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

www.malotaux.eu/conferences

# Ultimate Goal of a What We Do

Quality on Time

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

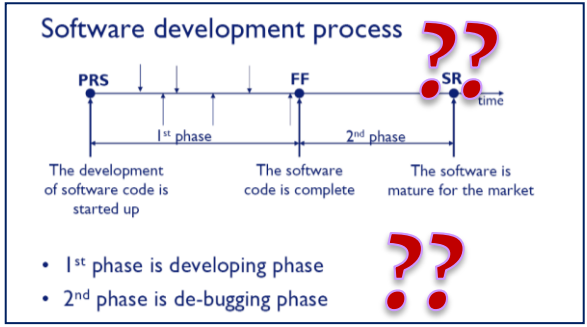
Providing the customer with

- what he needs
- at the time he needs it
- to be satisfied
- to be more successful than he was without it

Constrained by (win - win)

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

# Is software without defects possible ?




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

Better quality costs less

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 ?

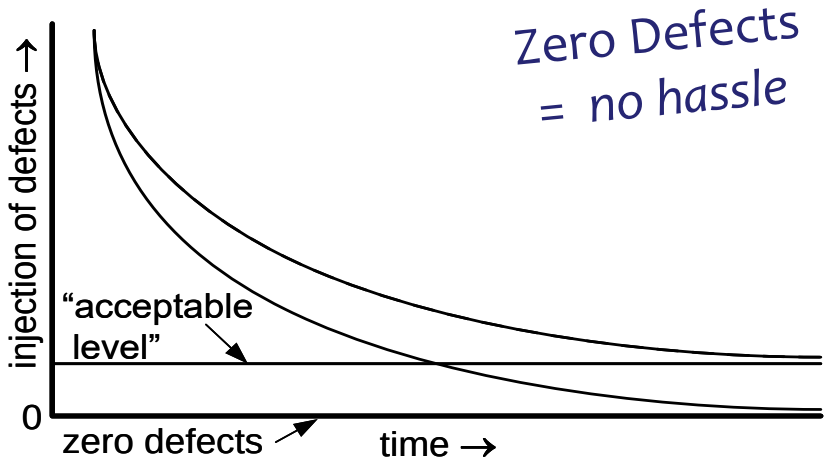
# What is a defect ?

- A defect is the cause of a problem experienced by any of the stakeholders while relying on our results
- Making the customer more successful implies *no defects*

- All we have to do is delivering results without defects
- Do we ?

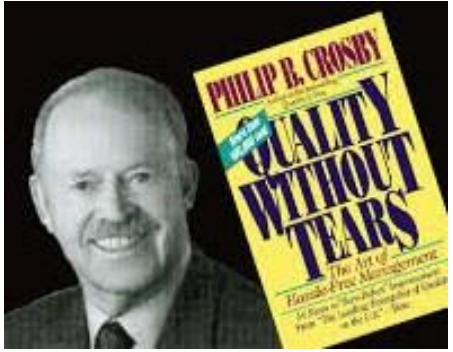
# What is Zero Defects

- Zero Defects is an *asymptote*




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

# 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)

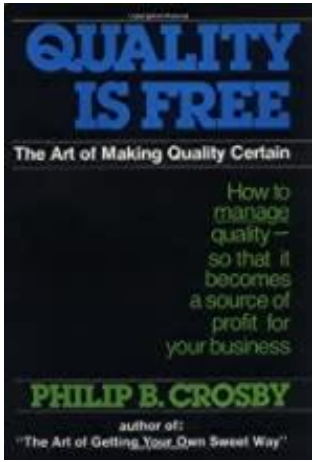
Philip Crosby, 1970

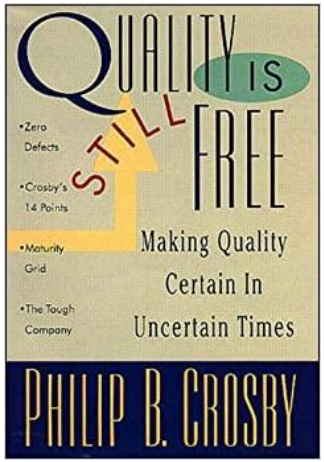

- The purpose is customer success (~~not~~ customer satisfaction)

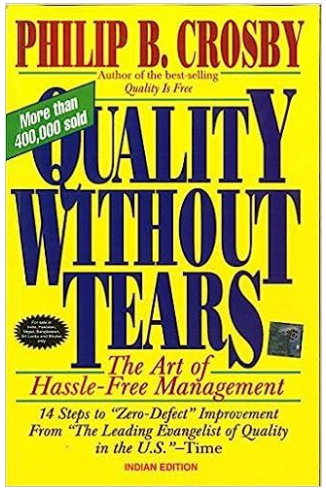
Added by Philip Crosby Associates, 2004

# Philip Crosby

[Quality is Still Free]



