Quality On Time

How to deliver the right results at the right time, no excuses needed

www.malotaux.eu/conferences
www.malotaux.eu/booklets

Niels Malotaux

+31-655 753 604 niels@malotaux.eu www.malotaux.eu

Niels Malotaux

- Independent Engineering and Team Coach
- Expert in helping teams and organizations to quickly become
  - More effective - doing the right things better
  - More efficient - doing the right things better in less time
  - More predictable - delivering as needed
- Getting projects back on track
- Embedded Systems architect (electronics/firmware)
- Project types
  electronics, firmware, software, space, road, rail, telecom, industrial control, parking system

Delivering
Quality On Time
the Right Result
at the Right Time

Who are you ?

Schedule, we’ll try to keep ☺

	24 October
9:00~10:30	1:30
break	0:15
10:45~12:00	1:15
lunch	1:00
13:00~14:30	1:30
break	0:15
14:45~16:30	1:45

During breaks:

- Relax
- Discuss
- Exercise some more

Is there a Quality On Time problem ?

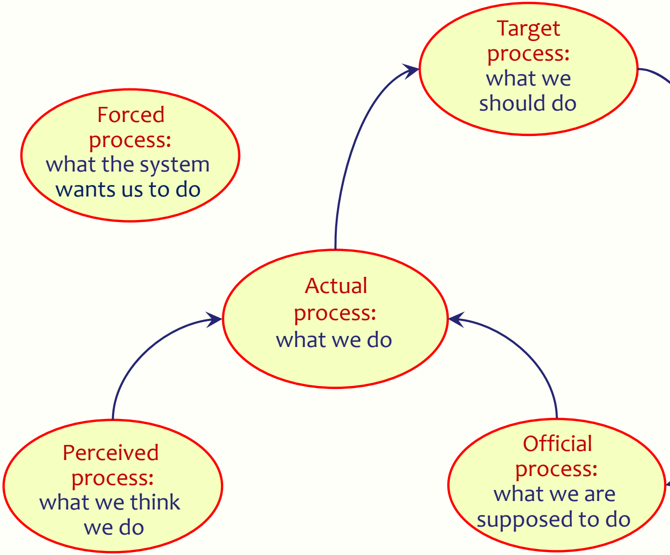
- What made you decide to attend ?
- Do you/your teams deliver the Right Results ?
- Do you/your teams deliver the Right Results at the Right Time ?
- What is the Right Result ?
- What is the Right Time ?

Exercise 1: Initial overview of your work

- The Goal of your current work or project
- The most important stakeholder(s) (Who is waiting for it?)
- The most important requirement(s) (What are they waiting for?)
- How much value improvement do the stakeholder(s) expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What should and can you/your team 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 should and can you/your team have achieved the coming week
- What do you think you should and can do to contribute to achieving that
  (Don't plan what you shouldn't or cannot do - By the end of the week everything will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work or project

Anything to be improved in your situation?

Five processes


Forced process: what the system wants us to do
Target process: what we should do
Actual process: what we do
Perceived process: what we think we do
Official process: what we are supposed to do

Goal of this workshop

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

Warning:

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

But you shouldn't need one either

Ultimate Goal of a What We Do

(for our salary)

Quality on Time

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

- Providing the customer with
  - what they need
  - at the time they need it
  - to be satisfied
  - to be more successful than they were 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

Quality

“We must deliver value !”

We don’t deliver value

We should create the *conditions* for the *users* to let the value *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

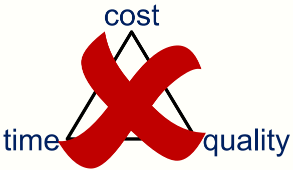
Sprint demo ??

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

- Is this what you do ?

Does quality cost more ?

- The ~~cost is not~~ benefit is in the quality
- The cost is in the non-quality



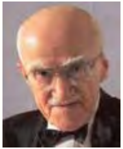
The *right* quality done right costs *less*

Quality guru’s

kaizen.com
KAIZEN™ INSTITUTE
Know your Customer
Creating customer value
Let it Flow
Targeting zero waste
MU DA
The Core of KAIZEN™
改善
Pure Improvement
Everyone, Everywhere, Every Day
Go to Gemba
Following the actual
Be Transparent
Speaking with real data
Empower People
Organizing your teams
Practice, Learn and Improve
Be Determined

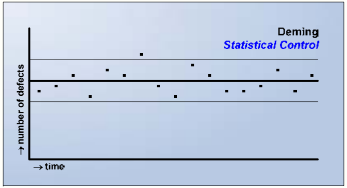
- Shewhart - Economic Control of Quality 1931
- Deming - Japan 1950, Out of the crisis 1986
- Juran - Japan 1954, Quality handbook 1951
- Crosby - Quality is Free 1979, Zero Defects 1961
- Imai - Kaizen 1986, Gemba Kaizen 1997

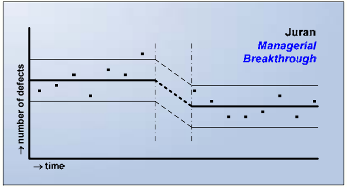
Deming

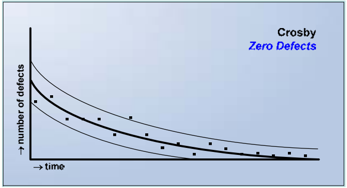


Juran

Crosby

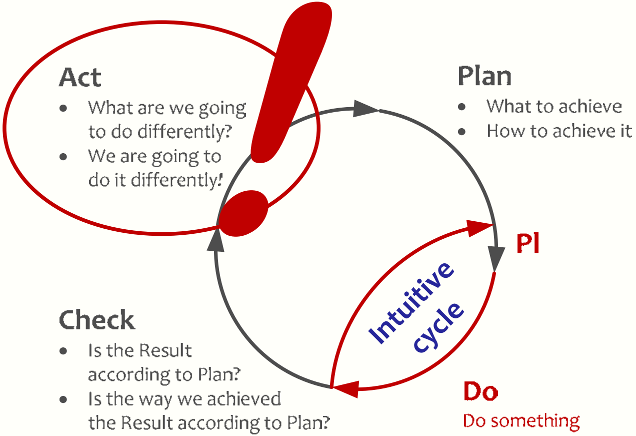






The secret weapon: PDCA

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



Act

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

Plan

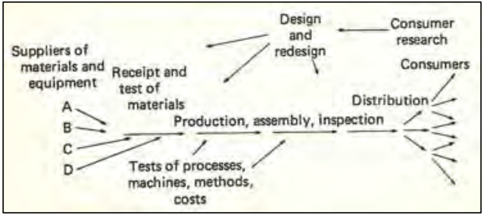
- What to achieve
- How to achieve it

Check

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

Do

Do something


Design and redesign
Consumer research
Suppliers of materials and equipment
Receipt and test of materials
Consumers
Distribution
A
B
C
D
Production, assembly, inspection
Tests of processes, machines, methods, costs

Deming: Out of the Crisis

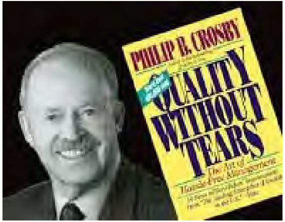
Deming talking to Japanese Top Management in 1950

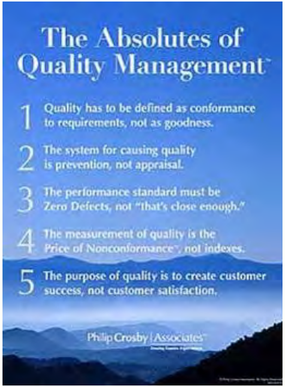
Crosby (1926-2001) - Absolutes of Quality

- 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.

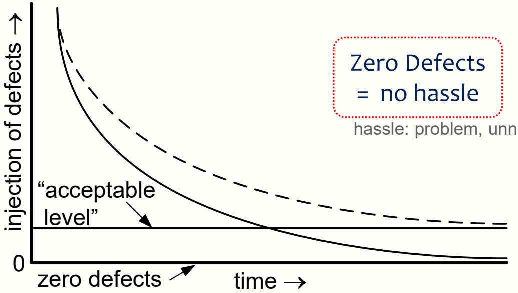
5 The purpose of quality is to create customer success, not customer satisfaction.

Philip Crosby | Associates

Is Zero Defects possible ?

- We are not 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

hassle: problem, unnecessary difficulty

- 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

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 (no pain - no gain)
- 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)
- Are we testers or QA ?

Prevention: Root Cause Analysis

cause

solution

root cause

root solution

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

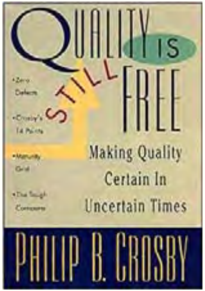
- Cause:
  The error that caused the defect
- Root Cause:
  What caused us to make the error that caused the defect

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

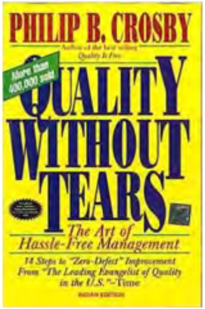
Philip Crosby

[Quality is Still Free]

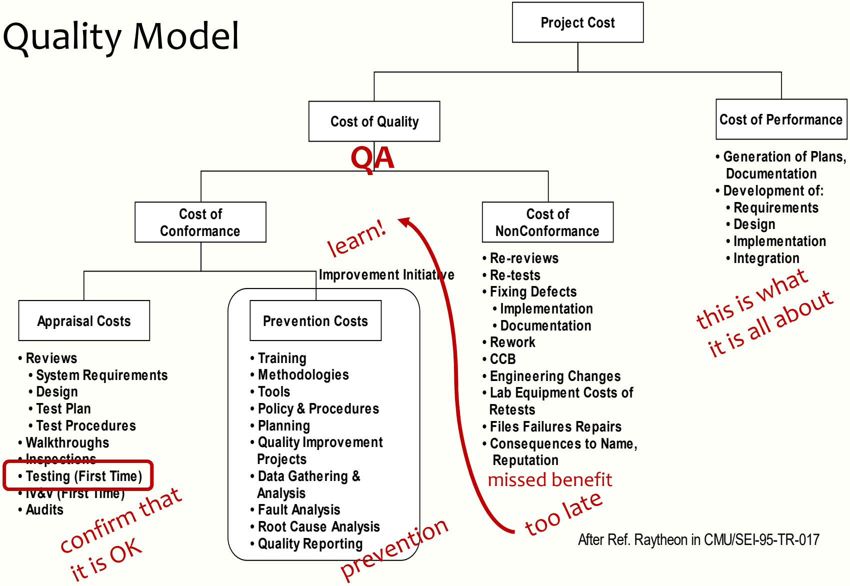
QUALITY IS FREE
The Art of Making Quality Certain
How to manage quality – so that it becomes a source of profit for your business
PHILIP B. CROSBY
author of:
"The Art of Getting Your Own Sweet Way"



- Conventional wisdom says that error is inevitable
- As long as the performance standard requires it,
  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 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


PHILIP B. CROSBY
Author of the best selling Quality Is Free
More than 400,000 sold
QUALITY WITHOUT TEARS
The Art of Hassle-Free Management
14 Steps to "Zero-Defect" Improvement From "The Leading Evangelist of Quality in the U.S."—Time

Cost of Quality Model



Project Cost

Cost of Quality

QA

Cost of Performance

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

this is what it is all about

Cost of Conformance

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

learn!

Cost of NonConformance

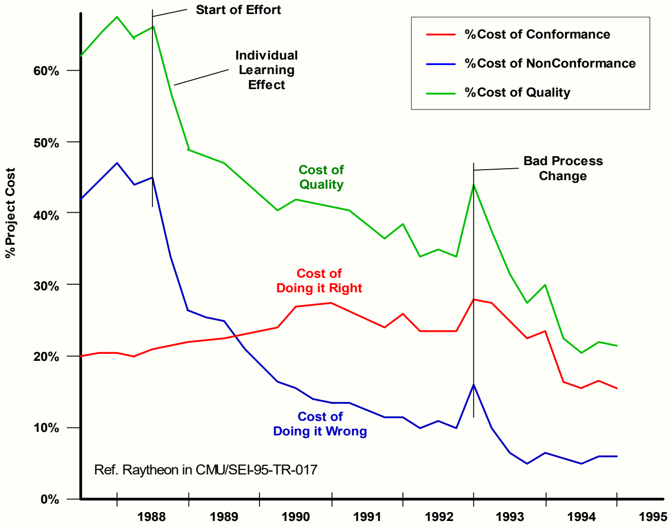
- Re-reviews
- Re-tests
- Fixing Defects
  - Implementation
  - Documentation
- Rework
- CCB
- Engineering Changes
- Lab Equipment Costs of Retests
- Files Failures Repairs
- Consequences to Name, Reputation

missed benefit

too late

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

Cost of Quality

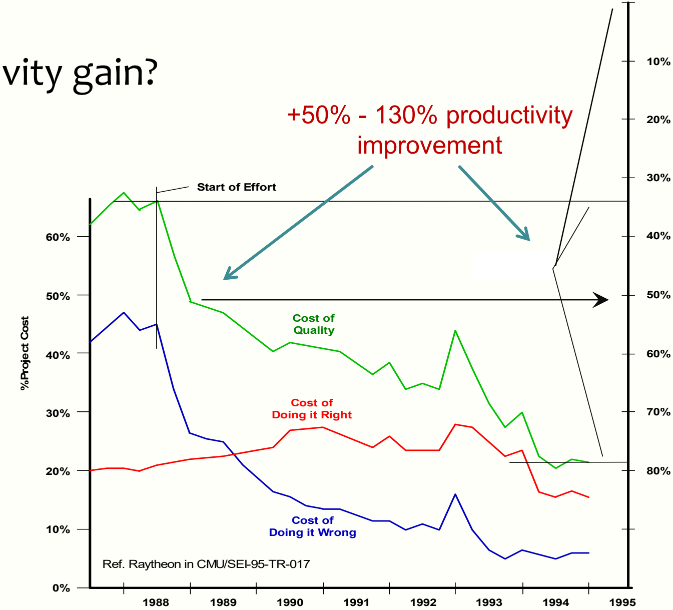


Ref. Raytheon in CMU/SEI-95-TR-017

Raytheon working on continuous improvement

https://kilthub.cmu.edu/ndownloader/files/12069764

How much productivity gain?



