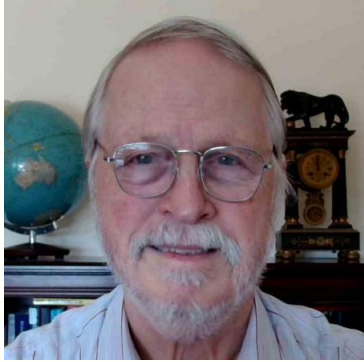
Niels Malotaux:
In my experience the 'zero defects' attitude results in 50% less defects almost overnight

Examples of how to move towards Zero Defects

Niels Malotaux

niels@malotaux.eu

Niels Malotaux



- Independent Engineering and Project Coach
- Expert helping teams and organizations quickly to become
  - More effective - doing the right things better
  - More efficient - doing the right things better in less time
  - More predictable - delivering as needed
- Project rescue (helping solving 'impossible' deadlines)
- Embedded Systems architect (electronics/firmware)
- Project types
  electronic products, firmware, software, space, road, rail,
  telecom, building automation, industrial control, parking system

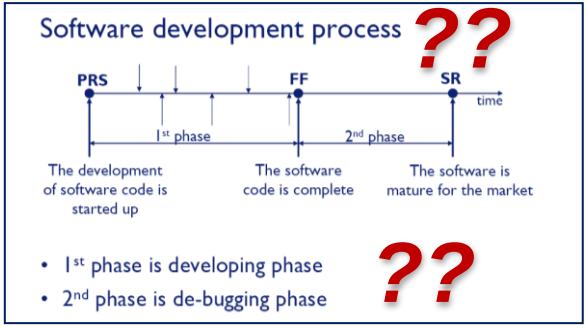
niels@malotaux.eu

Is software without defects possible ?

- How many defects are acceptable ?
- Requirements specify a certain number of defects ?
- Check the required number has been produced ?

In your work

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



Requirement:

- ~~Deliver 7 bugs in every sprint~~

User Story:

- ~~As a customer, I like to experience some bugs, so that I regularly can complain~~

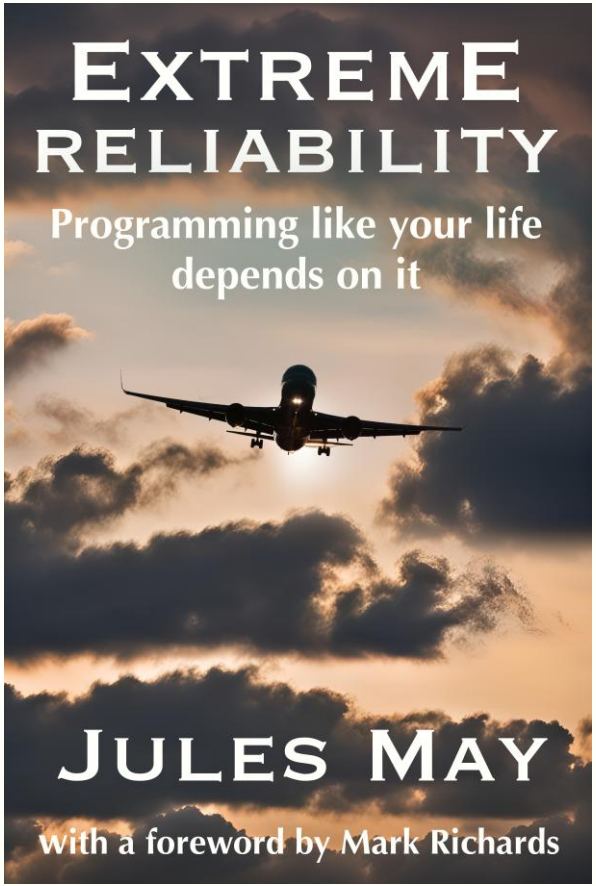
Better quality costs less

Jules May: Extreme Reliability

All my experience has convinced me
that it takes less time to code it right first time
than knowingly doing it wrong

It's not just a long-term benefit:

it's *quicker in the short term as well*



New QA manager at a large bank

No defects in the first two weeks of use

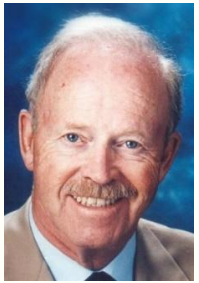
What is a defect ?

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

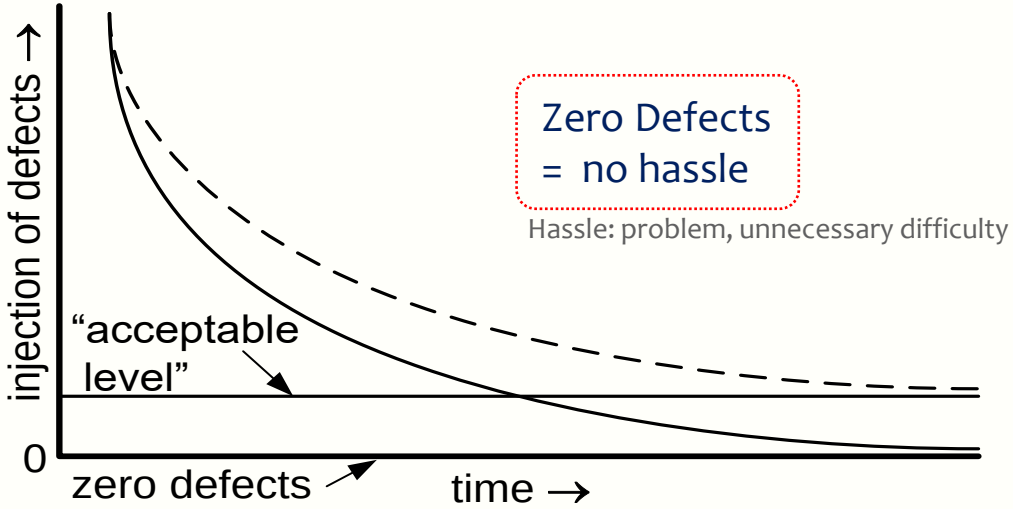
What is Zero Defects ?

