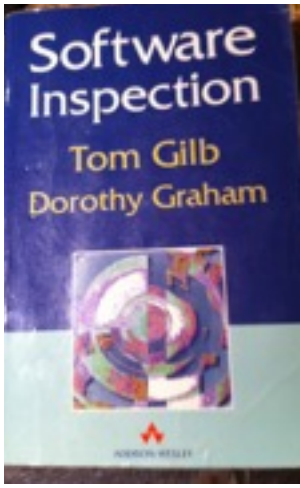
Quantifying All Qualities:
Practical Methods for articulating and specifying all qualitative attributes of all systems quantitatively.
The basis for Quality Engineering

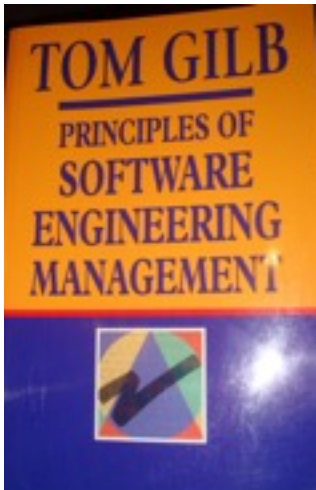
by

Tom @ Gilb . Com

GILB.com

Master





Quality

"We have this simple philosophy that quality is the best business plan"

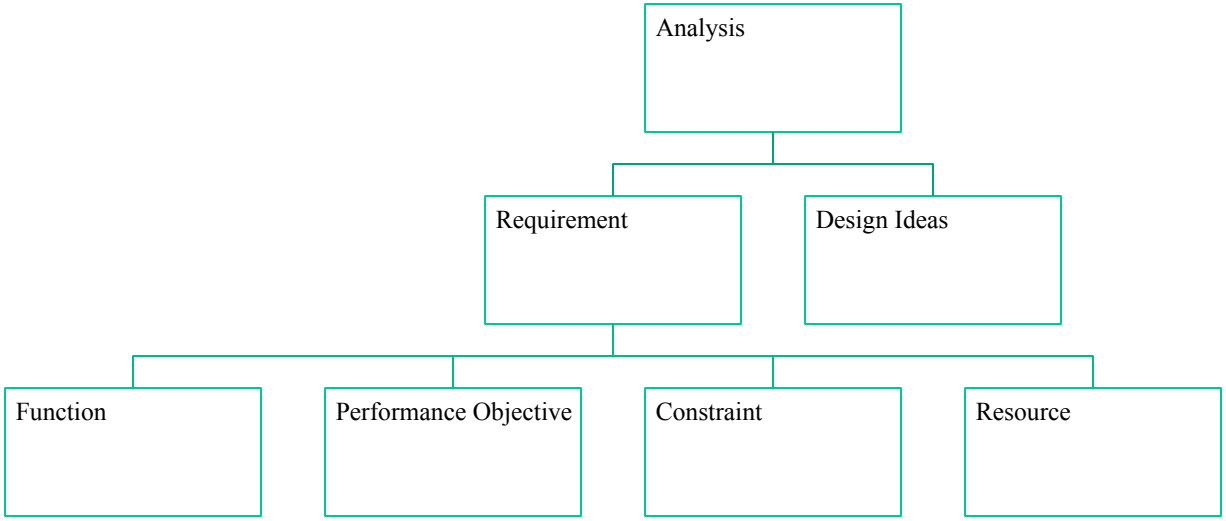
Quote 1.1 A. John Lasseter, Chief Creative Officer, Founder, Pixar. Wired UK, Dec. 2015, p.145.

Recent (20 Sept, 2011) Report on Gilb Evo method (Richard Smith, Citigroup)

- http://rsbatechnology.co.uk/blog:8
- Back in 2004, I was employed by a large investment bank in their FX e-commerce IT department as a business analyst.
- The wider IT organisation used a complex waterfall-based project methodology that required use of an intranet application to manage and report progress.
- However, it's 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.**
- The 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;
  - I have **used Evo** (albeit in disguise sometimes) on two large, high-risk projects in front-office investment banking businesses, and several smaller tasks.
  - 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,
  - 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.
  - In the end, the new system responsible for 10s of USD billions of notional risk, **successfully went live over 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"

Requirement De-composition



Performance #3

#	Tag	Owner	Version	Status
P1.1	**Off Market Event.Detection Interval**	A. Senior Trader	0.2	DRAFT
Ambition:	Minimise time taken to detect that an Off-Market Event has already occurred			
Scale:	Time, in seconds, from occurrence of an [Off-Market Event type] for [FX product] in [Location of Trading System] to Detection, measured during [Time Period].			
Benchmark:	[Published Customer Quote, EM Spot, Another London Bank, Autumn 2010] more than 14400 secs *<- Known market incident at A N Other Bank October 2010 (A Senior Trader)*			
Past:	[Published Customer Quote, EM Spot, London, February 2011] 600 ± 30 secs *<- THB Incident 3 Feb 2011 (A Senior Trader)*			
Goal:	[Published Customer Quote, {EM Spot, G10 Spot}, {London, New York}, end Sept 2011] 1-5 secs?? *<- A Senior Trader 23 March 2011*			
Goal:	Goal[Published Customer Quote, {Any Forward, Any Swap}, London, end Dec 2011] 30 secs *<- A Senior Trader 23 March 2011 Forward and Swap pricing less time-sensitive to off-market deviations <- A N Other Trader 25 March 2011*			
Stretch:	[Published Customer Quote, {Any Spot}, London, end Dec 2011] <0.1 secs *"Estimated state-of-art" <- Mr. Analyst 24 March 2011*			

EVO: Impact Estimation

Requirements:					Designs:				Totals:
		Past		Goal	*Step:*	PSS.Step 1			
					Design Idea:	DI2.4	DI2.5	DI2.6	
#P3.2	**Correction.Cancellation Time**								
	[ECM, London, eFX Desk]	600 secs	->	1 sec		300s	0s	10s	890s
						±150s		±5s	±155s
						50%±25%	0%	98%±1%	148%±26%
	[Quote Control, London, eFX Desk]	600 secs	->	1 sec		0s	10s	0s	590s
							±5s		±5s
							98%±1%		98%±1%
Sum Of Performance:						**50%±25%**	**98%±1%**	**98%±1%**	
#R3.1.1	**Budget.Work.Team**								
	[Developer, H1 2012]	0wd	->	25wd		0.5wd	3wd	10wd	13.5wd
						±0.5wd	±2wd	±5wd	±7.5wd
						2%±2%	12%±12%	40%±20%	54%±34%
	[Tester, H1 2012]	0wd	->	15wd		0.5wd	1wd	3wd	4.5wd
						±0.5wd	±1wd	±2wd	±3.5wd
						3%±3%	7%±7%	20%±14%	30%±24%
	[Project Manager, H1 2012]	0wd	->	10wd		0.5wd	0.5wd	2wd	3wd
						±0.5wd	±0.5wd	±1wd	±2wd
						5%±5%	5%±5%	5%±5%	15%±15%
Sum Of Costs:						**10%±10%**	**24%±24%**	**65%±39%**	
Performance To Cost Ratio:						**5.0**	**4.1**	**1.5**	

Planguage: Objectives & Requirements

Gilb Fest 2014

23rd - 27th June 2014 , London, UK

Man-Chie Tse &
Ravinder Singh Kahlon
{Man-Chie, Ravi}@dkode.co

dkode Limited, London, United Kingdom.
University of Ulster, Northern Ireland, United Kingdom

BROUGHT TO YOU BY

dkode .co

University of ULSTER

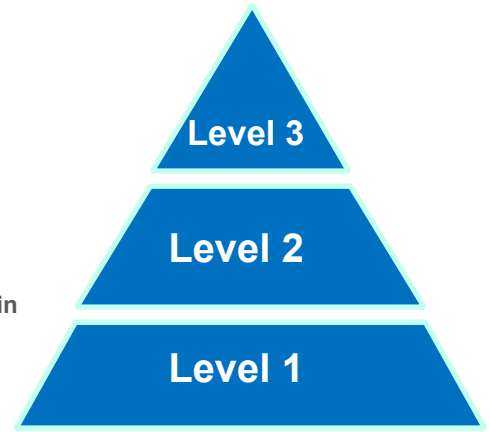
Overview


1
Case Study
2
The Driving Force
3
Objectives
4
Methodology
5
Process Analysis
6
Requirements & Planguage
?

CASE STUDY

THE MISSION:
(1) **Improve Performance,**
(2) **Maximise results,**
(3) **Minimise effort time.**

- An organisation is seeking to utilise technology to support employees workforce processes.
- Looking to evolve the multi skills of employees skills and expertise domain knowledge.
- Currently, confined to a 20 year old process. This process model can no longer meet the service orientation demands for nature of today's environment.

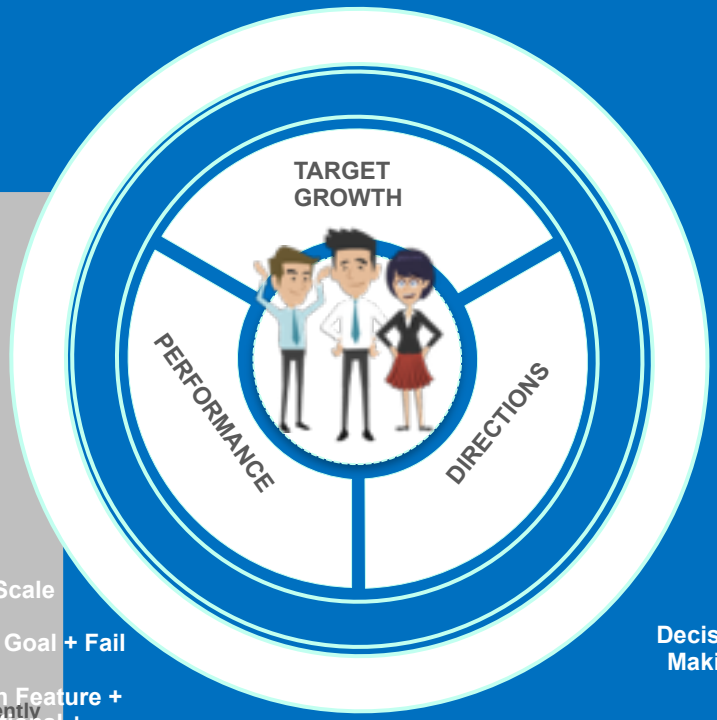


Vision + Rationale

Stakeholders

Function + Performance requirements

£500k ONLY
BUDGET



Decision Making

Scale

Past + Goal + Fail

Design Feature + Rational + Dependency

Service Model

Targets

Financial Savings

CESS RESULTS

Step A

Step B

Process

Process

Process

Process

Step D

Step C

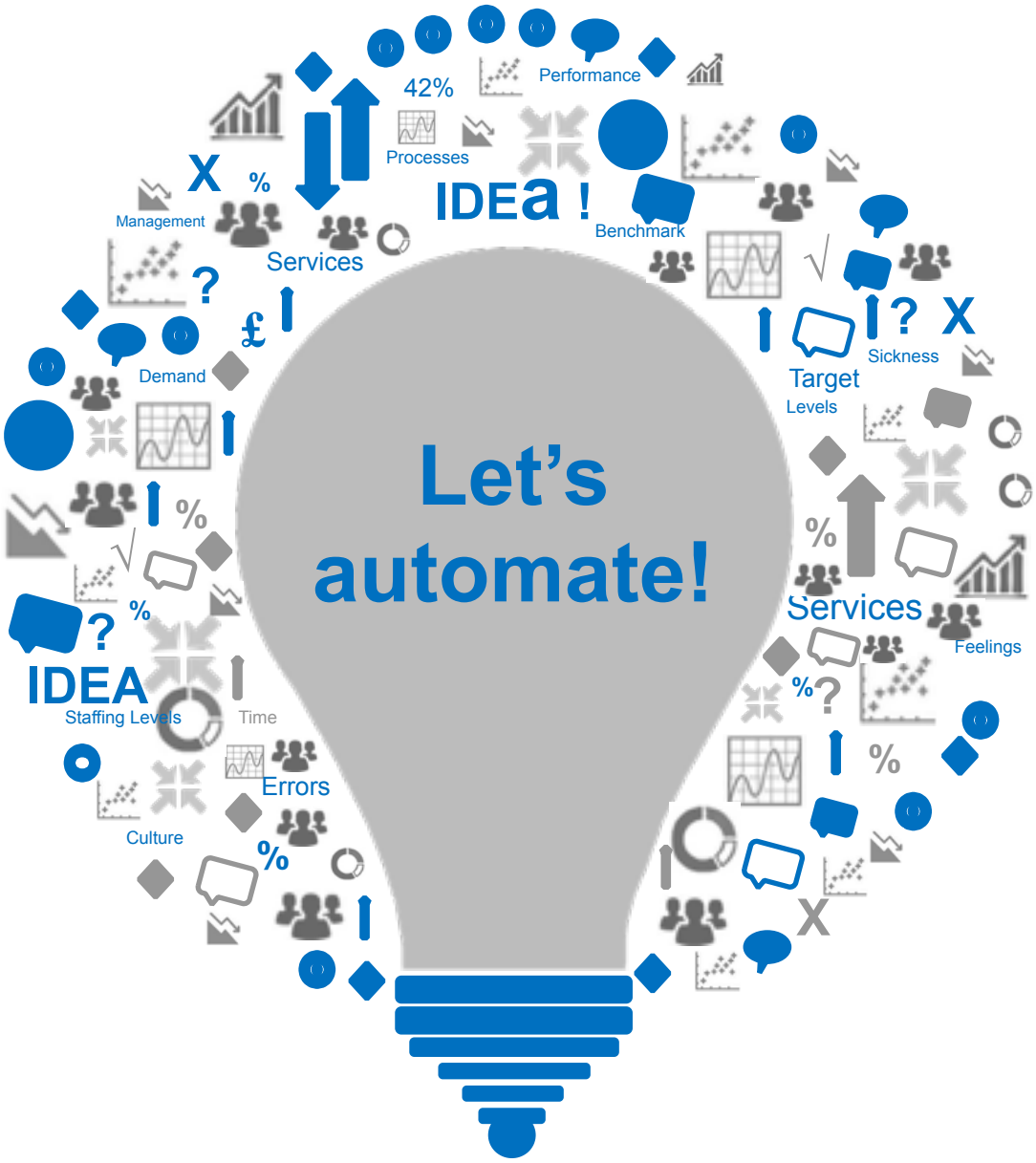
t Opportunity
ance workforce

Replacing Manual Work with Technology

Our Impact
Evolve Processes

We need to work differently

THE IDEA


Let's automate!
42%
Performance
Processes
IDEa !
Benchmark
Management
Services
Sickness
Target
Levels
Demand
Services
Feelings
IDEA
Staffing Levels
Time
Errors
Culture

