Power to the Programmers !

Agile Change thru Software Engineering
Quantified Proven Real Best Practices

Geecon, Prague, 23 Oct 2015
@ImTomGilb
Tom at Gilb dot com
Gilb.com
These slides are at
http://www.gilb.com/dl821

(Gilb site Slides Downloads)
http://tinyurl.com/GilbGeecon
Has slides and my Agile papers
And historic papers, Raytheon , Mays DPP etc.

The Leader of the Revolution
Motto "Join or Die"

"Code **or**
***Create*,**
To determine your *fate*"

Thesis for this talk
Theme **High Level Objectives**

- If management decides on clear, quantified, improvement objectives (RARE EVENT!)
- Then the 'troops' can very effectively …
  - More effectively than any management led process ever reports in practice
- Deliver rapid effective and profitable improvement
  - In the direction of these quantified high level goals

Effective because

- Grass roots (developers) can change and measure often and early
  - Management cannot change and measure, early and often

Effective because

- Grass roots (developers) can change and measure often and early
  - Management cannot change and measure, early and often

"Because 'special cause' variation is 'assignable' (to a specific cause),
workers, supervisors or middle managers that have direct knowledge of the assignable cause,
best address this type of specific intervention."
(Deming interpretation, Wikipedia)

Sorry !
(not *really* ☺)

- I *like* fully filled-up, BUSY, DENSE, slides
- They reflect *my* reality: ***detailed facts***, like ‘code’
- If you don’t like dense slides
  - Close your eyes for the rest of this talk

- You can download my slides afterwards, and study them deeply,
  - when you feel more-technically receptive and motivated

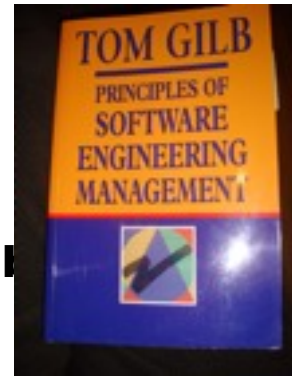
PS If you prefer very simple slides
and presentations
see https://www.youtube.com/watch?v=kOfK6rSLVTA
or Google: 'Tom Gilb TEDx'
Same talk as Oct 22 2015 Geecon

Confessions of a Coder

- **I was a programmer (1958-1978),**
  - **But I decided I wanted more power and influence**
    - **on the quality and usefulness of my work**
    - **I did not want to be part of the 50% totally failed IT projects**
    - **I wanted my projects to ALWAYS succeed**
  - **And I was tired of being told what to do b managers and users**
    - **Who did not strike me as blindingly savvy**
- **So I became a real** 'Software Engineer'
  - **I did not just change my 'title'**
  - **I really turned to ENGINEERING**

Agile Grandpa

- **The Agile 'Grandfather'**
  - **Practicing 'Agile' IT Projects since 1960**
  - **Preaching Agile since 1970's (Comp. Weekly UK)**
  - **Acknowledged Pioneer by Agile Gurus and Research**
    - **Beck, Sutherland, Highsmith, Cohn, Larman etc.**
    - **Ask me for details on this! I am too shy to show it here!**
- **Agile Practice**
  - **IT: for decades (Kai and Tom)**
  - **Organisations: for Decades (Citigroup, Intel, HP, Boeing)**
- **Books: Presenting Agile: Incremental Delivery**
  - **Principles of Software Engineering Management (1988) the B**
    Beck and others refer to.
  - **Competitive Engineering (2005)**
  - **'Evo': (Kai, evolving, 55 iterations)**
  - **1976 Software Metrics book**
    - **As detailed in 1988 PoSEM citations**
  - **NEW 'Competitive Planning' manuscript**
  - **http://tinyurl.com/competitiveplanning**



OK I am not that shy!

Agile References:

"Tom Gilb invented Evo, arguably the first Agile process. He and his son Kai have been working with me in Norway to align what they are doing with Scrum.

Kai has some excellent case studies where he has acted as Product Owner. He has done some of the most innovative things I have seen in the Scrum community."

Jeff Sutherland, co-inventor of Scrum, 5Feb 2010 in Scrum Alliance Email.

"Tom Gilb's Planguage referenced and praised at #scrumgathering by Jeff Sutherland. I highly agree" Mike Cohn, Tweet, Oct 19 2009

"I've always considered Tom to have been the original agilist. In 1989, he wrote about short iterations (each should be no more than 2% of the total project schedule). This was long before the rest of us had it figured out." Mike Cohn http://blog.mountaingoatsoftware.com/?p=77

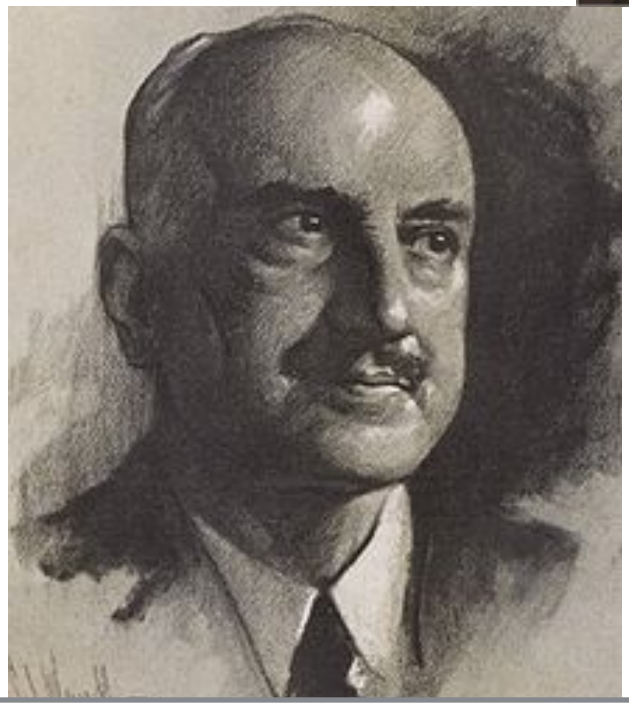
Comment of Kent Beck on Tom Gilb's book , "Principles of Software Engineering Management": " A strong case for evolutionary delivery – small releases, constant refactoring, intense dialog with the customer". (Beck, page 173).

In a mail to Tom, Kent wrote: "I'm glad you and I have some alignment of ideas. I stole enough of yours that I'd be disappointed if we didn't :-), Kent" (2003)

Jim Highsmith (an Agile Manifesto signatory) commented: "Two individuals in particular pioneered the evolution of iterative development approached in the 1980's – Barry Boehm with his Spiral Model and Tom Gilb with his Evo model. I drew on Boehm's and Gilb's ideas for early inspiration in developing Adaptive Software Development. …. Gilb has long advocated this more explicit (quantitative) valuation in order to capture the early value and increase ROI" (Cutter It Journal: The Journal of Information Technology Management, July 2004page 4, July 2004).

Will we never learn ?

- **“Those who cannot remember the past are condemned to repeat it.”**
- ***The Life of Reason (1905-1906)***
  - Vol. I, ***Reason in Common Sense***



Jorge Agustín Nicolás Ruiz de Santayana y Borrás,
known as **George Santayana**
(December 16, 1863 – September 26, 1952),
was a philosopher, essayist, poet, and novelist.

Grandpa Guru Tom Speaks

- **I am your historian.**
- **I joined IBM in 1958**
- **And lived intensively through the entire computer age**
- **I'll tell you what I have learned, before I go.**
- **But this might be your last chance.**
- **You, and your teachers, have missed all other such opportunities up to now ….**
- **Are YOU doomed to repeat the errors of the software past?**

Basic ideas: of this talk

- **Power to the Programmers**
  - **Delegation of power to programmers is a smart idea.**
  - **It is provably and measurably smarter than**
    - **leaving the power with**
  - **managers (BOO !)**
    - **to design the developer's own work environment, and**
  - **with IT architects (BOO !) to design the technology,**
    - **that we are then told to code.**
  - **Delegating the power to DEVELOPERS (YESSSS !) ,**
    - **to create a better working-environment,**
    - **and to design the technology for our stakeholders,**
    - **is better - because**
      - **developers are closer to the action,**
      - **are more informed in practical detail;**
      - **and they can rapidly and frequently, test and measure, that their ideas really work.**

Tom, telling 300 IT Architects that they are ridiculous, incompetent, immature, embarrassing, and pompous (diplomatically, of course!)

VIDEO = http://vimeo.com/user22258446/review/79092608/600e7bd650

How?

- Make *developers* **responsible**
  - **for delivery of the 'quantified' critical requirements**
    - (Performance, Qualities, cost, deadline)
- Give them the **freedom to decide the right designs**
  - With immediate responsibility to *measure* that they are delivering the results
- Get the 'unprofessional' users and **customers 'off their backs'**
  - Avoid receiving features and stories
    - which are usually amateur design, by people who have no overview or responsibility or design ability (users and customers, and managers)
- **Elevate your talent** by becoming a **real 'software ENGINEER'**
  - With coding-expert craftsmanship, as your basic talent

Cases: Raytheon and IBM
use 'Defect prevention Process'
('DPP',= CMM Level 5) to
EMPOWER DEVELOPERS
TO *RADICALLY CHANGE* THEIR OWN *WORK ENVIRONMENT*

- 

Designing Your Own Organization ?

CTO

- Architect
- **Standards**
- Audit
- Project Management

WE decide our fate

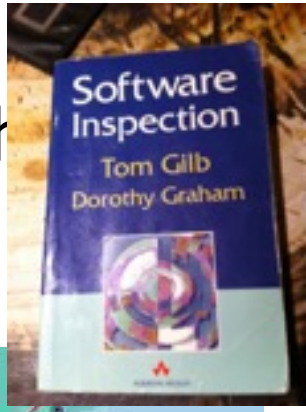
1. Identify major defects With bad costs
2. Find Common Cause of Defects
3. Find and do Change to eliminate common cause
4. Results Bugs, Productivity, Cost
5. Spread to the larger organization

Background 1970-1980 MANAGERS FAIL

- **Michael Fagan and Ron Radice co-invent 'Software Inspection'**
  - **The intent was to collect data on bugs and defects**
  - **Use it to find frequent common causes**
  - **To improve development processes**
  - **The attitude was explicitly**
    - **'managers should manage' (MEF to TsG)**
  - **THEY FAILED TO GET REAL PROCESS IMPROVEMENT**

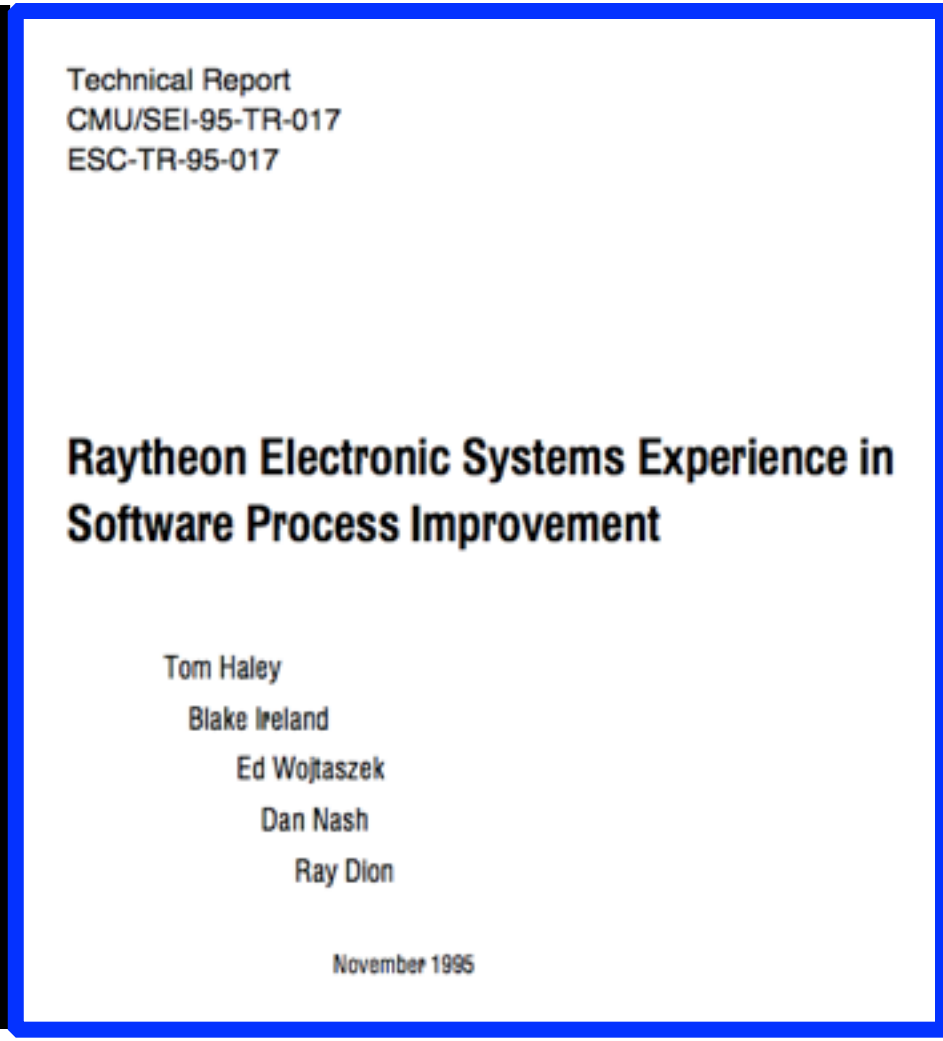
1980
The 'Troops' succeed, where the Generals Failed

