Planning for Quality Delivery

Producing even more business value in less time

www.malotaux.nl/conferences

Niels Malotaux

N R Malotaux Consultancy

+31 655 753 604

niels@malotaux.nl

www.malotaux.nl

Niels Malotaux

- Team and Organizational Coach
- Expert in helping optimizing performance

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

Result Management

Schedule, we’ll try to keep ☺

Who is Who and Who is doing what ?

- Developer ?
- Tester ?
- QA ?
- Architect ?
- Product Owner ?
- Scrum Master ?
- Team Member ?
- Customer ?
- Manager ?
- Consultant ?
- Coach ?

Who’s responsible ?

Everyone in the team !

Homework

- The Goal of your current work or project
- The Definition of Success
- The most important stakeholder (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what could you do about it? - Failure is not an option!)
- What you think you should and can do *the coming week* to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- Any issues you expect with the above or otherwise with your work or project

Is there a problem ?

What made you decide to come to this workshop ?

Are you working in projects ?

- What is a project ?
  - Clear start
  - Clear end
  - Something special in between
- ETVX
  - Entry – Task – Verify – Exit

- Every project should improve something, otherwise it’s waste
- Is it clear what your project (or work) is improving ?

Are your projects successful ?

- Delivering Quality On Time ?
- 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 ?
- The essence is not *what* but *how well*

Is being on time important ?

- What is ‘on time’ ?
- Will we be on time ?
- If yes: How do we know ?
- If no: Why ?
- Failure is not an option:
  - What can we do about it ?

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

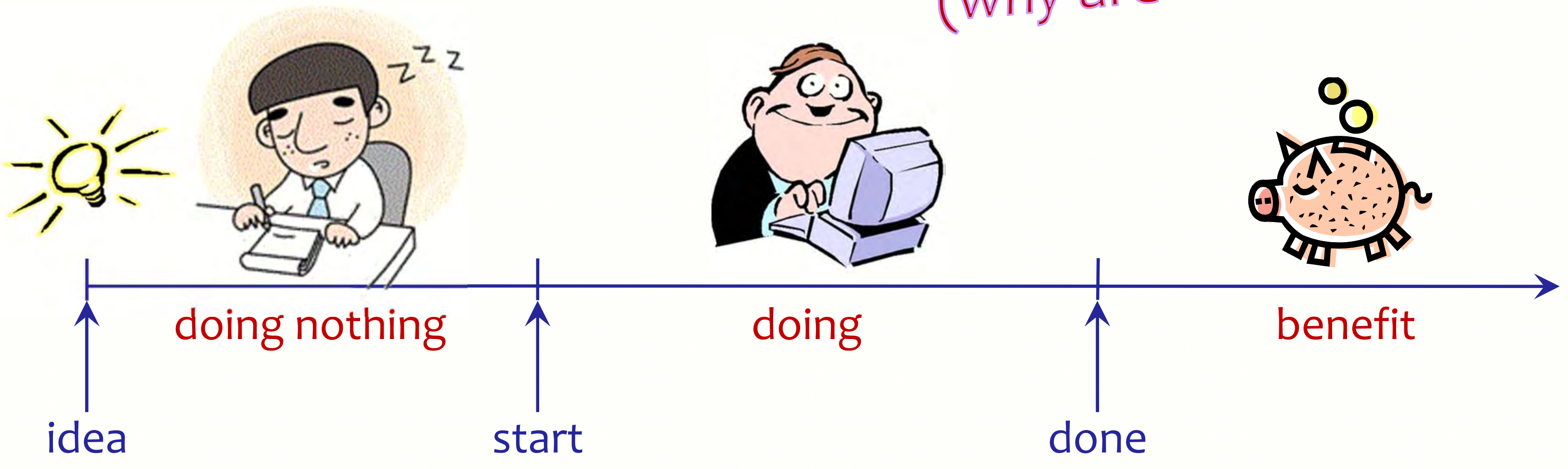
- What is the cost of the project per day ?
- Do you know how much *you* cost per day ?
  Note: that's not what you get !
- If you don't know the benefit, assume 10 times the cost
- How can you make decisions, if you don't know ?

- No need for exact numbers - it'll be a lot anyway
- Do you know the benefit of your projects ?
- Do you know the penalty for delay ?
- Who is paying for the extra time ?

The Importance of Time

Business Case

(why are we doing it)



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

Return on Investment (ROI)

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

Delivery time is a *Requirement*

- Delivery Time is a Requirement, like all other Requirements
- How come most projects are late ???
- Apparently all other Requirements
  are more important than Delivery Time
- Are they really ?
- How about your current project ?

- Can Agile be late ?
- Do you have no deadlines ?

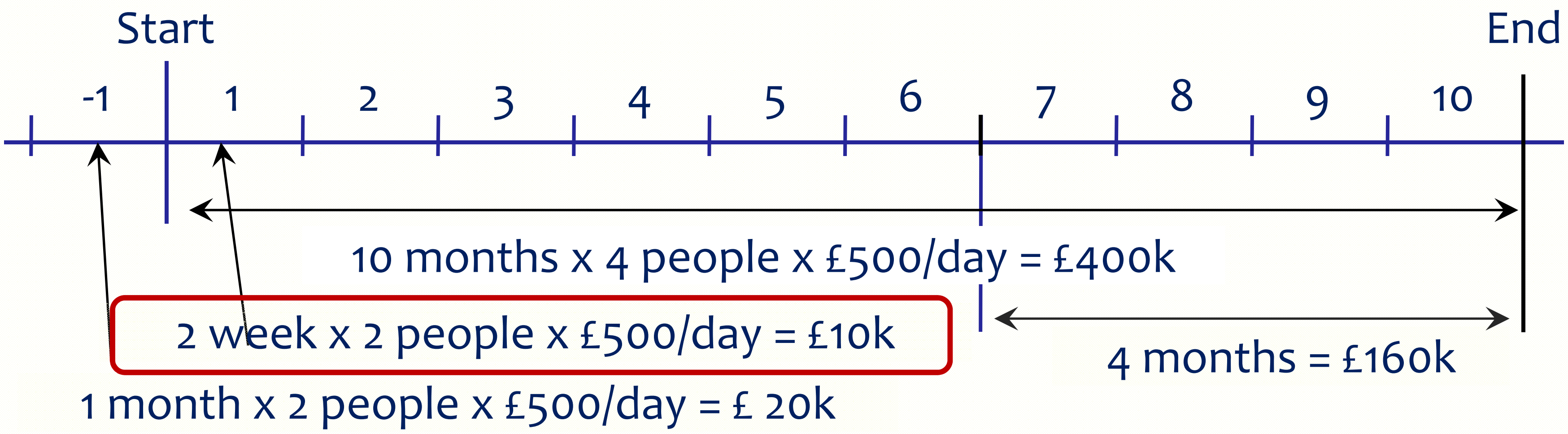
Fallacy of ‘*all*’ requirements

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

We’re Agile ! Requirements will ‘emerge’ !

- The *real* requirements don’t change
- Our *perception* of the solution may change

The Cost of Time



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

2 week x 2 people x £500/day = £10k

4 months = £160k

1 month x 2 people x £500/day = £ 20k

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

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

Did anyone tell you to go faster ?

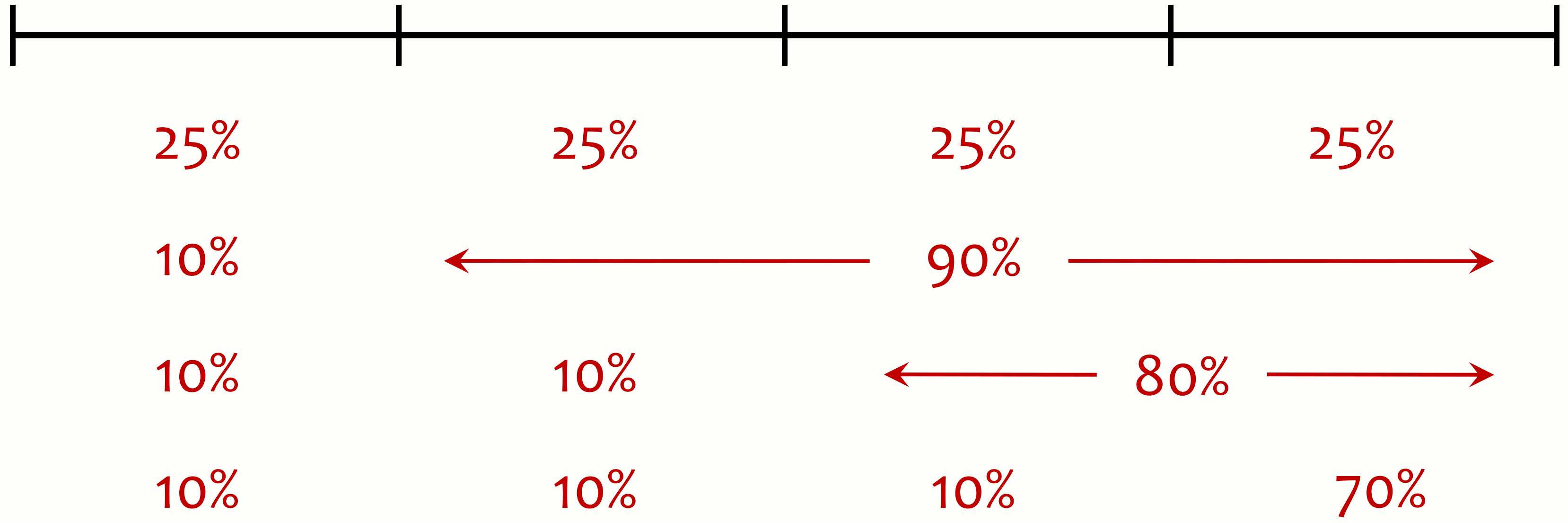
Better quality costs less

- Produce more ! → bad quality → produce less
- Produce quality ! → produce more

Quick delivery of a solution that doesn’t work means *no delivery*

The problem is: it’s counter-intuitive

4 week project



25% 25% 25% 25%

10% ← 90% →

10% 10% ← 80% →

10% 10% 10% 70%

Causes of Delay

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

Causes of causes

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

Intuition often points us in the wrong direction

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

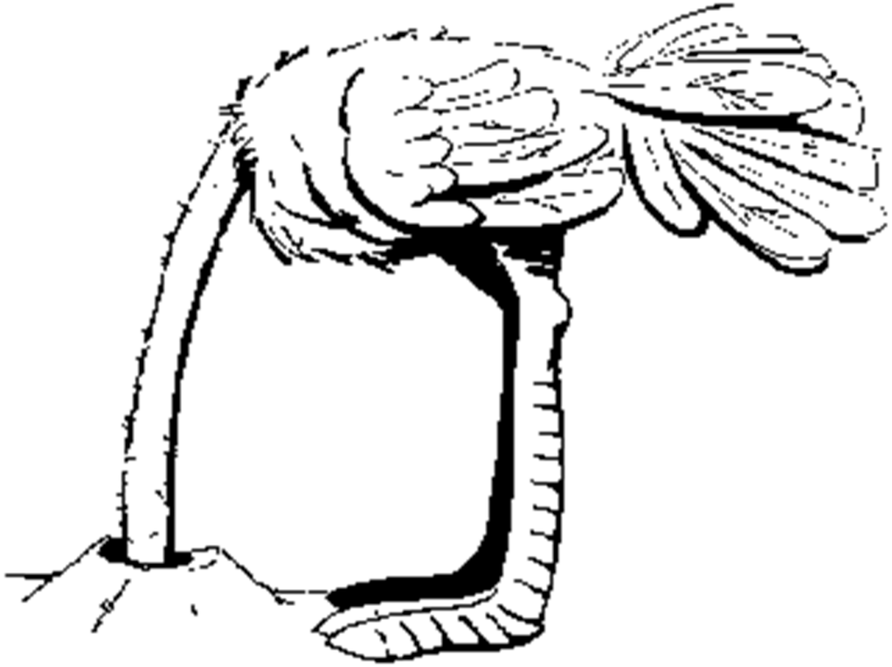
- Delivering the Right Result at the Right Time

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 ?



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

The challenge

Failure is not an option

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

Goals for these two days

- 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 workshop you don't have an excuse any more !

But you shouldn't need one either

Universal Goal

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

Evolutionary Project Management elements (Evo) – Tom Gilb

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

Why

- What
- How much
- Are we done

Zero Defects Attitude

How

Check as early as possible

Right Result

Quality On Time

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 ?

Right 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 ?

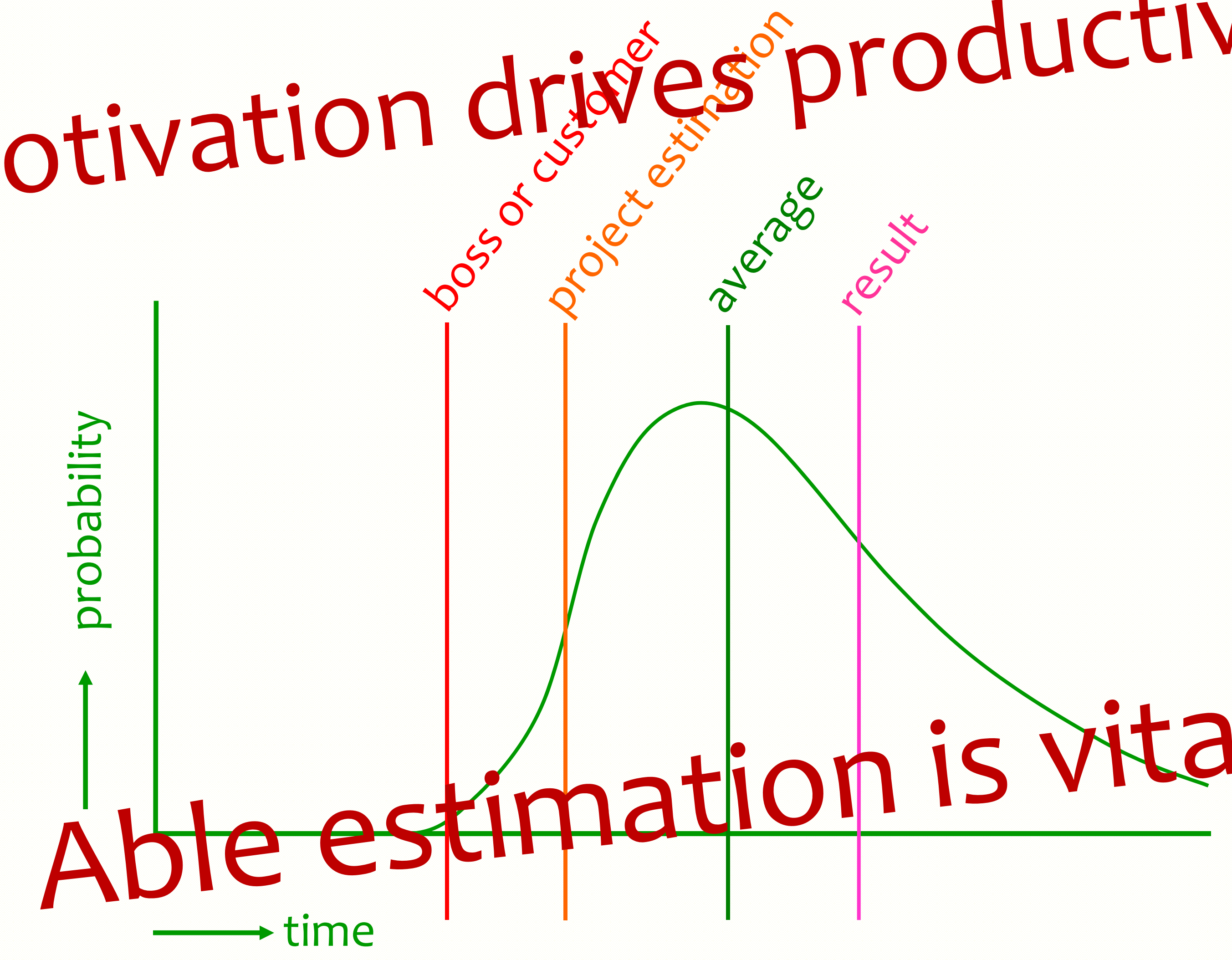
- Providing the customer with
  - what he needs
  - at the time he needs it
  - to be satisfied
  - to be more successful than before
- Constrained by (win - win)
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

- What did you find out during this exercise ?

Estimation Exercise

Lead time

Motivation drives productivity



Able estimation is vital

Estimation Exercise

Are you an optimistic or a realistic estimator?

Let’s find out !

Project:

Multiplying two numbers of 4 figures

Example

$$\begin{array}{r} \mathbf{0000} \\ \mathbf{0000}\ \times \\ \hline \mathbf{00000000} \end{array}$$

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

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

What was the real requirement ?

Assumptions, assumptions …

Better assume that many assumptions are wrong.

Check !

Elements in the exercise

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

Human Behavior

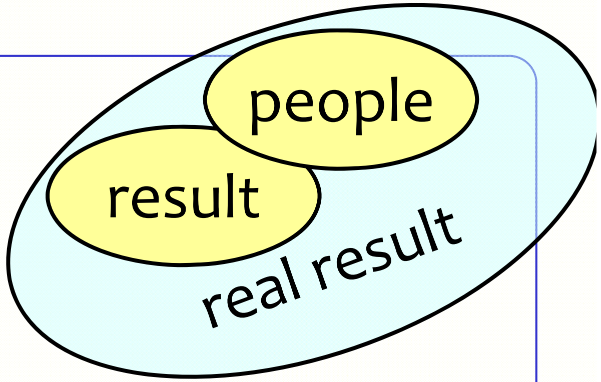
Human Behavior

- Systems are conceived, designed, implemented, maintained, used, and tolerated *(or not)* by people
- People react quite predictably
- However, often differently from what we intuitively think

- Most 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
- Actually, the humans aren’t acting unpredictably
  - Because it happens again and again
  - Human error results from physiological and psychological limitations (and capabilities !) of humans

People responsible for success

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

Discipline

- Control of wrong inclinations
- Even if we know how it should be done …
  (if nobody is watching …)
- Discipline is very difficult
- Romans 7:19
  - The good that I want to do, I do not …

→ Helping each other (watching over the shoulder)

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

→ Making mistakes (provides short window of opportunity)

→ Openness (management must learn how to cope)

Intuition

- Makes 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 (yet)
- Coaching is about redirecting intuition

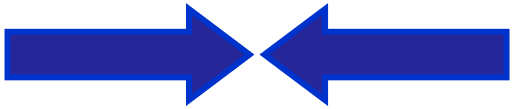
Is intuition wrong, or is the design wrong ?

Sleeping pills

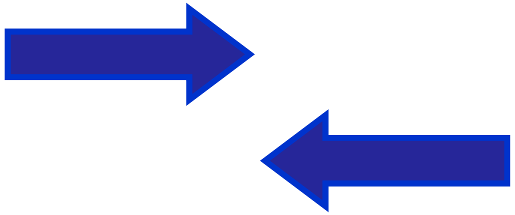
Activation pills

Communication

- Talking as near as possible past each other



To each other



Past 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)
- Intuitive understanding and insight (www.oxforddictionaries.com)
- What people say and what they do is not always the same
- The head knows, but the heart decides
- Hidden emotions are often the drivers of behavior
- Customers who said they wanted lots of different ice cream flavors from which to choose,
  still tended to buy those that were fundamentally vanilla
- So, trying to find out what the real value to the customer is, can show many paradoxes
- Better not simply believe what they say: check!

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

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

  it’s no excuse for failure of your project

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

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

People *oppose* change !

- People are not against change
- People (sub-consciously) don’t like uncertainty
- People can cope with uncertainty for a short time
- Any project changes something
  and thus introduces uncertainty

Excuses, excuses, excuses …

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

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

Excuses exercise

- What’s the most recent excuse you heard ?
- What’s the most recent excuse you used yourself ?

Mistakes, unnecessary things

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

- 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

We failed because of politics

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

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

Culture

- It failed because of the existing culture

(no good excuse !)

- Culture is the result of how people work together
- Culture can't be changed
- Culture *can* change
- By doing things differently

Don't talk *about* each other

- Talk *with* each other
- Short-Circuiting saves a lot of time

Quality

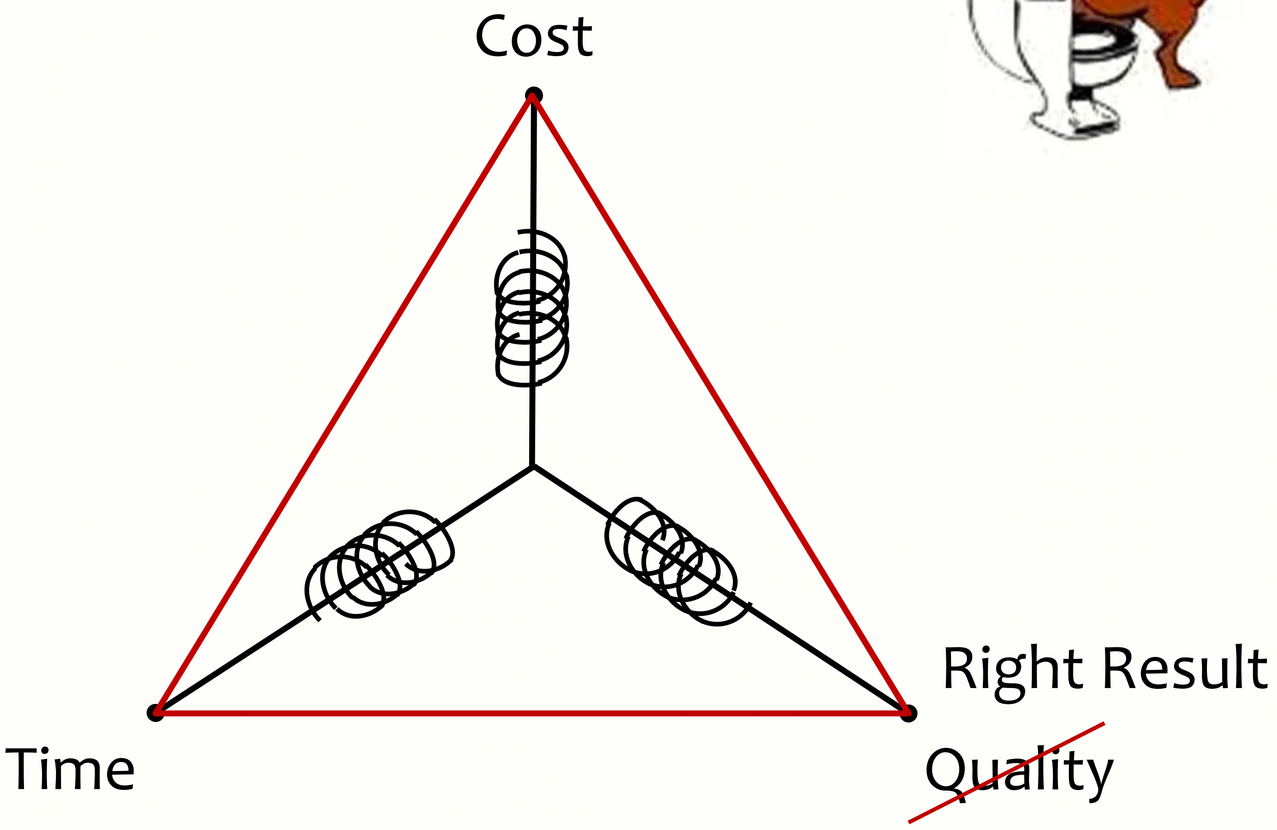
What is Quality ?

- I know it when I see it … ?

- The right result
- 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’

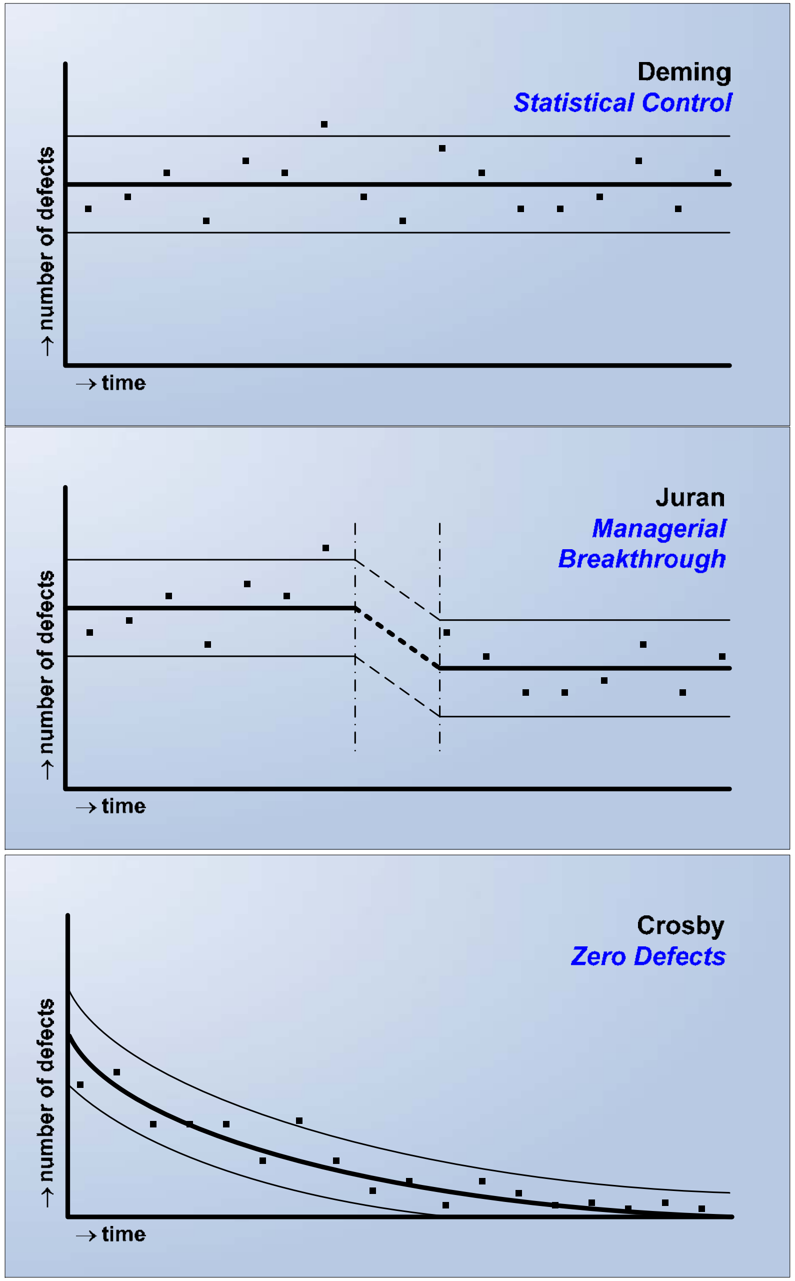


The *right* quality costs *less*

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



Do we deliver quality (value) ?

“We must deliver value !”

A project doesn’t deliver value

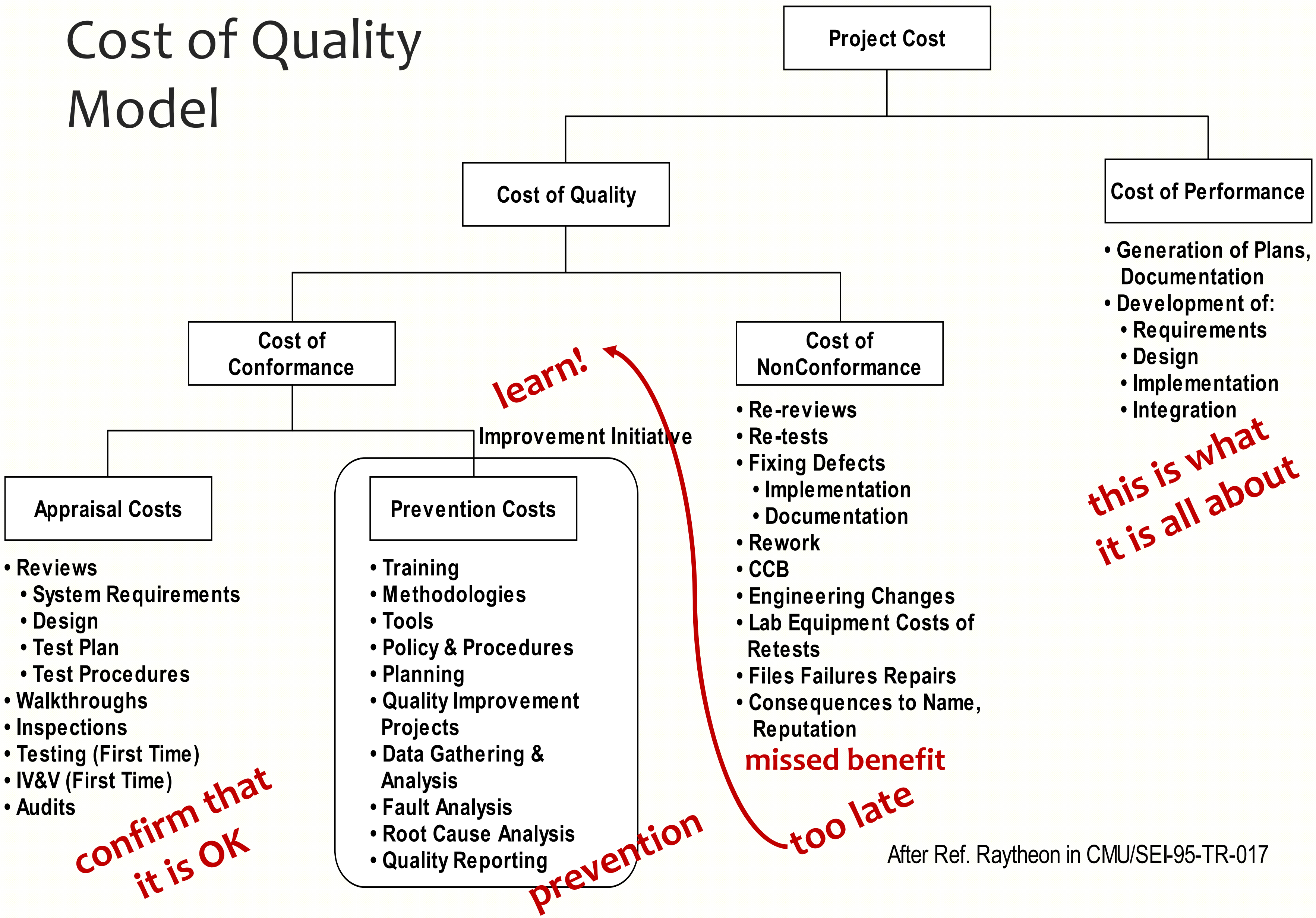
A project should create the *conditions*
for the *users* to let the quality *emerge*

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

Cost of Quality Model



Project Cost

Cost of Quality

Cost of Performance

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

this is what it is all about

Cost of Conformance

Cost of NonConformance

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

missed benefit

too late

learn!

Improvement Initiative

Appraisal Costs

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

confirm that it is OK

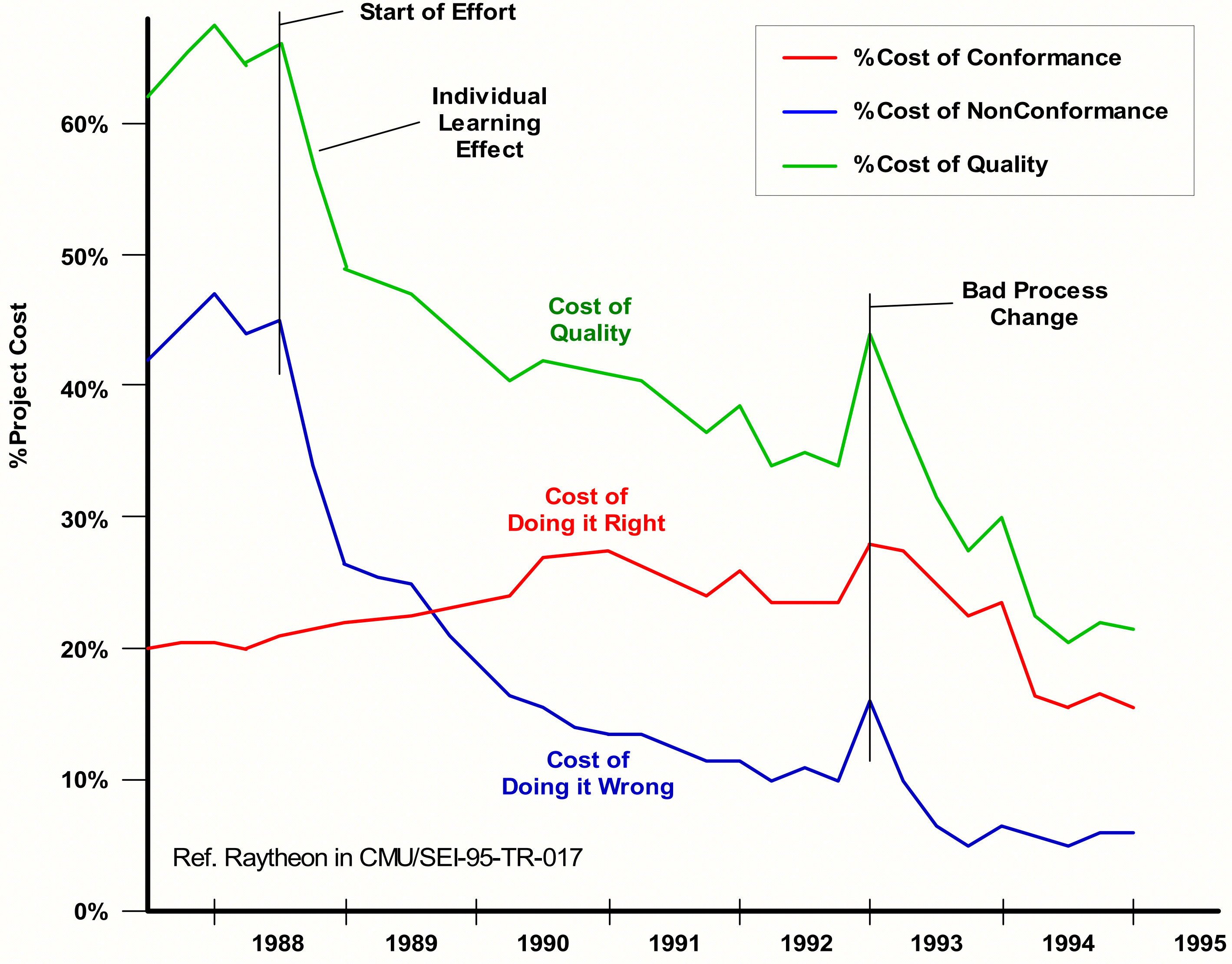
Prevention Costs

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

prevention

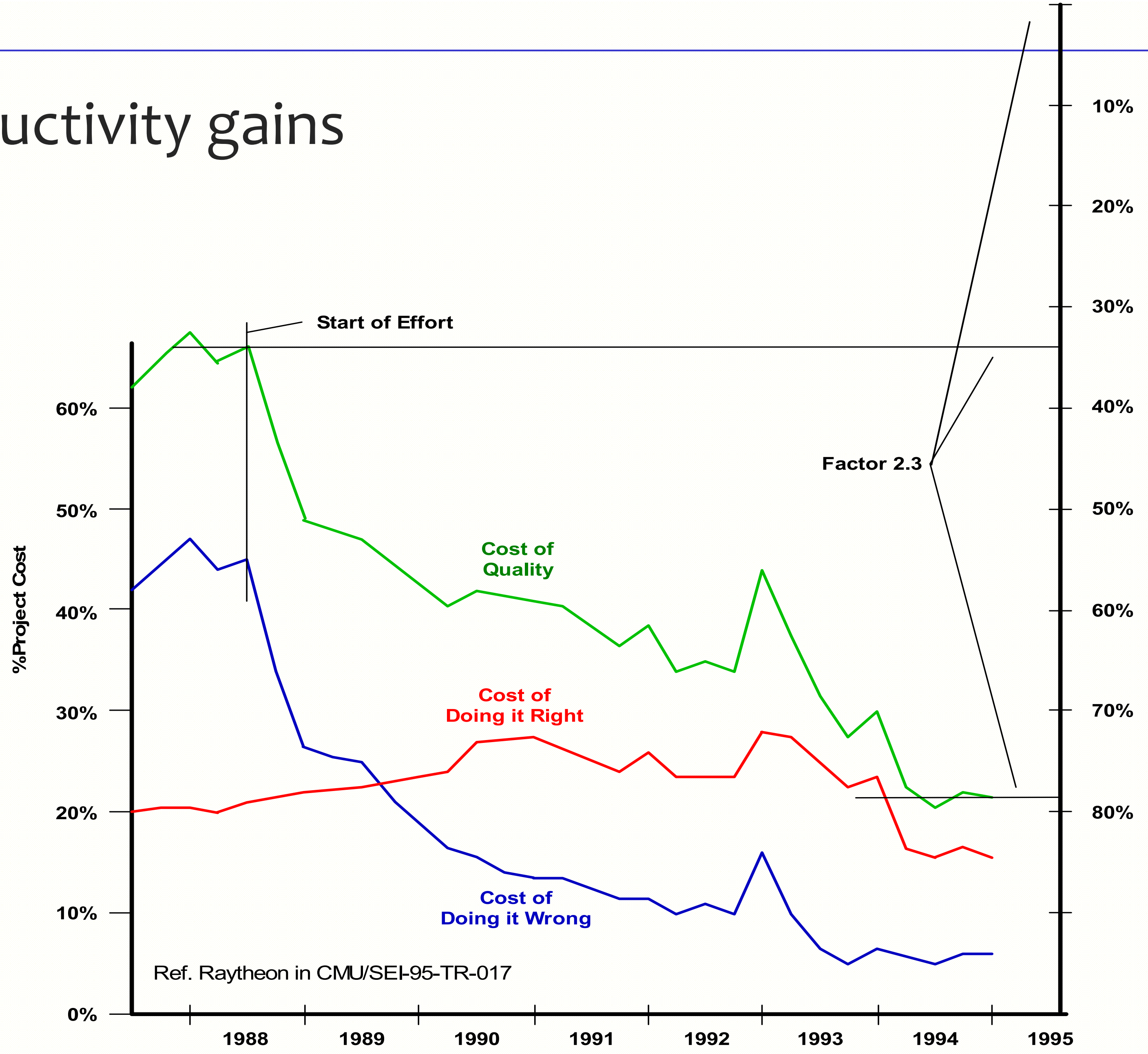
After Ref. Raytheon in CMU/SEI-95-TR-017

Cost of Quality



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

Productivity gains



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

Examples how to move towards Zero Defects

Niels Malotaux

www.malotaux.eu/conferences

Niels Malotaux:
»In my experience the ’zero defects’ attitude results in 50% less defects almost overnight.«

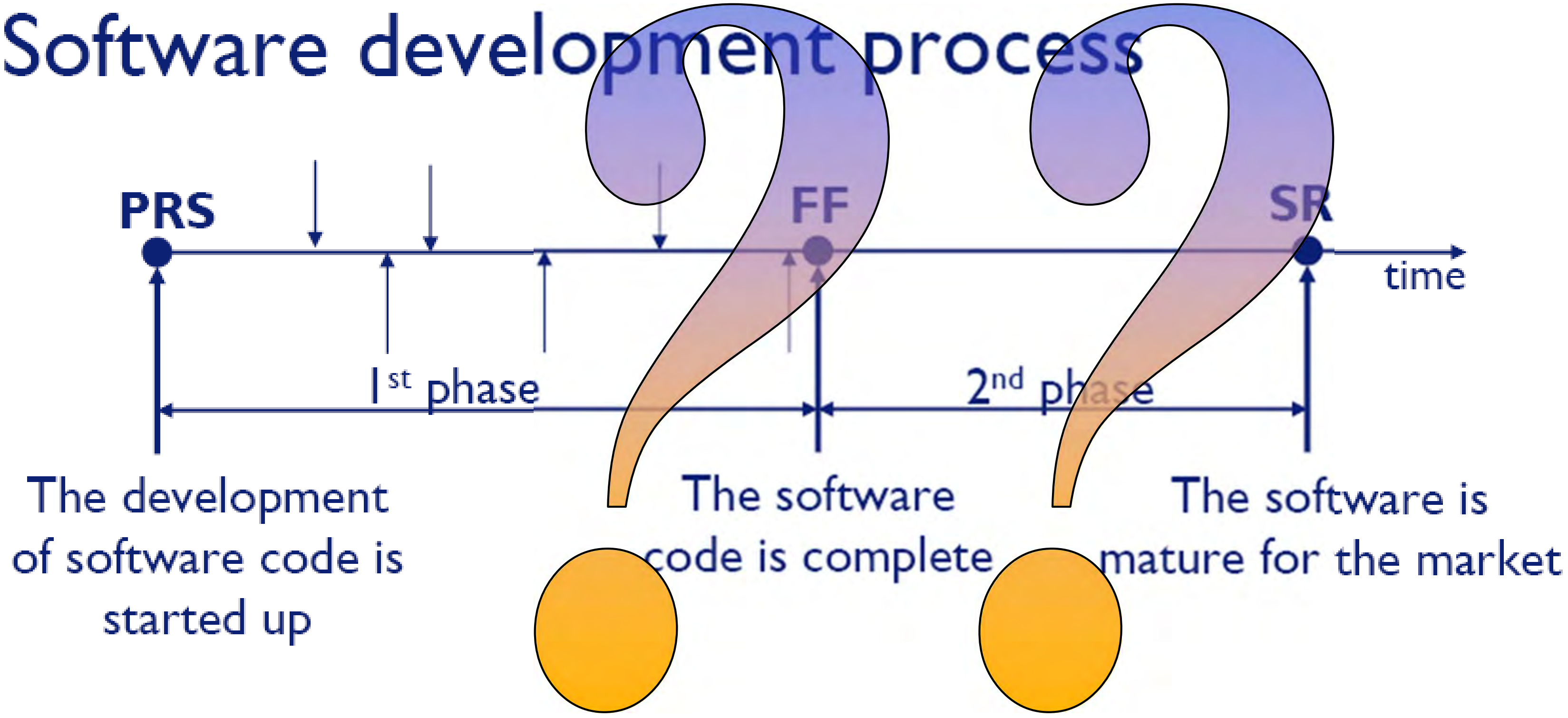
Do we deliver Zero Defect software ?