THE DRIVING FORCE

:(:

ambiguous objectives

Organise our services around our service demand and not our old habits

Excellent workforce with utilising modern practices

Services to run more efficiently & decrease wastage

Compliance by monitoring and achieving value

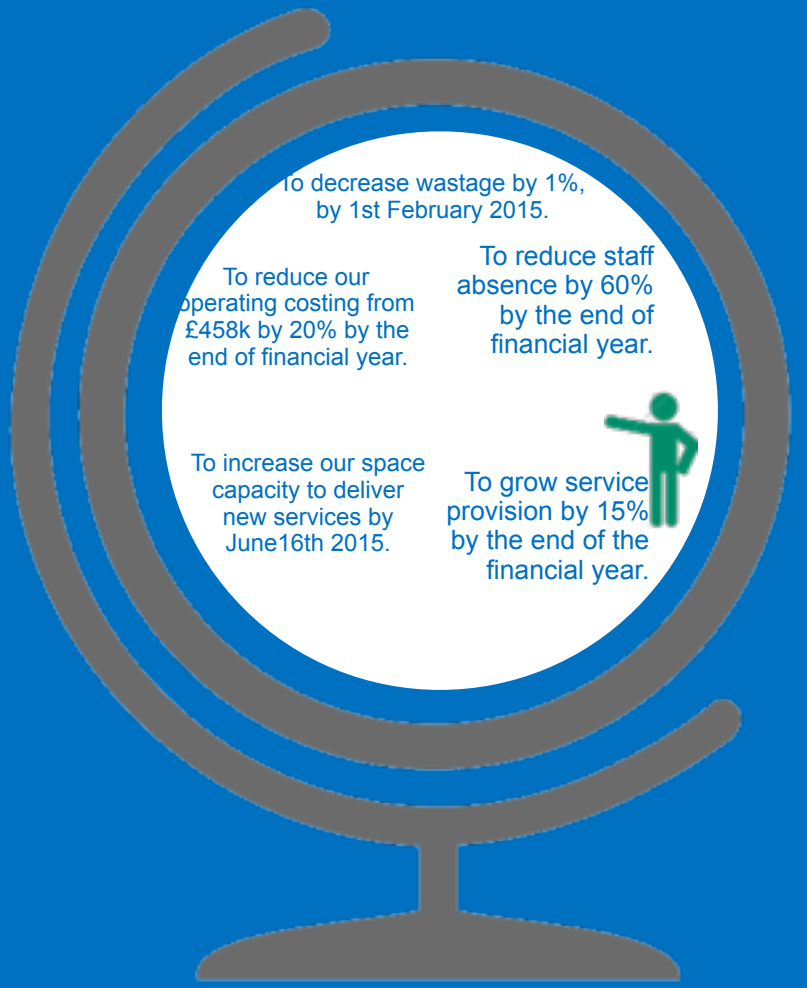
Be a stronger organisation by investing significant money in IT to build the work force

OBJECTIVE = QUANTIFY = REQUIREMENT

The Importance of OBJECTIVEs

quantified gives defined

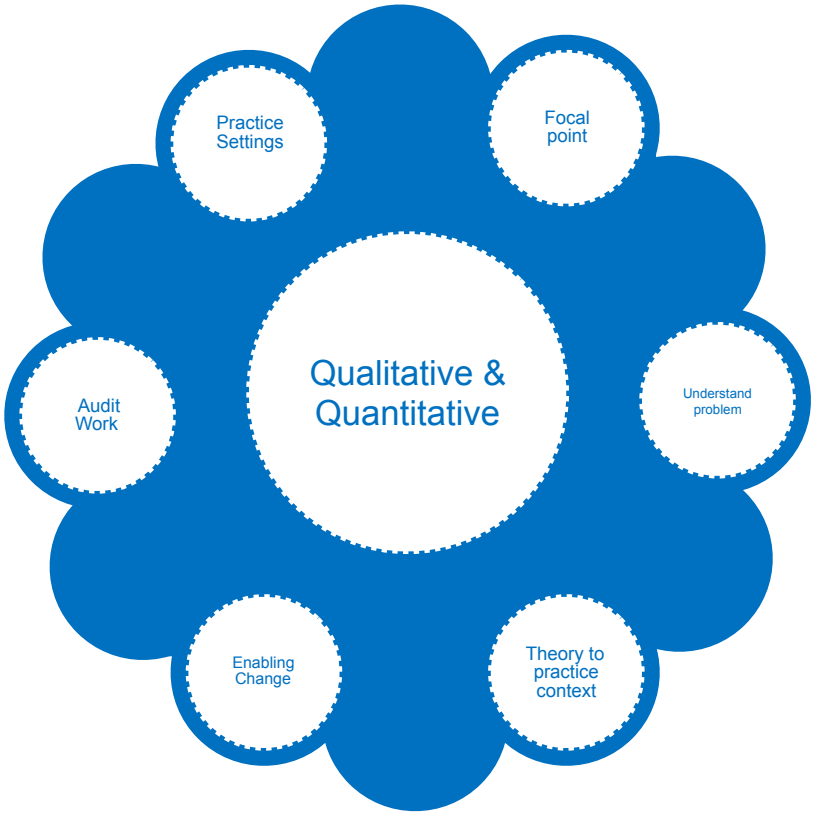
MEANING!



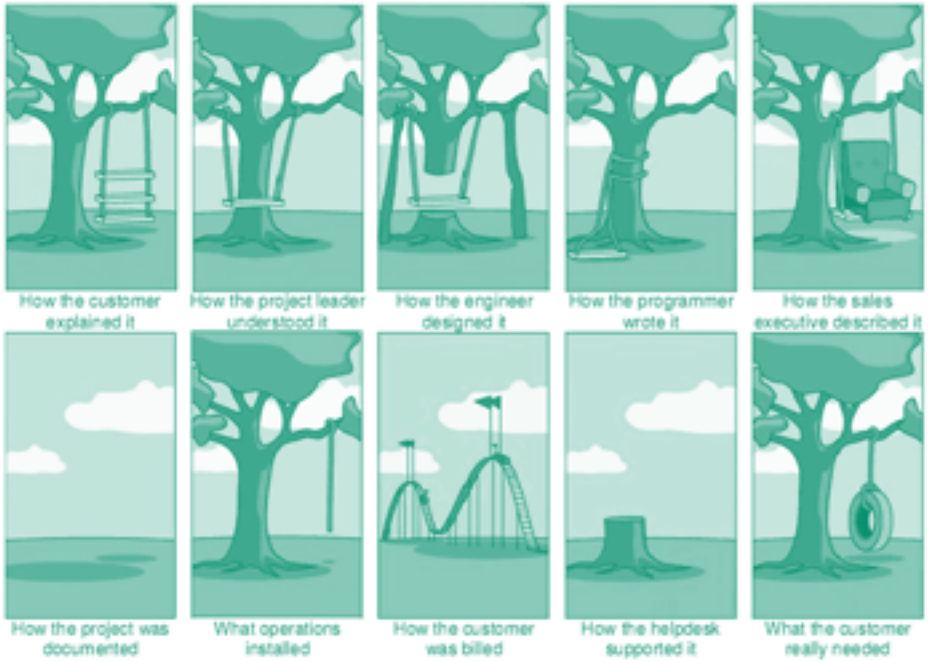
METHOD & ISSUES

- Determining what components and attributes made up the process
- Evaluate the process to develop an understanding of obstacles
- Assess the results in line with the organisation strategies and goals
- Monitor and evaluate the behaviour change in performance over a set period duration.

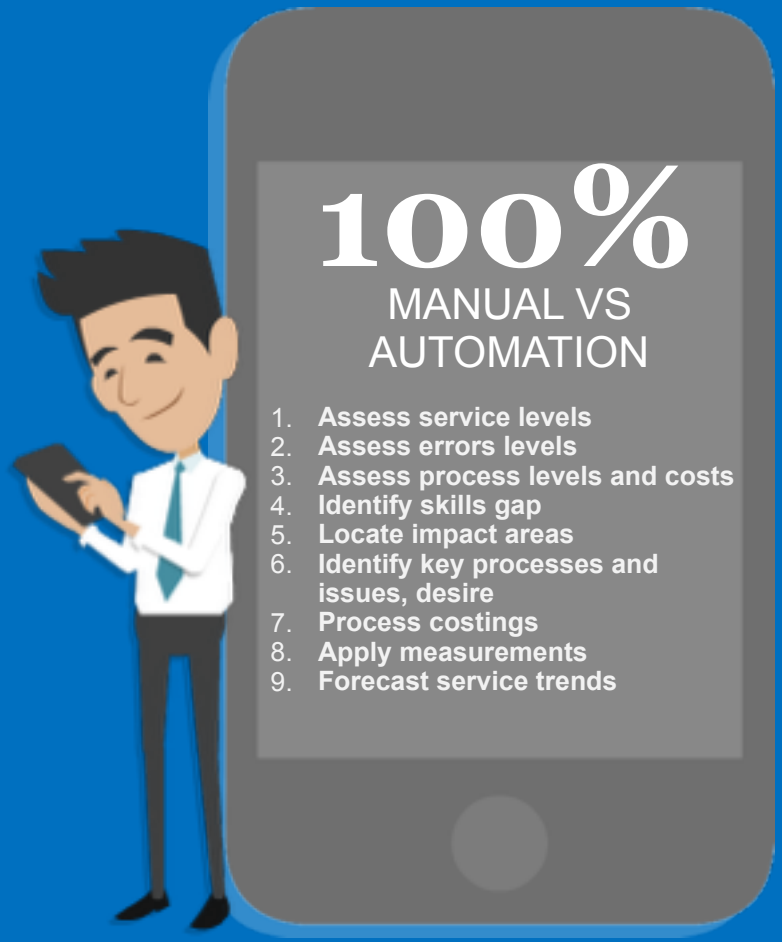
- general perception
- the adoption and process change
- the motivations to change
- the major factors influencing the initiation and the links of change
- obstacles and
- the benefits for both employees and the organisation.



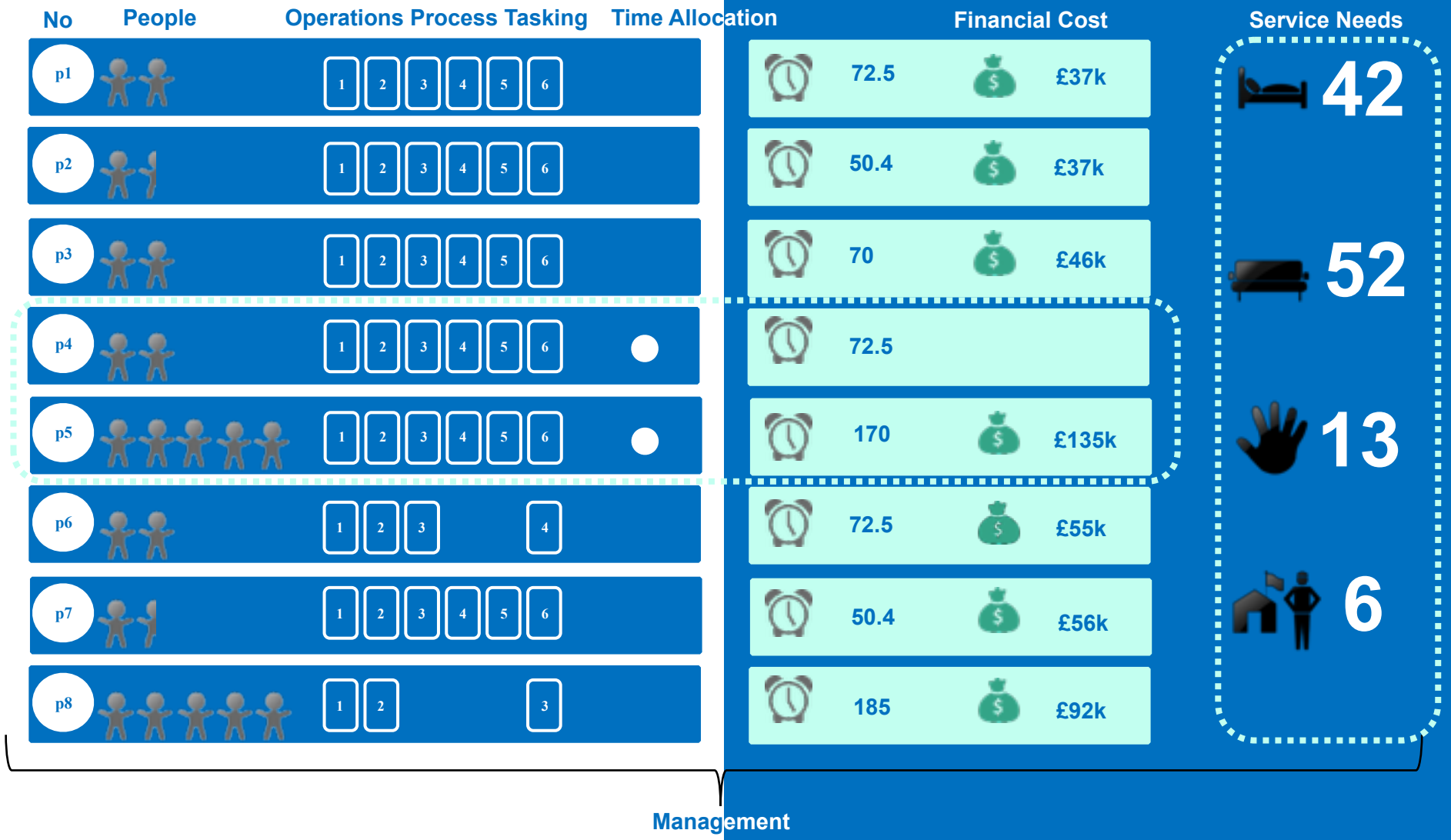
PRIORITIES OPTIMISATION



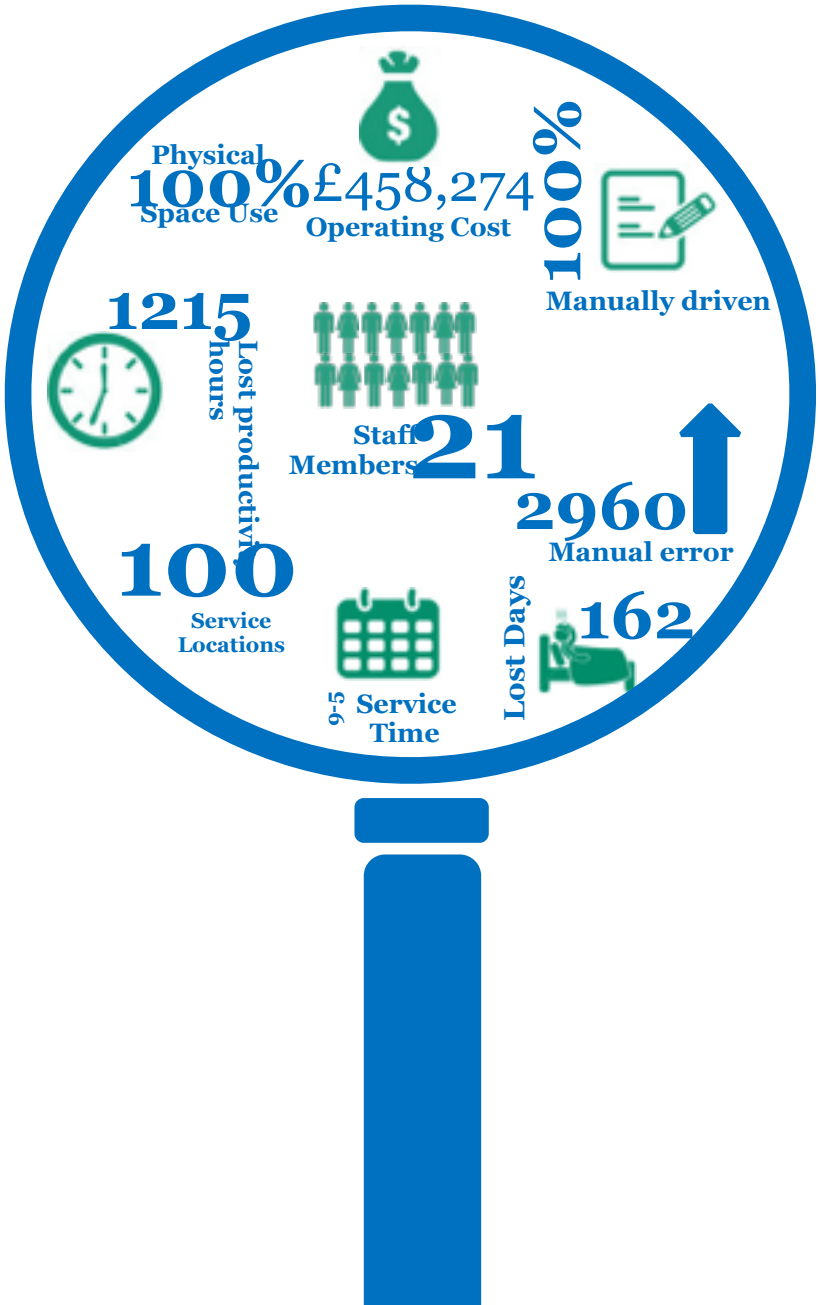
	Objectives	Pref 1	Pref 2	Number
1	*Decrease staff time*	*6*	*6*	••
2	*Increase provision of services with external*	*1*	*7*	•
3	*Reduce errors & manual work*	*7*	*1*	••••••••
4	*Reduce stock holding value*	*2*	*5*	••••
5	*Reduce wastage*	*3*	*4*	••••••••••••••••
6	*Improve stock control and rotation*	*4*	*2*	••••••••••••••
8	*Reduce number of absence*	*5*	*3*	••••••••



