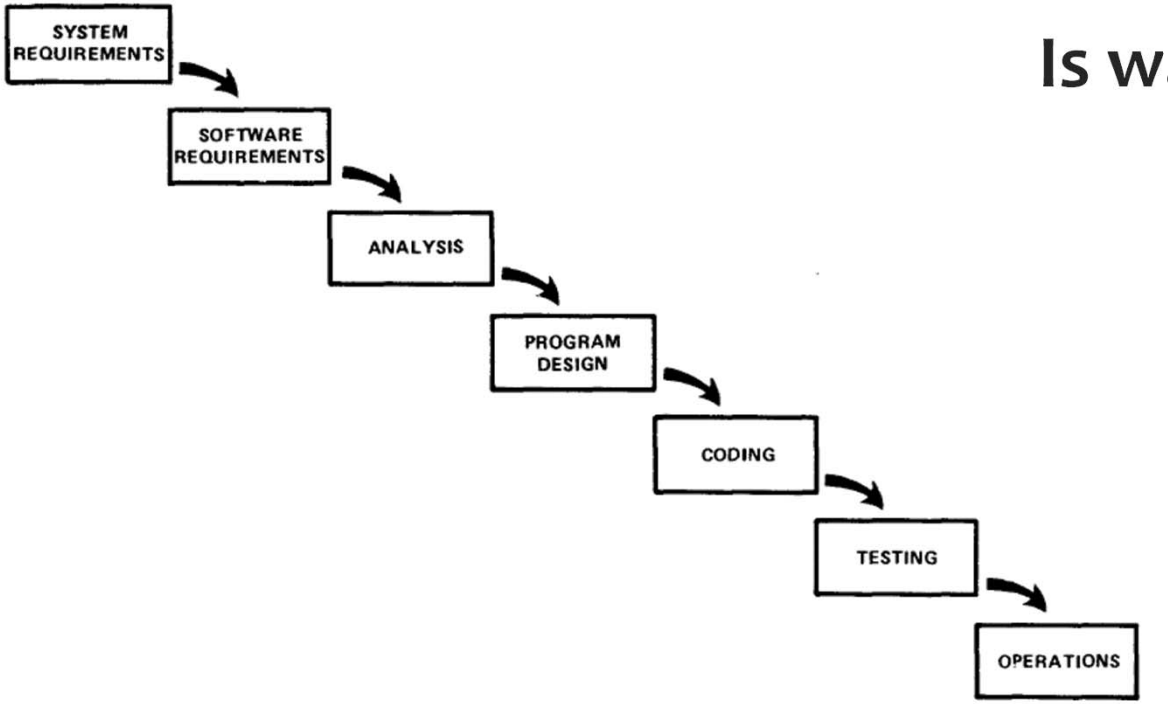


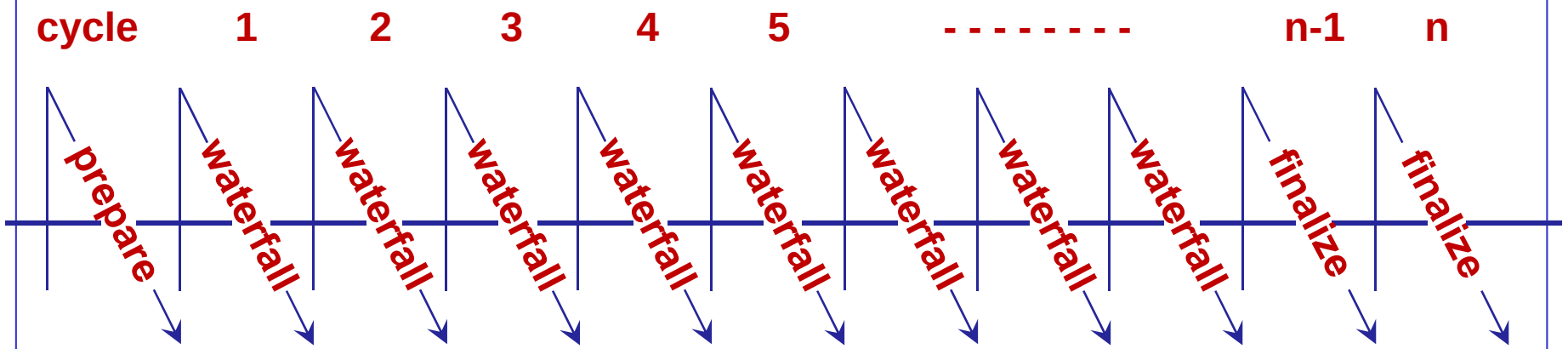
Project evaluations



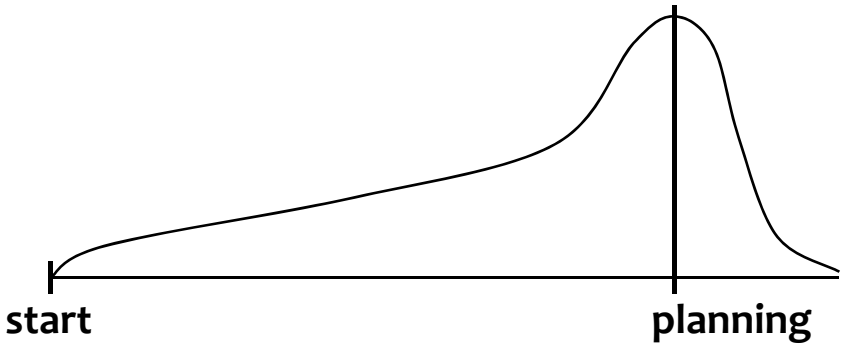


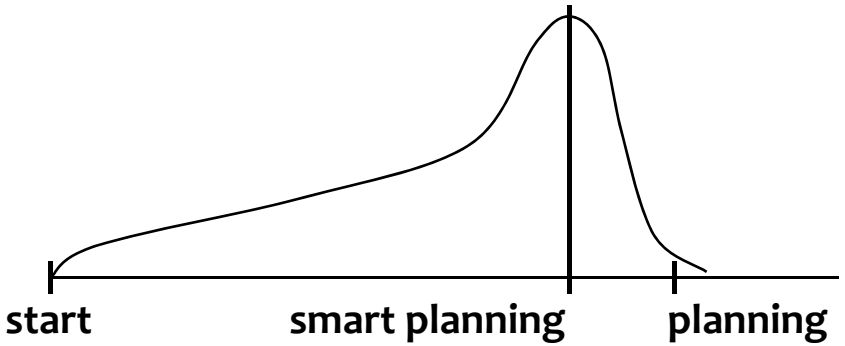
Is waterfall wrong ?

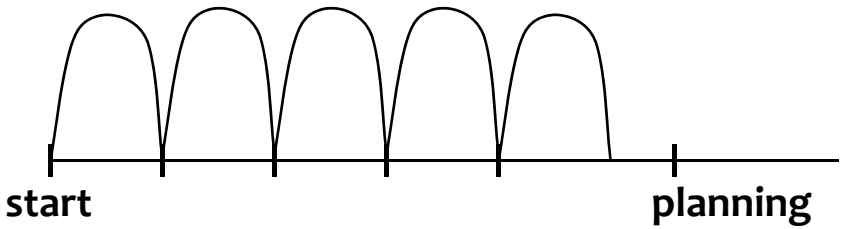




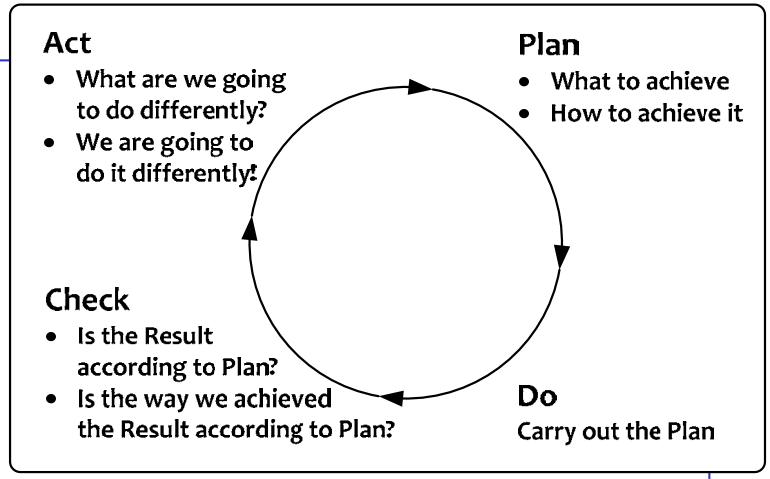
Development cycles







Knowledge
how to achieve the goal



If we

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

then we can

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

and we can

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

Known for decades

Do we still have to talk about this ?





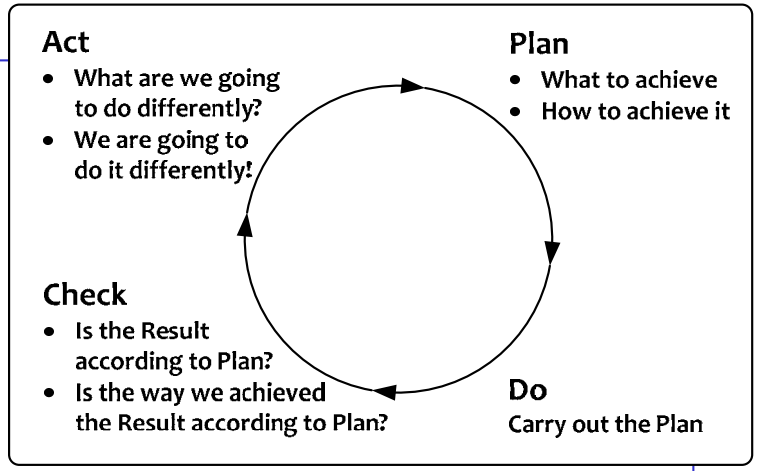




- **Benjamin Franklin** (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- **Henry Ford** (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow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

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

Things to consider

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

Evo Project Planning

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

Efficiency of what we do

Effectiveness of what we do

What will happen and what will we do about it ?

Evolutionary Project Planning
prevention is better than cure

Did you prepare ?

- List of the work you have to do the coming weeks
- List of what else you will do
- How much time you have available for the work
- Marked the parts of the work you will have to do the first week
- How much time you have available for this work
- Will you be able to complete all the work you think you have to do the coming week ?
- If yes: How do you know ?
- If no: Why not ?

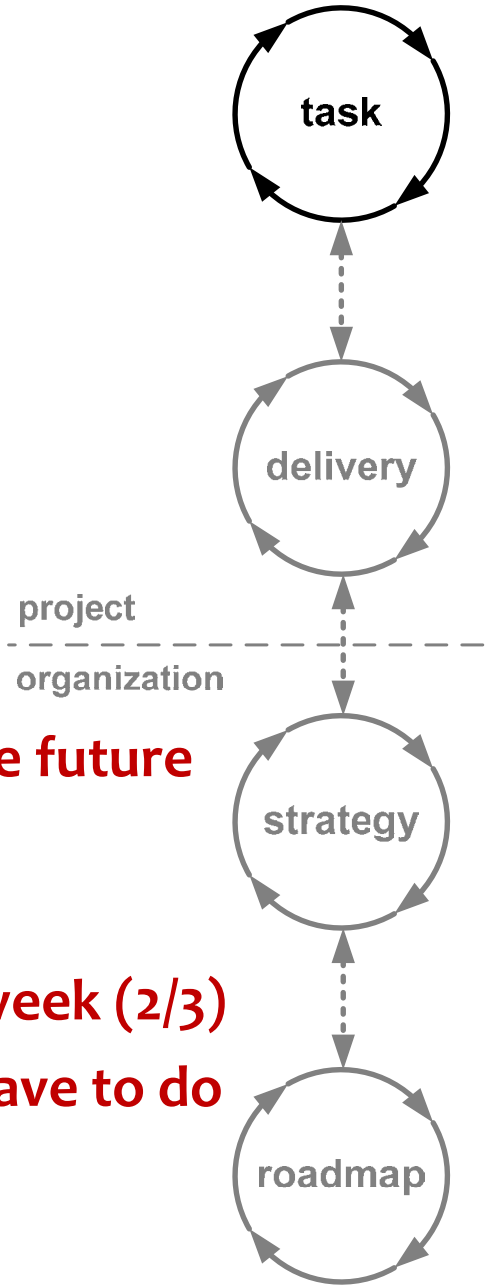
To-do lists

- **Are you using to-do lists?** **→ EXERCISE**
  - Use the list you prepared
  - 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

Evo Planning: Weekly TaskCycle

- Are we *doing* the right things,
  in the right order,
  to the right level of detail for now
- Optimizing estimation (見積もる) , planning and tracking (予実管理) 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



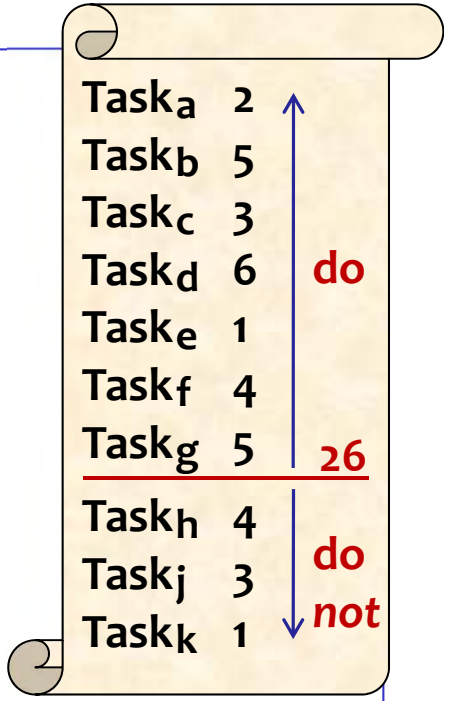
Effort and Duration 正味時間 - 延べ時間

- Days estimation → duration (calendar time)
- Hours estimation → effort

- Effort variations and duration 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 (正味) 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



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

At the end of the week

- **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 time spent on unplanned Tasks**
    - **Too much time spent on other Tasks**

    Discuss what the causes may be and decide how to improve (Check and Act)
- **Conclude unfinished Tasks after *dealing with the consequences***
  - **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**