- How many defects are acceptable ?
- Do the requirements specify a certain number of defects ?
- Do you check that the required number has been produced ?

In your work

- How much time is spent putting defects in ?
- How much time is spent trying to find and fix them ?
- Do you sometimes get repeated issues ?
- How much time is spent on defect prevention ?

Software development process


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

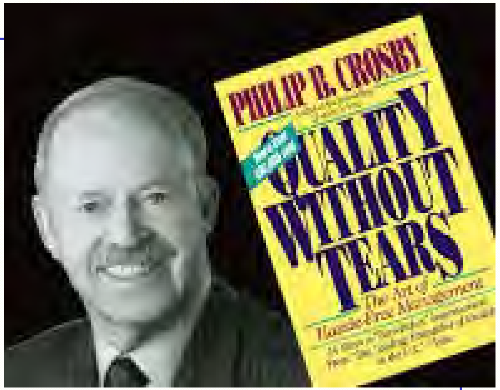
- I st phase is developing phase
- 2nd phase is de-bugging phase

Who is the (main) customer of Testing and QA ?

Deming
(1900-1993)

- Deming:
  - Quality comes not from testing, but from *improvement of the development process*
  - Testing does *not* improve quality, nor guarantee quality
  - It's too late
  - The quality, good or bad, is already in the product
  - You cannot test quality into a product
- Who is the main customer of Testing and QA ?
- What do we have to deliver to these customers ?
  What are they *waiting for* ?
- Testers and QA are *consultants* to development
- Testing and QA *shouldn't delay* the delivery - How ?

Crosby (1926-2001) - Absolutes of Quality



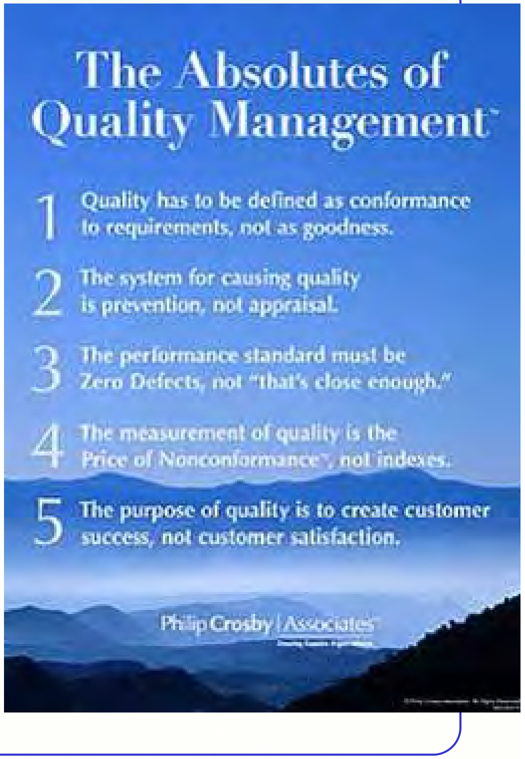
- Conformance to requirements
- Obtained through prevention
- Performance standard is zero defects
- Measured by the price of non-conformance (PONC)

Philip Crosby, 1970

- The purpose is customer success

  (not customer satisfaction)

Added by Philip Crosby Associates, 2004



The Absolutes of Quality Management™

1 Quality has to be defined as conformance to requirements, not as goodness.

2 The system for causing quality is prevention, not appraisal.

3 The performance standard must be Zero Defects, not "that's close enough."

4 The measurement of quality is the Price of Nonconformance™, not indexes.

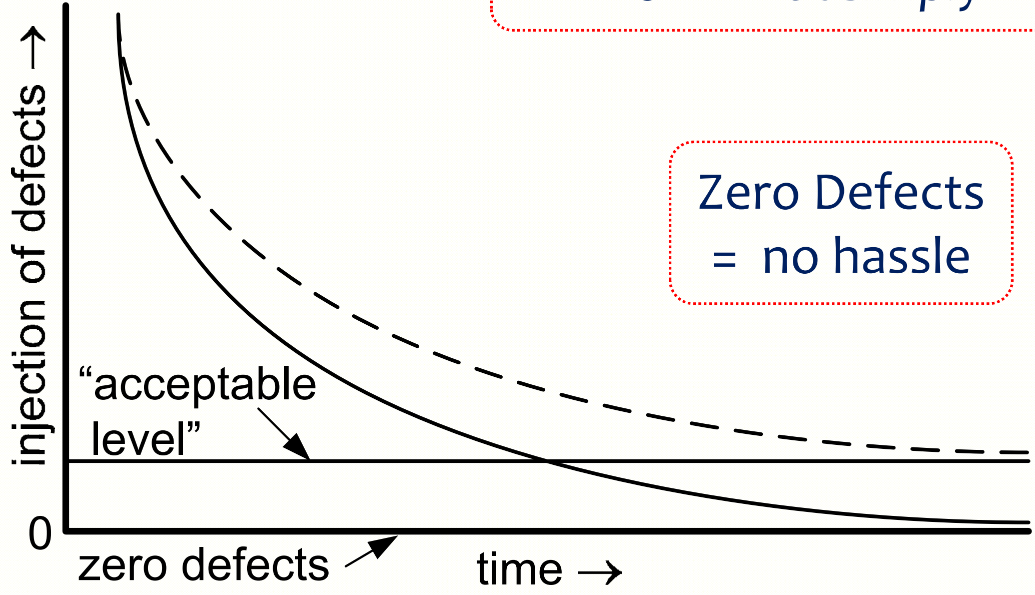
5 The purpose of quality is to create customer success, not customer satisfaction.

Philip Crosby | Associates

What is Zero Defects

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

- Zero Defects is an *asymptote*



Zero Defects
= no hassle

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

Conformance to requirements

- We meet the agreed requirements

but …

- 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

Ultimate Goal of a What We Do

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

Philip Crosby

[Quality is Still Free]

- Conventional wisdom says that error is inevitable
- As long as the performance standard requires it,
  then this self-fulfilling prophecy 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 your developers get upset ?*)
- 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)

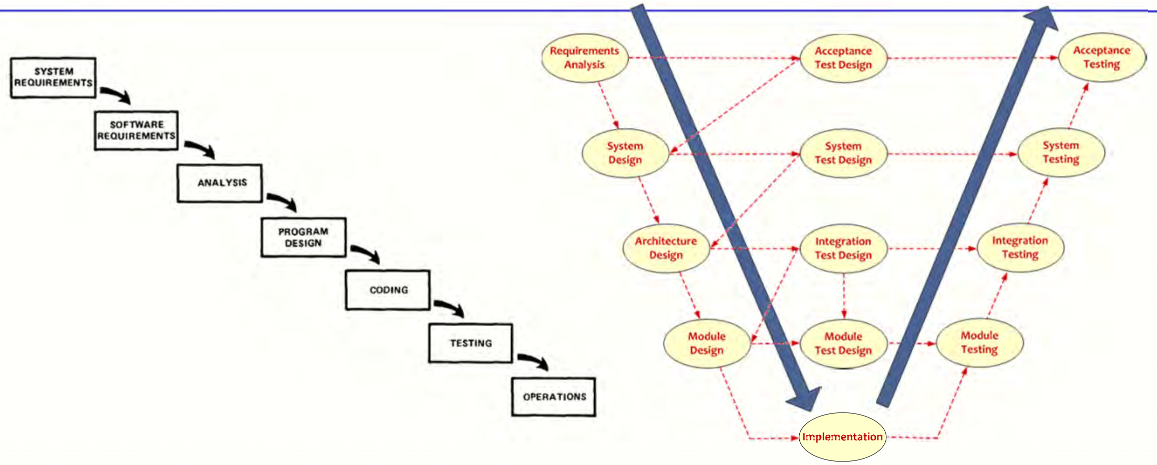
Prevention: Root Cause Analysis

- Is Root Cause Analysis routinely performed – *every time* ?
- What is the Root Cause of a defect ?

- Cause:
  The error that caused the defect
- Root Cause:
  What *caused us* to make the error that caused the defect

- Without proper Root Cause Analysis ,
  we're *doomed to repeat the same errors*

W-model



SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

ANALYSIS

PROGRAM DESIGN

CODING

TESTING

OPERATIONS

Requirements Analysis

Acceptance Test Design

Acceptance Testing

System Design

System Test Design

System Testing

Architecture Design

Integration Test Design

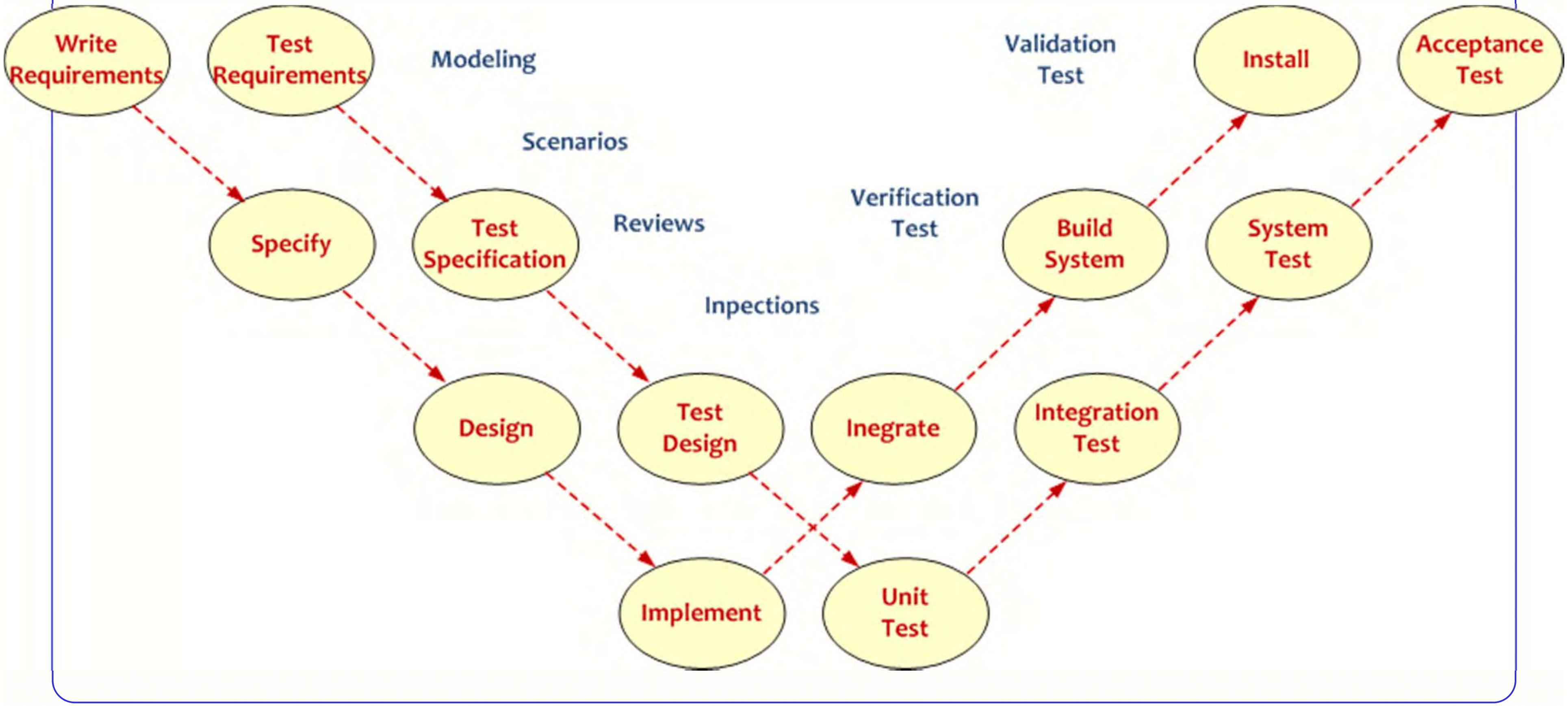
Integration Testing

Module Design

Module Test Design

Module Testing

Implementation



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

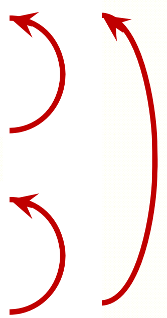
Unit Test

Some Examples

We're not perfect,
but the customer shouldn't find out

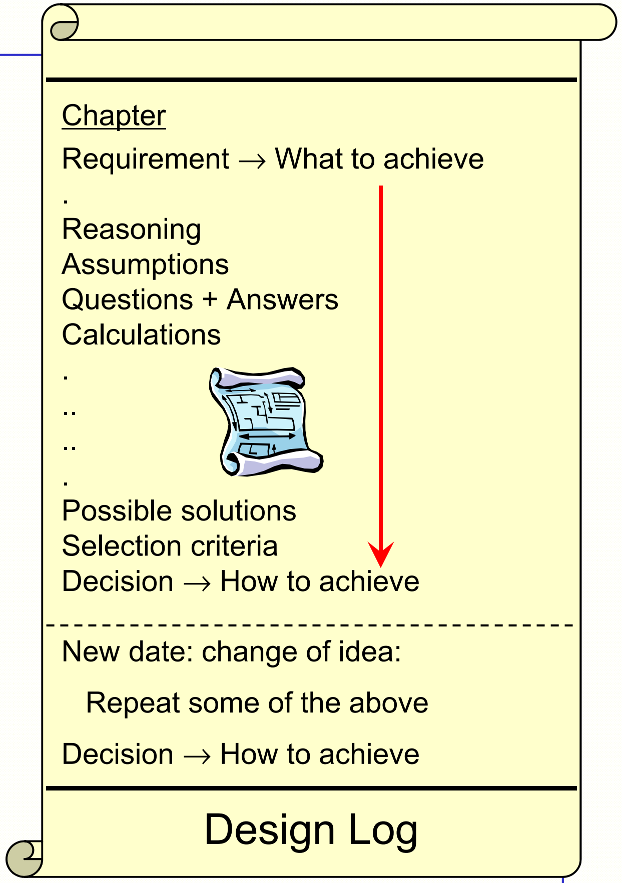
Design techniques

- Design
- Review
- Code
- Review



Iterate as needed

- Test (no questions, no issues)
- If issue in test: no Band-Aid: start all over again:
  Review: What’s wrong with the design ?
- Reconstruct the design (if there is no design)



Chapter

Requirement → What to achieve

.

Reasoning

Assumptions

Questions + Answers

Calculations

.

..

..

.

Possible solutions

Selection criteria

Decision → How to achieve

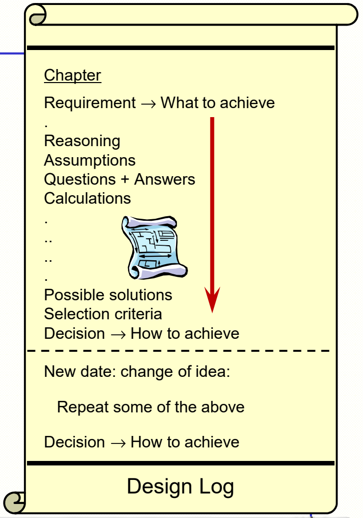
New date: change of idea:

Repeat some of the above

Decision → How to achieve

Design Log

Case: In the pub



Chapter

Requirement → What to achieve

.

Reasoning

Assumptions

Questions + Answers

Calculations

.

..

..

.

Possible solutions

Selection criteria

Decision → How to achieve

New date: change of idea:

Repeat some of the above

Decision → How to achieve

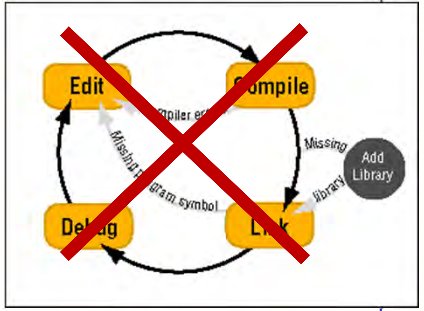
Design Log

James:

Niels, this is Louise

Louise, this is Niels, who taught me about

DesignLogging - Tell what happened



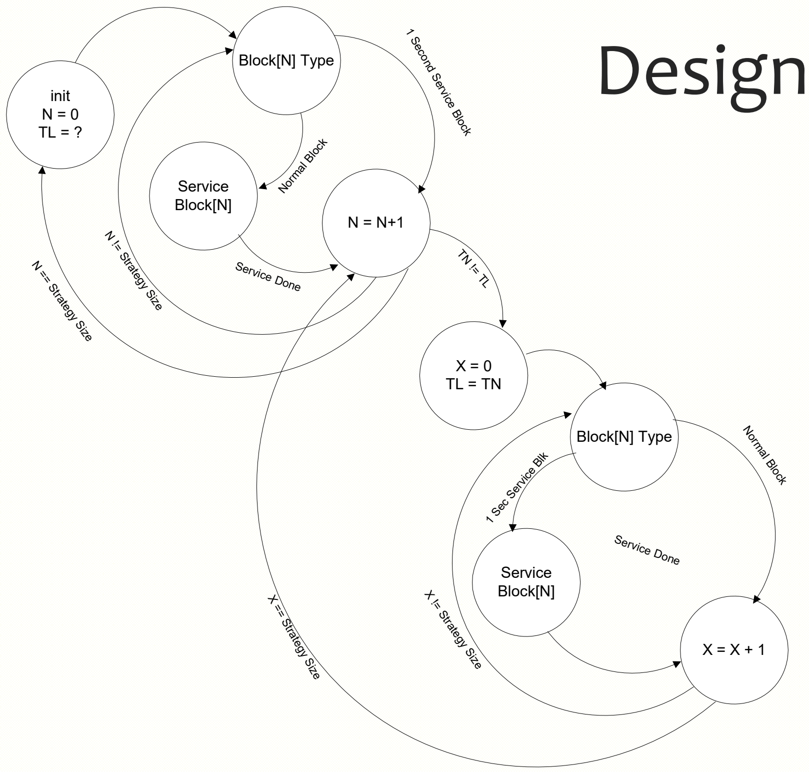
Louise:

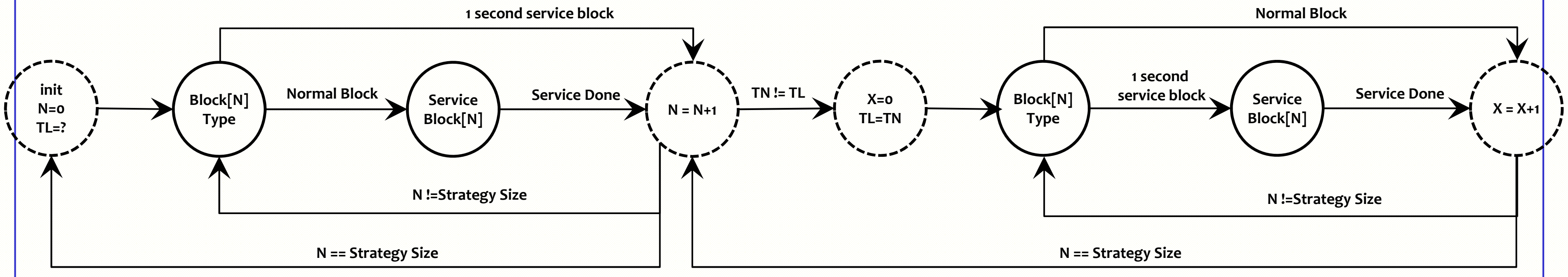
- *We had only 7 days to finish some software*
- *We were working hard, coding, testing, coding, testing*
- *James said we should stop coding and go back to the design*
- *"We don't have time !" - "We've only 7 days !"*
- *James insisted*
- *We designed, found the problem, corrected it, cleaned up the mess*
- *Done in less than 7 days*
- *Thank you!*

What James told me recently

- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a ‘final’ review, with another colleague
- Based in his expertise, *the solution was completely reworked*
- Actually, two features were delivered and deployed
  - One that was design and code reviewed had no issues after deployment
  - Other one, was the source of quite some defects
- In summary, this success has proved instrumental in buy-in for DesignLogs which are now embedded in the development process

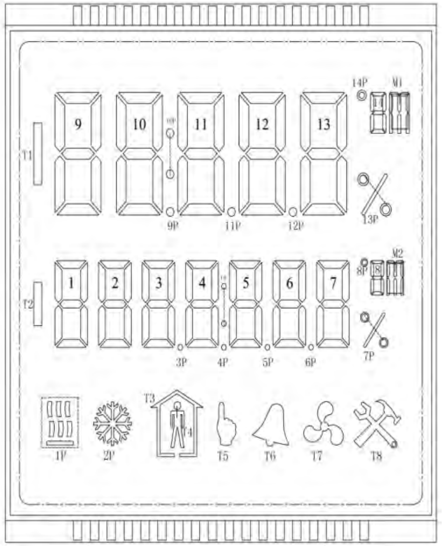
Design can be done in many ways

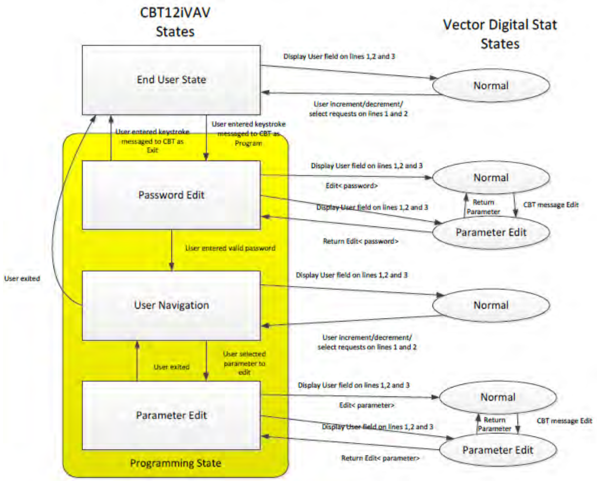




Sorry, some pictures removed for confidentiality

47 page interface description





CBT12iVAV States

Vector Digital Stat States

End User State

Normal

Display User field on lines 1,2 and 3

User increment/decrement/ select requests on lines 1 and 2

User entered keystroke messaged to CBT as Exit

User entered keystroke messaged to CBT as Program

Password Edit

Display User field on lines 1,2 and 3

Edit< password>

Normal

Return Parameter

CBT message Edit

Display User field on lines 1,2 and 3

Parameter Edit

Return Edit< password>

User entered valid password

User exited

User Navigation

Display User field on lines 1,2 and 3

Normal

User increment/decrement/ select requests on lines 1 and 2

User exited

User selected parameter to edit

Parameter Edit

Display User field on lines 1,2 and 3

Edit< parameter>

Normal

Return Parameter

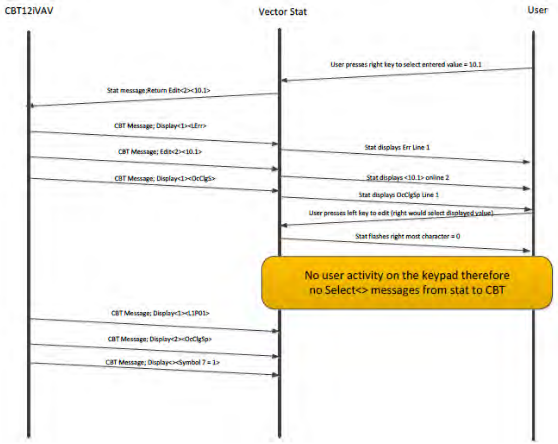
CBT message Edit

Display User field on lines 1,2 and 3

Parameter Edit

Return Edit< parameter>

Programming State



CBT12iVAV

Vector Stat

User

User presses right key to select entered value = 10.1

Stat message;Return Edit<2><10.1>

CBT Message; Display<1><LErr>

CBT Message; Edit<2><10.1>

Stat displays Err Line 1

CBT Message; Display<1><OcClgS>

Stat displays <10.1> online 2

Stat displays OcClgSp Line 1

User presses left key to edit (right would select displayed value)

Stat flashes right most character = 0

No user activity on the keypad therefore no Select<> messages from stat to CBT

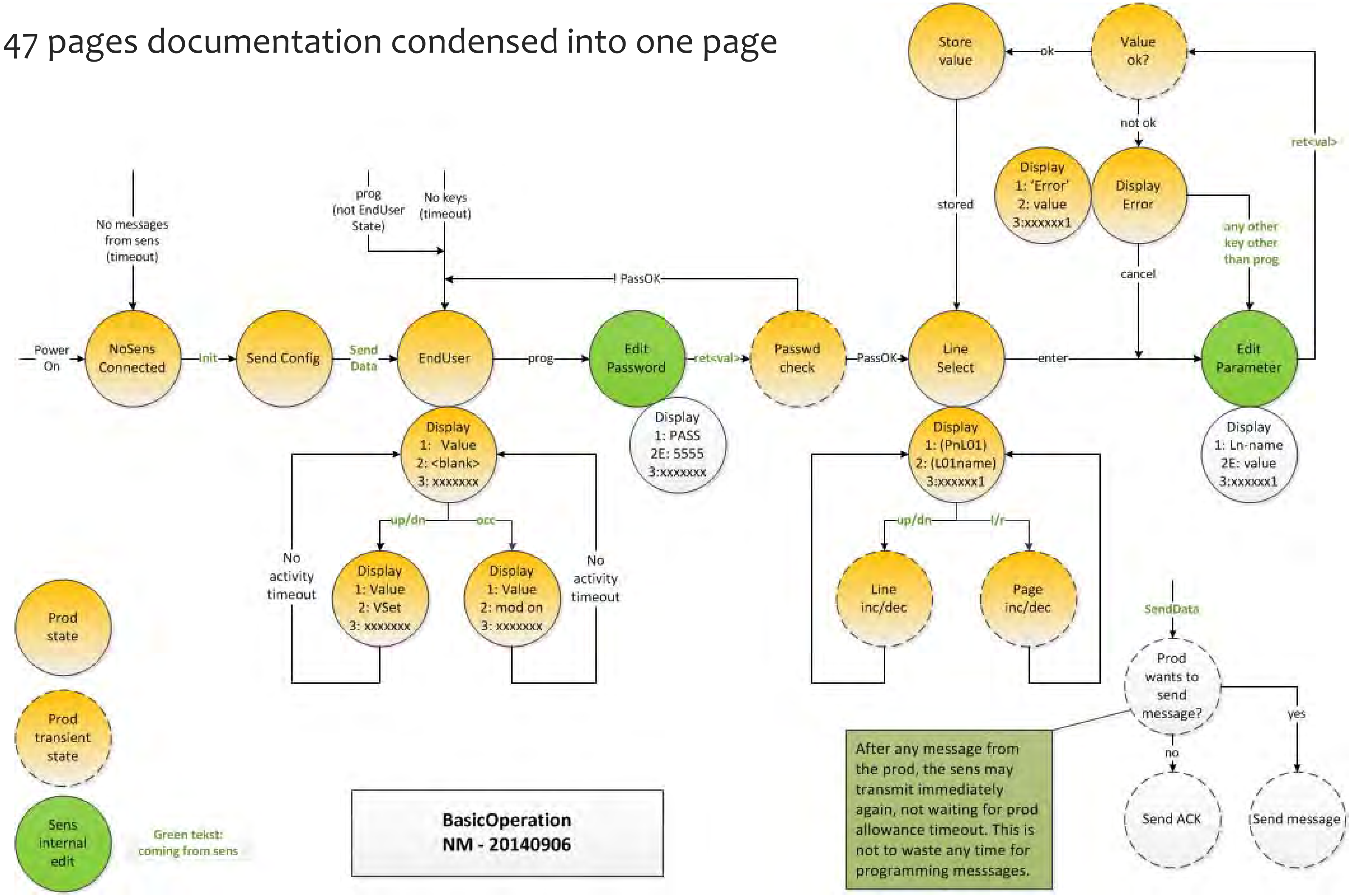
CBT Message; Display<1><L1P01>

CBT Message; Display<2><OcClgSp>

CBT Message; Display<><Symbol 7 = 1>

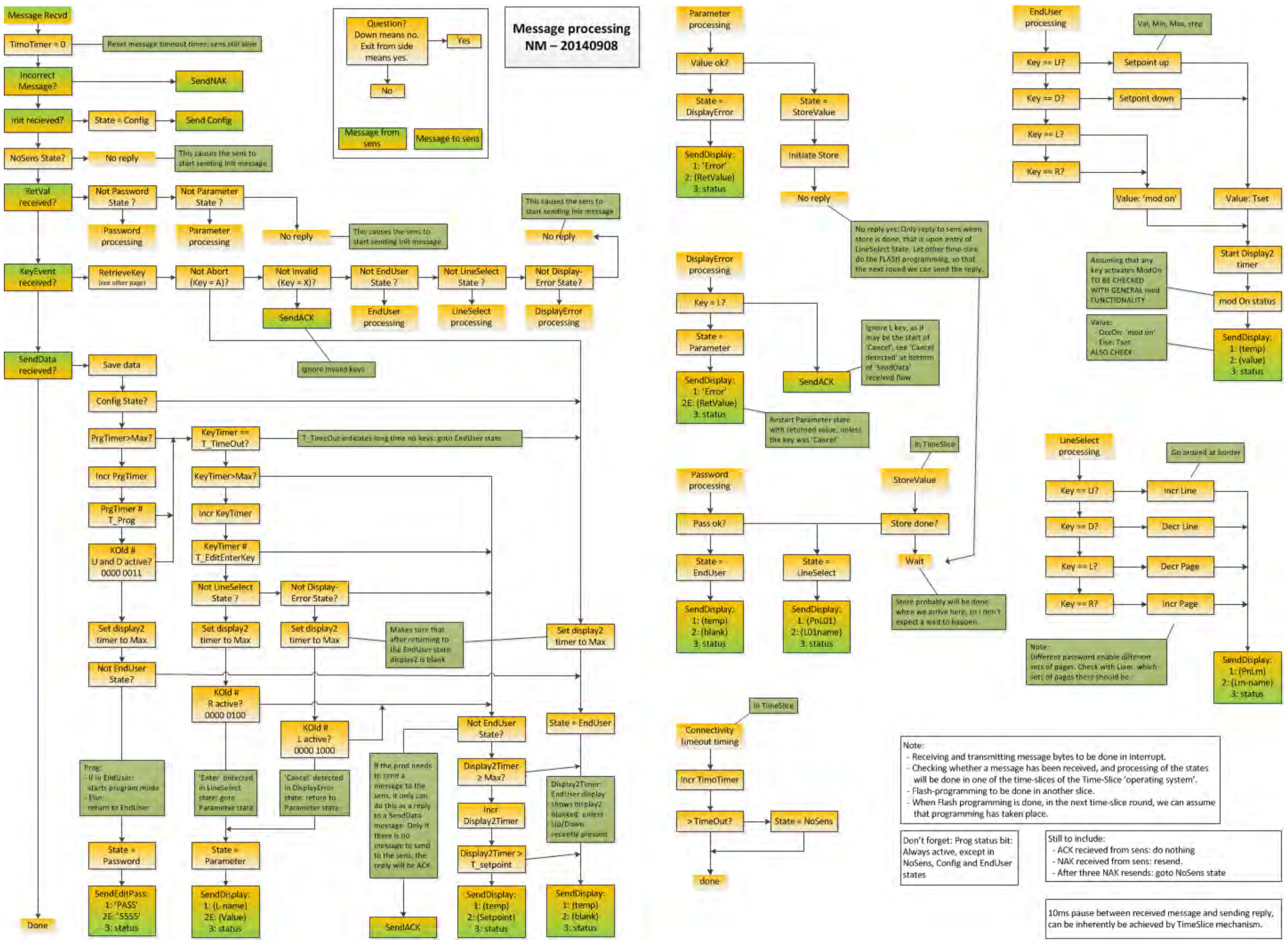
Choose the appropriate design

47 pages documentation condensed into one page



After any message from the prod, the sens may transmit immediately again, not waiting for prod allowance timeout. This is not to waste any time for programming messsages.

Design example



Message processing
NM – 20140908

Question?
Down means no.
Exit from side
means yes.
Yes
No
Message from sens
Message to sens

Message Recvd
TimoTimer = 0
Reset message timeout timer: sens still alive
Incorrect Message?
SendNAK
Init recieved?
State = Config
Send Config
NoSens State?
No reply
This causes the sens to start sending Init message
RetVal received?
Not Password State ?
Password processing
Not Parameter State ?
Parameter processing
No reply
This causes the sens to start sending Init message
KeyEvent received?
RetrieveKey (see other page)
Not Abort (Key = A)?
Not Invalid (Key = X)?
SendACK
Ignore invalid keys
Not EndUser State ?
EndUser processing
Not LineSelect State ?
LineSelect processing
Not Display-Error State?
DisplayError processing
No reply
This causes the sens to start sending Init message

SendData recieved?
Save data
Config State?
PrgTimer>Max?
Incr PrgTimer
PrgTimer # T_Prog
KOld # U and D active? 0000 0011
Set display2 timer to Max
Not EndUser State?
Prog:
- If in EndUser: starts program mode
- Else: return to EndUser
State = Password
SendEditPass:
1: 'PASS'
2E: '5555'
3: status
KeyTimer == T_TimeOut?
T_TimeOut indicates long time no keys: goto EndUser state
KeyTimer>Max?
Incr KeyTimer
KeyTimer # T_EditEnterKey
Not LineSelect State ?
Set display2 timer to Max
KOld # R active? 0000 0100
'Enter' detected in LineSelect state: goto Parameter state
State = Parameter
SendDisplay:
1: (L-name)
2E: (Value)
3: status
Not Display-Error State?
Set display2 timer to Max
KOld # L active? 0000 1000
'Cancel' detected in DisplayError state: return to Parameter state
Makes sure that after returning to the EndUser state, display2 is blank
If the prod needs to send a message to the sens, it only can do this as a reply to a SendData message. Only if there is no message to send to the sens, the reply will be ACK
SendACK
Not EndUser State?
Display2Timer ≥ Max?
Incr Display2Timer
Display2Timer > T_setpoint
SendDisplay:
1: (temp)
2: (Setpoint)
3: status
Set display2 timer to Max
State = EndUser
Display2Timer: EndUser display shows display2 blanked: unless Up/Down recently pressed
SendDisplay:
1: (temp)
2: (blank)
3: status
Done

Parameter processing
Value ok?
State = DisplayError
SendDisplay:
1: 'Error'
2: (RetValue)
3: status
State = StoreValue
Initiate Store
No reply
No reply yet: Only reply to sens when store is done, that is upon entry of LineSelect State. Let other time-slice do the FLASH programming, so that the next round we can send the reply.

DisplayError processing
Key = L?
State = Parameter
SendDisplay:
1: 'Error'
2E: (RetValue)
3: status
Restart Parameter state with returned value, unless the key was 'Cancel'
SendACK
Ignore L key, as it may be the start of 'Cancel', see 'Cancel detected' at bottom of 'SendData' received flow.

Password processing
Pass ok?
State = EndUser
SendDisplay:
1: (temp)
2: (blank)
3: status
State = LineSelect
SendDisplay:
1: (PnL01)
2: (L01name)
3: status

In TimeSlice
StoreValue
Store done?
Wait
Store probably will be done when we arrive here, so I don't expect a wait to happen.

In TimeSlice
Connectivity timeout timing
Incr TimoTimer
> TimeOut?
State = NoSens
done

EndUser processing
Key == U?
Setpoint up
Val, Min, Max, step
Key == D?
Setpont down
Key == L?
Key == R?
Value: 'mod on'
Value: Tset
Start Display2 timer
mod On status
Assuming that any key activates ModOn TO BE CHECKED WITH GENERAL mod FUNCTIONALITY
SendDisplay:
1: (temp)
2: (value)
3: status
Value:
- OccOn: 'mod on'
- Else: Tset
ALSO CHECK

LineSelect processing
Key == U?
Incr Line
Go around at border
Key == D?
Decr Line
Key == L?
Decr Page
Key == R?
Incr Page
Note:
Different password enable different sets of pages. Check with Liam, which sets of pages there should be
SendDisplay:
1: (PnLm)
2: (Lm-name)
3: status

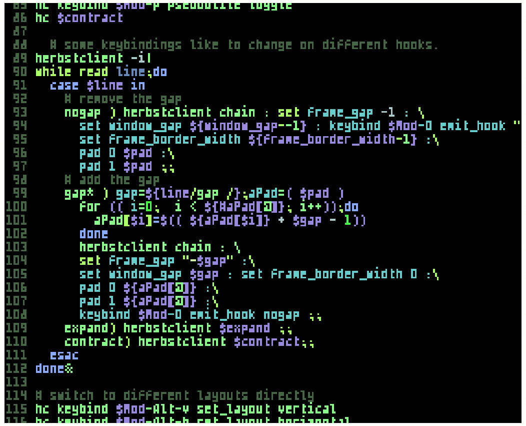
Note:
- Receiving and transmitting message bytes to be done in interrupt.
- Checking whether a message has been received, and processing of the states will be done in one of the time-slices of the Time-Slice 'operating system'.
- Flash-programming to be done in another slice.
- When Flash programming is done, in the next time-slice round, we can assume that programming has taken place.