A LOOK AT THE PROCESS



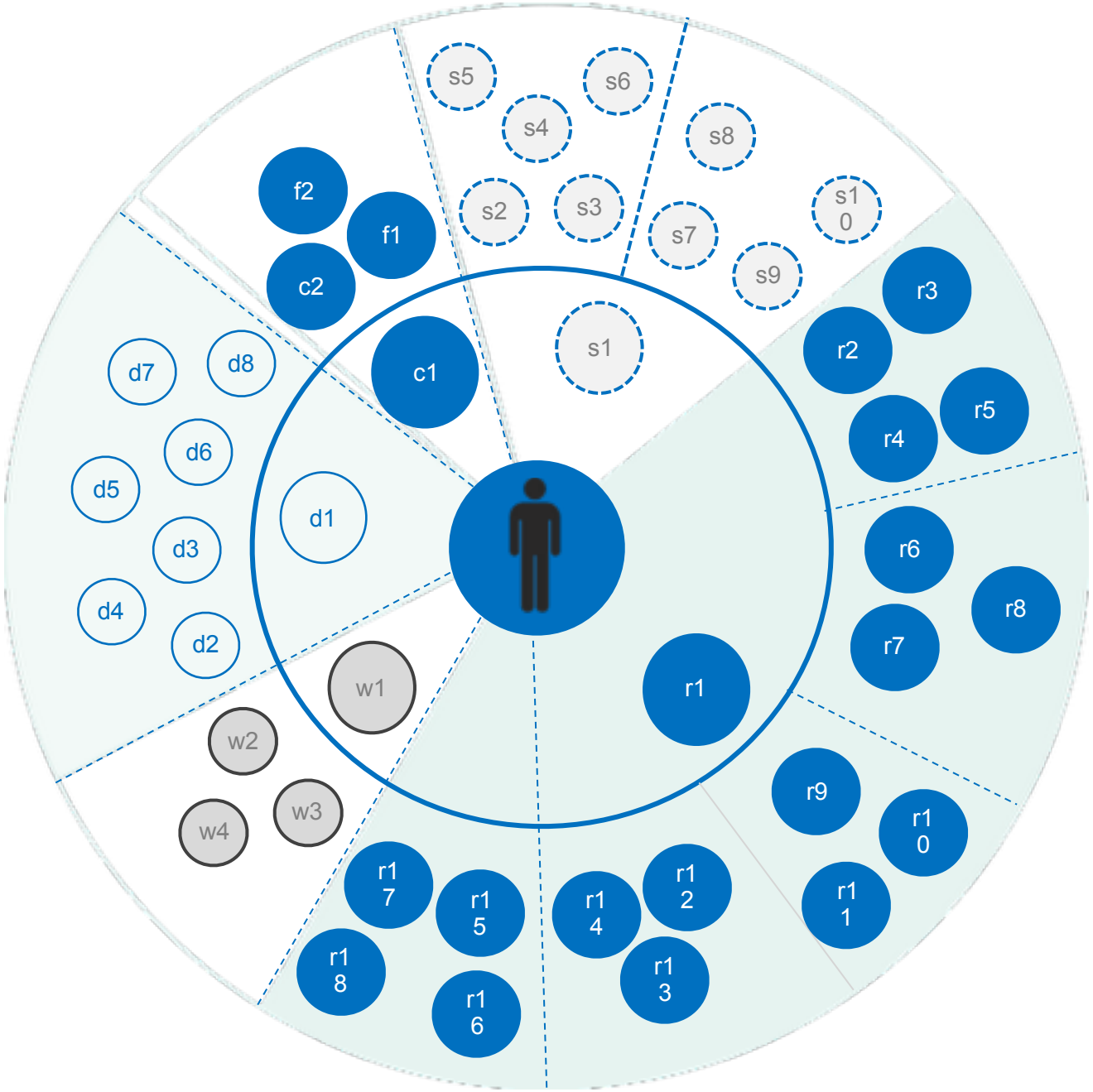
*



?

Image & Reputation	Service Portfolio & Innovation	Job Evolvement	Profit Maximisation	Customisation	Competition	Growth	The organisation service model	Complexity, knowledge, progress	Market growth, society

STAKEHOLDERS



RE-DESIGNING *for the* BIGGER PICTURE

TRADITIONAL
(manual)

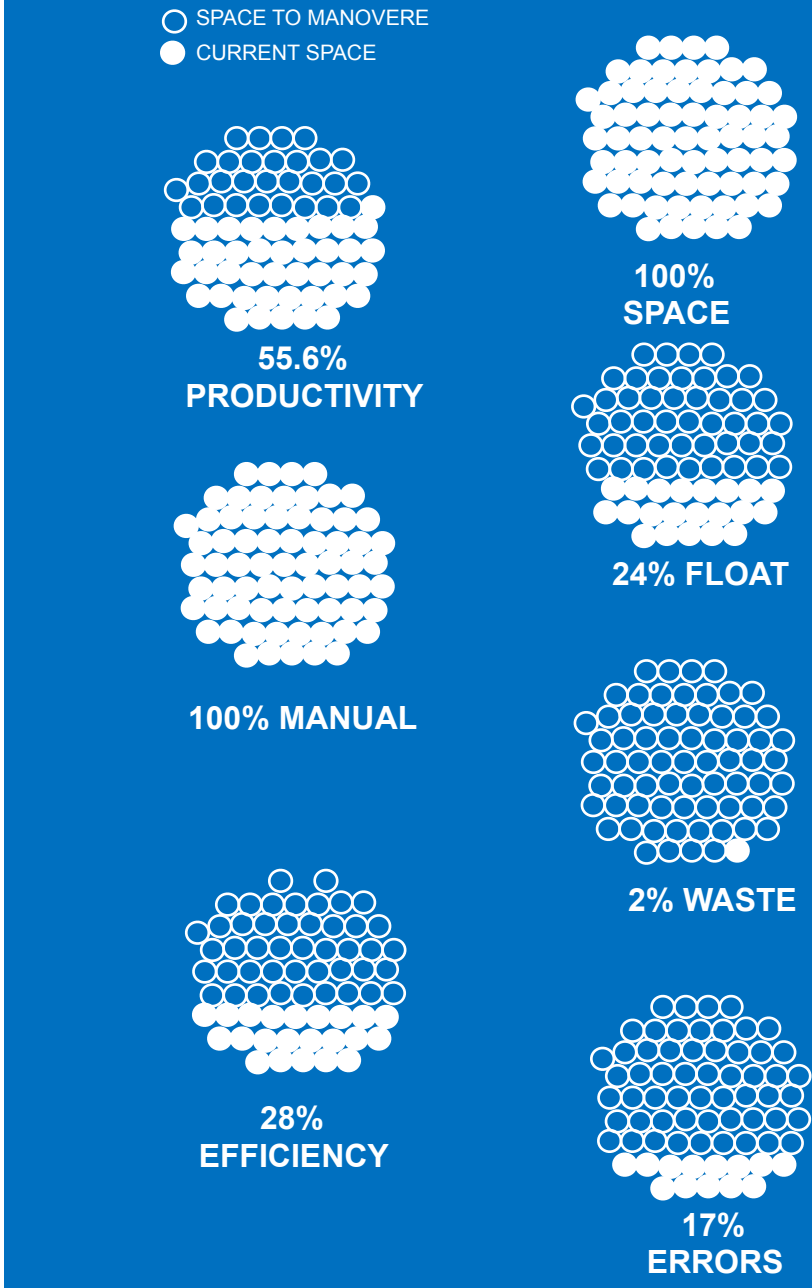
MODERN
(automation)

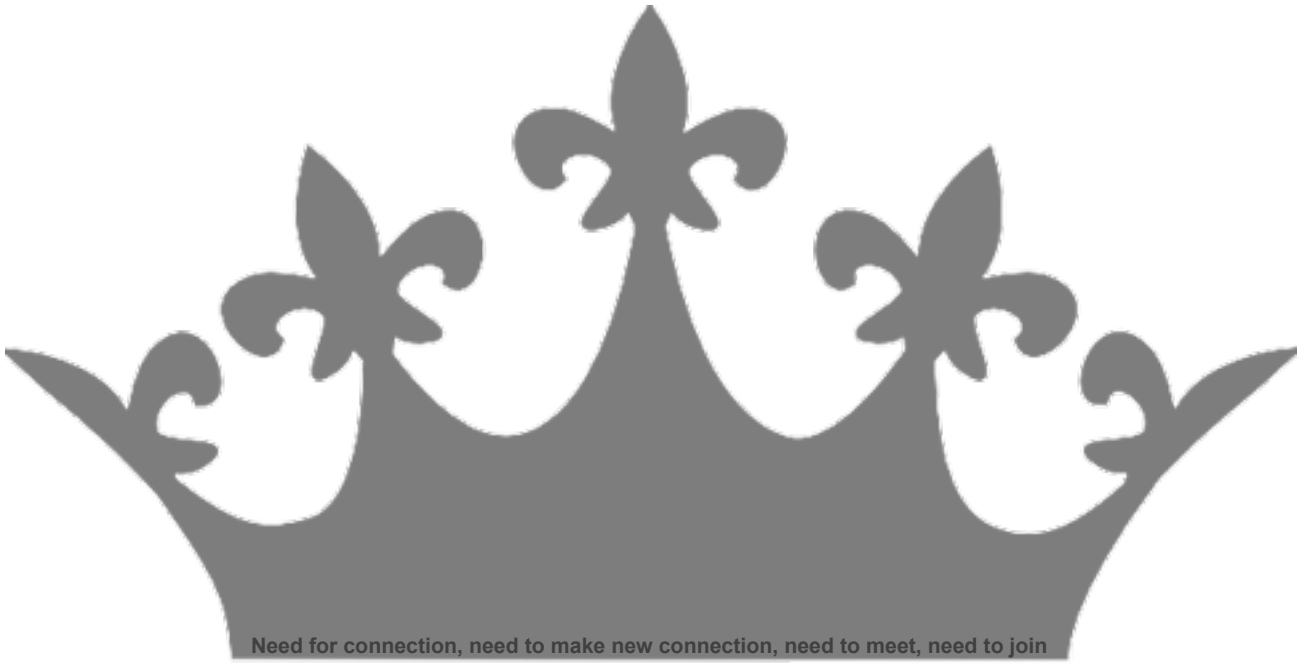
WORKLOAD QUANTIFICATION

Figures	Volume In	Volume Out	Float	Transactions
Yearly	*14,365,680*	*8,165,520*	*6,200,160*	*92880*
Monthly	*1,197,140*	*680,460*	*516,680*	*7740*
Weekly	*299,285*	*170,115*	*129,170*	*1935*
Daily	*59,857*	*34,023*	*25,834*	*387*

Figures	Errors	Recycle	Wastage	Loss days	Productivity (hours)
Yearly	*2960*	*2964*	*£29255.09*	*162*	*1215*
Monthly	*246.67*	*247*	*£2437.924*	*13.5*	*101.25*
Weekly	*61.66*	*61.75*	*£609.48*	*3.375*	*25.31*
Daily	*12.33*	*12.35*	*£121.89*	*0.7*	*5.06*

Budget	FY09	FY10	FY11	FY12	FY13
Amount (£m)	*93m*	*99m*	*105m*	*111m*	*117m*
% of waste	*0.6%*	*0.9%*	*1.4%*	*1.9%*	*2%*