We aren't perfect,
but the customer shouldn't find out

- Zero Defects is an *asymptote*



Niels Malotaux:
In my experience the 'zero defects' attitude results in 50% less defects almost overnight

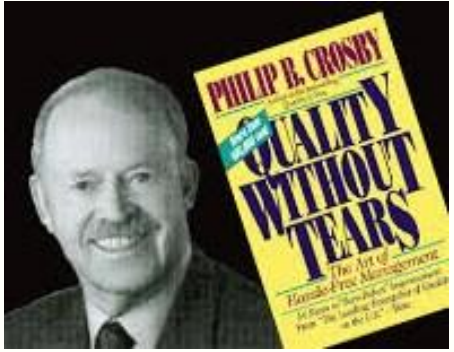


Zero Defects
= no hassle

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

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

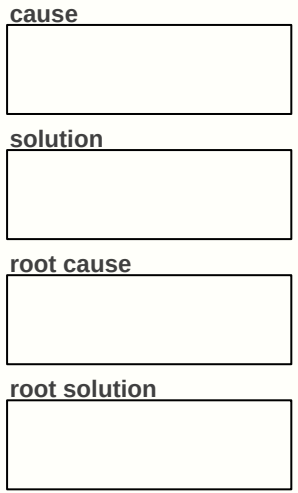
Universal goal

Quality on Time

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

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

Feeding prevention: Root Cause Analysis

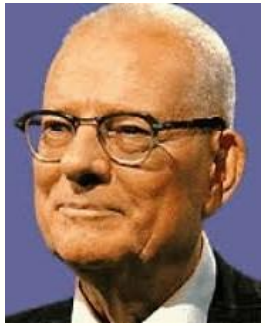


- Is Root Cause Analysis routinely performed - every time ?
- What is the difference between *Cause* and *Root Cause* ?

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

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

Doesn't Testing and QA 'assure' quality ?



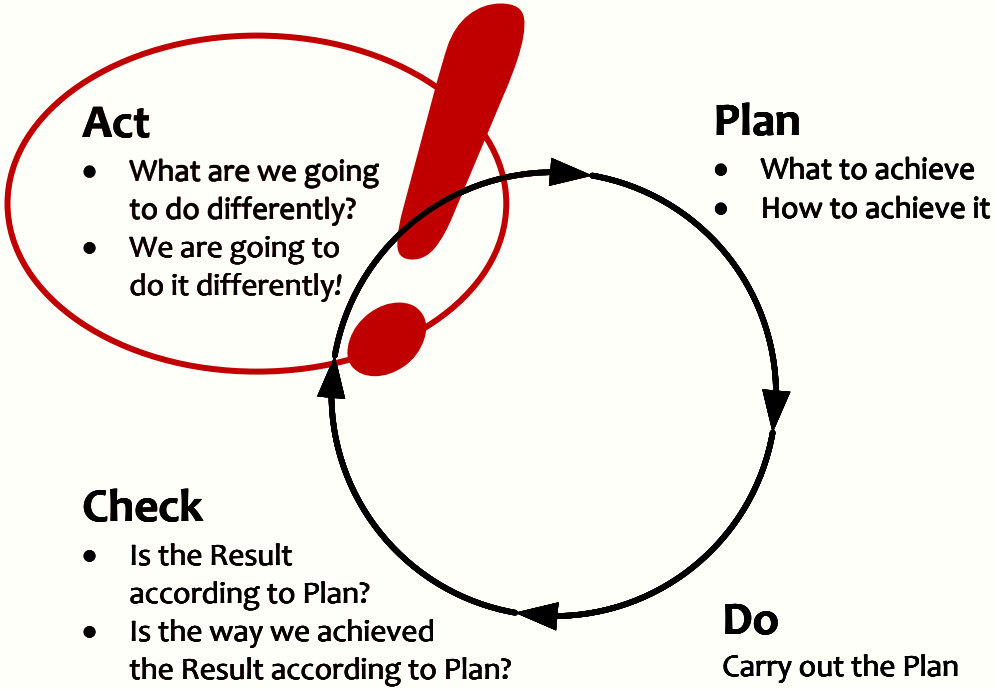
Deming
(1900-1993)

- 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 are the main customers of Testing and QA ?
- What to deliver to these customers ?
  What are they *waiting for* ?
- Testers and QA are *consultants* to development

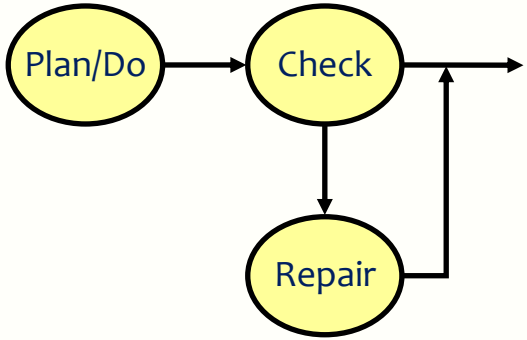
- Providing the customer with
  - what they need
  - at the time they need it
  - to be satisfied
  - to be more successful than without it
- Constrained by (win - win)
  - what the customer can afford
  - what we mutually beneficially and satisfactorily can deliver
  - in a reasonable period of time