Weekly 3-Step Procedure

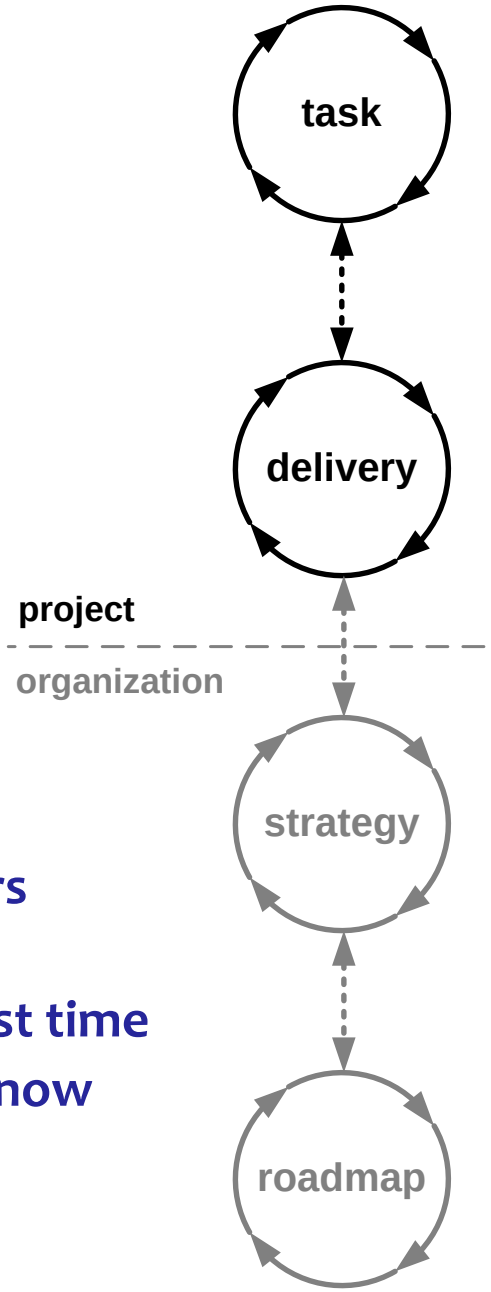
- **Individual preparation**
  - Conclude current tasks
  - What to do next
  - Estimations
  - How much time available
- **Modulation with / coaching by Project Management (1-on-1)**
  - **Status** (all tasks done, completely done, not to think about it any more ?)
  - **Priority check** (are these really the most important things ?)
  - **Feasibility** (will it be done by the end of the week ?)
  - **Commitment and decision**
- **Synchronization with group (team meeting)**
  - **Formal confirmation** (this is what we plan to do)
  - **Concurrency** (do we have to synchronize ?)
  - **Learning**
  - **Helping**
  - **Socializing**

The Real Benefit of TaskCycles

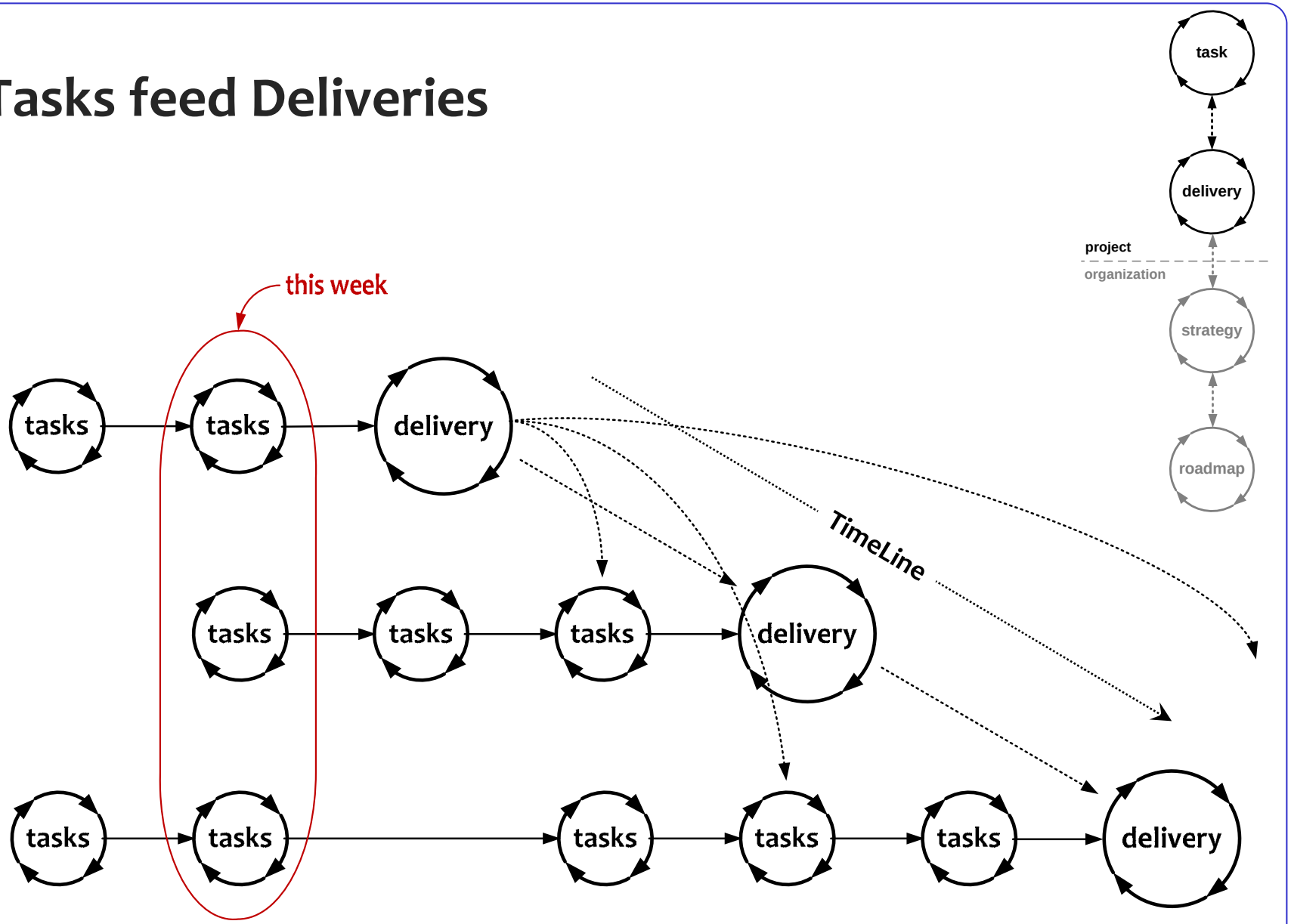
- **We see issues before they cause trouble**
- **And deal with them before they cause trouble**

DeliveryCycle

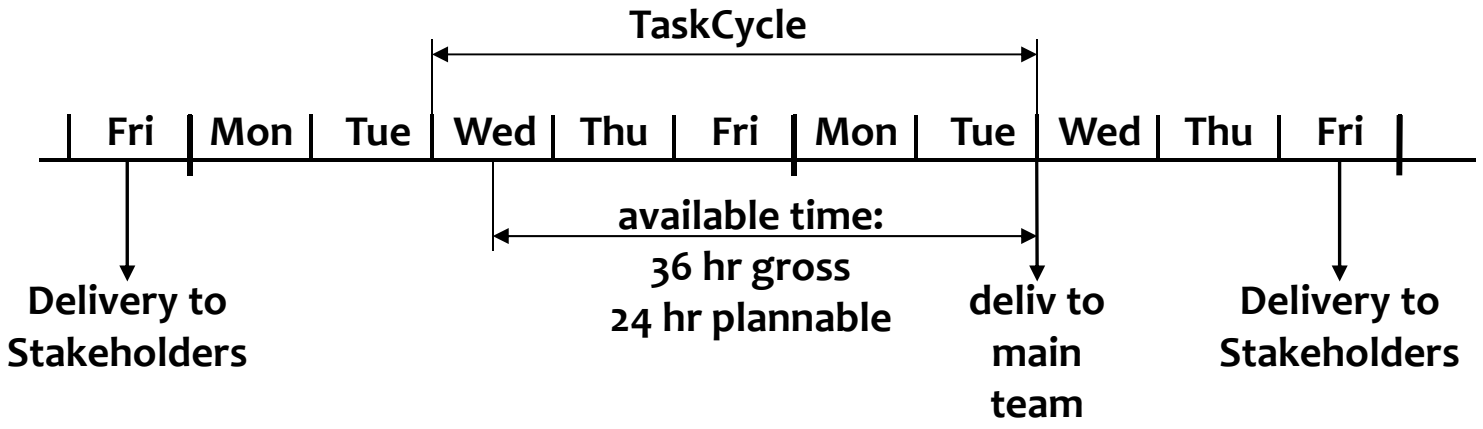
- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
     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

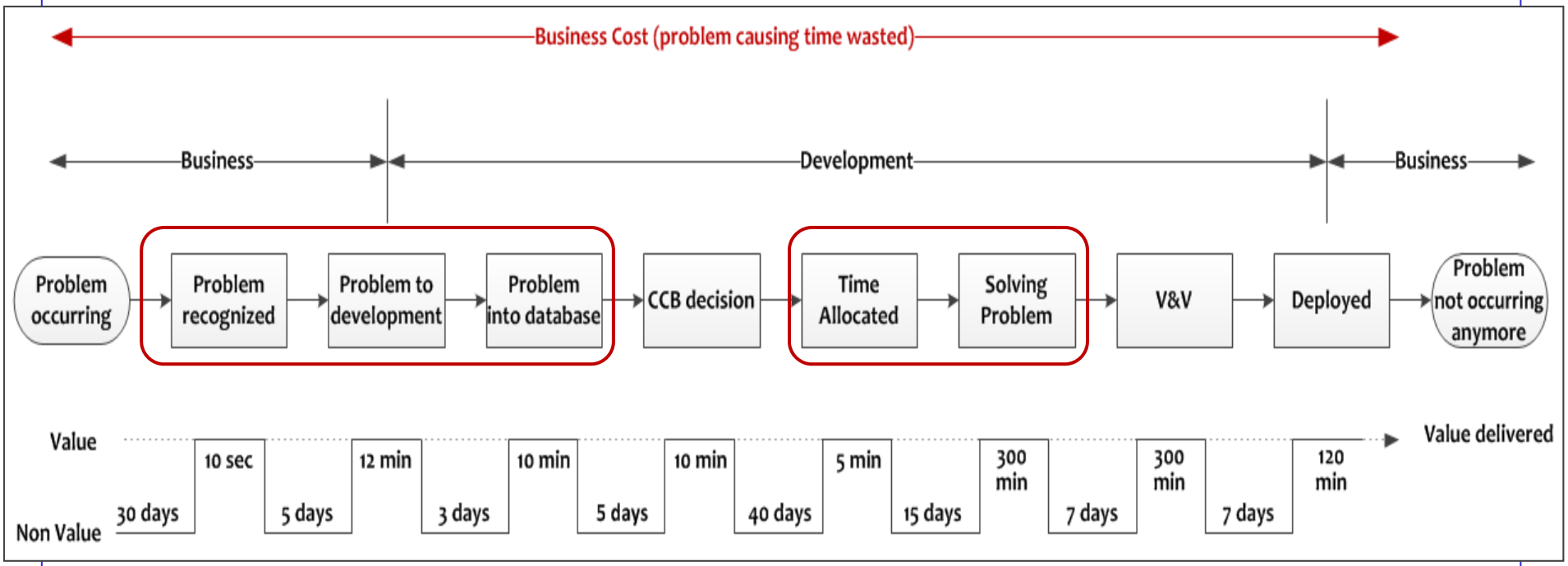
Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	42

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

Jerome	
XMLa	3
XMLb	3
...	

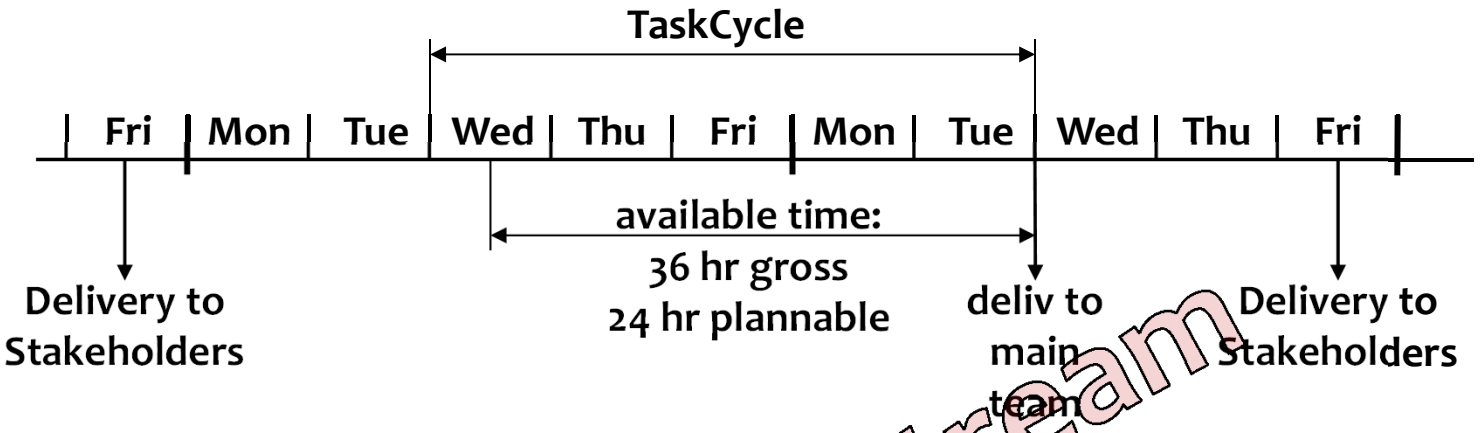
Value stream mapping

(actually: Cost Stream Mapping: adding cost, time and imperfections)



- **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 5万円/day ≈ 2300万円
- **Net Cost of Value:** 1.6 days → 3 people x 1.6 days x 10万円/day ≈ 50万円

Designing a Delivery



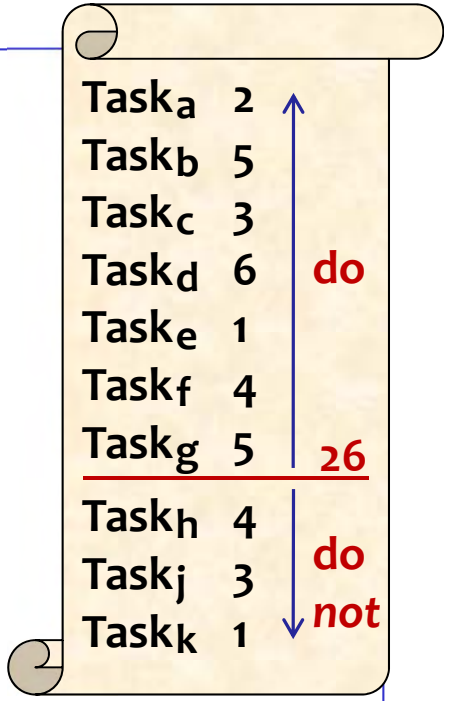
Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

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

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

Jerome	
XMLa	3
XMLb	3
...	