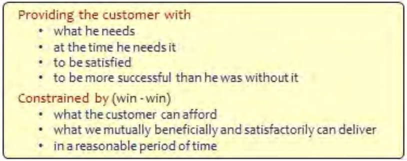
Ref. Raytheon in CMU/SEI-95-TR-017

Did you ever review ?

- Code ?
- Design ?
- Requirement ?

We're QA: What has this to do with us ?

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



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

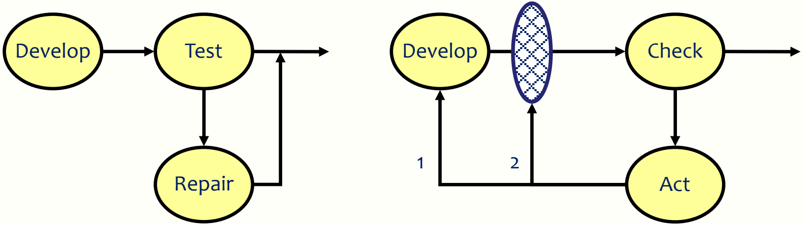
Constrained by (win - win)

- what the customer can afford
- what we mutually beneficially and satisfactorily can deliver
- in a reasonable period of time

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 should find it's OK



What we often see

What we should expect

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

On Time

On Time

- Yesterday ?
- Deadline ?
- Managers dream ?
- Time to market ?
- Time to profit ?

Compromise between what is *needed*
and what is *possible*

Is it difficult to be on time ?

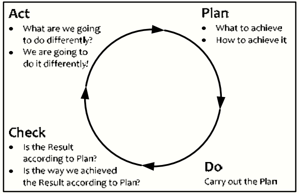
- Did anyone miss a plane ?
- What did you feel ?
- Why did it happen ?
- Did it happen again ?

Will your current delivery be on time ?

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

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

unless we do things *differently* and *better*



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?

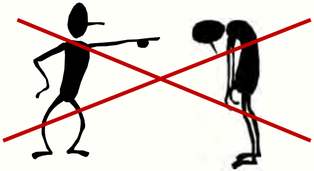
Do

Carry out the Plan

If we don’t learn from history, we are doomed to repeat it

We don’t have to be late
We can do much better

Isn’t that the Responsibility of some Manager ?

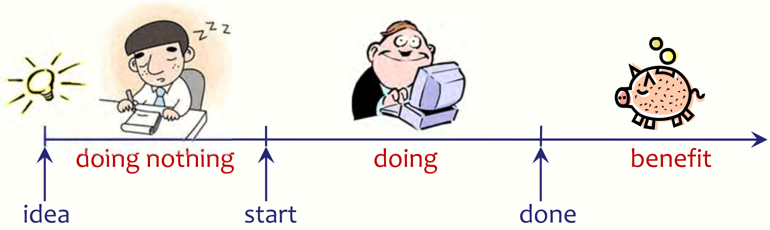


- A Manager may be *responsible* for delivering the right result at the right time
- The Worker’s work and decisions *determine* the *result*, and the *time* it is delivered
- This makes all of us *as responsible* as Management

The Importance of Time

Business Case

(why are we doing it)



doing nothing

doing

benefit

idea

start

done

Return on Investment (ROI)

+ Benefit of doing - huge (otherwise we should do something else)
– Cost of doing - 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

Exercise 2: Cost of one day of (unnecessary) delay

- What is the cost of your team per day ?
- What do you cost per day ?
  Note: that’s not what you get !
- If you don’t know the benefit, assume 10 times the cost
- Calculate the cost of one day of delay

Exercise 2: Cost of one day of delay

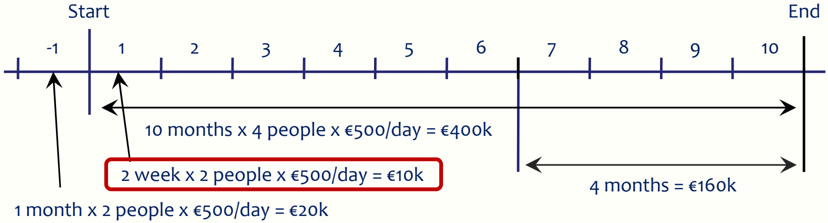
- What is the cost of your team per day ?
- What do you cost per day ?
  Note: that's not what you get !
- If you don't know the benefit, assume 10 times the cost
- Calculate the cost of one day of delay

- If €400 per day
- 5 people x €400 = €2000
- Cost of delay 10 x €2000 = €20,000

if 5x:

- Cost of delay 5 x €2000 = €10,000

Time vs Budget ? - VOIP introduction project



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

  PO → "Don't waste another 10k. Start working !"
- 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 !

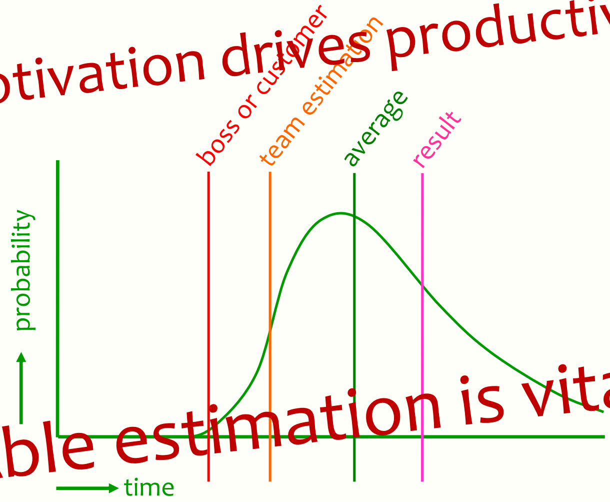
Exercise 3: To-do lists

- Are you using to-do lists ?
  - List the things you are going to do the coming week
  - Did you check how much time you have available the coming week ?
  - Does what you have to do fit in the available time ?
  - How would you know ?
  - 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

Estimation

Lead time



Motivation drives productivity

Able estimation is vital

Exercise 4: Estimation

Are you an optimistic or a realistic estimator?

Let's find out !

Project:

Multiplying two numbers of 4 figures

Example

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

How much time do you need to complete this Project ?

Write down !

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

Assumptions, assumptions ...

Better assume that our assumptions can be wrong

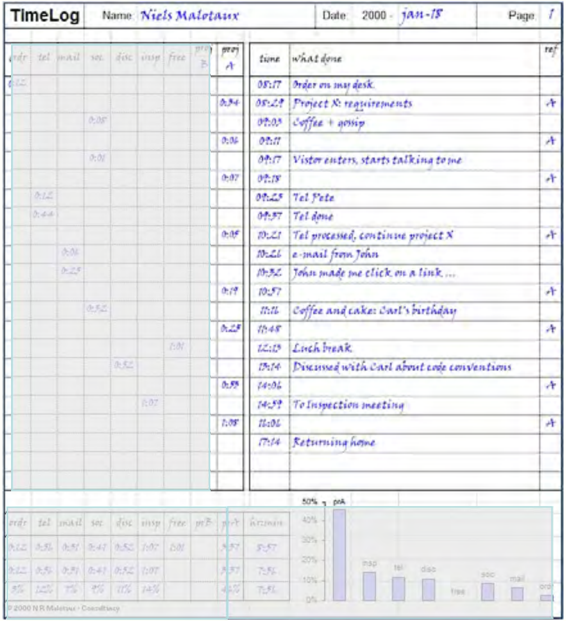
Check !

Elements in the exercise

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

TimeLog

To get an idea of
where your time is going



TimeLog Name: Niels Malotaux Date: 2000 - jan-18 Page 1

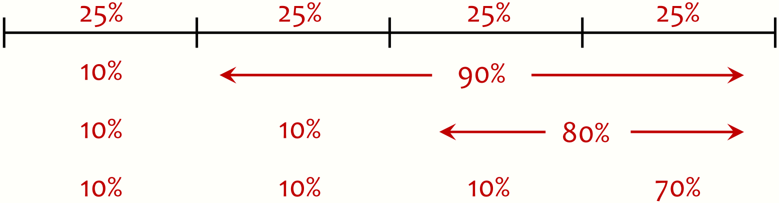
ordr	tel	mail	soc	disc	insp	free	proj B	proj A	time	what done	ref
0:12									08:17	Order on my desk	
								0:34	08:29	Project X: requirements	A
			0:08						09:03	Coffee + gossip	
								0:06	09:11		A
			0:01						09:17	Vistor enters, starts talking to me	
								0:07	09:18		A
	0:12								09:25	Tel Pete	
	0:44								09:37	Tel done	
								0:05	10:21	Tel processed, continue project X	A
		0:06							10:26	e-mail from John	
		0:25							10:32	John made me click on a link ...	
								0:19	10:57		A
			0:32						11:16	Coffee and cake: Carl's birthday	
								0:25	11:48		A
						1:01			12:13	Lunch break	
				0:52					13:14	Discussed with Carl about code conventions	
								0:53	14:06		A
					1:07				14:59	To Inspection meeting	
								1:08	16:06		A
									17:14	Returning home	

ordr	tel	mail	soc	disc	insp	free	prB	prA	hrs:min
0:12	0:56	0:31	0:41	0:52	1:07	1:01		3:37	8:57
0:12	0:56	0:31	0:41	0:52	1:07			3:37	7:56
3%	12%	7%	9%	11%	14%			46%	7:56

© 2000 N R Malotaux - Consultancy

www.malotaux.eu/?id=downloads#timelog

4 week project



25% 25% 25% 25%

10% ← 90% →

10% 10% ← 80% →

10% 10% 10% 70%

How long do such projects usually take ?

Did anyone tell or yell to go faster ?

- 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

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 work ?

Causes of Spending Time Unnecessarily

- Some typical causes
  - 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 challenge

Failure is not an option

- Getting and keeping the deliveries 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 ?

How to be On Time ?

Exercise 5:

- Are your deliveries usually on time ?

- Quality compromised to be on time ?

- Are your deliveries usually on time ?
- If yes, is the quality compromised for being on time ?
  - That’s not ‘on time’ !
  - What we deliver should simply work

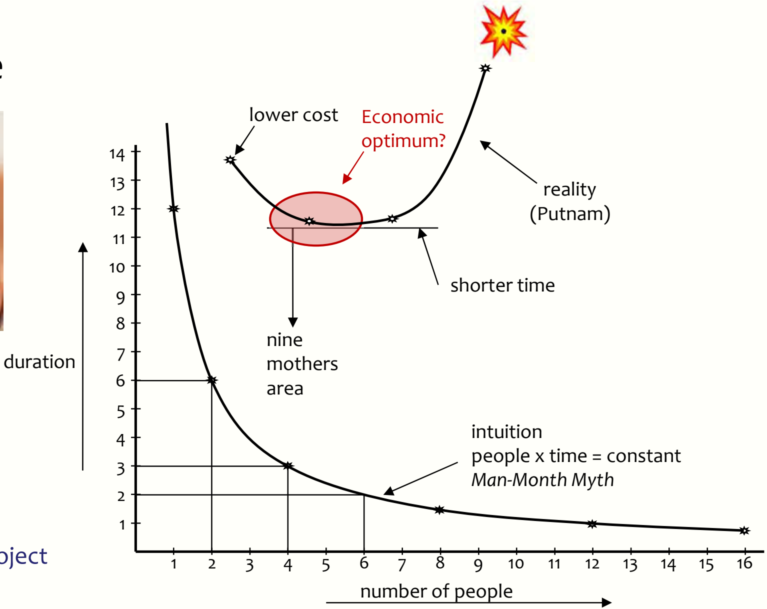
- How can we save time, without compromising quality ?

- 7 options

Deceptive options

1. Hoping for the best (fatalistic)
2. Going for it (macho)
3. Working Overtime (fooling ourselves and our boss)
4. 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

5. Adding people

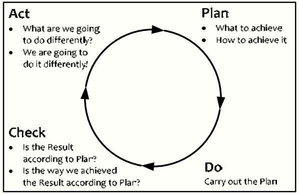


Brooks' Law (1975)

Adding people to a late project *makes it later*

6. Saving time

Continuous elimination of waste (www.malotaux.eu/?id=essenceoflean)

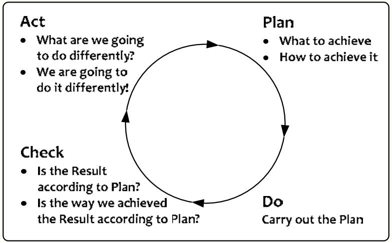


Act
What are we going to do differently?
We are going to do it differently!
Plan
What to achieve
How to achieve it
Check
Is the Result according to Plan?
Is the way we achieved the Result according to Plan?
Do
Carry out the Plan

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 (www.malotaux.eu/?id=evo)
  - *Not* doing what later proves to be superfluous
- Efficiency in *how* we do it - doing things differently
  - The product (www.malotaux.eu/?id=designlog)
    - Using proper and most efficient solution, instead of the solution we always used
  - The project (www.malotaux.eu/?id=projectmanagement)
    - Spending less time, instead of immediately doing it the way we always did
  - Continuous improvement and prevention processes (www.malotaux.eu/?id=PDCA)
    - Constantly learning doing things better and overcoming bad tendencies
- Efficiency in *when* we do it - right time, right order (www.malotaux.eu/?id=timeline)
- TimeBoxing - much more efficient than FeatureBoxing (www.malotaux.eu/?id=timeboxing)

Evolutionary (Evo) Principles



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

It’s not a method or a ‘framework’

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

Perhaps some alternatives …

No cure - no pay

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

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

Value

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

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

- How do we know we delivered value ?
- How much ?