- Robert Mays and Carol L. Jones, at IBM Research Triangle Park, NC
- Invent 'Defect Prevention Process' → Ch
- Major idea:
  - Delegate power to devs to
    - Analyze their OWN defects
    - And fix their OWN process
- THAT *WORKED*



Software Process Improvement at Raytheon

- Source : Raytheon Report 1995
  - http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=12403 (this is a header to the download) Tested May 2014
  - Search "Dion & Raytheon" (Dion is Florida retired in 2014)
  - http://resources.sei.cmu.edu/asset_files/TechnicalReport/1995_005_001_16415.pdf
- An excellent example of process improvement driven by measurement of improvement
- Main Motor:
  - "Document Inspection", Defect Detection
- Main Driver:
  - "Defect Prevention Process" (DPP)



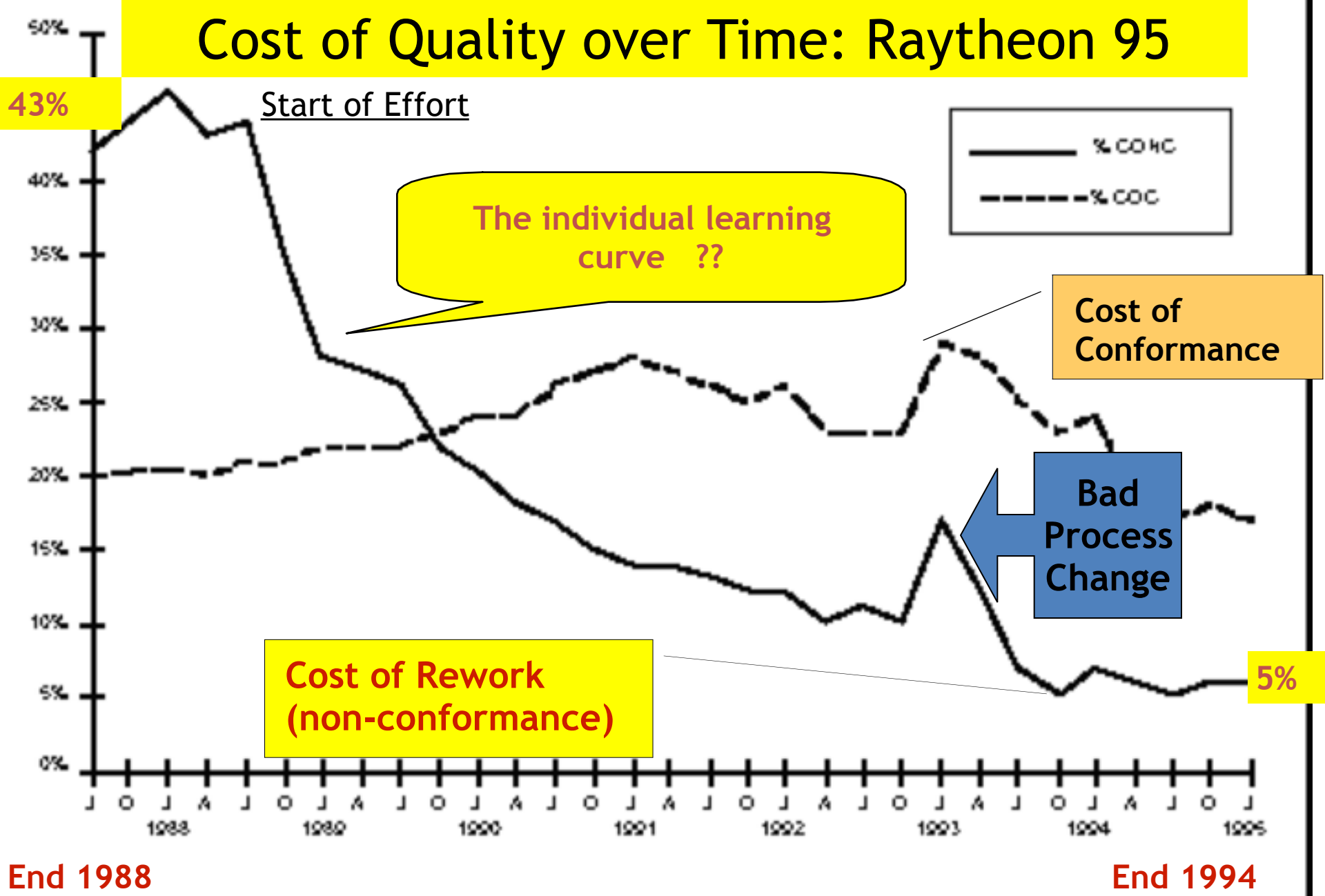
Technical Report
CMU/SEI-95-TR-017
ESC-TR-95-017

Raytheon Electronic Systems Experience in Software Process Improvement

Tom Haley
Blake Ireland
Ed Wojtaszek
Dan Nash
Ray Dion

November 1995

Cost of Quality over Time: Raytheon 95



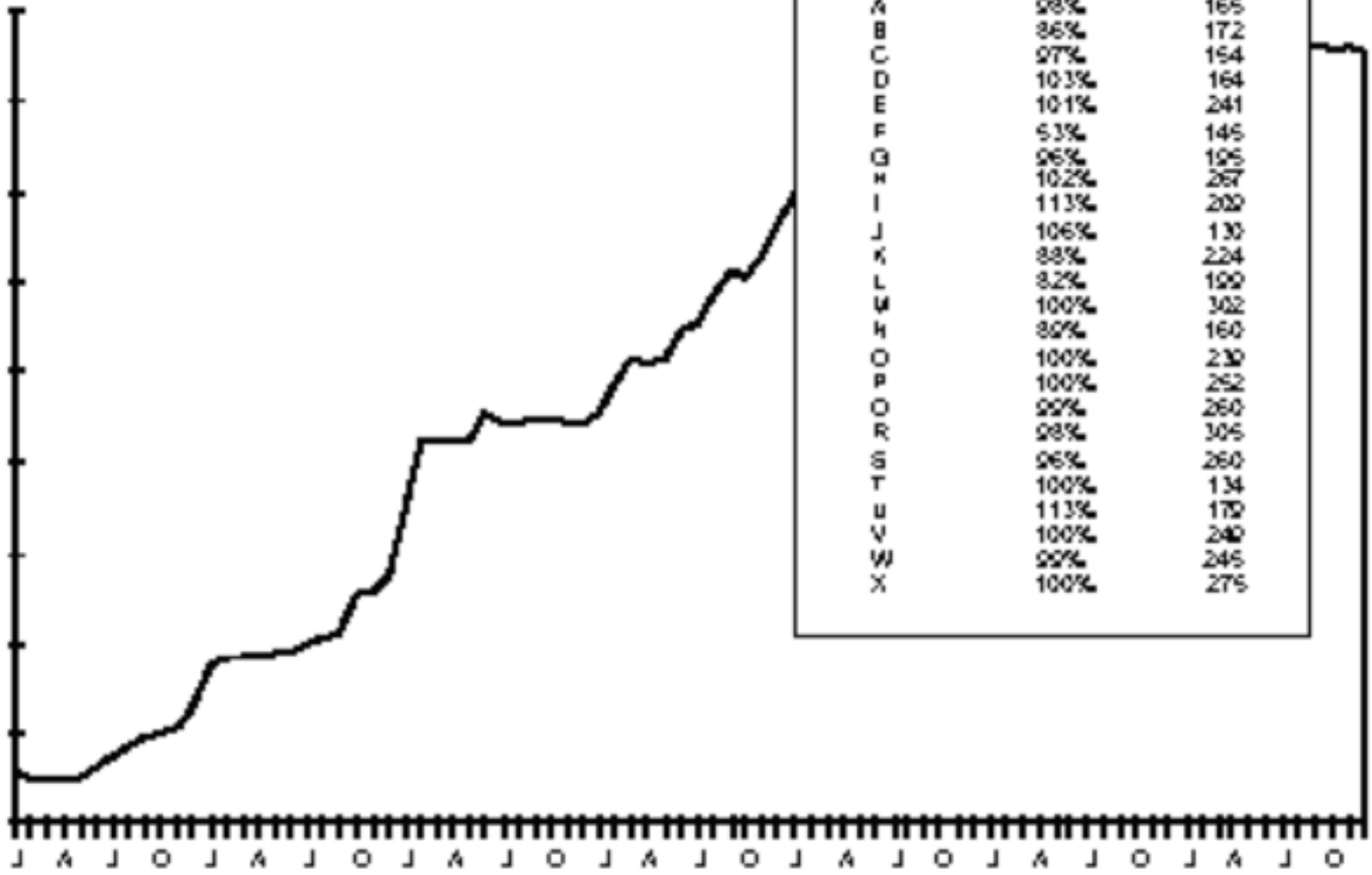
End 1988

End 1994

Figure 8: Cost of Quality Versus Time

Raytheon 95 Software Productivity 2.7X better

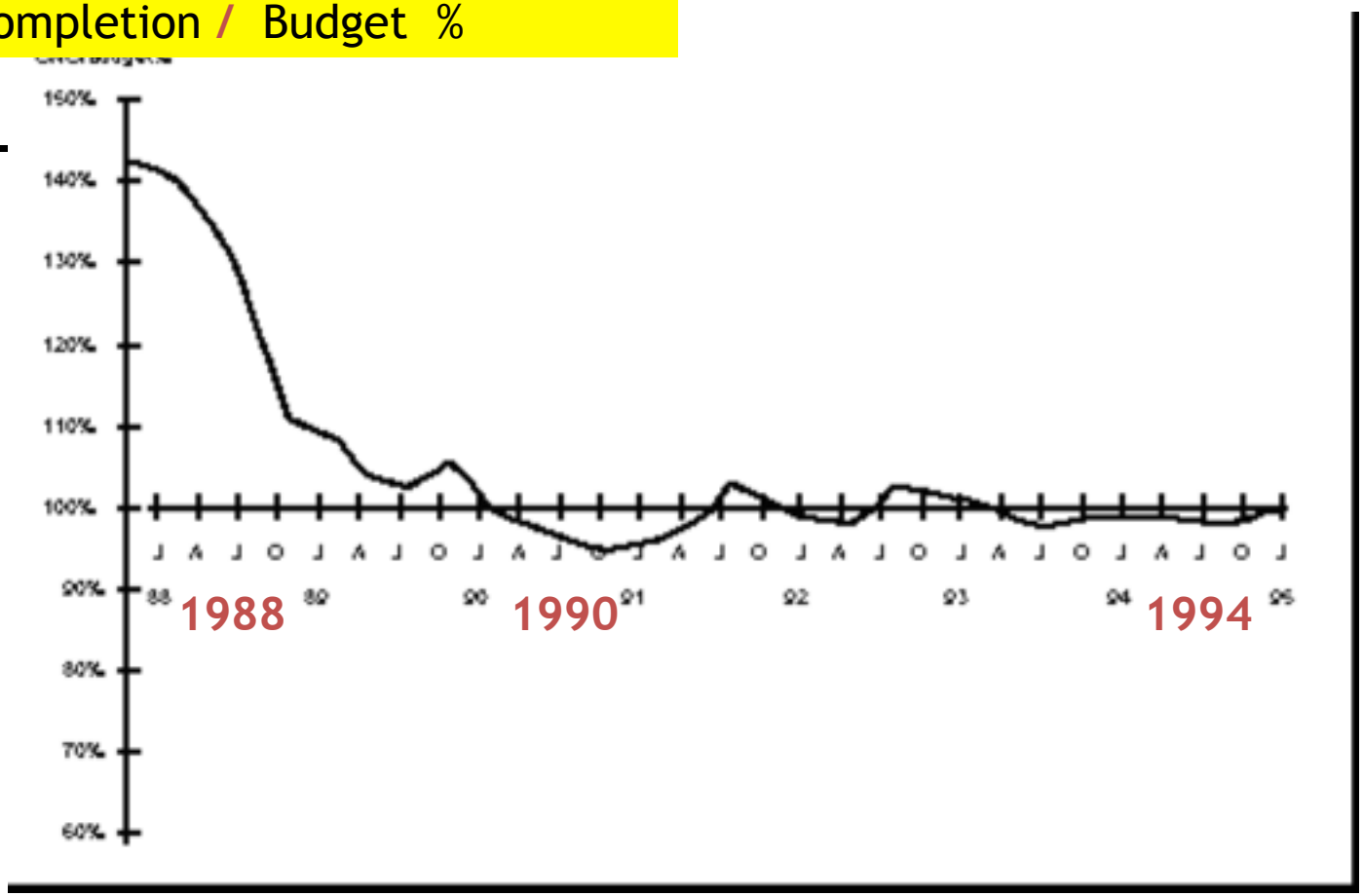
Productivity



Project	CAC/BuD	Productivity
A	98%	165
B	86%	172
C	97%	164
D	103%	164
E	101%	241
F	93%	146
G	96%	195
H	102%	267
I	113%	202
J	106%	130
K	88%	224
L	82%	199
M	100%	302
N	89%	160
O	100%	239
P	100%	252
Q	99%	260
R	98%	305
S	96%	260
T	100%	134
U	113%	170
V	100%	240
W	99%	245
X	100%	275