TaskCycle Exercise



- How much time do you have available
- 2/3 of available time is net plannable time
- What is most important to do (update your 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 ?

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 ?

Realistic estimation in 3 weeks

- In 3 weeks people can change estimation from optimistic to realistic
  - 1st week 40%
  - 2nd 80%
  - 3rd week 100%

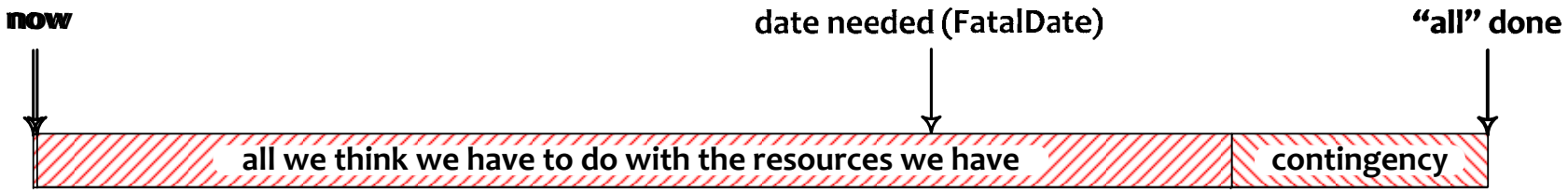
TimeLine

How to make sure we get the right results at the right time

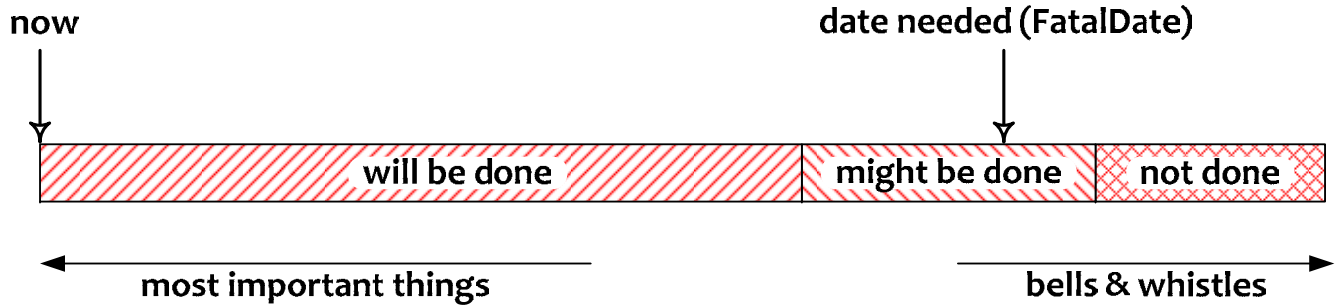
If it easily fits ...

TimeLine

What the customer wants, he cannot afford



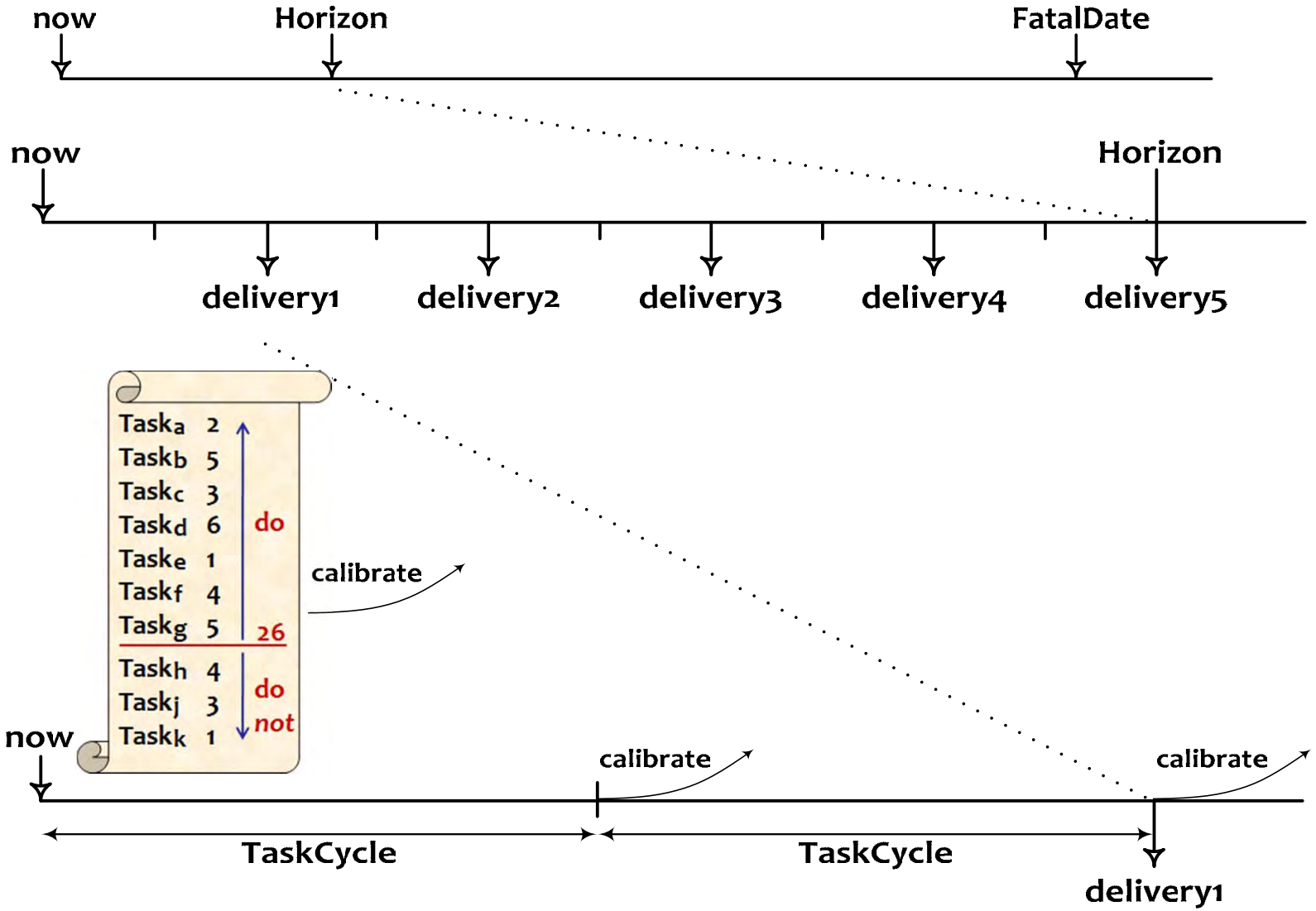
Standard Projects



Evo

- **Better 80% 100% done, than 100% 80% done**
- **Let it be the most important 80%**

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

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

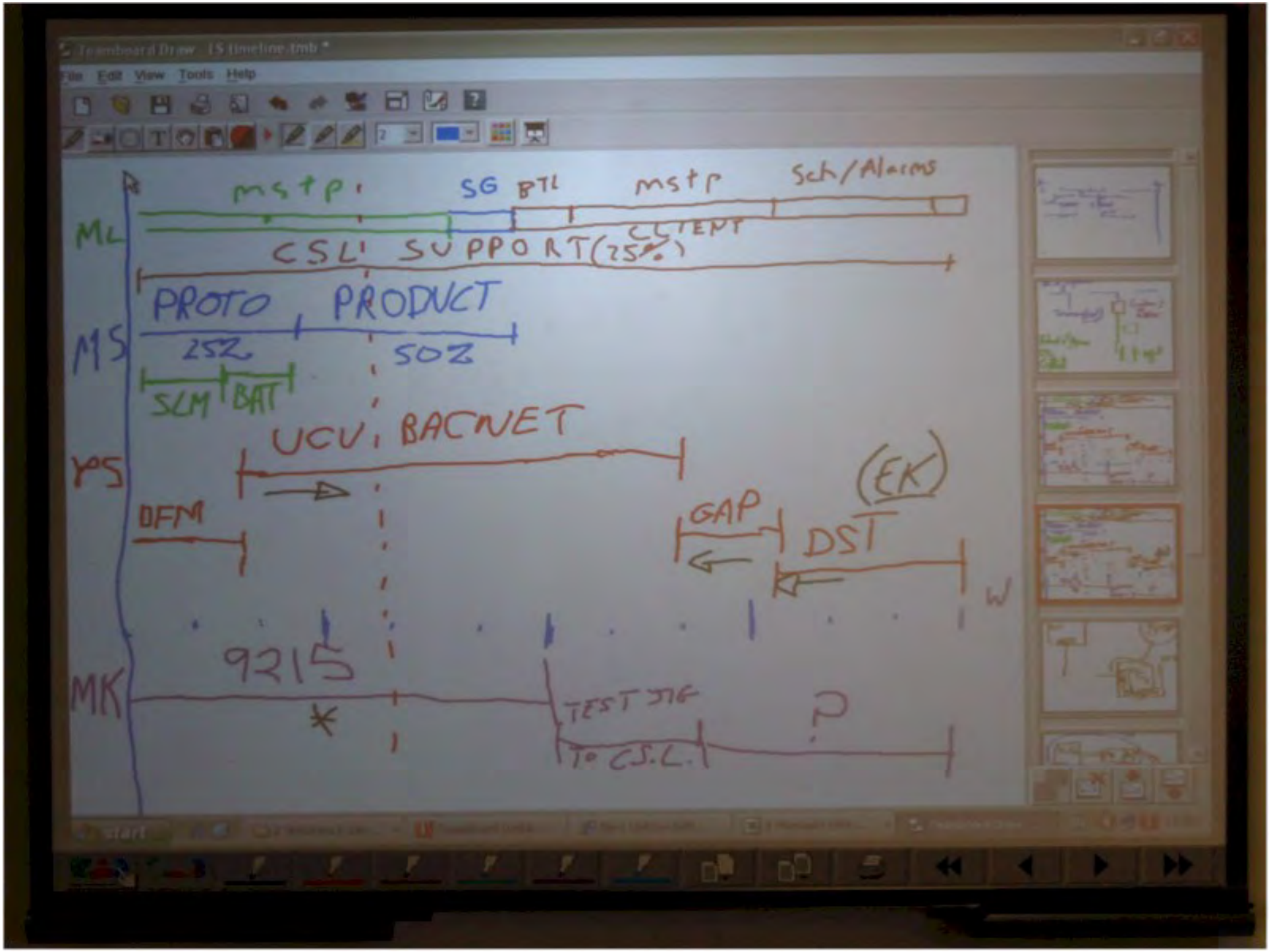
Predicting *what* will be done *when*

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

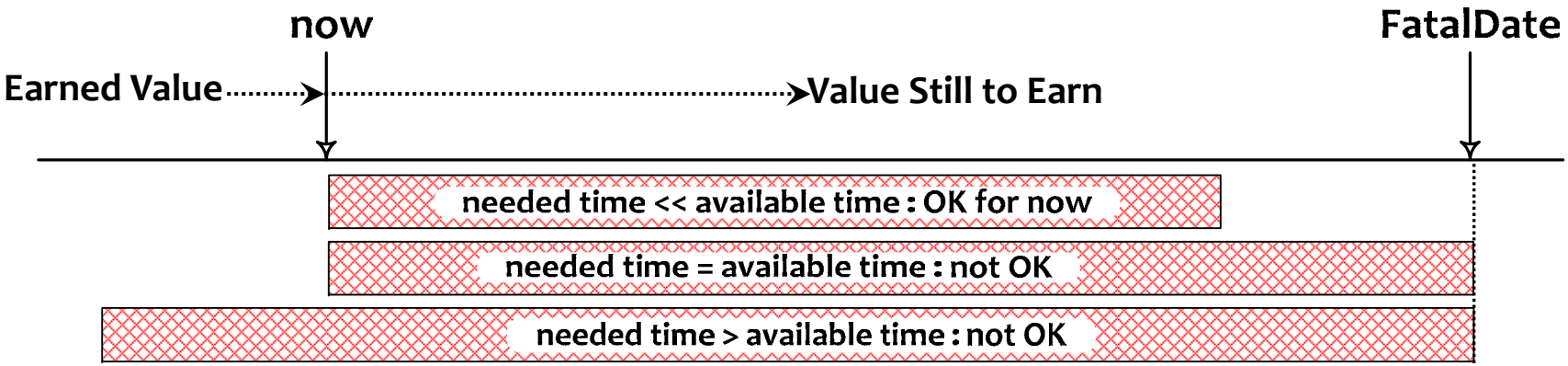
Product/Portfolio/Resource Management

- 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


mstp
SG BTL
mstp
Sch/Alarms
ML
CSL SUPPORT(25%)
CLIENT
PROTO
PRODUCT
MS
25%
50%
SCM
BAT
UCV BACNET
YS
(EK)
DFM
GAP
DST
W
9215
MK
*
TEST JIG
TO C.S.L.
?