Perceived value

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

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

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

Murphy’s Law

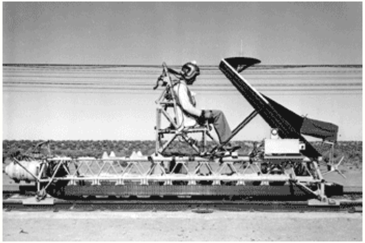
- Whatever can go wrong, will go wrong
- Should we accept fate ??

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 they cannot happen



Preflection, foresight, prevention
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

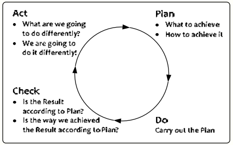
Only if we change our way of working, the result may be different

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

Prespectives

Only with prevention we can save precious time

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



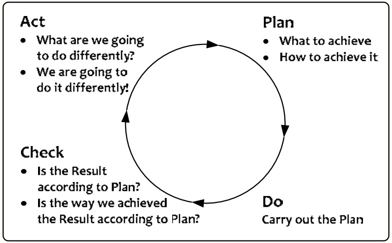
Act
- What are we going to do differently?
- We are going to do it differently!

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

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*

doing the right things

then we can

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

and we can

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

doing the right things right

Known for decades

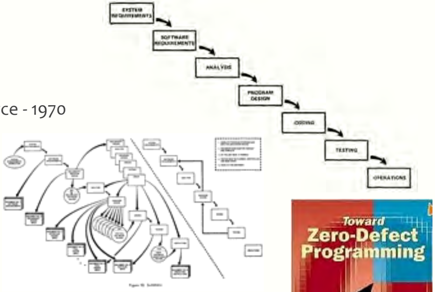
- Benjamin Franklin (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve
- Henry Ford (1863-1947)
  - My Life and Work (1922)
    - We have eliminated a great number of wastes
  - Today and Tomorrow (1926)
    - Learning from waste, keeping things clean and safe, better treated people produce more
- Toyoda's (Sakichi, Kiichiro, Eiji) (1867-1930, 1894-1952, 1913-2013)
  - 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)

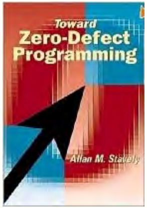
Do we still have to talk about this ?

Eliminating Waste
Not doing what doesn't yield value

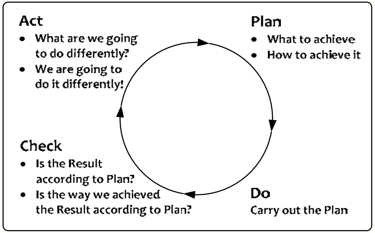
There is nothing new in software too

- 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 costs less
- 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





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?

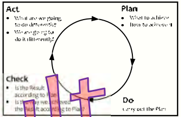
Do
Carry out the Plan

- 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 find out
- 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


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

Why

What
How much
Are we done

How

Check and learn
as early as possible

Zero Defects Attitude

Right Result

Quality On Time

Evolutionary Planning - Niels

- 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

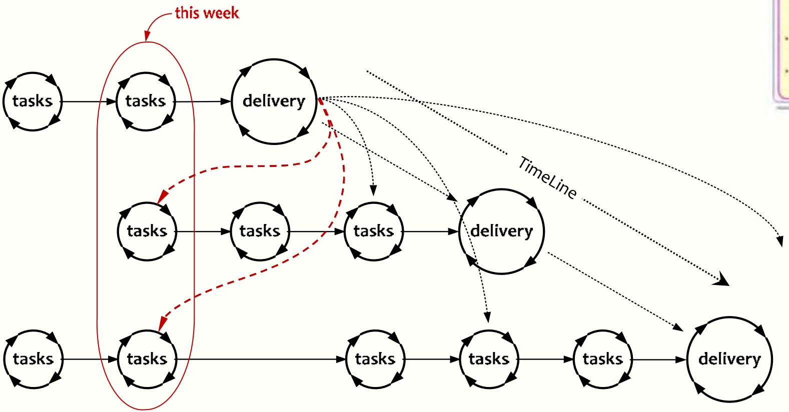
What will happen, and
what will we do about it ?

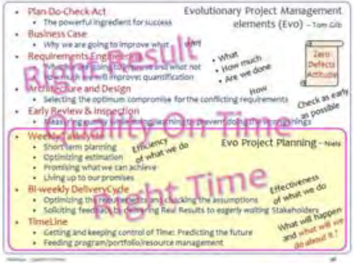
www.malotaux.eu/?id=processes

Evolutionary Planning

Producing even more
in less time

Tasks feed Deliveries





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 and feedback learning to prevent doing the wrong things

Evo Project Planning – Niels

- 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

Making best use of limited available time

- If the work is done, the time is already spent
- If we still have to do the work, we can decide
  - What is really important
  - What is less important
  - What we must do
  - What we can do
  - What we are going to do
  - What we are *not* going to do
- Therefore we plan first, in stead of finding out later
- We cannot work in the past

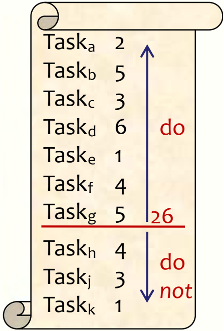
Evo Planning: Weekly TaskCycle

- 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

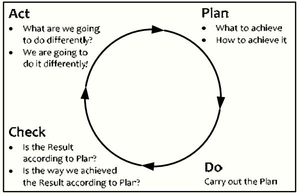


Task$_a$	2
Task$_b$	5
Task$_c$	3
Task$_d$	6
Task$_e$	1
Task$_f$	4
Task$_g$	5
Task$_h$	4
Task$_j$	3
Task$_k$	1

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

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

Every week: reflecting and preflecting



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

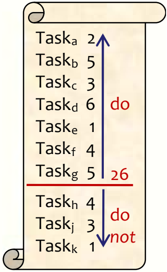
Plan
- What to achieve
- How to achieve it

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

Do
Carry out the Plan

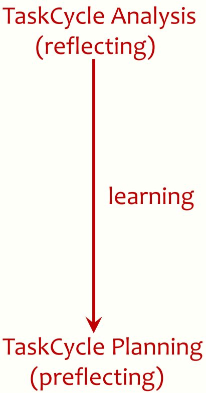
- Was all planned work really done?
- If a Task was not completed, we learn:
  - Time spent but the work not done? → effort estimation problem
    - What did I think then, what do I know now, learn (Check and Act)
  - Time not spent? → time management problem
    - Too much distraction
    - Too much time spent on other (poorly-estimated) Tasks
    - Too much time spent on other things
- Close unfinished Tasks after having dealt with the consequences
  - Feed the disappointment of the "failure" into your intuition mechanism
  - Define remaining Tasks, and put on the Candidate Task List
  - Declare the Task finished after having taken the consequences
- Continue with planning the Tasks for the next week

Immediate consumption of metrics



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

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 team x		
		~~kkkkkkkkk~~				for team x		
		mmmmmm	5			for team x		
		~~nnnnnnnn~~				for team x		
		~~pppppppp~~				for team y		
		qqqqqqqq	12			for team y		
		rrrrrrrrrrrr	6			for team y		
		~~ssssssssss~~				for team y		
		~~ttttttttttt~~				for team y		
			26					



TaskCycle Analysis (reflecting)

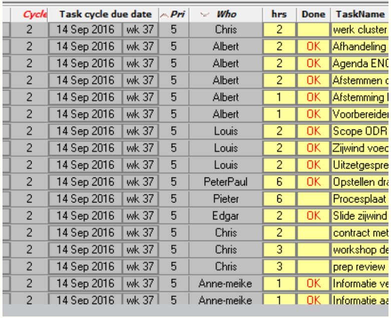
learning

TaskCycle Planning (preflecting)

Weekly 3-Step Procedure

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

Modulation costs less than Generation

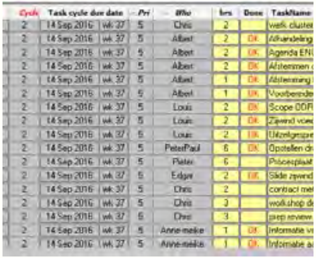


Cycle	Task cycle due date		Pri	Who	hrs	Done	TaskName
2	14 Sep 2016	wk 37	5	Chris	2		werk cluster
2	14 Sep 2016	wk 37	5	Albert	2	OK	Afhandeling
2	14 Sep 2016	wk 37	5	Albert	2	OK	Agenda EN
2	14 Sep 2016	wk 37	5	Albert	2	OK	Afstemmen
2	14 Sep 2016	wk 37	5	Albert	1	OK	Afstemming
2	14 Sep 2016	wk 37	5	Albert	1	OK	Voorbereide
2	14 Sep 2016	wk 37	5	Louis	2	OK	Scope ODR
2	14 Sep 2016	wk 37	5	Louis	2	OK	Zijwind voe
2	14 Sep 2016	wk 37	5	Louis	2	OK	Uitzetgespre
2	14 Sep 2016	wk 37	5	PeterPaul	6	OK	Opstellen dr
2	14 Sep 2016	wk 37	5	Pieter	6		Procesplaat
2	14 Sep 2016	wk 37	5	Edgar	2	OK	Slide zijwind
2	14 Sep 2016	wk 37	5	Chris	2		contract met
2	14 Sep 2016	wk 37	5	Chris	3		workshop de
2	14 Sep 2016	wk 37	5	Chris	3		prep review
2	14 Sep 2016	wk 37	5	Anne-meike	1	OK	Informatie ve
2	14 Sep 2016	wk 37	5	Anne-meike	1	OK	Informatie aa

Why is this important ?

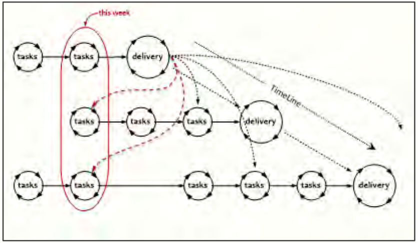
- Individual preparation
  - Conclude current tasks
  - What to do next
  - How much time available
  - Estimates
- Modulation with peer / coach
  - Status
  - Priority check
  - Feasibility
  - Commitment and decision
- Synchronization with group (team meeting)
  - Formal confirmation
  - Concurrency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

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



Cycle	Task cycle due date		Pri	Who	hrs	Done	TaskName
2	14 Sep 2016	wk 37	5	Chris	2		werk cluster
2	14 Sep 2016	wk 37	5	Albert	2	OK	Afhandeling
2	14 Sep 2016	wk 37	5	Albert	2	OK	Agenda ENI
2	14 Sep 2016	wk 37	5	Albert	2	OK	Afstemmen
2	14 Sep 2016	wk 37	5	Albert	1	OK	Afstemming
2	14 Sep 2016	wk 37	5	Albert	1	OK	Voorbereide
2	14 Sep 2016	wk 37	5	Louis	2	OK	Scope OOR
2	14 Sep 2016	wk 37	5	Louis	2	OK	Zijwind voe
2	14 Sep 2016	wk 37	5	Louis	2	OK	Uitzetgespre
2	14 Sep 2016	wk 37	5	PeterPaul	6	OK	Opstellen dr
2	14 Sep 2016	wk 37	5	Pieter	6		Procesplaat
2	14 Sep 2016	wk 37	5	Edgar	2	OK	Slide zijwind
2	14 Sep 2016	wk 37	5	Chris	2		contract met
2	14 Sep 2016	wk 37	5	Chris	3		workshop de
2	14 Sep 2016	wk 37	5	Chris	3		prep review
2	14 Sep 2016	wk 37	5	Anne-meike	1	OK	Informatie vi
2	14 Sep 2016	wk 37	5	Anne-meike	1	OK	Informatie a

DeliveryCycle



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

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

- Not more than 2 weeks

Task Cycle ↔ Delivery Cycle

Doing	Delivering

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

Optimizing

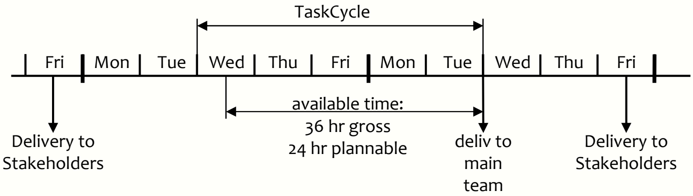
Estimation, planning, tracking	Requirements, assumptions

Selecting

Highest priority tasks	Most important values
≤ 1 week	≤ 2 weeks

Always done, 100% done

Designing a Delivery



TaskCycle

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

Delivery to Stakeholders

available time:
36 hr gross
24 hr plannable

deliv to main team

Delivery to Stakeholders

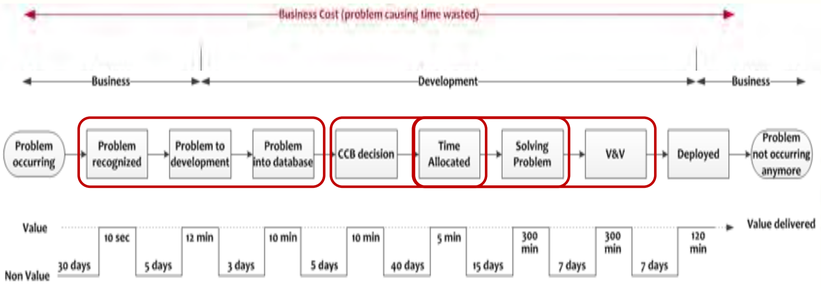
Serge (ProjLead)	
MbWA	3
Planning nxt wk	3
Work for deliv	4
-	6
-	2
-	1
-	5
Total	24

Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	40 34

Gregory (later)	
Draft design	0
Finish design	0
…	

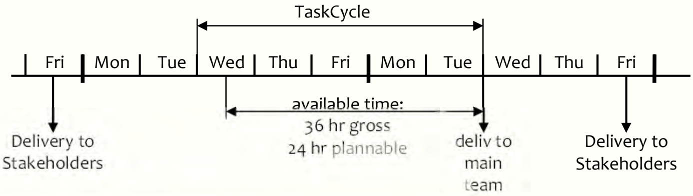
Jerome	
XMLa	3
XMLb	3
…	

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

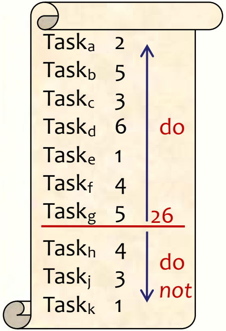
Gregory	
Draft design	6
Finish design	6
Work for deliv	3
-	1
-	2
-	2
-	3
-	5
-	6
XMLa	4
XMLb	4
Total	42

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

Jerome	
XMLa	3
XMLb	3
...	