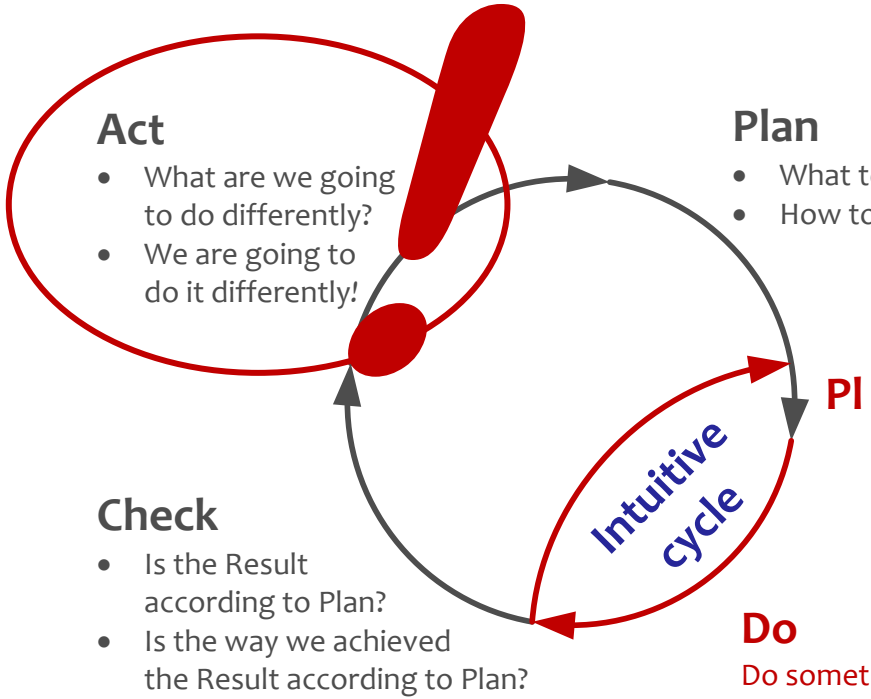



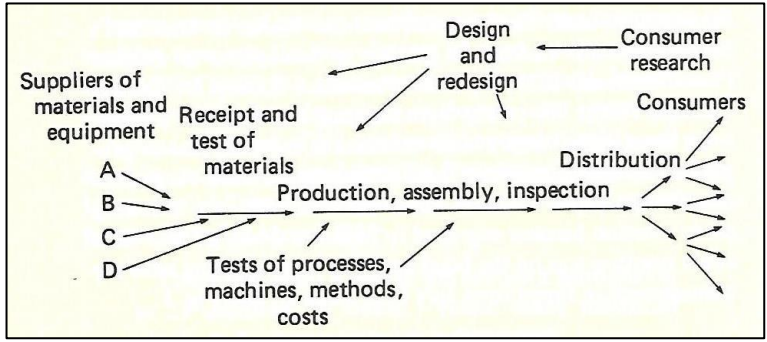




- Conventional wisdom says that error is inevitable
- As long as the performance standard requires it, this self-fulfilling prophecy will come true
- Most people will say: People are humans and humans make mistakes
- And people do make mistakes, particularly those who do not become upset when they happen
- 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

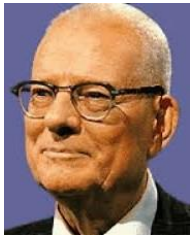
# The essential ingredient: the PDCA Cycle