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



- **Value Still to Earn**

versus

- **Time Still Available**

If the match is over, you cannot score a goal

Deceptive options

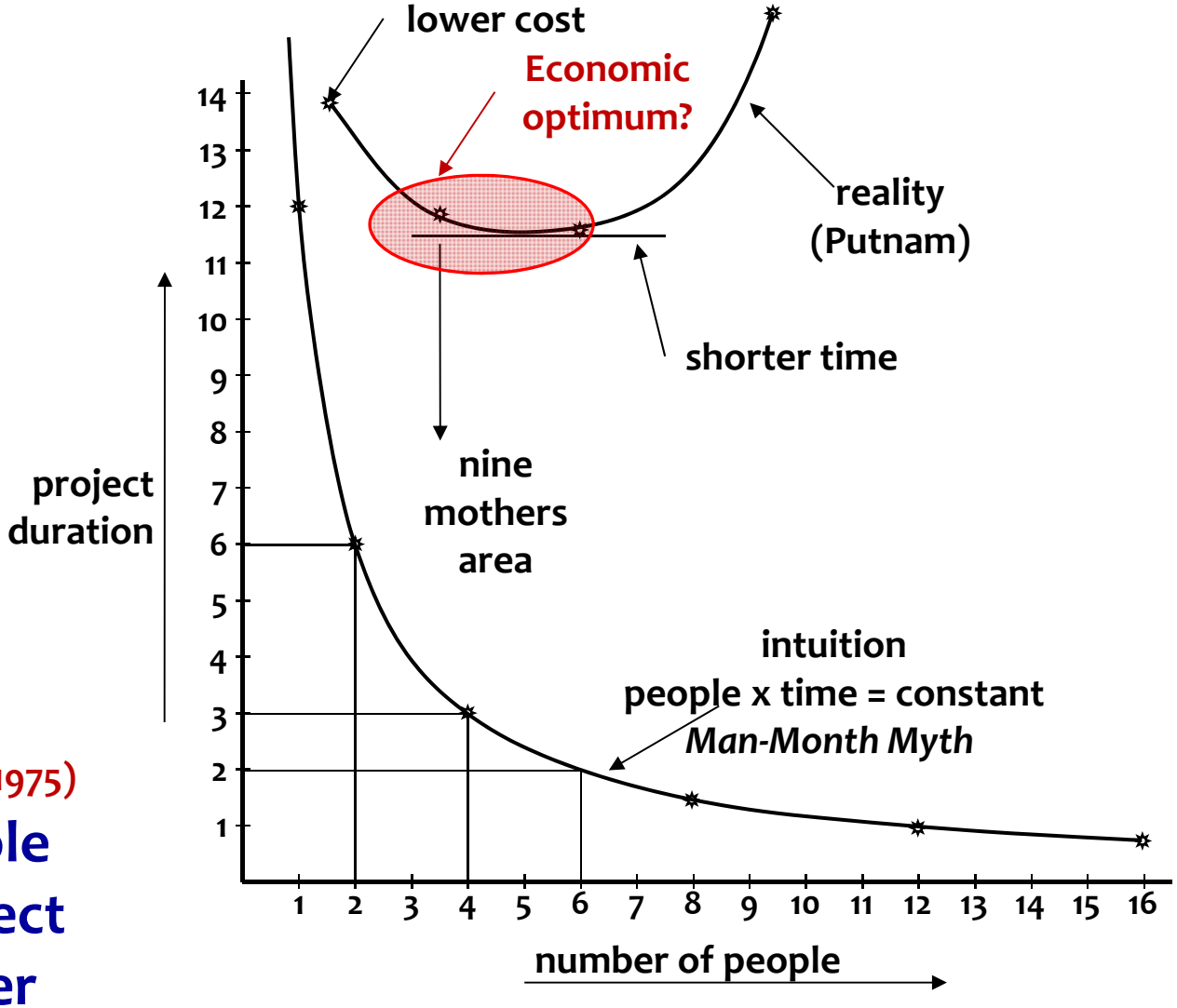
- **Hoping for the best** (fatalistic)
- **Going for it** (macho)
- **Working Overtime** (fooling ourselves)
- **Moving the deadline**
  - Parkinson's Law
    - Work expands to fill the time for its completion
  - Student Syndrome
    - Starting as late as possible,
      only when the pressure of the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

Adding people to a late project ...

makes it later

(Brooks' Law, 1975)

The Myth of the Man-Month



Brooks' Law (1975)

Adding people to a late project makes it later

Saving time

Continuous elimination of waste

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

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

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 !

Preparation for tomorrow

1. **Make a TimeLine for your work or project**
   - Do you know the FatalDay ?
   - Are there Starting Deadlines already passed ?
   - What will you do about any issues ?
2. **Find a relevant document for review**
   - Look at it as a reviewer
   - What would you find if you review ?

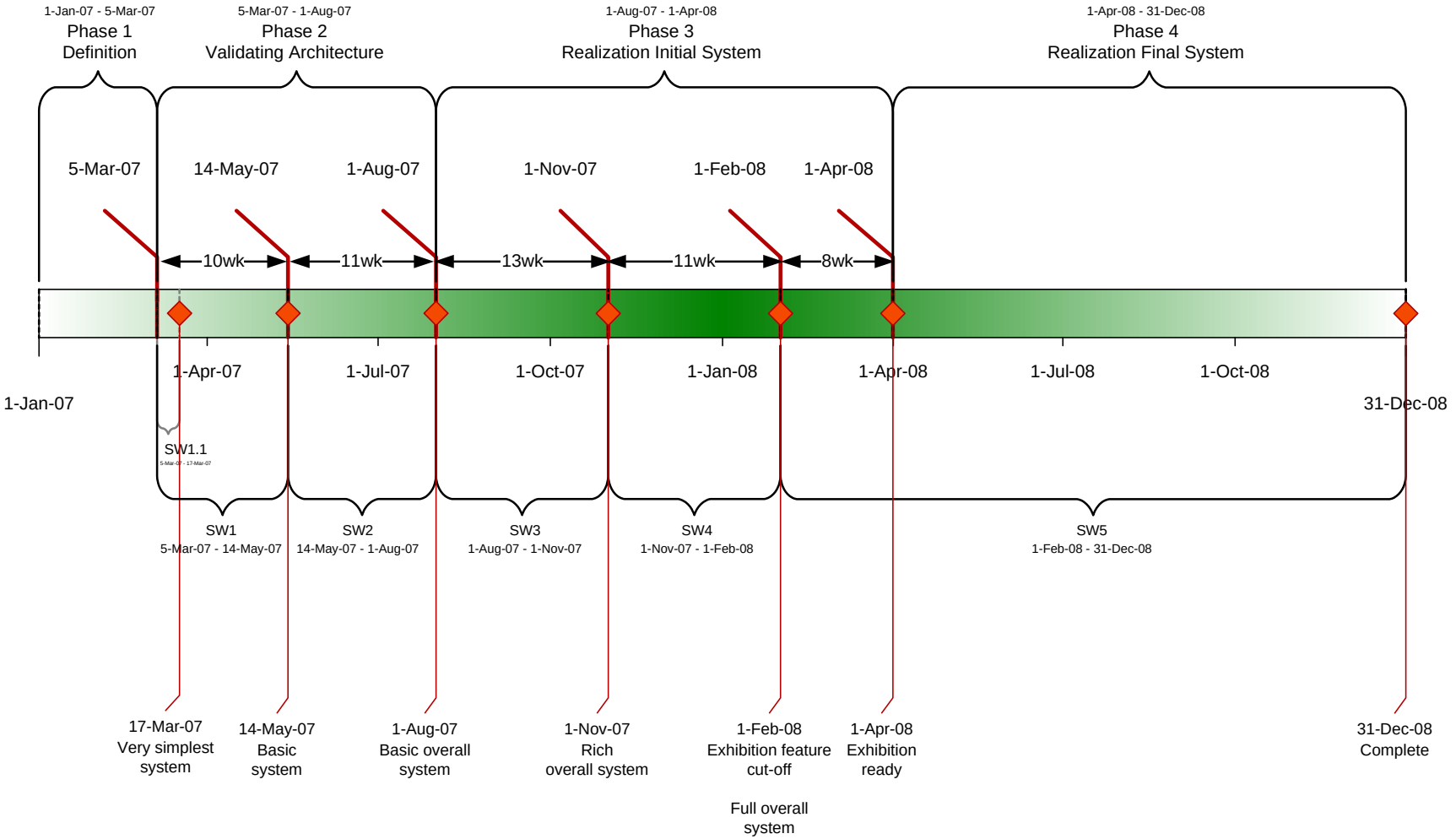
Day 4

Did you prepare ?

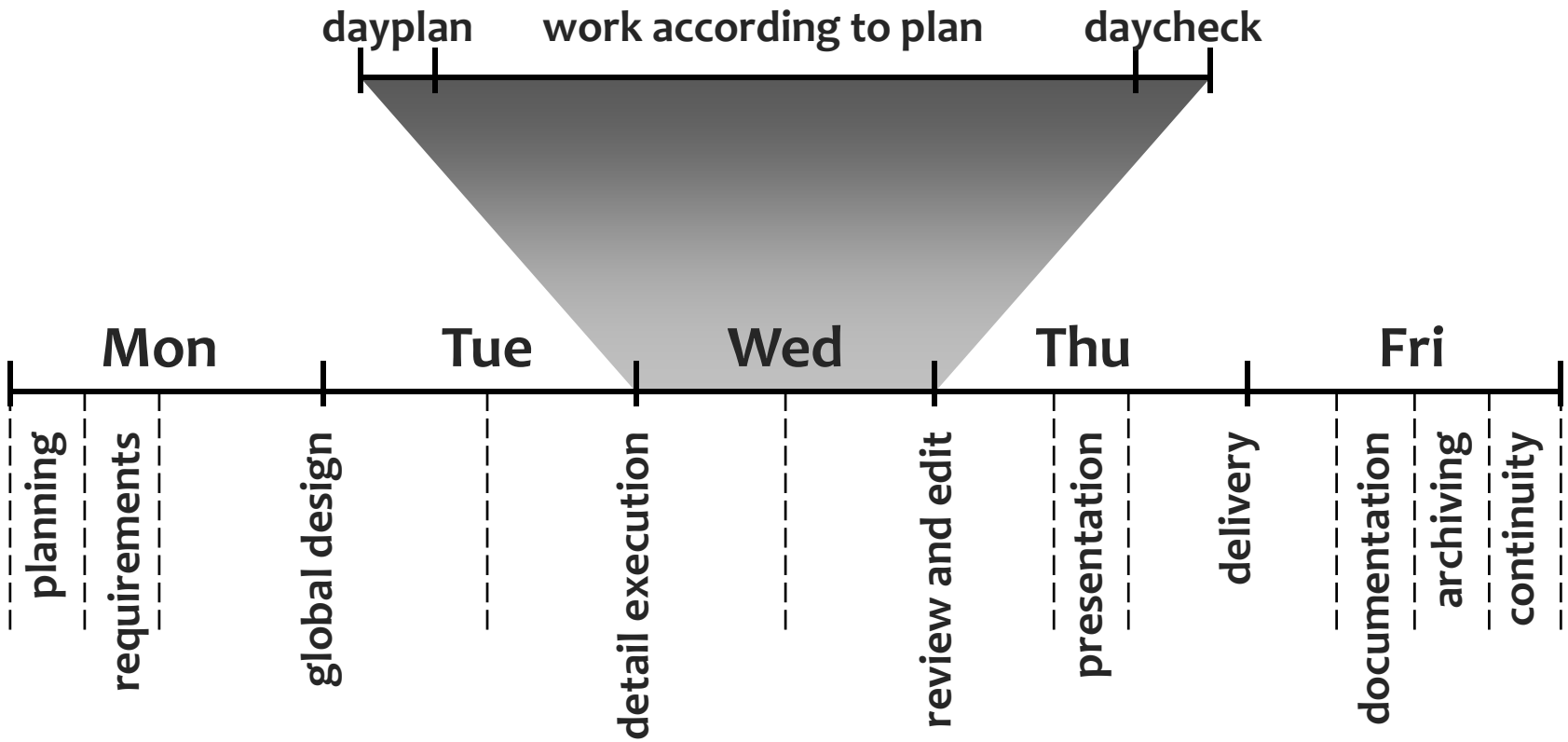
1. **Make a TimeLine for your work or project**
   - Do you know the FatalDay ?
   - Are there Starting Deadlines already passed ?
   - What will you do about any issues ?
2. **Find a relevant document for review**
   - Look at it as a reviewer
   - What would you find if you review ?

TimeLine examples

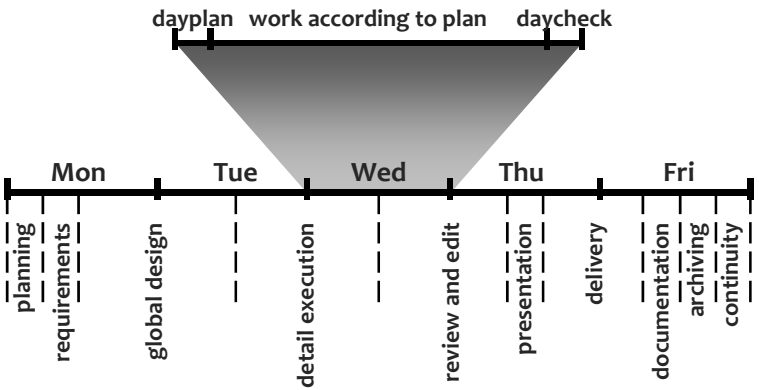
TimeLine example



5 day project model

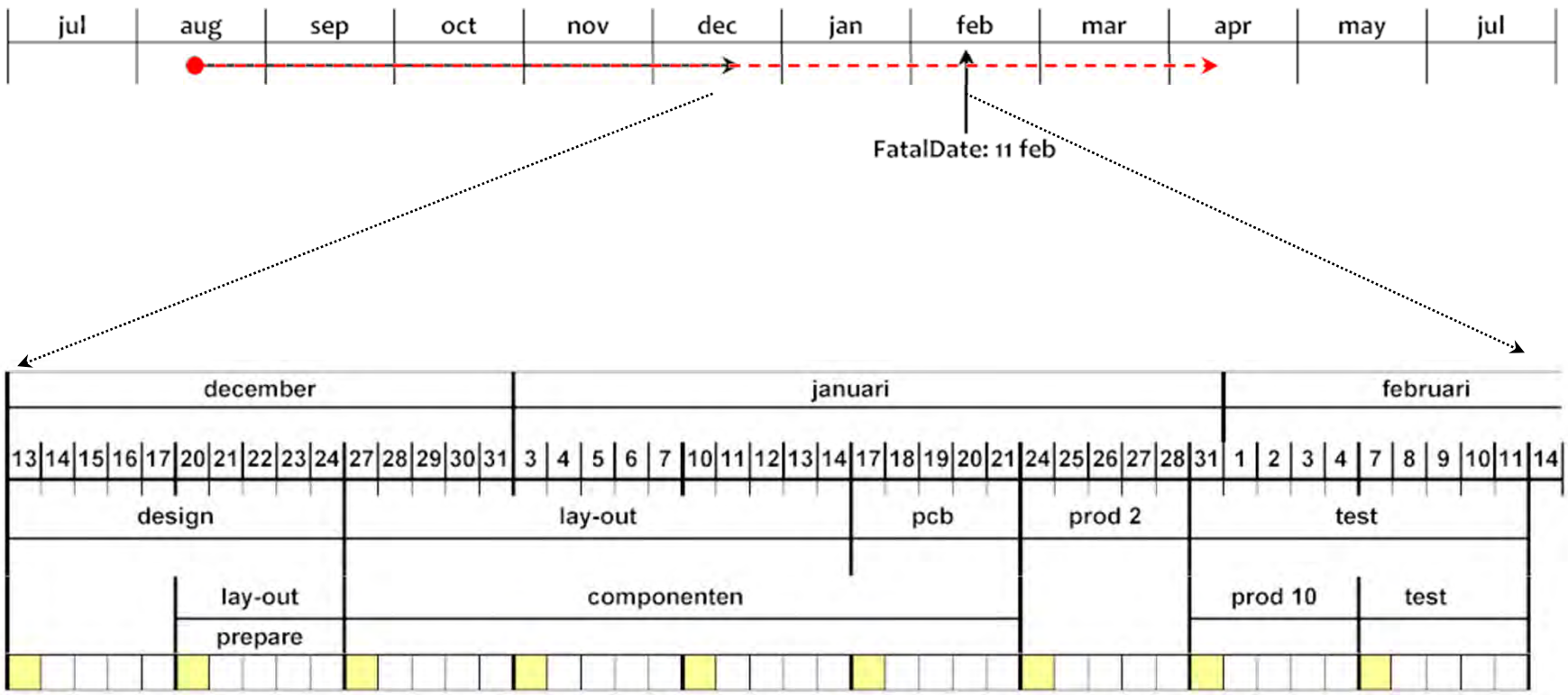


Available TimeBoxes



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

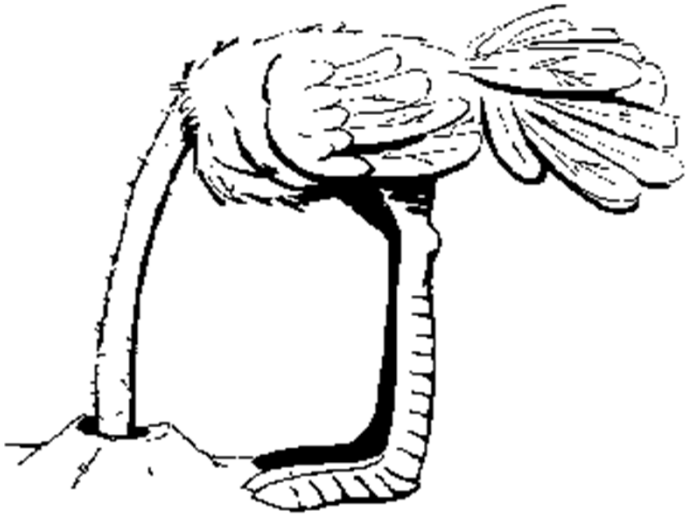
TimeLine planning



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 ?