Optimizing Value Stream by Prevention

Exercise 6: TaskCycle planning



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

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

Business Case

Business Case

- *Why* are we doing it ?
- Why to *improve* ?
- Drives the decision making processes
- To continually align the progress to the dynamic business objectives
- Can change during development
- Stakeholders
- Total LifeCycle - cradle to cradle

Higher Productivity

- All functionality we produce *does already exist*
- The real reason for our deliveries is improving on what people already do

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

How many Business Cases ?

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

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

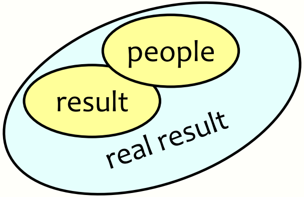
- Actually, every Stakeholder has their own Business Case

Stakeholders & Requirements

Initial overview of your work

- The Goal of your current work or project
- The most important stakeholder(s) (Who is waiting for it?)
- The most important requirement(s) (What are they waiting for?)
- How much value improvement do the stakeholder(s) expect (3 or 7?)
- Any deadlines? (No deadlines: it will take longer)
- What should and can you/your team 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 should and can you/your team have achieved the coming week
- What do you think you should and can do to contribute to achieving that
  (Don't plan what you shouldn't or cannot do - By the end of the week everything will be done)
- What value you will have delivered by the end of the week and how to prove it
- Any issues you expect with the above or otherwise with your work or project

Stakeholders are (not only) people

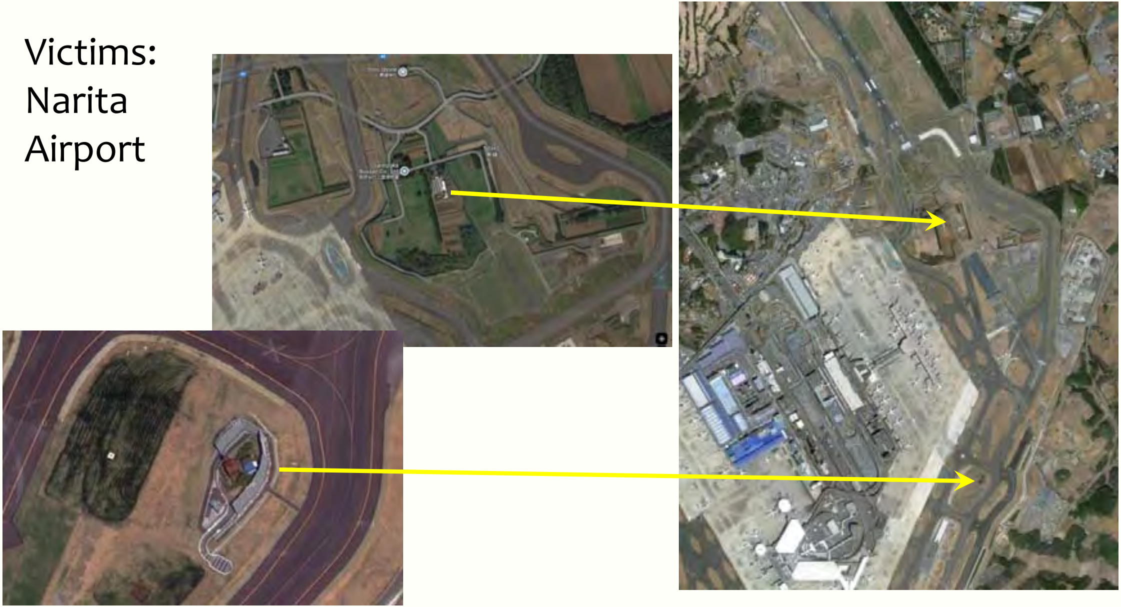


- Every project has some 30±20 Stakeholders
- Stakeholders have a stake in what we do
- The concerns of Stakeholders are often contradictory
  - Apart from the Customer *they don't pay*
  - Then they *have no reason to compromise !*
- Some Stakeholders are *victims* of what we do

  They have no reason for the project to succeed, on the contrary
- Risks, happening in almost every project
- No excuse to fail !

Victims can be a big Risk

Victims: Narita Airport



What are the Requirements for us ?

- Requirements are what the Stakeholders require

but for us ...

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

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

No Stakeholder?

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

The Goal of our work

Quality on Time

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

- Providing the customer with
  - what they need
  - at the time they need it
  - to be satisfied
  - to be more successful than they were 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 wish specification

Nice Input

to be taken seriously

Wish Specification

exercise 7

- 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 ?)

Case

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

Requirements have Rules

Some examples:

- Rule 1: Unambiguous (to the intended readership)
- Rule 2: Clear to test
- Rule 3: All quality requirements must be expressed quantitatively
- Rule 4: No design (solutions) in the requirements

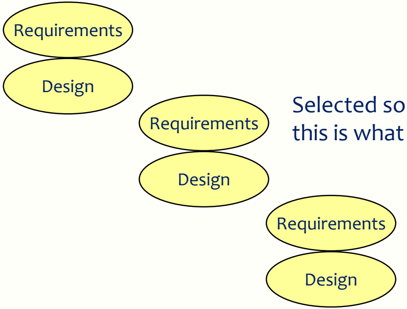
Typical requirements found:

- 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

No Design in the requirements, but ...

Needs: what do we need

Options: how can we do it



Requirements

Design

Requirements

Design

Requirements

Design

Selected solution:
this is what we chose as a solution

Requirement: What the acquirer cares about: ‘how good it should be’

Design: Set of decisions made by development: ‘how to be good’

Design provides the Requirements for the next level

Qualities

https://en.wikipedia.org/wiki/List_of_system_quality_attributes

- accessibility
- accountability
- accuracy
- adaptability
- administrability
- affordability
- agility
- auditability
- autonomy
- availability
- compatibility
- composability
- confidentiality
- configurability
- correctness
- credibility
- customizability
- debuggability
- degradability
- determinability
- demonstrability
- dependability
- deployability
- discoverability
- distributability
- durability
- effectiveness
- efficiency
- evolvability
- extensibility
- failure transparency
- fault-tolerance
- fidelity
- flexibility
- inspectability
- installability
- integrity
- interchangeability
- interoperability
- learnability
- localizability
- maintainability
- manageability
- mobility
- modifiability
- modularity
- observability
- operability
- orthogonality
- portability
- precision
- predictability
- process capabilities
- producibility
- provability
- recoverability
- redundancy
- relevance
- reliability
- repeatability
- reproducibility
- resilience
- responsiveness
- reusability
- robustness
- safety
- scalability
- seamlessness
- self-sustainability
- serviceability
- securability
- simplicity
- stability
- standards compliance
- survivability
- sustainability
- tailorability
- testability
- timeliness
- traceability
- transparency
- ubiquity
- understandability
- upgradability
- usability
- vulnerability

Requirements with Planguage

ref Tom Gilb

SMART

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, 2018], 8 min [average, 2018], 83 min [max, 20184]

Current: < 4 min [competitor y, Jan 2018] ← <who said this?>, <Survey April 2018>

Attainable

Record: 57 sec [competitor x, Jan 2018]

Wish: < 2 min [2022Q3, new system available] ← CEO, 19 Jan 2021, <document ...>

Requirements:

Time

Traceable

Tolerable: < 10 min [99%, Q4] ← SLA

Realizable

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

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

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

Is this a Requirement ?

or ‘nice input’, to be taken seriously ?

“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.”

Design

Need

How “immediately?”

Need

How “off” to warrant detection?

Ref http://rsbatechnology.co.uk

Using 5 Whys

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

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 manage customer trading (safely: no unexpected loss)
4. To ensure that senior management will, based on current business performance, in the future agree to expand e-trading business, to other customer segments and business areas
5. To meet business medium / long-term financial targets

Using 5 Whys for testing

What is the most important reason for testing ?

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%
- Immediately (<happening>?)
  - Scale: seconds after <detection>
  - Current: 600 sec (=10 min)
  - Goal: 1 sec

Ref http://rsbatechnology.co.uk

Prioritize solutions by Impact Estimation

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

Quality

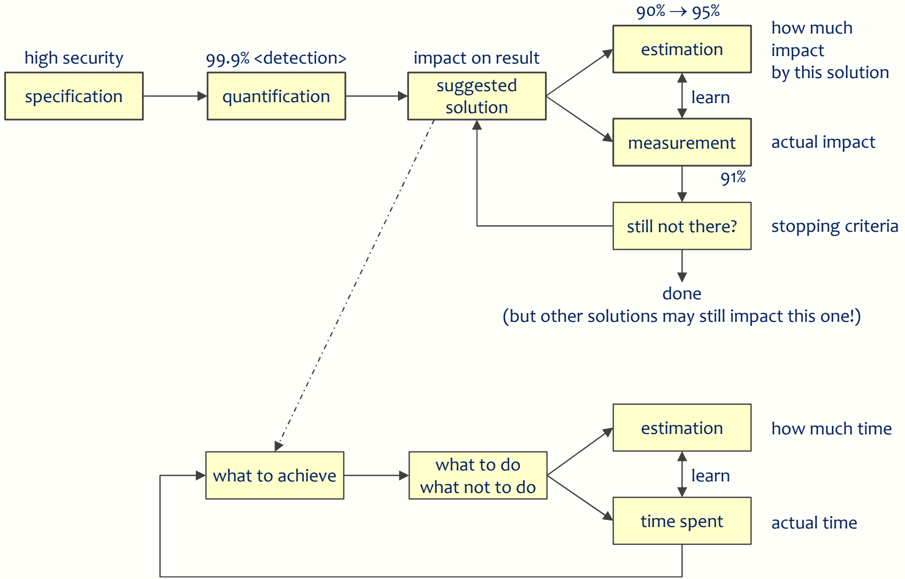
organising the

- what
- why
- for whom
- how much

on Time

organising the

- how
- when



high security

specification

99.9% <detection>

quantification

impact on result

suggested solution

90% → 95%

estimation

how much impact by this solution

learn

measurement

actual impact

91%

still not there?

stopping criteria

done

(but other solutions may still impact this one!)

what to achieve

what to do what not to do

estimation

how much time

learn

time spent

actual time

Examples of Scales (re-use of Requirements !)

Availability

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

Adaptability

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

Usability

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

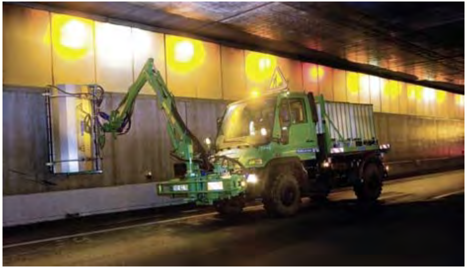
Reliability

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

Integrity

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

Availability



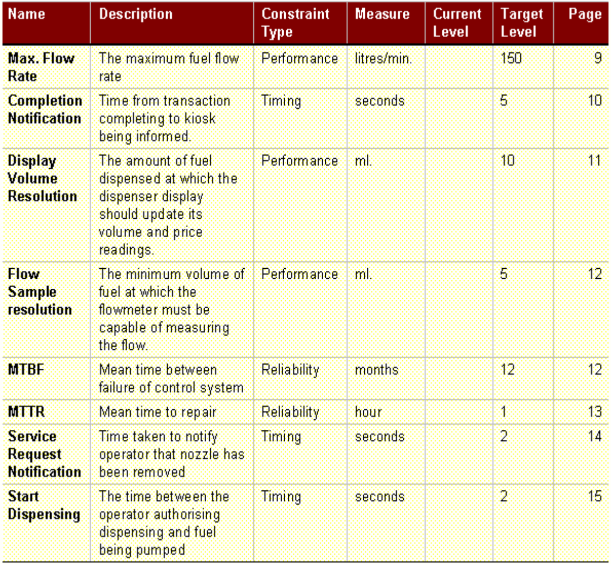
- Dependability.Availability
  - Readiness for service
  - Scale: % of <TimePeriod> a <System> is <Available> for its <Tasks>
- Probability that the system will be functioning correctly when it is needed
- Examples
  - (preventive) maintenance may decrease the availability
  - Snow on the road
  - Telephone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

Availability

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

Quantified Requirements

found on Internet



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

How about your requirements ?

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

Exercise 8: Try Planguage on your requirement

Definition:

RQ28:

Scale:

Meter:

Benchmarks (Playing Field):

Past:

Current:

Record:

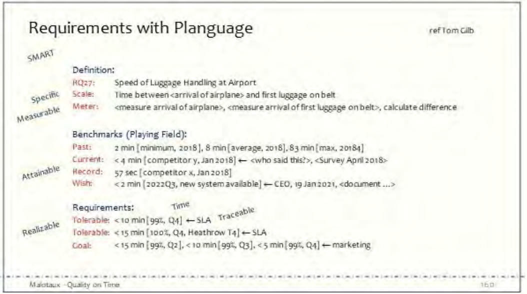
Wish:

Requirements:

Tolerable:

Tolerable:

Goal:



Requirements with Planguage

ref Tom Gilb

SMART

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, 2018], 8 min [average, 2018], 83 min [max, 20184]
Current: < 4 min [competitor y, Jan 2018] ← <who said this?>, <Survey April 2018>
Attainable
Record: 57 sec [competitor x, Jan 2018]
Wish: < 2 min [2022Q3, new system available] ← CEO, 19 Jan 2021, <document …>

Requirements:
Time
Traceable
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

Malotaux - Quality on Time 160

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

Is your TaskPlan for the coming week still valid ?