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

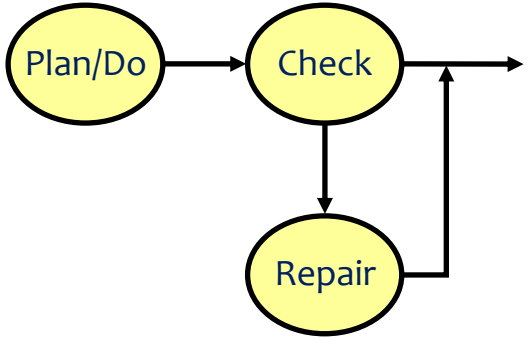


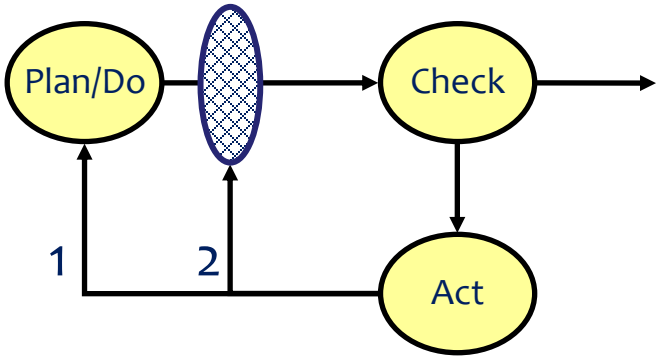

Deming: Out of the Crisis



# How can we prevent defects ?




What we often see

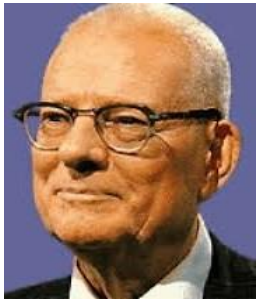



What we should expect

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

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

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



Deming
(1900-1993)

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

# Root Cause Analysis to feed prevention

- 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*

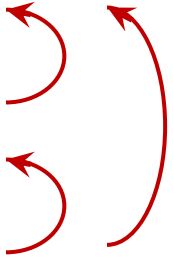
# Some Examples

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

# Design techniques

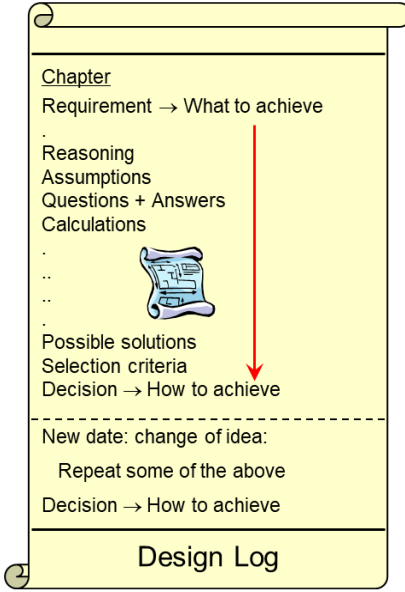
Cleanroom

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




# 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*

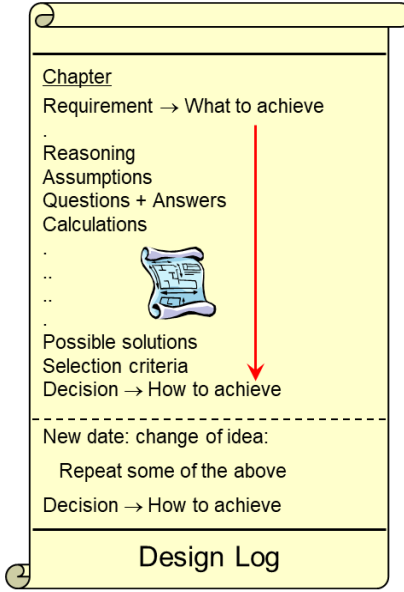
*"We don't have time !" - "We've only 7 days !"*