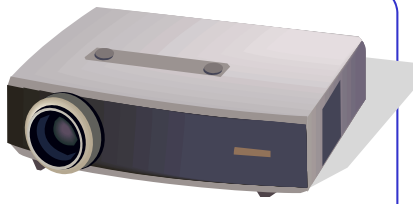
In stead of complaining about a problem …

(Stuck in the Check-phase)

Let's do something about it !

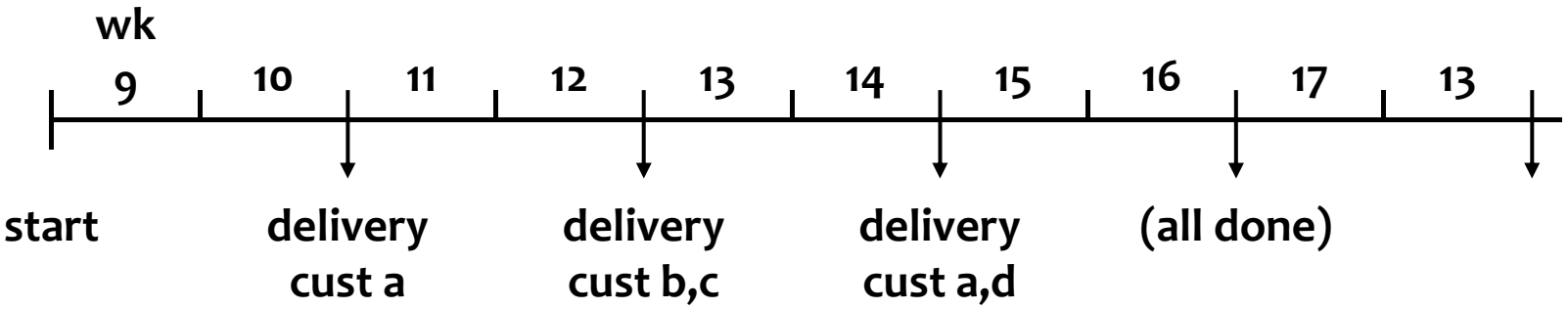
(Moving to the Act-phase)

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

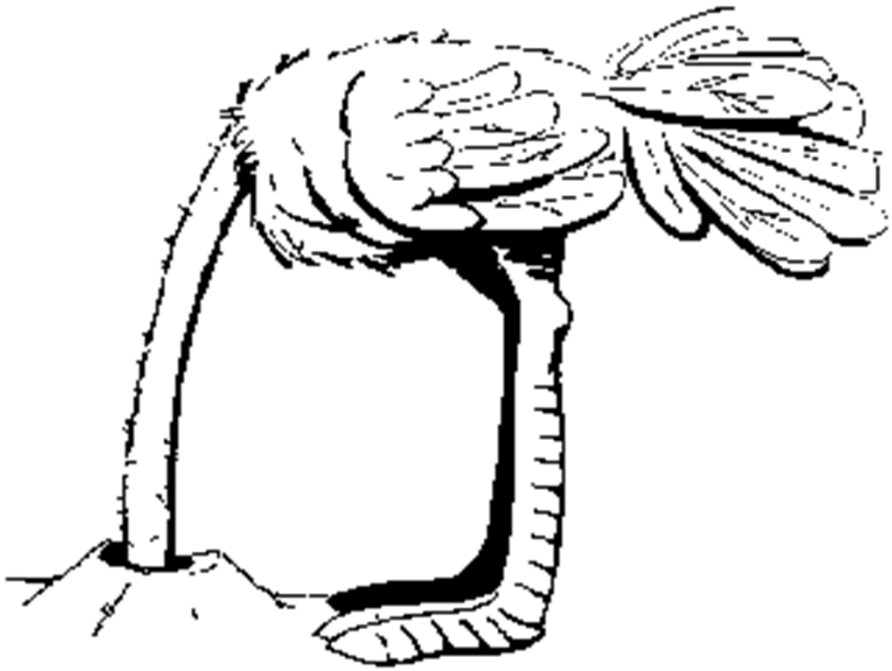
- "We'll have a solution at that date … Will you be ready for it ?"
  An other customer could be more eagerly waiting
- Most promising customers

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

Some details

If we add something …

If we add something, something else will not be done

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

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:

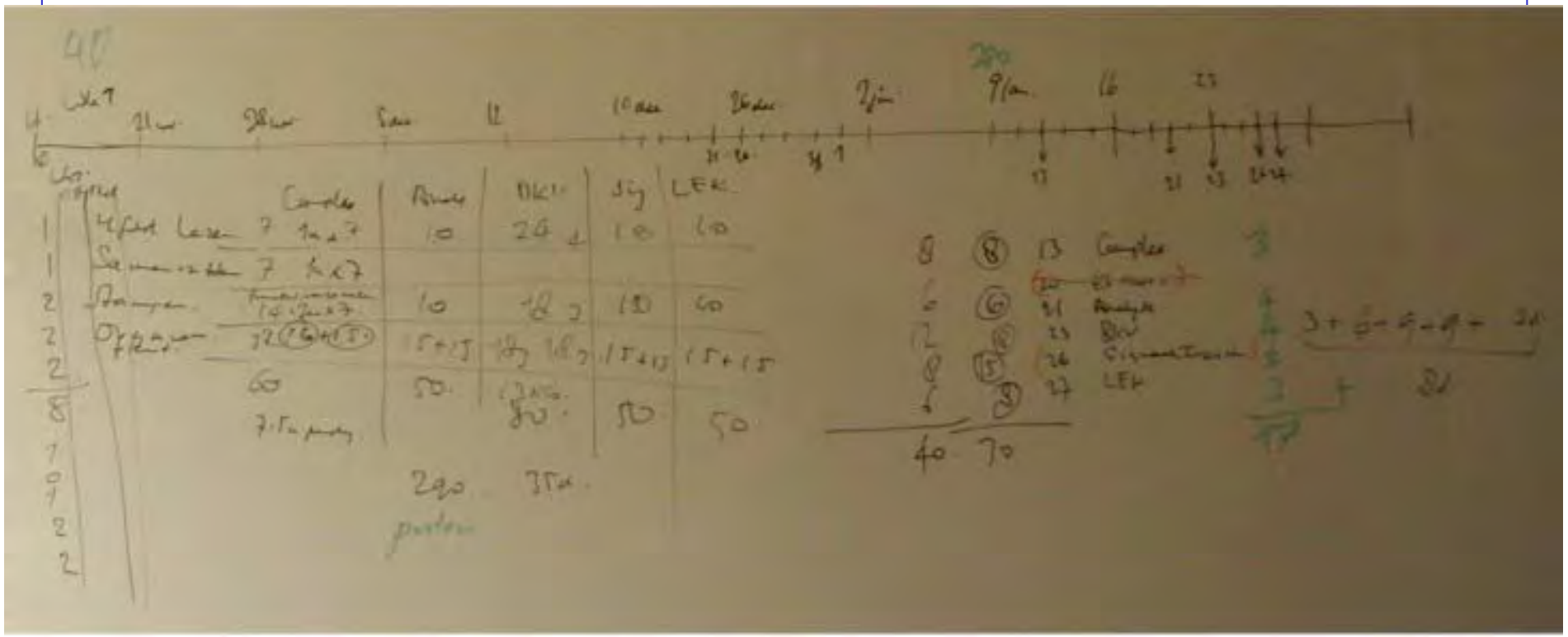
1. **Define the expected Results of the new Task properly**
2. **Estimate the time needed to perform the new Task, to the level of detail really needed**
3. **Go to your task planning tool (many projects use the ETA tool)**
4. **Decide which of the planned Tasks is/are going to be sacrificed (up to the 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.**

TimeLine exercise example

- **Preparing for student exams**

*

What we did



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!)

How to check we wrote the right things

Do you ever make a mistake?

- People make mistakes
- We are people

- If we are doing something we are also producing defects

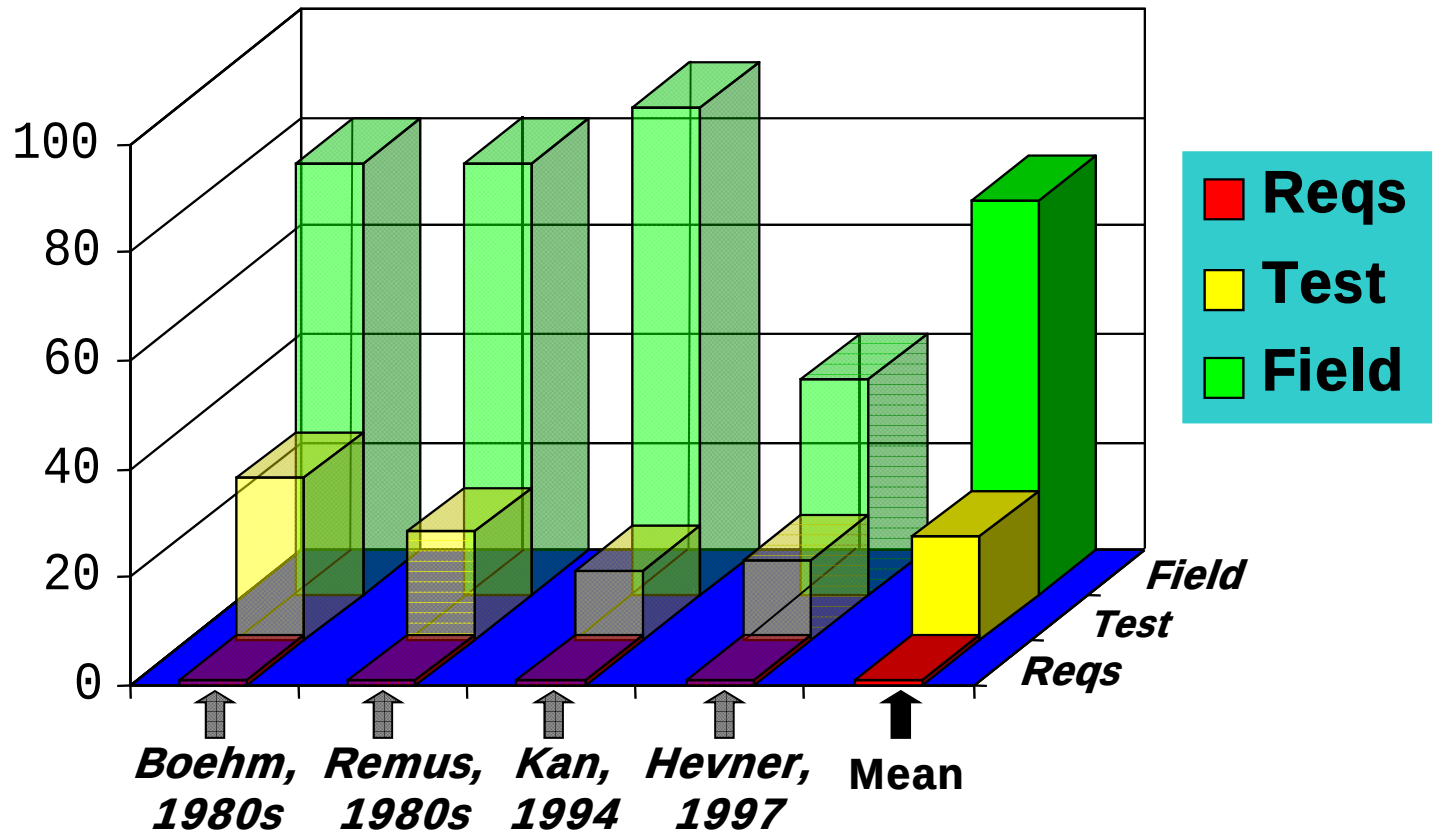
- If we *think* we are ready, *there are still defects*