COMMON / CONTEXT / ACTIVITY / QUALIFIER

REQUIREMENTS

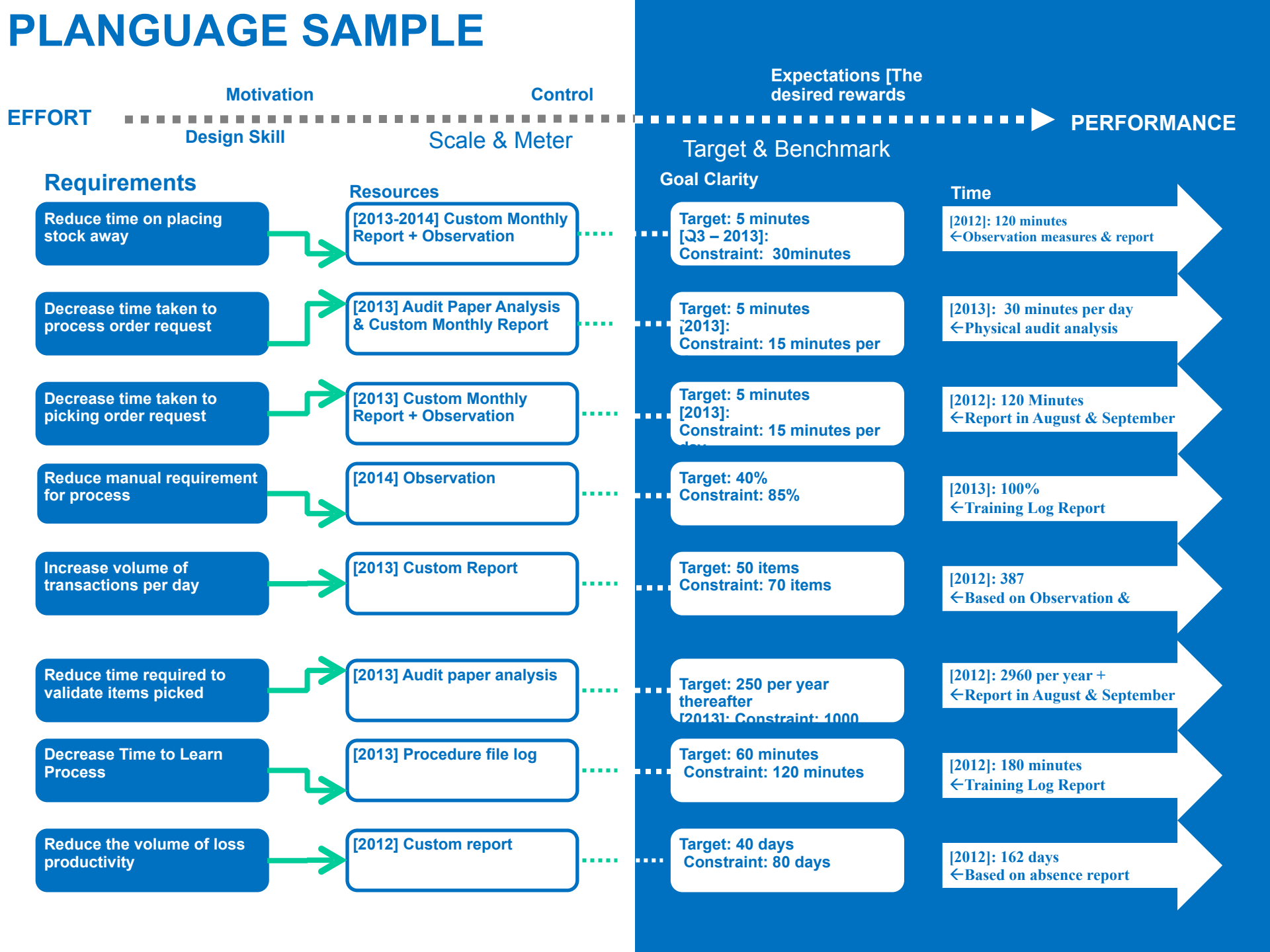
is

KING

REQUIREMENTS LOG

	Impacts [Functions]	Impacts [Intended performance requirements]	Impacts [Intended scale]	Impact Past	Impact Tolerable	Impact Goal
1	*Enter Content [Item] request*	*Efficiency.Effort Saving. Reduce Time for [user] stock away*	*Average time take taken for define [request type: default=user]*	*[<2013, OBS, User, 120 minutes]*	*30 minutes*	*5 minutes*
2	*Submit [Order] Request*	*Efficiency.Effort Saving. Reduce Time for [user] process order*	*Average time take taken for define [request type: default=user]*	*[<2013, OBS, User, 30 minutes]*	*15 minutes*	*5 minutes*
3	*Process[Order] Request*	*Efficiency.Effort Time Saving. Reduce [TIME] for [user] process order*	*Average time take taken for define [request type: default=manual collating]*	*[<2013, OBS, User, 120 minutes]*	*15 minutes*	*5 minutes*
4	*Process[Type]*	*Effort Time Saving. Process Reduce [%] for [user] process order*	*100% manual*	*[<2013, OBS, User, 100%]*	*40%*	*85% automated*
5	*Usability.[Transactions] Request*	*Average Number of [Transactions] per day*	*387 transactions*	*[<2013, CR, Transactions produced 387]*	*50 items*	*70 items*
6	*Uability.Reduce number of manual [Errors]*	*Average Number [Errors] of process*	*2960 per year*	*[<2013, CR, Errors produced 2960]*	*1000*	*250 per year thereafter decrease*
7	*Validate.[picking]*	*Efficiency.Elapse Time Saving. Reduce [TIME] to validate*	*Average time take for [User] to validate*	*[<2013, OBS, 50 minutes]*	*35 minutes*	*20 minutes*
8	*Productivity.[staff]*	*Average Number [days] loss productivity*	*162 days*	*[2012, CR, 162 days]*	*80 days*	*40 days*
9	*Distribution. [Accessibility]*	*Accessibility.Elapse Time Access*	*System Access Time*	*[<2012, INT, Open Time, 9am – 5pm]*	*9am – 9pm*	*Anytime*
10	*Cost.Reduction*	*Reduction.Average cost of waste*	*Average [waste]*	*[<2012, CR, Waste, 2% = £24k]*	*1%*	*0.5%*
11	*Time.Costing to [Process]*	*Cost.Cost saving. Reduce cost in process segment*	*Average [cost] process*	*[2014, RP, £150,640]*	*£90,000*	*£75,000*
12	*Time.[Learn]*	*Learn ability. Elapse Time learning. Reduce time on training*	*Average time taken for [request type: default=user] to learn process*	*[2013, INT, Learner, 1 day*	*3 hours*	*1 hour*
13	*Efficiency.[Space]*	*Efficiency.Space Saving. Increase [SPACE]*	*100% Space capacity*	*[2013, OBS, 100%]*	*60%*	*30%*
14	*OperatingProcess.[Cost]*	*Cost.Cost saving. Reduce cost in process segment*	*Operating [cost]*	*[2013,RP, £500k]*	*£400k*	*£360k*

PLANGUAGE SAMPLE



THANK YOU!

Questions?

Quantifying Music

Lean QA Audience at ACCU 2012 "Surely you cannot quantify 'Music' ?"

- I claimed
  - we can quantify any variable quality of any system
- I replied:
  - *I'll do it in a lightening talk here at ACCU*

What is the problem,
in quantifying music?

- Can you quantify this music?

Black-Eyed Peas song ”I gotta Feeling” gets 8.9 of 10 from Hit Song Science software

Frank Micelotta/Getty Image

The Black Eyed Peas' single "I Gotta Feeling" received a hit score of 8.9 out 10 with Music Intelligenc Solutions' new software Hit Song Science.

“There's no magic in that; it's math”

- "[It's] a series of **algorithms that we use**
  - to look at what's **the potential of a song**
  - to **be sticky with a listener ...**
- To have **those patterns in the music** that would
  - ***correspond*** with what **human brain waves would find pleasing**”

CEO David Meredith

- A study conducted by the Harvard Business School found that the software was accurate 8 out of 10 times.http://www.npr.org/templates/story/story.php?storyId=113673324

Measurable Attributes of Hits

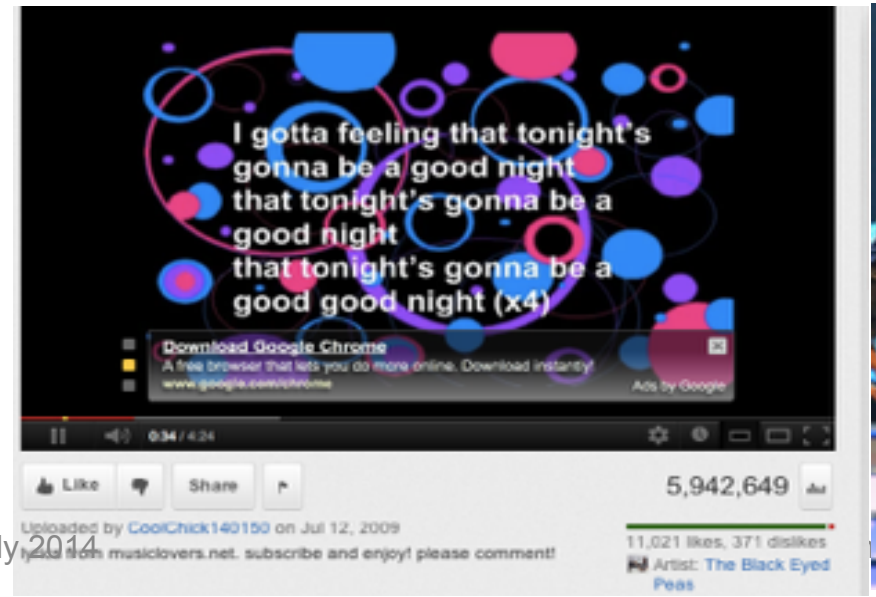
Meredith says his software evaluates songs over sixty elements including

Melody	**Rhythm**
Harmony	**Fullness of sound**
Tempo	**Noise**
Pitch	**Brilliance**
Octave	**Chord**
Beat	**progression**

http://edition.cnn.com/2008/WORLD/europe/03/07/spiritof.music/

YouTube Measures

- Number of Likes and Dislikes
  11,021 Likes, 371 Dislikes (April 26, 2012)
- Number of times video has been viewed
  5,942.649 Views (April 26, 2012)



By **Survey**: Most Wanted Attributes

- Yudkin reports on a web-based survey into American musical tastes conducted by Komar and Melamid in 1996
- If you want to please the greatest number of Americans (72% ± 12%) consider
  - Male and female solo voices
  - R&B with a love theme
  - Small ensemble of musicians
  - Length of about 5 minutes
  - Moderate pitch, tempo and volume

http://www.bu.edu/cfa/music/faculty/yudkin/

Most Unwanted Attributes

To appeal to only about 200 Americans

- Extreme length
- Wide range of dynamics, tempo and pitch in abrupt succession
- An operatic soprano singing atonally
- A cowboy song with political slogans
- A children's choir singing holiday songs
- Large orchestra featuring harp, accordion and bagpipes

http://www.bu.edu/cfa/music/faculty/yudkin/

There are samples of two songs written by David Soldier with lyrics by Nina Mankin to these wanted and unwanted guidelines about 19 minutes into Yudkin's lecture

Some potentially quantifiable Quality dimensions of Music

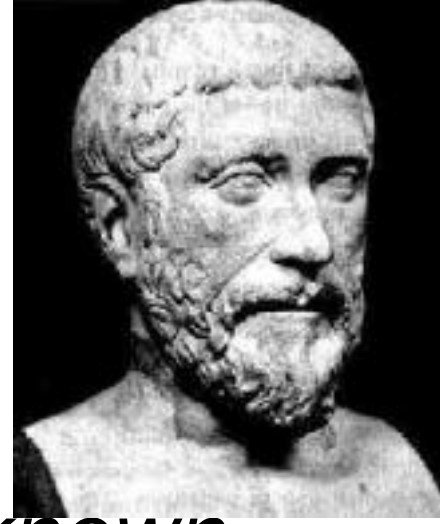
Brainstormed by Steve F. and Rachel D. At lunch

- In tune
- Applause
- Moving
- Encores
- Repeat Gigs
- Busking Hat Collection
- MRI Brain Scan
- Downloads
- Utube Reviews
- Royalties
- … (many more!!)

Examples in Planguage

- **Music.Moving:**
- **Type: primary music quality attribute**
- **Ambition Level: the majority of listeners feel moved to tears or strong physical emotional reactions.**
- **Scale: the % of defined [Listeners] hearing defined [Music] under defined [Environments] who reports a defined [Emotion] at a defined [Strength]**
- **Goal [1st UK Release, Music = Hip Hop**

Philolaus on Numbers



- Over four hundred years BC,
- a Greek by the name of
- **Philolaus** of Tarentum said :
- *" Actually, everything that can be known has a Number;*
  - *for it is impossible to grasp anything with the mind or to recognize it without this (number)."*

Best regards (Aug 2005),
N.V.Krishnawww.microsensesoftware.com

How to Quantify any Qualitative Requirement

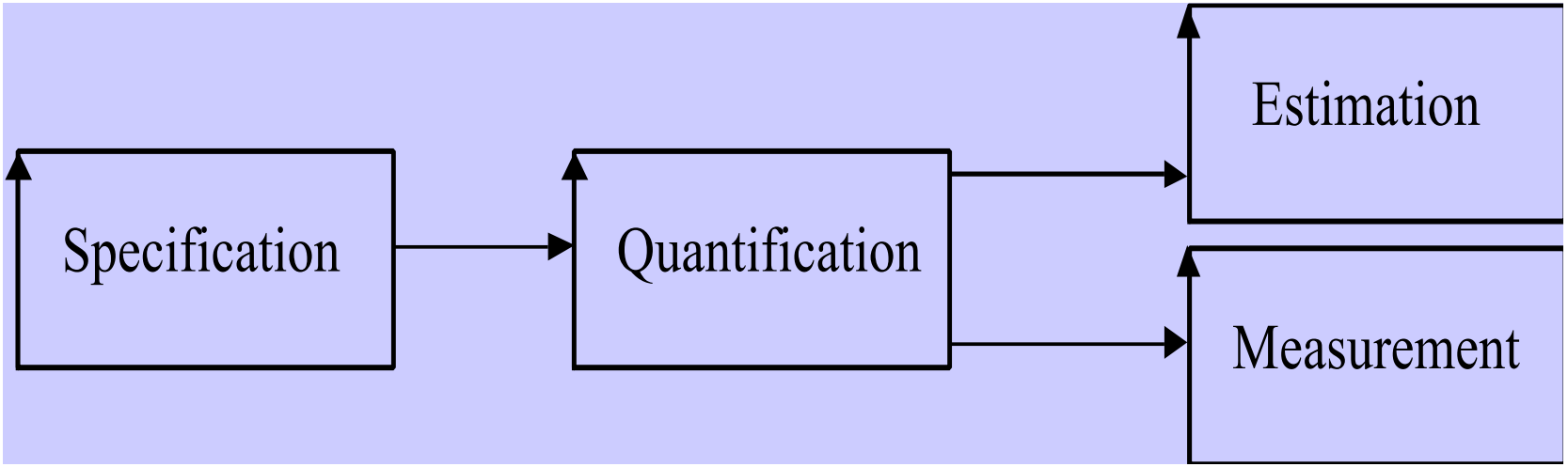


Diagram from 'Competitive Engineering.' book.

Quality Quantification Methods #1

- Common Sense, Domain Knowledge
  - Decompose “until quantification becomes obvious”.
  - Then use Planguage specification:
    - **Scale**: define a measurement scale
    - **Meter**: define a test or process for measuring on the scale
    - **Past**: define benchmarks, old system, competitors on the scale
    - **Goal**: define a committed level of future stakeholder quality, on your scale.

Maintainability:
Type: Complex Quality Requirement.
Includes: {Problem Recognition, Administrative Delay, Tool Collection, Problem Analysis, Change Specification, Quality Control, Modification Implementation, Modification Testing {Unit Testing, Integration Testing, Beta Testing, System Testing}, Recovery}.

Problem Recognition:
Scale: Clock hours from defined [Fault Occurrence: Default: Bug occurs in any use or test of system] until fault officially recognized by defined [Recognition Act: Default: Fault is logged electronically].
Administrative Delay:
Scale: Clock hours from defined [Recognition Act] until defined [Correction Action] initiated and assigned to a defined [Maintenance Instance].
Tool Collection:
Scale: Clock hours for defined [Maintenance Instance: Default: Whoever is assigned] to acquire all defined [Tools: Default: all systems and information necessary to analyze, correct and quality control the correction].
Problem Analysis:
Scale: Clock time for the assigned defined [Maintenance Instance] to analyze the fault symptoms and be able to begin to formulate a correction hypothesis.
Change Specification:
Scale: Clock hours needed by defined [Maintenance Instance] to fully and correctly describe the necessary correction actions, according to current applicable standards for this.
Note: This includes any additional time for corrections after quality control and tests.
Quality Control:
Scale: Clock hours for quality control of the correction hypothesis (against relevant standards).
Modification Implementation:
Scale: Clock hours to carry out the correction activity as planned. "Includes any necessary corrections as a result of quality control or testing."
Modification Testing:
Unit Testing:
Scale: Clock hours to carry out defined [Unit Test] for the fault correction.
Integration Testing:
Scale: Clock hours to carry out defined [Integration Test] for the fault correction.
Beta Testing:
Scale: Clock hours to carry out defined [Beta Test] for the fault correction before official release of the correction is permitted.
System Testing:
Scale: Clock hours to carry out defined [System Test] for the fault correction.
Recovery:
Scale: Clock hours for defined [User Type] to return system to the state it was in prior to the fault and, to a state ready to continue with work.

Source: This above is an extension of some basic ideas from Ireson, Editor, Reliability Handbook, *McGraw Hill, 1966* (Ireson 1966).

Quality Quantification Methods #**2**, **Look it up in a book**

Chapter
5

SCALES OF MEASURE

How to Quantify

Maintainability:
Type: Complex Quality Requirement.
Includes: {Problem Recognition, Administrative Delay, Tool Collection, Problem Analysis, Change Specification, Quality Control, Modification Implementation, Modification Testing {Unit Testing, Integration Testing, Beta Testing, System Testing}, Recovery}.