Achieving Project Predictability: Raytheon 95

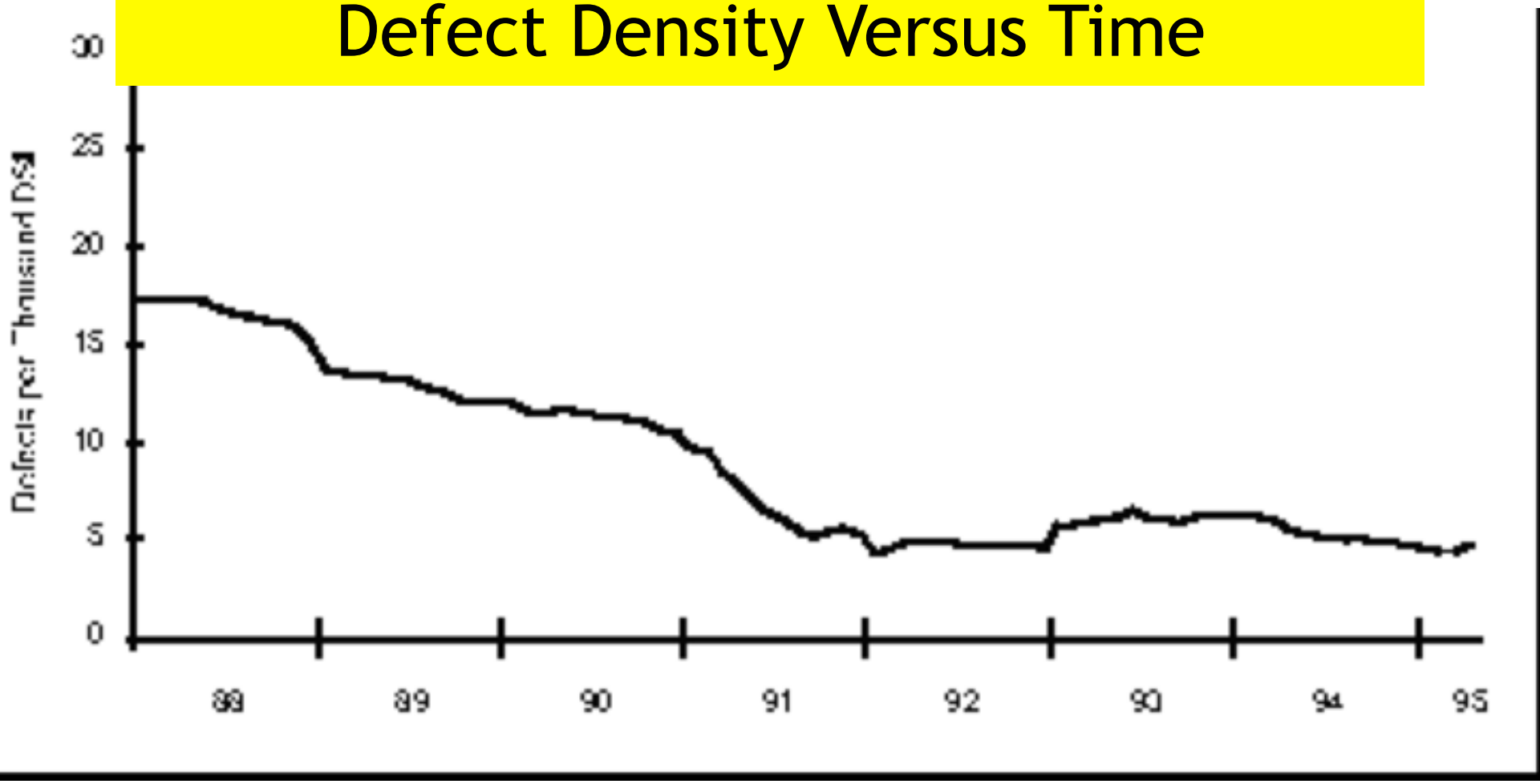
Cost At Completion / Budget %



Examples of Process Improvements: Raytheon 95

- **Process Improvements Made**
- **Erroneous interfaces during integration and test** -
  - Increased the detail required for interface design during the requirements analysis phase and preliminary design phase - Increased thoroughness of inspections of interface specifications
- **Lack of regression test repeatability -**
  - Automated testing - Standardized the tool set for automated testing - Increased frequency of regression testing
- **Inconsistent inspection process -**
  - Established control limits that are monitored by project teams - Trained project teams in the use of statistical process control - Continually analyze the inspection data for trends at the organisation level
- **Late requirements up-dates** -
  - Improved the tool set for maintaining requirements traceability - Confirm the requirements mapping at each process phase
- **Unplanned growth of functionality during Requirements Analysis**
  - - Improved the monitoring of the evolving specifications against the customer baseline - Continually map the requirements to the functional proposal baseline to identify changes in addition to the passive monitoring of code growth - Improved requirements, design, cost, and schedule tradeoffs to reduce impacts

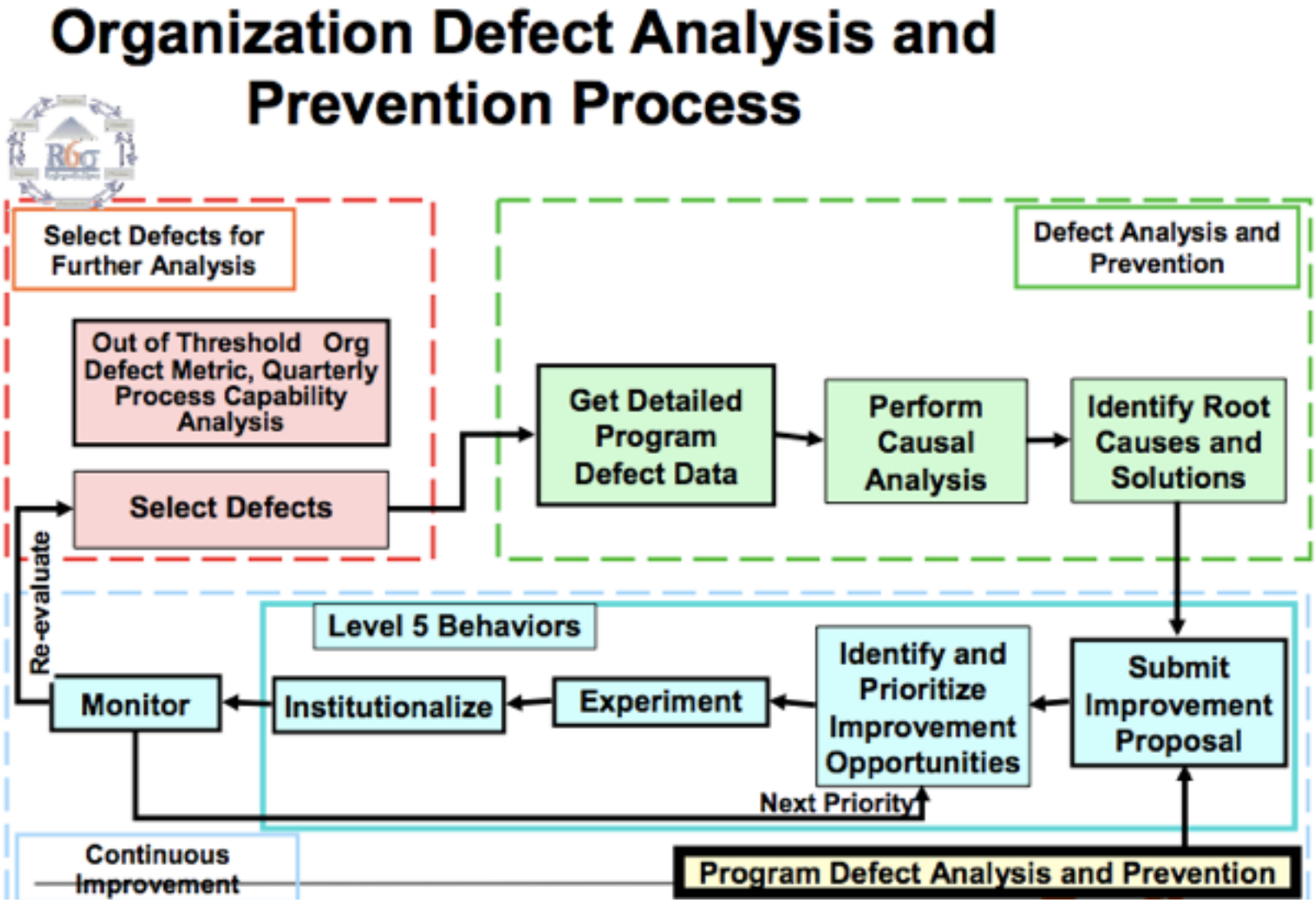
Overall Product Quality: Raytheon 95 (Bug density going down by 3:1) Defect Density Versus Time



Return On Investment

- $7.70 per $1 invested at Raytheon
- Sell your improvement program to top management on this basis
- Set a concrete target for it
  - PLAN [Our Division, 2 years hence] 8 to 1

The DPP Process



What's Going on Here?

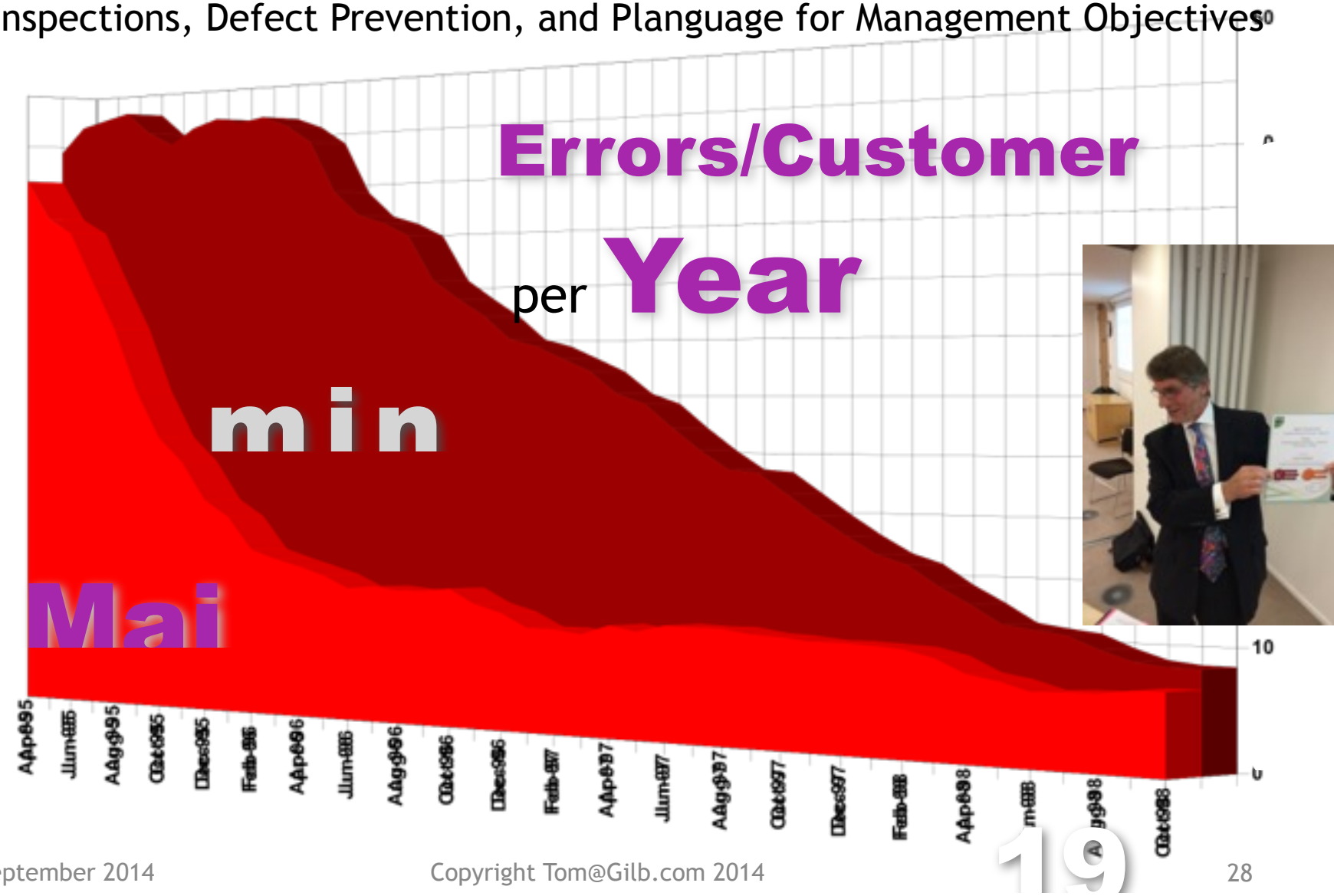
- 1,000 programmers
  - Later joined by 1,000 merged new programmers
  - Are
    - Analyzing their **own** bugs and spec defects
    - Suggesting their **own** work environment changes
    - And reducing their 43% rework by 10 X
- Power has been delegated to the programmers