Costs of defects

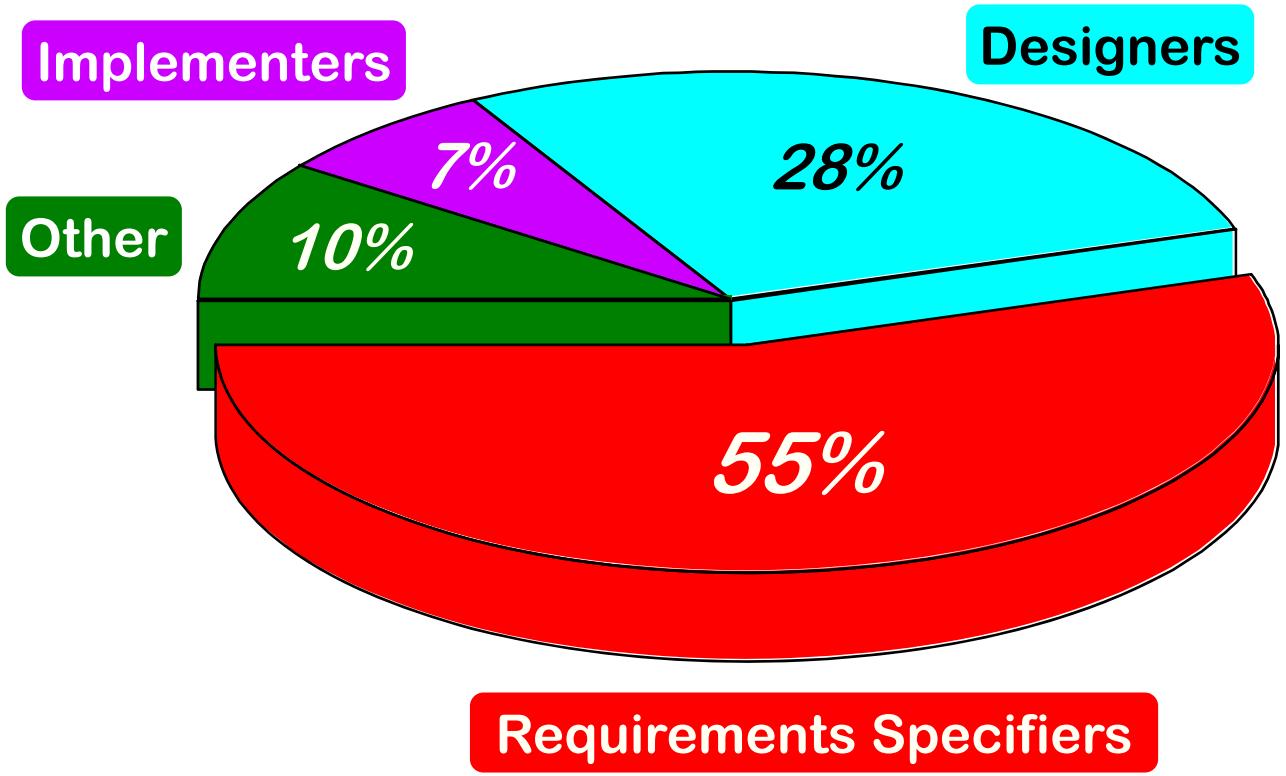
The longer a defect stays in the system, the more it costs to find and repair

Cost of Requirements Defects

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

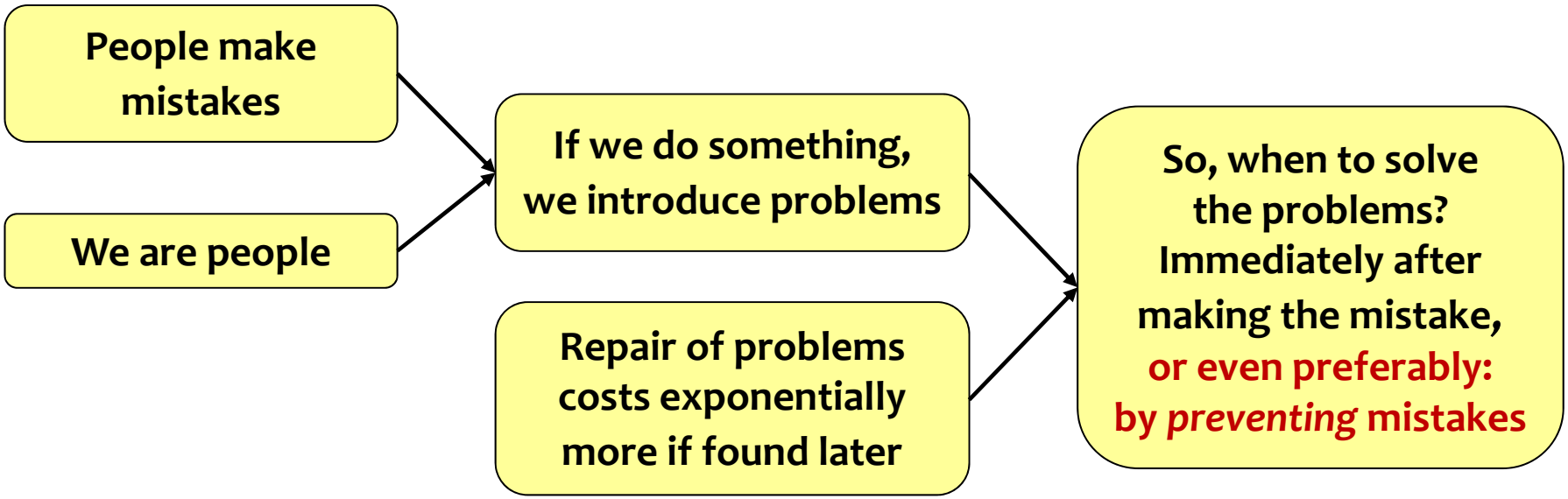


Typical Defect Injectors (*cost* breakdown)



After Bender Associates, *1996*

Inevitable consequence



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

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

Are you reviewing?

*

Many types of Review to choose from

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

Techniques

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

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

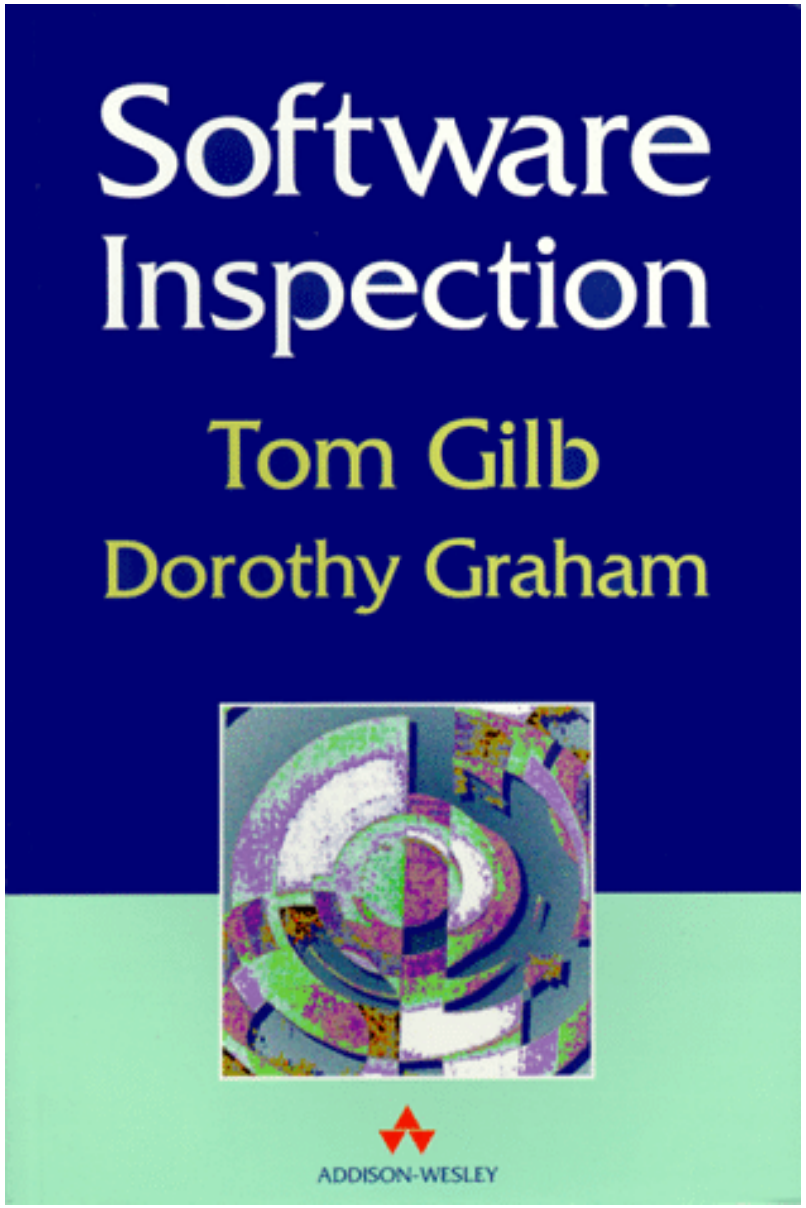
Would you like to base further work or decisions on a document of unknown quality?

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



A ready to use recipe …

16 page
Inspection Manual

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

Inspection Manual

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

Version: 0.43 (Changed *Plan* into *Goal*)
Date: Oct 13, 2007
Owner: Niels Malotaux
Status: not inspected
Intended readership: anybody interested in or busy with inspections

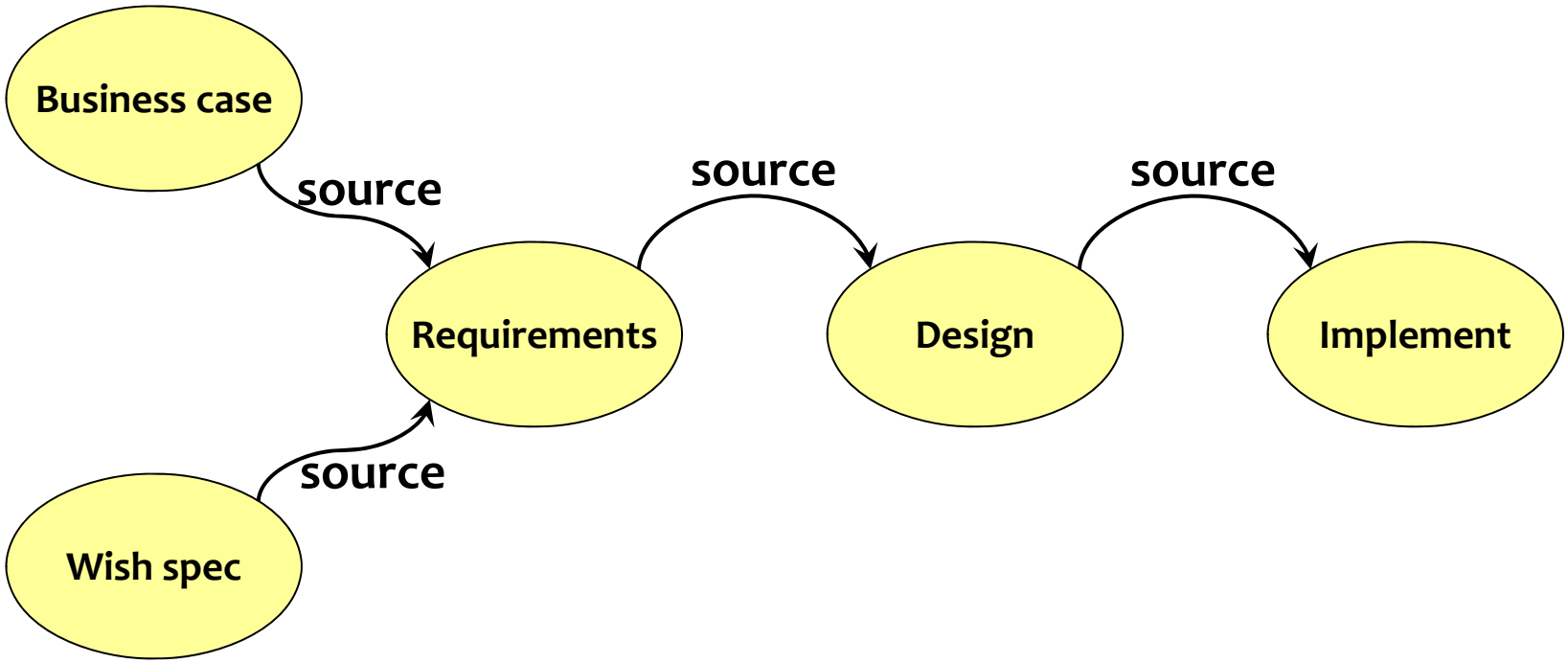
Note: Most of these texts are originally taken from the book:
"Software Inspection" by Tom Gilb and Dorothy Graham
Addison Wesley, 1993, ISBN 0-201-63181-4, and from
web-sites, such as www.result-planning.com (Tom Gilbs web-site)
This is a starting point from which the procedures, rules, etc.
may be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

Let's review

- Do we have a document ?
- Select one representative page
- Make some copies
- Review
- Then we'll discuss the result of the review

Simple Rule for Reviews

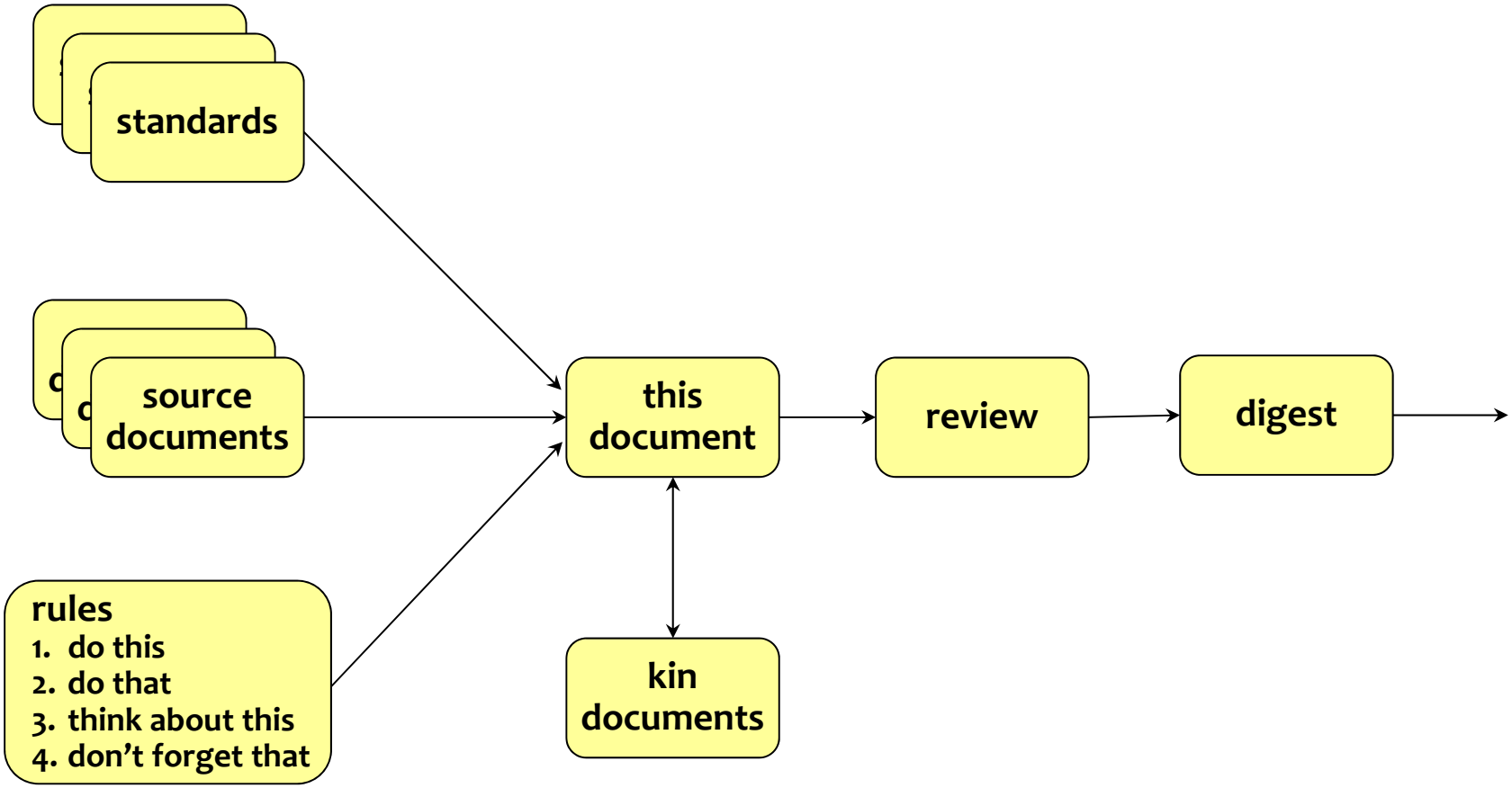
"We don't review unless there is a source document"



Now review again

- **Any difference ?**

Document generation



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?