The essential ingredient: the PDCA Cycle

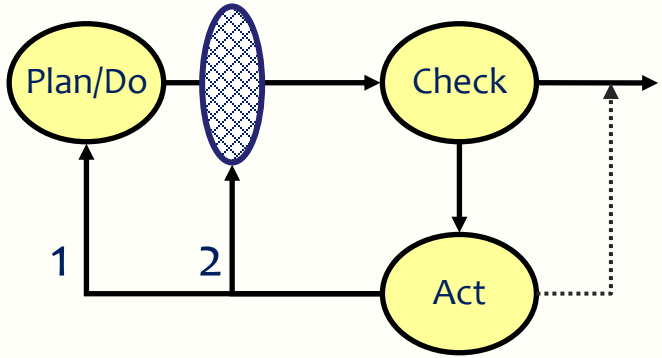
(Shewhart Cycle - Deming Cycle - Plan-Do-Study-Act Cycle - Kaizen)



How can we prevent defects ?



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 ?

Some Examples

We’re not perfect,

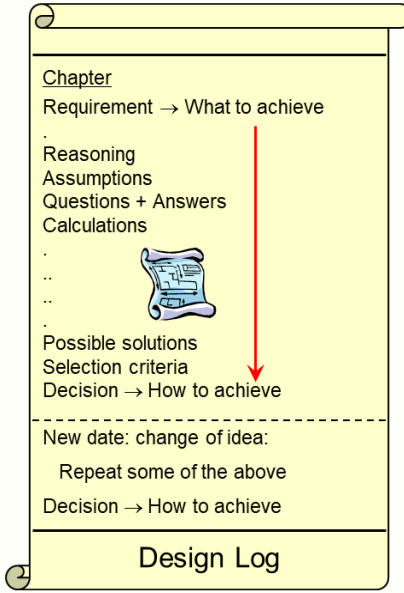
but the customer shouldn’t find out

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

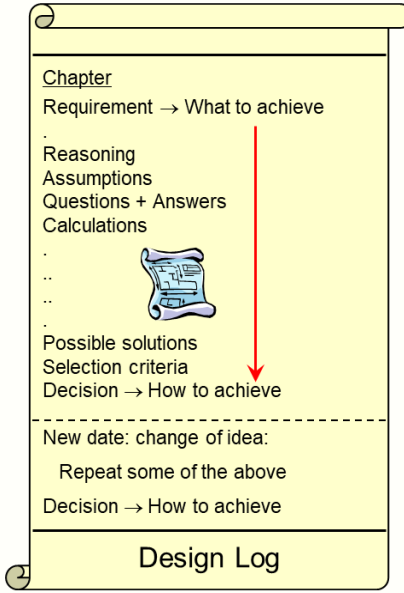
Case: In the pub

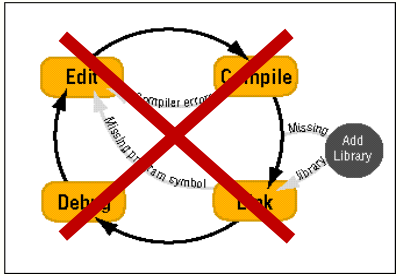
James:

Niels, this is Louise
Louise, this is Niels, who taught me about DesignLogging
Tell what happened

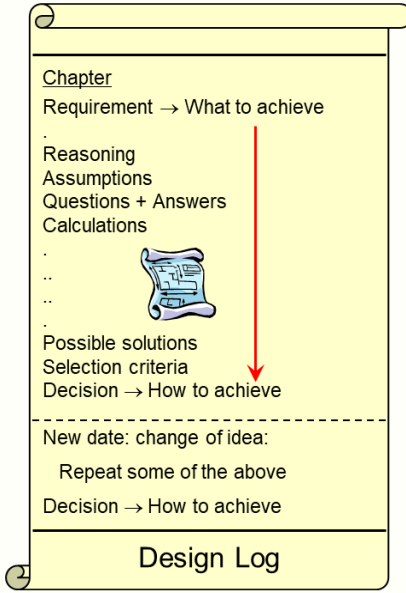
Louise:

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!