Improving the *Reliability* Attribute

Primark, London (Gilb Client)

see case study Dick Holland, "Agent of Change" from Gilb.com

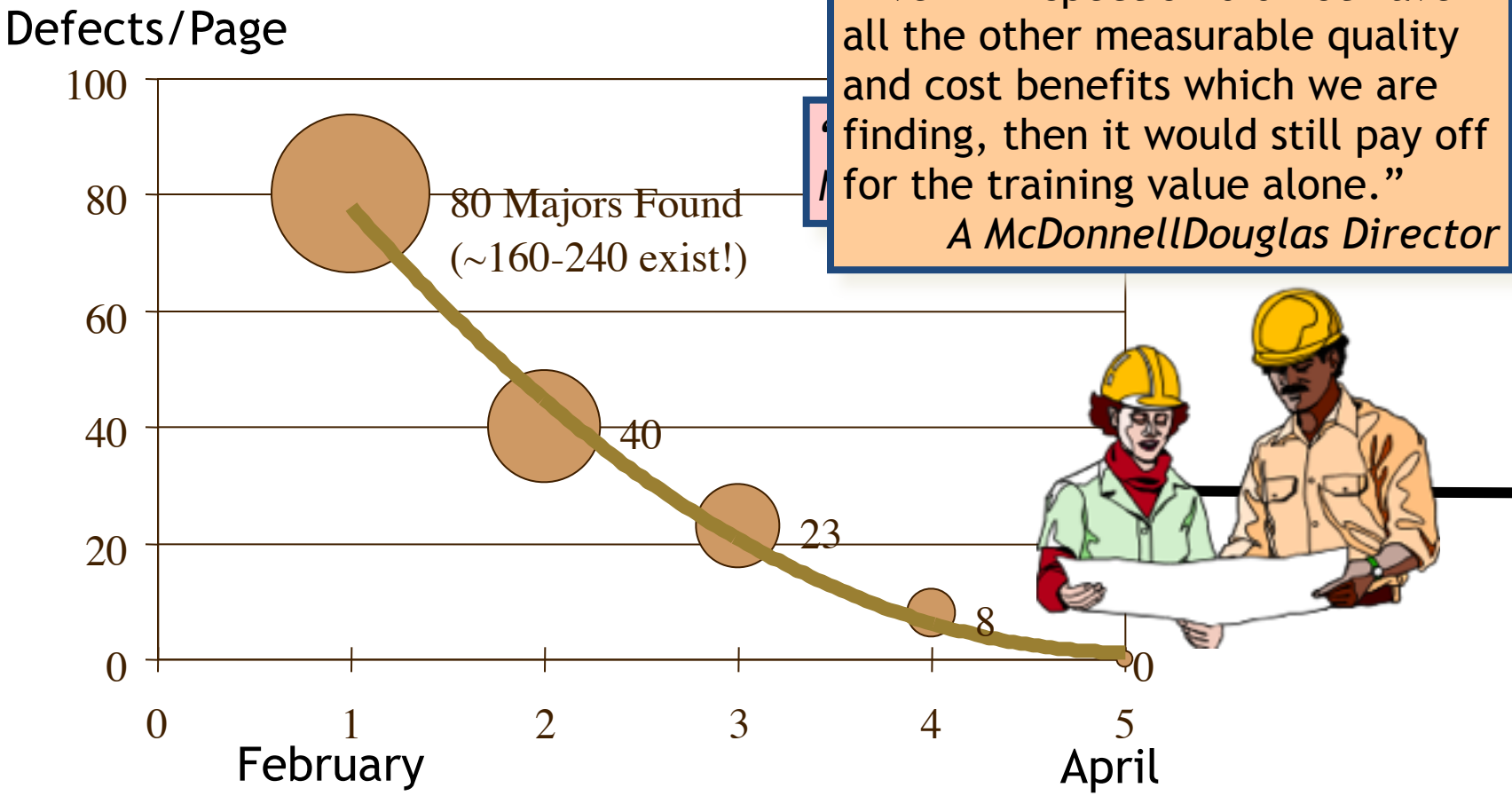
Using, Inspections, Defect Prevention, and Planguage for Management Objectives



Positive Motivation
Personal Improvement

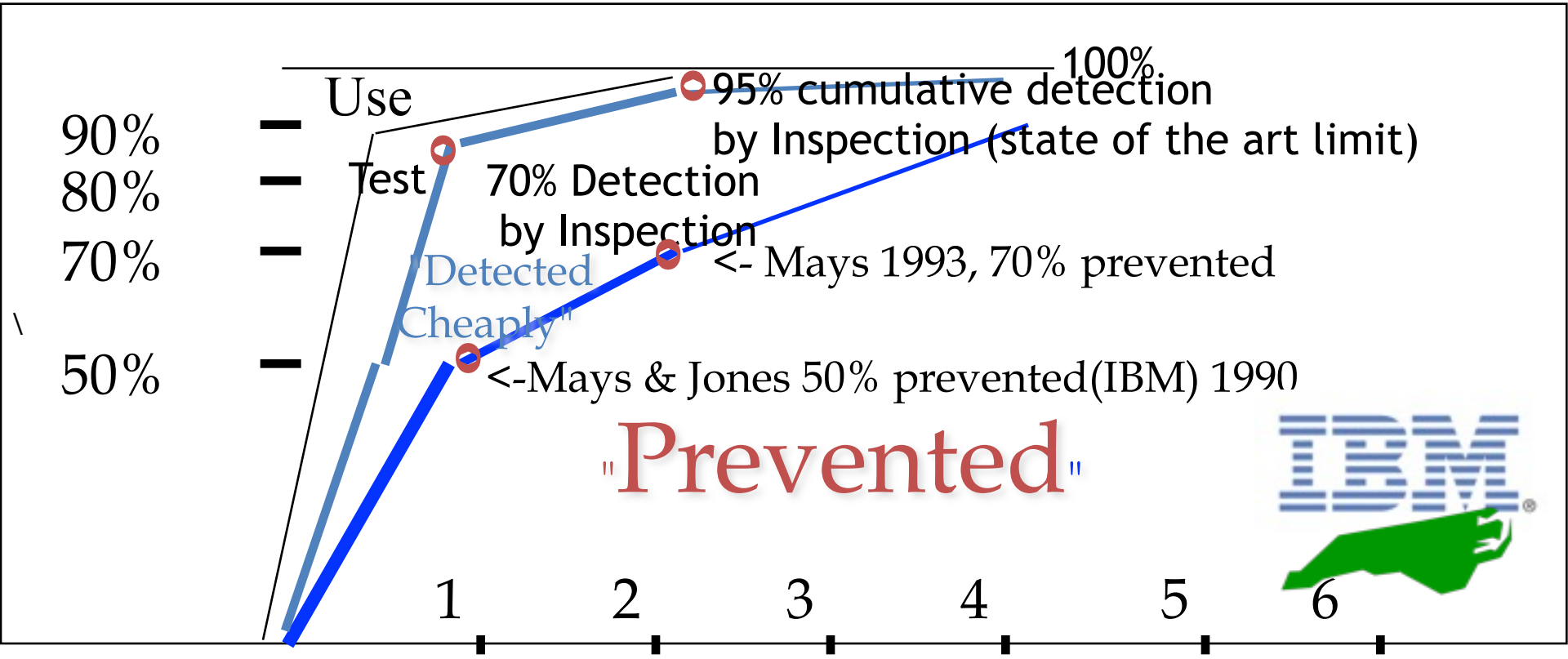
"We find an hour of doing Inspection is worth ten hours of company classroom training."
A McDonnell-Douglas line manager

"Even if Inspection did not have all the other measurable quality and cost benefits which we are finding, then it would still pay off for the training value alone."
A McDonnellDouglas Director



Inspections of Gary's Designs

Prevention + Pre-test Detection is the most effective and efficient



- Prevention data based on state of the art prevention experiences (IBM RTP), Others (Space Shuttle IBM SJ 1-95) 95%+ (99.99% in Fixes)
- Cumulative Inspection detection data based on state of the art Inspection (in an environment where prevention is also being used, IBM MN, Sema UK, IBM UK)

IBM MN & NC DP Experience

- **2162 DPP Actions implemented**
  - between Dec. 91 and May 1993 (30 months)<-Kan
- RTP about 182 per year for 200 people.<-Mays 1995
  - 1822 suggested ten years (85-94)
  - 175 test related
- RTP 227 person org<- Mays slides
  - 130 actions (@ 0.5 work-years
  - 34 causal analysis meetings @ 0.2 work-years
  - 19 action team meetings @ 0.1work-years
  - Kickoff meeting @ 0.1 work-years
  - TOTAL costs 1% of org. resources

- ROI DPP 10:1 to 13:1, internal 2:1 to 3:1
- Defect Rates at all stages 50% lower with DPP

The ICL Bill of Rights for Company Communication (by TsG)

1. You have a right to know precisely what is expected of you.

2. You have a right to clarify things with colleagues, anywhere in the organization.

3. You have a right to initiate clearer definitions of objectives and strategies.

4. You have a right to get objectives presented in measurable, quantified formats.

5. You have a right to change your objectives and strategies, for better performance.

6. You have the right to try out new ideas for improving communication.

007. You have the right to fail when trying, but also to kill failures quickly.

8. You have a right to constructively challenge higher-level objectives and strategies.

9. You have a right to be judged objectively on your performance against measurable objectives.

10. You have a right to offer constructive help to colleagues to improve communication.

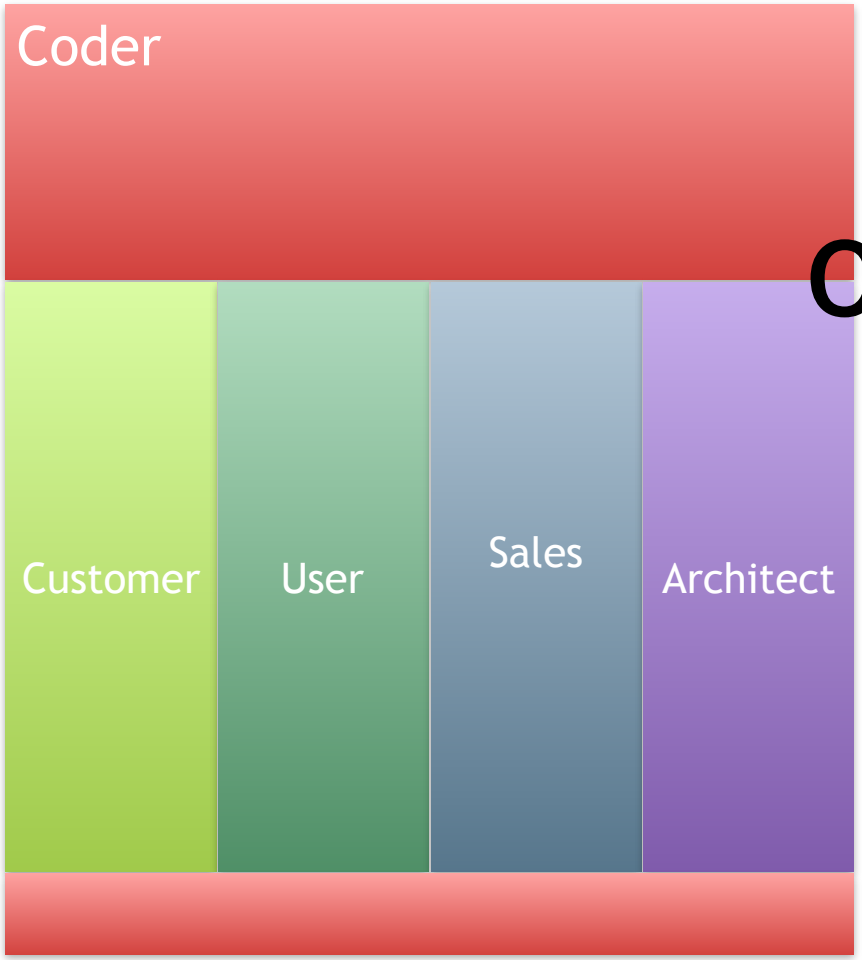
Summary DPP
Managers: 0 Devs : 1

- Developers are *better* at managing their own work environment, than their managers are
- 'Directors' should NOT design the work environment
- Developers should 'evolve the environment'
  - through practical deep personal insights,
  - and take responsibility for their own work situation

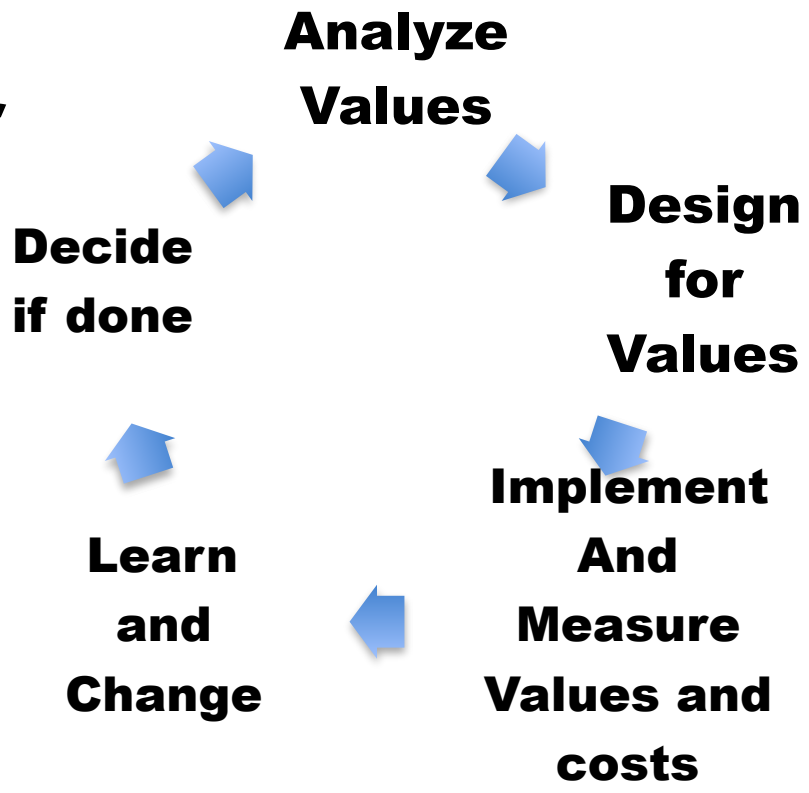
Case: Delegating Software product design to the Developers

"In the interest of overcoming my reluctance to delegate, starting Monday I want you to do all of my worrying for me."