Defect classes

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

Rules

- **Rules are the law for documents**
- **Defect = Rule violation**
  not: "I think this is wrong"

- **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

Generic Specification Rules

(see Inspection Manual)

GE0	(def)	Generic engineering specification rules apply to all engineering documents as required best practices
GE1	(relevant)	All statements should be relevant to the subject
GE2	(complete)	There should not be any significant omissions
GE3	(consistent)	Statements should be consistent with other statements in the same or related documents
GE4	(unambiguous)	All specifications should be unambiguous to the intended readership
GE5	(note)	Comments, notes, suggestions, not official part of document shall be clearly marked (“”, *ital*, /**/)
GE6	(brief)	All specifications shall be as brief as possible, to support their purpose, for the intended readership
GE7	(clarity)	All specifications shall result in clarity to the intended readership regarding it’s purpose or intent (the burden is on author, not the reader) *Note: It is not enough that statements are unambiguous. They must contain clarity of purpose: why is it there?*
GE8	(elementary)	Statements shall be broken into their most elementary form *Note: This is so that they each can be cross-referenced externally (Traceability)*
GE9	(unique)	Specifications shall have a single instance in the entire project documentation
GE10	(source)	Statements shall have source info (spec ← source)
GE11	(risk)	The author should clearly indicate any information which is uncertain or poses any risk to the project, using indications like: {<vaguely defined>, ?, ??, 70% ±20, suitable comments or notes}
GE12	(verifiable)	All statements should be verifiable
GE13	(true)	The statement is simply not true

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

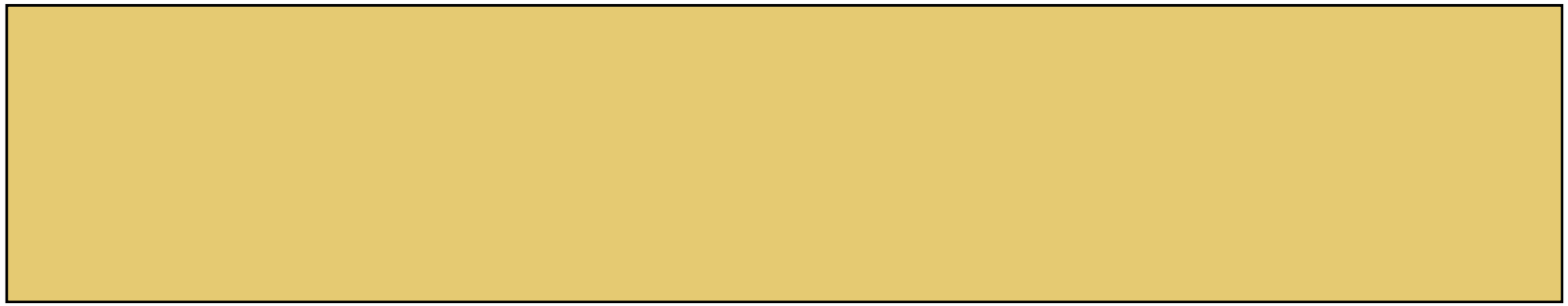
Optimum Checking Rate

- 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")

Optimum checking rate

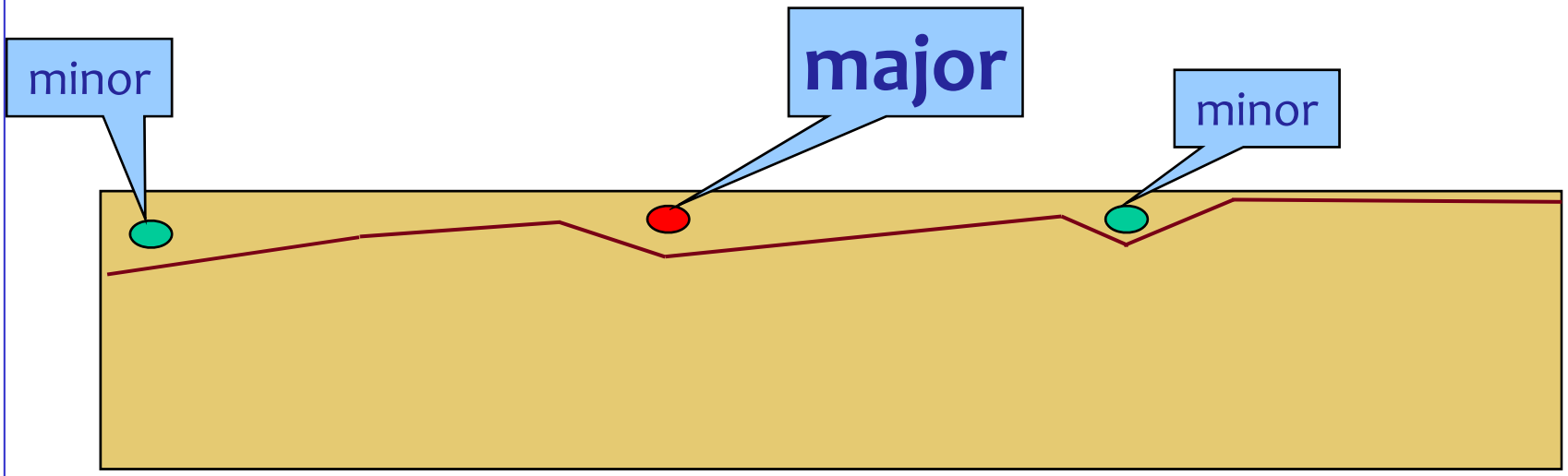
Ref. Dorothy Graham



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

Ref. Dorothy Graham

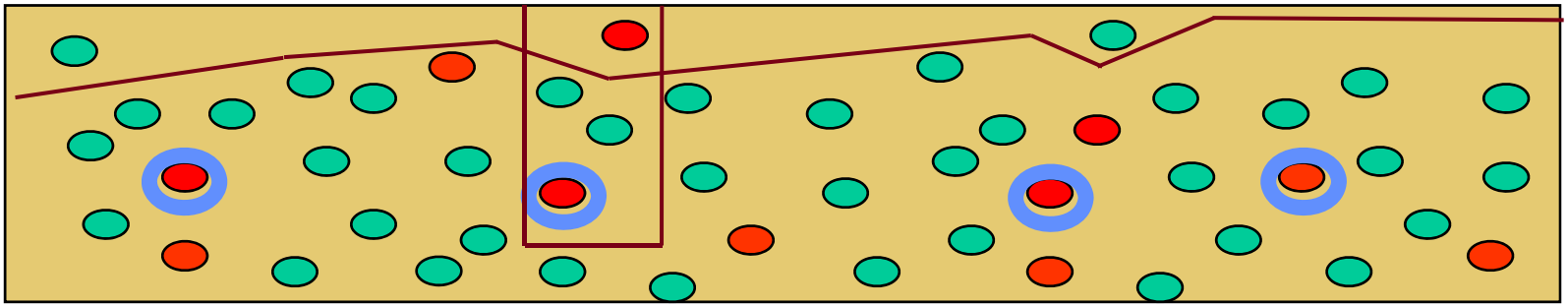
Review "Thoroughness"?



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

Inspection Thoroughness

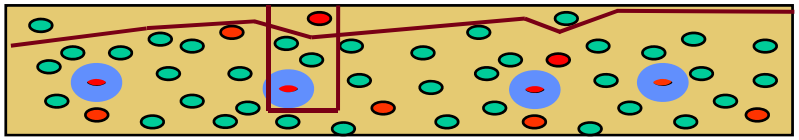
Ref. Dorothy Graham



- **Inspection can find deep-seated defects**
- **All of that type 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 !**

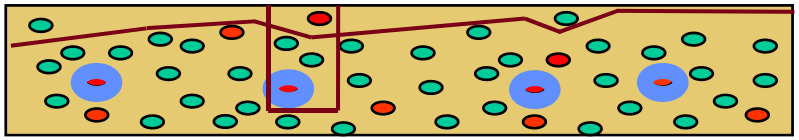
Gilb/Graham Inspection



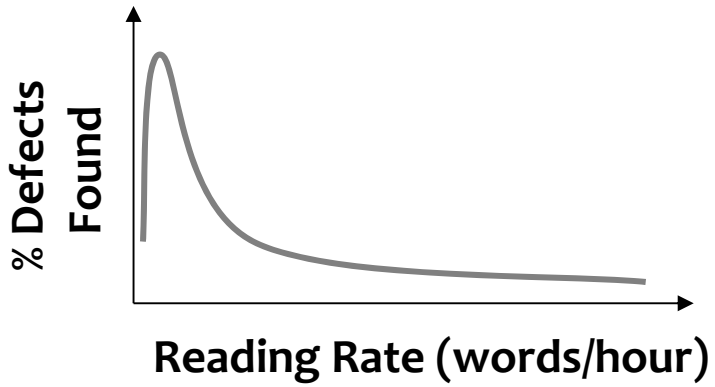
Gilb/Graham inspection differs from other types of inspection in some or all of these ways:

- **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

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!

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
- ✓ 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*

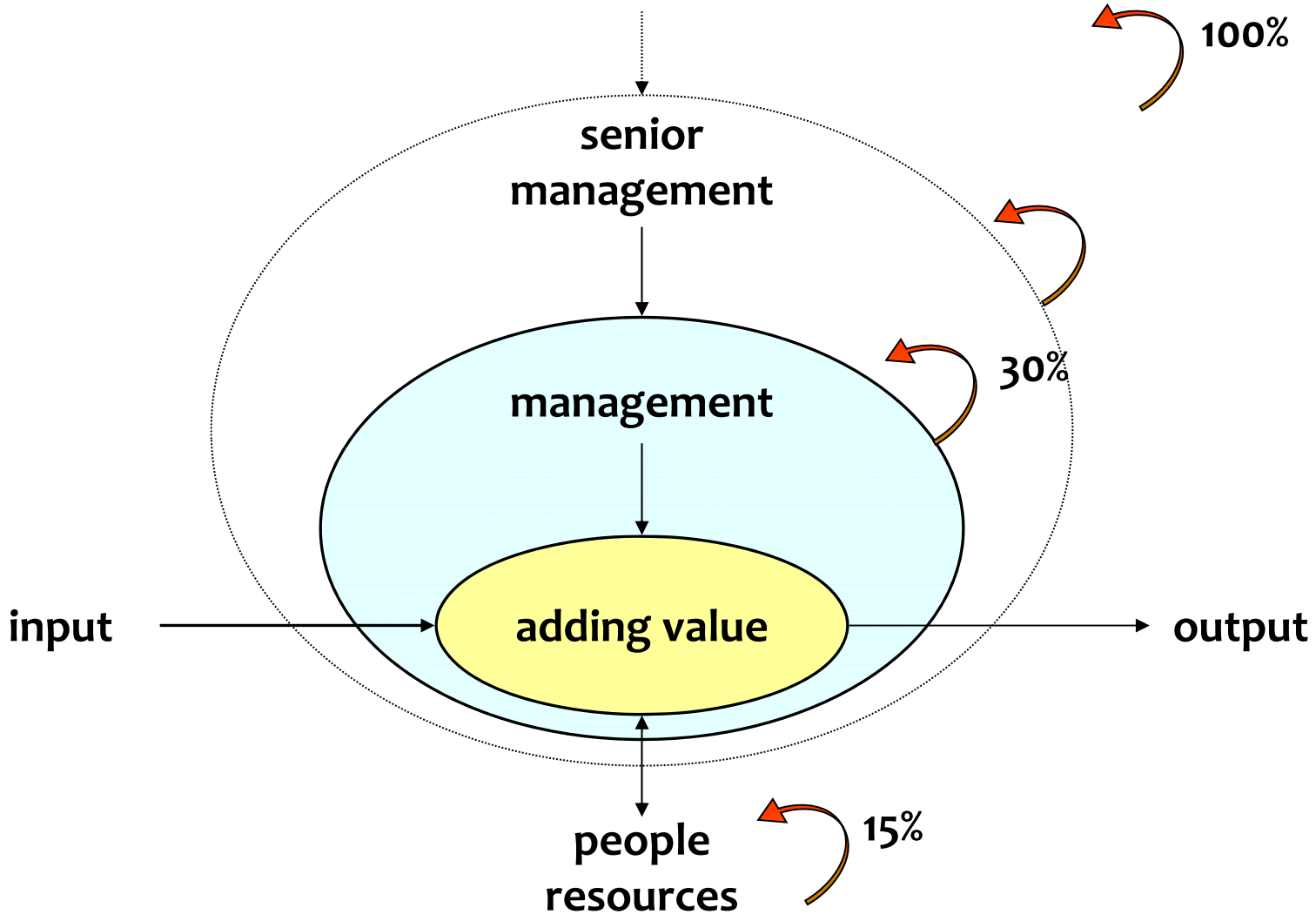
Nice job! Early detection has big benefits - BUT…