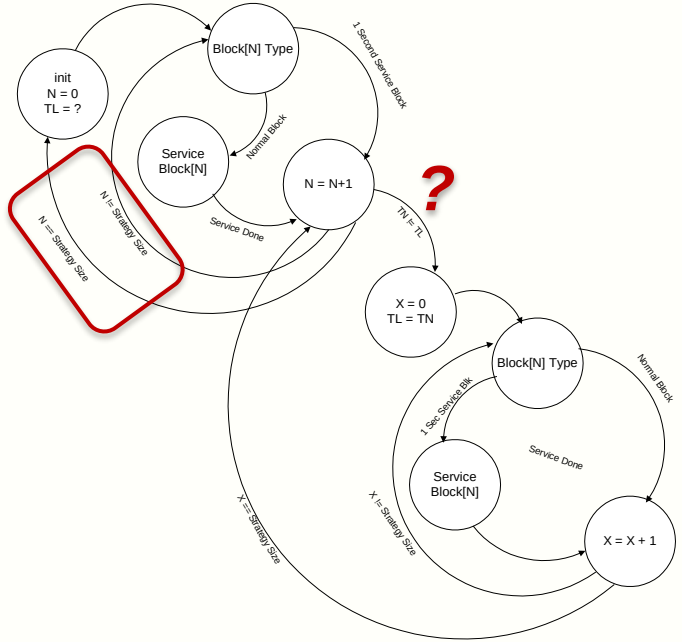
What James told me afterwards

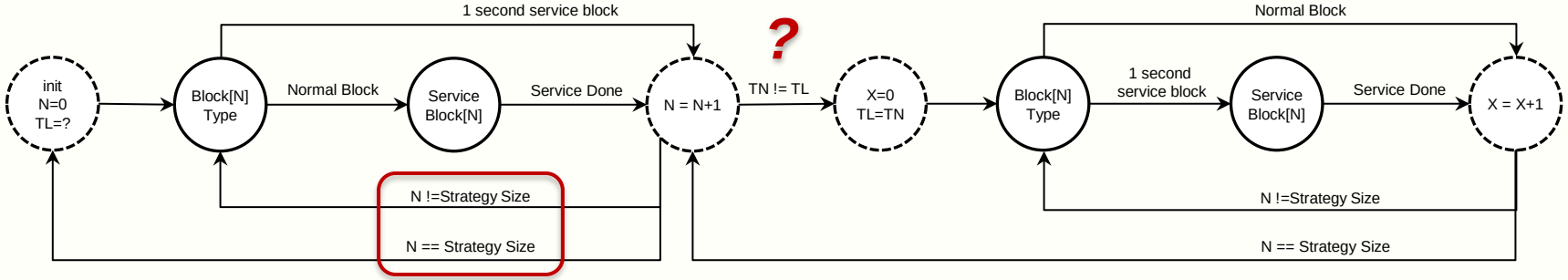


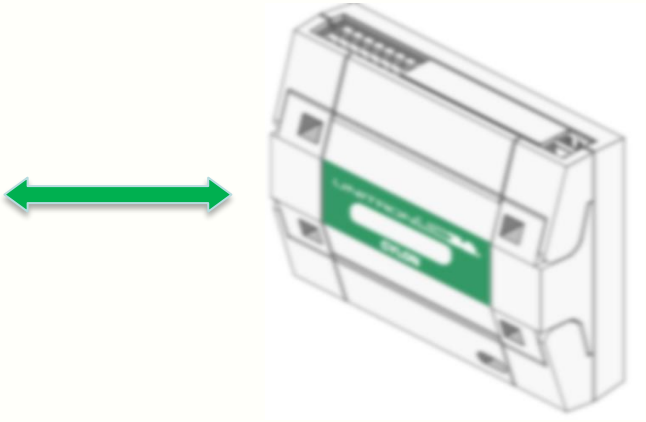
- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a 'final' review, with another colleague
- Based on his expertise, *the solution was completely reworked*
- Actually, two features were delivered and deployed
  - The one that was design and code reviewed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

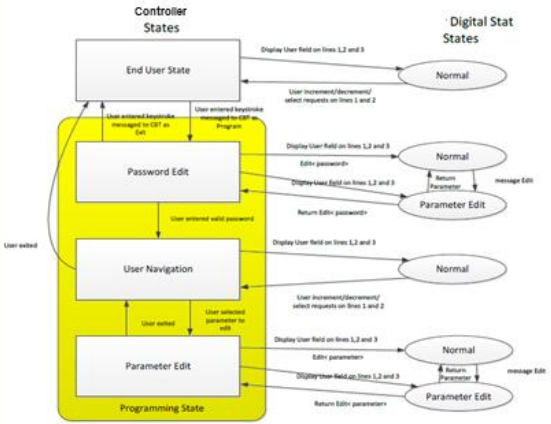
- Only few ways are useful
- Don't waste reviewer's time