Don't forget: Prog status bit: Always active, except in NoSens, Config and EndUser states

Still to include:
- ACK recieved from sens: do nothing
- NAK received from sens: resend.
- After three NAK resends: goto NoSens state

10ms pause between received message and sending reply, can be inherently be achieved by TimeSlice mechanism.

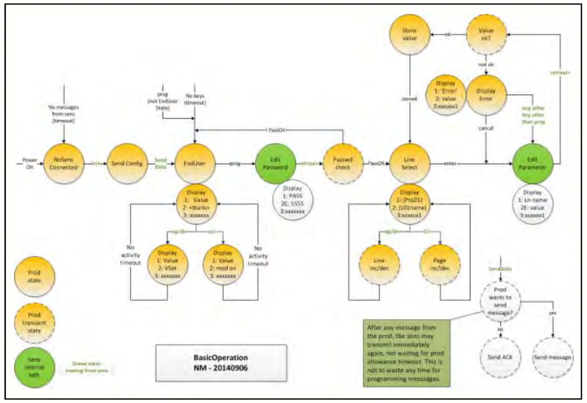
What is better than reviewing code ?



```
hc keybind $Mod-p pseudotile toggle
hc $contract

  # some keybindings like to change on different hooks.
herbstclient -i|
while read line;do
  case $line in
    # remove the gap
    nogap ) herbstclient chain : set frame_gap -1 : \
      set window_gap ${window_gap--1} : keybind $Mod-0 emit_hook "
      set frame_border_width ${frame_border_width-1} :\
      pad 0 $pad :\
      pad 1 $pad ;;
    # add the gap
    gap* ) gap=${line/gap /};aPad=( $pad )
      for (( i=0;  i < ${#aPad[@]}; i++));do
        aPad[$i]=$(( ${aPad[$i]} + $gap - 1))
      done
      herbstclient chain : \
      set frame_gap "-$gap" :\
      set window_gap $gap : set frame_border_width 0 :\
      pad 0 ${aPad[@]} :\
      pad 1 ${aPad[@]} :\
      keybind $Mod-0 emit_hook nogap ;;
    expand) herbstclient $expand ;;
    contract) herbstclient $contract;;
  esac
done&

# switch to different layouts directly
hc keybind $Mod-Alt-v set_layout vertical
hc keybind $Mod-Alt-h set_layout horizontal
```

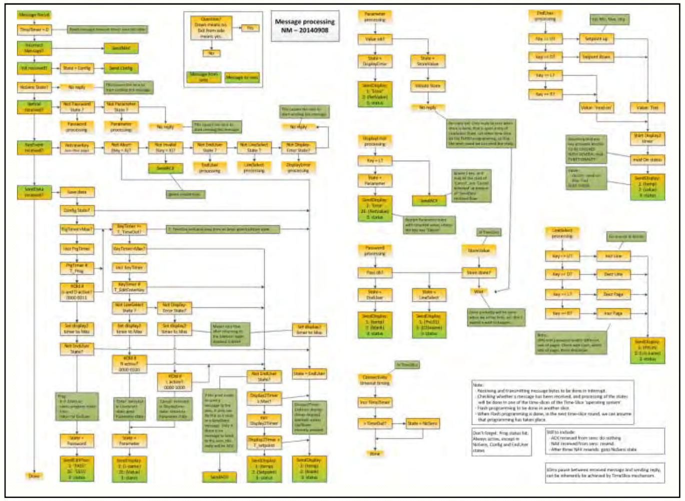
- Do you ever review software ?
- What do you review ?

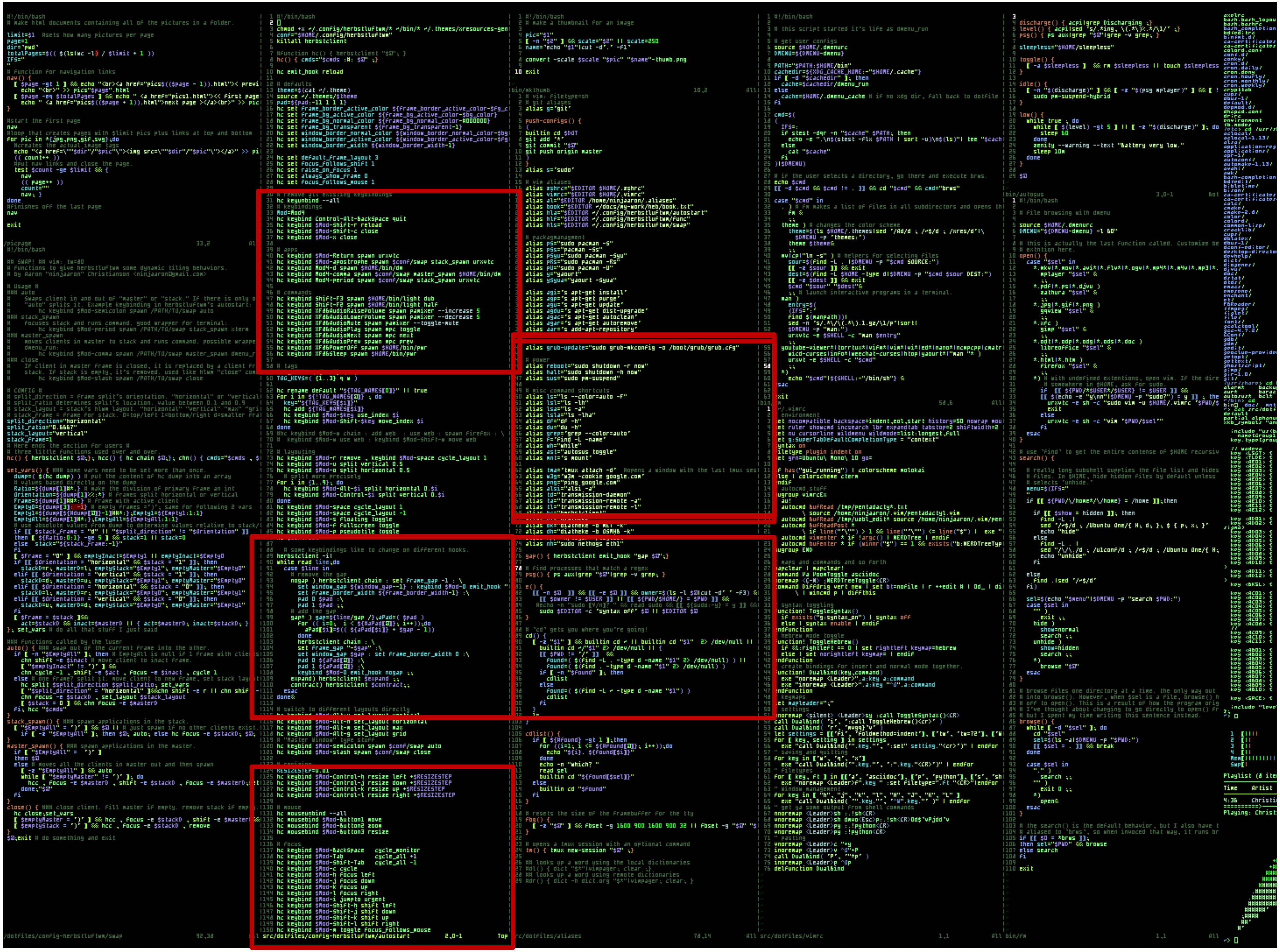
- What is better than reviewing code ?
  - May I review the design first ?



After any message from the prod, the sens may transmit immediately again, not waiting for prod allowance timeout. This is not to waste any time for programming messsages.

BasicOperation
NM - 20140906





```
#!/bin/bash
# make html documents containing all of the pictures in a folder.

limit=$1  #sets how many pictures per page
page=1
dir=`pwd`
totalPages=$(( $(ls|wc -l) / $limit + 1 ))
IFS="
"
# function for navigation links
nav() {
  [ $page -gt 1 ] && echo "<br><a href="pics$(($page - 1)).html">< previ
    echo "<br>" >> pics"$page".html
  [ $page -eq $totalPages ] && echo " <a href="pics1.html"><< first page
    echo " <a href="pics$(($page + 1)).html">next page ></a><br>" >> pic
}

#start the first page
nav
#loop that creates pages with $limit pics plus links at top and bottom
for pic in *{jpg,png,gif,svg};do
  #creates the actual image tags
  echo "<a href=\""$dir"/"$pic"\"><img src=\""$dir"/"$pic"\"></a>" >> pi
  (( count++ ))
  #put nav links and close the page.
  test $count -ge $limit && {
    nav
    (( page++ ))
    count=""
    nav; }
done
#finishes off the last page
nav

exit


/picpage                                              33,2          Al
#!/bin/bash

## SWAP! ## vim: tw=80
# functions to give herbstluftwm some dynamic tiling behaviors.
# by Aaron "ninjaaron" Christianson <ninjaaron@gmail.com>

# Usage #
### auto
#    Swaps client in and out of "master" or "stack." If there is only o
#    "auto" splits it. Example keybinding in herbstluftwm's autostart:
#        hc keybind $Mod-semicolon spawn /PATH/TO/swap auto
### stack_spawn
#    Focuses stack and runs command. good wrapper for terminal:
#        hc keybind $Mod-period spawn /PATH/TO/swap stack_spawn xterm
### master_spawn
#    moves clients in master to stack and runs command. possible wrappe
#    dmenu_run:
#        hc keybind $Mod-comma spawn /PATH/TO/swap master_spawn dmenu_r
### close
#    If client in master frame is closed, it is replaced by a client fr
#    stack. If stack is empty, it's removed. used like hlwm "close" com
#        hc keybind $Mod-slash spawn /PATH/TO/swap close

# CONFIG #
# split_direction = frame split's orientation. "horizontal" or "vertical
# split_ratio determines split's location. value between 0.1 and 0.9
# stack_layout = stack's hlwm layout. "horizontal" "vertical" "max" "gri
# stack_frame = frame for stack. 0=top/left 1=bottom/right d=smaller fra
split_direction="horizontal"
split_ratio="0.6667"
stack_layout="vertical"
stack_frame=1
# Here ends the section for users #
# three little functions used over and over.
hc() { herbstclient $@;}; hcc() { hc chain $@;}; chn() { cmds="$cmds , $

set_vars() { ### some vars need to be set more than once.
  dump=( $(hc dump) ) # put the content of hc dump into an array
  # values based directly on the dump
  Ratio=${dump[1]#*.} # make the division of primary frame an int
  Orientation=${dump[1]%%:*} # frames split horizontal or vertical
  Frame=${dump[1]##*:} # frame with active client
  Empty0=${dump[3]: -1} # empty frames =')'; same for following 2 vars
  Empty1=${dump[${#dump[@]}-1]##*:};Empty1=${Empty1:1:1}
  EmptyAll=${dump[1]#*:};EmptyAll=${EmptyAll:1:1}
  # use absolute values from dump to determine values relative to stack/
  if [[ $stack_frame = "d" || "$split_direction" != "$Orientation" ]]
  then [ ${Ratio:0:1} -ge 5 ] && stack=1 || stack=0
  else  stack="${stack_frame:-1}"
  fi
  [ $Frame = "0" ] && emptyInact=$Empty1 || emptyInact=$Empty0
  if [[ $Orientation = "horizontal" && $stack = "1" ]]; then
    stackD=r; masterD=l; emptyStack="$Empty1"; emptyMaster="$Empty0"
  elif [[ $Orientation = "vertical" && $stack = "1" ]]; then
    stackD=d; masterD=u; emptyStack="$Empty1"; emptyMaster="$Empty0"
  elif [[ $Orientation = "horizontal" && $stack = "0" ]]; then
    stackD=l; masterD=r; emptyStack="$Empty0"; emptyMaster="$Empty1"
  elif [[ $Orientation = "vertical" && $stack = "0" ]]; then
    stackD=u; masterD=d; emptyStack="$Empty0"; emptyMaster="$Empty1"
  fi
  [ $Frame = $stack ]&&
    act=$stackD && inact=$masterD || { act=$masterD; inact=$stackD; }
}; set_vars # do all that stuff I just said

### functions called by the luser
auto() { ### swap out of the current frame into the other.
  if [ -n "$EmptyAll" ]; then # EmptyAll is null if 1 frame with clie ts
    chn shift -e $inact # move client to inact frame.
    [ "$emptyInact" != ')' ] &&
    chn cycle -1 , shift -e $act , focus -e $inact , cycle 1
  else # one frame? split it. move client to new frame, set stack lay ut
    hc split $split_direction $split_ratio; set_vars
    [ "$split_direction" = "horizontal" ]&&chn shift -e r || chn shif  -
    chn focus -e $stackD , set_layout $stack_layout
    [ $stack = 0 ] && chn focus -e $masterD
  fi; hcc "$cmds"
}
stack_spawn() { ### spawn applications in the stack.
  [ "$EmptyAll" = ')' ] && $@ || # just spawn if no other clients exist.
    if [ -z "$EmptyAll" ]; then $@; auto; else hc focus -e $stackD; $@;
}
master_spawn() { ### spawn applications in the master.
  if [ "$EmptyAll" =  ')' ]
  then $@
  else # moves all the clients in master out and then spawn
    [ -z "$EmptyAll" ] && auto
    while [ "$emptyMaster" != ')' ]; do
      hcc , focus -e $masterD , shift -e $stackD , focus -e $masterD; et
    done;"$@"
  fi
}
close() { ### close client. fill master if empty. remove stack if emp y.
  hc close;set_vars
  [ $emptyMaster = ')' ] && hcc , focus -e $stackD , shift -e $master &&
  [ $emptyStack = ')' ] && hcc , focus -e $stackD , remove
}
$@;exit # do something and exit




/dotfiles/config-herbstluftwm/swap                          92,38
```

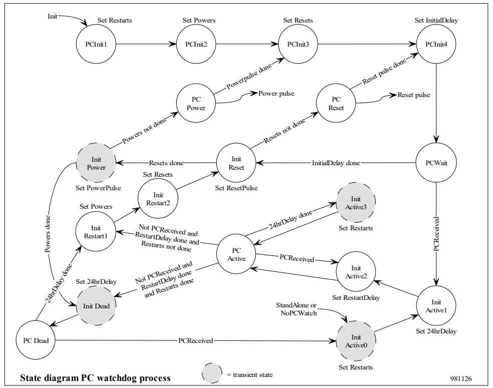
```
#!/bin/bash

chmod +x ~/.config/herbstluftwm/* ~/bin/* ~/.themes/xresources-gen
conf="$HOME/.config/herbstluftwm"
killall herbstclient

#function hc() { herbstclient "$@"; }
hc() { cmds="$cmds :#: $@" ;}

hc emit_hook reload

# defaults
theme=$(cat ~/.theme)
source ~/.themes/$theme
pad=${pad:-11 1 1 1}
hc set frame_border_active_color ${frame_border_active_color-$fg_c
hc set frame_bg_active_color ${frame_bg_active_color-$bg_color}
hc set frame_bg_normal_color ${frame_bg_normal_color-#000000}
hc set frame_bg_transparent ${frame_bg_transparent-1}
hc set window_border_normal_color ${window_border_normal_color-$bg
hc set window_border_active_color ${window_border_active_color-$fg
hc set window_border_width ${window_border_width-1}

hc set default_frame_layout 3
hc set focus_follows_shift 1
hc set raise_on_focus 1
hc set always_show_frame 0
hc set focus_follows_mouse 1

# remove all existing keybindings
hc keyunbind --all
# keybindings
Mod=Mod4
hc keybind Control-Alt-BackSpace quit
hc keybind $Mod-Shift-r reload
hc keybind $Mod-Shift-c close
hc keybind $Mod-x close

# apps
hc keybind $Mod-Return spawn urxvtc
hc keybind $Mod-apostrophe spawn $conf/swap stack_spawn urxvtc
hc keybind Mod4-d spawn $HOME/bin/dm
hc keybind Mod4-comma spawn $conf/swap master_spawn $HOME/bin/dm
hc keybind Mod4-period spawn $conf/swap stack_spawn urxvtc

# commands
hc keybind Shift-F3 spawn $HOME/bin/light dub
hc keybind Shift-F2 spawn $HOME/bin/light half
hc keybind XF86AudioRaiseVolume spawn pamixer --increase 5
hc keybind XF86AudioLowerVolume spawn pamixer --decrease 5
hc keybind XF86AudioMute spawn pamixer --toggle-mute
hc keybind XF86AudioPlay spawn mpc toggle
hc keybind XF86AudioNext spawn mpc next
hc keybind XF86AudioPrev spawn mpc prev
hc keybind XF86PowerOff spawn $HOME/bin/pwr
hc keybind XF86Sleep spawn $HOME/bin/pwr

# tags

TAG_KEYS=( {1..3} q w )

hc rename default "${TAG_NAMES[0]}" || true
for i in ${!TAG_NAMES[@]} ; do
  key="${TAG_KEYS[$i]}"
  hc add ${TAG_NAMES[$i]}
  hc keybind $Mod-$key use_index $i
  hc keybind $Mod-Shift-$key move_index $i
done
#hc keybind $Mod-w chain : add web  : use web : spawn firefox : \
#  keybind $Mod-w use web : keybind $Mod-Shift-w move web

# layouting
hc keybind $Mod-r remove , keybind $Mod-space cycle_layout 1
hc keybind $Mod-u split vertical 0.5
hc keybind $Mod-o split horizontal 0.5
  # split more precisely
for i in {1..9}; do
  hc keybind $Mod-Alt-$i split horizontal 0.$i
  hc keybind $Mod-Control-$i split vertical 0.$i
done
hc keybind $Mod-space cycle_layout 1
hc keybind $Mod-space cycle_layout -1
hc keybind $Mod-s floating toggle
hc keybind $Mod-f fullscreen toggle
hc keybind $Mod-p pseudotile toggle


  # some keybindings like to change on different hooks.
herbstclient -il
while read line;do
  case $line in
    # remove the gap
    nogap ) herbstclient chain : set frame_gap -1 : \
      set window_gap ${window_gap--1} : keybind $Mod-0 emit_hook "
      set frame_border_width ${frame_border_width-1} :\
      pad 0 $pad :\
      pad 1 $pad ;;
    # add the gap
    gap* ) gap=${line/gap /};aPad=( $pad )
      for (( i=0;  i < ${#aPad[@]}; i++ ));do
        aPad[$i]=$(( ${aPad[$i]} + $gap - 1))
      done
      herbstclient chain : \
      set frame_gap "-$gap" :\
      set window_gap $gap : set frame_border_width 0 :\
      pad 0 ${aPad[@]} :\
      pad 1 ${aPad[@]} :\
      keybind $Mod-0 emit_hook nogap ;;
    expand) herbstclient $expand ;;
    contract) herbstclient $contract;;
  esac
done&

# switch to different layouts directly
hc keybind $Mod-Alt-v set_layout vertical
hc keybind $Mod-Alt-h set_layout horizontal
hc keybind $Mod-Alt-m set_layout max
hc keybind $Mod-Alt-g set_layout grid
# "Master Window" type stuff
hc keybind $Mod-semicolon spawn $conf/swap auto
hc keybind $Mod-slash spawn $conf/swap close

# resizing
RESIZESTEP=0.01
hc keybind $Mod-Control-h resize left +$RESIZESTEP
hc keybind $Mod-Control-j resize down +$RESIZESTEP
hc keybind $Mod-Control-k resize up +$RESIZESTEP
hc keybind $Mod-Control-l resize right +$RESIZESTEP

# mouse
hc mouseunbind --all
hc mousebind $Mod-Button1 move
hc mousebind $Mod-Button2 zoom
hc mousebind $Mod-Button3 resize

# focus
hc keybind $Mod-BackSpace   cycle_monitor
hc keybind $Mod-Tab         cycle_all +1
hc keybind $Mod-Shift-Tab   cycle_all -1
hc keybind $Mod-c cycle
hc keybind $Mod-h focus left
hc keybind $Mod-j focus down
hc keybind $Mod-k focus up
hc keybind $Mod-l focus right
hc keybind $Mod-i jumpto urgent
hc keybind $Mod-Shift-h shift left
hc keybind $Mod-Shift-j shift down
hc keybind $Mod-Shift-k shift up
hc keybind $Mod-Shift-l shift right
hc keybind $Mod-m toggle focus_follows_mouse
src/dotfiles/config-herbstluftwm/autostart          2,0-1          Top
```

```
#!/bin/bash
# make a thumbnail for an image

pic="$1"
[ -n "$2" ] && scale="$2" || scale=250
name=`echo "$1"|cut -d'.' -f1`

convert -scale $scale "$pic" "$name"-thumb.png

exit


bin/mkthumb                                         10,2          All
# vim: filetype=sh
# git aliases
alias g='git'

push-configs() {
(
builtin cd $DOT
git add '*'
git commit "$@"
git push origin master
)
}
alias s='sudo'

# vim aliases
alias zshrc="$EDITOR $HOME/.zshrc"
alias vimrc="$EDITOR $HOME/.vimrc"
alias al="$EDITOR /home/ninjaaron/.aliases"
alias book="$EDITOR ~/docs/my-work/heb/book.txt"
alias hla="$EDITOR ~/.config/herbstluftwm/autostart"
alias hlf="$EDITOR ~/.config/herbstluftwm/func"
alias hls="$EDITOR ~/.config/herbstluftwm/swap"

# packagmanagment
alias pS="sudo pacman -S"
alias pSs="pacman -Ss"
alias pSyu="sudo pacman -Syu"
alias pRs="sudo pacman -Rs"
alias pU="sudo pacman -U"
alias y="yaourt"
alias ySyua="yaourt -Syua"

alias agi='s apt-get install'
alias agp='s apt-get purge'
alias agu='s apt-get update'
alias agdu='s apt-get dist-upgrade'
alias agac='s apt-get autoclean'
alias agar='s apt-get autoremove'
alias aar='s add-apt-repository'

alias grub-update="sudo grub-mkconfig -o /boot/grub/grub.cfg"

# power
alias reboot="sudo shutdown -r now"
alias halt="sudo shutdown -h now"
alias sus="sudo pm-suspend"

# misc command shortcuts
alias ls="ls --color=auto -F"
alias lsl="ls -lh"
alias lsa="ls -a"
alias lsla="ls -lha"
alias df="df -h"
alias du="du -h"
alias grep='grep --color=auto'
alias f='find -L -name'
alias wh="while"
alias ast="autosus toggle"
alias mnt='s mount'

alias tma='tmux attach -d'  #opens a window with the last tmux ses
alias w3g='w3m -cookie google.com'
alias png="ping google.com"
alias alsi="alsi -a"
alias td="transmission-daemon"
alias ta="transmission-remote -a"
alias tl="transmission-remote -l"
alias hc="herbstclient"

alias dk="diatheke -b NET -k"
alias dh="diatheke -b OSMHB -k"

alias nh="sudo nethogs eth1"

gap() { herbstclient emit_hook "gap $@";}

# find processes that match a regex
psg() { ps aux|grep "$@"|grep -v grep; }

e() {
  [[ -n $@  ]] && [[ -e $@ ]] && owner=$(ls -l $@|cut -d' ' -f3) &
    [[ $owner != $USER ]] || [[ ${PWD/$HOME/} = $PWD ]] &&
   #echo -n "sudo [Y/n]? " && read sudo && [[ ${sudo:-y} = y ]] &&
    sudo $EDITOR -c 'syntax off' $@ || $EDITOR $@
}

# `cd` gets you where you're going!
cd() {
  [ -z "$1" ] && builtin cd ~ || builtin cd "$1"  2> /dev/null ||
    builtin cd ~/"$1" 2> /dev/null || {
    [[ $PWD != '/' ]]  &&
      found=( $(find -L . -type d -name "$1" 2> /dev/null) ) ||
      found=( $(find . -type d -name "$1" 2> /dev/null) )
    if [ -n "$found" ]; then
      cdlist
    else
      found=( $(find -L ~ -type d -name "$1") )
      cdlist
    fi
  }
  ls
}

cdlist() {
  if [ ${#found} -gt 1 ];then
    for ((i=1; i <= ${#found[@]}; i++));do
      echo "${i}. ${found[$i]}"
    done
    echo -n "which? "
    read sel
    builtin cd "${found[$sel]}"
  else
    builtin cd "$found"
  fi
}

# resets the size of the framebuffer for the tty
fbg() {
  [ -z "$@" ] && fbset -g 1600 900 1600 900 32 || fbset -g "$@" "$
}

# opens a tmux session with an optional command
tm() { tmux new-session "$@" ;}

## looks up a word using the local dictionaries
#dl() { dict "$*"|vimpager; clear ;}
## looks up a word using remote dictionaries
#dr() { dict -h dict.org "$*"|vimpager; clear; }



src/dotfiles/aliases                                 78,14          All
```

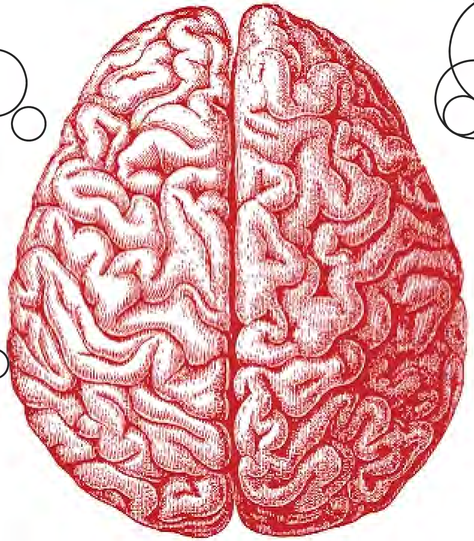
```
#!/bin/bash

# this script started it's life as dmenu_run

# get user configs
source $HOME/.dmenurc
DMENU=${DMENU-dmenu}

PATH="$PATH:$HOME/bin"
cachedir=${XDG_CACHE_HOME:-"$HOME/.cache"}
if [ -d "$cachedir" ]; then
  cache=$cachedir/dmenu_run
else
  cache=$HOME/.dmenu_cache # if no xdg dir, fall back to dotfile
fi

cmd=$(
(
  IFS=:
  if stest -dqr -n "$cache" $PATH; then
    echo -e ".\n$(stest -flx $PATH | sort -u)\n$(ls)"| tee "$cach
  else
    cat "$cache"
  fi
)|$DMENU)

# if the user selects a directory, go there and execute brws.
echo $cmd
[[ -d $cmd && $cmd != . ]] && cd "$cmd" && cmd="brws"

case "$cmd" in
  . ) # fm makes a list of files in all subdirectors and opens th
    fm &
    ;;
  theme ) # changes the color scheme
    theme=$(ls $HOME/.themes|sed '/Ad/d ; /~$/d ; /xres/d'|\
      $DMENU -p 'themes:')
    theme $theme&
    ;;
  mv|cp|"ln -s" ) # helpers for selecting files
    sour=$(find -L . |$DMENU -p "$cmd SOURCE:")
    [[ -z $sour ]] && exit
    dest=$(find -L $HOME -type d|$DMENU -p "$cmd $sour DEST:")
    [[ -z $dest ]] && exit
    $cmd "$sour" "$dest"&
    ;; # launch interactive programs in a terminal.
  man )
    entry=$(
    (IFS=':'
    find $(manpath))|
    sed -n 's/.*\/\(.*\).1.gz/\1/p'|sort|
    $DMENU -p "man:")
    urxvtc -e $SHELL -c "man $entry"
    ;;
  youtube-viewer*|torrtux*|vifm*|vim*|vi*|ed*|nano*|ncmpcpp|cmatr
    wicd-curses|info*|weechat-curses|htop|yaourt*|"man "* )
    urxvt -e $SHELL -c "$cmd"
    ;;
  *)
    echo "$cmd"|${SHELL:-"/bin/sh"} &
esac

exit
bin/dm                                               58,6          All
"~/.vimrc
" environment
set nocompatible backspace=indent,eol,start history=50 nowrap mou
set ruler showcmd incsearch lbr expandtab tabstop=2 shiftwidth=2
set nu cursorline wildmenu wildmode=list:longest,full
let g:SuperTabDefaultCompletionType = "context"
syntax on
filetype plugin indent on
set gfn=Ubuntu\ Mono\ 10 go=

if has("gui_running") | colorscheme molokai
else | colorscheme cterm
endif
" autocmd stuff
augroup vimrcEx
  au!
  autocmd BufRead /tmp/pentadacty*.txt
        \ source /home/ninjaaron/.vim/pentadactyl.vim
  autocmd BufRead /tmp/uzbl_edit* source /home/ninjaaron/.vim/pen
  autocmd BufReadPost *
        \ if line("'\"") > 1 && line("'\"") <= line("$") |  exe "
  autocmd vimenter * if !argc() | NERDTree | endif
  autocmd bufenter * if (winnr("$") == 1 && exists("b:NERDTreeTyp
augroup END

" maps and commands and so forth
mapclear | mapclear!
command Va VoomToggle asciidoc
noremap <C-N> :NERDTreeToggle<CR>
command DiffOrig vert new | set bt=nofile | r ++edit # | 0d_ | di
      \ | wincmd p | diffthis

" syntax toggling
function! ToggleSyntax()
  if exists("g:syntax_on") | syntax off
  else | syntax enable | endif
endfunction
" hebrew mode toggle
function! ToggleHebrew()
  if &l:rightleft == 0 | set rightleft keymap=hebrew
  else | set norightleft keymap= | endif
endfunction
" create bindings for insert and normal mode together.
function! DualBind(key,command)
  exe "noremap <Leader>".a:key a:command
  exe "inoremap <Leader>".a:key "^O".a:command
endfunction
" keymaps
let mapleader=";"
" settings
nnoremap <silent> <Leader>sy :call ToggleSyntax()<CR>
call DualBind( 'i', ':call ToggleHebrew()<cr>' )
call DualBind( 'r', 'mvgq}`v' )
let settings = [['fi', 'foldmethod=indent'], ['tw', 'tw=72'], ['W
for [ key, setting ] in settings
  exe "call DualBind('".key."', ':set" setting."<cr>')" | endfor
" saving and quitting
for key in ['w', 'q', 'x']
  exe "call DualBind('".key."', ':".key."<CR>')" | endfor
" filetypes
for [ key, ft ] in [['a', 'asciidoc'], ['p', 'python'], ['s', 'sh
  exe "noremap <Leader>f".key " :set filetype=".ft "<CR>"|endfor
" Window management
for key in [ "h", "j", "k", "l", "H", "J", "K", "L" ]
  exe "call DualBind( '".key."', '^W".key."' )" | endfor
" get ya some output from shell commands
nnoremap <Leader>sh :.!sh<CR>
vnoremap <Leader>sh dmvo<Esc>p:.!sh<CR>0d$`vPjdd`v
nnoremap <Leader>py :.!python<CR>
vnoremap <Leader>py :!python<CR>
" pasting
vnoremap <Leader>c "+y
inoremap <Leader>v ^O"+P
call DualBind( 'P', '"*p' )
inoremap <Leader>p ^Op
delfunction DualBind









All src/dotfiles/vimrc                               1,1
```

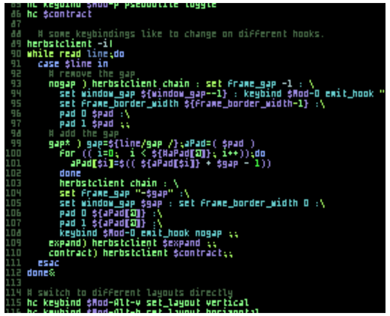
```

discharge() { acpi|grep Discharging ;}
level() { acpi|sed 's/.*ing, \(.*\)%.*/\1/' ;}
psg() { ps aux|grep "$@"|grep -v grep; }

sleepless="$HOME/sleepless"

toggle() {
  [ -a $sleepless ]  && rm $sleepless || touch $sleepless
}

idle() {
  [ -n "$(discharge)" ] && [ -z "$(psg mplayer)" ] && [ !
    sudo pm-suspend-hybrid
}

low() {
  while true ; do
    while [ $(level) -gt 5 ] || [ -z "$(discharge)" ]; do
      sleep 60
    done
    zenity --warning --text "Battery very low."
    sleep 10m
  done
}

$@


bin/autosus                                          3,0-1         Bot
#!/bin/bash

# file browsing with dmenu

source $HOME/.dmenurc
DMENU="${DMENU-dmenu} -l 60"

# this is actually the last function called. Customize be
# extntion here.
open() {
  case "$sel" in
    *.mkv|*.mov|*.avi*|*.flv*|*.ogv|*.mp4*|*.m4v|*.mp3|*.
      mplayer "$sel" &
      ;;
    *.pdf|*.ps|*.djvu )
      zathura "$sel" &
      ;;
    *.jpg|*.gif|*.png )
      gqview "$sel" &
      ;;
    *.xfc )
      gimp "$sel" &
      ;;
    *.odt|*.odp|*.odg|*.ods|*.doc )
      libreoffice "$sel" &
      ;;
    *.html|*.htm )
      firefox "$sel" &
      ;;
    *) # with undefined extentions, open vim. If the dire
      # somewhere in $HOME, ask for sudo.
      if  [[ ${PWD/*$USER*/$USER} != $USER ]] &&
      [[ $(echo -e "y\nn"|$DMENU -p "sudo?") = y ]] ; the
        urxvtc -e sh -c "sudo vim -u $HOME/.vimrc '$PWD/$
        exit
      else
        urxvtc -e sh -c "vim '$PWD/$sel'"
      fi
  esac
}

# use `find` to get the entire contense of $HOME recursiv
search() {

  # really long subshell supplies the file list and hides
  # files. In $HIME, hide hidden files by defautl unless
  # selects 'unhide.'
  menu=$(IFS="
  "
  if [[ ${PWD/\/home*/\/home} = /home ]];then

    if [[ $show = hidden ]]; then
      find -L .|
      sed '/~$/d ; /Ubuntu One/{ H; d; }; $ { p; x; }'
      echo "hide"
    else
      find -L . |
      sed "/\/\./d ; /ulconf/d ; /~$/d ; /Ubuntu One/{ H;
      echo "unhide"
    fi

  else
    find .|sed '/~$/d'
  fi)

  sel=$(echo "$menu"|$DMENU -p "search $PWD:")
  case $sel in
    "" )
      exit ;;
    hide )
      show=normal
      search ;;
    unhide )
      show=hidden
      search ;;
    *)
      browse "$@"
  esac
}

# browse files one directory at a time. the only way out
# into browse(). However, when $sel is a file, browse() h
# off to open(). This is a result of how the program orig
# I've thought about changing to go directly to open() fr
# but I spent my time writing this sentence instead.
browse() {
  while [ -d "$sel" ]; do
    cd "$sel"
    sel=$(ls -a|$DMENU -p "$PWD:")
    [[ $sel = . ]] && break
  done

  case $sel in
    "." )
      search ;;
    "" )
      exit 0 ;;
    *)
      open&
  esac
}

# the search() is the default behavior, but I also have t
# aliased to `brws`, so when invoced that way, it runs br
if [[ $0 = *brws ]];
then sel="$PWD" && browse
else search
fi

exit









All bin/fm                                           1,1           All
```

```
axelrc
bash.bash_logou
bash.bashrc
bash_completion
bdfeditrc
binfmt.d/
ca-certificates
ca-certificates
colord.conf
conf.d/
conky/
cron.d/
cron.daily/
cron.deny
cron.hourly/
cron.monthly/
cron.weekly/
crypttab
cups/
dbus-1/
default/
depmod.d/
dhcpcd.conf
drirc
environment
fb.modes
/etc> cd /usr/sh
aclocal/
aclocal-1.13/
alsa/
application-reg
applications/
apr-1/
autoconf/
automake-1.13/
avahi/
awk/
bash-completion
bdfedit/
bibletime/
bison/
ca-certificates
ca-certificates
calc/
cmake/
cmake-2.8/
color/
colord/
common-lisp/
cracklib/
cups/
dblatex/
dbus-1/
dconf-editor/
desktop-director
devhelp/
dict/
diagnose/
djvu/
doc/
dstat/
dtds/
emacs/
emesene/
enchant/
et/
FBReader/
ffmpeg/
figlet/
file/
fonts/
gcalctool/
gcc-4.7.2/
GConf/
gdb/
gdm/
gedit/
geoclue-provider
getopt/
gettext/
ghostscript/
gimp/
gir-1.0/
git/
/usr/share> cd 
alarm*     backu
aur*       barac
autosus*   belt*
~/bin> cd
bin  doc/  mnt
~> cat src/dotf
default
partial alphanu
xkb_symbols "an

  include "us(b
    name[Group1
  key.type[grou

  // wadova
  key  <LSGT> {
  key  <TLDE> {
  key  <AE01> {
  key  <AE02> {
  key  <AE03> {
  key  <AE04> {
  key  <AE05> {
  key  <AE06> {
  key  <AE07> {
  key  <AE08> {
  key  <AE09> {
  key  <AE10> {
  key  <AE11> {
  key  <AE12> {

  key  <AD01> {
  key  <AD02> {
sigma>
  key  <AD03> {
  key  <AD04> {
  key  <AD05> {
  key  <AD06> {
  key  <AD07> {
  key  <AD08> {
  key  <AD09> {
  key  <AD10> {
>
  key  <AD11> {
  key  <AD12> {
>
  key  <BKSL> {

  key  <AC01> {
  key  <AC02> {
  key  <AC03> {
  key  <AC04> {
  key  <AC05> {
  key  <AC06> {
  key  <AC07> {
  key  <AC08> {
  //
  key  <AC09> {
  key  <AC10> {
  key  <AC11> {

  key  <AB01> {
  key  <AB02> {
  key  <AB03> {
  key  <AB04> {
  key  <AB05> {
  key  <AB06> {
  key  <AB07> {
  key  <AB08> {
  key  <AB09> {
  key  <AB10> {

  key  <SPCE> {

  include "leve
};
~> 


  1  [||||
  2  [||
  3  [||
  4  [||
  Mem[|||||||||
  Swp[|

Playlist (8 ite

Time    Artist

4:36    Christi
==========>
Playing: Christ









~> 
```