- After the exercises and examples ?
- If not: what would change ?

Now we are already much more efficient

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

but …

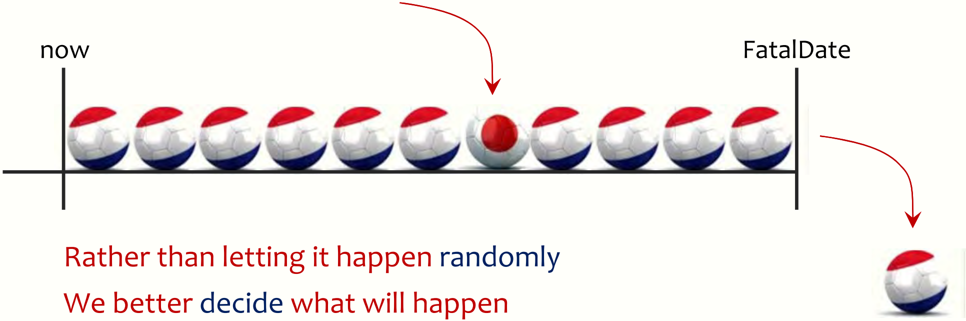
- How do we make sure the whole project is done on time ?

Evolutionary Planning

TimeLine

If we add something …

If we add something, something else will not be done

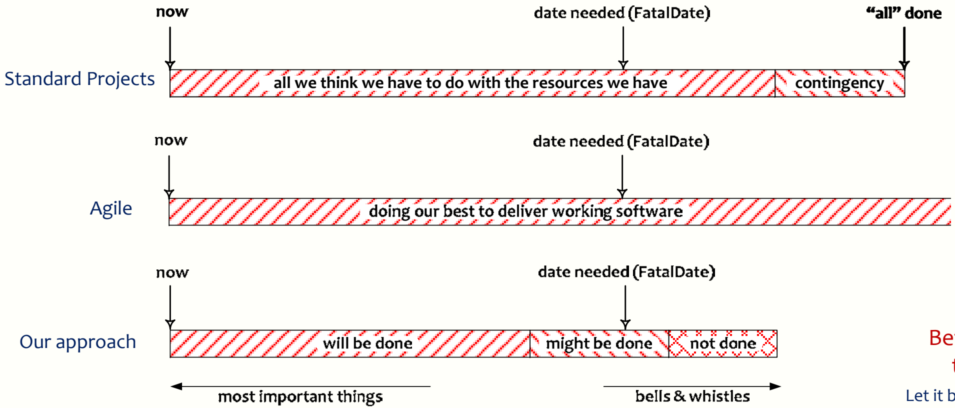


Rather than letting it happen randomly

We better decide what will happen

TimeLine

How do we know that we do, and get,
what is needed, *when* it's needed ?



Standard Projects

now

date needed (FatalDate)

"all" done

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

contingency

Agile

now

date needed (FatalDate)

doing our best to deliver working software

Our approach

now

date needed (FatalDate)

will be done

might be done

not done

most important things

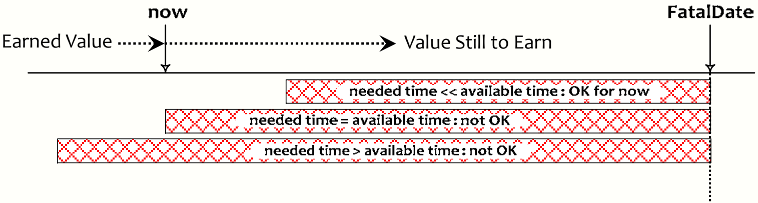
bells & whistles

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

Let it be the most important 80%

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

What are we going to do about it ?
because
Failure is not an option


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

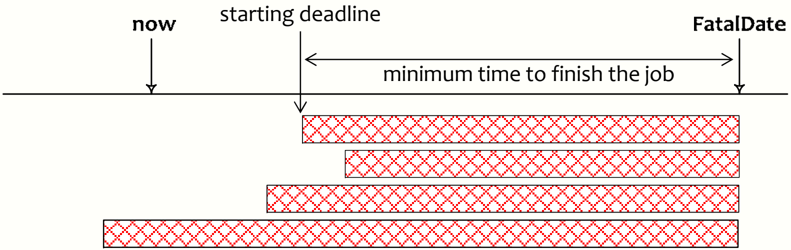
- Value Still to Earn ←versus→ Time Still Available
- If it doesn't fit ... count backwards
- If the match is over, we cannot score a goal

Even more important: Starting Deadlines

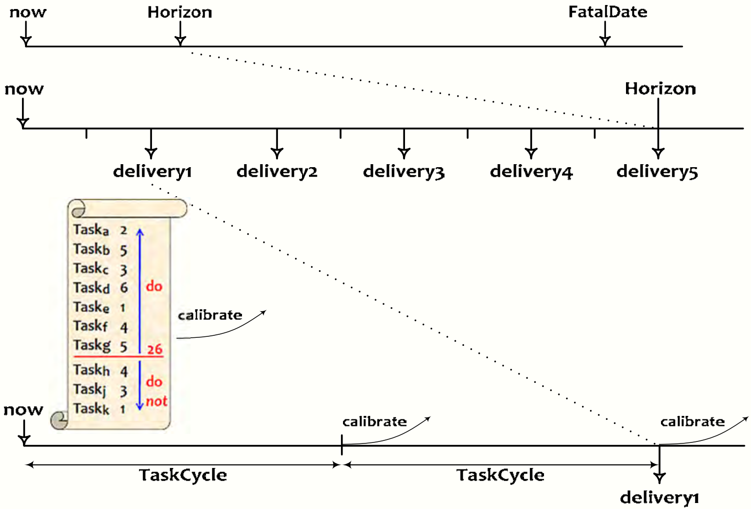
To meet Delivery Deadlines, focus on Starting Deadlines

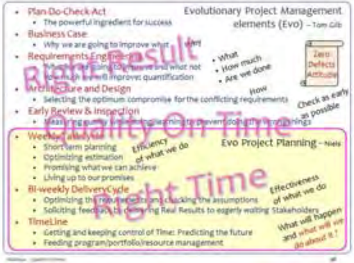
Starting Deadline

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



Result to Tasks and back




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 [illegible] learning to prevent doing the wrong things

Evo Project Planning – Niels

- Weekly [illegible]
  - 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

TimeLine: Predicting *what* may be done *when*

21/15 = 1.4

Line	Activity	Estim	Spent	Still to spend	Ratio real/est	Calibr factor	Calibr still to	Date done
1	Activity 1	2	2	0	1.0			
2	Activity 2	5	5	1	1.2	1.0	1	30 Mar 2009
3	Activity 3	1	3	0	3.0			
4	Activity 4	2	3	2	2.5	1.0	2	1 Apr 2009
5	Activity 5	5	4	1	1.0	1.0	1	2 Apr 2009
6	Activity 6	3				1.4	4.2	9 Apr 2009
7	Activity 7	1				1.4	1.4	10 Apr 2009
8	Activity 8	3				1.4	4.2	16 Apr 2009
↓	↓							
16	Activity 16	4				1.4	5.6	2 Jun 2009
17	Activity 17	5				1.4	7.0	11 Jun 2009
18	Activity 18	7				1.4	9.8	25 Jun 2009

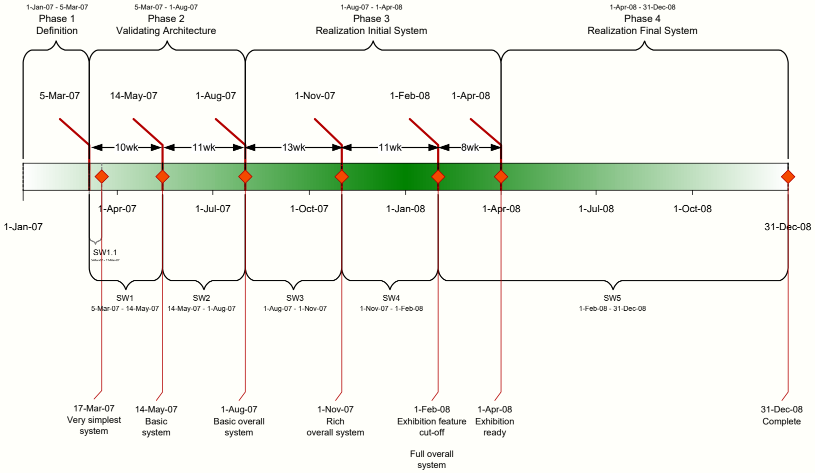
Product/Portfolio/Resource Management

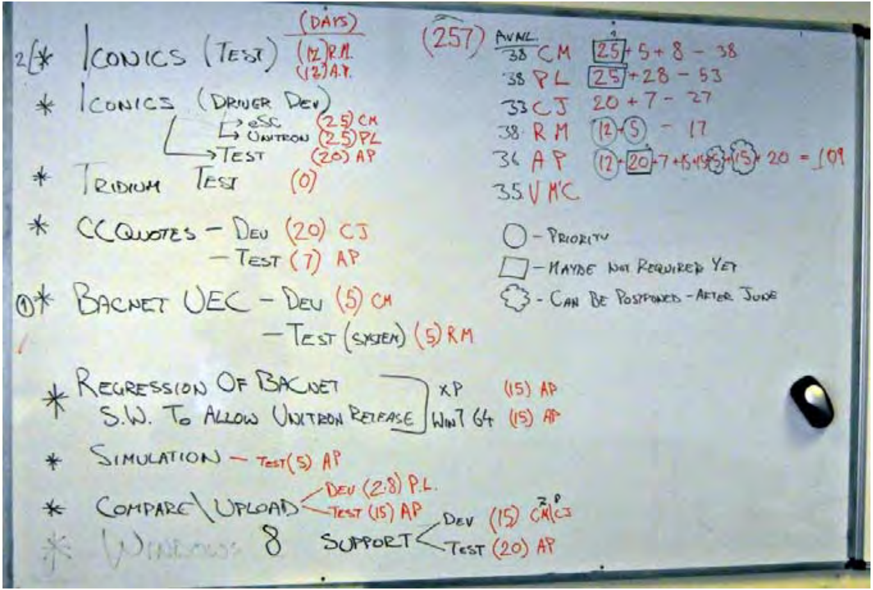
- Current Program/Portfolio/Resource Management is based on hope
- More a game than management

- With TimeLine we can provide PPR Management with *sufficiently* reliable data
- To start managing

TimeLine examples

TimeLine example





(DAYS)

2 [* ICONICS (TEST) (12) R.M. (12) A.P.

* ICONICS (DRIVER DEV)
→ eSC (25) CM
→ UNITRON (25) PL
→ TEST (20) AP

* TRIDIUM TEST (0)

* CCQUOTES – DEV (20) CJ
– TEST (7) AP

① * BACNET VEC – DEV (5) CM
– TEST (SYSTEM) (5) RM

* REGRESSION OF BACNET S.W. TO ALLOW UNITRON RELEASE
XP (15) AP
WIN7 64 (15) AP

* SIMULATION – TEST (5) AP

* COMPARE\UPLOAD
DEV (2.8) P.L.
TEST (15) AP

* WINDOWS 8 SUPPORT
DEV (15) CM\CJ
TEST (20) AP

(257) AVAIL.