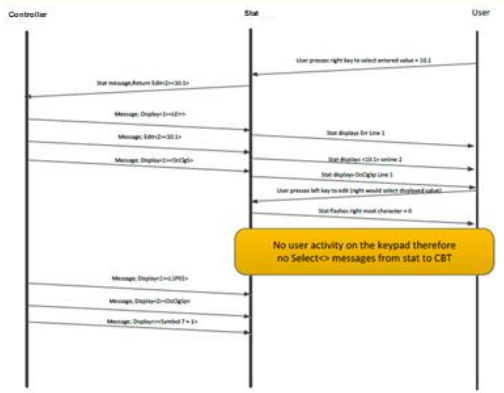






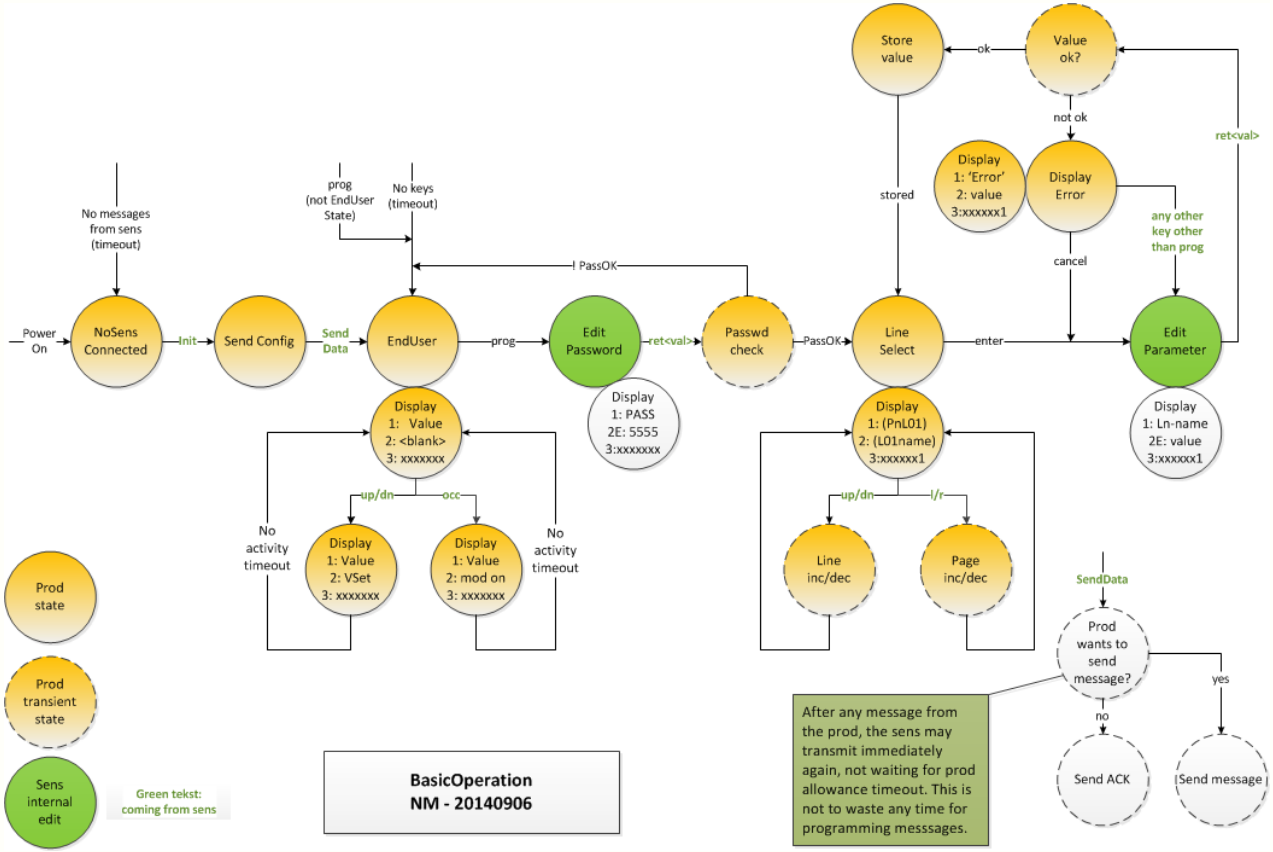
Useful design ?