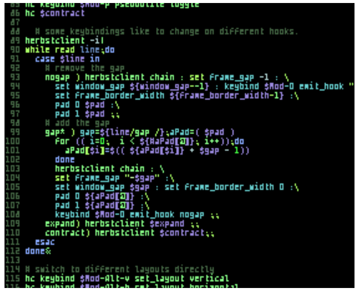


State diagram PC watchdog process

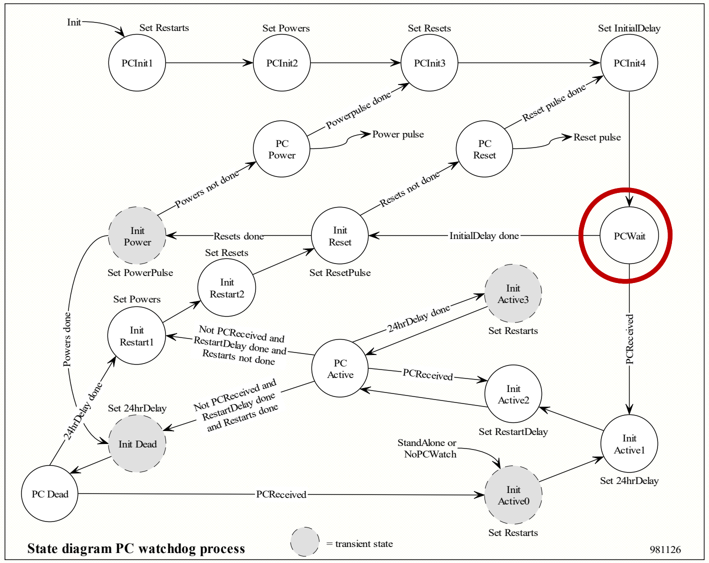




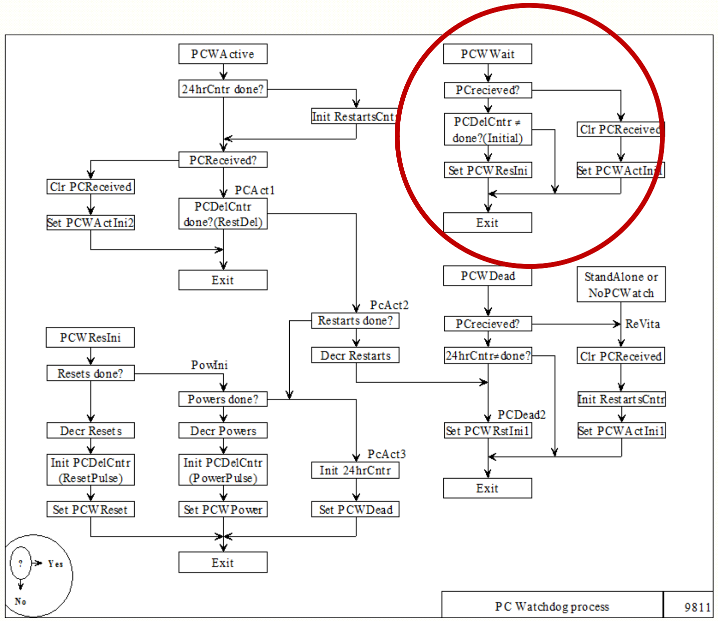
```
hc keybind $Mod-p pseudotile toggle
hc $contract

  # some keybindings like to change on different hooks.
herbstclient -il
while read line;do
  case $line in
    # remove the gap
    nogap ) herbstclient chain : set frame_gap -1 : \
      set window_gap ${window_gap--1} : keybind $Mod-0 emit_hook "
      set frame_border_width ${frame_border_width-1} :\
      pad 0 $pad :\
      pad 1 $pad ;;
    # add the gap
    gap* ) gap=${line/gap /};aPad=( $pad )
      for (( i=0;  i < ${#aPad[@]}; i++));do
        aPad[$i]=$(( ${aPad[$i]} + $gap - 1))
      done
      herbstclient chain : \
      set frame_gap "-$gap" :\
      set window_gap $gap : set frame_border_width 0 :\
      pad 0 ${aPad[0]} :\
      pad 1 ${aPad[0]} :\
      keybind $Mod-0 emit_hook nogap ;;
    expand) herbstclient $expand ;;
    contract) herbstclient $contract;;
  esac
done&

# switch to different layouts directly
hc keybind $Mod-Alt-v set_layout vertical
hc keybind $Mod-Alt-h set_layout horizontal
```

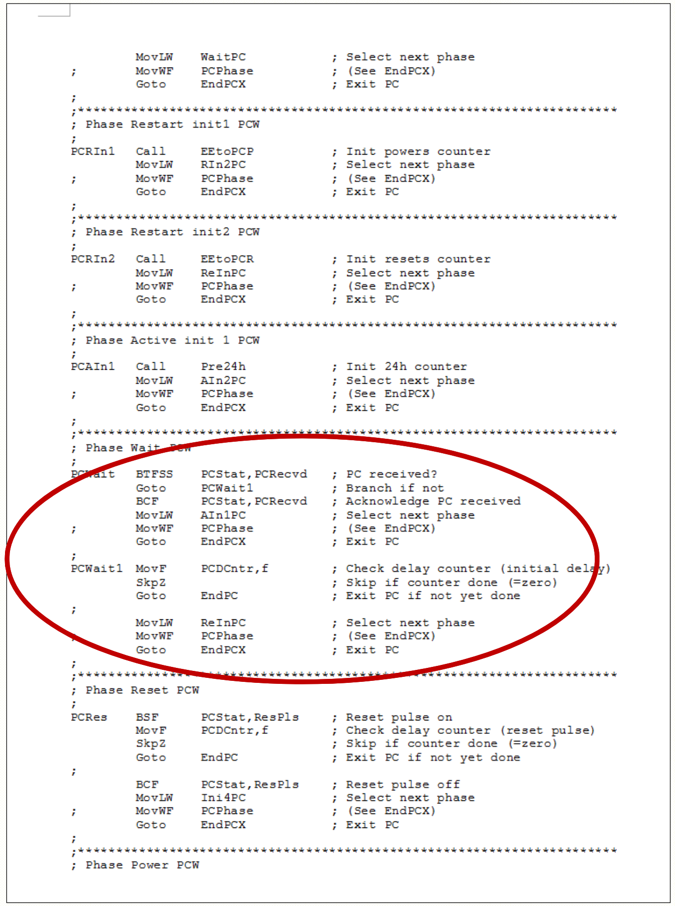


```
hc keybind $Mod-p pseudotile toggle
hc $contract

  # some keybindings like to change on different hooks.
herbstclient -il
while read line;do
  case $line in
    # remove the gap
    nogap ) herbstclient chain : set frame_gap -1 : \
      set window_gap ${window_gap--1} : keybind $Mod-0 emit_hook "
      set frame_border_width ${frame_border_width-1} :\
      pad 0 $pad :\
      pad 1 $pad ;;
    # add the gap
    gap* ) gap=${line/gap /};aPad=( $pad )
      for (( i=0;  i < ${#aPad[@]}; i++));do
        aPad[$i]=$(( ${aPad[$i]} + $gap - 1))
      done
      herbstclient chain : \
      set frame_gap "-$gap" :\
      set window_gap $gap : set frame_border_width 0 :\
      pad 0 ${aPad[0]} :\
      pad 1 ${aPad[0]} :\
      keybind $Mod-0 emit_hook nogap ;;
    expand) herbstclient $expand ;;
    contract) herbstclient $contract;;
  esac
done&

# switch to different layouts directly
hc keybind $Mod-Alt-v set_layout vertical
hc keybind $Mod-Alt-h set_layout horizontal
```



State diagram PC watchdog process





```
          MovLW    WaitPC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Restart init1 PCW
;
PCRIn1    Call     EEtoPCP             ; Init powers counter
          MovLW    RIn2PC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Restart init2 PCW
;
PCRIn2    Call     EEtoPCR             ; Init resets counter
          MovLW    ReInPC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Active init 1 PCW
;
PCAIn1    Call     Pre24h              ; Init 24h counter
          MovLW    AIn2PC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Wait PCW
;
PCWait    BTFSS    PCStat,PCRecvd      ; PC received?
          Goto     PCWait1             ; Branch if not
          BCF      PCStat,PCRecvd      ; Acknowledge PC received
          MovLW    AIn1PC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
PCWait1   MovF     PCDCntr,f           ; Check delay counter (initial delay)
          SkpZ                         ; Skip if counter done (=zero)
          Goto     EndPC               ; Exit PC if not yet done
;
          MovLW    ReInPC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Reset PCW
;
PCRes     BSF      PCStat,ResPls       ; Reset pulse on
          MovF     PCDCntr,f           ; Check delay counter (reset pulse)
          SkpZ                         ; Skip if counter done (=zero)
          Goto     EndPC               ; Exit PC if not yet done
;
          BCF      PCStat,ResPls       ; Reset pulse off
          MovLW    Ini4PC              ; Select next phase
;         MovWF    PCPhase             ; (See EndPCX)
          Goto     EndPCX              ; Exit PC
;
;******************************************************************
; Phase Power PCW
```

Just reviewing code doesn’t solve consistency issues

Case: Scrum Sprint Planning

- What is the measure of success for the coming sprint ?
- “What a strange question !
  We're Agile, so we deliver working software. Don't you know ?”

- Note: Users are not waiting for *software*:
  they’re waiting for *improved performance* of what they’re doing

- How about a requirement for 'Demo': No Questions – No Issues
- That is impossible !!

- They actually succeeded !

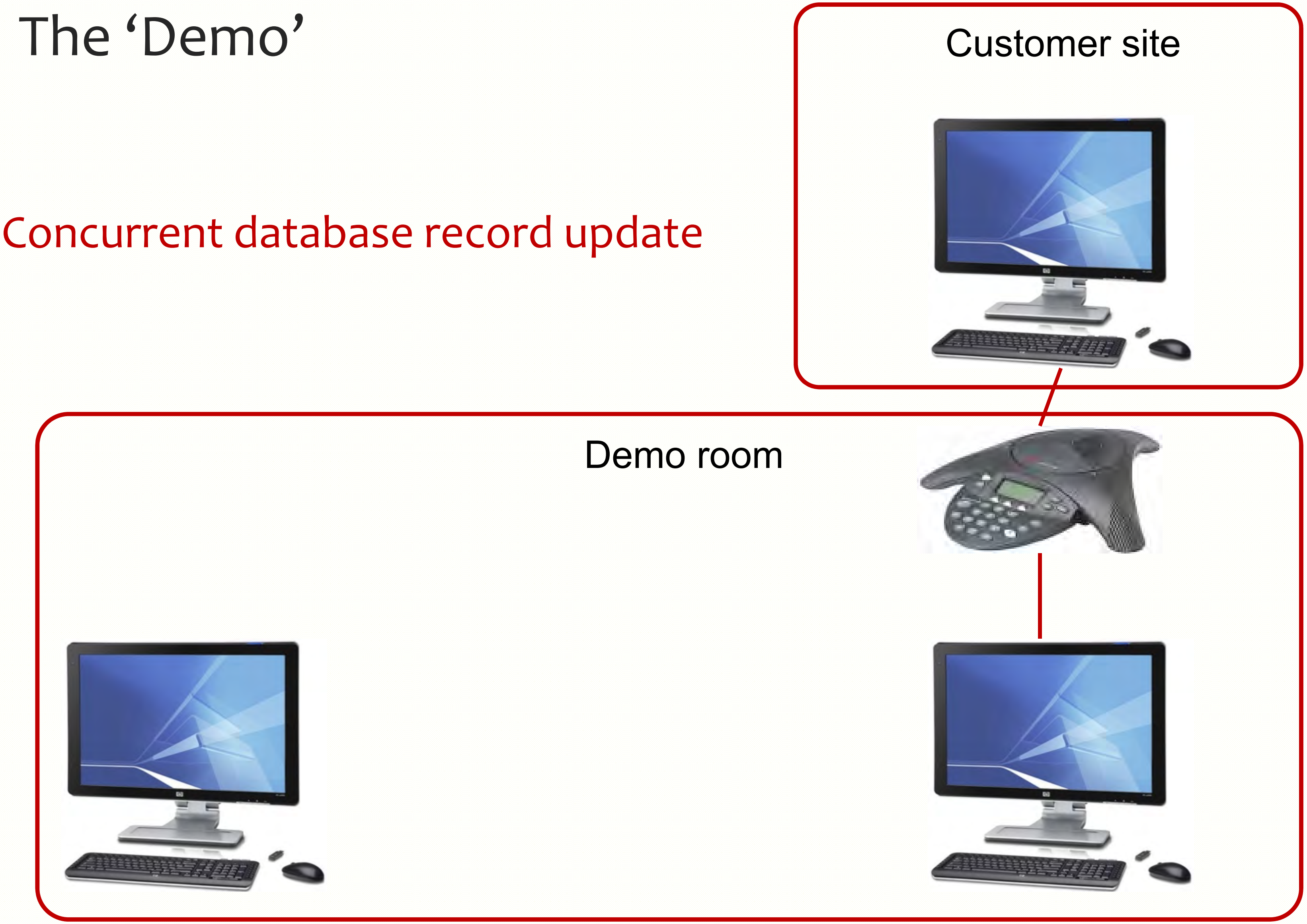
Demo ??

- Give the delivery to the stakeholders
- Keep your hands handcuffed on your back
- Keep your mouth shut
- and o-b-s-e-r-v-e what happens
- Seeing what the stakeholders actually do provides so much better feedback
- Then we can 'talk business' with the stakeholders

- Is this what you do ?

The ‘Demo’

Concurrent database record update



Customer site

Demo room

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)

How much legwork is being done in your project ?

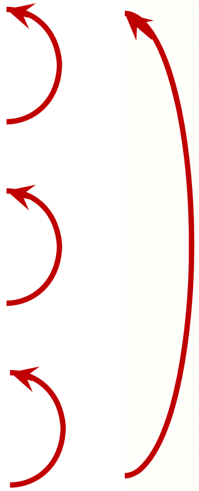
- Requirements/specifications were trashed out with product management
- Technical analysis was done and
- Detail design for the first delivery

At the first delivery:

- James: *How is the delivery? (quality versus expectation)*
- Adrian: *It's exactly as expected, which is absolutely unprecedented for a first delivery;*
  the initial legwork has really paid off

Basic approach

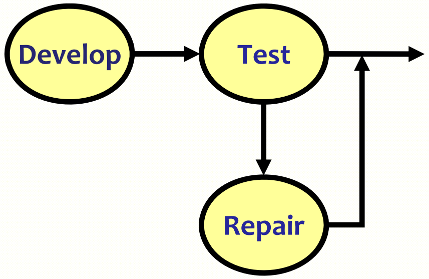
- Design the requirement
- Review
- Design implementation
- Review
- Implement (code)
- Review
- Test doesn't find issues (because there are no issues)



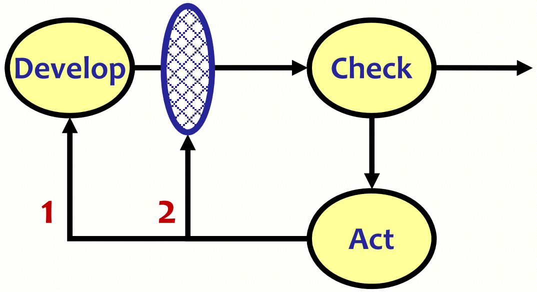
Iterate fast, as needed

What’s in it for testers ?

- Did we see much testing in the previous ?
- Testing shouldn’t find anything (because there should be no issues)
- Did you ever find similar issues as you found before?
  - First time: Developers ‘fault’
  - Second time: Testers ‘fault’



What we often see



What we should expect

- QA to help developers to produce less and less defects

Dijkstra (1972)

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

However:

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

but it is hopelessly inadequate for showing their absence

Conventional testing:

- Pursuing the very effective way to show the presence of defects

The challenge is, however:

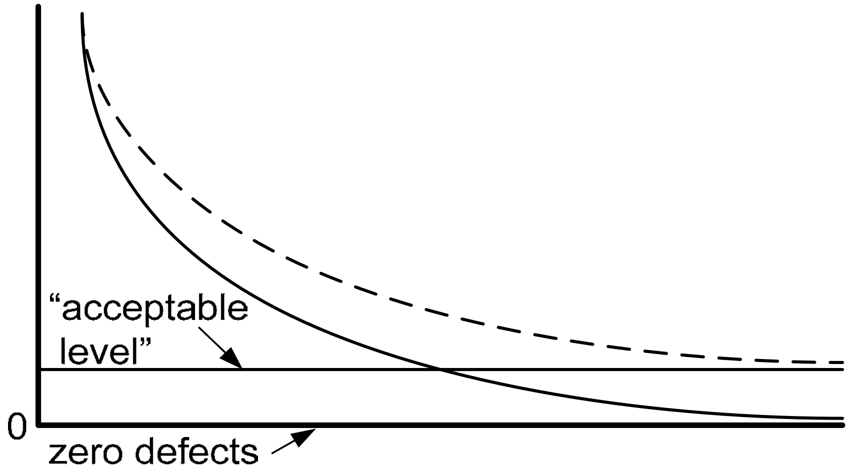
- Making sure that there are no defects (development)
- How to show their absence if they're not there (testing ?)

Do we deliver Zero Defect products ?

- How many defects do you think are acceptable ?
- Do the requirements specify a certain number of defects ?
- Do you check that the required number has been produced ?

In your projects

- How much time is spent putting defects in ?
- How much time is spent trying to find and fix them ?
- Do you sometimes get repeated issues ?
- How much time is spent on defect prevention ?
- Could you use “No Questions – No Issues” ?



Approaching Zero Defects is Absolutely Possible

If in doubt, let's talk about it

Niels Malotaux

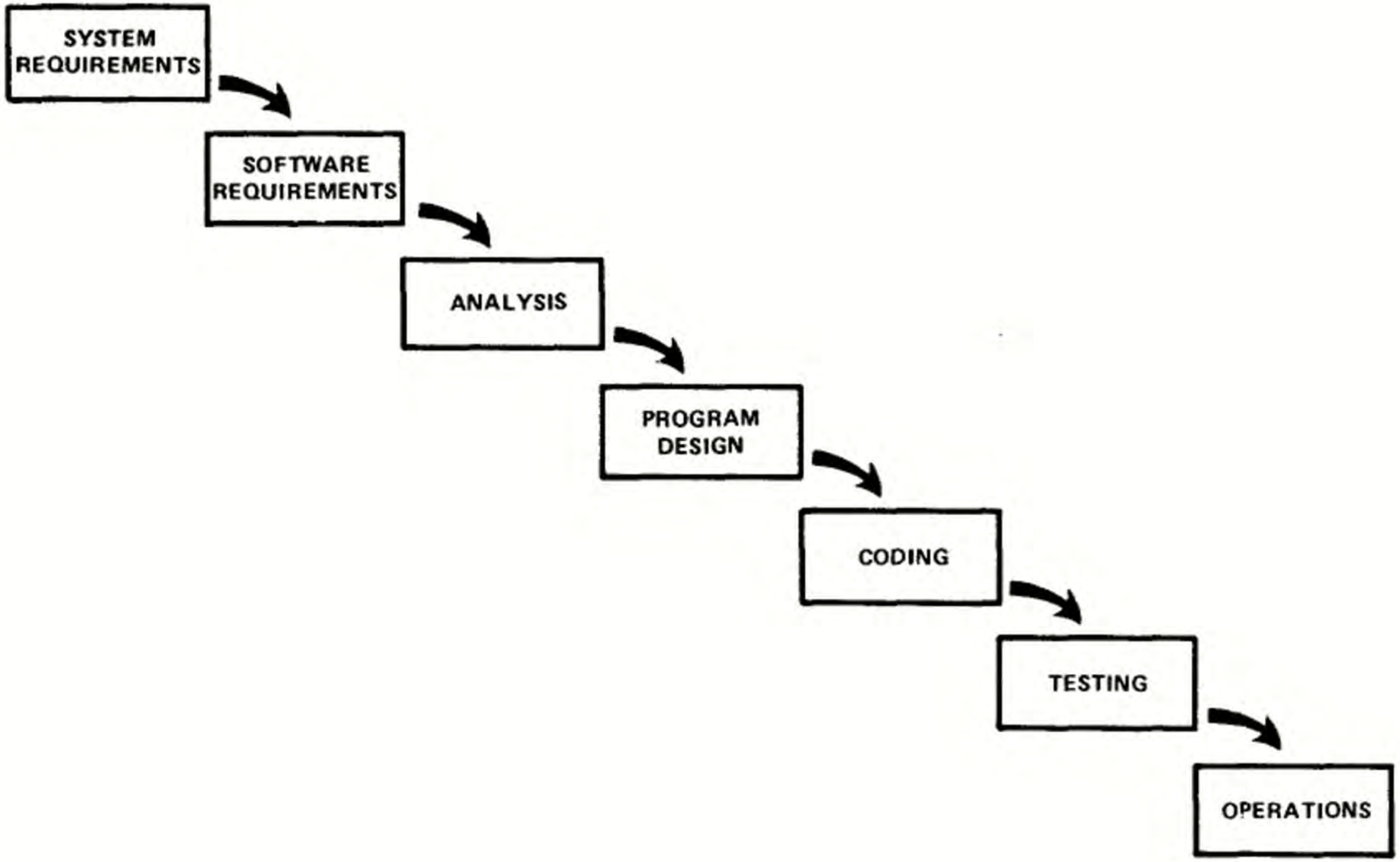
niels@malotaux.eu

www.malotaux.eu/conferences

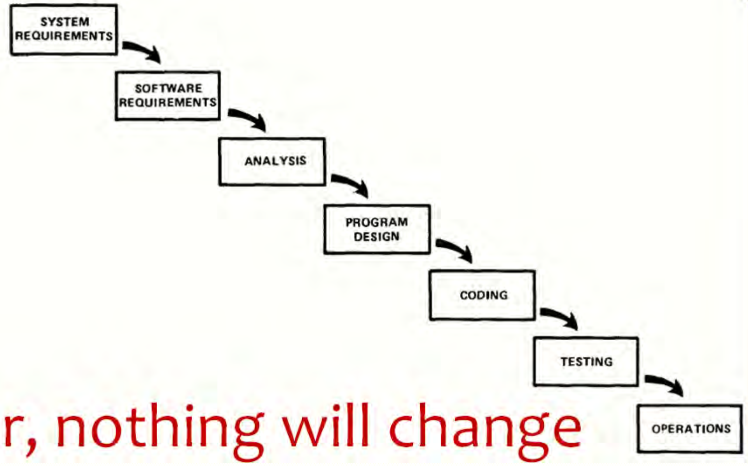
Project Life Cycles

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*

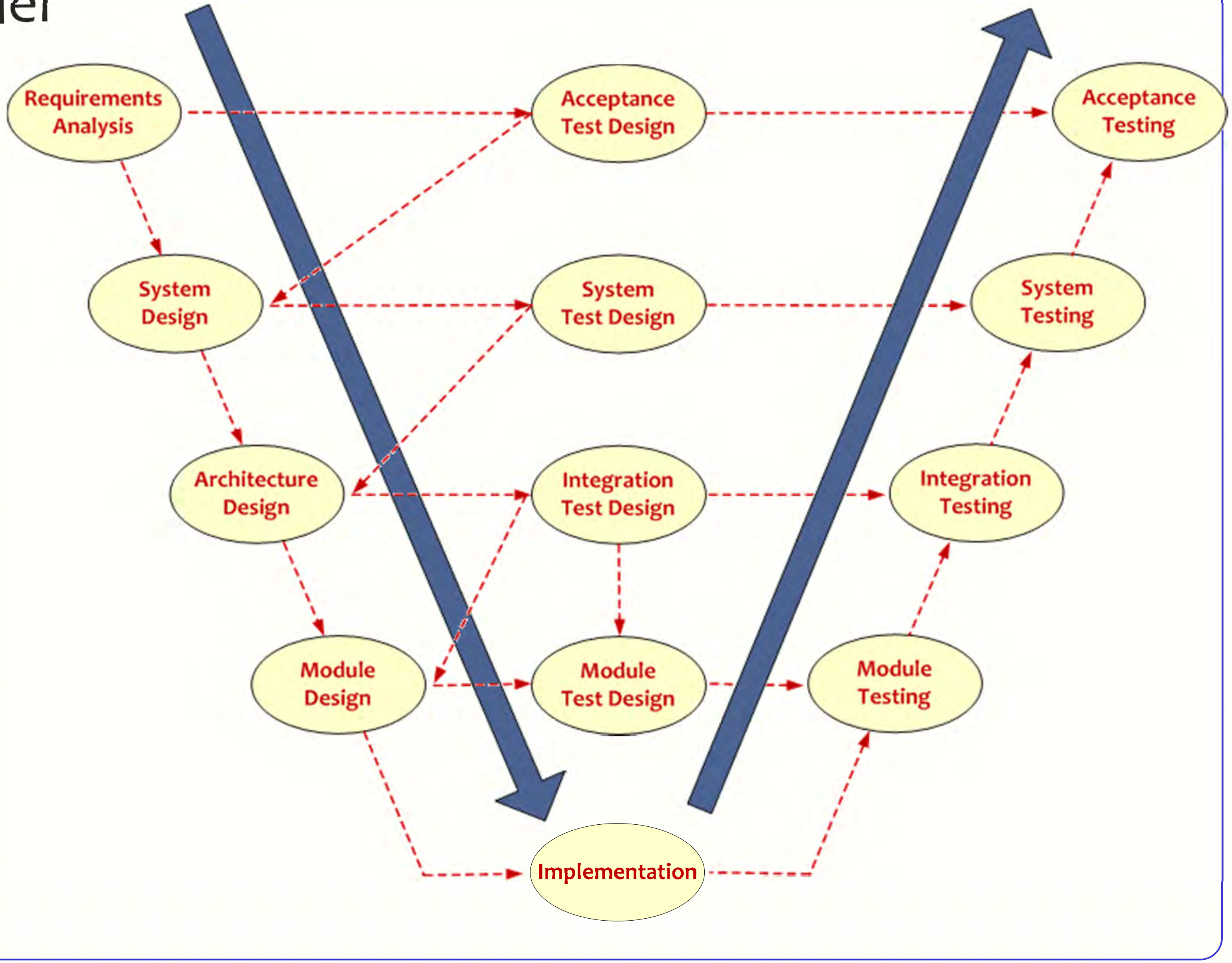
  Even most production doesn’t run smoothly the first time, it has to be tuned

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

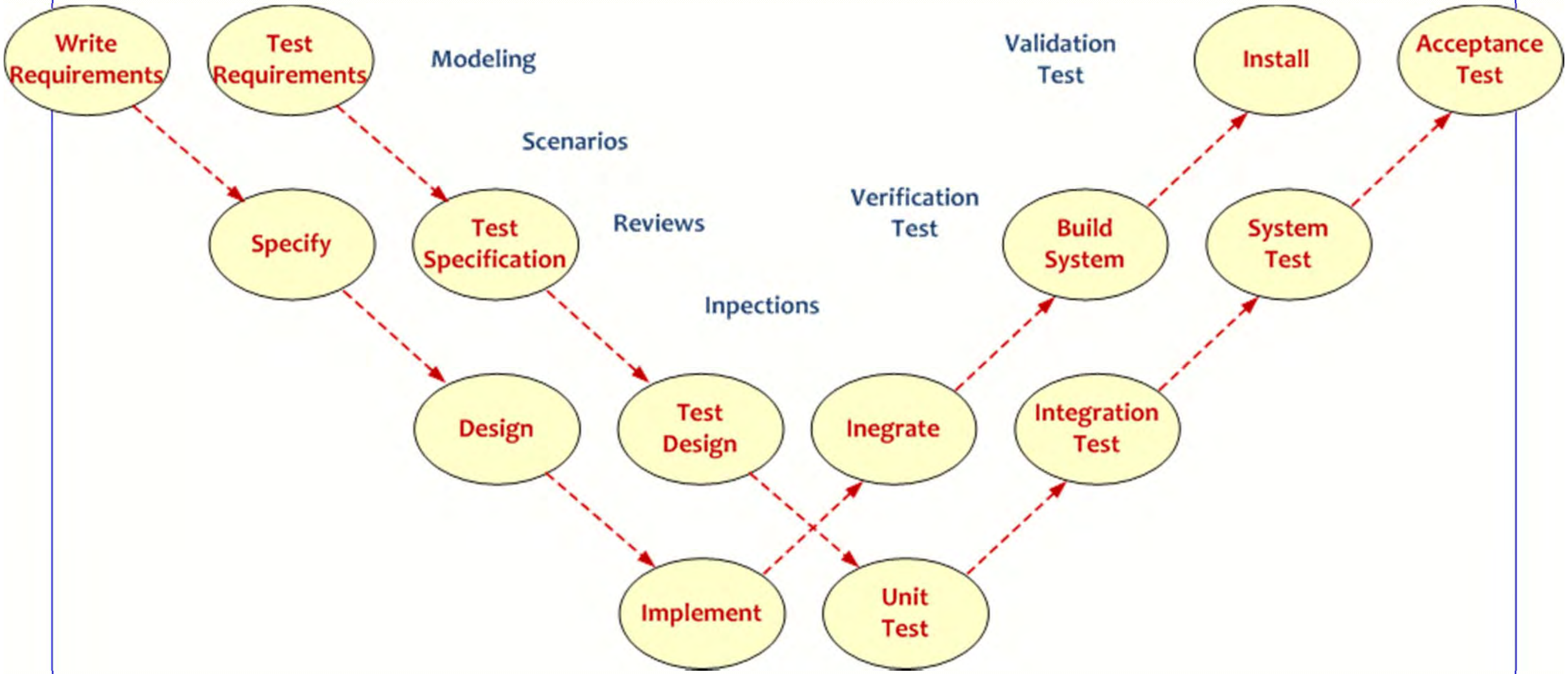
Problem - Solution

Problem known	-	Solution known	=	production
Problem known	-	Solution unknown	=	development
Problem unknown	-	Solution known	=	many IT projects
Problem unknown	-	Solution unknown	=	no problem ?

V-Model


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

W-model can be used for every Sprint

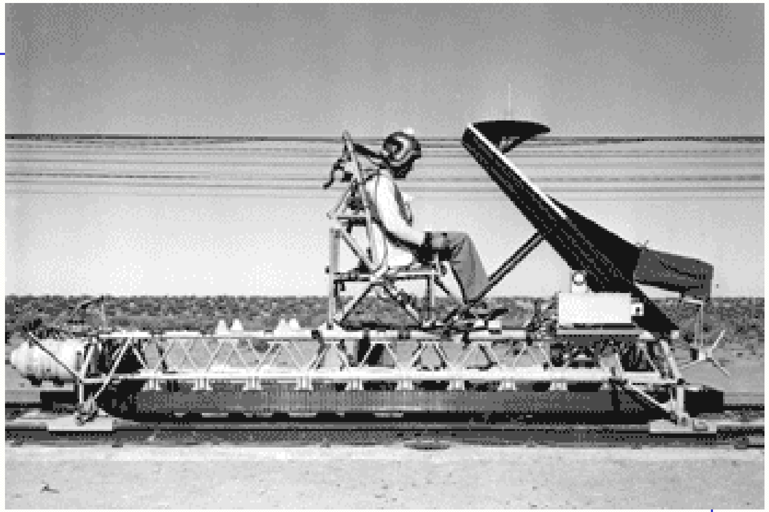


All Models are wrong

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*

Do you use Retrospectives ?

Do we really learn from what happened ?

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

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

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

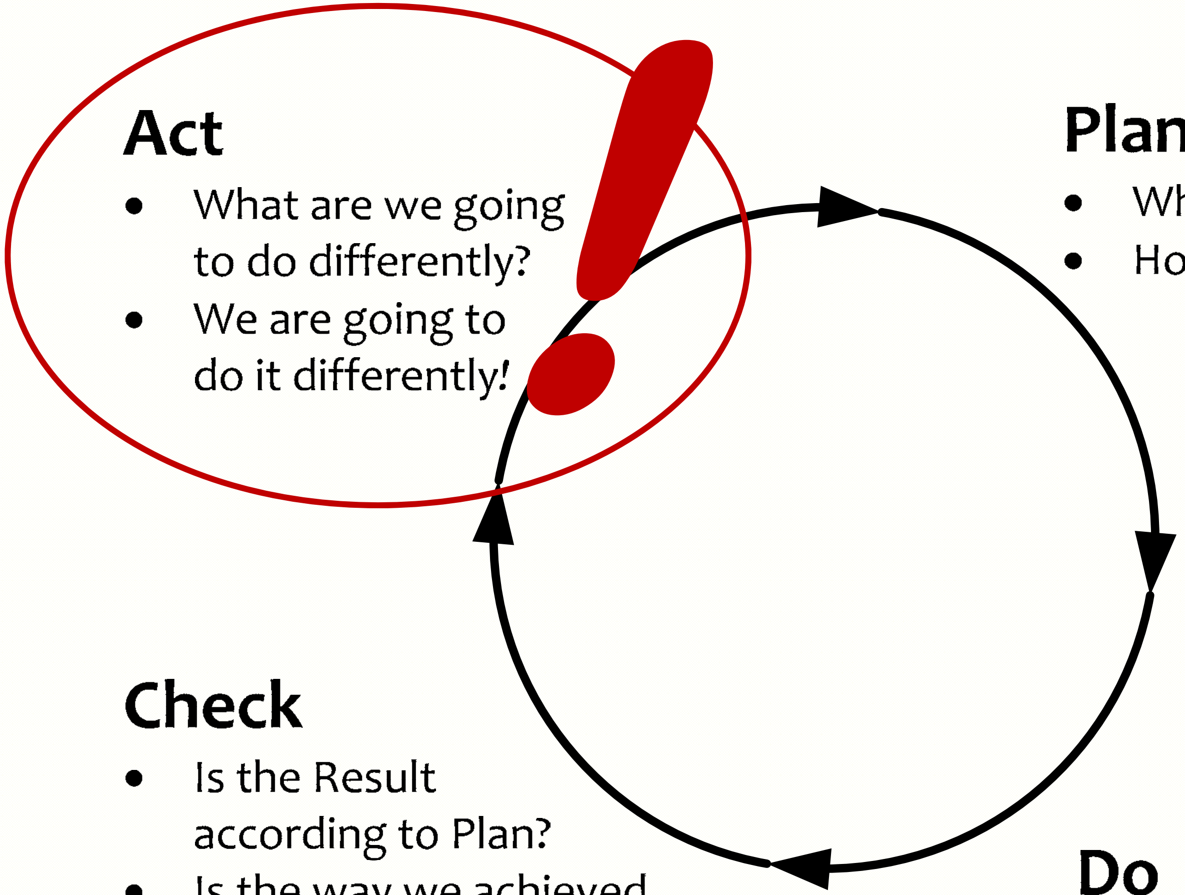
- Hindsight is easy, but reactive
- Foresight is less easy, but proactive
- Reflection is for hindsight and learning
- Preflection is for foresight and prevention

Only with *prevention* we can save precious time

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

The essential ingredient: the PDCA Cycle

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



Act

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

Plan

- What to achieve
- How to achieve it

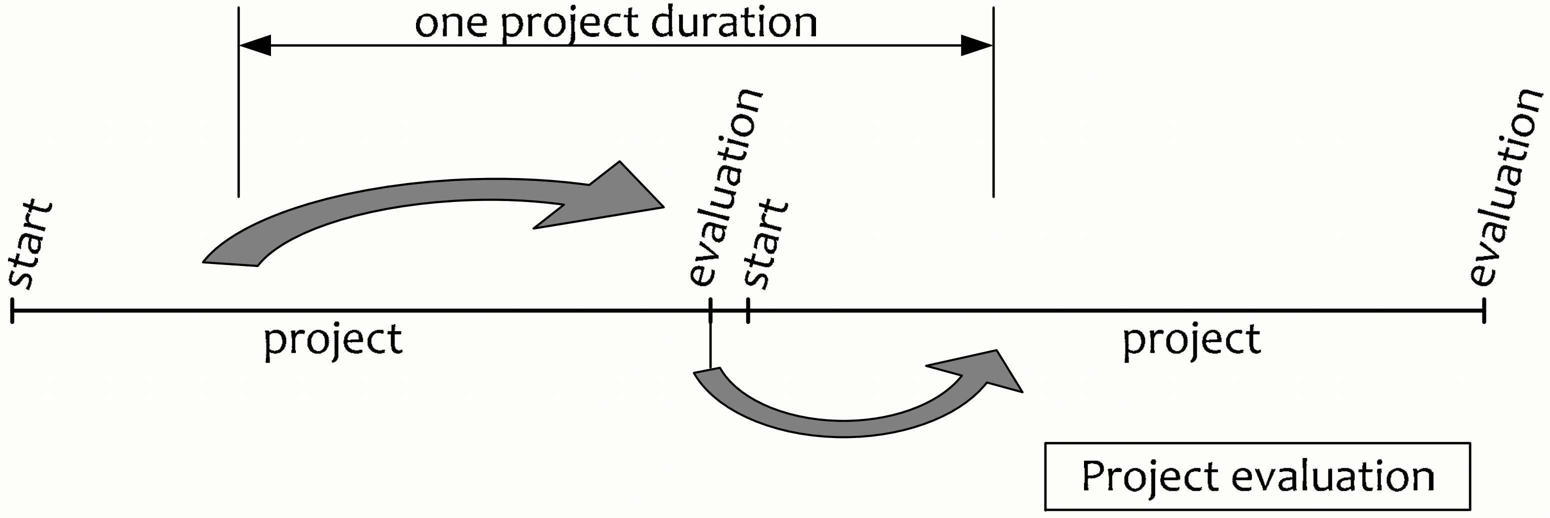
Do

Carry out the Plan

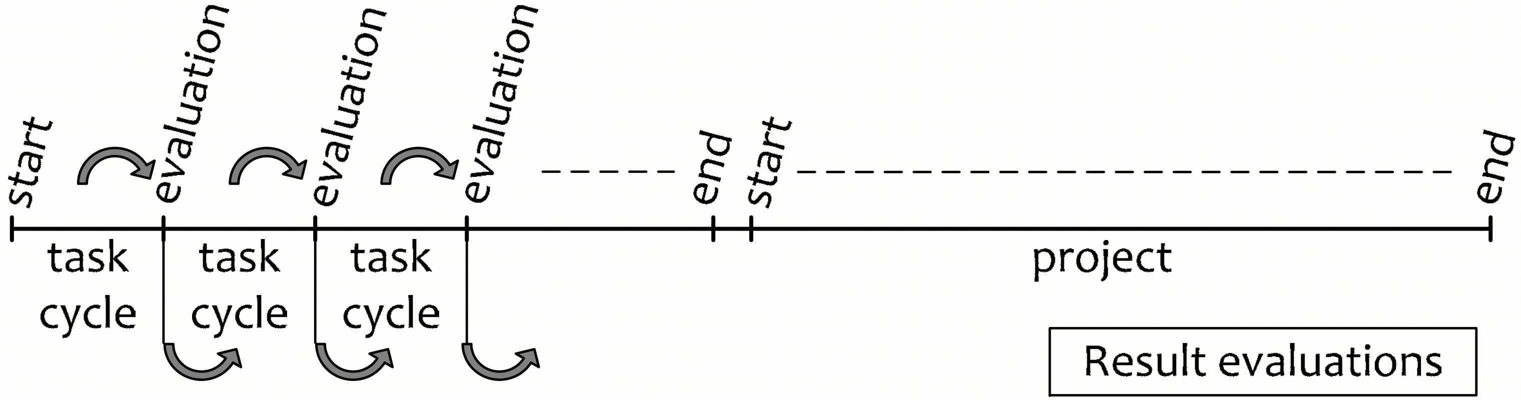
Check

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

Project evaluations

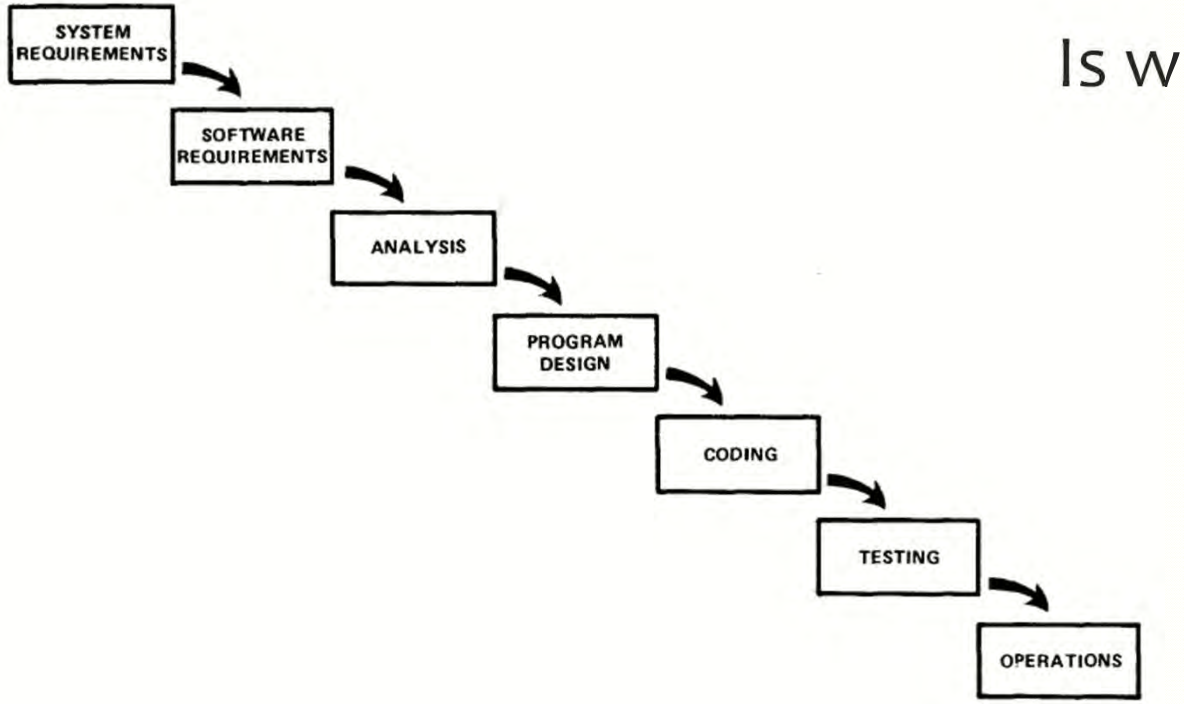


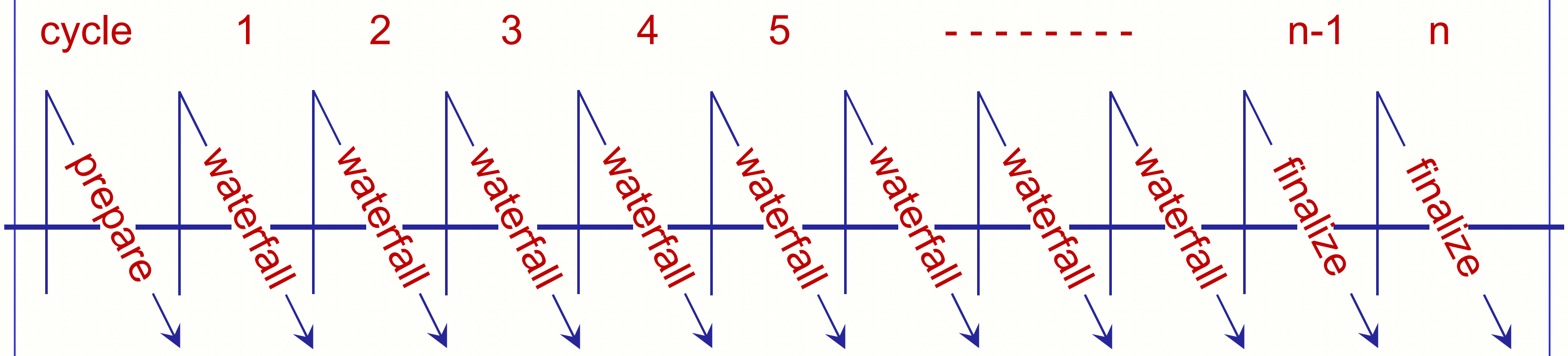
Project evaluation



Result evaluations

Is waterfall wrong ?