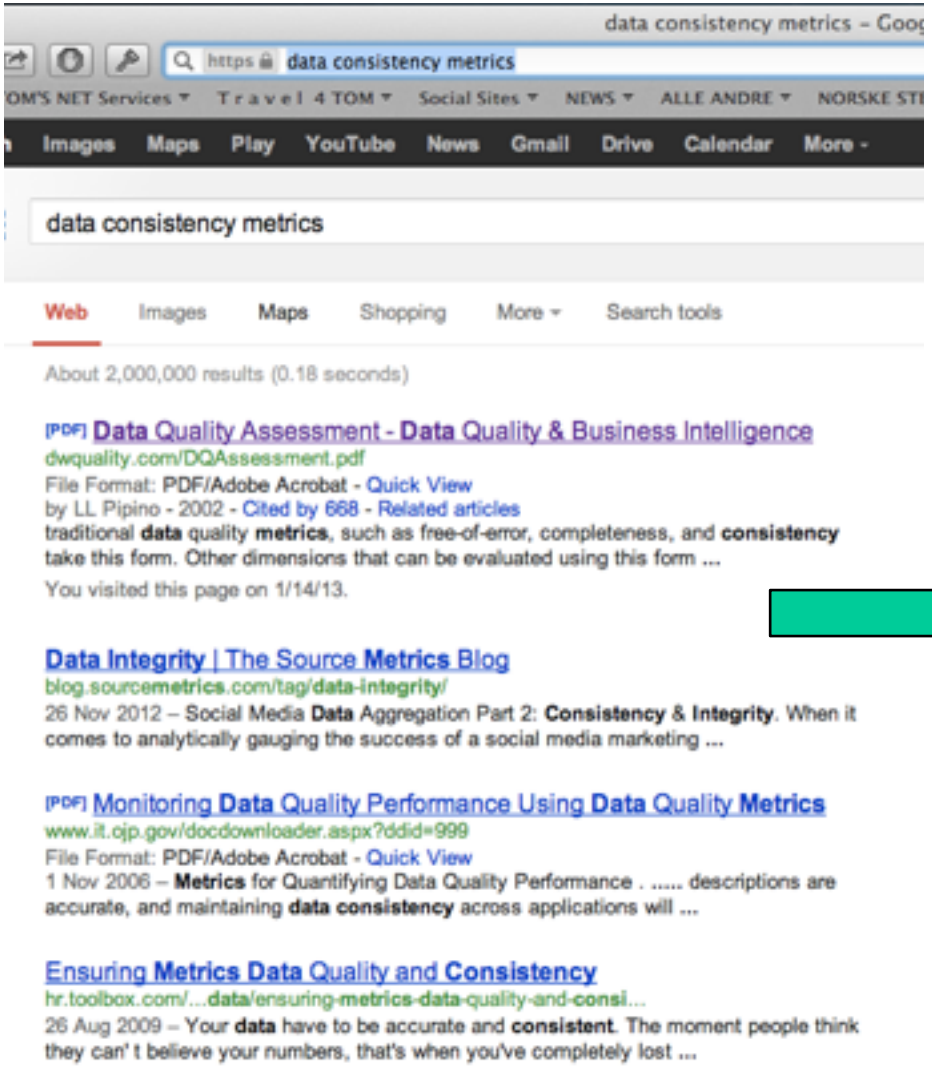
Problem Recognition:
Scale: C
system]
electron
Adminis
Scale: C
assigne
Tool Co
Scale: C
acquire
and qua
Problem
Scale: C
toms an
Change
Scale: C
the nece
Note: Th
Quality
Scale: C
Modific
Scale: C
correctio
Modific
Unit T
Scale:
Integr
Scale:
Beta T
Scale:
releas
Syste
Scale:
Recove
Scale: C
fault and, to a state ready to continue with work.

Source: This above is an extension of some basic ideas from Ireson, Editor, Reliability Handbook, *McGraw Hill, 1966* (Ireson 1966).

Quality Quantification Methods #**2**,
Look it up in a book

Tool Collection:
Scale: Clock hours for defined [Maintenance Instance: Default: Whoever is assigned] to acquire all defined [Tools: Default: all systems and information necessary to analyze, correct and quality control the correction].

Quality Quantification Methods #**3**, Google It



dwquality.com/DQAssessment.pdf

e 1. Data quality dimensions.

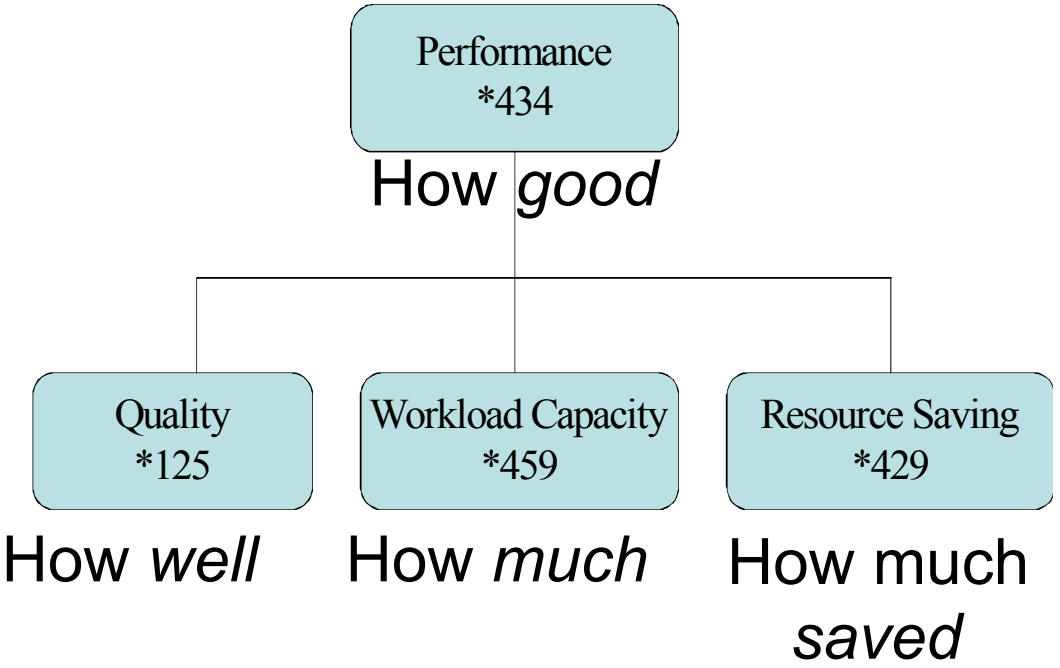
Dimensions	Definitions
Accessibility	the extent to which data is available, or easily and quickly retrievable
Appropriate Amount of Data	the extent to which the volume of data is appropriate for the task at hand
Believability	the extent to which data is regarded as true and credible
Completeness	the extent to which data is not missing and is of sufficient breadth and depth for the task at hand
Concise Representation	the extent to which data is compactly represented
Consistent Representation	the extent to which data is presented in the same format
Ease of Manipulation	the extent to which data is easy to manipulate and apply to different tasks
Free-of-Error	the extent to which data is correct and reliable
Interpretability	the extent to which data is in appropriate languages, symbols, and units, and the

Quality: the concept, the noun

Planguage Concept *125, Version: March 20, 2003

A '**quality**' is

- a scalar attribute -|-|-|-|- (Scale symbol)
- reflecting 'how *well*' ------Past Level<---------->
- a system functions. (Fn)------Past Level<-------->



Quality *is characterized by these traits (from CE book)*

1. **Quality describes 'how well' a function is done.**
2. **Quality describes the *partial effectiveness* of a function (as do all other performance attributes).**
3. **Quality is *valued* to *some* degree by *some* stakeholders of the system**
4. ***More* quality is generally *valued* by stakeholders; especially if the increase is free, or lower cost, than the value of the increase.**
5. **Quality attributes can be *articulated* independently of the particular means (designs) used for reaching a specific quality level –**
6. **even though all quality levels *depend* on the particular designs used to achieve them.**
7. **A particular quality can be a described in terms of a *complex* concept, consisting of multiple elementary quality concepts.**
8. **Quality is *variable* (along a definable scale of measure: as are all scalar attributes).**
9. **Quality levels are capable of being specified *quantitatively* (as are all scalar attributes).**
10. **Quality levels can be *measured* in practice.**
11. **Quality levels can be traded off to some degree; with other system attributes valued more by stakeholders.**
12. **Quality can never be perfect (100%), in the real world.**
13. **There are some levels of a particular quality that may be outside the state of the art; at a defined time and circumstance.**
14. **When quality levels increase towards perfection, the resources needed to support those levels tend towards infinity.**

Quality *is characterized by these traits*

1. Quality describes 'how well' a function is done.
2. Quality describes the *partial effectiveness* of a function (as do all other performance attributes).
3. Quality is *valued* to *some* degree by *some* stakeholders of the system
4. *More* quality is generally *valued* by stakeholders; especially if the increase is free, or

9. Quality levels are capable of being specified *quantitatively* (as are all scalar attributes).

11. Quality levels can be traded off to some degree; with other system attributes valued more by stakeholders.
12. Quality can never be perfect (100%), in the real world.
13. There are some levels of a particular quality that may be outside the state of the art; at a defined time and circumstance.
14. When quality levels increase towards perfection, the resources needed to support those levels tend towards infinity.

Love Quantification

a 4.5 minute lightening Talk at ACCU Conference, Oxford April 15 2010

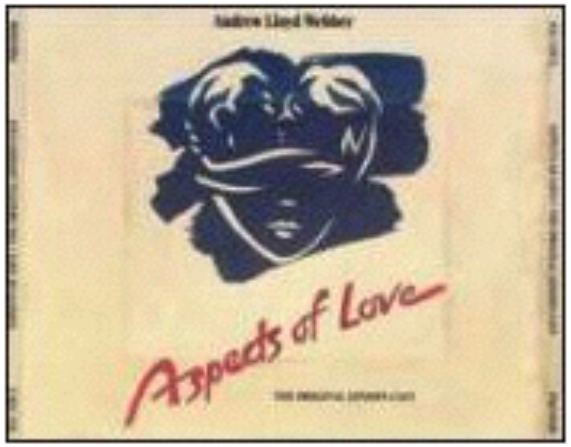
Class Exercise: Aspects of Love, or Love is a many splendored thing!

- METHOD
  - Make a list of love's many aspects
  - Quantify *one* random requirement, for love
    - To show that all of the aspects can be similarly quantified

Love Attributes:
Brainstormed By Dutch Engineers

- **Kissed-ness**
- **Care**
- **Sharing**
- **Respect**
- **Comfort**
- **Friendship**
- **Sex**
- **Understanding**
- **Trust**

- Support
- Attention
- Passion
- Satisfaction
- ...
- ...
- ...



Trust Defined

- **Love.Trust.Truthfulness**

  Ambition: No lies.

  Scale:

  Average **Black** lies/month from [defined sources].

  Meter:

  independent confidential log from sample of the defined sources.

  Past Lie Level:

  Past [My Old Mate, 2004] 42 <-Bart

  Goal

  [My Current Mate, Year = 2005] Past Lie Level/2

  Black: Defined: Non White Lies

- Other aspects of Trust:
- 1. 'Truthfulness'
  2. **Broken Agreements**
  3. **Late Appointments**
  4. **Late delivery**
  5. **Gossiping to Others**

Camaraderie (Real Case UK)

Ambition: *to maintain an exceptionally high sense of good personal feelings and co-operation amongst all staff: family atmosphere, corporate patriotism. In spite of business change and pressures.*

Scale: **probability that individuals enjoy the working atmosphere so much that they would not move to another company for less than 50% pay rise**.

Meter: Apparently real offer via CD-S

Past [September 2001] 60+ % <- R & CD

Goal [Mid 2002] 10%, [End 2002] **<1%** <- R & CD

Rationale:

maintain staff number, and morale as core of business and business predictability for customers.

My ‘Christian’ Friend

- Lawrence Day. Seattle Washington
- “Love is not quantifiable”
  - Not in Bible
  - Little guidance from God and Jesus

Love: Biblical Dimensions
<- Lawrence Day, Boeing

The biblical citation (Book of First Corinthians, Chapter 13) I included gives the quantification of the term "love" (agape in Greek). The 'quantification' for love would be as follows:

---------->

A person who loves acts the following way toward the person being loved:

1. **suffereth long**
2. **is kind**
3. **envieth not**
4. **vaunteth not itself, vaunteth...:**
   or, is not rash (Vaunt = extravagant self praise)
5. **is not puffed up**
6. **Doth not behave itself unseemly**
7. **seeketh not her own**
8. **is not easily provoked**
9. **thinketh no evil**
10. **Rejoiceth not in iniquity (=an unjust act)**
11. **rejoiceth in the truth**
12. **Beareth all things**
13. **believeth all things**
14. **hopeth all things**
15. **endureth all things**
16. **never faileth**

A Paper on ‘Love Quantified’

http://www.gilb.com/tiki-download_file.php?fileId=335

Love Quantified

By:

Lawrence E. Day

for

Dr. Larry Beebe

And

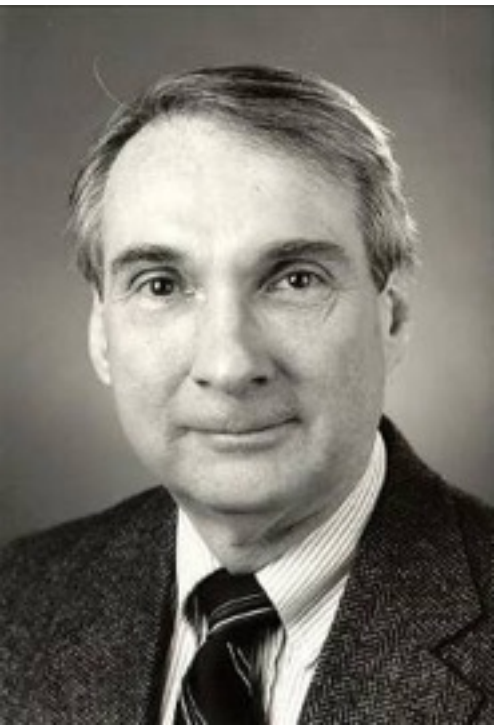
Dr. Raghu Korrapati

Table of Cor

Mathematical Models of Love & Happiness

http://sprott.physics.wisc.edu/ lectures/love&hap/

(This talk)