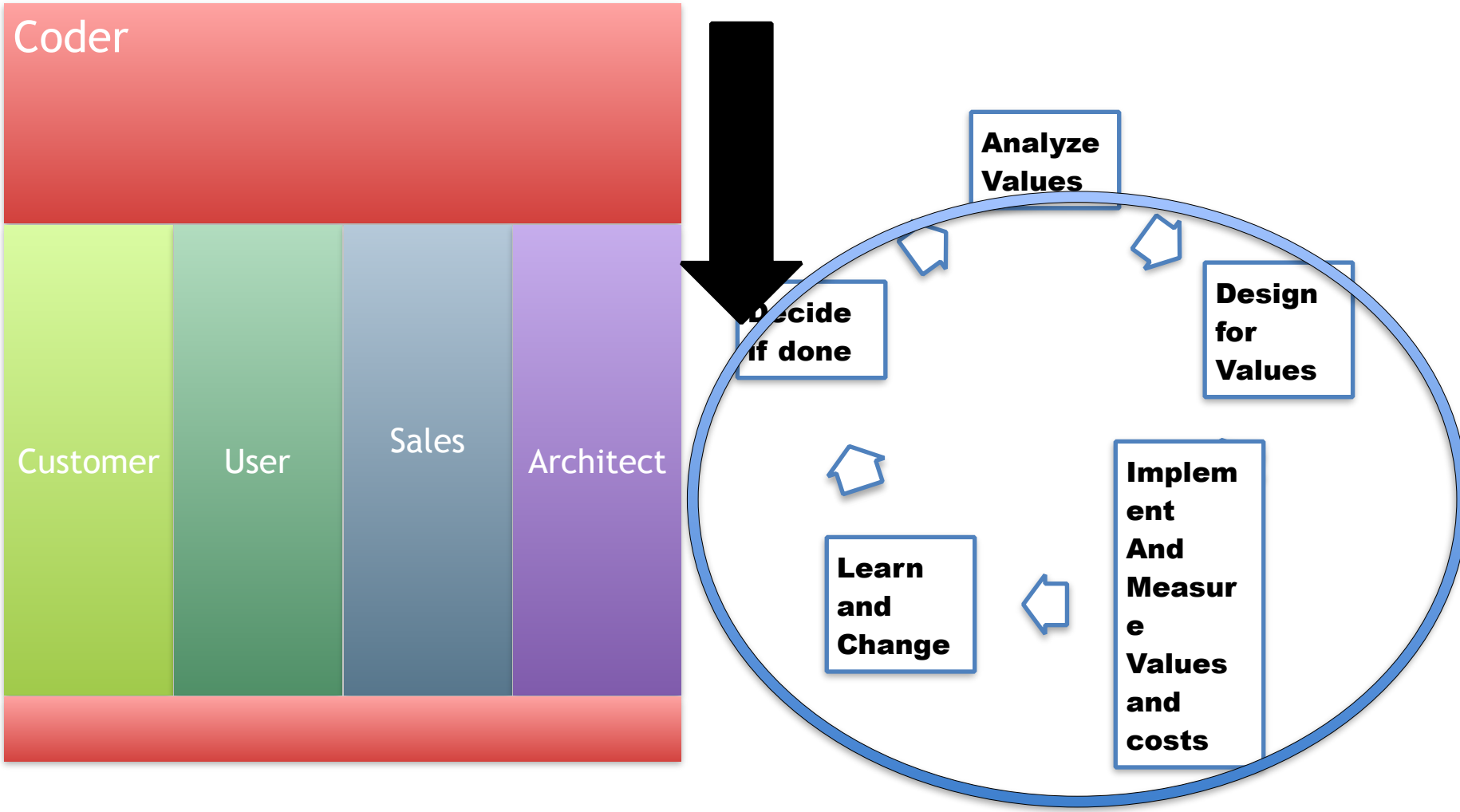
Product/IT System Design



or



Programmer Team does *design* and *measurement* of their design



The Confirmit Case Study 2003-2014

Their product =

confirmit®

Trond Johansen

We gave them a 1 day briefing on our Evo method and Planguage

That's all they needed to succeed!

They were Real engineers

Customer Successes in Corporate Sector

Real Example of 1 of the 25 Quality Requirements

Usability.Productivity:

Scale for quantification: Time in minutes to set up a typical specified Market Research-report

Past Level [Release 8.0]: 65 mins.,

Tolerable Limit [Release 8.5]: 35 mins.,

Goal [Release 8.5]: 25 mins.

Trond Johansen

Shift: from Function to Quality

- **Our new focus is on the daily operations of our Market Research users,**
  - **<u>not</u> a list of features. that they might or might not like. 50% never used!**
  -
  - **We KNOW that increased efficiency, which leads to more profit, will please them.**
  - **The '45 minutes actually saved x thousands of customer reports'**
    - **= big $$$ saved**
- **After one week we had defined more or less all the requirements for the next version (8.5) of Confirmit.**

Quantified Value Delivery Project Management in a Nutshell

Quantified Value Requirements, Design, Design Value/cost estimation, Measurement of Value Delivery, Incremental Project Progress to Date

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Weekly

Estimates

Priority

Next week Warning metrics

Cumulative weekly progress metric

Constraint

Target

Every user, every day, was using an average of 65 minutes to set up a report

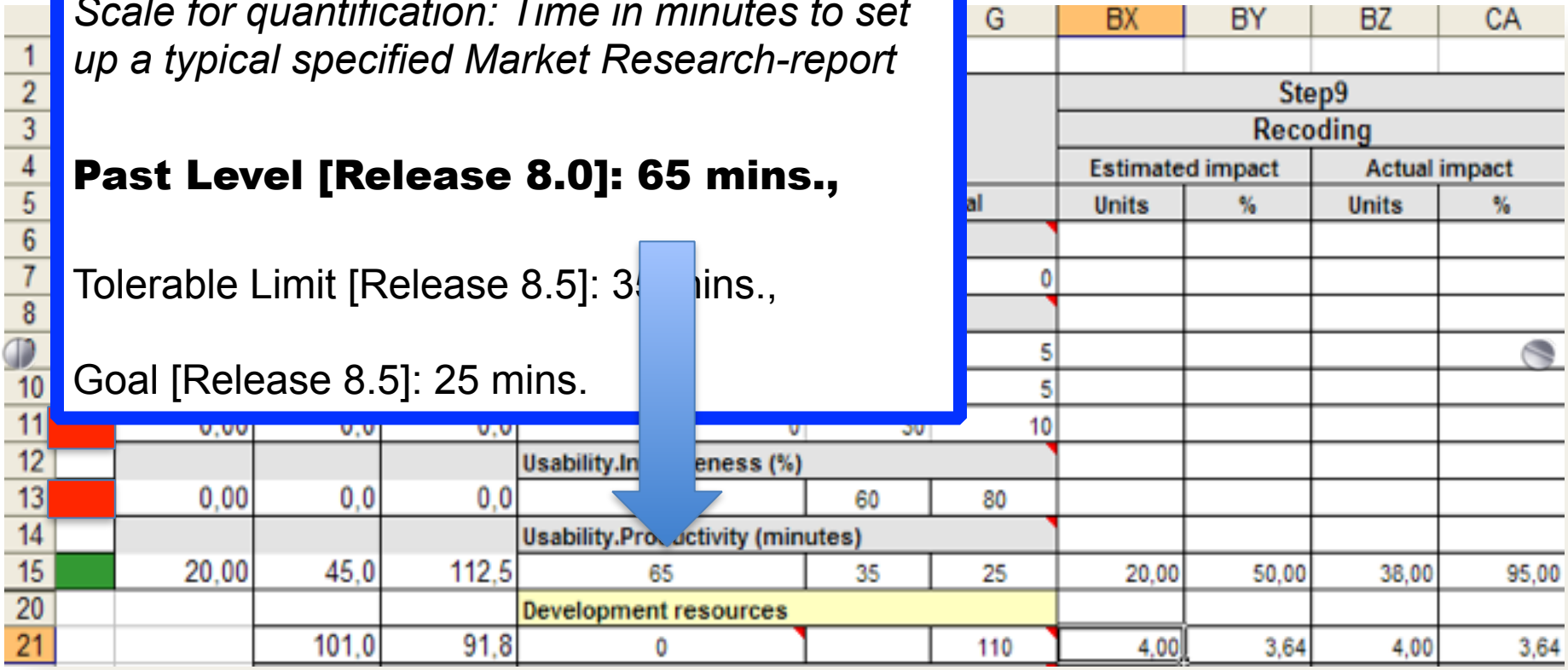
Usability.Productivity

Scale for quantification: Time in minutes to set up a typical specified Market Research-report

Past Level [Release 8.0]: 65 mins.,

Tolerable Limit [Release 8.5]: 3[illegible]ins.,

Goal [Release 8.5]: 25 mins.



								G	BX	BY	BZ	CA
1												
2									Step9			
3									Recoding			
4									Estimated impact		Actual impact	
5								al	Units	%	Units	%
6												
7								0				
8												
9								5				
10								5				
11		0,00	0,0	0,0		0	30	10				
12					Usability.In eness (%)							
13		0,00	0,0	0,0		60	80					
14					Usability.Pro ctivity (minutes)							
15		20,00	45,0	112,5	65	35	25		20,00	50,00	38,00	95,00
20					Development resources							
21			101,0	91,8	0		110		4,00	3,64	4,00	3,64

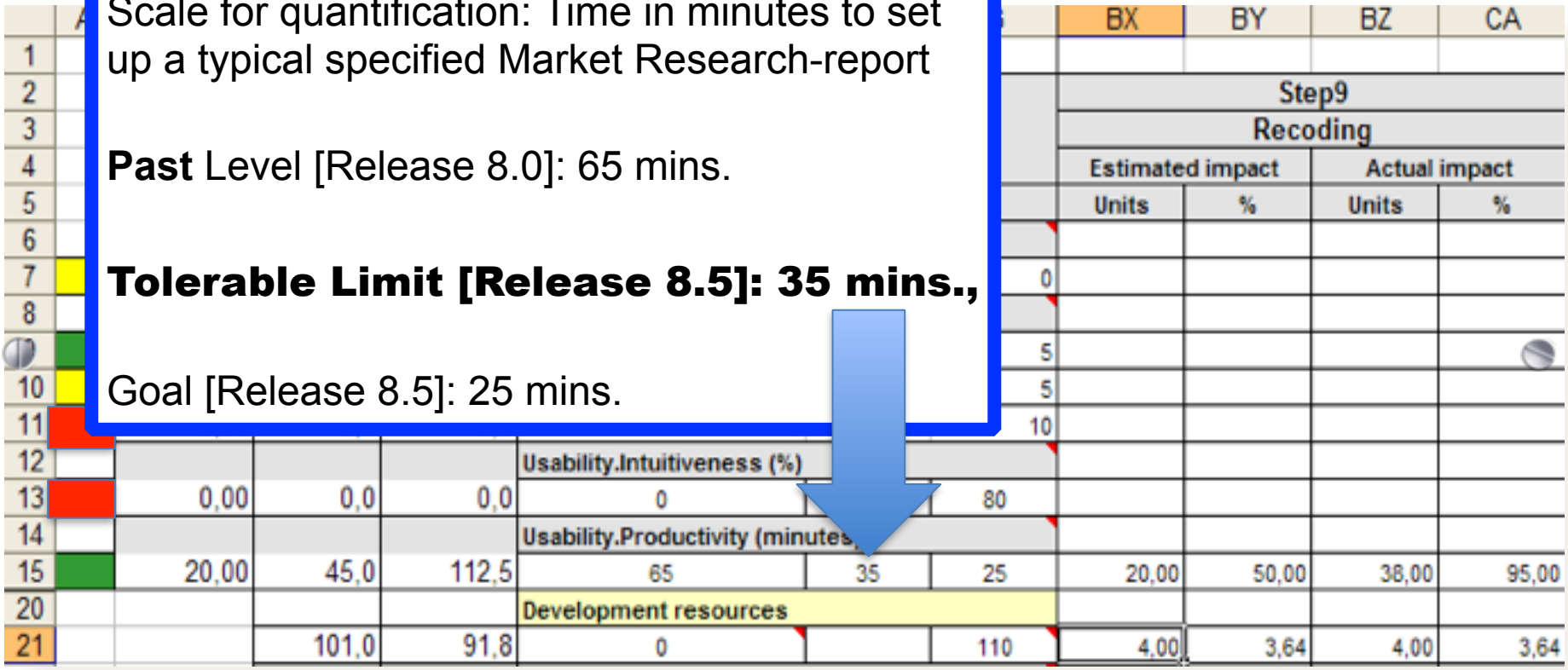
The worst acceptable case requirement, for the next quarterly world release, is 35 minutes, or better; less is 'intolerable'

Usability.Productivity
Scale for quantification: Time in minutes to set up a typical specified Market Research-report

Past Level [Release 8.0]: 65 mins.