38	CM	$\boxed{25}$ + 5 + 8 – 38
38	PL	$\boxed{25}$ + 28 – 53
33	CJ	20 + 7 – 27
38	RM	⑫ + ⑤ – 17
36	AP	⑫ + $\boxed{20}$ + 7 + 15 + 15 + 5 + 15 + 20 = 109
35	V M'C	

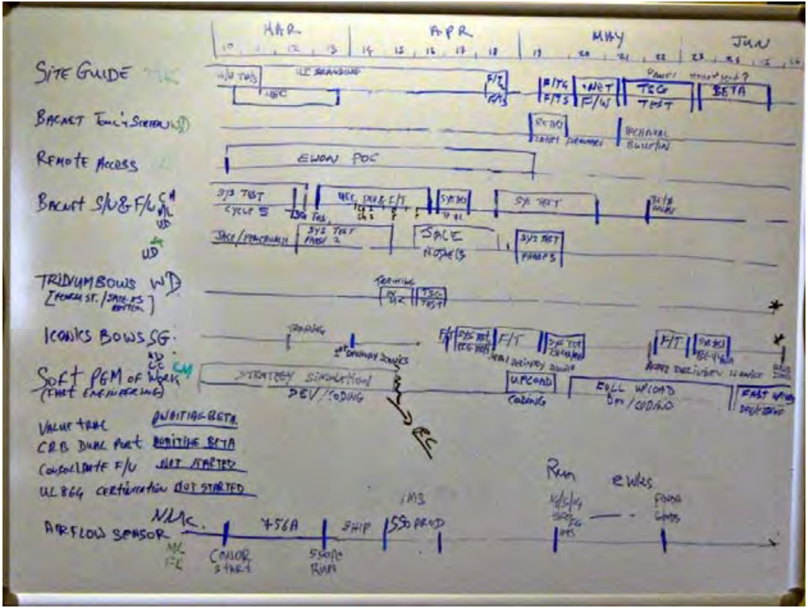
○ – PRIORITY

□ – MAYBE NOT REQUIRED YET

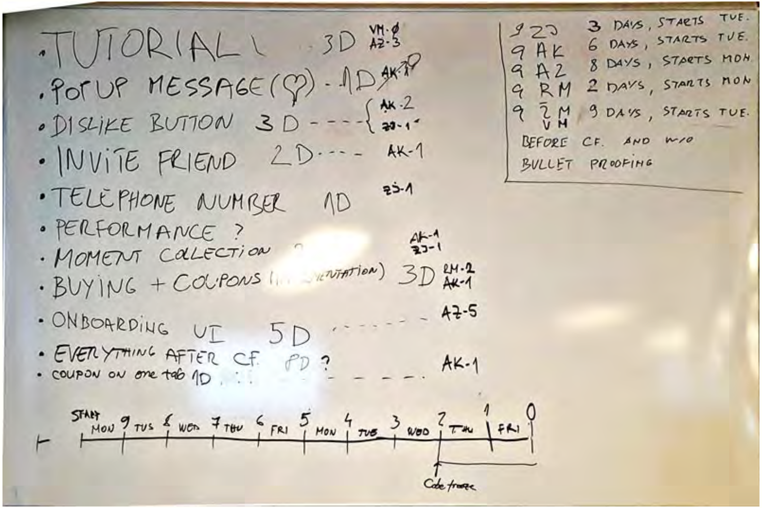
– CAN BE POSTPONED – AFTER JUNE

Designing

what to do when



Deadline in 10 days

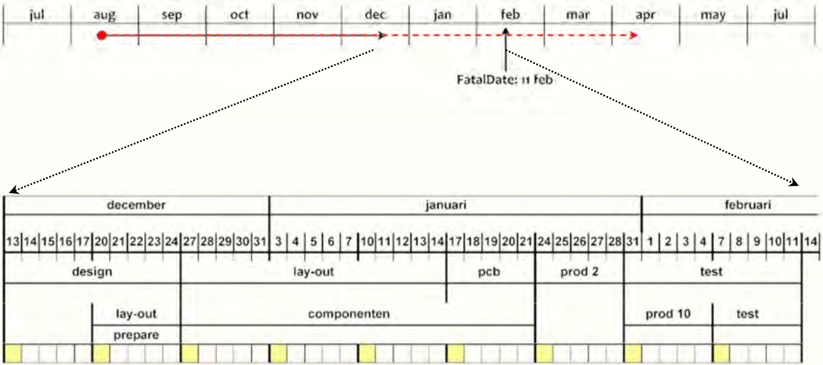


- TUTORIAL 3D VM-Ø AZ-3
- POPUP MESSAGE (♡) 1D AK-~~1~~2
- DISLIKE BUTTON 3D ---- { AK-2 ZJ-1
- INVITE FRIEND 2D ---- AK-1
- TELEPHONE NUMBER 1D ZJ-1
- PERFORMANCE ?
- MOMENT COLLECTION ? AK-1 ZJ-1
- BUYING + COUPONS (IMPLEMENTATION) 3D RM-2 AK-1
- ONBOARDING UI 5D ------ AZ-5
- EVERYTHING AFTER CF. 8D ? AK-1
- COUPON ON one tab 1D ------

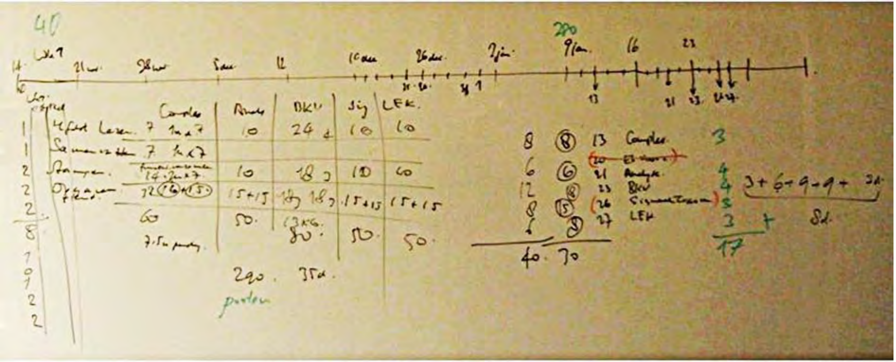
9	ZJ	3 DAYS, STARTS TUE.
9	AK	6 DAYS, STARTS TUE.
9	AZ	8 DAYS, STARTS MON.
9	RM	2 DAYS, STARTS MON
9	ZM VM	9 DAYS, STARTS TUE.

BEFORE CF. AND W/O BULLET PROOFING

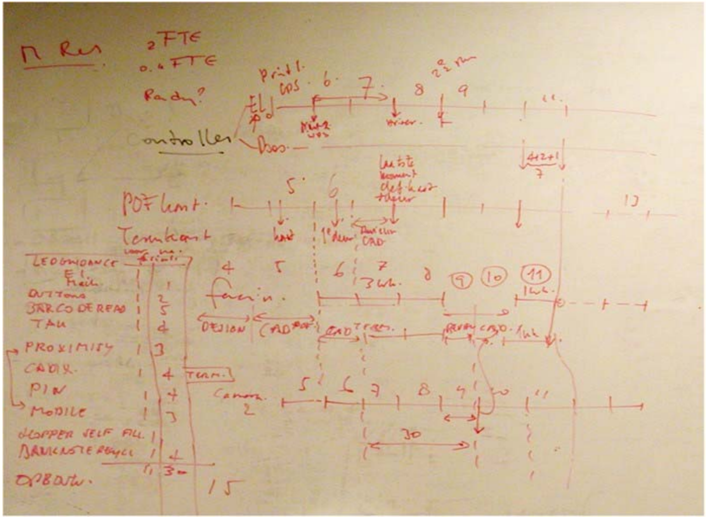
TimeLine planning



Preparing for student exams



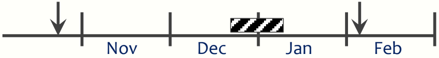
Synchronizing TimeLines: Counting back



Can we still do it ?
If not, why bother ?

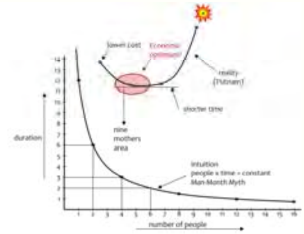
- Buying trains from the catalogue, but some changes
- Cannot change everything: limited set of focus areas
- Example:
  - Lifting train for maintenance
    - Supplier - lift
    - Maintenance - cable

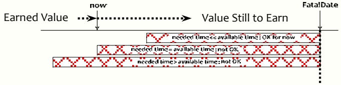
- How much time left ?
  - Supplier people already working on the final design
- What still to do? Does that fit the available time ?
  - Talk to our maintenance, talk to supplier, decision, agreement
- Why waste your time ?
- What is Plan B ?



Exercise 9: TimeLine for your Project

- Take a 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 (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 (Failure is not an option !)
- Case 3: go ahead (but don't waste 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

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 !

No excuse anymore !

- Delivering Quality on Time isn't really difficult
- I showed you some examples of how to do it
- So, there is no excuse anymore

  if you're not sure, just ask !
- From now on: just deliver the Right Results at the Right Time
- No complaining or excuses
- Magic Mantra:

What are we going to do about it ?!

What now ?

www.malotaux.eu/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

9 Predictable Projects (2012) - How to deliver the Right Results at the Right Time

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

www.malotaux.eu/inspections

Inspection pages

Inquiries: niels@malotaux.eu

Quality On Time

How to deliver the right results at the right time, no excuses needed

www.malotaux.eu/conferences
www.malotaux.eu/booklets

Niels Malotaux

+31-655 753 604 niels@malotaux.eu www.malotaux.eu

Help !

We have a QA Problem !

Niels Malotaux

niels@malotaux.eu
www.malotaux.eu/conferences (TestCon 2019)
www.malotaux.eu/booklets - booklet#8

www.malotaux.eu/doc.php?id=137 video TestCon2019: https://tinyurl.com/yckd87af

We have a QA problem !

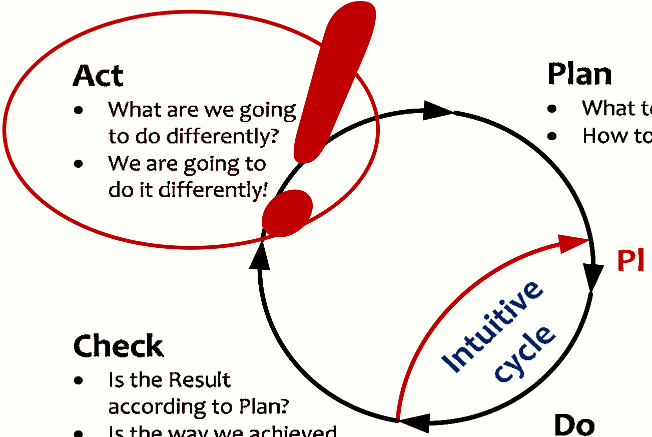
- Large stockpile of modules to test (hardware, firmware, software)
- You shall do Full Regression Tests
- Full Regression Tests take about 15 days each
- Too few testers ("Should we hire more testers ?")
- Senior Tester paralyzed
- Can you help us out ?

The essential ingredient: the PDCA Cycle

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

Deming

www.malotaux.eu/?id=PDCA



Act

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

Plan

- What to achieve
- How to achieve it

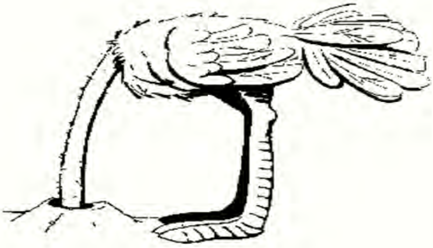
Check

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

Do

Do something

Carry out the Plan



Instead of complaining about a problem …

(Stuck in the Check-phase)

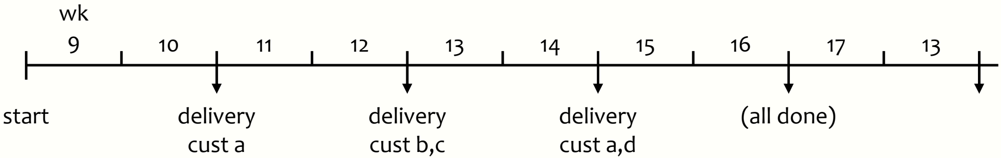
Let’s do something about it !

(Moving to the Act-phase)

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

Line	Activity	Estim	Alternative	Junior tester	Developers	Customer	Will be done ? (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	1 26	1			Sev	
9	Package 8.2	1	1			?	
10	Package 8.3	1	1			Chrt	
11	Package 8.4	1	1			Chrt	
12	Package 8.5	1.1	1.1			Yet	
13	Package 8.6	3	3			Yet	
14	Package 8.7	0.1	0.1			Cli	
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)

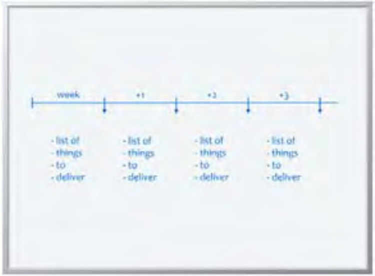
Selecting the priority order of customers to be served

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

Can we make an important customer happy the next day ?

Line	Activity	Estim	Alternative	Junior tester	Developers	Customer	Will be done (now=22Feb)
1	Package 1	17	2	17	4	HT	
2	Package 2	8	5		10	Chrt	
3	Package 3	14	7	5	4	BMC	
4	Package 4 (wait for feedback)	11				McC?	
5	Package 5	9	3		5	Ast	
6	Package 6	17	3	10	10	?	
7	Package 7	4	1		3	Cli	
8	Package 8.1	1	1			Sev	
9	Package 8.2	1	1			?	
10	Package 8.3	1	1			Chrt	
11	Package 8.4	1	1			Chrt	24 Feb
12	Package 8.5	1.1	1.1			Yet	26 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		

Result

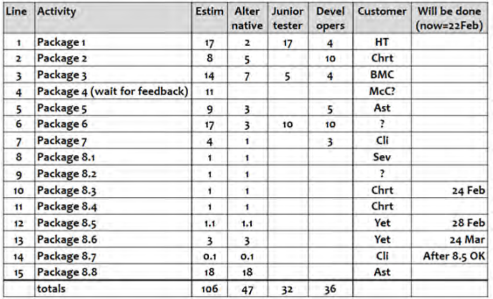


- Tester empowered
- Done in 9 weeks
- So-called “Full Regression Testing” was redesigned
- Customers systematically happy and amazed
- Kept up with development ever since
- Increased revenue

Later:

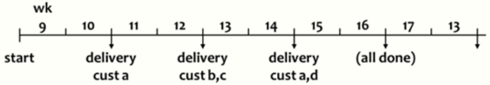
- Tester promoted to product manager
- Still coaching successors how to plan

TimeLine principles



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

- Cutting the work into chunks
- Estimating (usually takes very little time)
- Adding up (this averages the uncertainties !)
- Usually doesn't fit in the available time
- Find strategies to solve the dilemma
- Select 'best' strategy
- Predict what will happen when
- Learn and repeat every week, keeping predictions up-to-date



Help !

We have a QA Problem !

Problem Solved

Niels Malotaux

niels@malotaux.eu
www.malotaux.eu/conferences
www.malotaux.eu/booklets - booklet#8

Some extra

Active Synchronization

Somewhere around us, there is the bad world.

If we are waiting for a result outside our control, there are three possible cases:

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

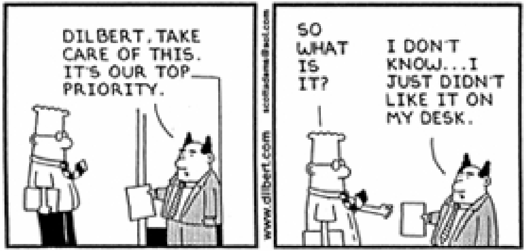
- If you are not sure (case 2), better assume case 3
- From other Evo projects you should expect case 1
- Evo suppliers behave like case 1

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

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

Interrupts

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


DILBERT, TAKE CARE OF THIS. IT'S OUR TOP PRIORITY.
www.dilbert.com
scottadams@aol.com
SO WHAT IS IT?
I DON'T KNOW... I JUST DIDN'T LIKE IT ON MY DESK.