J. C. Sprott

Department of Physics

University of Wisconsin - Madison

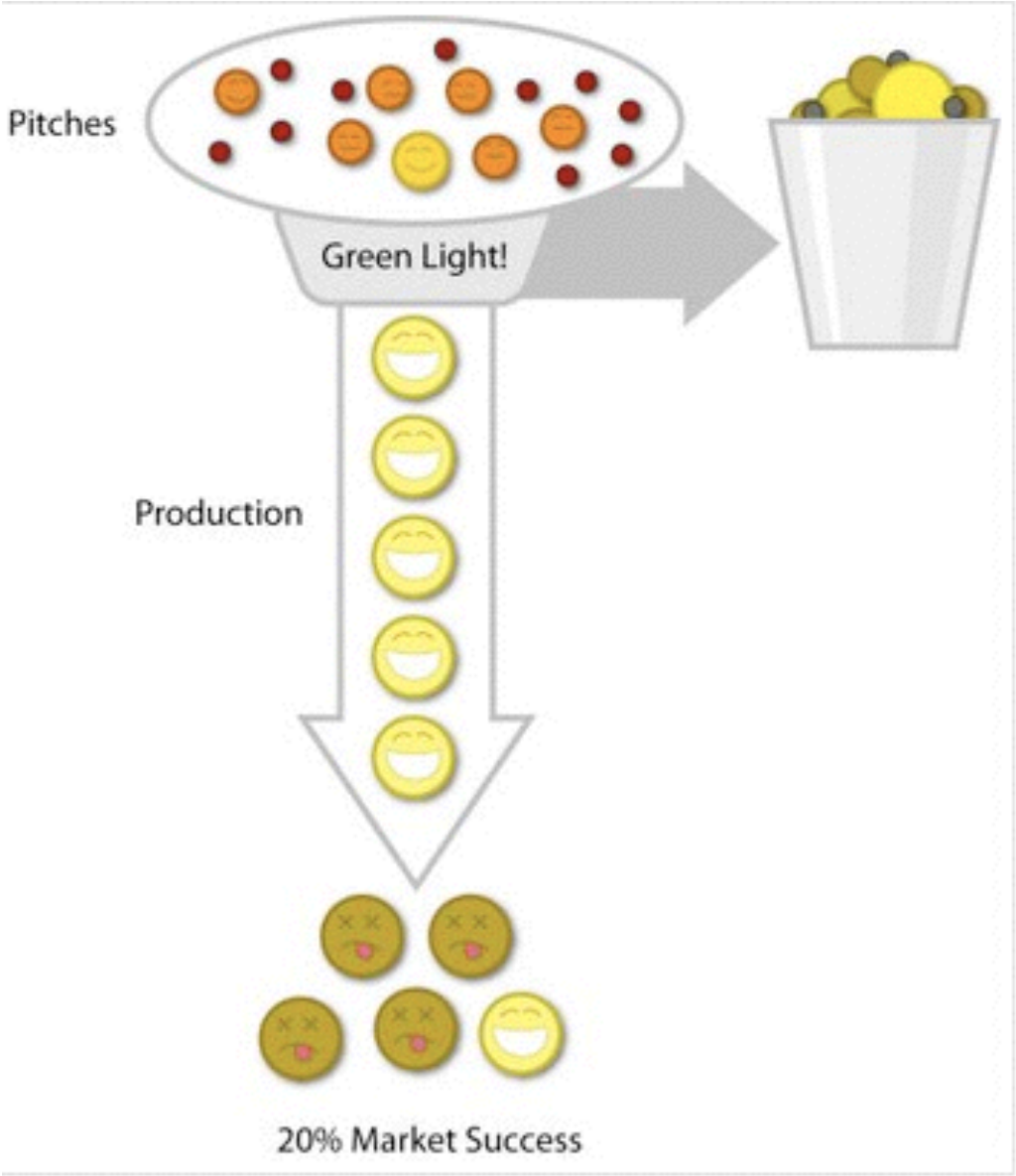
Presented to the

Chaos and Complex Systems Seminar

in Madison, Wisconsin

on February 6, 2001

Horror Project Requirements Case

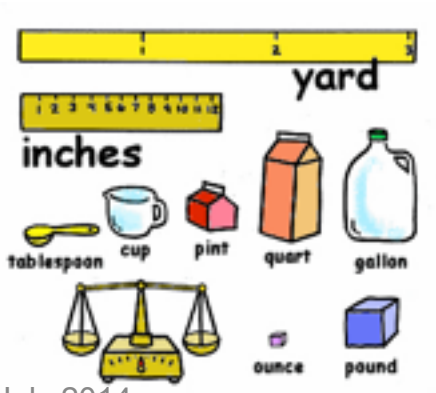


Based On Real Case 2006-8

Summary of Top '8' Project Objectives

Real Example of ***Lack*** of Scales

- **Defined** Scales of Measure:
  - Demands ***comparative* thinking**.
  - Leads to requirements that are unambiguously **clear**
  - Helps Team be **Aligned** with the Business



*1. Central to The Corporations business strategy is to be the world's **premier** integrated_ <domain> service **provider**.*

*2. Will provide a much more efficient **user** experience*

*3. Dramatically scale back the **time** frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to **generate** the desired **products***

*4. Make the system much **easier** to **understand** and **use** than has been the case for previous system.*

*5. A primary goal is to provide a much more **productive** system **development** environment than was previously the case.*

*6. Will provide a richer set of functionality for **supporting** next-generation logging **tools** and applications.*

*7. **Robustness** is an essential system requirement (see rewrite in example below)*

*8. Major improvements in **data quality** over current practices*

This lack of clarity cost them $100,000, 000

The Lesson

- If management does not clarity the main reasons for a software development project, QUANTITATIVELY,
- It can cost $100,000,000+ and 8 years of wasted time

What the Project Manager Wanted after $160,000,000* was spent

"Able to add features *without fear* …
Able to improve code *without fear* …
Able to incorporate improved technology *without fear* …
Able to rapidly adapt to changing requirements …

Code that's easy to maintain …
Code that's uniform, easy to understand …
Code that's readily and thoroughly testable …"

* The number was sometimes quoted at $100 million, and by 2008 it was certainly much higher, no deliveries had taken place by May 2008.

What the CIO Director Told Me

"In 1998 I voted to vete this project start because the requirements were insufficient.

But I was overruled by the other directors (including the current CEO)"

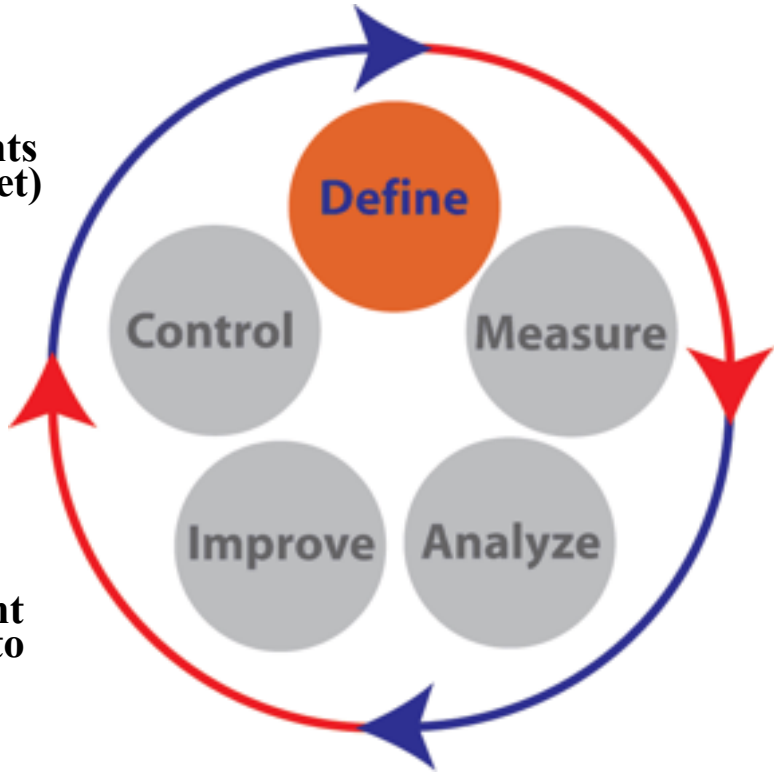
Main Hypothesis by Gilb:

1. **The requirements are unacceptably unclear.**
2. **The project has proceeded to throw masses of detail ('design') at the unacceptably unclear requirements.**
3. **There is no objective way to decide if any of the built or planned detail is necessary or sufficient to meet the unclear requirements.**
4. **There is no point whatsoever in continuing the project on this basis (the bad requirements).**

   Because there is no way to determine if the project is progressing towards any reasonable goals.

Suggested Practical Actions for HORROR Project.

1. **Stop all HORROR Project Effort based on the old plans**
2. **Adopt a new 'policy' for running this project**
3. **Quickly (in a week or 2) rewrite the top level requirements.**
    1. **Review the current business and technical environment to see if new and different requirements are more appropriate than the current (3.13 2003 set)**
    2. **Quantify all the top few objectives**
    3. **Estimate the value of reaching the objectives**
    4. **Get the objectives approved by top management**
        1. **This is not the same as project funding approval.**
        2. **It just says we would value reaching these objectives**
        3. **And we don't know of any better ones.**
4. **Let a 'qualified' system architect decide the best way to deliver the results.**
    1. **The big question is how much, if any of the current HORROR project investment can be applied, and to what degree the results need to be evolved into the current customer product and environment.**
    2. **Approve the architecture**
5. **Don't ever pour money into the project unless real measurable improvements are promised and delivered in short cycles.!**



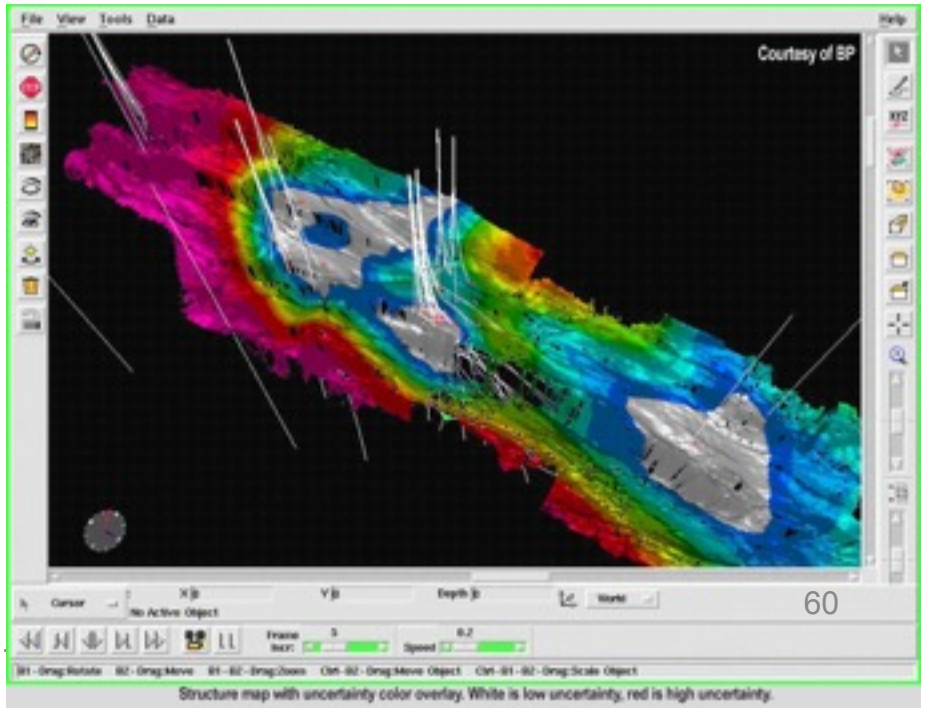
1. Seamless ROCKfield data and workflow

Central to THE CORPORATION's ROCKfield business strategy is to **be the world's premier INTEGRATED** ROCKfield service provider. Software is a key enabling technology towards providing this integration. As an active contributor to this overall strategy, Horror will provide the following:

Broad MINESITE data coverage.

Horror will be able to tap a **broad variety** of data about the well and its environment. Each of the Horror products will be able to store and exchange all of the following data types, e.g. wireline will be able to access MINING data, etc. These data types include:

- *GILB COMMENT: There is no attempt to define '**seamless**' quantitatively so that we can **measure** and **track** the final **result**.*
- *The content of the rest of the requirement is an equally vague set of functional requirements (like "will support standard Windows OLE compound document functionality").*
- *It is not at all clear how well these things will be done (no performance or quality requirements for these are mentioned.*
- *The result is likely to be that the function is there but has substandard user quality and performance.*
- *We need to define the user experience – how fast, how easy.*
- *We need to define the end state that would make us the worlds premier provider.*
- *We have not even got close to it.*



2. Dramatic boost in operational efficiency

HORROR will provide a

much **more efficient user experience**

by

automating a number of routine activities

and by **removing restrictions on when or how a number of activities may be performed.**

These improvements include:

As-you-go product generation HORROR will provide the following features

to **dramatically scale back the time** frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to generate the desired products

by

semi-automating and/or performing these activities as the data comes in.

GILB ANALYSIS:

- ***There is no unambiguous definition of 'operational efficiency' (no defined Scale or Scales of measure).***
- ***There is no defined level on that (undefined) scale that tells us what is Dramatic (and when it is dramatic (short term levels, longer term levels, competitor levels). Goal, Stretch, Trend levels to use Planguage terms.***
- ***The 'efficient user experience' is not at all defined in terms quantified***
- ***In short this requirement completely fails, where is could have easily succeeded (in 1998) to specify the level of operational efficiency that the product would measurably achieve.***
- ***The rest of the specification with features like***

'Automated depth adjustment for data acquired since last deviation survey'

are merely suggested design elements,

that will only contribute to the operational efficiency if they are well designed and implemented to a defined level of impact on

the (yet undefined quantified definition of operational efficiency).

These design ideas do not belong here at all (this applies to all the requirements at this level).

They should be in a separate architecture or design specification, that suggested appropriate designs for

3. Much easier to understand and use

A critical requirement for HORROR's success is to make the software much easier to understand and use than has been the case for previous CORPORATION MINE software.

Benefits of this requirement **include**

reduced training time, **better utilization** of system features

and fewer operational errors.

As an aid in achieving this objective, HORROR has adopted a new use-case centric development process,

which makes the users and their use of the system a focal point of the development

The intent is to design for and evaluate usability continually during the development process rather than fixing it at the end.

(And it goes on about processes and designs)

- **Gilb Comment**: essentially same criticism as above. This concept could be defined quantitatively (See Usability, Gilb CE Chapter 5, www.gilb.com download).
- **'To understand'** needs definition (scale) and '**much easier'** needs specification of numeric points on the scale for various users and tasks.
- The rest of the requirement makes the systemic mistake of diving into **specific design detail ("Minimized panes., Docked and undocked panes, Product generation console"** for example).
- These are badly defined, and badly justified designs for an undefined problem.
- We would end up building them into the system and there is no guarantee that we would end up getting the 'operational efficiency' we need (since we have not even decided what we want!).

"A primary goal of HORROR is to provide a **much more productive software development environment than was previously the case.**

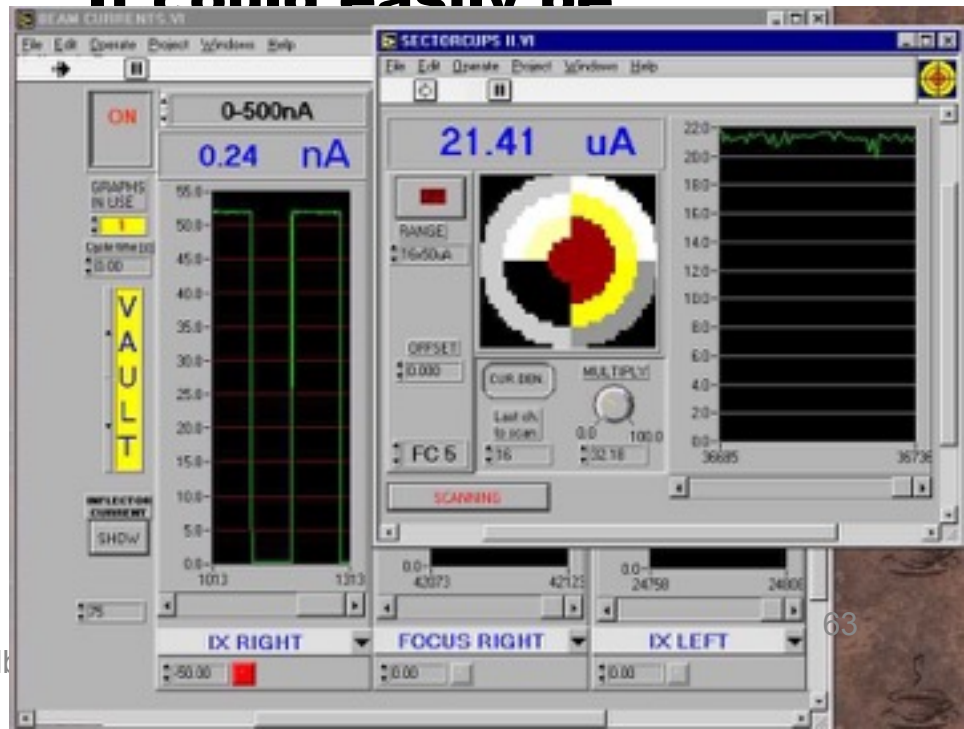
- In addition to traditional software development by professional software personnel,
  - –this goal is aimed at **facilitating the development of exploratory or custom software or reports by personnel such as tool or interpretation algorithm developers whose software expertise is more modest.**
- A related aspect of this goal is that the **software development difficulty should scale,**
  - – i.e. simple applications should **be easy to develop.**

GILB COMMENT:

- **SAME COMMENTS AS ABOVE**
- **The Major concept (Productivity) is NOT defined.**

No level of productivity is numerically and testably set.