Tolerable Limit [Release 8.5]: 35 mins.,

Goal [Release 8.5]: 25 mins.



	BX	BY	BZ	CA
2	Step9			
3	Recoding			
4	Estimated impact		Actual impact	
5	Units	%	Units	%
6				
7				
8				
10				
11				
12				
13				
14				
15	20,00	50,00	38,00	95,00
20				
21	4,00	3,64	4,00	3,64

7					0		
10					5		
11					5		
12					10		
13	0,00	0,0	0,0	Usability.Intuitiveness (%)			
14							
15	20,00	45,0	112,5	Usability.Productivity (minutes)			
20					65	35	25
21		101,0	91,8	Development resources			
					0		110

The committed target level requirement, the 'Goal', is to get the user task down to 25 minutes or better.

Usability.Productivity

Scale for quantification: Time in minutes to set up a typical specified Market Research-report

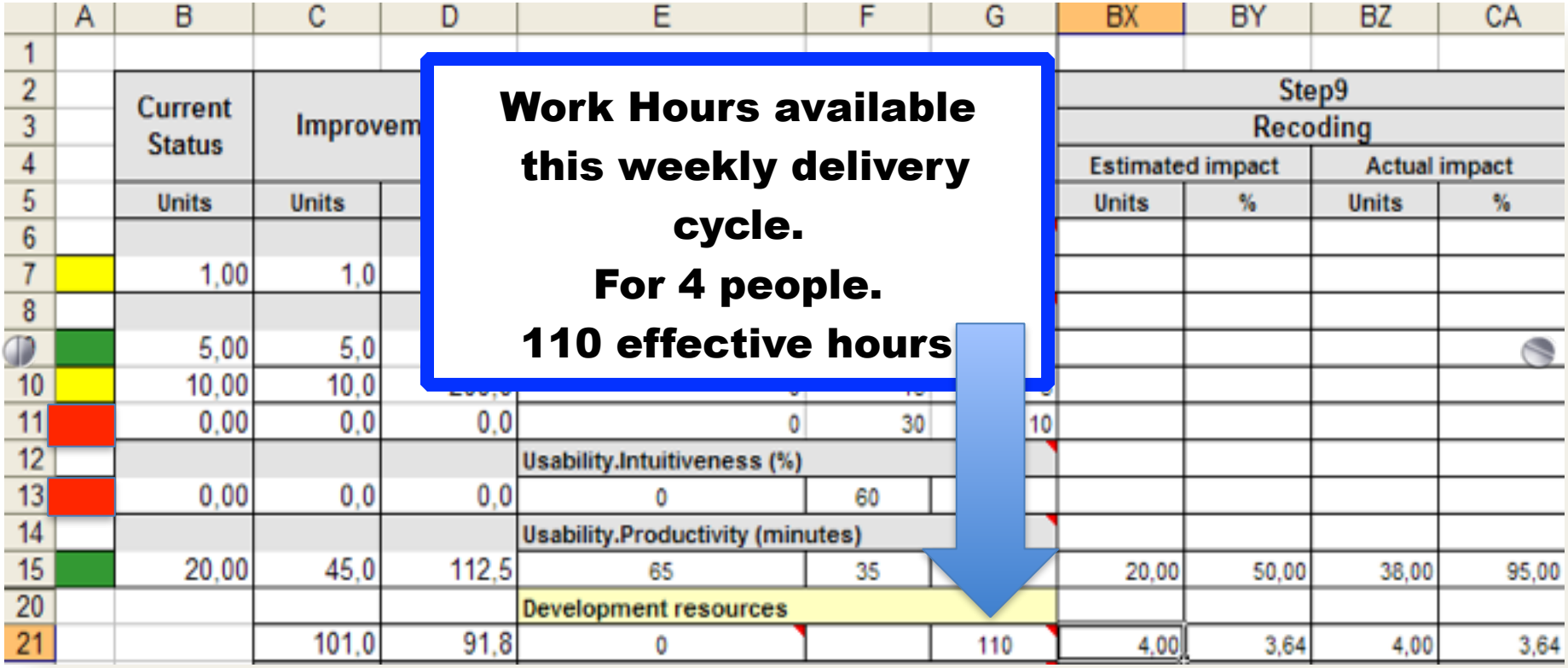
Past Level [Release 8.0]: 65 mins.,

Tolerable Limit [Release 8.5]: 35 mins.,

Goal [Release 8.5]: 25 mi

	A	B	C								CA
1											
2		Current Status	Imp								
3											
4										ctual impact	
5		Units	Units							ts	%
6											
7		1,00									
8											
9		5,00									
10		10,00	1								
11		0,00									
12											
13		0,00									
14											
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

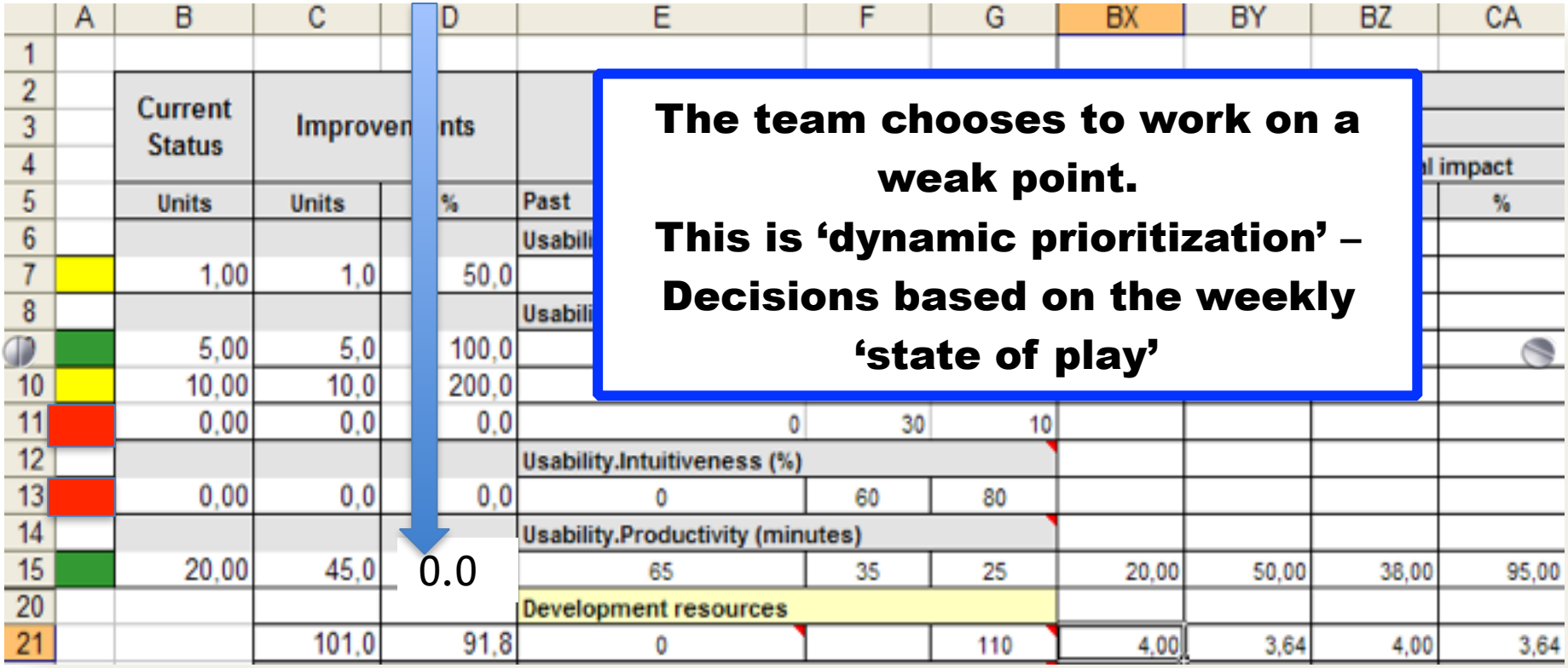
The weekly ‘value delivery cycle’ resource is 110 work-hours (4 days, effective time for the team of 3 to 4 people)



	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvem					Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units					Units	%	Units	%
6											
7		1,00	1,0								
8											
9		5,00	5,0								
10		10,00	10,0								
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60					
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35		20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Work Hours available this weekly delivery cycle.
For 4 people.
110 effective hours

The developer team can choose the requirement they want to prioritize, and work on, this week. They chose the 0.0 (no improvement yet, in last 8 weeks) of the 'Productivity requirement



	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improven	nts							
3											
4											l impact
5		Units	Units	%	Past						%
6					Usabili						
7		1,00	1,0	50,0							
8					Usabili						
9		5,00	5,0	100,0							
10		10,00	10,0	200,0							
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	0.0	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

The team chooses to work on a weak point.
This is 'dynamic prioritization' – Decisions based on the weekly 'state of play'

Every user, every day, was using an average of 65 minutes to set up a report. We want a 40 minute improvement to that, to 25 minutes

Usability.Productivity

Scale for quantification: Time in minutes to set up a typical specified Market Research-report

Past Level [Release 8.0]: 65 mins.,

Tolerable Limit [Release 8.5]: 35 mins.,

Goal [Release 8.5]: 25 mins.

								BX	BY	BZ	CA
1											
2								Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5								Units	%	Units	%
6											
7											
8											
10											
11											
12											
13											
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

The team has a 30 minute ‘design’ meeting, to suggest designs which might help move from 65 minutes for the task, towards the 25 minute Goal level

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

‘Recoding’ is the name of 1 of 12 suggested, brainstormed, designs for saving user effort, by any member of the developer team

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

‘Recoding’ was estimated, by the suggester, to save 20 minutes time for the users

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

‘Recoding’ was also estimated to take the entire 4 day delivery cycle available. No time left to add more solutions, in order to try to get closer to the target, on this delivery cycle.

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	[illegible]	35	25	20,0	50,00	38,00	95,00
20					Development resources						
21			101,0	91,0	0			4,00	3,64	4,00	3,64

And 20 minutes saving, was the best 'impact' estimated from the 12 total suggestions made by the team members. So 'Recoding' (of marketing codes) was chosen as the best thing to do that week.

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

And 20 minutes saving, is equivalent to 50% of the way betweem Past and Goal (65 – 25 = 40, 20/40 = 50%).

This is another way of expressing the expected impact of Recoding

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

The team commits to the ‘Recoding’ solution. They code, test and handover to Microsoft usability Labs in Washington State, who volunteered to independently measure all the Usability designs.

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

The result was a saving, or improvement of 38 minutes, or 95% of the way to the target requirement of 25 minutes

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

This was not good enough for Trond Johansen.

And he did not want to use 1 of the 3 remaining weeks to release (10, 11, 12th weeks) in order to get to 100% of the target.

So, he asked one team member to spend the weekend tuning the 'Recoding' solution.