- 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:
- Define the expected Results of the new Task properly
- Estimate the time needed to perform the new Task, to the level of detail really needed
- Go to your task planning tool (many projects use the ETA tool)
- Decide which of the planned Tasks is/are going to be sacrificed (up to the number of hours needed for the new Task)
- Weigh the priorities of the new Task against the Task(s) to be sacrificed
- Decide which is more important
- If the new Task is more important: replan accordingly
- 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
- Now we are still working on planned Tasks.

Requirements carved in stone ?

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

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

Architecture and Design

Design is always a compromise

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

example:

- Performance ↑
- Budget (time, money) ↓

Design and requirements

- Design:
  Finding the best compromise between the conflicting requirements

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

~~MoSCoW~~ ?

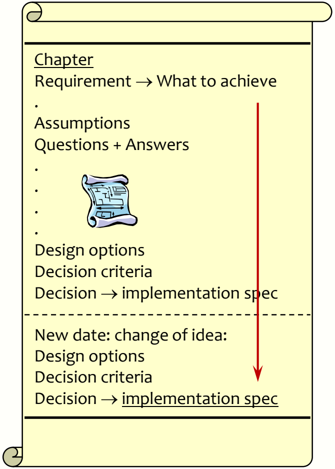
Design Process

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

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

DesignLog (project level)

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



Chapter

Requirement → What to achieve

.

Assumptions

Questions + Answers

.

.

.

.

Design options

Decision criteria

Decision → implementation spec

New date: change of idea:

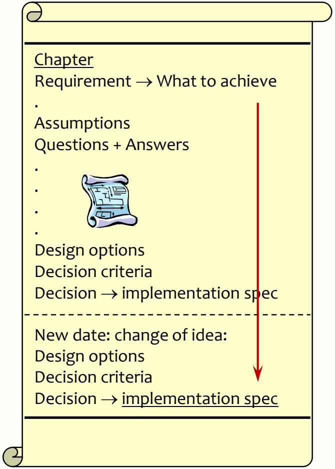
Design options

Decision criteria

Decision → implementation spec

ProcessLog (department / organization level)

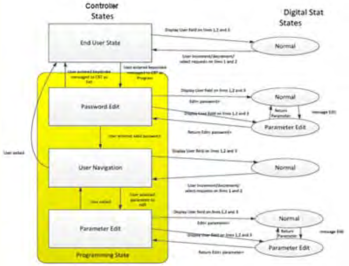
- In computer, not loose notes, not in e-mails, not handwritten
  - Text
  - Drawings!
  - On subject order
  - Initially free-format
  - For all to see
- All concepts contemplated
  - Requirement
  - Assumptions
  - Questions
  - Available techniques
  - Calculations
  - Choices + reasoning:
    - If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

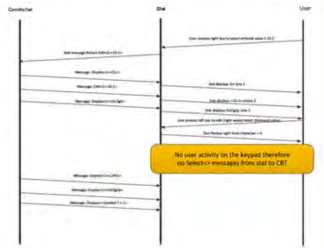


Chapter
Requirement → What to achieve
.
Assumptions
Questions + Answers
.
.
.
.
Design options
Decision criteria
Decision → implementation spec

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

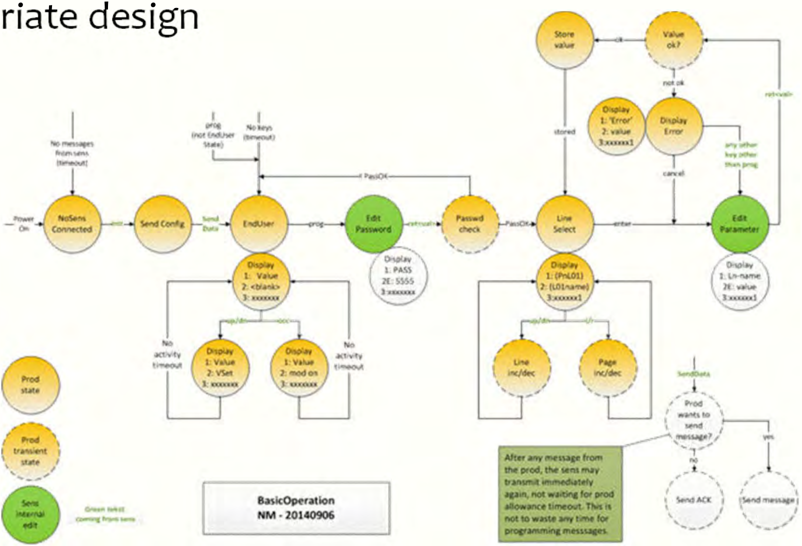
Useful design ?




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

Choose the appropriate design

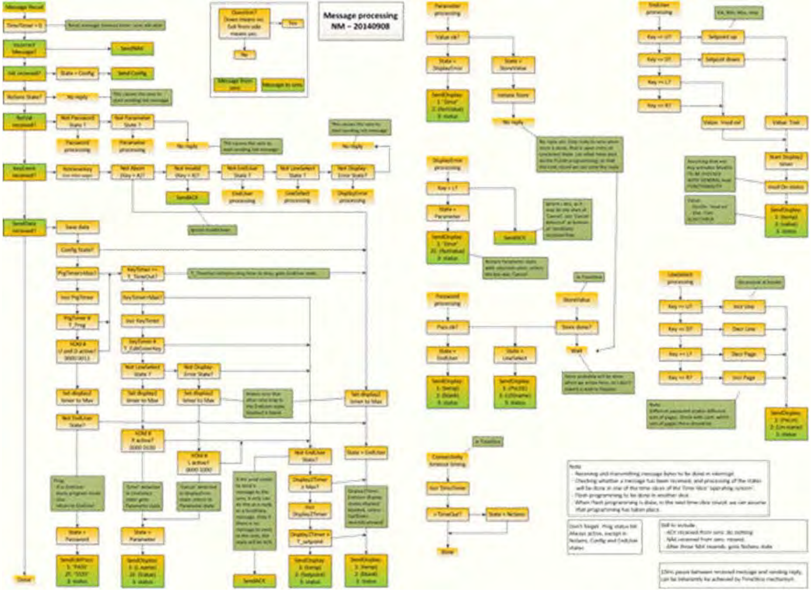
47 pages documentation condensed into one page



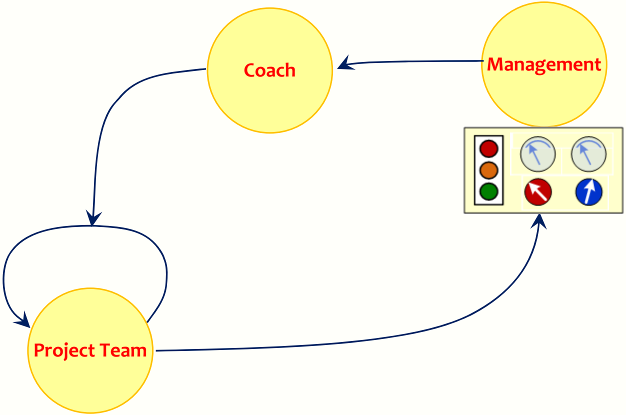
BasicOperation
NM - 20140906

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.

How could it look like ?



Local Loop Principle



What is a defect ?

A defect is the cause of a problem
experienced by any of the stakeholders
while relying on our results

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

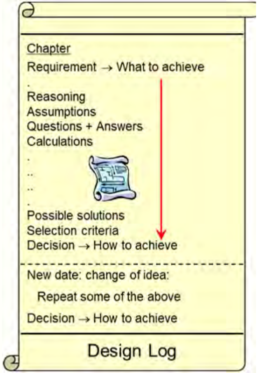
Design techniques

Cleanroom

- 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 the design description is lacking)
- What happens if you ask "Can I see the DesignLog ?"



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

In the pub

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

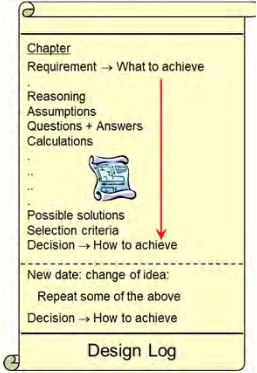
"We don't have time !" - "We've only 7 days !"

James insisted

We designed, found the problem, corrected it, cleaned up the mess

Done in less than 7 days

Thank you!



Chapter

Requirement → What to achieve

Reasoning

Assumptions

Questions + Answers

Calculations

Possible solutions

Selection criteria

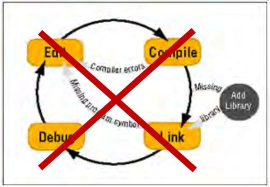
Decision → How to achieve

New date: change of idea:

Repeat some of the above

Decision → How to achieve

Design Log

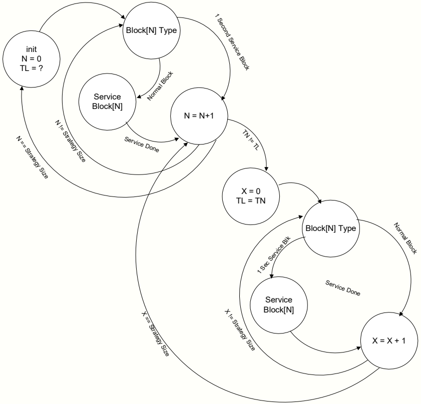


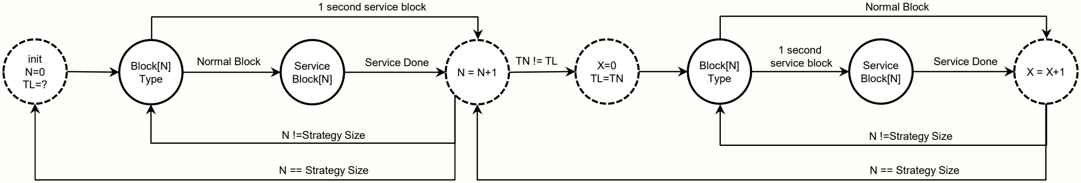
What James told me afterwards

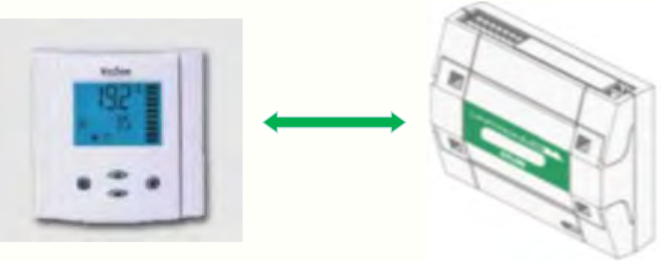
- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a ‘final’ review, with another colleague
- Based in his expertise, *the solution was completely reworked*
- Actually, two features were delivered and deployed
  - The one that was design and code reviewed had no issues after deployment
  - The 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

There are many ways to represent a design

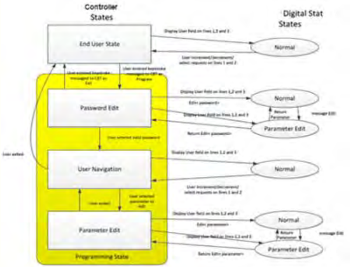
- Only few are useful
- Don't waste reviewer's time

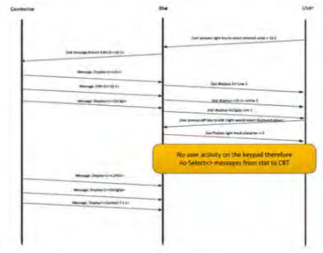






Useful design ?