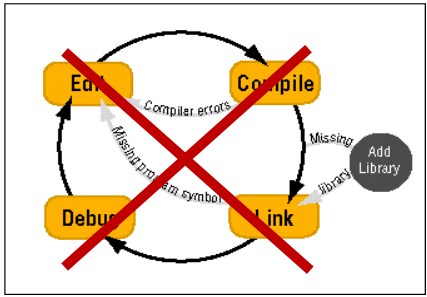
*James insisted*

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

*Done in less than 7 days*

*Thank you!*



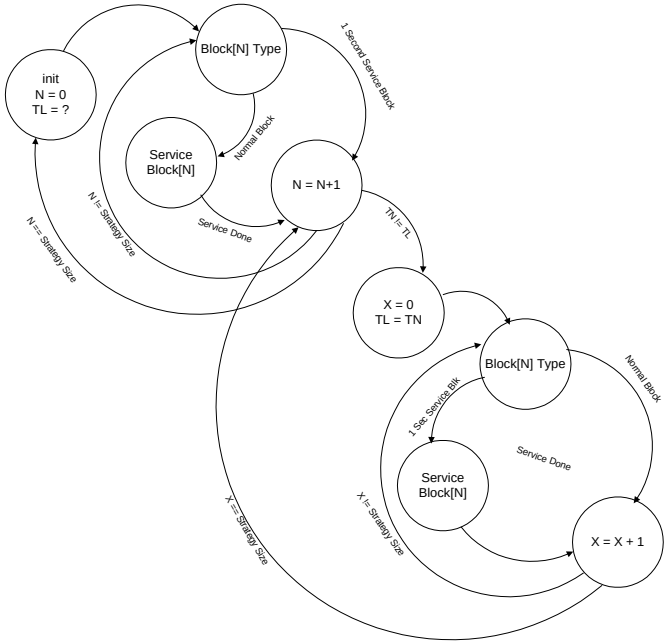



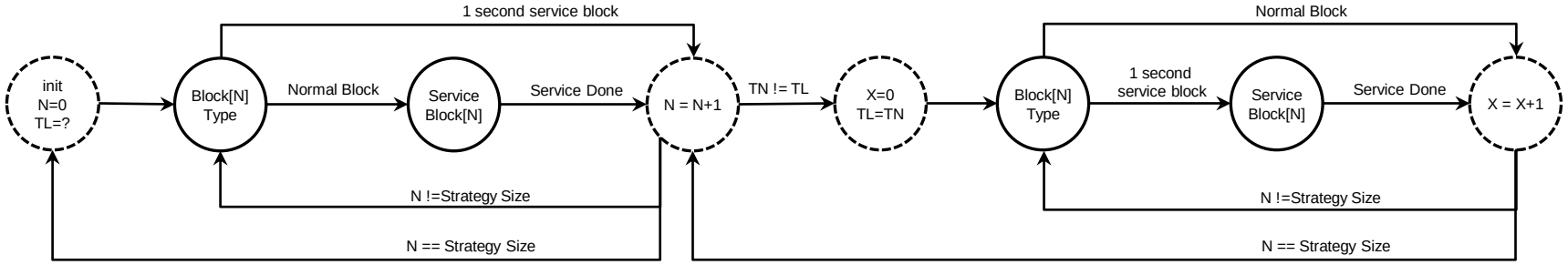
# What James told me afterwards

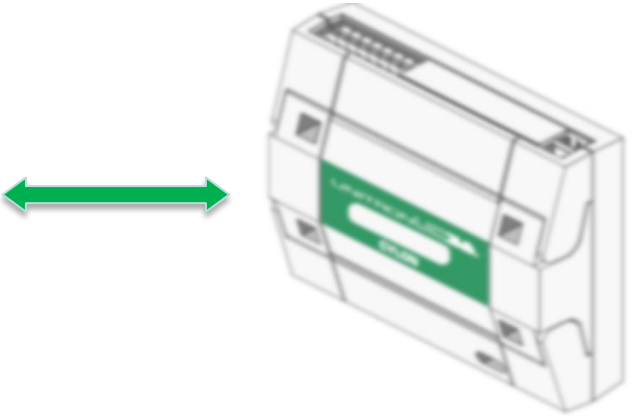
- I gave the design to two colleagues for review
- Louise corrected some minor issues
- It went into a ‘final’ review, with another colleague
- Based in his expertise, *the solution was completely reworked*
- Actually, two features were delivered and deployed
  - The one that was design and code reviewed 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

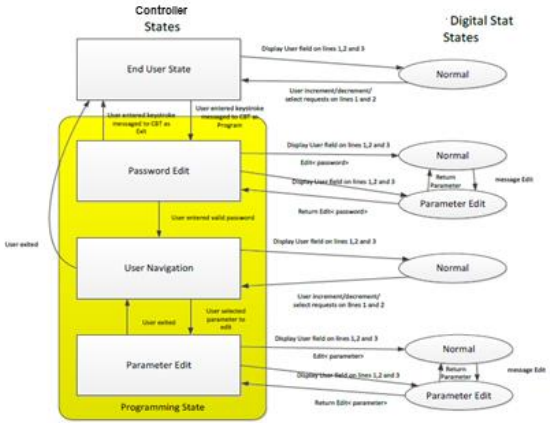
- Only few are useful
- Don't waste reviewer's time