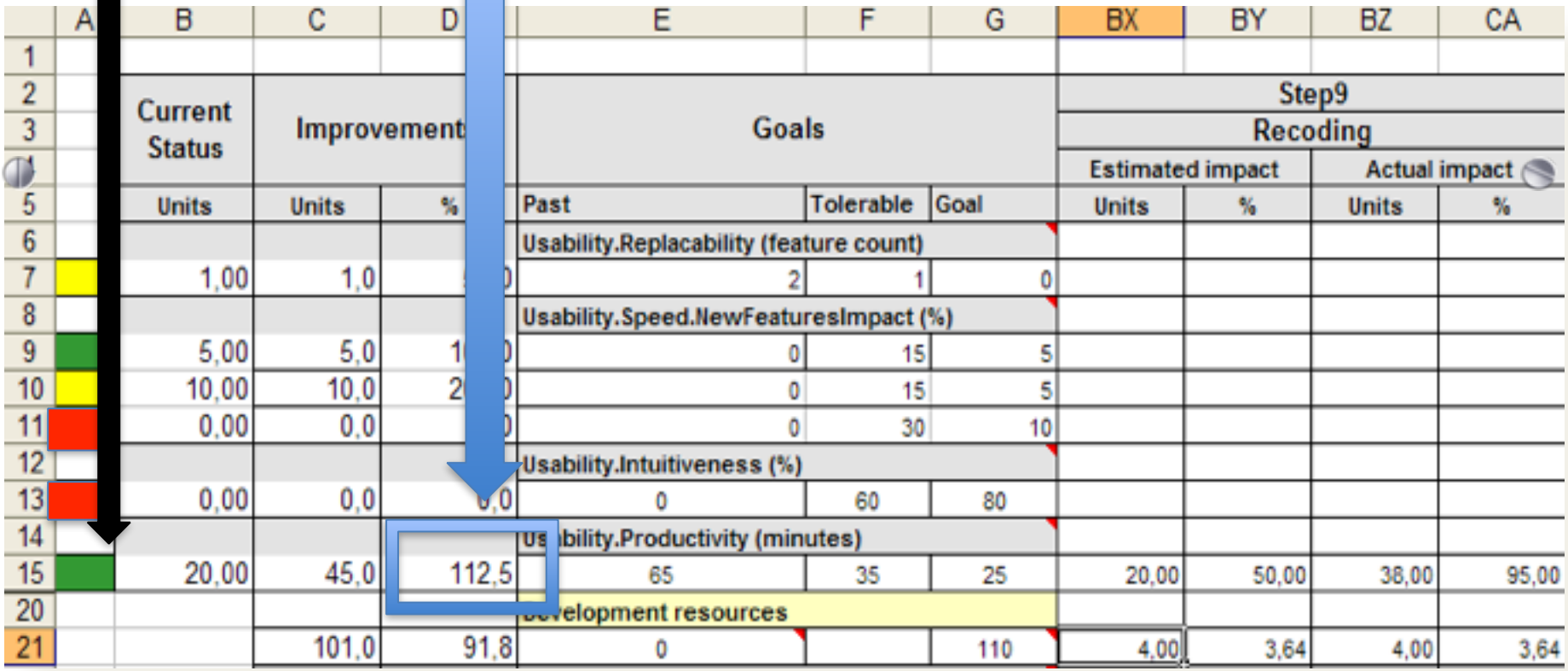
And he managed to get the timing down to 20 minutes.

12.5% more than the 25 minutes targeted.

Thus total impact is 112.5%

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvement		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0		2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	1	0	15	5				
10		10,00	10,0	2	0	15	5				
11		0,00	0,0		0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

And the priority flag turns Green (no priority, Goal reached)



	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvement		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0		2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	1	0	15	5				
10		10,00	10,0	2	0	15	5				
11		0,00	0,0		0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

EVO Plan Confirmit 8.5 in **Evo Step Impact *Measurement***

4 product areas were attacked in all: **25 Qualities** concurrently, one quarter of a year. Total development staff = 13

9

8

3

3

Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	0
			Usability.ResponseTime.ExportRep (min		s
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewRepo	seco	s)
1,0	14,0	100,0	15		1
			Development resources		
203,0			0		91

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,0	35	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0	800	100,0	800	0	0
			Runtime.Concurrency (number of users)		
350	110	146,7	150	500	1000
			Development resources		
64			0		84

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		48

Confirmit Evo Weekly Value Delivery Cycle

	Development Team	Users (PMT, Pros, Doc writer, other)	CTO (Sys Arch, Process Mgr)	QA (Configuration Manager & Test Manager)
Friday	✓ PM: Send Version N detail plan to CTO + prior to Project Mgmt meeting ✓ PM: Attend Project Mgmt meeting: 12.00-15.00 ✓ Developers: Focus on genereal maintenance work, documentation.		✓ Approve/reject design & Step N ✓ Attend Project Mgmt meeting: 12-15	✓ Run final build and create setup for Version N-1. ✓ Install setup on test servers (external and internal) ✓ Perform initial crash test and then release Version N-1
Monday	✓ Develop test code & code for Version N	✓ Use Version N-1		✓ Follow up CI ✓ Review test plans, tests
Tuesday	✓ Develop Test Code & Code for Version N ✓ Meet with users to Discuss Action Taken Regarding Feedback From Version N-1	✓ Meet with developers to give Feedback and Discuss Action Taken from previous actions	✓ System Architect to review code and test code	✓ Follow up CI ✓ Review test plans, tests
Wednesday	✓ Develop test code & code for Version N			✓ Review test plans, tests ✓ Follow up CI
Thursday	✓ Complete Test Code & Code for Version N ✓ Complete GUI tests for Version N-2			✓ Review test plans, tests ✓ Follow up CI

Evo's impact on Confirmit product qualities 1st Qtr

- Only 5 highlights of the 25 impacts are listed here

Description of requirement/work task

confirmit

Release 8.5

Developers love 'Empowered Creativity'

- **EVO has resulted in**
  - **increased motivation and**
  - **enthusiasm amongst developers,**
  - **it opens up for empowered creativity**

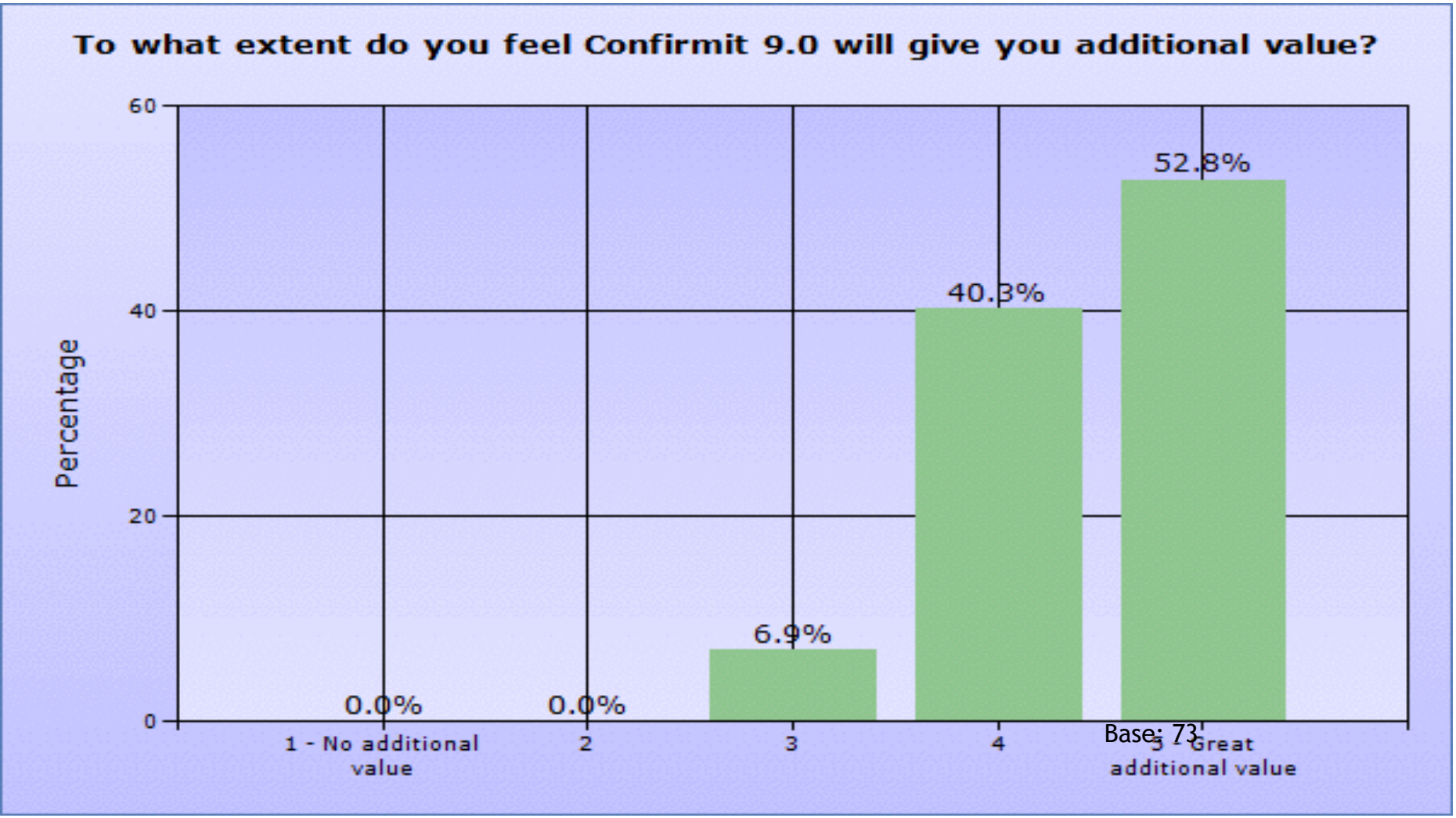
- **Developers**
  - **embraced the method and**
  - **saw the value of using it,**
  - **even though they found parts of Evo difficult to understand and execute (without training)**

Trond Johansen

Initial Customer Feedback on the new Confirmit 9.0

November 24th, 2004

Initial perceived value of the new release (Base 73 people)



Evo's impact on Confirmit 9.0 product qualities Results from the second quarter of using Evo. 1/2

Product quality	Description	Customer value
Intuitiveness	Probability that an inexperienced user can intuitively figure out how to set up a defined Simple Survey correctly.	Probability increased by **175%**
Productivity	Time in minutes for a defined advanced user, with full knowledge of 9.0 functionality, to set up a defined advanced survey correctly.	Time reduced by **38%**

Product quality	Description	Customer value
Productivity	Time (in minutes) to test a defined survey and identify 4 inserted script errors, starting from when the questionnaire is finished to the time testing is complete and is ready for production. (Defined Survey: Complex survey, 60 questions, comprehensive JScripting.)	Time reduced by **83%** and error tracking increased by 25%

Evo's impact on Confirmit 9.0 product qualities Results from the second quarter of using Evo. 2/2

Product quality	Description	Customer value
Performance	Max number of panelists that the system can support without exceeding a defined time for the defined task, with all components of the panel system performing acceptable.	Number of panelists increased by **1500**%
Scalability	Ability to accomplish a bulk-update of X panelists within a timeframe of Z second	Number of panelists increased by **700%**
Performance	Number of responses a database can contain if the generation of a defined table should be run in 5 seconds.	Number of responses increased by **1400%**

Case:
Delegating
Developer Environment
to Developers
using **Multidimensional Engineering**

Technical debt

consequences of poor software architecture and software development within a codebase.

Causes of technical debt

1. **Business pressures**
2. **Lack of process or understanding**
3. **Lack of building loosely coupled components**,
4. **Lack of test suite**,
5. **Lack of documentation**,
6. **Lack of collaboration**
7. **Parallel**
8. **Delayed Refactoring**

There is a smarter way

- But it means we have to become **real** software *engineers*,

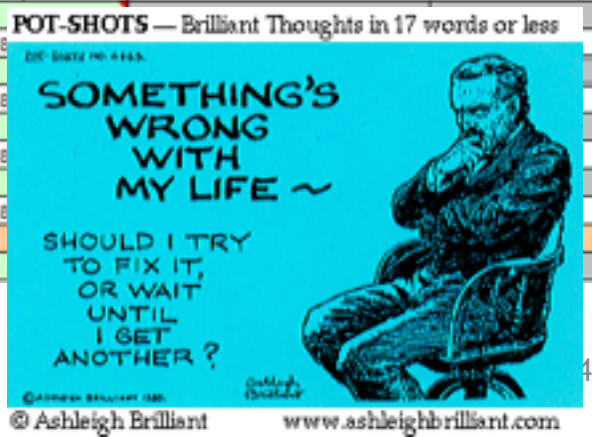
- Not just- - - *softcrafters**

- * coders, developers, programmers.
  - Term coined in
  - "Principles of Software Engineering Management", 1988, Gilb

Code quality – ”green” week
Empowered Creativity: for Maintainability

- Instead of Refactoring 1 day a week (failed)
- Let the Dev Teams <u>engineer</u> using ‘agile’ (Evo): Design Dev Quality in to their own process
- To meeting their own internal stakeholder Quality Objectives
- 1 week a month

Current Status		Improvement	Goals			Step 6 (week 14)		Step 7 (wee
	Units		Past	Tolerable	Goal	Estimated Impact	Actual Impact	Estimated Impact
	100,0	100,0	0	80	100			100
Speed								
	100,0	100,0	0	80	100	100	100	
Maintainability.Doc.Code								
	100,0	100,0	0	80	100	100	100	
InterviewerConsole								
NUnitTests								
	0,0	0,0	0	90	100			
PeerTests								
	100,0	100,0	0	90	100			100
FxCop								
	0,0	10,0	10	0	0			
TestDirectorTests								
	100,0	100,0	0	90	100			100
Robustness.Correctness								
	2,0	2,0	0	1	2	2	2	
Robustness.BoundaryConditions								
	0,0	0,0	0					
Speed								
	0,0	0,0	0					
ResourceUsage.CPU								
	100,0	0,0	100					
Maintainability.Doc.Code								
	100,0	100,0	0					
SynchronizationStatus								
NUnitTests								