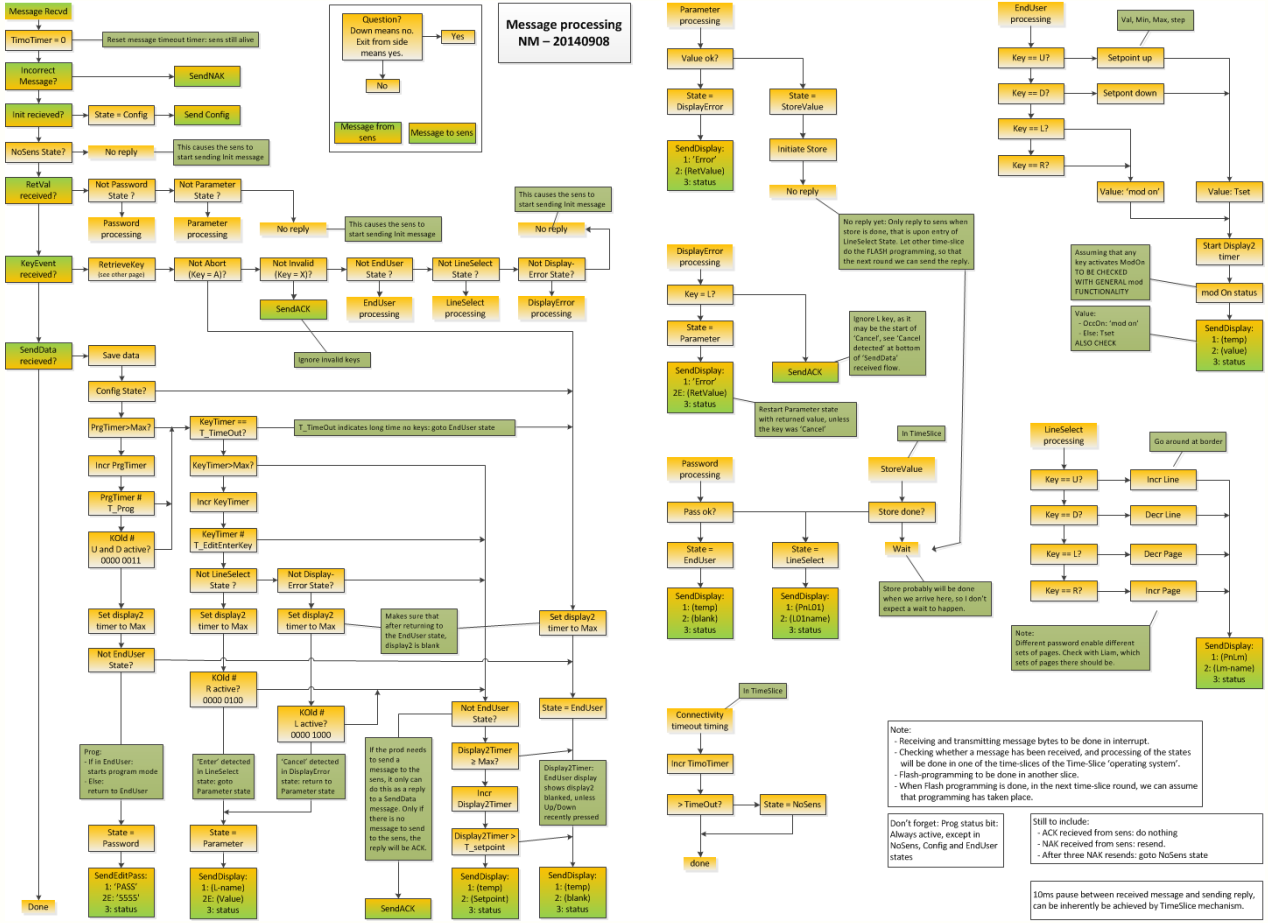


Choose the appropriate design

47 pages documentation
condensed into one page



Translating into flow

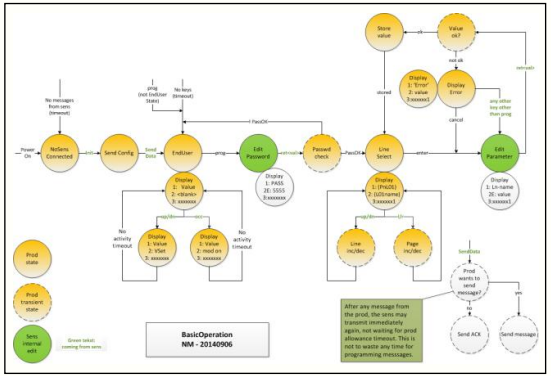


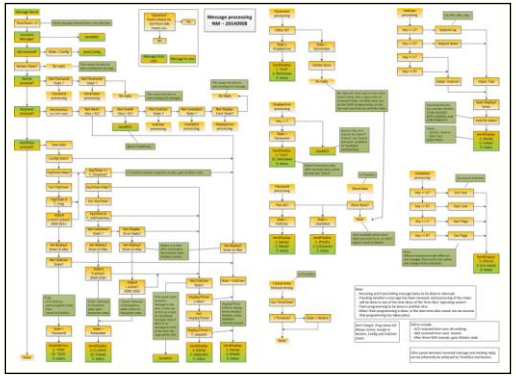
What is better than reviewing code ?

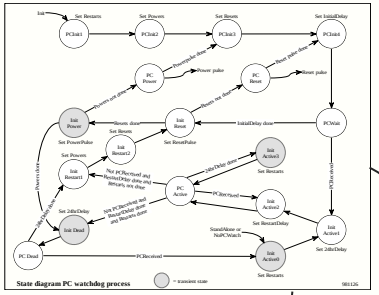
```
hc $contract

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

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

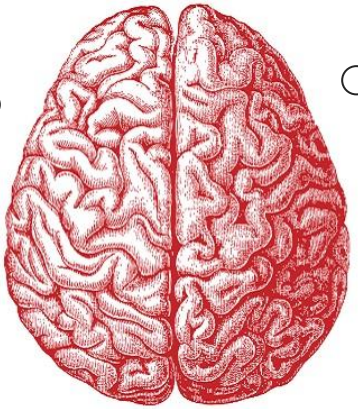
- Do you ever review software ?
- What do you review ?

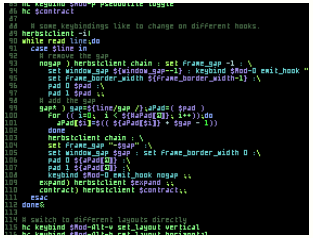
- What is better than reviewing code ?
  - May I review the design first ?

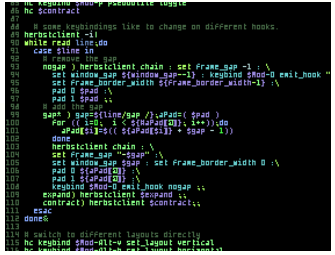


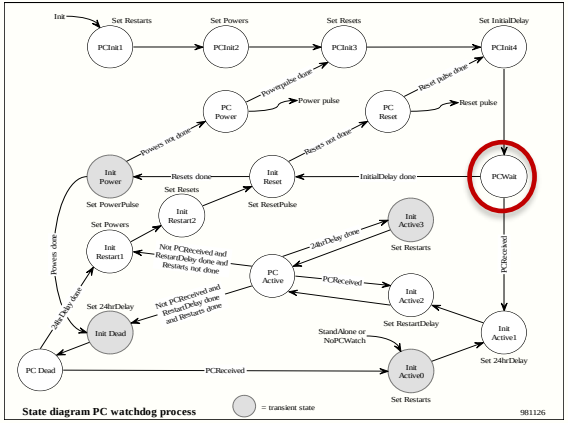


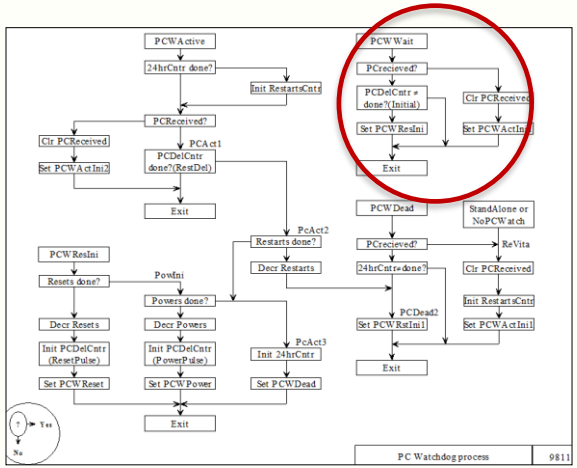

State diagram PC watchdog process
transient state