It could easily be



5. Rich support for next-generation tools and applications

"HORROR will provide

– a richer set of functionality

– for supporting

•next-generation logging tools

• and applications.

Provided features include:

Richer equipment model

HORROR will

•provide a

– richer equipment model that

– better fits modern hardware configurations.

•*GILB COMMENT:*

– **Total lack of quantified definition of what this "Supportability"** is.

•It could easily be defined as a clear quantified requirement.

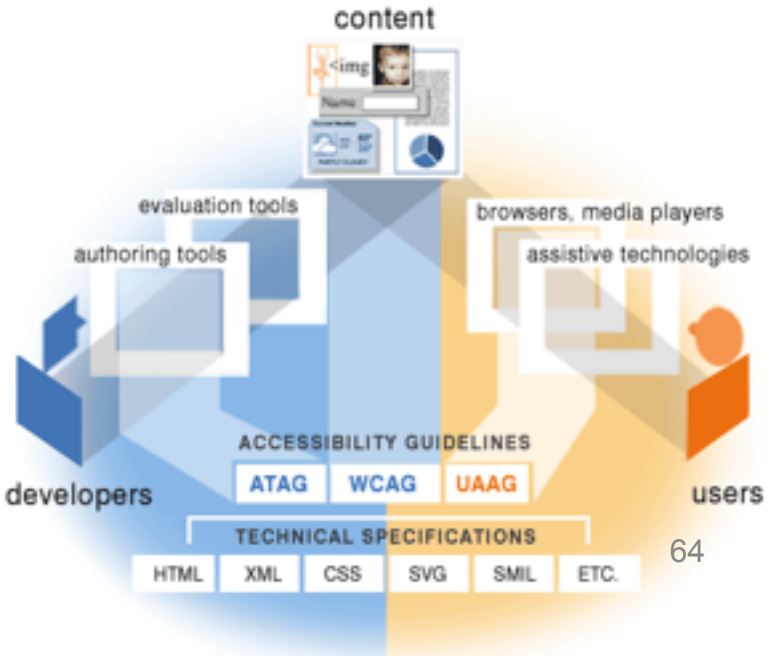
– Masses of ***nice sounding gratuitous design ideas***

–**unjustified** in relation to the (**undefined**) requirement.

– A license to keep on **implementing all these things endlessly**

– with no end in sight

–and no **responsibility** for costs or effects.



6. Rock solid robustness

- *While **robustness** is an **essential** HORROR requirement in all its uses, it is especially critical in MINING applications where the much longer job durations afford software defects (e.g. memory leaks) a greatly expanded opportunity to surface.*
- In this regard,
- HORROR will provide the following features or attributes:
  - ***Minimal down-time***
- A critical HORROR objective is to have **minimal downtime** *due to* **software failures**.
- This objective includes:
  - **Mean time between forced restarts > 14 days**
- HORROR's goal for mean time between forced restarts **is greater than 14 days**.
- *Comment: This figure does not include restarts caused by hardware problems, e.g. poorly seated cards or communication hardware that locks up the system. MTBF for these items falls under the domain of the hardware groups.*
  - **Restore system state < 10 minutes**
- Log scripts and test scripts, subsystem tests
  - ***Built-in testability***
- HORROR will provide the following features and attributes to facilitate testing.
  - **Tool simulators**

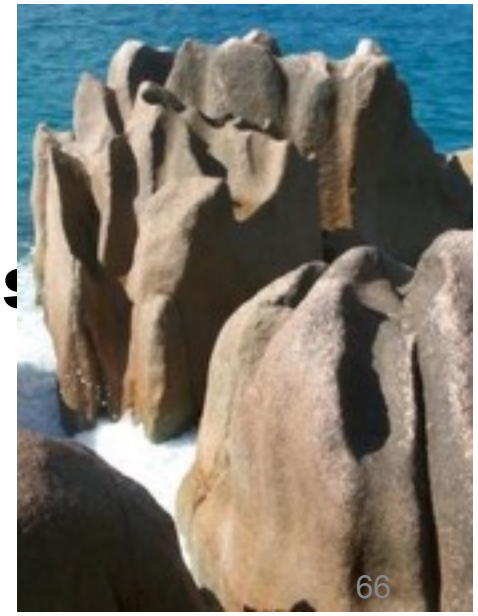
- *GILB COMMENT:*
  - *For once a reasonable attempt was made to quantify the meaning of the requirement!*
  - *But is could be done much better*
  -
  - *As usual the **set of designs** to **meet the requirement** do not belong here.*
  - *And none of them make any **assertion** about how well (to what degree) they will meet the defined numeric requirements.*
  - *And as usual another guarantee of eternal costs on pursuit of a poorly defined requirements is most of the content.*

"Rock Solid Robustness"
Defined Clearly in Planguage over a beer

Rock Solid Robustness:

Type: Complex Product Quality Requirement.

Includes: { Software Downtime, Restore Speed, Testability, Fault Prevention Capability, Fault Isolation Capability, Fault Analysis Capability, Hardware Debugging Capability}.



Software Downtime:

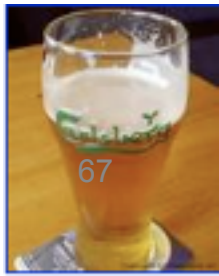
Software Downtime:
Type: Software Quality Requirement.
Ambition: *to have minimal downtime due to software failures <- HFA 6.1*
***Issue**: does this not imply that there is a system wide downtime requirement?*

Scale: <mean time between forced restarts for defined [Activity], for a defined [Intensity].>

Fail [Any Release or Evo Step, Activity = Recompute, Intensity = Peak Level] **14 days** <- HFA 6.1.1

Goal [By 2008?, Activity = Data Acquisition, Intensity = Lowest level] **: 300 days** ??

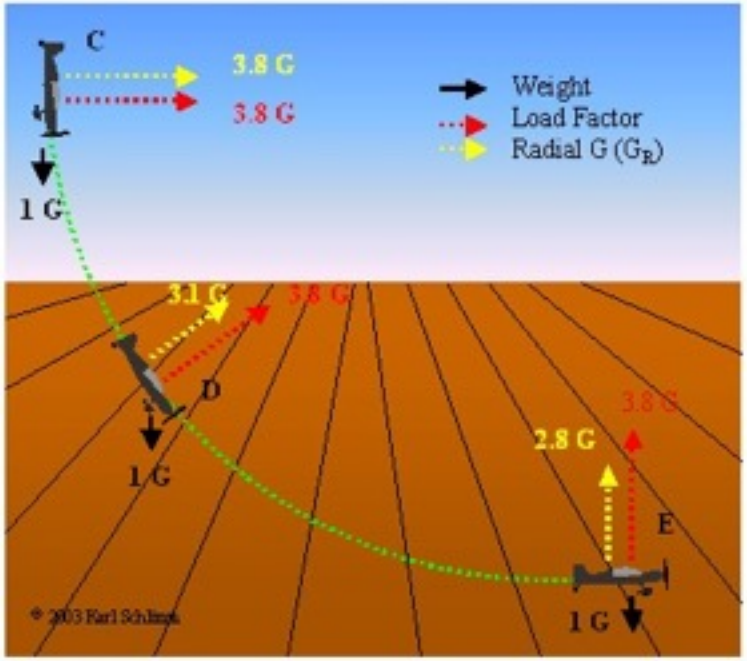
Stretch: **600 days**



Restore Speed:

Restore Speed:
Type: Software Quality Requirement.

Ambition: Should an error occur (or the user otherwise desire to do so), Horizon shall be abl to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.



Scale: Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.

Initiation: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps] 1 minute?

Fail [Initial and all subsequent released and Evo steps] 10 minutes. <- 6.1.2 HFA

Catastrophe: 100 minutes.

Testability:

Type: Software Quality Requirement.

Version: 20 Oct 2006-10-20

Status: Demo draft,

Stakeholder: {Operator, Tester}.

Ambition: Rapid-duration automatic testing of <critical complex tests>, with extreme operator setup and initiation.

Scale: the duration of a defined [Volume] of testing, or a defined [Type], by a defined [Skill Level] of system operator, under defined [Operating Conditions].

Goal [All Customer Use, Volume = 1,000,000 data items, Type = WireXXXX Vs DXX, Skill = First Time Novice, Operating Conditions = Field, {Sea Or Desert}. <10 mins.

Design Hypothesis: *Tool Simulators, Reverse Cracking Tool, Generation of simulated telemetry frames entirely in software, Application specific sophistication, for drilling – recorded mode simulation by playing back dump file, Application test harness console <-6.2.1 HFA*

Confirmit Case
Oslo Norway

Real Example of 1 of the 25 Quality Requirements

Usability.Productivity (taken from Confirmit 8.5, performed a set of predefined steps, to produce a standard MR Report.

development)

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.

Note: end result was actually 20 minutes

☺

Meter [Weekly Step]: Candidates with Reportal experience, and with knowledge of MR-specific reporting features

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		%		
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 Testing Estimates

Priority Next week Warning metrics based

Cumulative weekly progress metric

Constraint

Target

Snapshot End Week 9 of 12
for 1 of 4 4-developer teams

	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

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] 5 mins.,

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	10				
12					
13	80				
14	Usability.Productivity (minutes)				
15	65 35 25	20,00	50,00	38,00	95,00
20	Development resources				
21	0 110	4,00	3,64	4,00	3,64

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

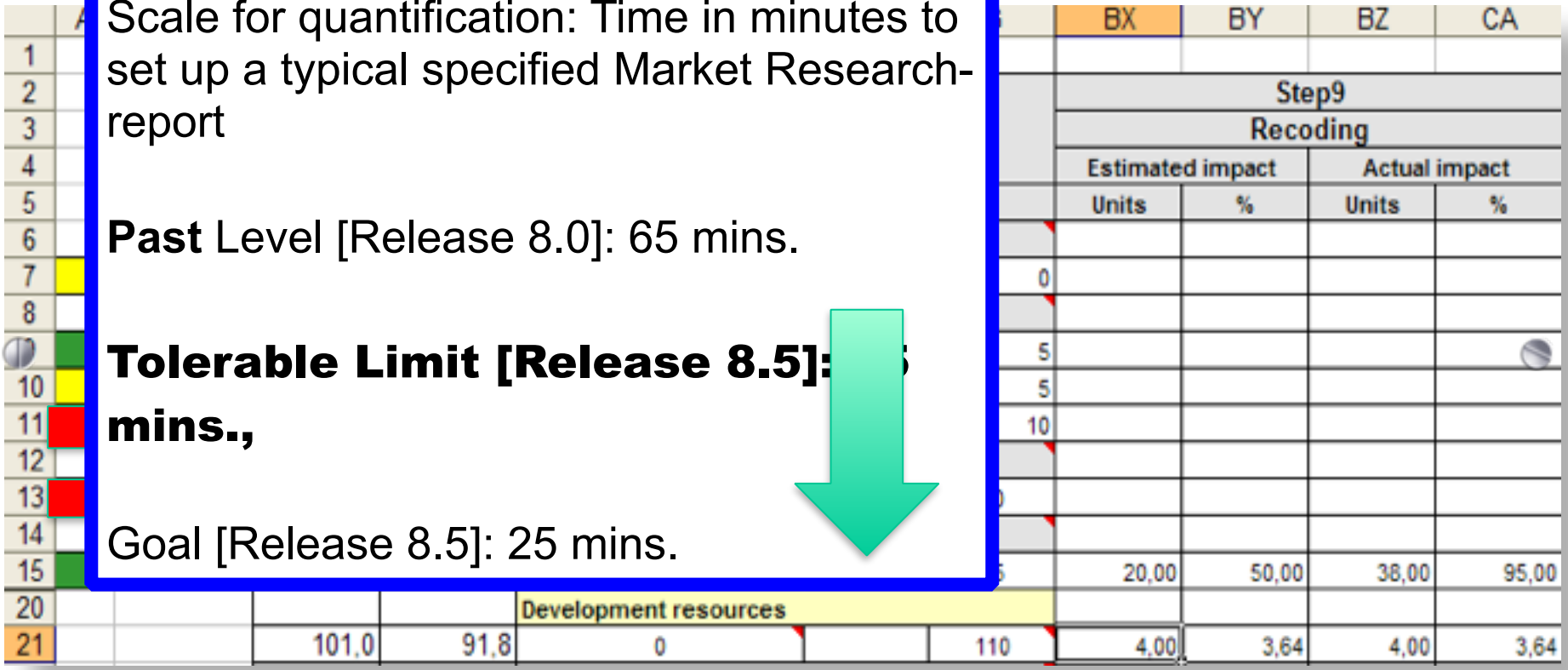
Usability.Productivity

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]: mins.,

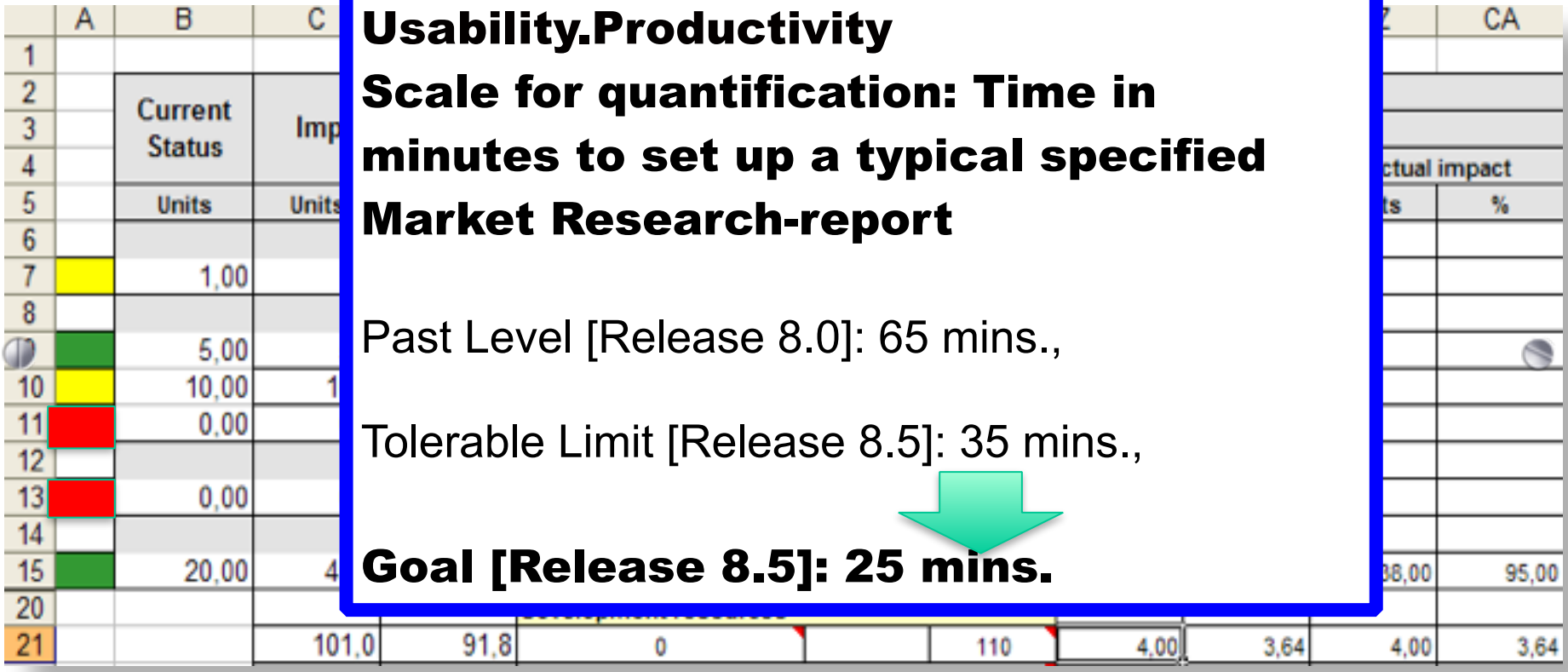
Goal [Release 8.5]: 25 mins.