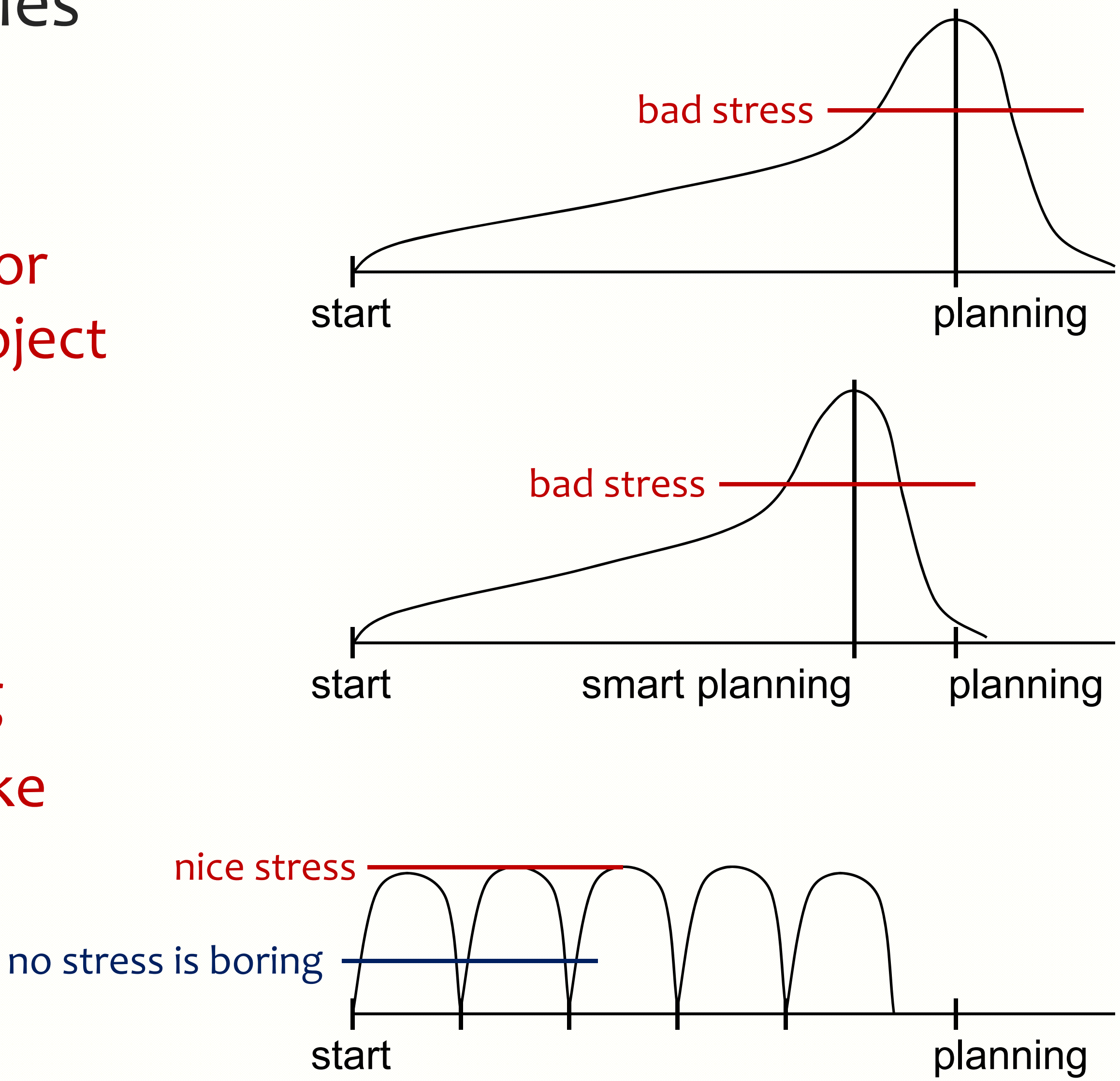

SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS


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

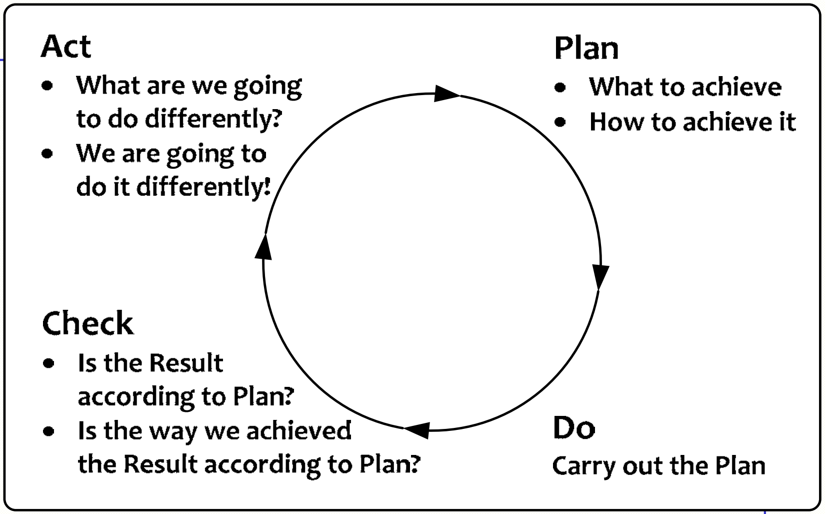
prepare
waterfall
waterfall
waterfall
waterfall
waterfall
waterfall
waterfall
waterfall
finalize
finalize

Development cycles

- Bad stress is bad for person and for project
- No stress is boring
- Nice stress feels like accomplishment, is sustainable



Knowledge
how to achieve the goal



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

Plan
- What to achieve
- How to achieve it

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

Do
Carry out the Plan

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

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

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

doing the right things

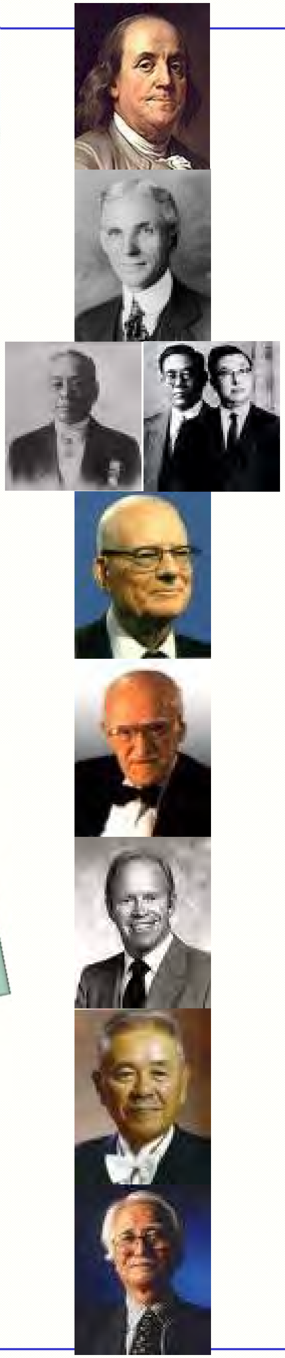
doing the right things right

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)

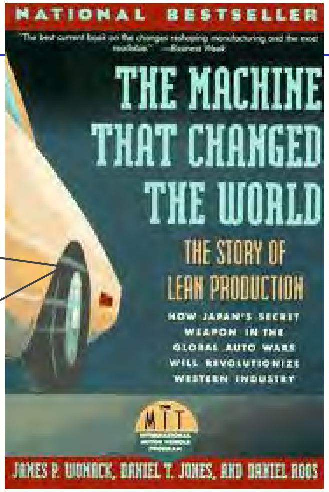
Eliminating Waste
Not doing what doesn't yield value

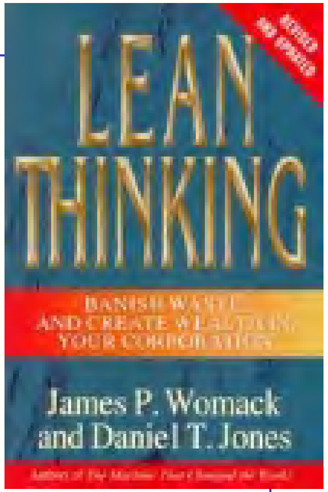


Lean

A lot of the cost of vehicles is based on:

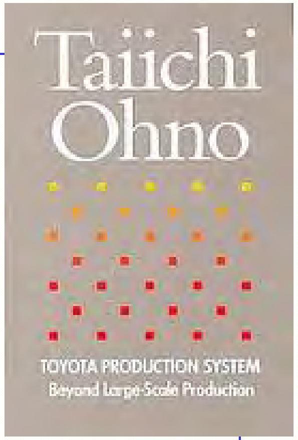
- **bad design**
- **poor management**
- **an attitude that problems, no matter how small, can be overlooked**


NATIONAL BESTSELLER
THE MACHINE THAT CHANGED THE WORLD
THE STORY OF LEAN PRODUCTION
HOW JAPAN'S SECRET WEAPON IN THE GLOBAL AUTO WARS WILL REVOLUTIONIZE WESTERN INDUSTRY
MIT
JAMES P. WOMACK, DANIEL T. JONES, AND DANIEL ROOS


LEAN THINKING
BANISH WASTE AND CREATE WEALTH IN YOUR CORPORATION
James P. Womack and Daniel T. Jones

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

Toyota Production System (TPS)



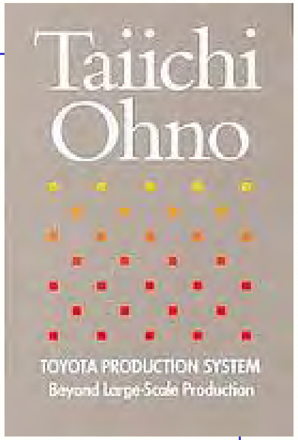
1950

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

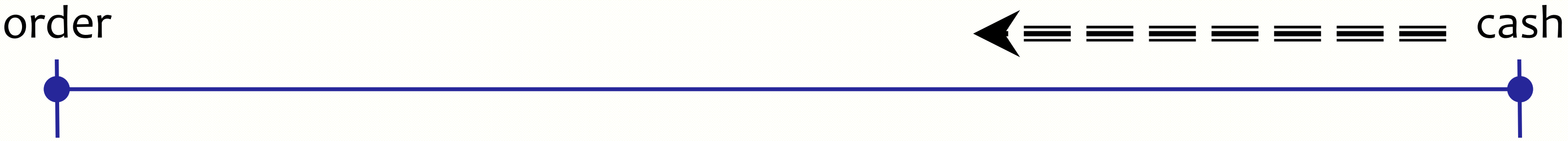
Four specific aims:

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

Taiichi Ohno - The Toyota Production System



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



Reducing the time by removing non-value-added wastes

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

Pillars of the TPS

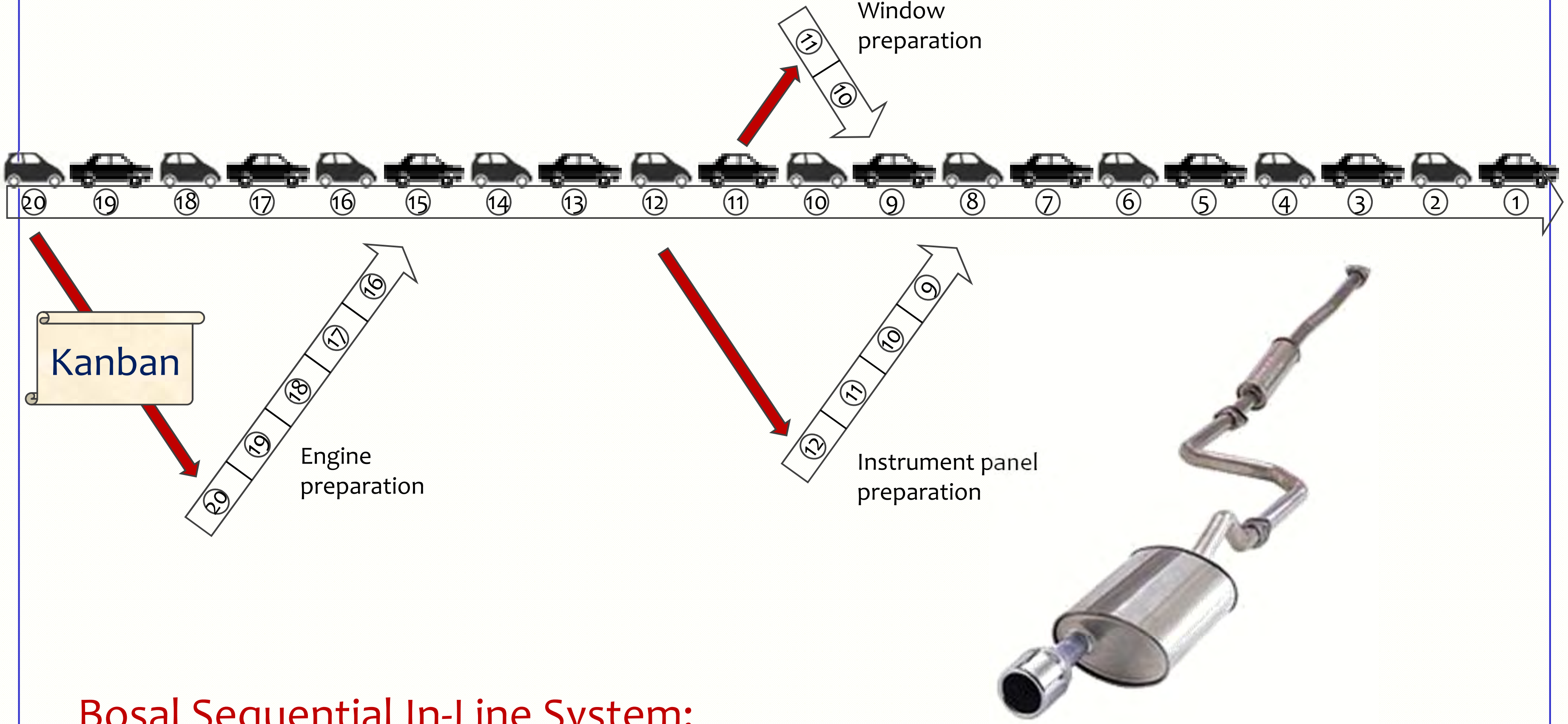
- Just in Time
  - No inventory
  - Doing the right things at the right time
- Perfection
  - Perfection is a condition for JIT to work
  - If a defect is found, stop the line, find cause, fix immediately
  - Continuous improvement of product, project and process
- Autonomation
  - The loom runs unattended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

  For development:

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

Just In Time delivery – *no inventory*

(after Ohno)



Window preparation

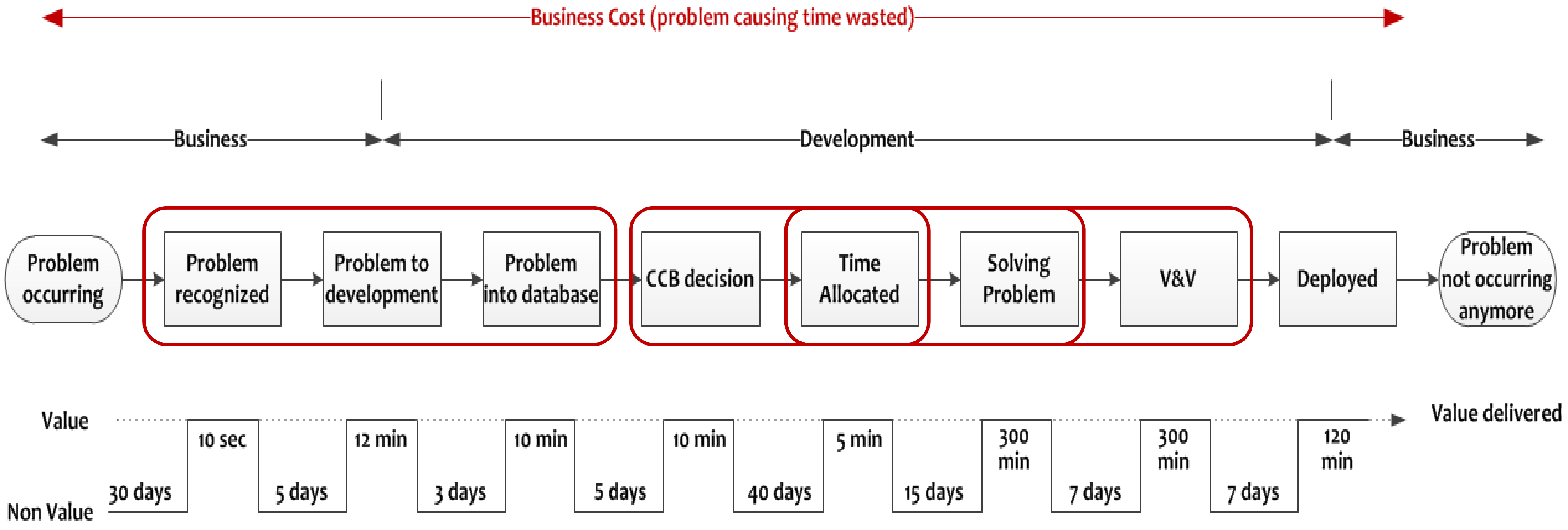
Kanban

Engine preparation

Instrument panel preparation

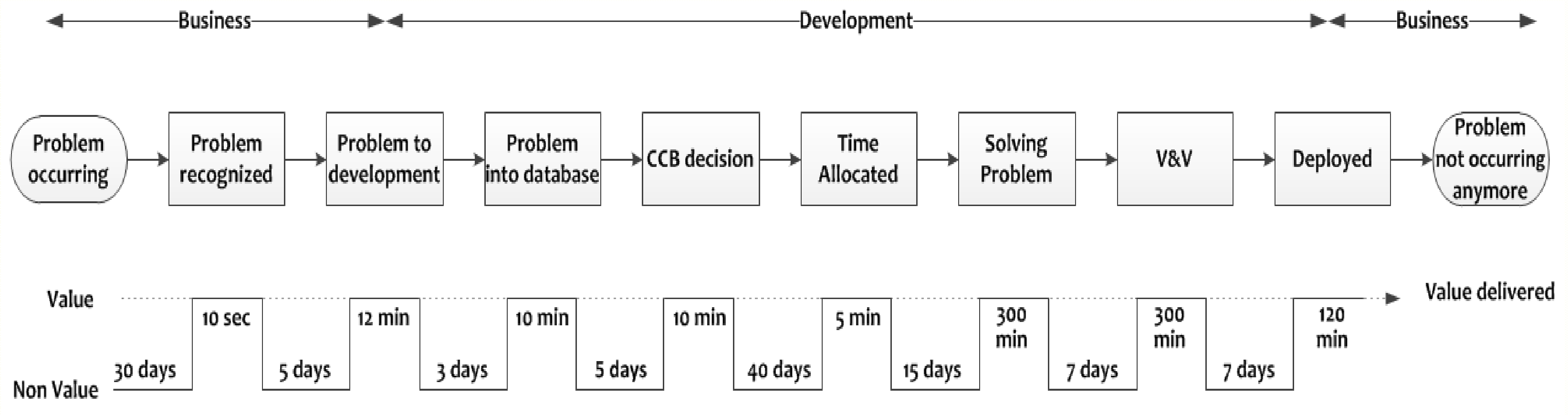
Bosal Sequential In-Line System:
We pioneered just-in-time delivery of exhaust systems - supplying systems to the assembly line within 80 minutes of receiving the order

Value stream example



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

Capacity = Work + Waste



Work Capacity

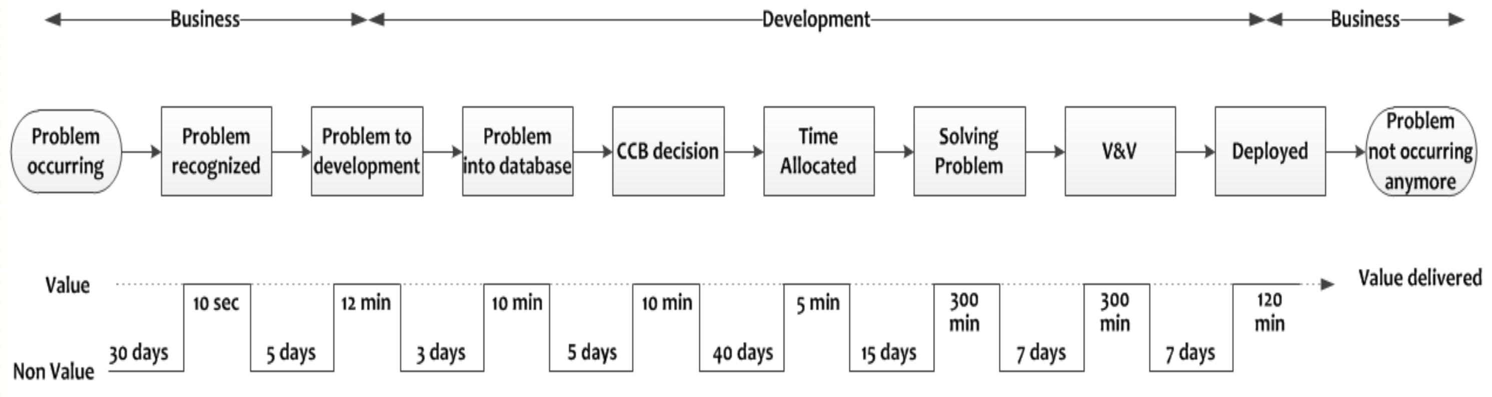
- Net Work, creating value
- Non-value adding, but necessary work
- Waste

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

Identifying waste

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

5-S

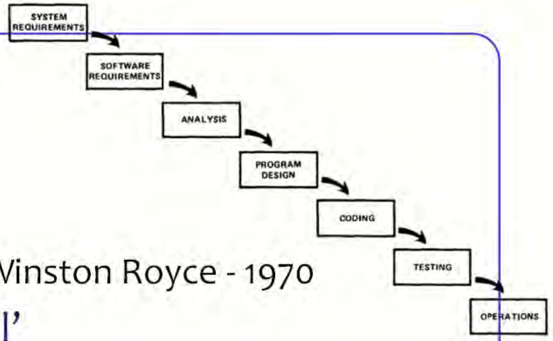


- Seiri - Remove unnecessary things → waste
- Seiton - Arrange remaining things orderly → flow
- Seiso - Keep things clean → uncovers hidden problems
- Seiketsu - Keep doing it, standardize → know what to improve
- Shitsuke - Keep training it → fighting entropy

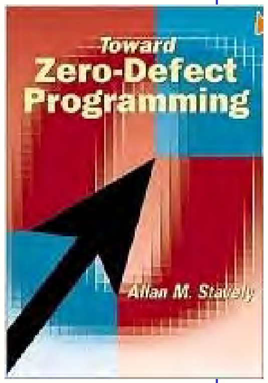
The 3 Mu’s to remove

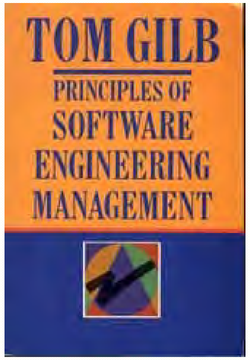
- Muda - Waste → minimize waste
- Mura - Irregularities → optimize flow
- Muri - Stress → sustainable pace

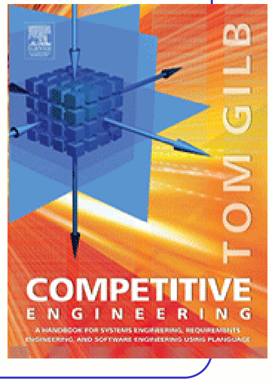
There is nothing new in software too



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



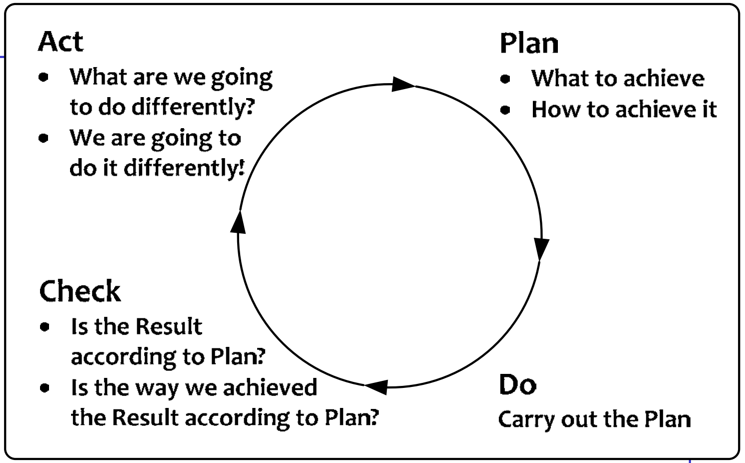



TOM GILB
COMPETITIVE
ENGINEERING
A HANDBOOK FOR SYSTEMS ENGINEERING, REQUIREMENTS ENGINEERING, AND SOFTWARE ENGINEERING USING PLANGUAGE

Lean things

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

Evo



Act

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

Plan

- What to achieve
- How to achieve it

Check

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

Do

Carry out the Plan

- Evo (short for Evolutionary…) uses PDCA consistently
- Applying the PDCA-cycle
  actively, deliberately, rapidly and frequently,
  for *Product*, *Project* and *Process*, based on ROI and highest value
- Combining Planning, Requirements- and Risk-Management into *Result Management*
- We know we are not perfect, but the customer shouldn't be affected
- Evo is about delivering Real Stuff to Real Stakeholders doing Real Things

"Nothing beats the Real Thing"

- Projects seriously applying Evo, routinely conclude successfully on time, or earlier

Evolutionary Project Management elements (Evo) – Tom Gilb

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

Why

- What
- How much
- Are we done

How

Zero Defects Attitude

Check as early as possible

Right Result

Quality On Time

Evo Project Planning

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

Efficiency of what we do

Right Time

Effectiveness of what we do

What will happen and what will we do about it ?

Evolutionary Planning

prevention is better than cure

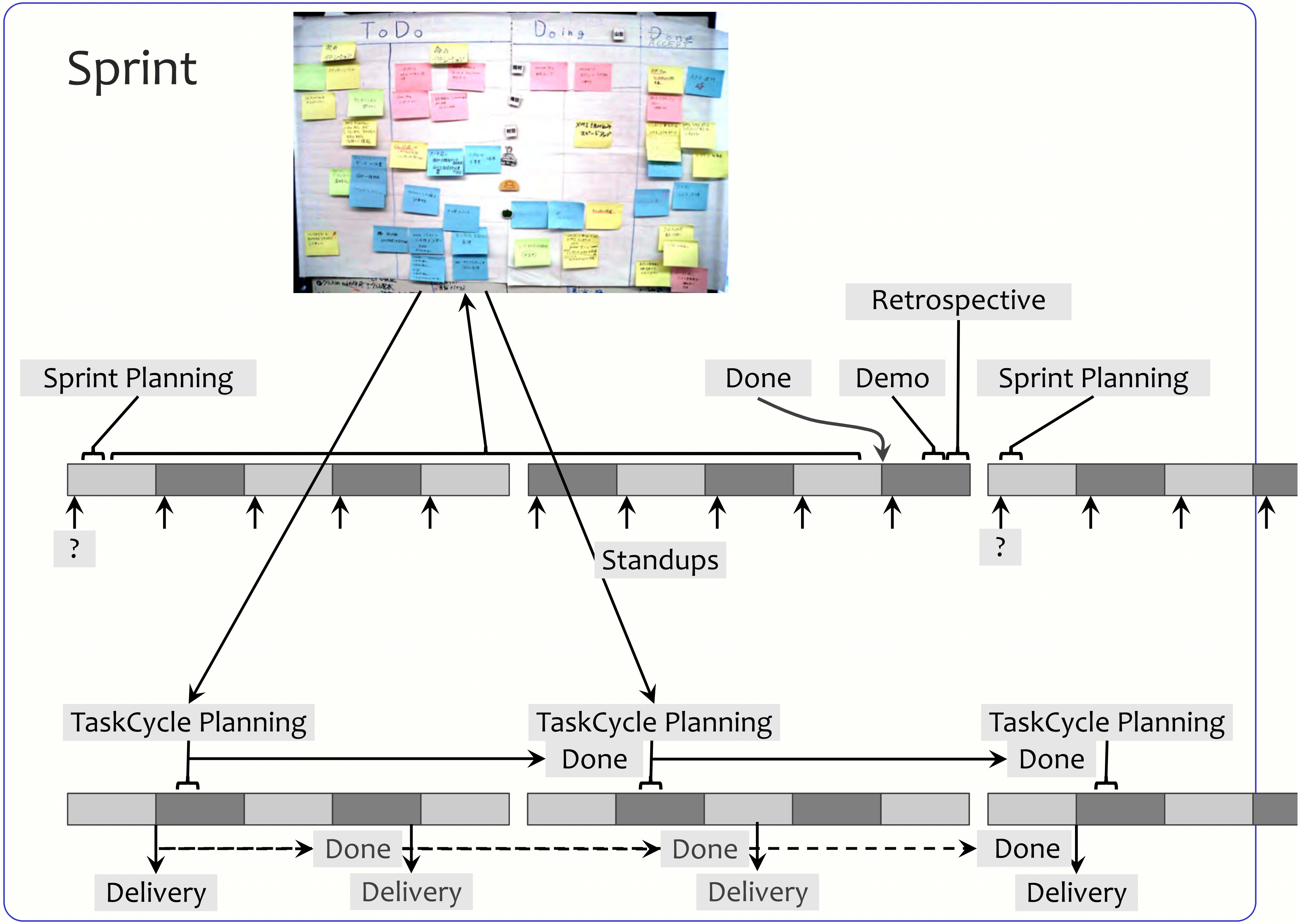
Did you prepare ?

- The Goal of your current work or project
- The Definition of Success
- The most important stakeholder (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what could you do about it? - Failure is not an option!)
- What you think you should and can do *the coming week* to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- Any issues you expect with the above or otherwise with your work or project

To-do lists

- Are you using to-do lists ?
  - List the things you have to do the coming week
  - Did you add effort estimates?
  - Did you check how much time you have available the coming week ?
  - Does what you have to do fit in the available time ?
  - Did you check what you can do and what you cannot do?
  - Did you take the consequence?
- Evo:
  - Because we are short of time, we better use the *limited available time* as best as possible
  - We don't try to do better than *possible*
  - To make sure we do the best possible, we *choose* what to do in the limited available time. We don't just let it happen randomly

Sprint



Retrospective

Sprint Planning

Done

Demo

Sprint Planning

?

Standups

?

TaskCycle Planning

TaskCycle Planning

TaskCycle Planning

Done

Done

Done

Done

Done

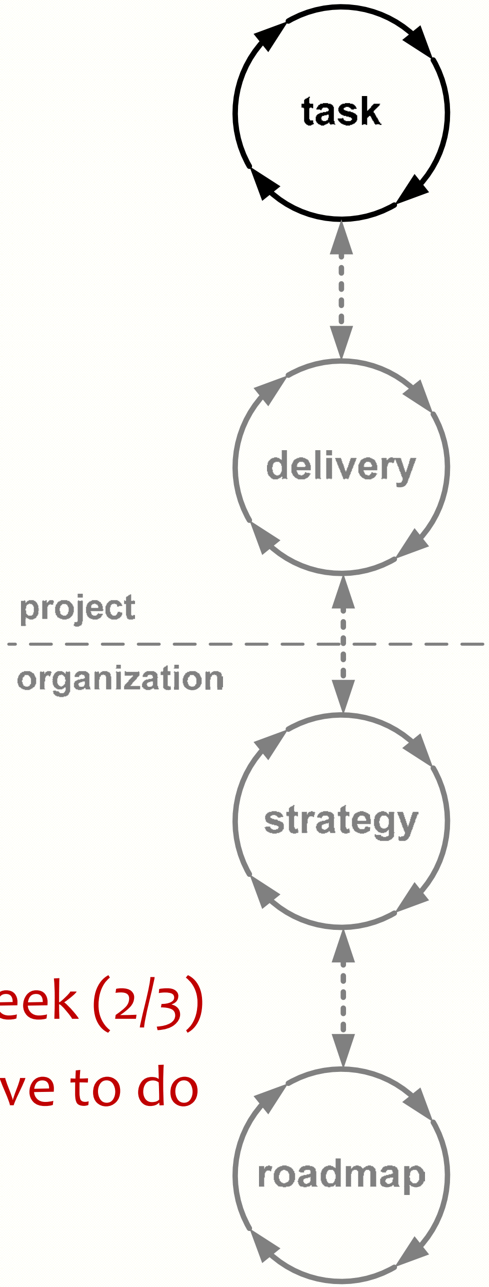
Delivery

Delivery

Delivery

Delivery

Evo Planning: Weekly TaskCycle



- Are we *doing* the right things,
  in the right order,
  to the right level of detail for now
- Optimizing estimation, planning and tracking abilities to better predict the future
- Select highest priority tasks, never do any lower priority tasks, never do undefined tasks
- There are only about 26 plannable hours in a week (2/3)
- In the remaining time: do whatever else you have to do
- Tasks are always done, 100% done

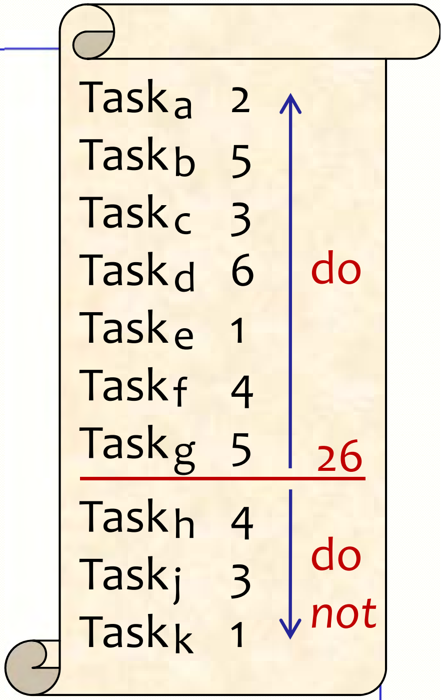
Effort and Lead Time

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

- Effort variations and lead time variations have different causes
- Treat them differently and keep them separate
  - Effort: complexity
  - Lead Time: time-management
    - (effort / lead-time ratio)

Every week we plan

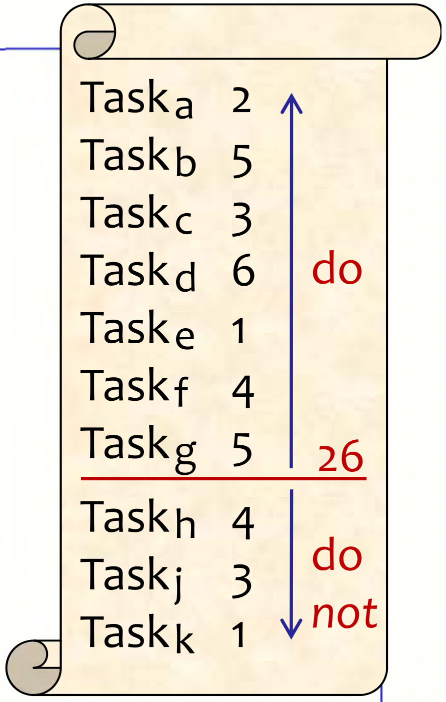
- How much time do we have available
- 2/3 of available time is net plannable time
- What is most important to do
- Estimate effort needed to do these things
- Which most important things fit in the net available time (default 26 hr per week)
- What can, and are we going to do
- What are we *not* going to do



Task$_a$	2	
Task$_b$	5	
Task$_c$	3	
Task$_d$	6	do
Task$_e$	1	
Task$_f$	4	
Task$_g$	5	26
Task$_h$	4	
Task$_j$	3	do
Task$_k$	1	*not*

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

TaskCycle Exercise



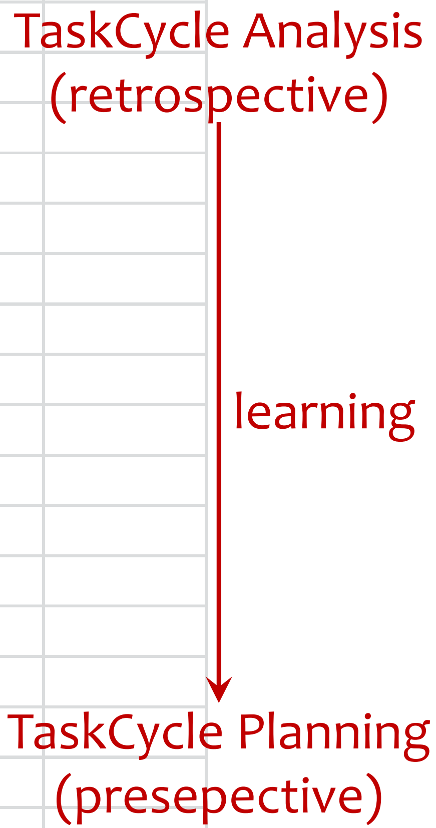
Task$_a$	2	
Task$_b$	5	
Task$_c$	3	
Task$_d$	6	do
Task$_e$	1	
Task$_f$	4	
Task$_g$	5	26
Task$_h$	4	
Task$_j$	3	do
Task$_k$	1	*not*

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

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

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



TaskCycle Analysis
(retrospective)

learning

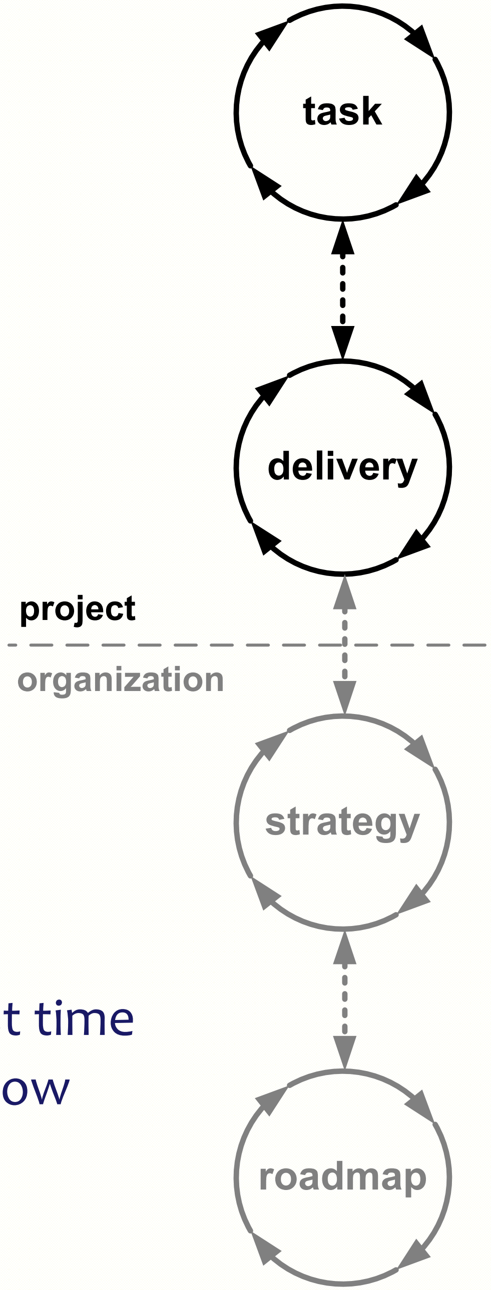
TaskCycle Planning
(presepective)

How fast can it go ?