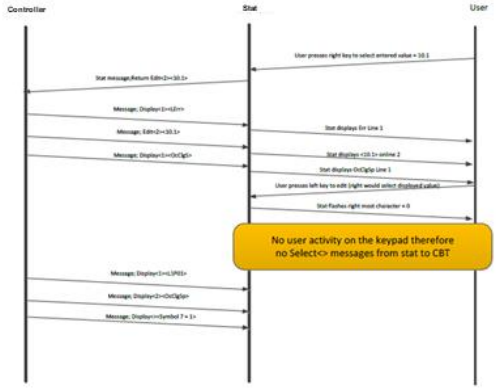




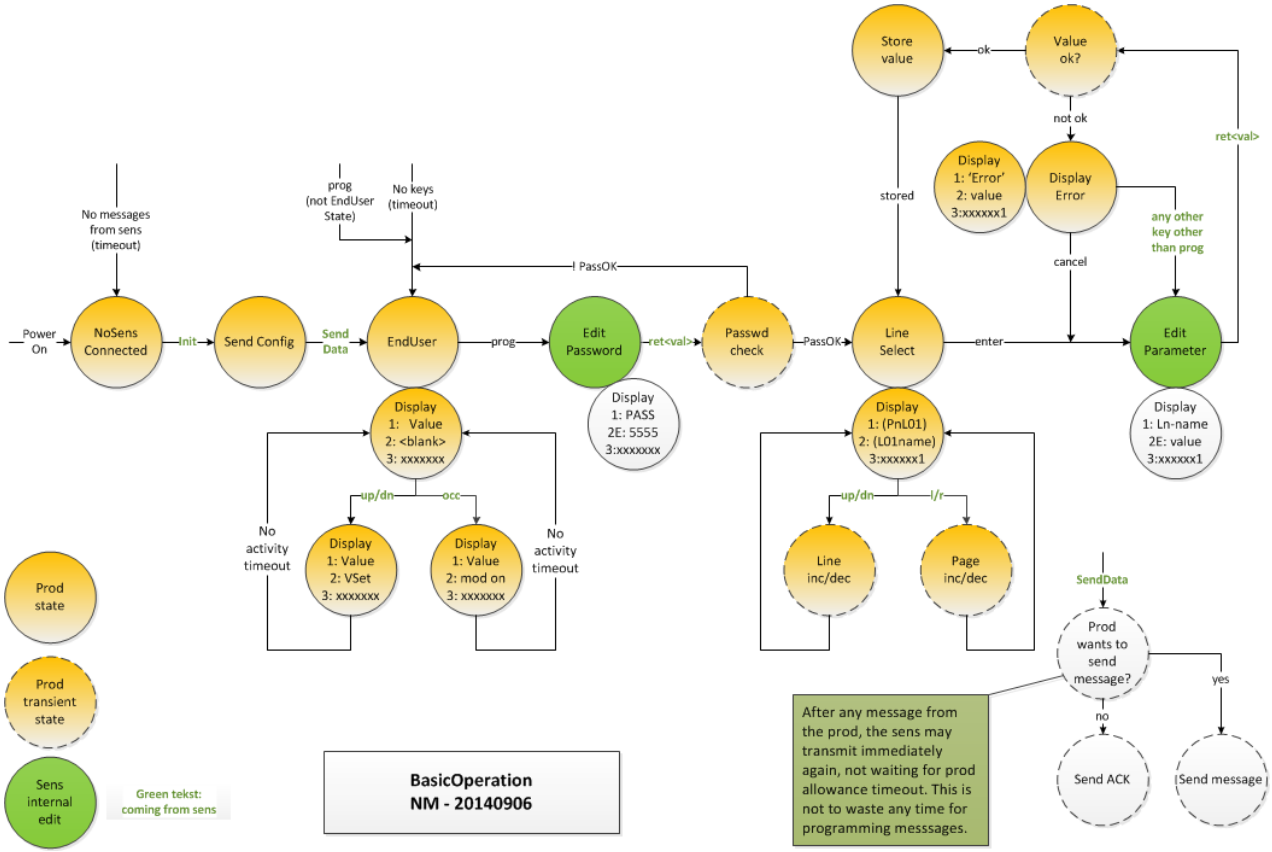
# Useful design ?