```
        MovLW   WaitPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Restart init1 PCW
;
PCRIn1  Call    EEtoPCP         ; Init powers counter
        MovLW   RIn1PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Restart init2 PCW
;
PCRIn2  Call    EEtoPCR         ; Init resets counter
        MovLW   ReInPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Active init 1 PCW
;
PCAIn1  Call    Pre24h          ; Init 24h counter
        MovLW   AIn2PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Wait PCW
;
PCWait  BTFSS   PCStat,PCRecvd  ; PC received?
        Goto    PCWait1         ; Branch if not
        BCF     PCStat,PCRecvd  ; Acknowledge PC received
        MovLW   AIn1PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
PCWait1 MovF    PCDCntr,f       ; Check delay counter (initial delay)
        SkpZ                    ; Skip if counter done (=zero)
        Goto    EndPC           ; Exit PC if not yet done
;
        MovLW   ReInPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Reset PCW
;
PCRes   BSF     PCStat,ResPls   ; Reset pulse on
        MovF    PCDCntr,f       ; Check delay counter (reset pulse)
        SkpZ                    ; Skip if counter done (=zero)
        Goto    EndPC           ; Exit PC if not yet done
;
        BCF     PCStat,ResPls   ; Reset pulse off
        MovLW   Ini4PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Power PCW
```

Case: Scrum Sprint Planning

- What is the measure of success for the coming sprint ?
- What a strange question ! We're Agile !
  We deliver working software. Don't you know ?
- Note: Users are not waiting for software:
  they just need *improved performance* of what they're doing
- How about a requirement for 'Demo': *No Questions – No Issues*
- That's impossible !!
- They actually succeeded !

Demo ??

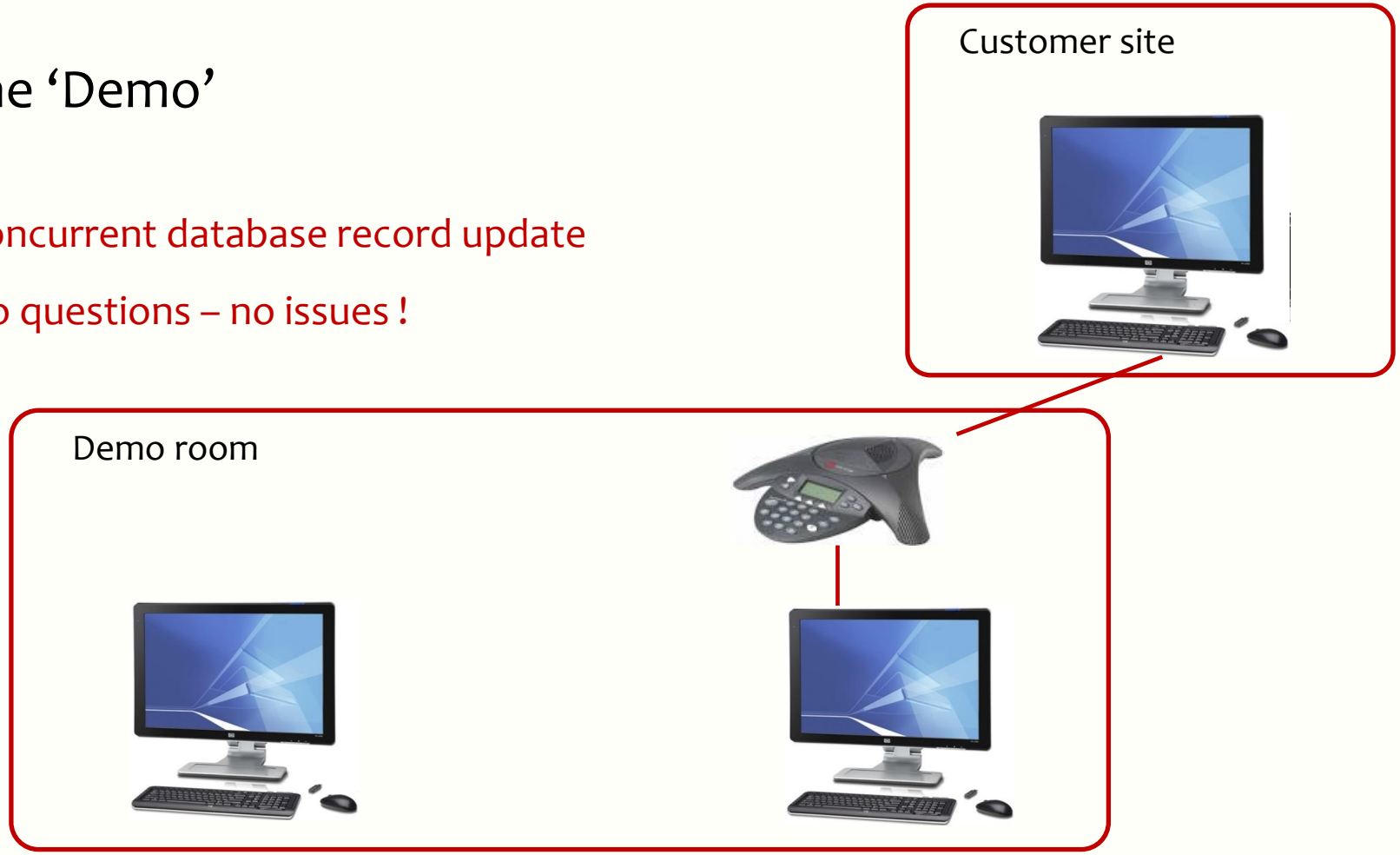
- Give the delivery to the stakeholders
- Zip your mouth
- Keep your hands handcuffed on your back
- and o-b-s-e-r-v-e what happens
- Seeing what the stakeholders actually do provides real feedback
- Then we can ‘talk business’ with the stakeholders

- Is this what you do ?

The ‘Demo’

Concurrent database record update