- Check the amount of work to do (to test) 1
- Chek the tasks assigne to me 2
- Pick up a task to test (recurrently) 1
- Test the tasks (when the previous are tested) 24
- Discuss the deliverables 3
- Prepare for the demo 4
- Hold the demo 2

- 41
- 37
- 26

DeliveryCycle



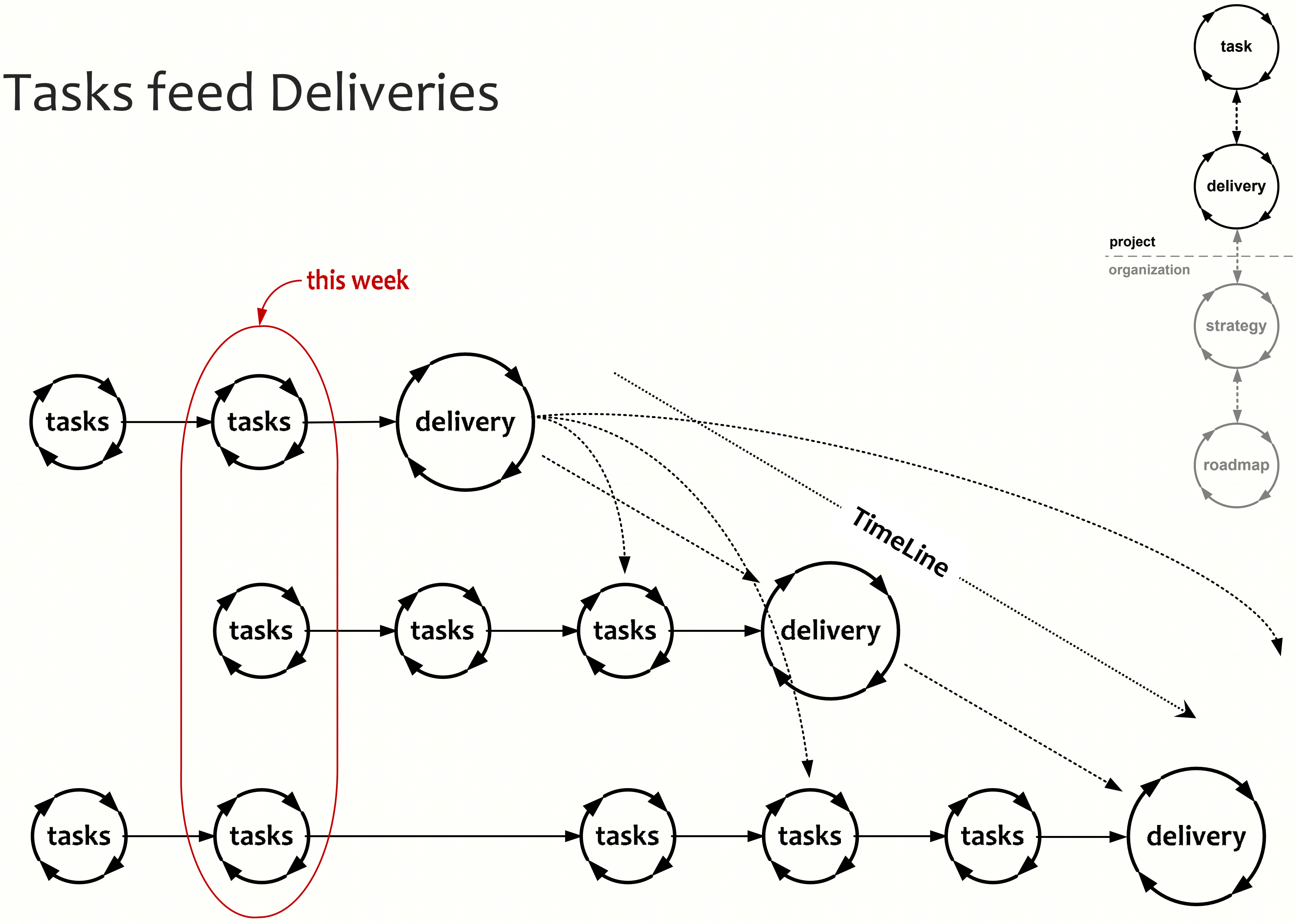
- Are we *delivering* the right things,
  in the right order
  to the right level of detail for now
- Optimizing requirements
  and checking assumptions
  1. What will generate the optimum feedback
  2. We deliver only to eagerly waiting stakeholders
  3. Delivering the juiciest, most important
    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

Do you demo at the end of a Sprint ?

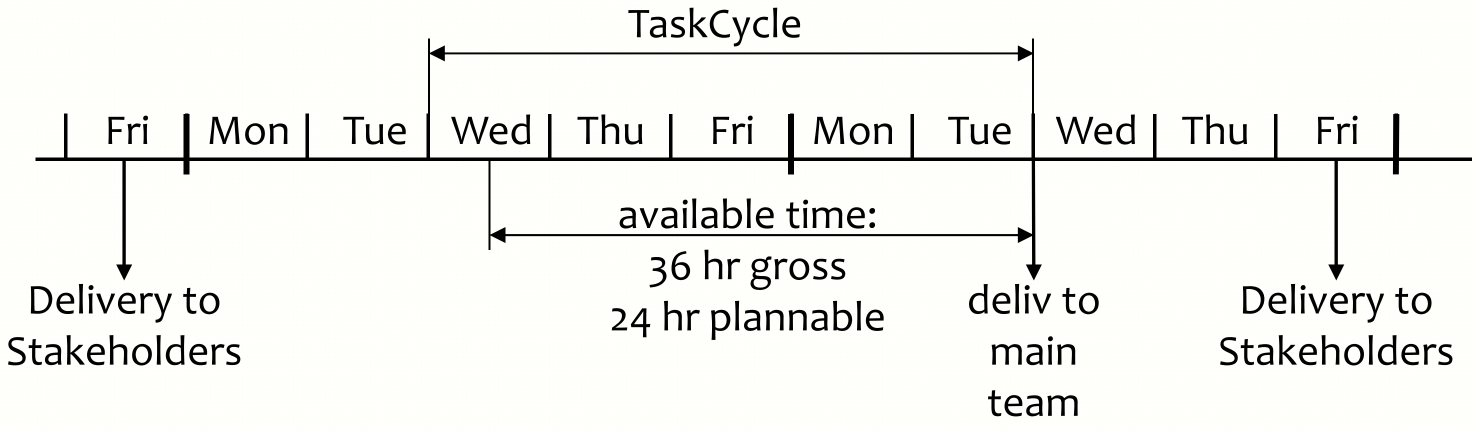
- Give the delivery to the stakeholders
- Keep your hands handcuffed on your back
- Keep your mouth shut
- and o-b-s-e-r-v-e what happens
- Seeing what the stakeholders actually do provides so much better feedback
- Then we can ‘talk business’ with the stakeholders

- Is this what you do ?
- Success criterion: “No Questions, No Issues”

Tasks feed Deliveries



Designing a Delivery



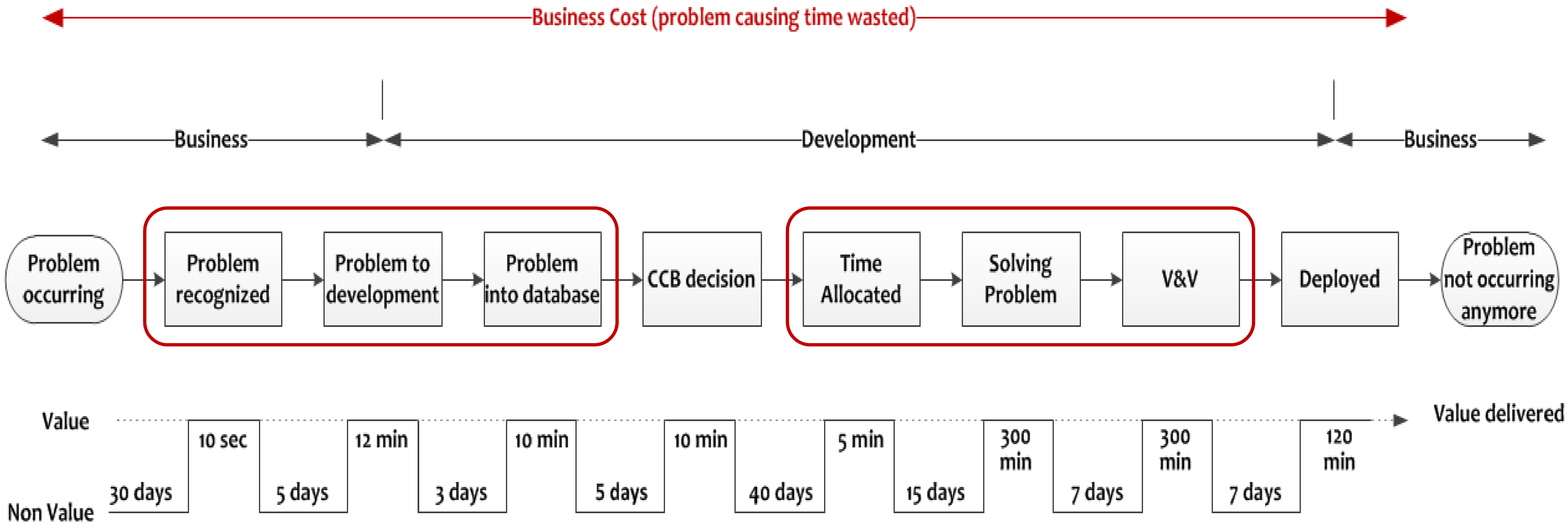
Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

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
...	

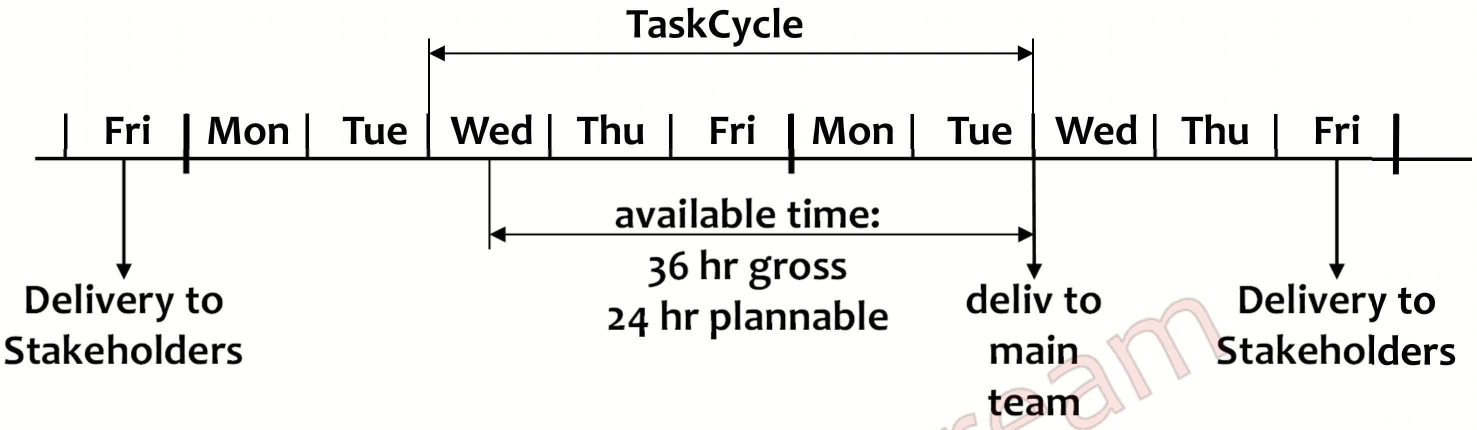
Jerome	
XMLa	3
XMLb	3
...	

Value stream mapping



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

Designing a Delivery



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

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

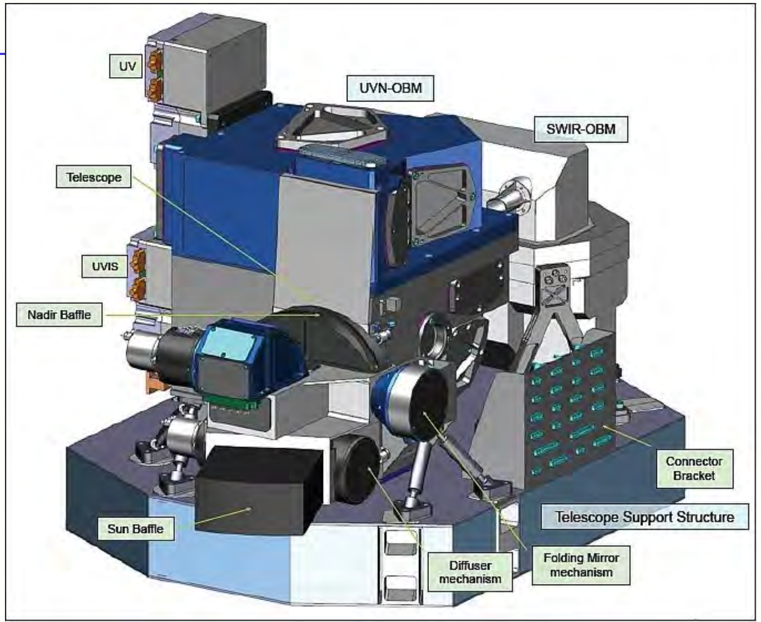
Gregory (later)	
Draft design	0
Finish design	0
...	

Jerome	
XMLa	3
XMLb	3
...	

Why is this important ?

- TaskCycle Planning is *not*
  just planning the work for the coming week
- Half (±30%) of what people do in projects later proves not having been necessary
- During the TaskCycle planning we can very efficiently see
  - What our colleagues think they're going to do
  - Make sure they're going to work on the most important things
  - Not on unnecessary things
  - In line with the architecture and design
  - Leading most efficiently to the goal of the delivery
- We'll see two cases where the architect
  led the project to success in record time

Earth Observation Satellite



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

Awful schedule pressure !

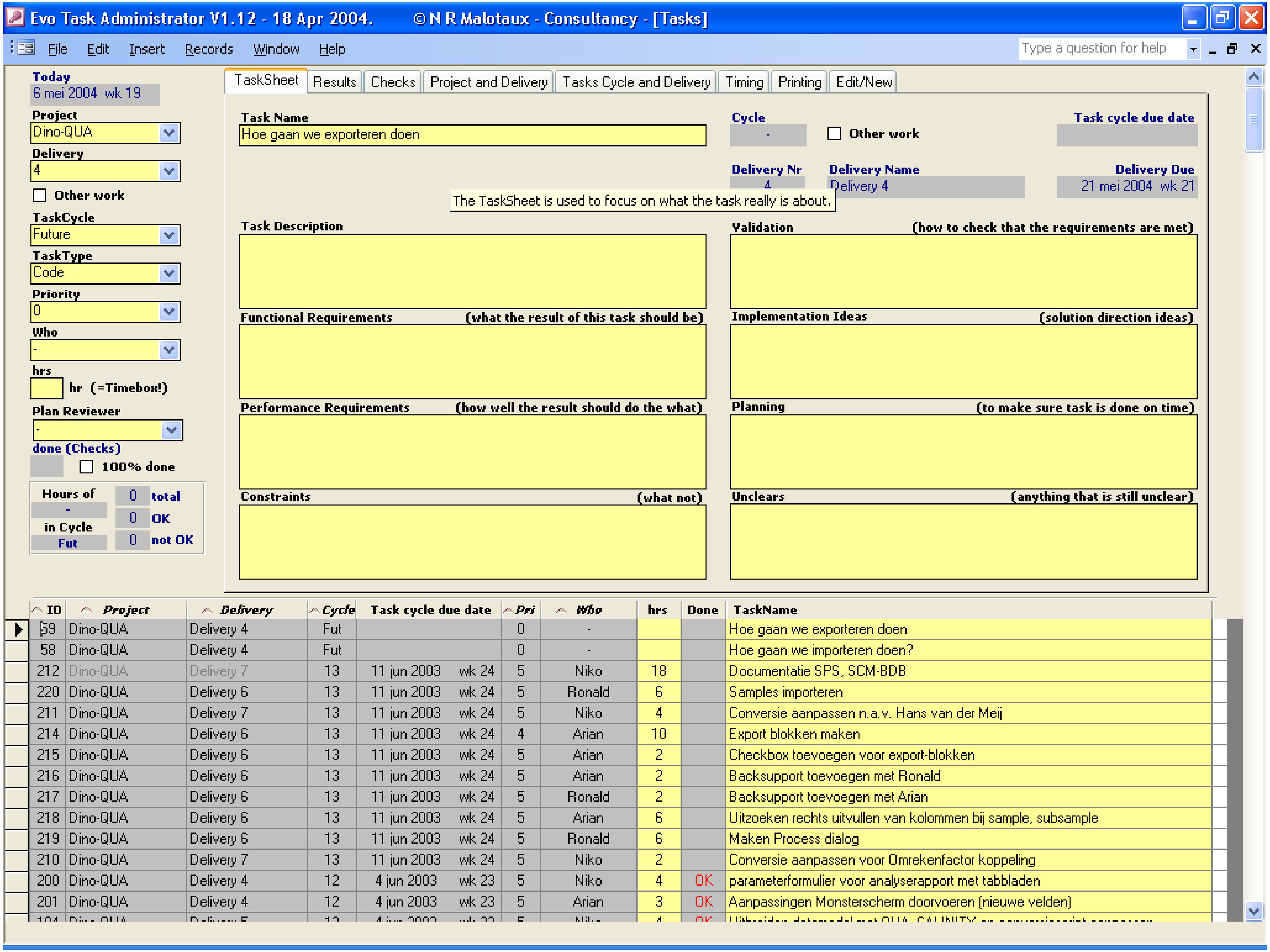
- Meeting with sub-contractors in three weeks
- Many documents to review
- Impossible deadline

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

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

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

available	2 x 26	52



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

Today: 6 mei 2004 wk 19

Project: Dino-QUA

Delivery: 4

□ Other work

TaskCycle: Future

TaskType: Code

Priority: 0

Who: -

hrs: hr (=Timebox!)

Plan Reviewer: -

done (Checks)

□ 100% done

Hours of - in Cycle Fut: 0 total, 0 OK, 0 not OK

Task Name: Hoe gaan we exporteren doen

Cycle: -

□ Other work

Task cycle due date

Delivery Nr: 4

Delivery Name: Delivery 4

Delivery Due: 21 mei 2004 wk 21

The TaskSheet is used to focus on what the task really is about.

Task Description

Validation (how to check that the requirements are met)

Functional Requirements (what the result of this task should be)

Implementation Ideas (solution direction ideas)

Performance Requirements (how well the result should do the what)

Planning (to make sure task is done on time)

Constraints (what not)

Unclears (anything that is still unclear)

ID	Project	Delivery	Cycle	Task cycle due date	Pri	Who	hrs	Done	TaskName
59	Dino-QUA	Delivery 4	Fut		0	-			Hoe gaan we exporteren doen
58	Dino-QUA	Delivery 4	Fut		0	-			Hoe gaan we importeren doen?
212	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	18		Documentatie SPS, SCM-BDB
220	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Samples importeren
211	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	4		Conversie aanpassen n.a.v. Hans van der Meij
214	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	4	Arian	10		Export blokken maken
215	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	2		Checkbox toevoegen voor export-blokken
216	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	2		Backsupport toevoegen met Ronald
217	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	2		Backsupport toevoegen met Arian
218	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Arian	6		Uitzoeken rechts uitvullen van kolommen bij sample, subsample
219	Dino-QUA	Delivery 6	13	11 jun 2003 wk 24	5	Ronald	6		Maken Process dialog
210	Dino-QUA	Delivery 7	13	11 jun 2003 wk 24	5	Niko	2		Conversie aanpassen voor Omrekenfactor koppeling
200	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Niko	4	OK	parameterformulier voor analyserapport met tabbladen
201	Dino-QUA	Delivery 4	12	4 jun 2003 wk 23	5	Arian	3	OK	Aanpassingen Monsterscherm doorvoeren (nieuwe velden)

Developing a new oscilloscope

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

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

Results

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

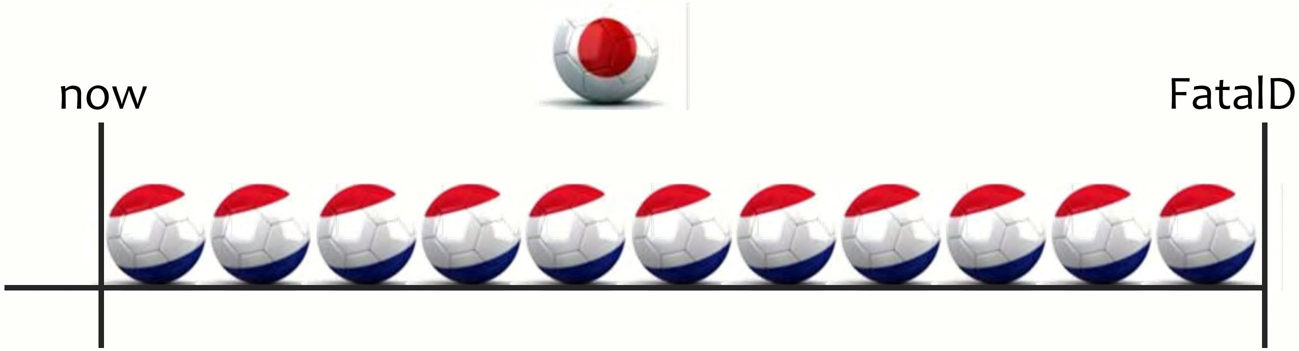
www.malotaux.nl/doc.php?id=19 chapter 4.7.1, page 70

Software project in Poland

- ‘Mission Impossible’: Delivery deadline in 6 weeks
- Will you succeed ?
- No !
- Failure is not an Option !
- Changed their way of working
- Delivered to amazed customer in 5 weeks
- Proudly confided: “Not working overtime !”

If we add something …

If we add something, something else will not be done



Rather than letting it happen randomly

We better decide what will happen

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 the fence?”
- What do you do?

Sorry, some pictures removed for confidentiality

- 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.

Quality on Time

- Evo development gradually delivers function and performance, while eating up resources
- Not just what to deliver, but also how we are going to deliver it and whether this is the right way to deliver it
- EvoPlanning prevents a lot of bad implementations before they are implemented, saving a lot of time

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 ?

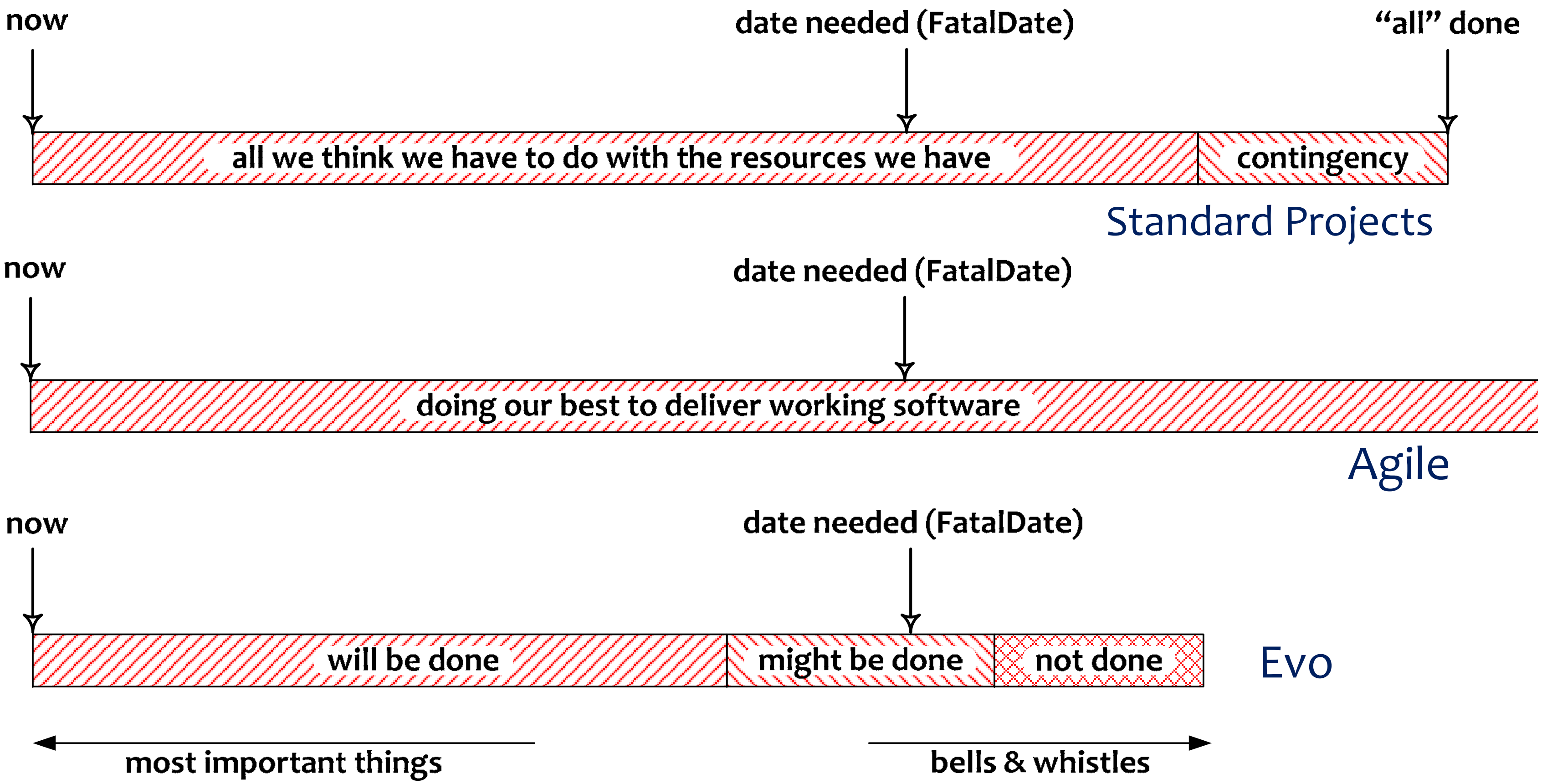
TimeLine

How to make sure we get

the Right Results at the Right Time

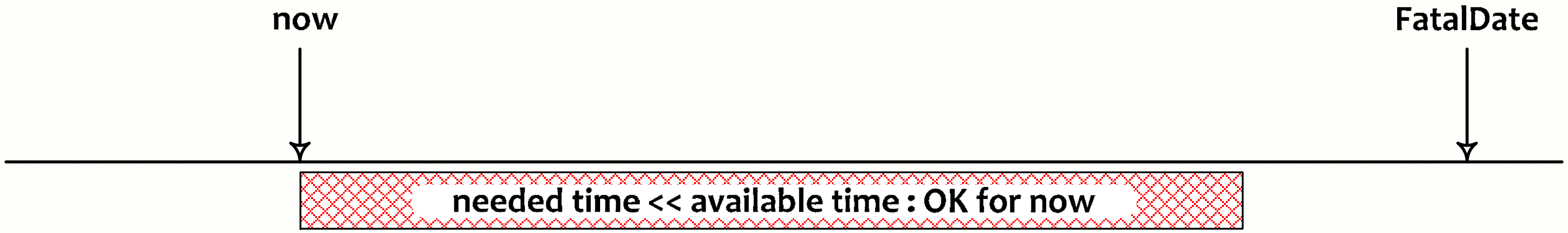
TimeLine

What the customer wants, he cannot afford



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

If it easily fits ...

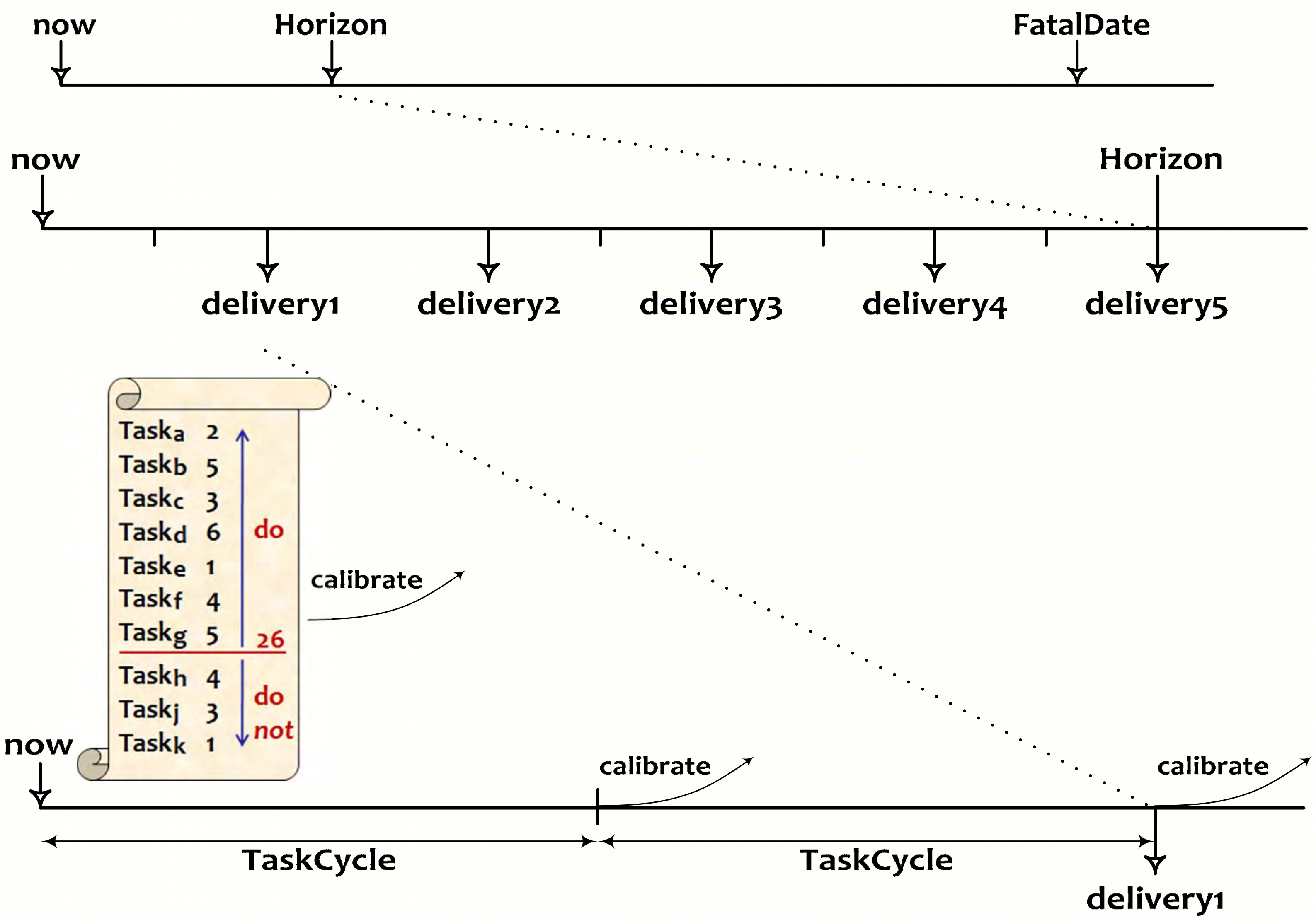


now

FatalDate

needed time << available time : OK for now

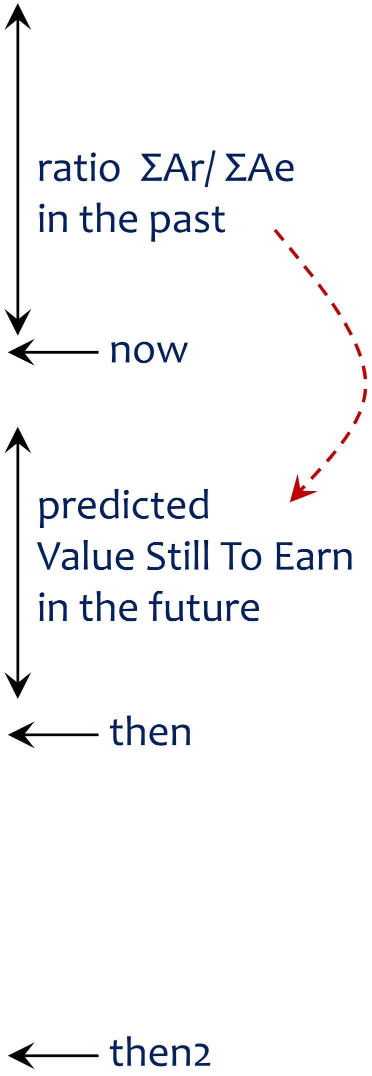
Result to Tasks and back



Task$_a$	2
Task$_b$	5
Task$_c$	3
Task$_d$	6
Task$_e$	1
Task$_f$	4
Task$_g$	5
Task$_h$	4
Task$_j$	3
Task$_k$	1

Calibration

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



Calibration Factor

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

Value Still To Earn

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

Predicting *what* will be done *when*

for the project

to report

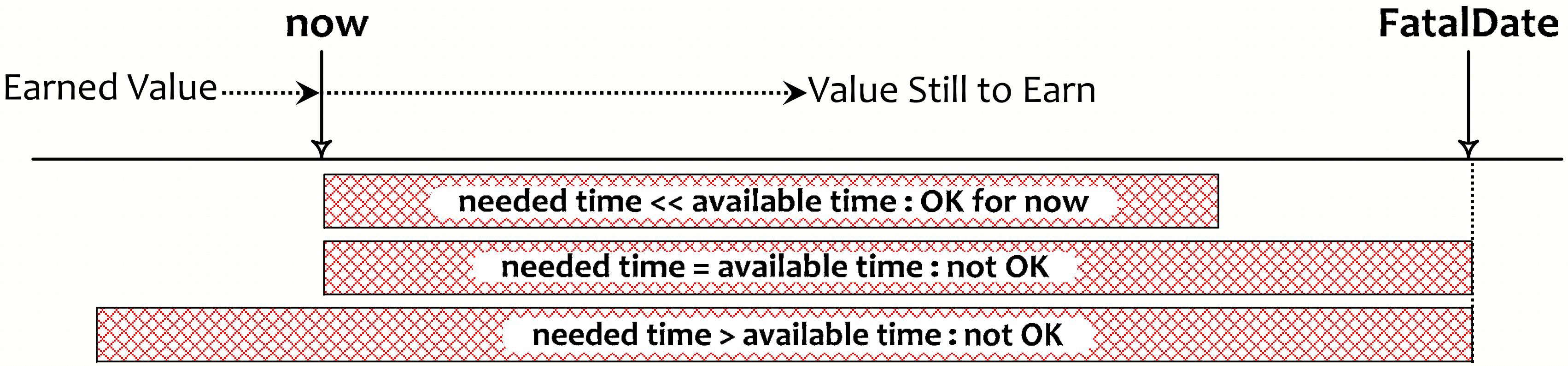
Line	Activity	Estim	Spent	Still to spend	Ratio real/es	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009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