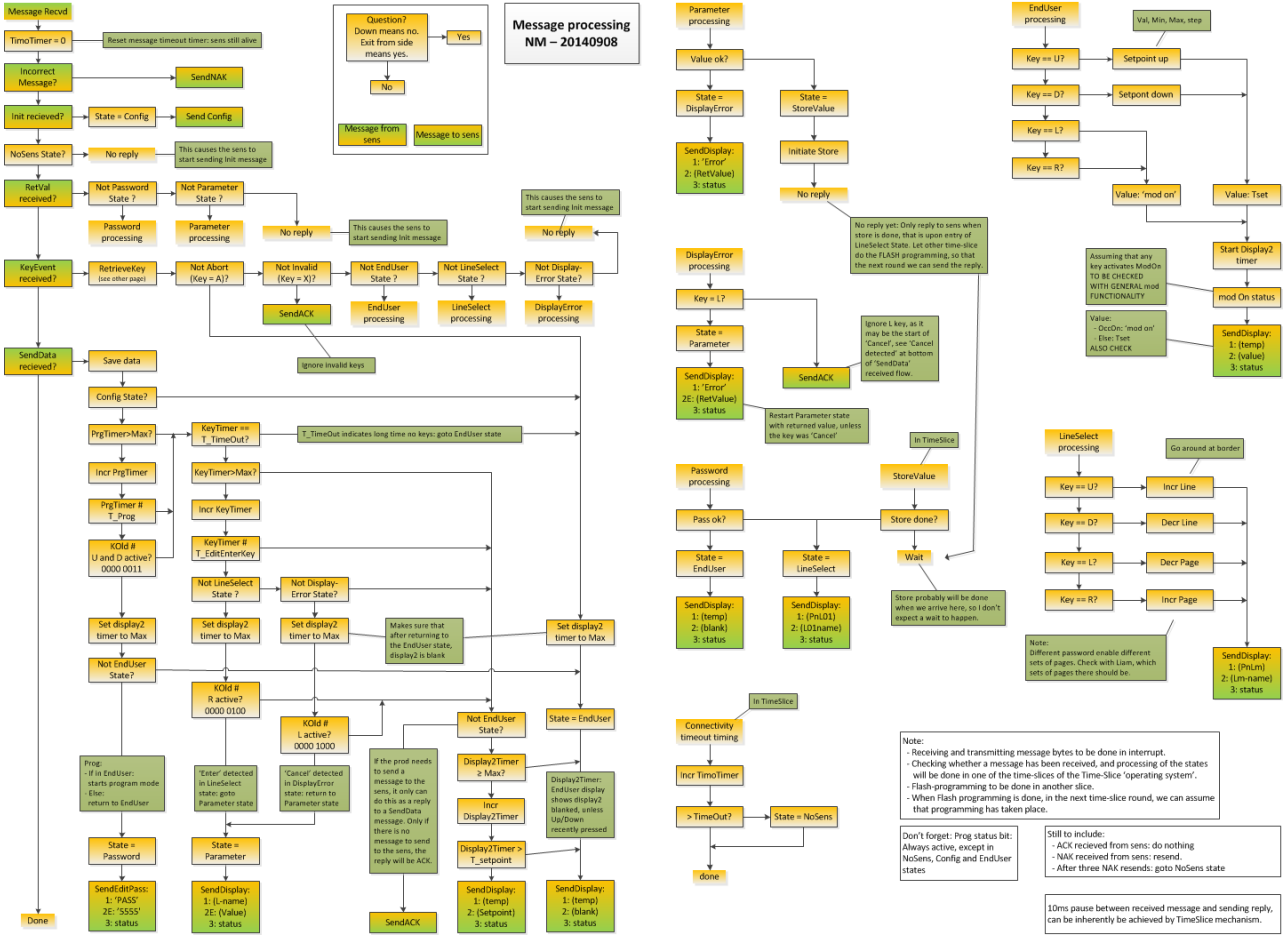



# Choose the appropriate design

47 pages documentation
condensed into one page



# How could it look like ?

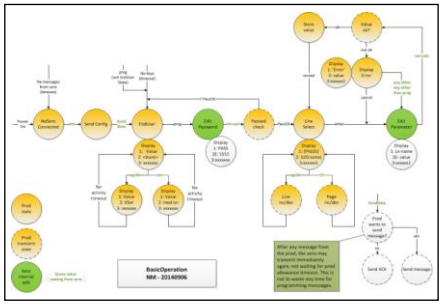


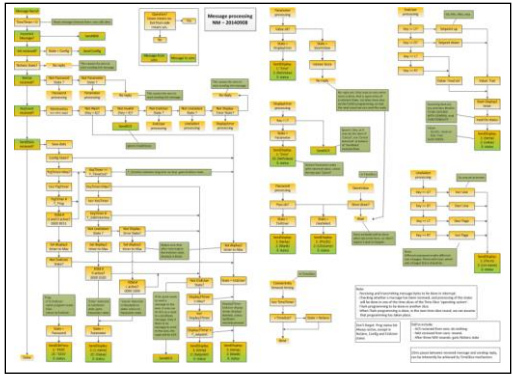
# What is better than reviewing code ?