Speed

Maintainability

Nunit Tests

PeerTests

TestDirectorTests

Robustness.Correctness

Robustness.Boundary Conditions

ResourceUsage.CPU

Maintainability.DocCode

SynchronizationStatus

Same Process as for their External (User, Customer) stakeholders

- 1. **define** better quality dev and testing **environment** QUANTITATIVELY
  - Scale of measure and Goal level
- 2. **Figure** out, brainstorm ANY systems engineering design or **architecture** to get to their self determined improvement goals
  - Not just code refactoring, but any tools, processes, motivations, hardware etc that WORK
- 3. **Implement**, **measure**
  - Keep the stuff that works
  - Dump the stuff that does not MEASURABLY work
- 4. Keep on trucking' (monthly, forever, or …)
  - DONE is when **devs** have no further improvement needs

The Monthly ‘Green Week’

User Week 1
- Select a Goal
- Brainstorm Designs
- Estimate Design Impact/Cost
- Pick best design
- Implement design
- Test design
- Update Progress to Goal

User Week 2
- Select a Goal
- Brainstorm Designs
- Estimate Design Impact/Cost
- Pick best design
- Implement design
- Test design
- Update Progress to Goal

User Week 3
- Select a Goal
- Brainstorm Designs
- Estimate Design Impact/Cost
- Pick best design
- Implement design
- Test design
- Update Progress to Goal

Developer Week 4
- **Select a Goal**
- **Brainstorm Designs**
- **Estimate Design Impact/Cost**
- **Pick best design**
- **Implement design**
- **Test design**
- **Update Progress to Goal**

Conclusion: Technical Debt

- **Developers**

  Acting like real software engineers

  Can engineer technical debt reduction

  It is NOT about refactoring, and patterns

  though if they work measurably best, we can use them.

  But, did you ever see measurement or re they just belief systems?

  It is about mature teams, with common goals, and practical experience, taking charge of their own fate

  If management resists, I suggest going on strike!

  Why should we suffer agonizing technical debt, wasting 50% or more of our work hours,

  Surely we have better things to do!

Cleanroom

In the Cleanroom Method, developed by IBM's Harlan Mi 1970-1980 they reported: IBM SJ 4/80

- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) "some ten years ago [Ed. about 1970] in a continuing evolution that is still underway:*
- *Ten years ago general management expected the worst from software projects – cost overruns, late deliveries, unreliable and incomplete software*
- *Today [Ed. 1980!], management has learned to expect on-time, within budget, deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and* **a ship in 45 incremental deliveries** *[Ed. Note 2%!]s***. Every one of those deliveries was on time and under budget**
- *A more extended example can be found in the NASA space program,*
- *- Where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million bytes of program and data for ground and space processors in over a dozen projects.*
- **- There were few late or overrun deliveries in that decade, and none at all in the past four years."**

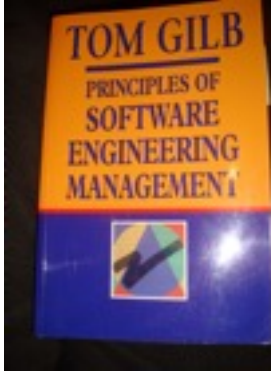
In the Cleanroom Method, developed by IBM's Harlan Mills (1980) they report
PERFECT SOFTWARE PROJECTS: by Feedback

- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division,*
- *- cost overruns, late deliveries, unreliable and incomplete software*
- *Today [Ed. 1980!], management has learned to expect on-time, within budget, deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating*
- *A mor*
- *- Whe*
- *- Ther*

in 45 incremental deliveries

were few late or overrun deliveries in that decade, and none at all in the past four years

Quinnan: IBM FSD Cleanroom *Dynamic Design to Cost*



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. . . yields valid cost plans linked to technical performance. Our practice carries cost management farther by introducing design-to-cost guidance. Design, development, and managerial practices are applied in an integrated way to ensure that software technical management is consistent with cost management. The method [illustrated in this book by Figure 7.10] consists of developing a design, estimating its cost, and ensuring that the design is cost-effective.' (p. 473)

He goes on to describe a design iteration process trying to meet cost targets by either redesign or by sacrificing 'planned capability.' When a satisfactory design at cost target is achieved for a single increment, the 'development of each increment can proceed concurrently with the program design of the others.'

'Design is an iterative process in which each design level is a refinement of the previous level.' (p. 474)

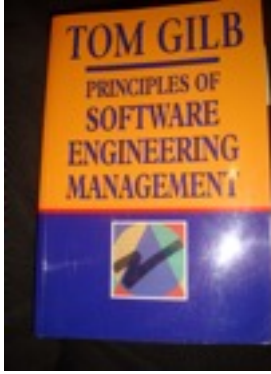
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability of learning from experience, won as each increment develops, and as the true cost of the increment becomes a fact.

'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77

This text is cut from Gilb: The Principles of Software Engineering Management, 1988

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. ... cost management farther by introducing design- ... an integrated way to ensure that software techn ... n this book by Figure 7.10] consists of develop ... ' (p. 473)

He goes on t ... sign or by sacrificing 'planned capability.' When a ... pment of each increment can proceed concurrent ...

'Design is an iterati ... 474)

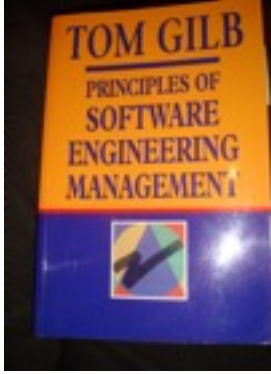
It is clear fro ... iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability of learning from experience, won as each increment develops, and as the true cost of the increment becomes a fact.

of developing a design, estimating its cost, and ensuring that the design is cost-effective

'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77
This text is cut from Gilb: The Principles of Software Engineering Management, 1988

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. . . yields valid cost plans linked to technical performance. Our practice carries cost management farther by introducing design-to-cost guidance. Design, development, and managerial practices are applied in an integrated way to ensure that software technical management is consistent with cost management. The method [illustrated in this book by Figure 7.10] consists of developing a design, estimating its cost, and ensuring that the design is cost-effective.' (p. 473)

He goes on to describe a design iteration process trying to meet cost targets by either redesign or by sacrificing 'planned capability.' When a satisfactory design at cost target is achieved for a single increment, the 'development of each increment can proceed concurrently with the program design of the others.'

'Design is an iterati **474)**

It is clear fro **iterate in seeking the appropriate balance** **ies of increments, thus reducing the compl** **n as each increment develops, and as th**

'When the developn **ing increments is computed.' (p. 474)**

Source: Robert E. Qu No. 4, 1980, pp. 466~77

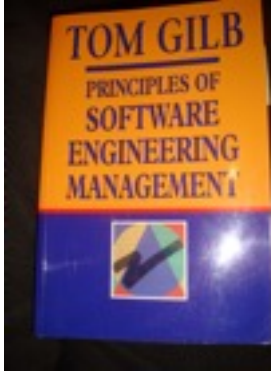
This text is cut from

iteration process trying to meet cost targets by either *redesign* or by *sacrificing* 'planned capability'

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost

Design is an iterative process

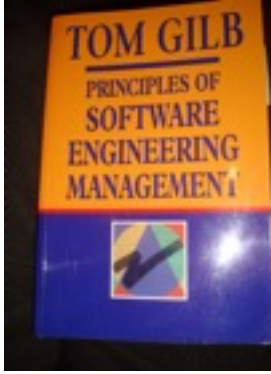
Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

but they iterate through a series of increments,
thus *reducing the complexity of the task*,
and *increasing the probability of learning from experience*

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

an estimate to complete the remaining increments is computed.

Richard Smith

A story of devs refusing to be told how to design by Bank IT architects. Focussing on a few critical value measurable Objectives;
and delivering **on time** for **full user satisfaction**: 100% success
Using Agile Evo: The Engineering Agile Method

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006"

Previous IT Project Management Methods: No 'Value delivery tracking'. No change reaction ability

Richard Smith

- "However, (**our old** project management methodology) main failings were that
- it almost **totally missed the ability to track delivery of actual *value* improvements to a project's stakeholders**,
- and **the ability to react to changes**
  - **in requirements and**
  - **priority**
  - **for the project's duration"**

We only had the illusion of control. But little help to testers and analysts

Richard Smith

- "The (old) toolset generated lots of charts and stats
- that provided **the illusion of risk control**.
- But actually provided very little help to the analysts, developers and testers actually doing the work at the coal face."

The proof is in the pudding;

Richard Smith

- "The proof is in the pudding;
- I have **used Evo**
  - *(albeit in disguise sometimes)*
  - on two large, high-risk projects in front-office investment banking businesses,
  - and several smaller tasks. "

Experience: if top level requirements are *separated* from design, the 'requirements' are **stable**!

Richard Smith

- "On the largest critical project,
- the original ***business functions & performance objective*** **requirements document,**
- ***which included*** **no design,**
- essentially remained ***unchanged***
- over the **14 months** the project took to deliver,…."

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006", Richard Smith

Dynamic (Agile, Evo) design testing: not unlike 'Lean Startup'

Richard Smith

- "… but the detailed **designs**
  - **(of the GUI, business logic, performance characteristics)**
- **changed** many many times,
- guided by lessons learnt
- and **feedback** gained by
- delivering a succession of early deliveries
- to real users"

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006", Richard Smith

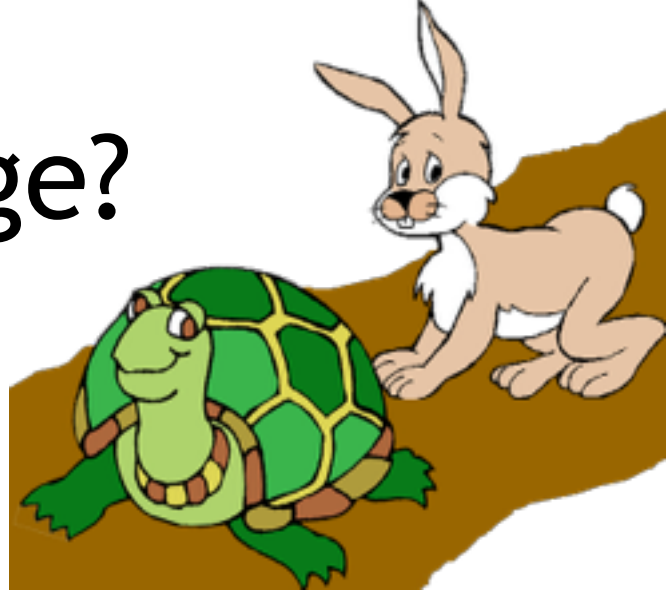
It looks like the stakeholders liked the top level system qualities, on first try

Richard Smith

- " In the end, the new system responsible for 10s of USD billions of notional risk,
- **successfully went live**
- **over one weekend**
- **for 800 users worldwide**,
- and **was seen as a big success**
- **by the sponsoring stakeholders."**

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006" , Richard Smith

Is it so hard to change?



- **NOT** if we **delegate power to the people in the trenches**
- And that means giving them information about the problems
- Letting them be driven by **stakeholder values** and goals
  - But finding their *own* solutions to these challenges
- And giving them a chance to **suggest**, and **make**, changes
- And giving them a chance to **measure** the success, or failure, of their own ideas
- To **learn** and try again
- To eternally perform a change process, at their own pace
- Supported, protected, and funded by management

- **YES IT IS damned HARD**
- If managers try top down, command and control
  - And dictate solutions like agile, lean, CMMI
  - With a deadline next year
- And hard if outside or inside consultants, are the source of the 'big change ideas'
  - It is the **many small practical ideas** that win in the long term

My 10 Principles of Improvement

Work Environment

1. Delegate to the doers
2. Measure the improvements
3. Let troops identify common cause defects
4. Let them suggest root causes
5. Let them suggest and try cures

Product Development

6. Let troops choose the value goal to work on
7. Let them estimate the power of their ideas
8. Let them decide which design to implement
9. Let them measure the results, this week and total to date
10. Credit them for the results, and reward success

The Revolution is here

- Programmers of the world Unite!

For a free *underground revolutionary Handbook*

for changing
Coder -> Software Engineer.
(The Revolution)
But it might take 10,000 hours to Master it all !
Email to **Tom @ Gilb . Com**
with Subject "**GeeCon 2015**"
if you also want my new book manuscript. 'Value Planning'
put 'VP' in subject

http://tinyurl.com/GilbGeecon

Will get you a copy of these slides
And my papers on Agile
And original historical papers
referred to in this talk
Mays, Mills, Holland, etc.

Go back a slide

- http://tinyurl.com/GilbGeecon