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



now

FatalDate

Earned Value

Value Still to Earn

needed time << available time : OK for now

needed time = available time : not OK

needed time > available time : not OK

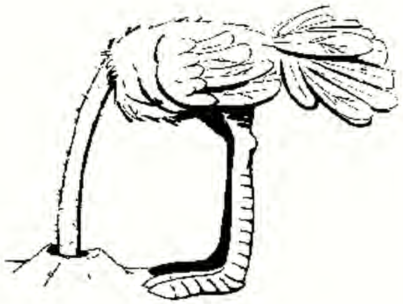
- Value Still to Earn

versus

- Time Still Available

If the match is over, you cannot score a goal

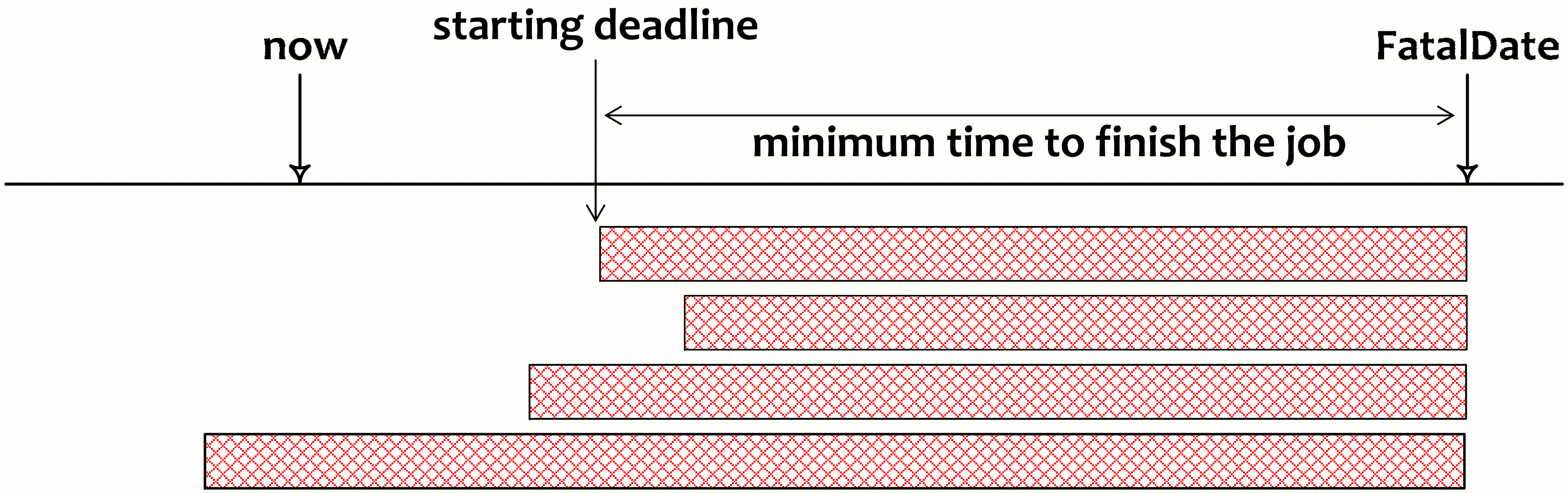
FatalDay



- We take time seriously
- 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*

Even more important: *Starting Deadlines*

- Starting deadline
  - Last day we can start to deliver by the end deadline
  - Every day we start later, we will end later



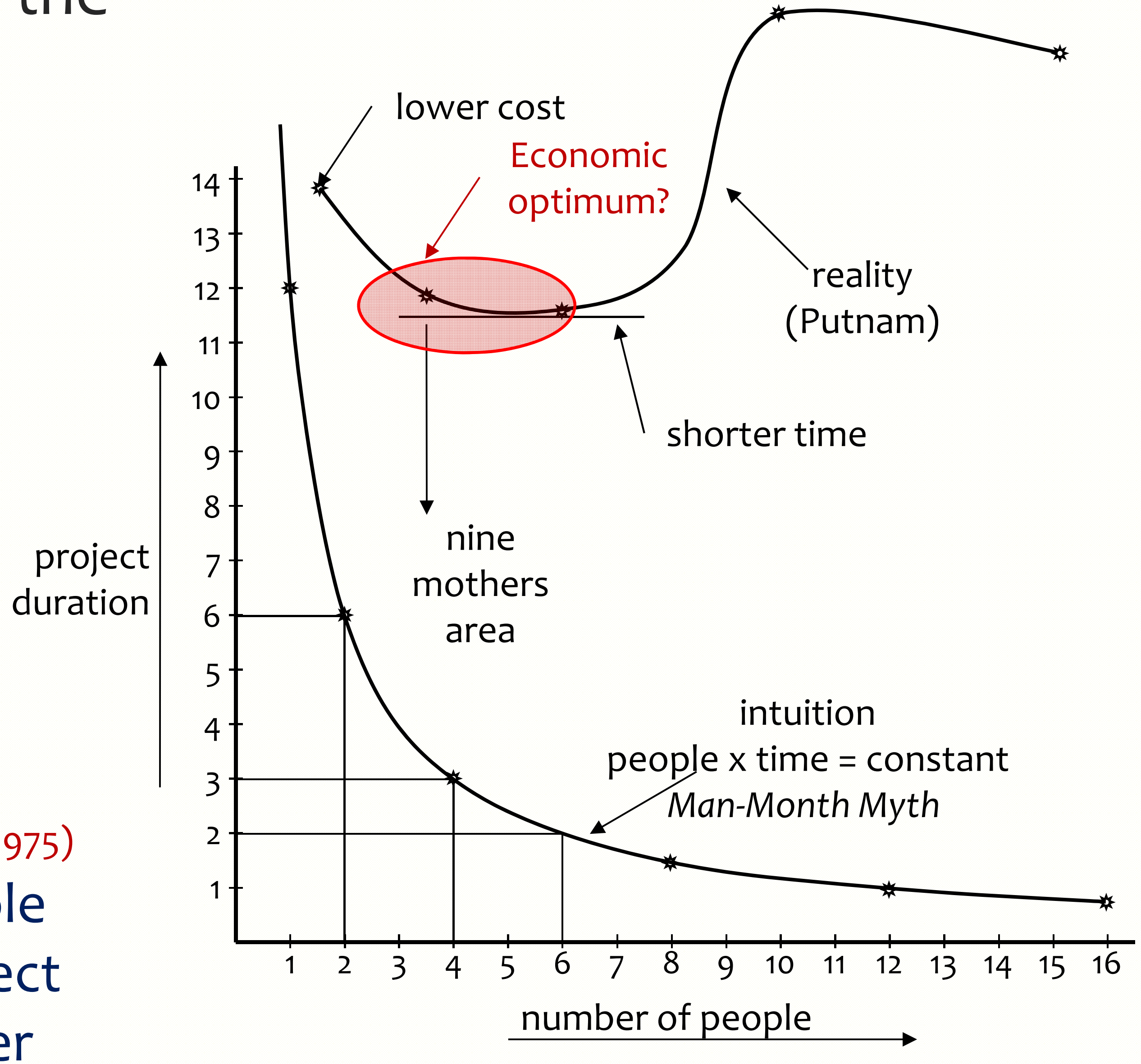
How can we be On Time ?

Deceptive options

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

Intuition often guides us into the wrong direction

The Myth of the Man-Month



Brooks' Law (1975)

Adding people to a late project makes it later

Saving time

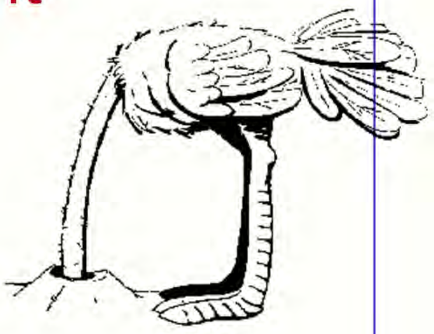
Continuous elimination of waste

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

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

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



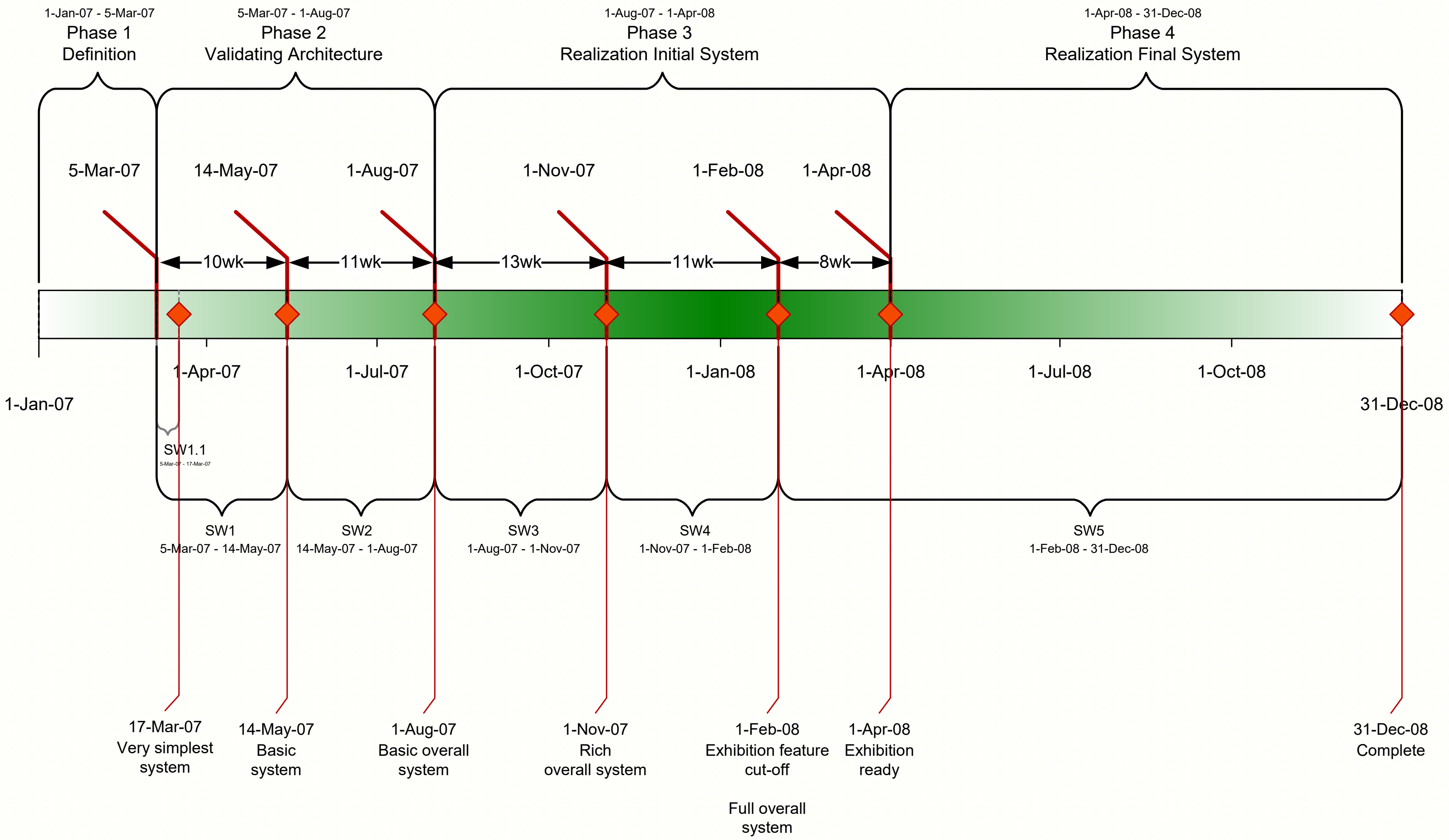
If, and only if, we are serious about time !

Estimation techniques used

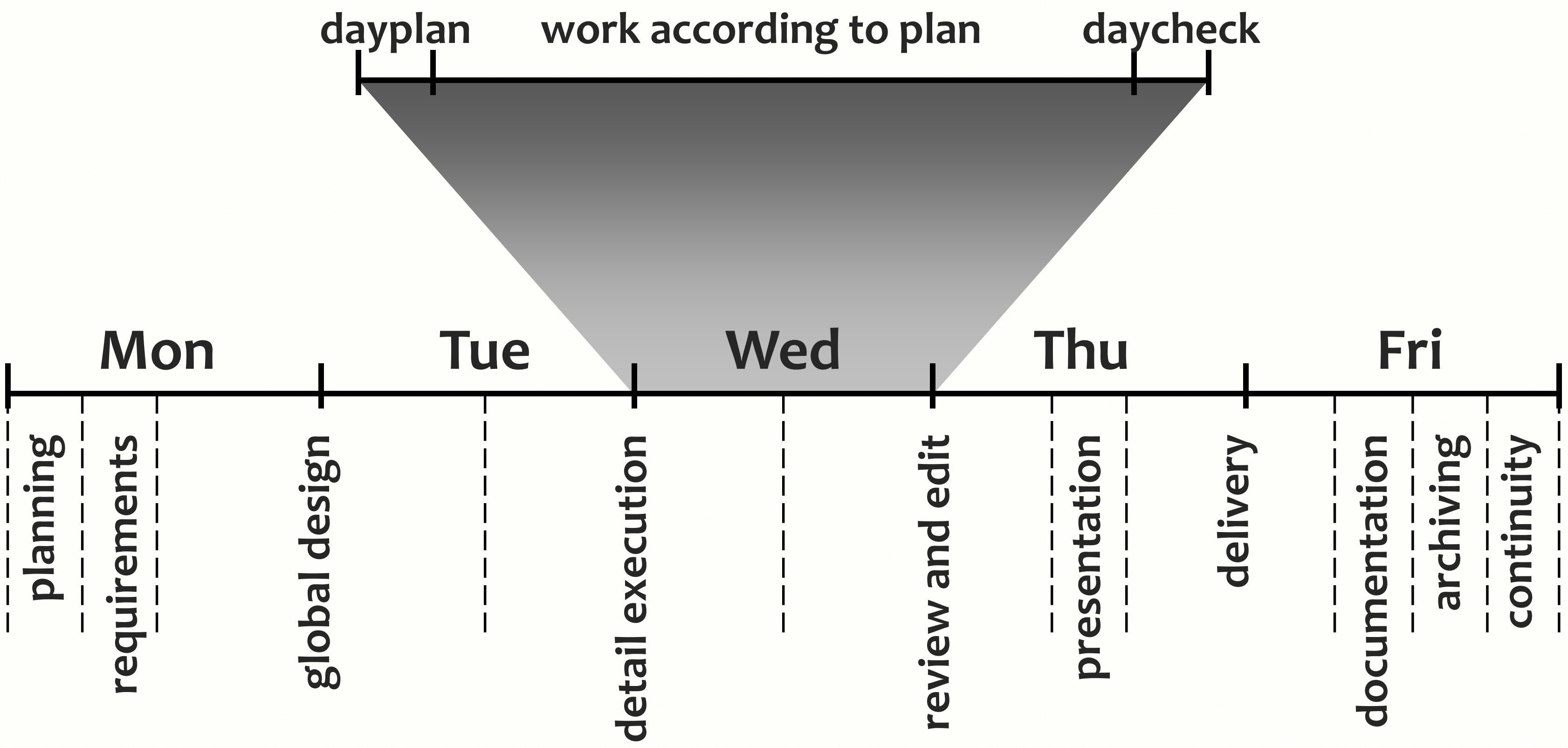
- Just-enough estimation (don't do unnecessary things)
  - Maximizing Return-on-Investment and Value Delivered
- Changing from optimistic to realistic predictions
  - Estimation of Tasks in the TaskCycle
  - Prediction what will be done when in TimeLine
- 0th order estimations (ball-park figures)
  -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

TimeLine examples

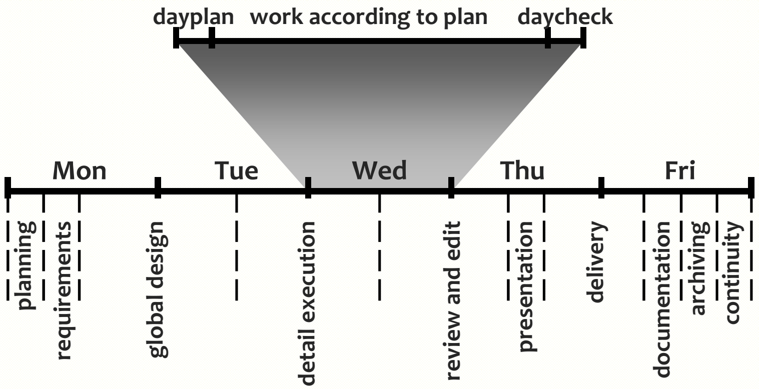
TimeLine example



5 day project model



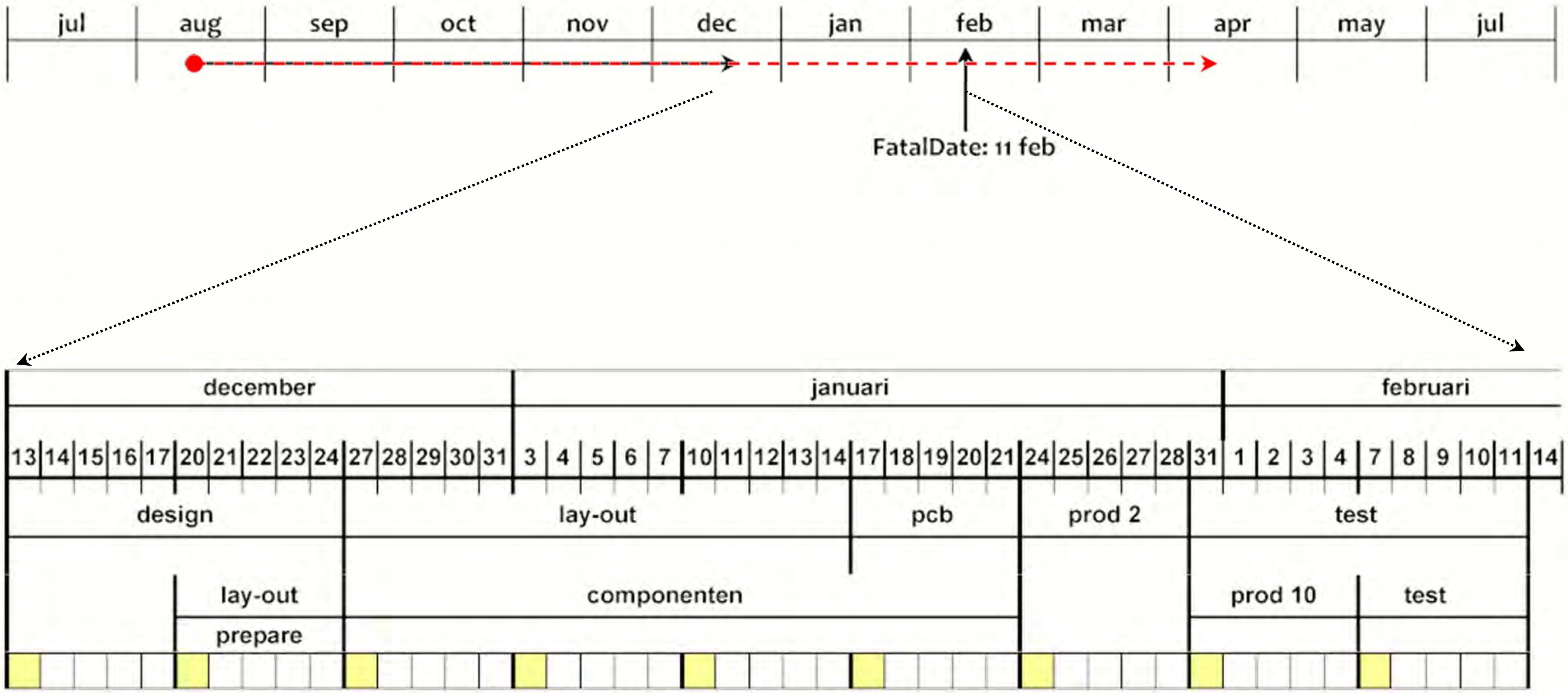
Available TimeBoxes



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

Sorry, picture removed for confidentiality



jul aug sep oct nov dec jan feb mar apr may jul

FatalDate: 11 feb

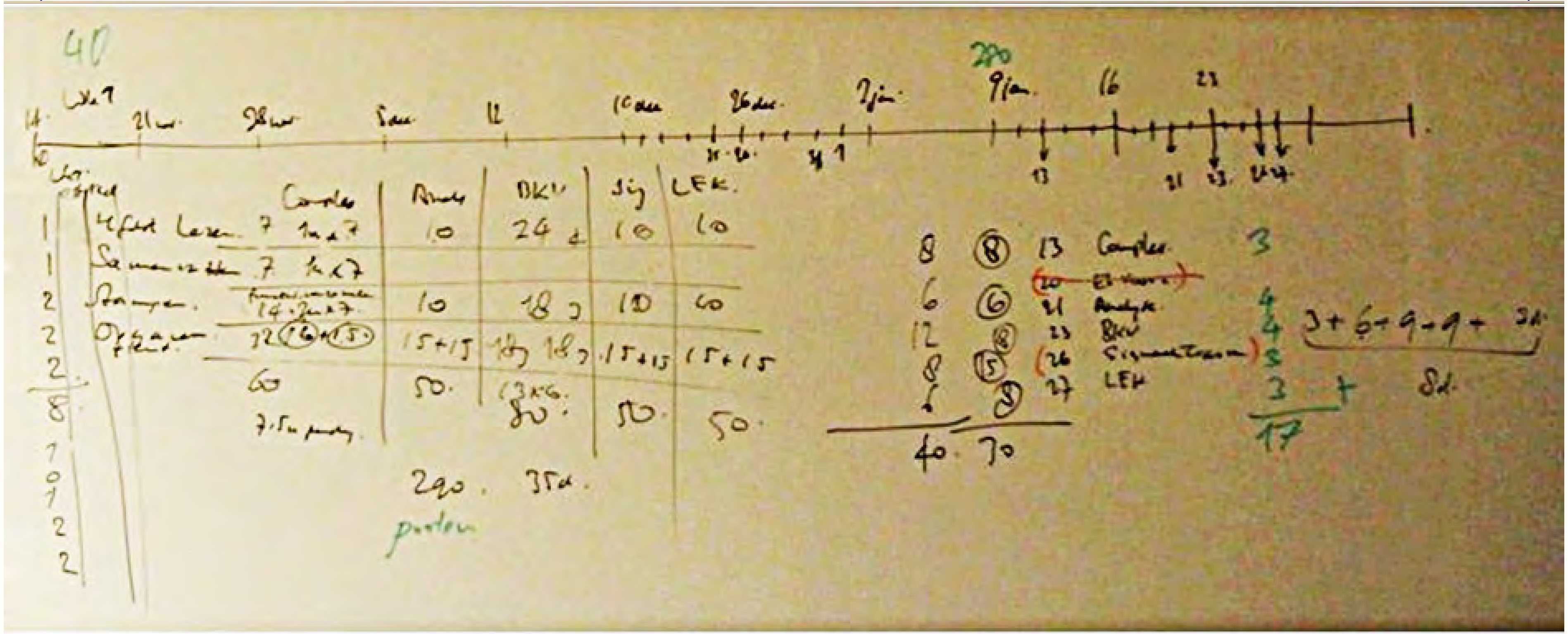
december januari februari

13 14 15 16 17 20 21 22 23 24 27 28 29 30 31 3 4 5 6 7 10 11 12 13 14 17 18 19 20 21 24 25 26 27 28 31 1 2 3 4 7 8 9 10 11 14

design lay-out pcb prod 2 test

lay-out prepare componenten prod 10 test

Preparing for student exams



Sorry, picture removed for confidentiality

Sorry, picture removed for confidentiality

Sorry, picture removed for confidentiality

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!)

Help !

We have a

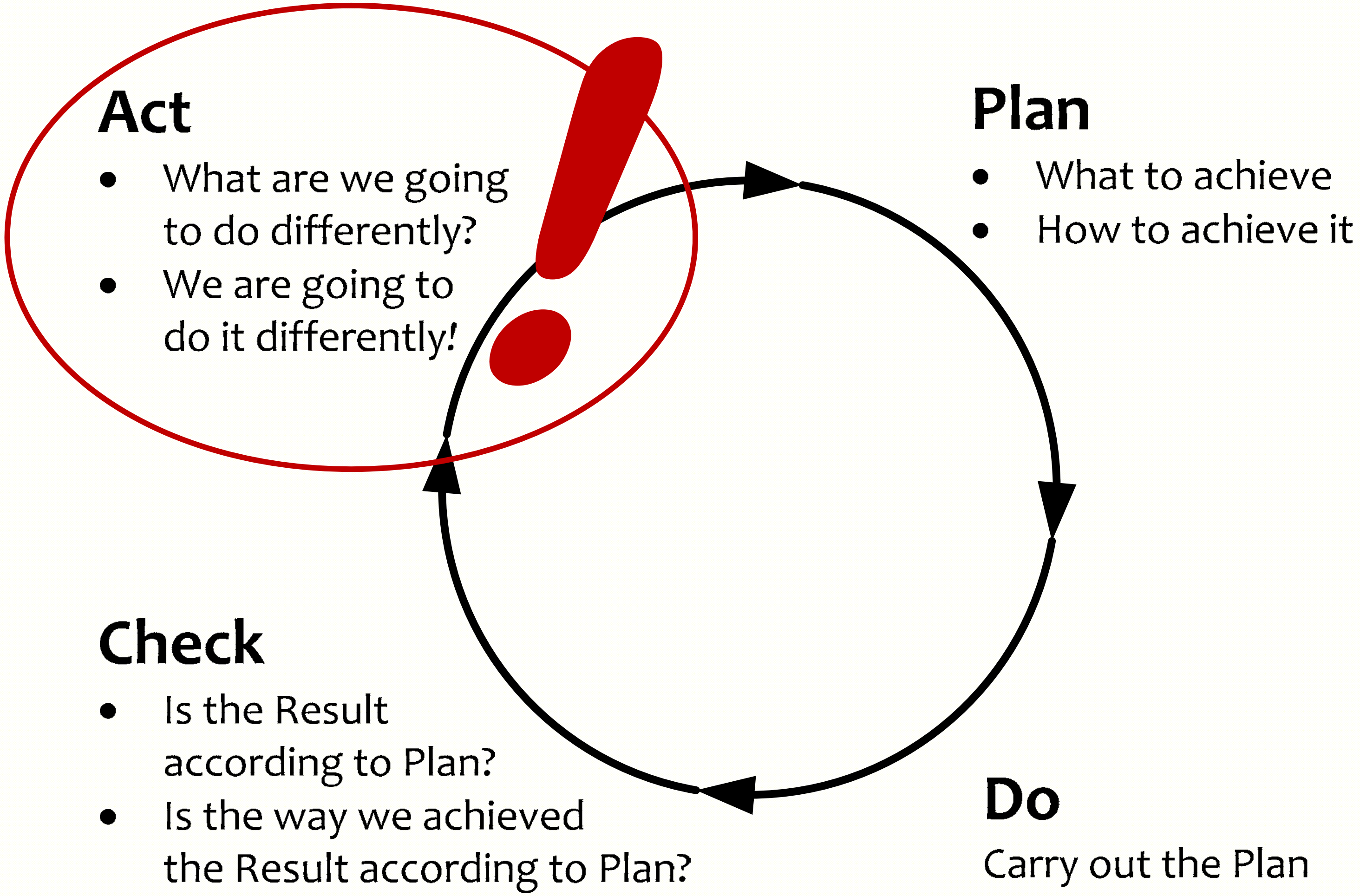
QA problem !

Help !
We have a QA problem !

- Large stockpile of modules to test
  (hardware, firmware, software)
- You shall do Full Regression Tests
- Full Regression Tests take about 15 days each
- Too few testers ("Should we hire more testers ?")
- Senior Tester paralyzed
- Can we do something about this?

www.malotaux.nl/booklets - booklet#8

Do you think you can help us ?



Act

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

Plan

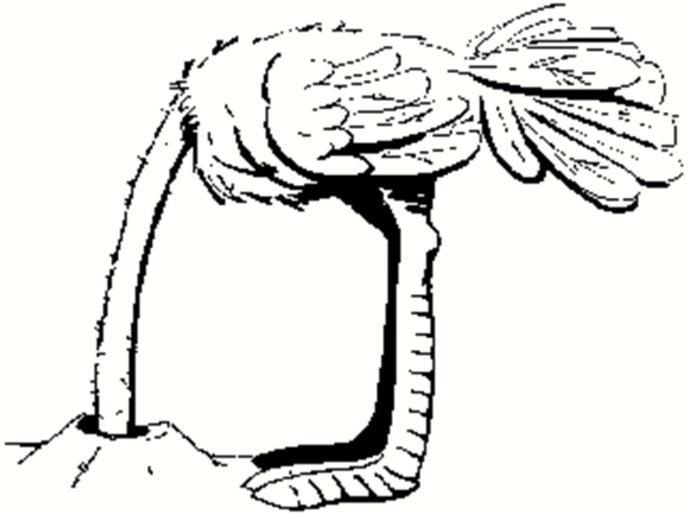
- What to achieve
- How to achieve it

Check

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

Do

Carry out the Plan



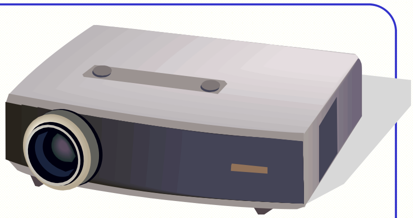
In stead of complaining about a problem …

(Stuck in the Check-phase)

Let’s do something about it !

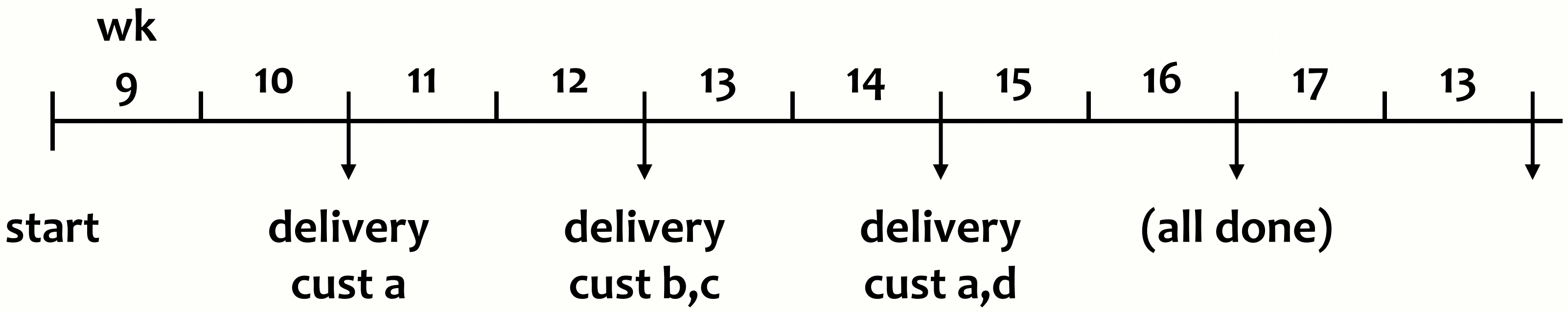
(Moving to the Act-phase)

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



Line	Activity	Estim	Alter native	Junior tester	Devel opers	Customer	Will be done (now=22Feb)
1	**Package 1**	17	2	17	4	HT	
2	**Package 2**	8	5		10	Chrt	
3	**Package 3**	14	7	5	4	BMC	
4	**Package 4 (wait for feedback)**	11				McC?	
5	**Package 5**	9	3		5	Ast	
6	**Package 6**	17	3	10	10	?	
7	**Package 7**	4	1		3	Cli	
8	**Package 8.1**	216	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



wk

9 10 11 12 13 14 15 16 17 13

start

delivery cust a

delivery cust b,c

delivery cust a,d

(all done)

Selecting the priority order of customers to be served

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

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

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*

Higher Productivity

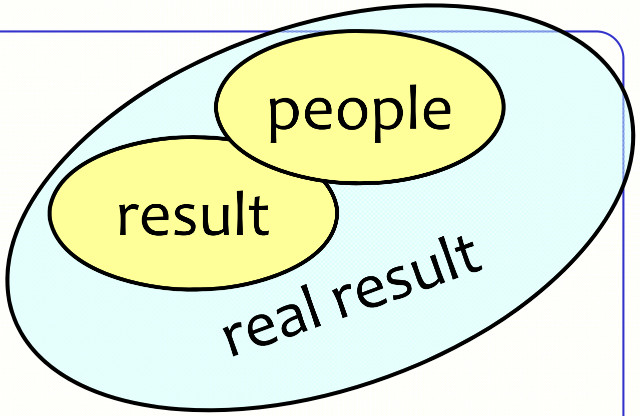
- All functionality we produce *does already exist*
- The real reason for running our projects is creating *better performance*
- Types of improvement:
  - Less loss
  - More profit
  - Doing the same in shorter time
  - Doing more in the same time
  - Being happier than before
- In short: *Adding Value*

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

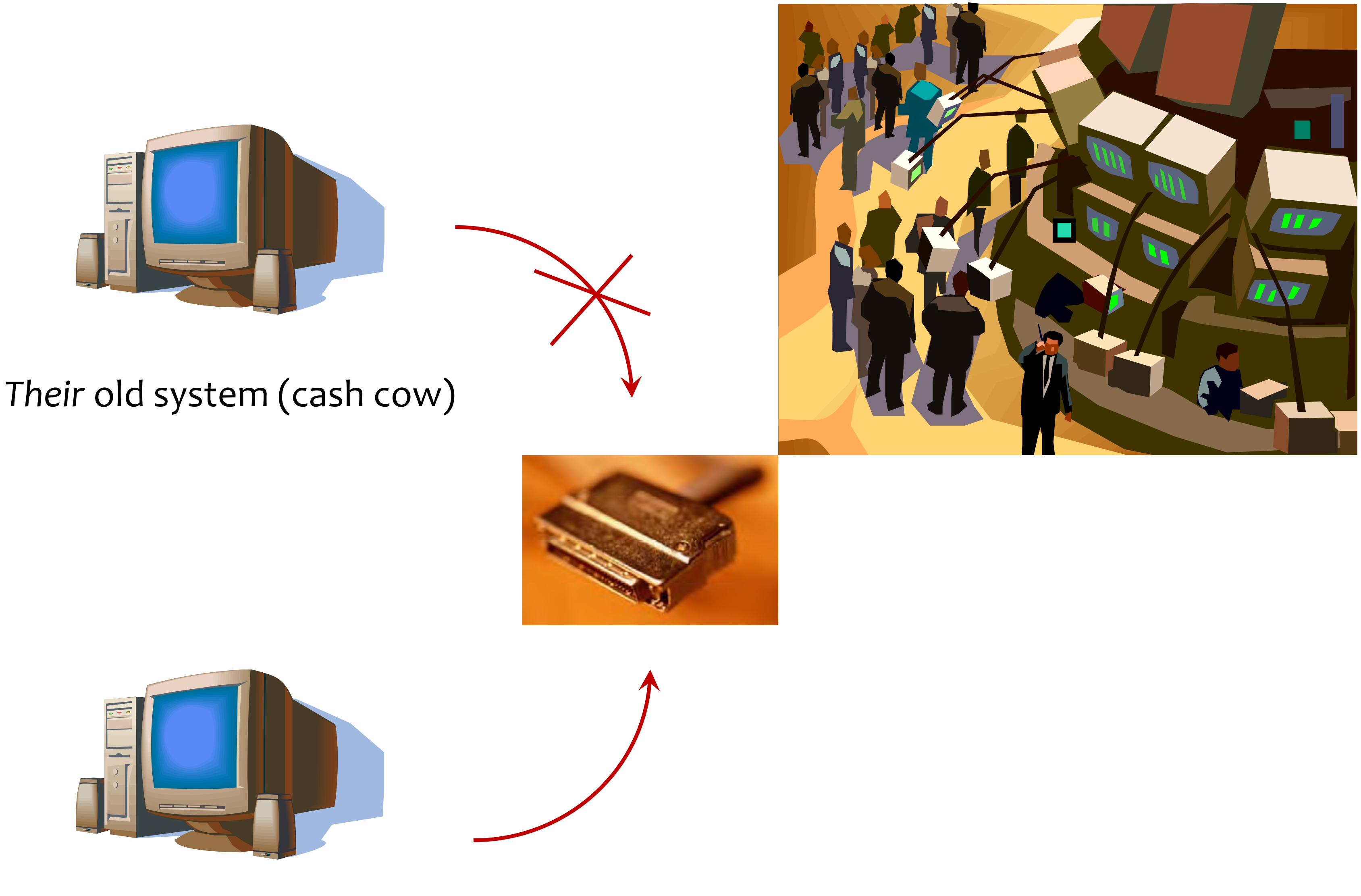
Stakeholders are (not only) people



- Every project has some 30±20 Stakeholders
- Stakeholders have a stake in the project
- The concerns of Stakeholders are often contradictory
    - Apart from the Customer *they don't pay*
    - So they *have no reason to compromise !*
- 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?

Exercise to create focus

- The most important stakeholder of your work
  (*Who* is waiting for it?)
- The most important real requirement
  (What is (s)he waiting for?)
- How much value improvement does this stakeholder expect
  (3 or 7?)
- Was this the focus of your work the coming week ?

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

Customer requirements ?

Nice input,
to be taken seriously

Customer Specification

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

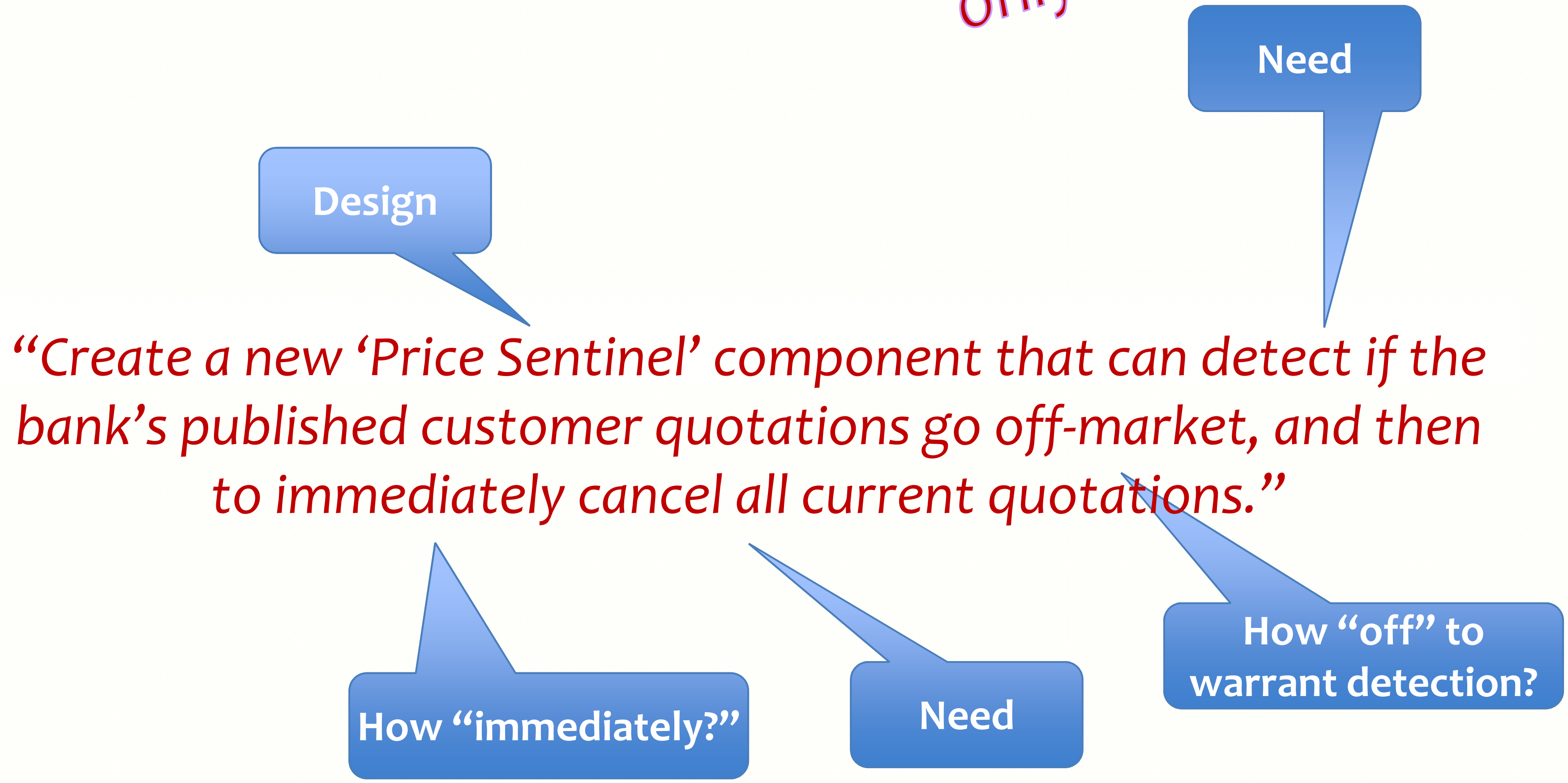
- Was it complete ?
- Was it clear ?
- Did it show the problem to be solved ? (or was it a *solution* ?)