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

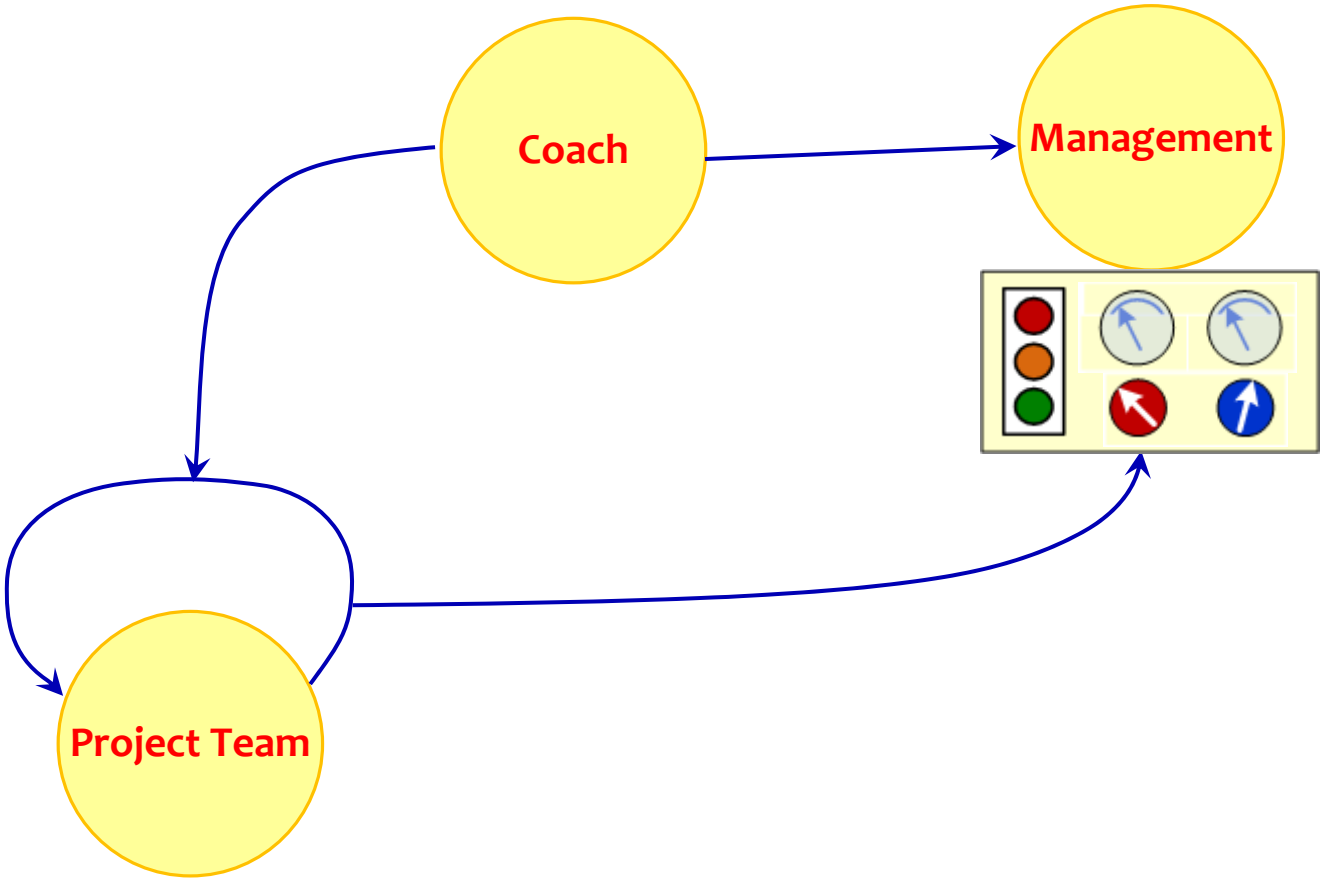
Cost-effective defect prevention is the bottom line

Management Issues

Simple model of Management



Local Loop Principle



Finally

Magic words

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

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?

- See more at http://www.malotaux.nl/?id=mantras

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

More

www.malotaux.nl/?id=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

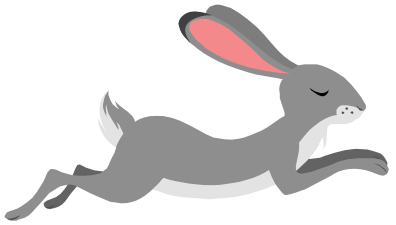
www.malotaux.nl/?id=inspections

Inspection pages

What now ?

Questions ?

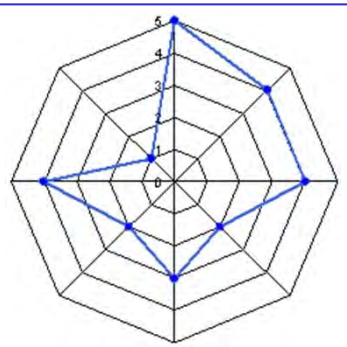
Agile or agile ?



What is Agile ?

- **A philosophy (Agile Manifesto)**

The Agile Manifesto (2001)



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

Through this work we have come to value:

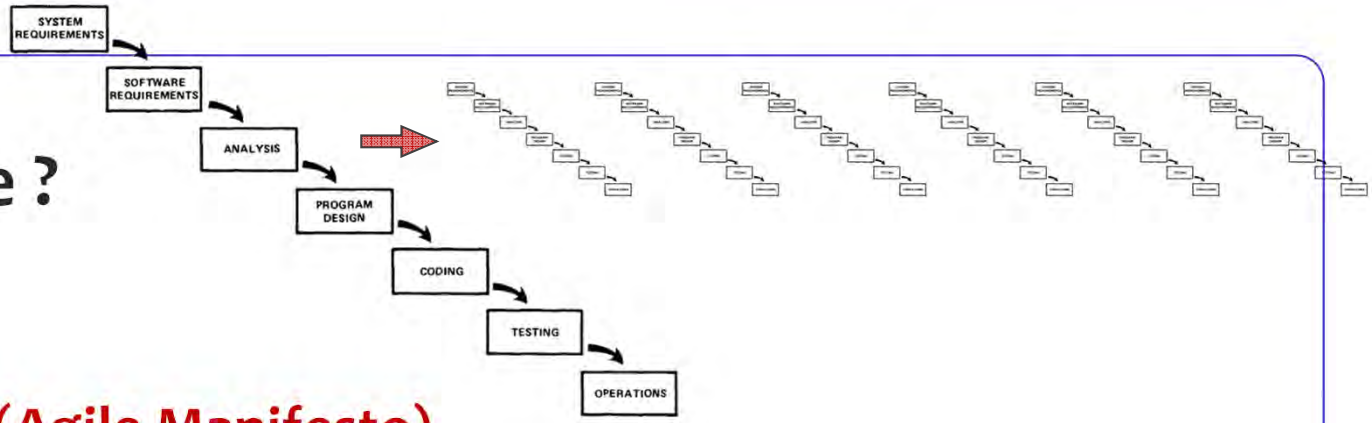
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

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

From the Principles behind the Agile Manifesto

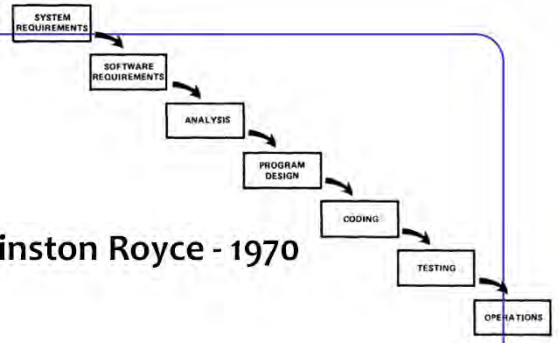
- **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software**
  Software is always part of a system
- **We welcome changing requirements, even late in development**
  If requirements have to change, let's *provoke* requirements change as quickly as possible
- **We deliver working software frequently;**
  Working software is the primary measure of progress
  What we deliver simply works
- **Business people and developers must work together daily**
  Do they ? Daily ?
- **Simplicity - the art of maximizing the amount of work not done**
  The art of not doing what is superfluous ! Why make it complex if we can keep it simple ?
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly**
  Not just retrospectives, but even more importantly: prespectives

What is Agile ?

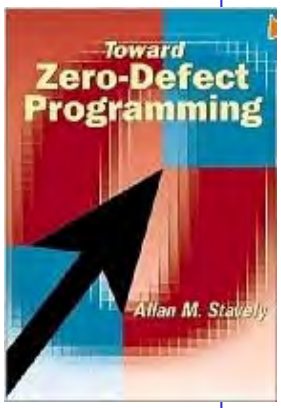


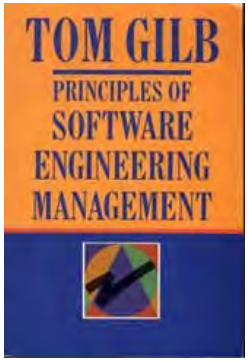
- A philosophy (Agile Manifesto)
- agile = ability to move quick, easy and adaptable
- Short iterations – not one Waterfall
- Delivering value (do we measure progress towards real value ?)
- Retrospectives (retrospectives on retrospectives: did it really work ?)
- Not a standard: You can make of it whatever you want
- XP - focus on software development techniques
- Scrum - very basic short term organization of development
- Are you agile if you religiously focus on a 'method' ?

The past was already ahead



- **Managing the development of large software systems** - Winston Royce - 1970
  - Famous 'Waterfall document': figure 2 showed a 'waterfall'
  - Text and other figures showed that Waterfall doesn't work
  - Anyone promoting Waterfall doesn't know or didn't learn from history
- **Cleanroom software engineering** - Harlan Mills - 1970's
  - Incremental Development - Short Iterations
  - Defect *prevention* rather than defect removal
  - Inspections to feed prevention
  - No unit tests needed
  - Statistical testing
  - If final tests fail: no repair - start over again
  - 10-times less defects at lower cost
  - Quality is *cheaper*
- **Evolutionary Delivery - Evo** - Tom Gilb - 1974, 1976, 1988, 2005
  - Incremental + Iterative + *Learning and consequent adaptation*
  - Fast and Frequent Plan-Do-Check-Act
  - Quantifying Requirements - Real Requirements
  - Defect *prevention* rather than defect removal





XP – eXtreme Programming

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

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

Scrum

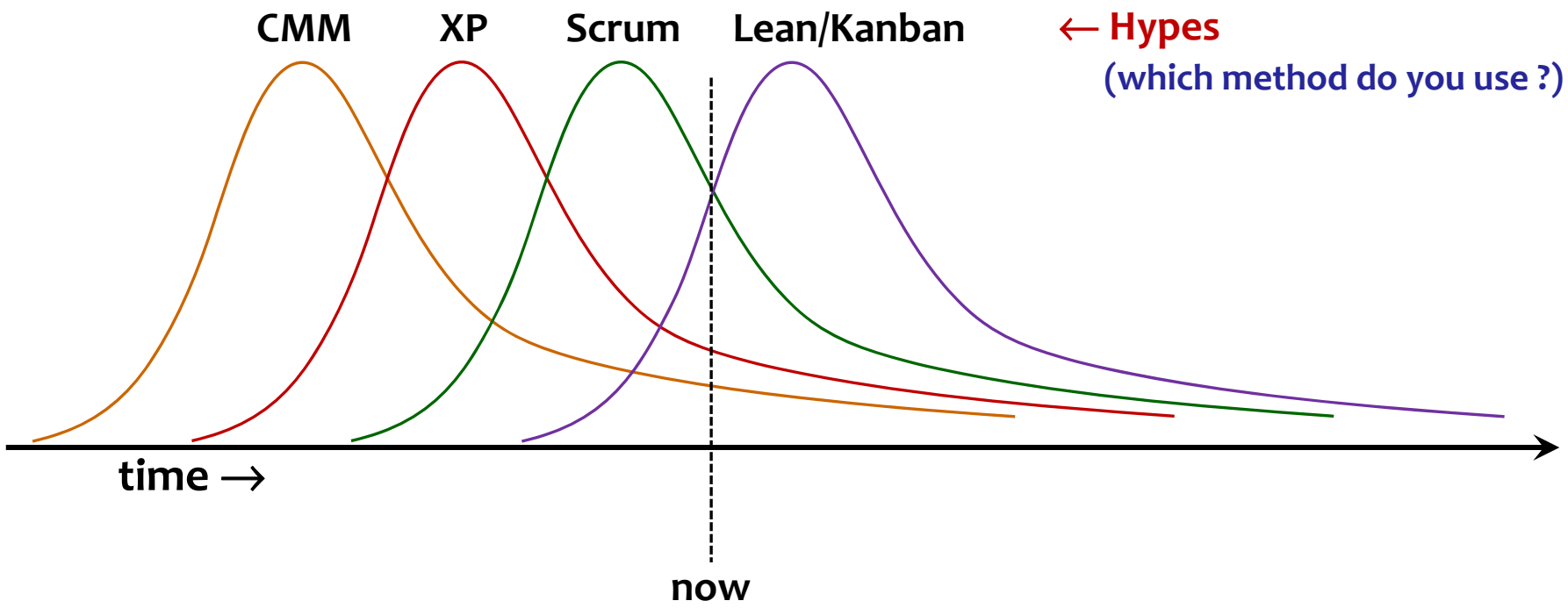
80% of Scrum projects are ScrumBut

- **Sprint**
  - 1 – 4 weeks
  - Sprint Planning meeting
  - Sprint Review meeting
  - Sprint Retrospective
- **Artefacts**
  - Product backlog
  - Sprint backlog
  - Sprint burn down chart

- **Roles**
  - Scrum Master (facilitates, coaches on rules)
  - Team – multifunctional (design, develop, test, etc)
  - Product Owner – voice of customer
- **Daily Scrum - Stand-up meeting**
  a. What have you done since yesterday
  b. What are you planning today
  c. Impediments limiting achieving your goals ?

a lot of ritual

It's not the method



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

What's usually missing in Agile ?

Ref Tom Gilb

Stakeholder Focus

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

Results Focus

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

Systems Focus

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

Planning

Ref Niels Malotaux

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

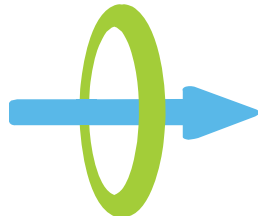
Essence of being really Lean & Agile

see also www.malotaux.nl/doc.php?id=79

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

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

Back to “We are Agile”



Still to do

Final assignment

(for Credit if Keio student)

Write a small report

- What did you learn
- What was missing
- Which questions do you still have ?
- What do you self think the answers to these questions are ?

Optional exercise

If you do this, you have to do it for 5 weeks !

See green booklet (website: booklet#2) chapter 6

- Week from <start date> till <due date>
- Gross time available: xx hr
- Net time available: yy hr
- List of tasks:
  - <short description of Task>, time needed, <ok>, <comment>
  - <short description of Task>, time needed, <ok>, <comment>
  - <short description of Task>, time needed, <ok>, <comment>

- The total time needed for all these tasks should be exactly equal to the net time available (yy hr)
- Fill in <ok> if the Task is done, completely done
- At the end of the week, all tasks should have ok
- At the end of the week, see chapter 6.5 of the green booklet

Schedule

24-27 September 2013

Tuesday	**17:15 ~ 18:45**	**19:00 ~ 20:30**
Wednesday	**17:15 ~ 18:45**	**19:00 ~ 20:30**
Thursday	**13:00 ~ 14:30**	**14:45 ~ 16:15** ← !!
Friday*	**17:15 ~ 18:45**	**19:00 ~ 20:30**

*** Note: Friday time may change: we will decide on Tuesday ← !!**

1 credit if you attend *all* lectures and participate in *all* exercises (Keio students only)