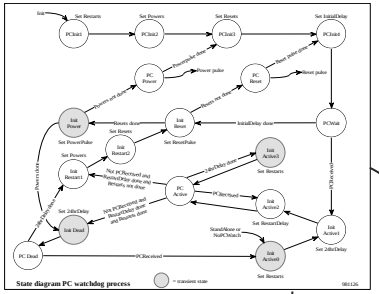
```
hc $contract
  # some keybindings like to change on different hooks.
herbstclient -il
while read line;do
  case $line in
    # remove the gap
    nogap ) herbstclient chain : set frame_gap -1 : \
      set window_gap ${window_gap--1} : keybind $Mod-D emit_hook "
      set frame_border_width ${frame_border_width-1}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

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

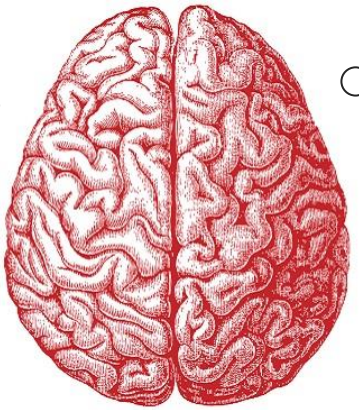
- Do you ever review software ?
- What do you review ?

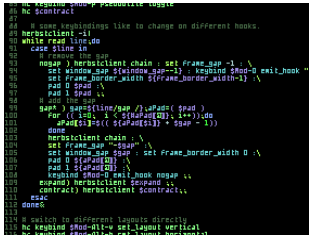
- What is better than reviewing code ?
  - May I review the design first ?

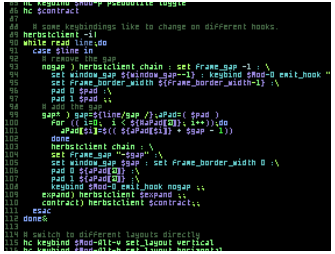


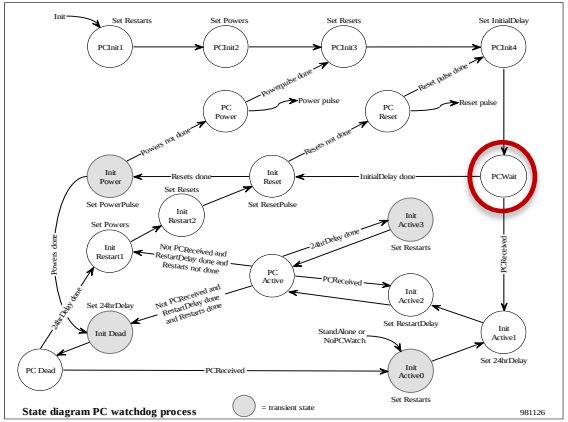


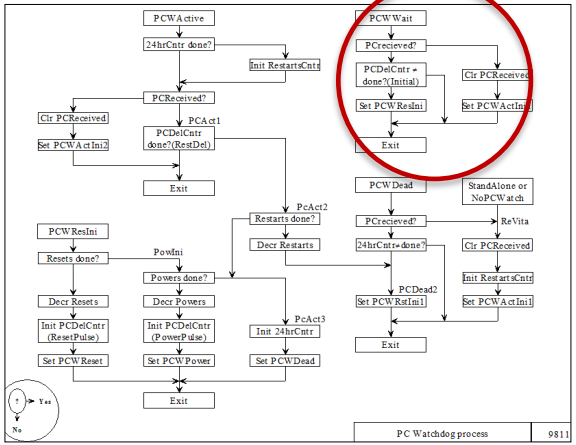

State diagram PC watchdog process
transient state










```
        MovLW   WaitPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Restart init1 PCW
;
PCRIn1  Call    EEtoPCP         ; Init powers counter
        MovLW   RIn2PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Restart init2 PCW
;
PCRIn2  Call    EEtoPCR         ; Init resets counter
        MovLW   ReInPC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Active init 1 PCW
;
PCAIn1  Call    Pre24h          ; Init 24h counter
        MovLW   AIn2PC          ; Select next phase
;       MovWF   PCPhase         ; (See EndPCX)
        Goto    EndPCX          ; Exit PC
;
;*****************************************************************
; Phase Wait PCW
;
PCWait  BTFSS   PCStat,PCRecvd  ; PC received?
        Goto    PCWait1         ; Branch if not
        BCF     PCStat,PCRecvd  ; Acknowledge PC received
        MovLW   AInlPC          ; 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, so 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
- How's that possible !!?
- They actually succeeded !