Do you have examples of requirements ?

Is this a Requirement ?

or ‘nice input’, to be taken seriously ?

First develop the problem, only then the solution



“Create a new ‘Price Sentinel’ component that can detect if the bank’s published customer quotations go off-market, and then to immediately cancel all current quotations.”

Ref http://rsbatechnology.co.uk

Using 5 Whys

Why do you need a “Price Sentinel” ?

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

Ref http://rsbatechnology.co.uk

First try

New ‘Price Sentinel’ component:

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

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

Prioritize solutions by Impact Estimation

	Kill button	Price Sentinel
Cancel 600 → 1 sec		
Cost		

Requirements with Planguage

ref Tom Gilb

Definition:

RQ27: Speed of Luggage Handling at Airport

Specific

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

Measurable

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

Benchmarks (Playing Field):

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

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

Attainable

Record: 57 sec [competitor x, Jan 2012]

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

Requirements:

Time

Realizable

Tolerable: < 10 min [99%, Q4] ← SLA

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

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

Requirements weren’t the problem

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

Tom Gilb quote

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

Requirements have Rules

Some examples:

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

Rule 2: No design (solutions) in the requirements

Rule 3: Unambiguous

Rule 4: Clear to test

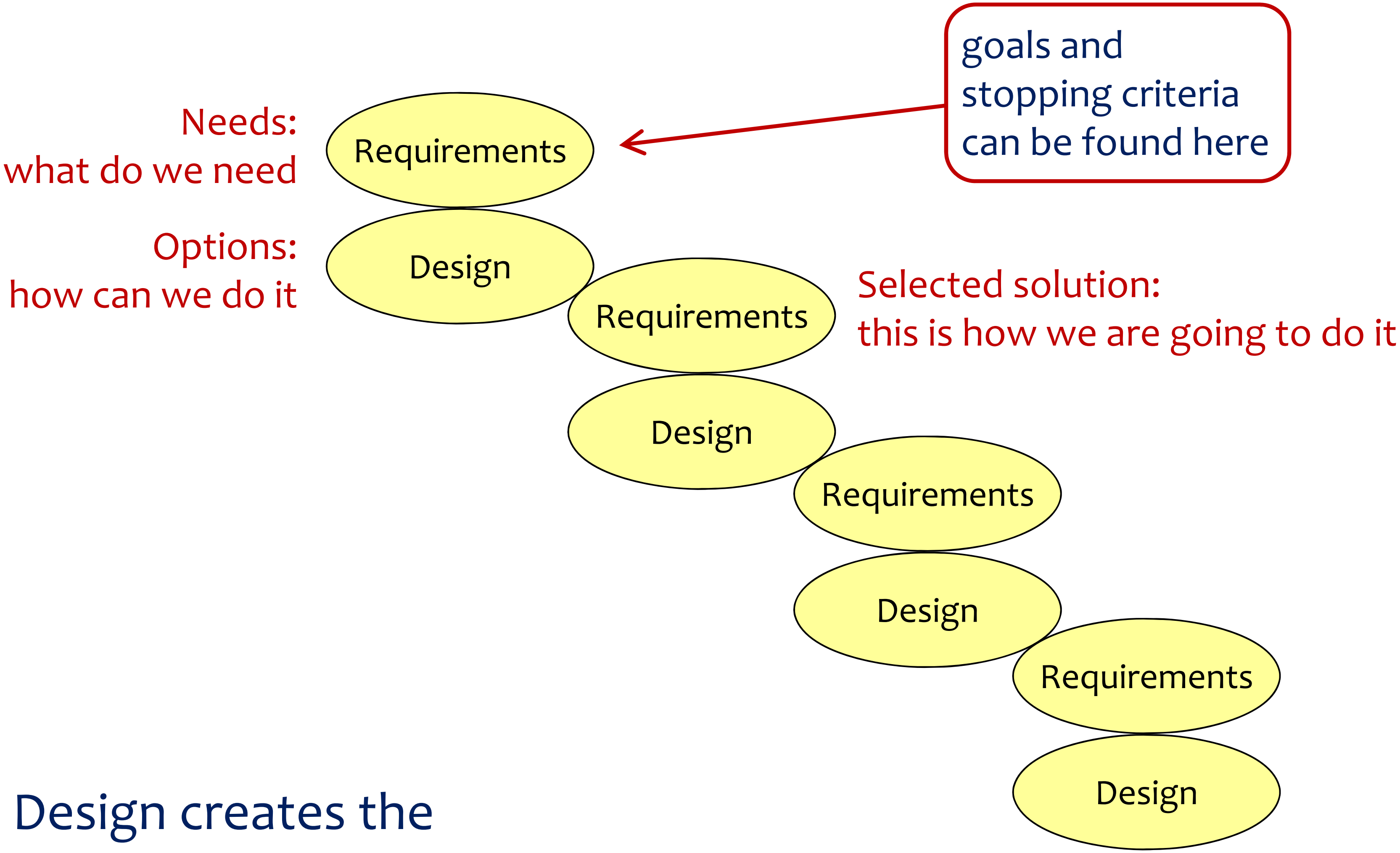
Typical requirements found:

- The system should be extremely user-friendly
- The system must work exactly as the predecessor
- The system must be better than before
- It shall be possible to easily extend the system's functionality on a modular basis, to implement specific (e.g. local) functionality
- It shall be reasonably easy to recover the system from failures, e.g. without taking down the power

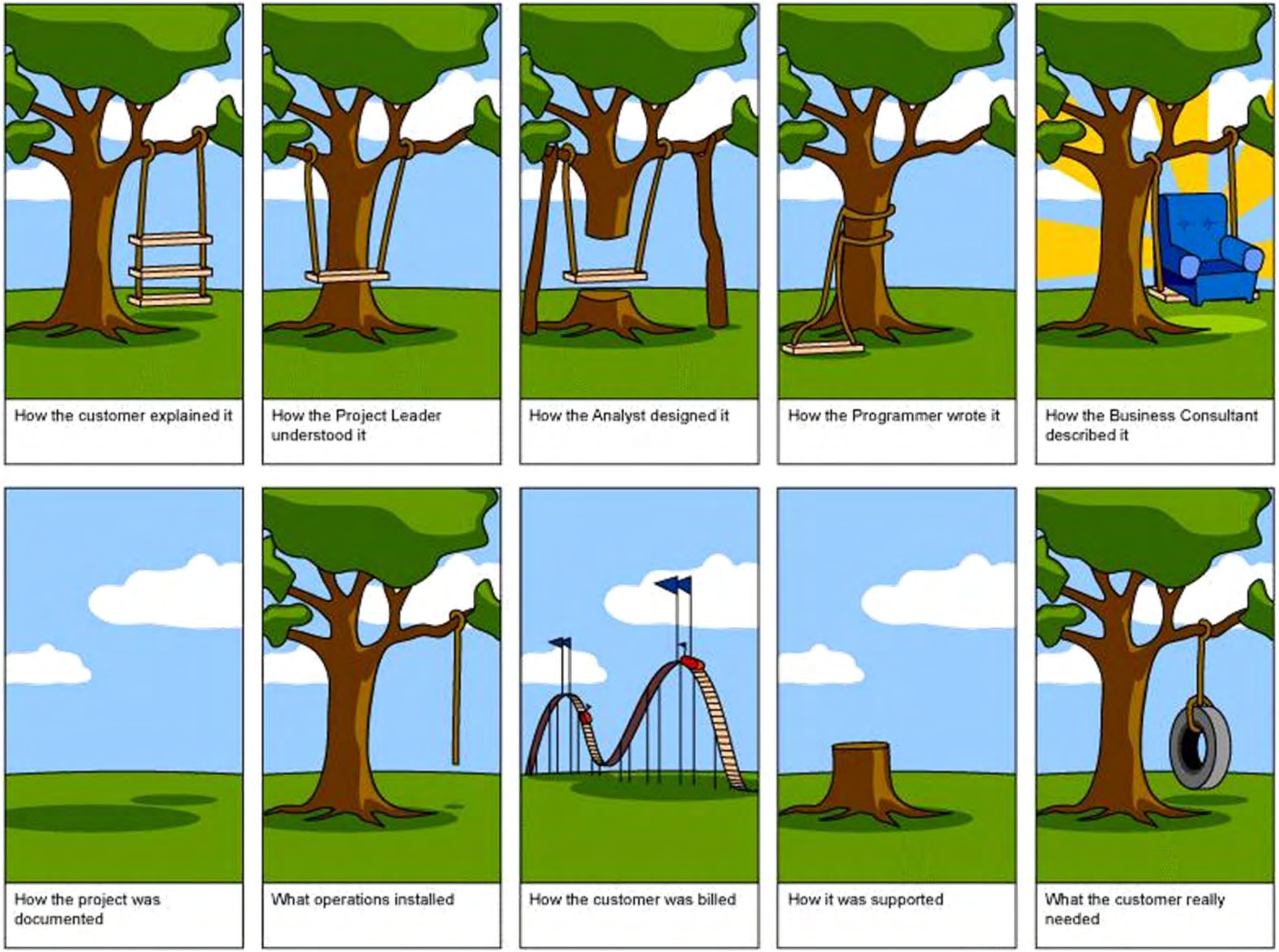
Recent project

- 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 ?

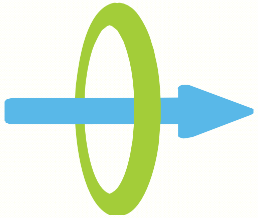
No Design in the Requirements, but ...



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)

Examples of Scales (re-use of Requirements !)

Availability

% of <Time Period> a <System> is <Available> for its <Tasks>

Adaptability

Time needed to <Adapt> a <System> from <Initial State> to <Final State> using <Means>

Usability

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

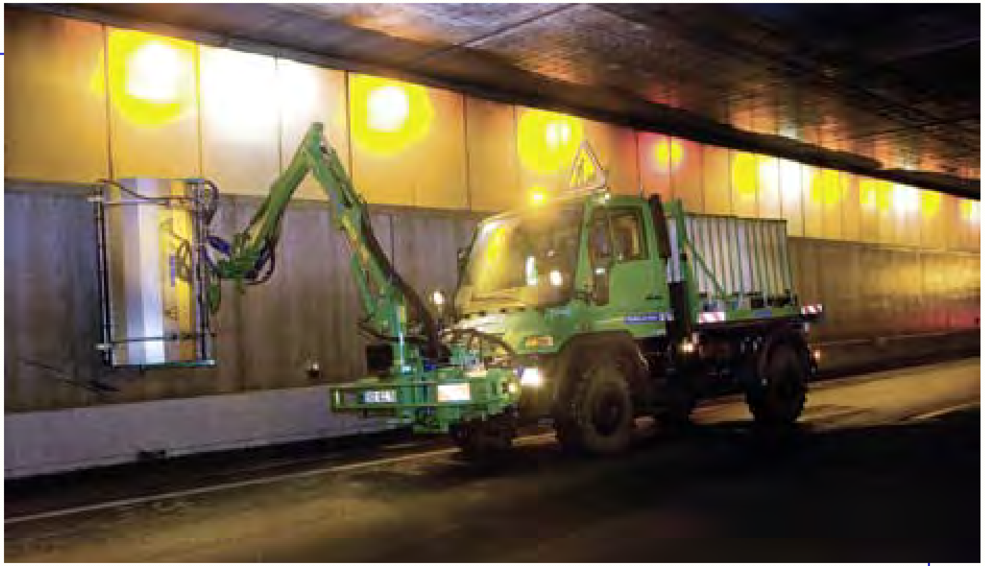
Reliability

Mean time for a <System> to experience <Failure Type> under <Conditions>

Integrity

Probability for a <System> to <Cope-with> <Attacks> under <Conditions>
Define "Cope-with" = {detect, prevent, capture}

Availability

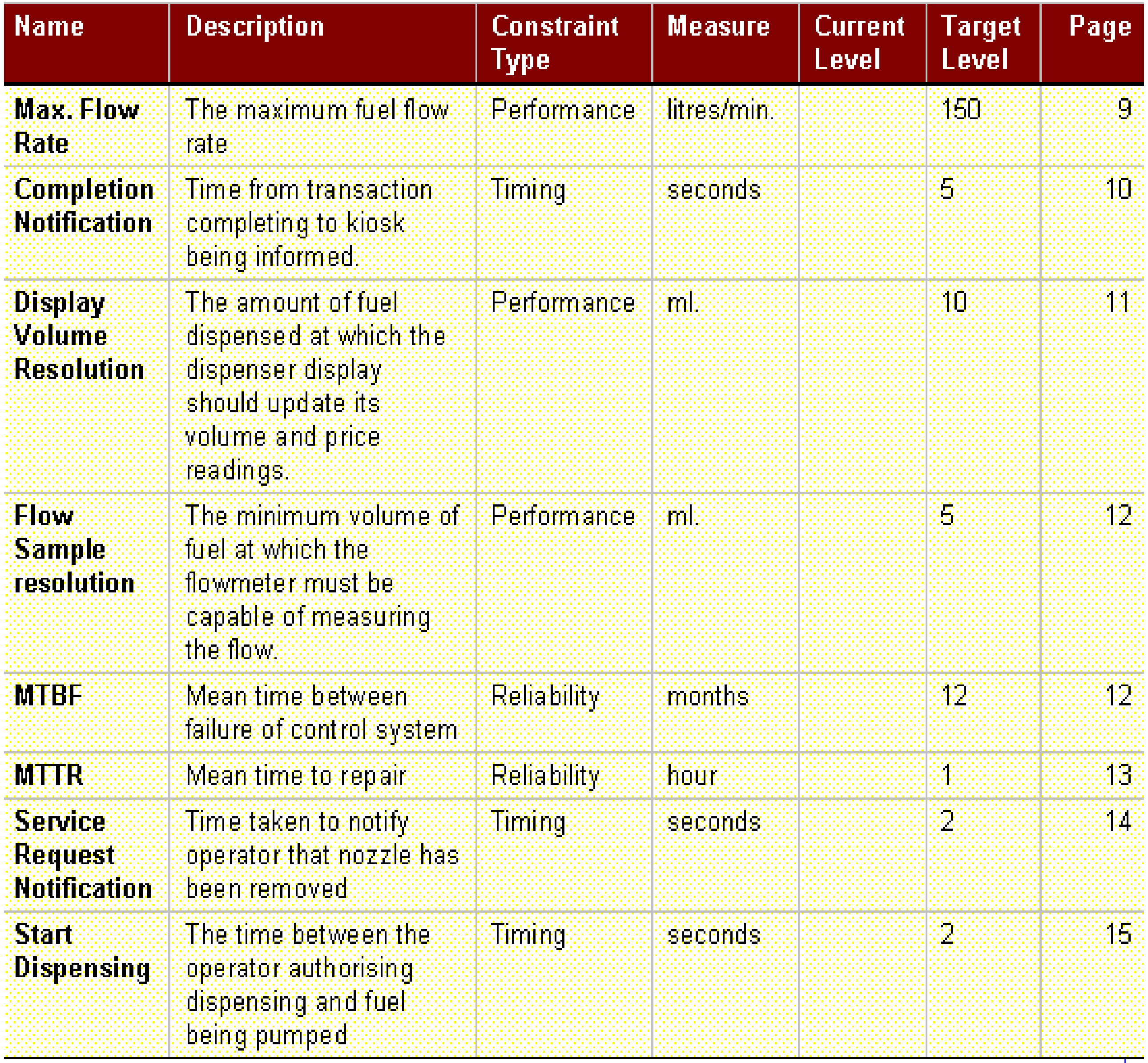


- Dependability.Availability
  - Readiness for service
  - Scale: % of <TimePeriod> a <System> is <Available> for its <Tasks>
- Probability that the system will be functioning correctly when it is needed
- Examples
  - (preventive) maintenance may decrease the availability
  - Snow on the road
  - Telephone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

Availability

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

Quantified Requirements



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

How about your requirements ?

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

Did anyone prepare ?

- The Goal of your current work or project
- The Definition of Success
- The most important stakeholder (Who is waiting for it?)
- The most important requirement for this stakeholder (What is he waiting for?)
- How much value improvement does this stakeholder expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What you and your team should and can have achieved in the coming 10 weeks
  (Will you succeed? If yes: great. If not: what could you do about it? - Failure is not an option!)
- What you think you should and can do *the coming week* to achieve what you're supposed to achieve
  (How do you make sure that by the end of the week all of this will be done)
- Any issues you expect with the above or otherwise with your work or project

Requirements exercise:

(groups of 2 or 3 people)

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

Try to use:

Definition:

- Ambition
- Scale
- Meter
- Stakeholders

Benchmarks:

- Past
- Current
- Record
- (Wish)

Requirements:

- Must/Fail/Tolerable
- Goal

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

Ambition	
Scale	
Meter	
Stakehldrs	

Past	
Current	
Record	
Wish	

Tolerable	
Goal	

How to specify results

How to select

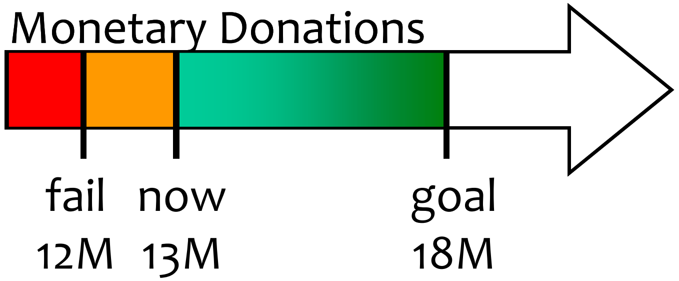
the right solution ?

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: ‘Measurable Value with Agile’
ACCU Overload Feb 2009, or
http://www.malotaux.nl/doc.php?id=10

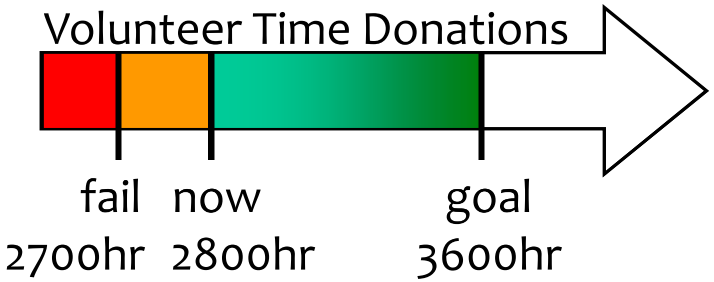
Objective: Monetary Donations



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

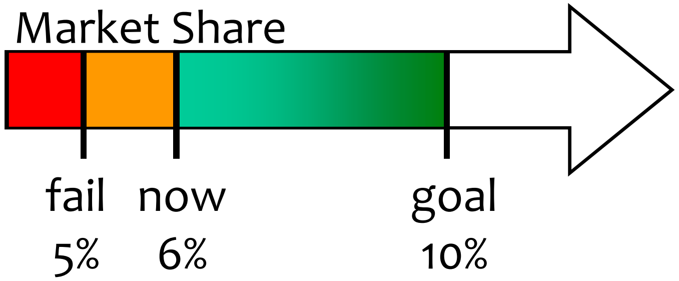
Objective: Volunteer Time (Natura) Donations



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%	
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	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	$\frac{\text{Benefits}}{\text{Cost}}$	

More

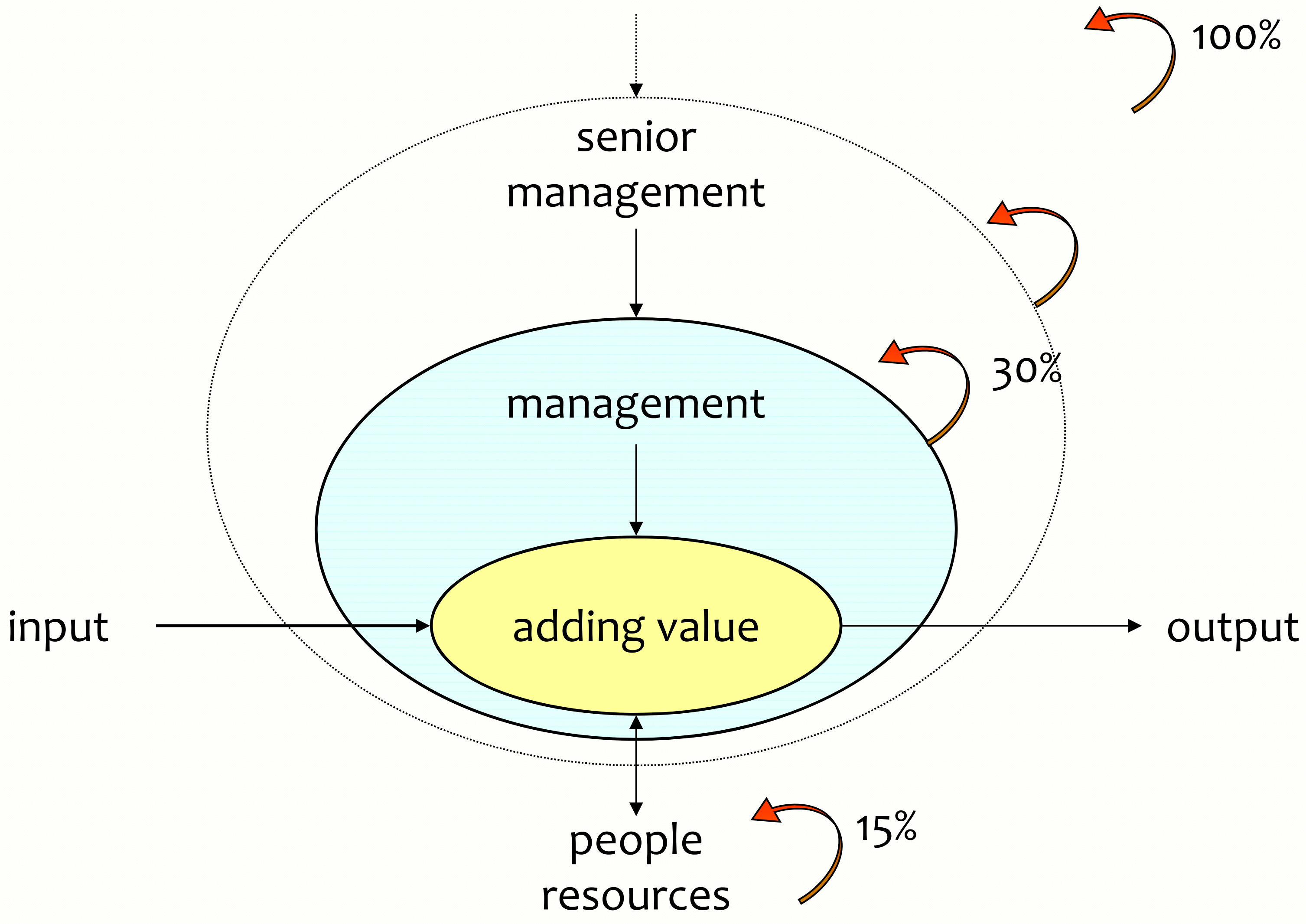
- Booklets – www.malotaux.nl/booklets
- Email – niels@malotaux.nl
- Some coaching of your team (and your management) on the spot

Exercise

- What will you be doing differently after this ?
- Requirements not only for the product,
  but now for *how you do your work*
- Is this also reflected in your weekly TaskList ?
  Otherwise it may not happen

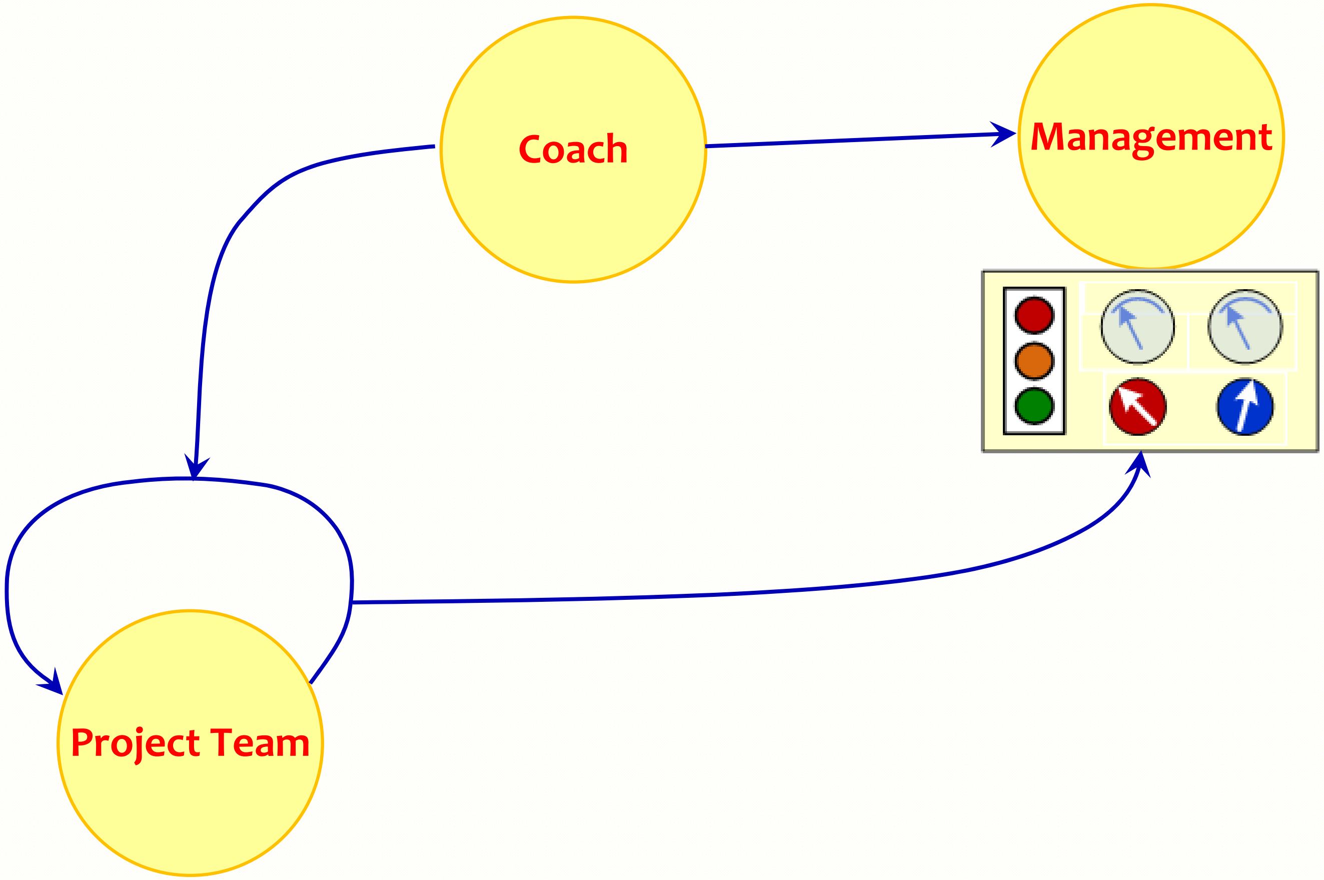
Management Issues

Simple model of Management



See www.malotaux.nl/managementmodel

Local Loop Principle



See www.malotaux.nl/localloop

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 it ?

- 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/booklets

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

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

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

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

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

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

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

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

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

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

www.malotaux.nl/inspections

Inspection pages

Planning for Quality Delivery

Producing even more business value in less time

www.malotaux.nl/conferences

Niels Malotaux

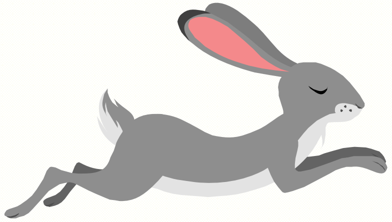
N R Malotaux
Consultancy

+31 655 753 604

niels@malotaux.nl

www.malotaux.nl

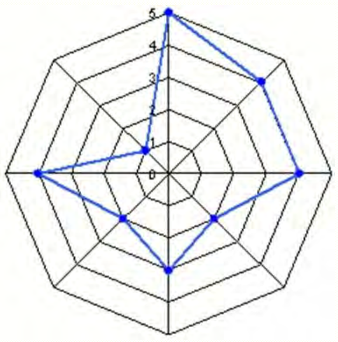
Agile or agile ?



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:

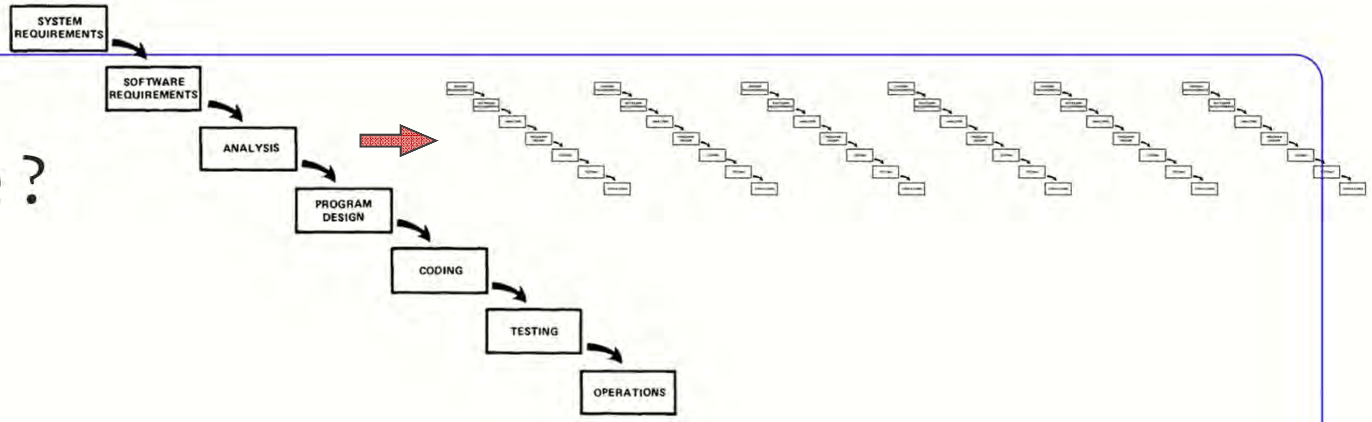
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

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

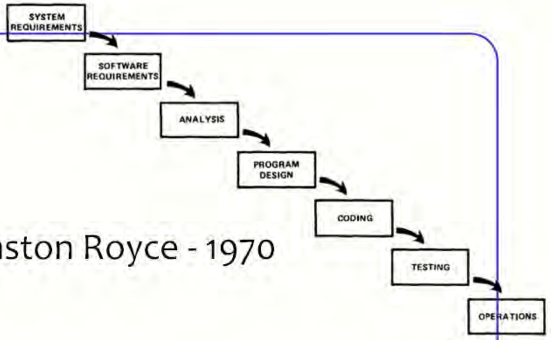
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
  Software is always part of a system
- We welcome changing requirements, even late in development
  We can handle them late, but early is better
  If requirements have to change, let's *provoke* requirements change as quickly as possible
- We deliver working software frequently;
  Working software is the primary measure of progress
  What we deliver simply works.
  If the working software doesn't do what it should, is that a measure of progress?
- Business people and developers must work together daily
  Do they ? Should 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 more importantly: prespectives

What is Agile ?

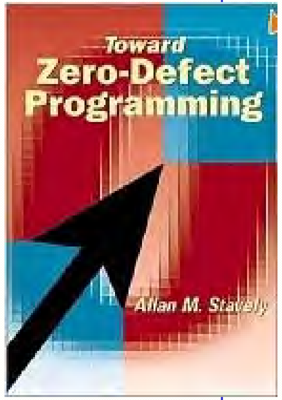


- A philosophy (Agile Manifesto)
- agile = ability to move quick, easily and adaptably
- Short iterations – not one Waterfall
- Delivering value (do we measure progress towards real value ?)
- Retrospectives (retrospectives on retrospectives: did it really work ?)
- Not a standard: You can make of it 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’ ?

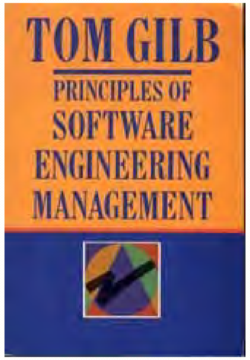
The past was already ahead

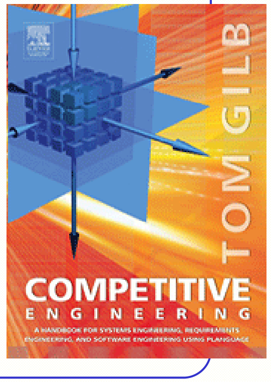


- Managing the development of large software systems - Winston Royce - 1970
  - Famous 'Waterfall document': figure 2 showed a 'waterfall'
  - Text and other figures showed that Waterfall doesn't work
  - Anyone promoting Waterfall doesn't know or didn't learn from history
- Cleanroom software engineering - 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




TOM GILB
COMPETITIVE
ENGINEERING
A HANDBOOK FOR SYSTEMS ENGINEERING, REQUIREMENTS ENGINEERING, AND SOFTWARE ENGINEERING USING PLANGUAGE

XP – eXtreme Programming

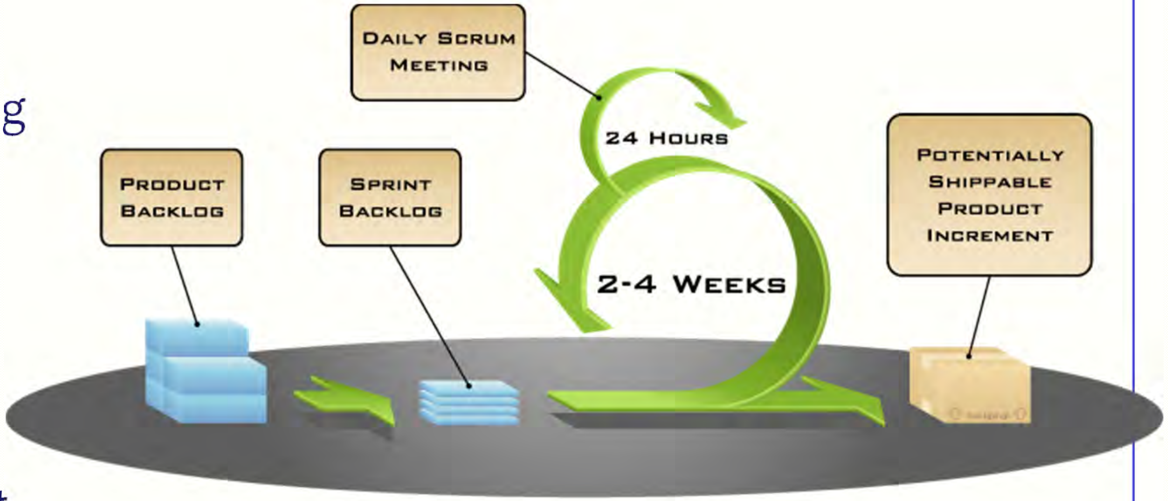
- Planning Game
- Metaphor
- Simple Design
- Testing (TDD)
- Refactoring
- Coding standards

- 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

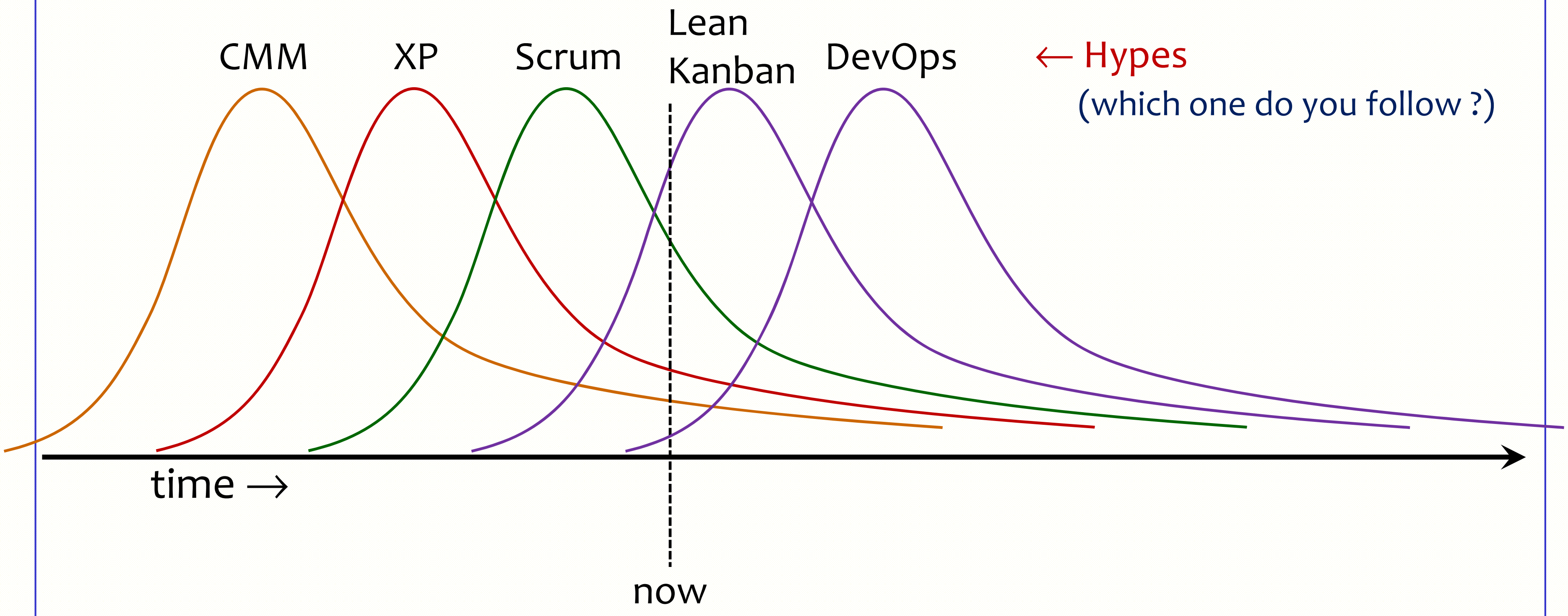
- Sprint
  - 1 – 4 weeks
  - Sprint Planning meeting
  - Sprint Review meeting
  - Sprint Retrospective
- Artefacts
  - Product backlog
  - Sprint backlog
  - Sprint burn down chart



- 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 - prespectives*
- Overall planning and prediction: when will what be done