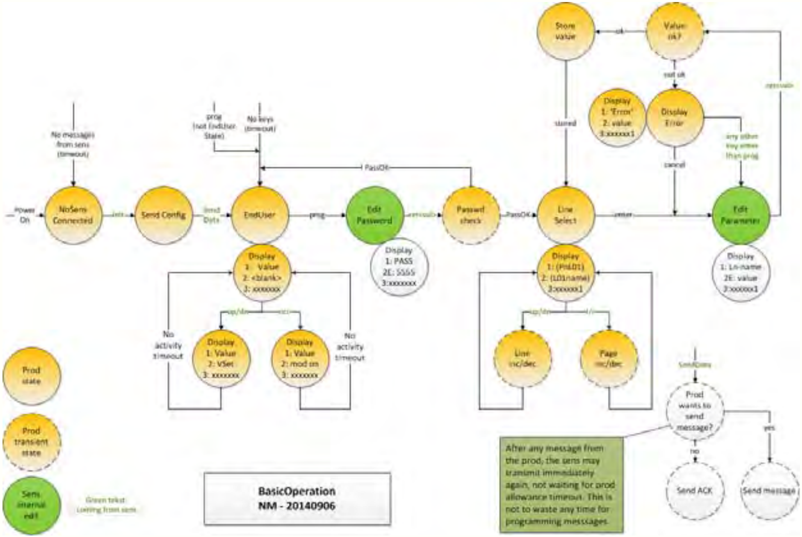




Controller
Stat
User
No user activity on the keypad therefore
no Select<> messages from stat to CBT

Choose the appropriate design

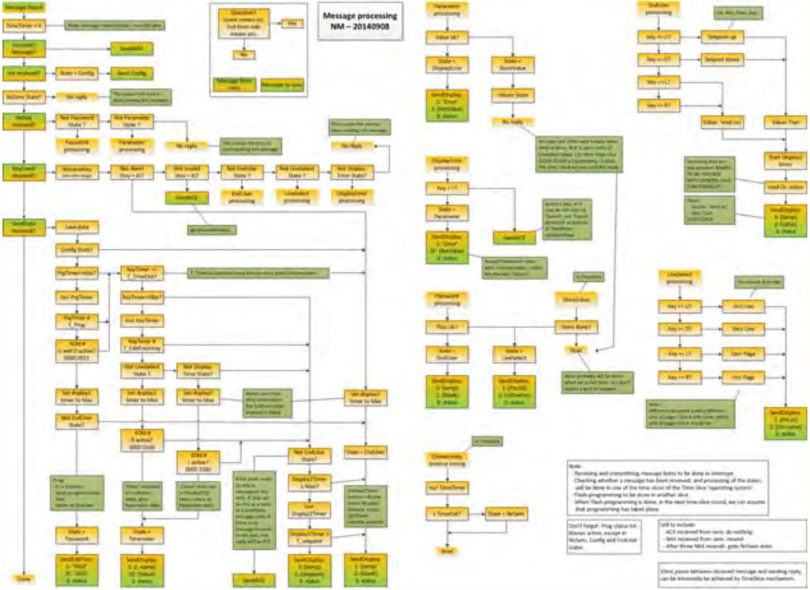
47 pages documentation condensed into one page


Power On
No messages from sens (timeout)
NoSens Connected
init
Send Config
Send Data
prog (not EndUser State)
No keys (timeout)
EndUser
PassOK
prog
Edit Password
Display 1: PASS 2E: SSSS 3:xxxxxxx
Passwd check
PassOK
Display 1: Value 2: <blank> 3: xxxxxxx
up/dn
No activity timeout
Display 1: Value 2: VSet 3: xxxxxxx
Display 1: Value 2: mod on 3: xxxxxxx
No activity timeout
Store value
ok
Value ok?
not ok
Display 1: 'Error' 2: value 3:xxxxxx1
Display Error
stored
cancel
any other key enter than prog
Line Select
enter
Edit Parameter
Display 1: (PnL01) 2: (L01name) 3:xxxxxx1
Display 1: Ln-name 2E: value 3:xxxxxx1
up/dn
Line inc/dec
Page inc/dec
SendData
Prod wants to send message?
yes
no
Send ACK
Send message
Prod state
Prod transient state
Sens internal edit
Green tekst coming from sens

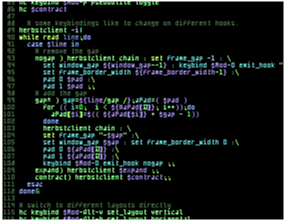
BasicOperation
NM - 20140906

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 messages.

How could it look like ?

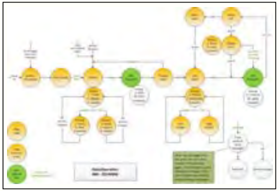


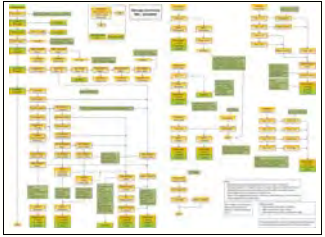
What is better than reviewing code ?

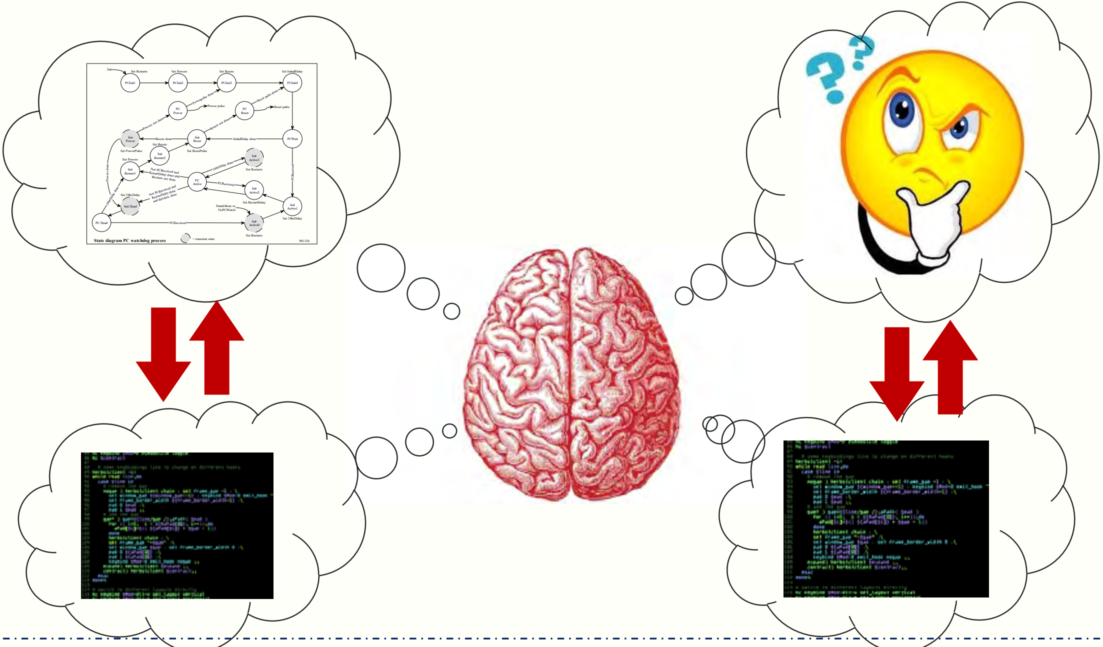


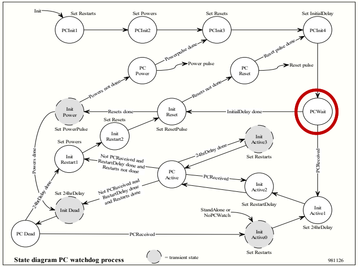
```
hc keybind $Mod-p pseudotile toggle
hc $contract

  # 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-O 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-O 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 ?

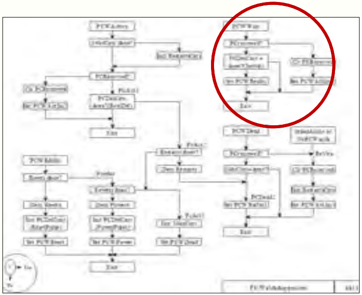


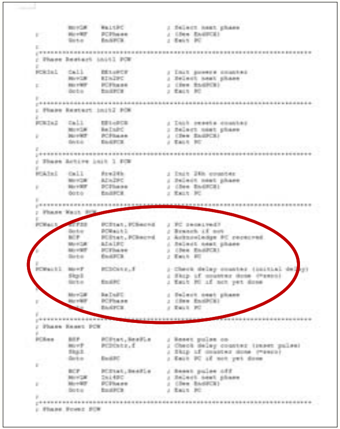



Init
Set Restarts
PCInit1
Set Powers
PCInit2
Set Resets
PCInit3
Set InitialDelay
PCInit4
PC Power
Powerpulse done
Power pulse
PC Reset
Reset pulse done
Reset pulse
Powers not done
Resets not done
Init Power
Resets done
Init Reset
InitialDelay done
PCWait
Set PowerPulse
Set Resets
Init Restart2
Set ResetPulse
Init Active3
Set Powers
Init Restart1
24hrDelay done
Set Restarts
Powers done
Not PCReceived and RestartDelay done and Restarts not done
PC Active
PCReceived
Init Active2
PCReceived
Set 24hrDelay
Not PCReceived and RestartDelay done and Restarts done
24hrDelay done
Init Dead
Set RestartDelay
StandAlone or NoPCWatch
Init Active1
Set 24hrDelay
PC Dead
PCReceived
Init Active0
Set Restarts
State diagram PC watchdog process
= transient state
981126



State diagram PC watchdog process

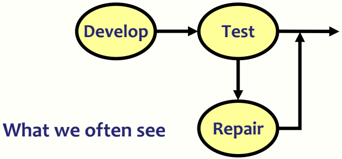


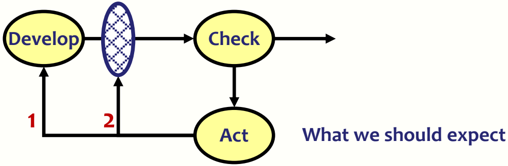


```
        MovLW   WaitPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCW)
        Goto    EndPCW          ; Exit PC
;
;**************************************************************************
; Phase Restart init1 PCW
;
PCRIn1  Call    EEtoPCP         ; Init powers counter
        MovLW   RIn2PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCW)
        Goto    EndPCW          ; Exit PC
;
;**************************************************************************
; Phase Restart init2 PCW
;
PCRIn2  Call    EEtoPCR         ; Init resets counter
        MovLW   ResInPC         ; Select next phase
;       MovWF   PCPhase         ; (See EndPCW)
        Goto    EndPCW          ; Exit PC
;
;**************************************************************************
; Phase Active init 1 PCW
;
PCAIn1  Call    Pre24h          ; Init 24h counter
        MovLW   AIn2PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCW)
        Goto    EndPCW          ; 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 EndPCW)
        Goto    EndPCW          ; 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   ResInPC         ; Select next phase
;       MovWF   PCPhase         ; (See EndPCW)
        Goto    EndPCW          ; 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 EndPCW)
        Goto    EndPCW          ; Exit PC
;
;**************************************************************************
; Phase Power PCW
```

What’s in it for QA ?

- 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’





- QA to help developers to produce less and less defects

Do we deliver Zero Defect software ?

Better quality costs less

- 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 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” ?

- Can you afford not to use these techniques?

The Simplest and Best Agile Project Method - ‘Evo’

ref. Tom Gilb

- Gather from all the key stakeholders the top few (5 to 10) most critical goals that the project needs to deliver

  Give each goal a reference name (a tag)
- For each goal, define a scale of measure and a ‘final’ goal level

  For example: Reliable: Scale: Mean Time Before Failure, Goal: 1 month
- Define approximately 4 budgets for your most limited resources

  For example: time, people, money, equipment
- Write up these plans for the goals and budgets

  Try to ensure this is kept to only one page
- Negotiate with the key stakeholders to formally agree the goals and budgets
- Plan to deliver some benefit

  that is, progress towards the goals in weekly (or shorter) increments (Evo steps)
- Implement the project in Evo steps

  Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures, for each performance goal and each resource budget.

  On a single page, summarize the progress to date towards achieving the goals and the costs incurred
- When all Goals are reached: ‘Claim success and move on’

  Free remaining resources for more profitable ventures

Human Behavior

Moved to the final keynote on 27 October

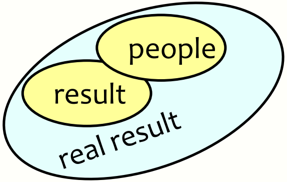
Human Behavior

- 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

Hey, they shouldn't behave like that ! Well if they do, they do. They're only human.

Movie: Six Days, Seven Nights

Is Human Behavior a risk?



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

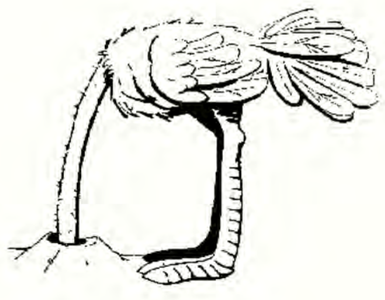
Excuses, excuses, excuses …

- 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”
- Every day we see a problem earlier, we have a day more to do something 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”
  - Because we didn’t help the person not to fail

The aim of Testing

- Being done as soon as the development is done
  - Well, almost

- Excuses, excuses, excuses
  - The developers are always late (What did we do to help them ?)
  - The developers don't take us seriously (Developers could ask testers for help)
  - The developers don't inject enough defects (Now testing becomes a real challenge)
  - We are the bearers of bad news (Find out what we're really supposed to do)
- Helping the development to be successful



Our problems are not the real problem,
the real problem is when we don’t do something about it

Every day we see a problem earlier,
we have a day more to do something about it

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

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

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

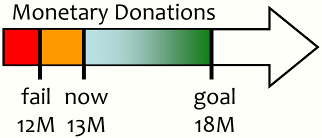
Ref Ryan Shriver - ACCU Overload Feb 2009

Requirements Case - Wish

- Organization collecting online giving for charities
- CEO: “Improve website to increase online giving for our ‘customers’ (charities)”
- Increasing market share for online giving
- Budget: 1M€ - 10 months
- Show results fast

Ref Ryan Shriver - ACCU Overload Feb 2009

Objective: Monetary Donations

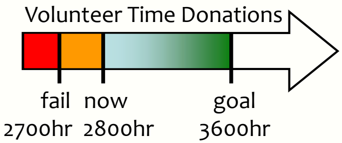


Name	Monetary Donations
Scale	Euro’s donated to non-profits through our website
Meter	Monthly Donations Report

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

Ref Ryan Shriver - ACCU Overload Feb 2009

Objective: Volunteer Time (Natura) Donations

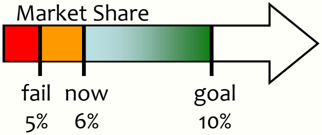


Name	Volunteer Time Donations
Scale	Hours donated to non-profits through our website
Meter	Monthly Donations Report

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

Ref Ryan Shriver - ACCU Overload Feb 2009

Goal: Market Share



Name	Market Share
Scale	Market Share %% online giving
Meter	Quarterly Industry Report

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

Ref Ryan Shriver - ACCU Overload Feb 2009

Impact Estimation example

Impact Estimation	Monthly Donations	Facebook integration	Image & video uploads	Total effect for requirement
€€ donations 13M€ → 18M€	80%	30%	50%	160%
Time donations 2800hr→3600hr	10%	50%	80%	140%
Market share 6% → 10%	30%	30%	20%	80%
Total effect per solution	120%	110%	150%	380%
Cost - money % of 1M€	30%	20%	50%	100%
Cost - time % of 10 months	40%	20%	50%	110%