No questions – no issues !



Delivery Strategy Suggestions

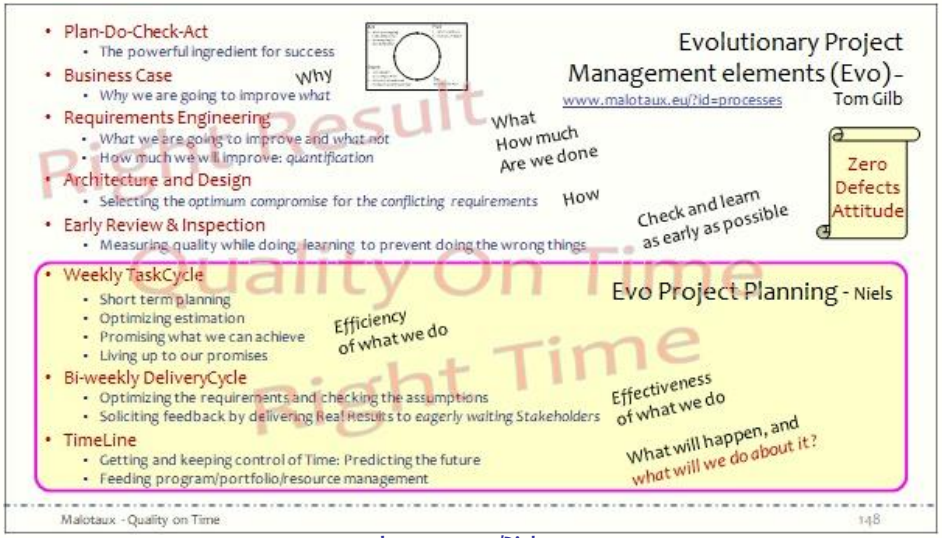
- What we deliver will be used by the appropriate users immediately, *within one week not making them less efficient than before*
- If a delivery isn't used immediately, we analyse and close the gap so that it will start being used (otherwise we don't get feedback)
- The proof of the pudding is when it's eaten and found tasty, *by them*, not just by us
- The users determine success, and whether they want to pay (we don't have to tell them, but it should be our attitude)
- Would you dare to deliver no-cure-no-pay ?

Some techniques shown

- Design
- Drawings
- DesignLog
- Review
- No Questions - No Issues

A *Zero Defects attitude* makes an immediate difference

Basic approach



www.malotaux.eu/?id=processes

- Improve the requirement
- Review
- Design implementation
- Review
- Implement (code ?)
- Review

Iterate fast, as needed

- Test doesn't find issues (because they're not there)

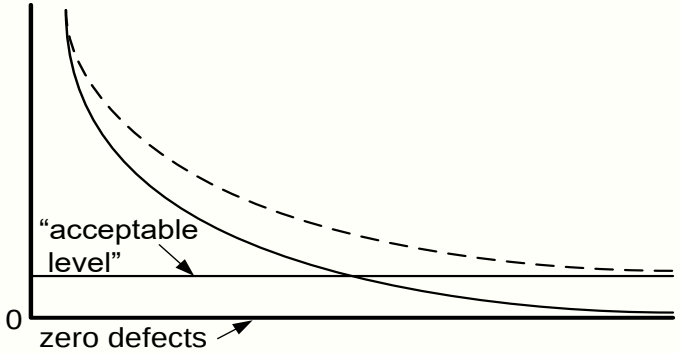
Do we deliver Zero Defect software ?

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 see issues repeated ?
- How much time is spent on defect prevention ?
- Could you use “No Questions – No Issues” ?



Approaching Zero Defects is Absolutely Possible

If in doubt, let's talk about it

Niels Malotaux

niels@malotaux.eu

www.malotaux.eu/conferences

More

Inquiries: niels@malotaux.eu

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 Behaviour in Projects (APCOSE 2008)
Human Behavioural 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

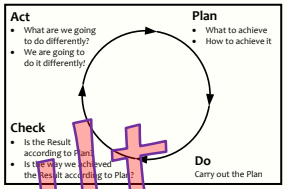
RS Measurable Value with Agile (Ryan Shriver - 2009)
Use of Evo Requirements and Prioritizing principles

www.malotaux.eu/inspections
Inspection pages

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
- 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



Why

What

How much

Are we done

How

Efficiency of what we do

Effectiveness of what we do

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

Check and learn as early as possible

Evolutionary Planning - Niels

Zero Defects Attitude

Right Result

Quality On Time

Right Time