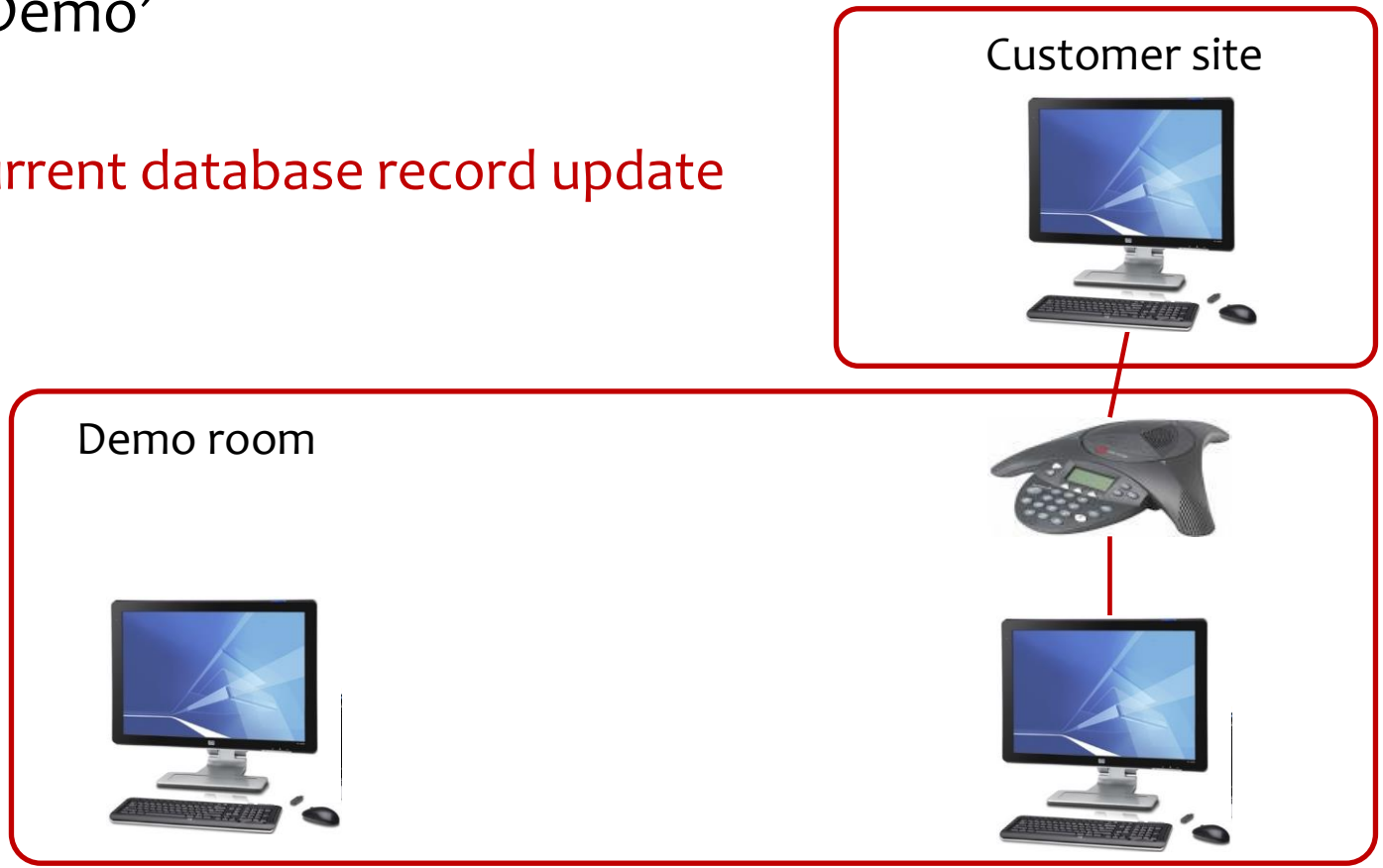
# Demo ??

- 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



# Delivery Strategy Suggestions (Requirements)

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

# Case: How much legwork is being done in your project ?

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

At the first delivery:

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

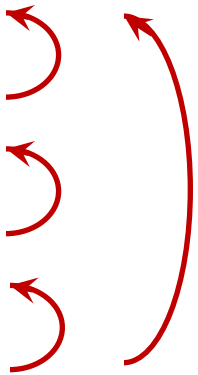
# Some techniques shown

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

A Zero Defects attitude makes an immediate difference

# Basic approach

- Design the requirement
- Review
- Design implementation
- Review
- Implement (code ?)
- Review
- Test doesn’t find issues (because they’re not there)



Iterate fast, as needed

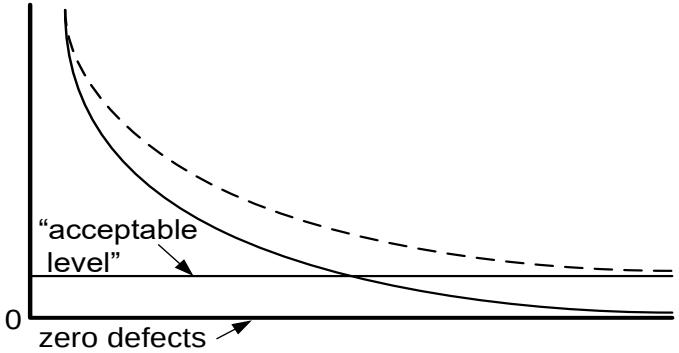
# Do we deliver Zero Defect software ?

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

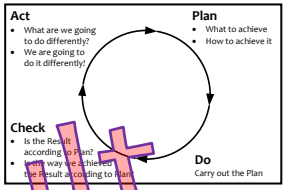
niels@malotaux.eu

www.malotaux.eu/conferences

# Evolutionary Project Management elements (Evo)

Tom Gilb

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




Why

What

How much

Are we done

How

Check and learn as early as possible

Zero Defects Attitude

Right Result

Quality On Time

Right Time

- 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

Evolutionary Planning - Niels

Efficiency of what we do

Effectiveness of what we do

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