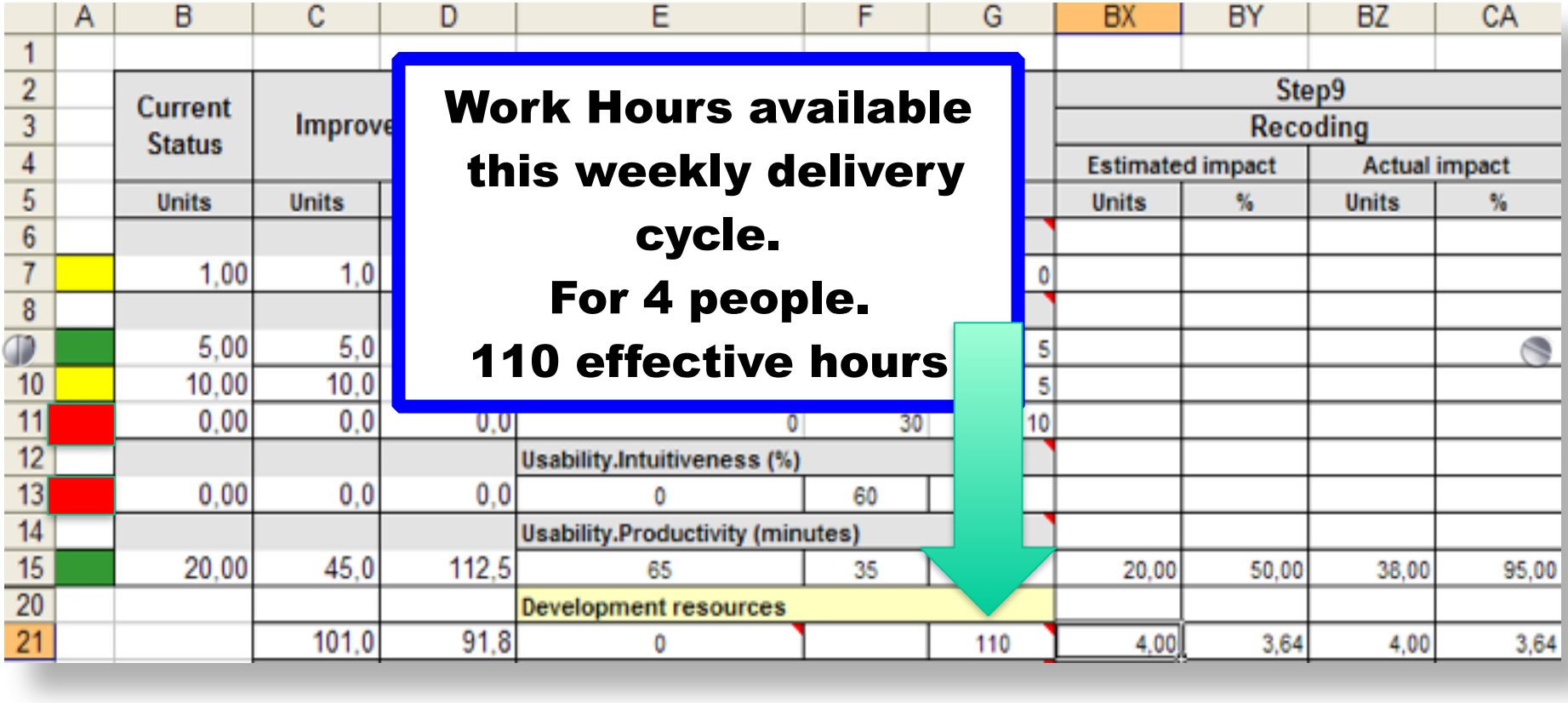
	BX	BY	BZ	CA
2	Step9			
3	Recoding			
4	Estimated impact		Actual impact	
5	Units	%	Units	%
6				
7				
8				
9				
10				
11				
12				
13				
14				
15	20,00	50,00	38,00	95,00
20				
21	4,00	3,64	4,00	3,64

			0		
			5		
			5		
			10		
			5		
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.



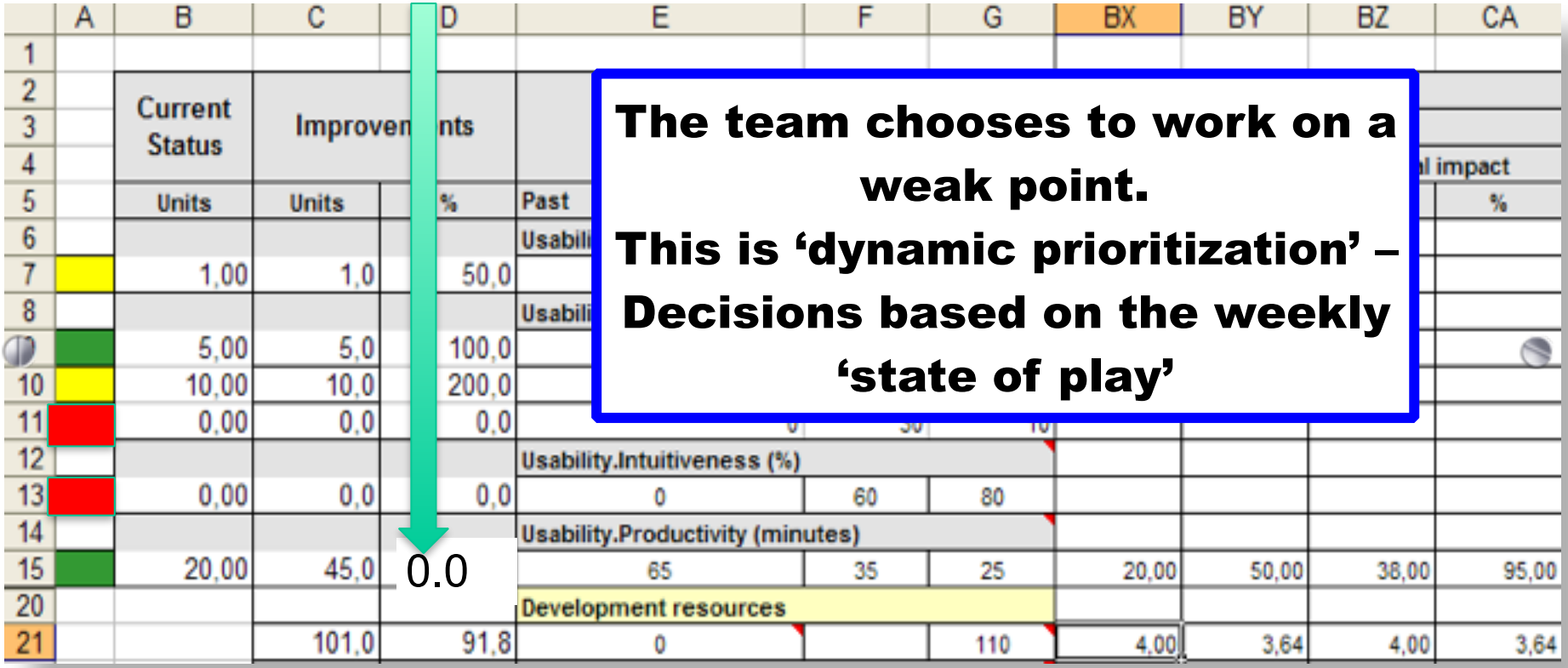
The weekly ‘value delivery cycle’ resource is 110 work-hours (4 days, effective time for the team of 3 to 4 people)



	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improve					Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units					Units	%	Units	%
6											
7		1,00	1,0				0				
8											
9		5,00	5,0				5				
10		10,00	10,0				5				
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							
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

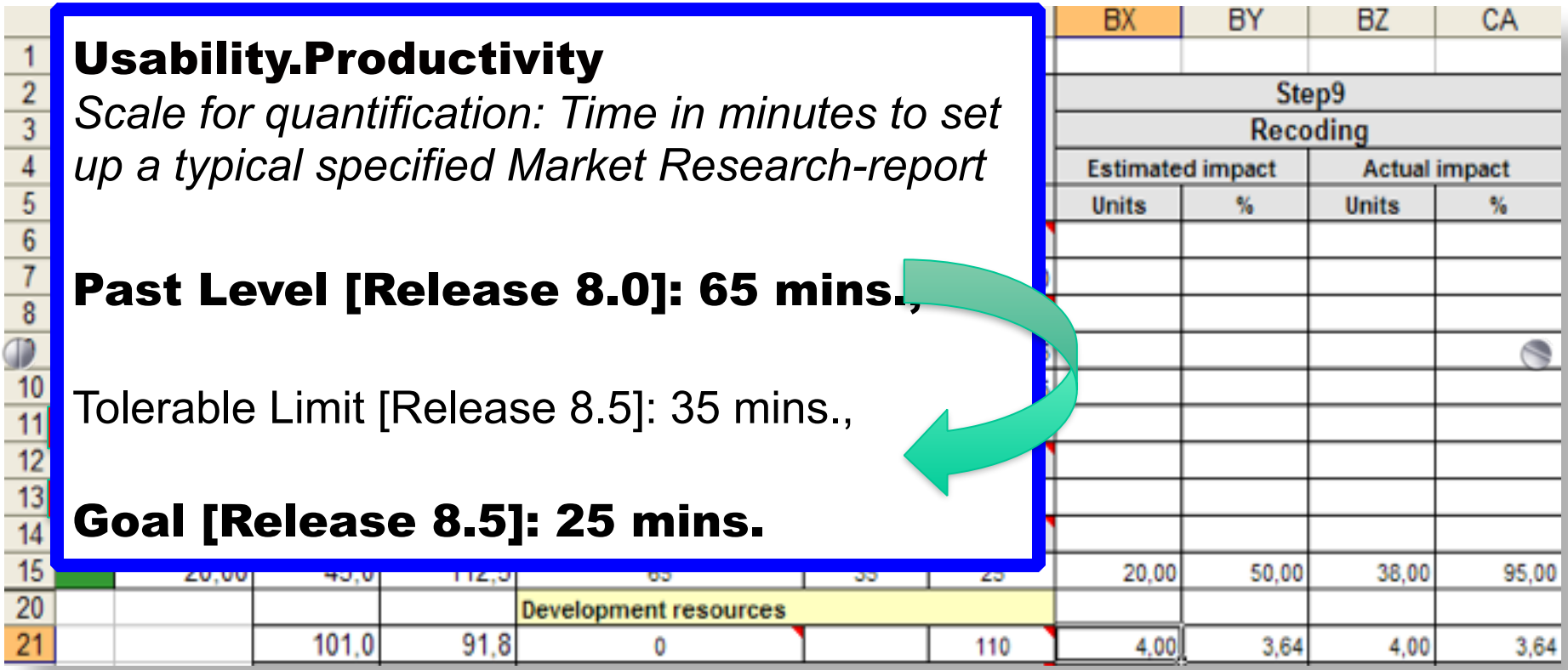
Usability.Productivity

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.,

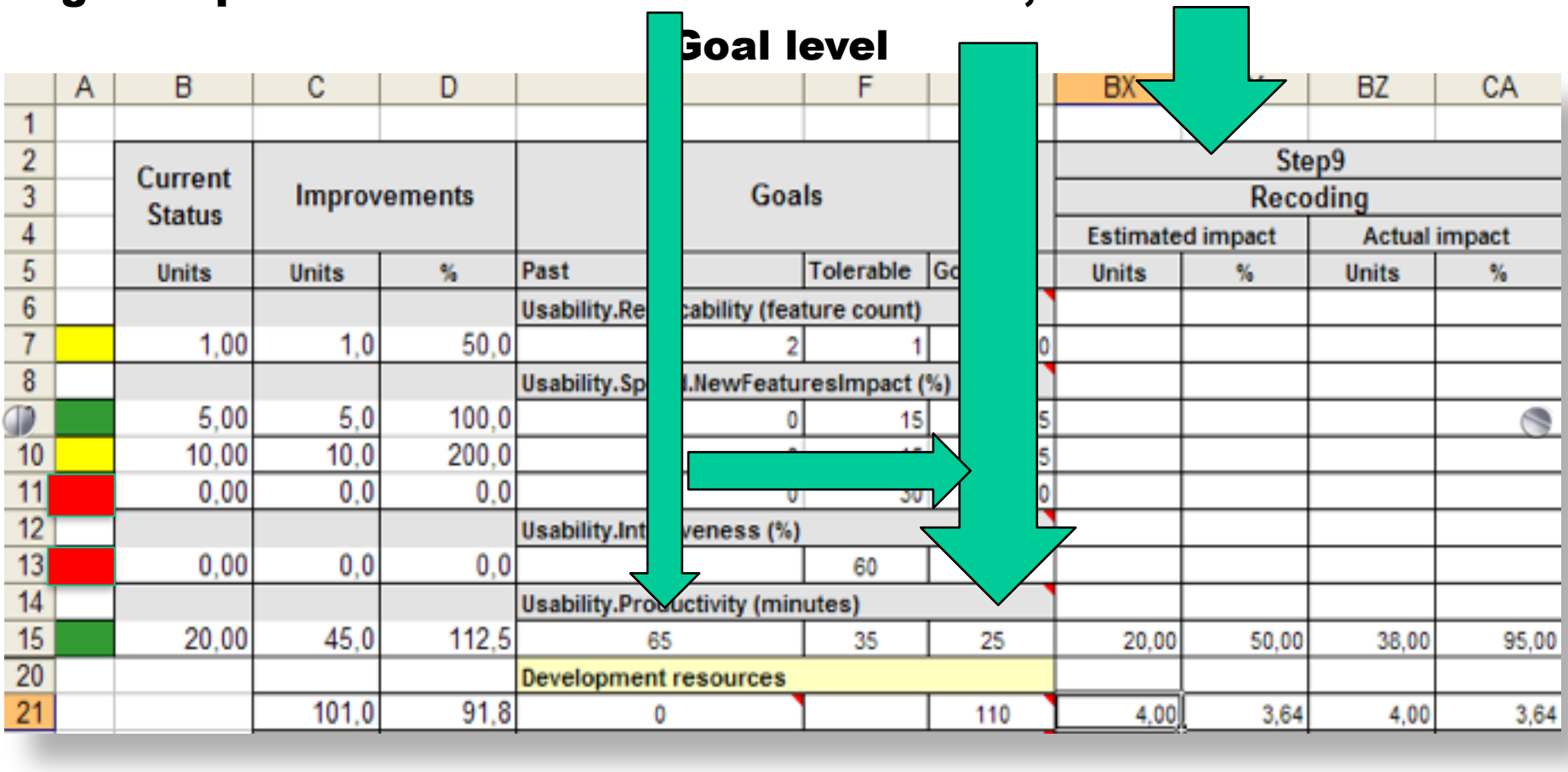
Goal [Release 8.5]: 25 mins.



	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				
15	20,00	50,00	38,00	95,00
20				
21	4,00	3,64	4,00	3,64

15							
20				Development resources			
21		101,0	91,8	0		110	

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

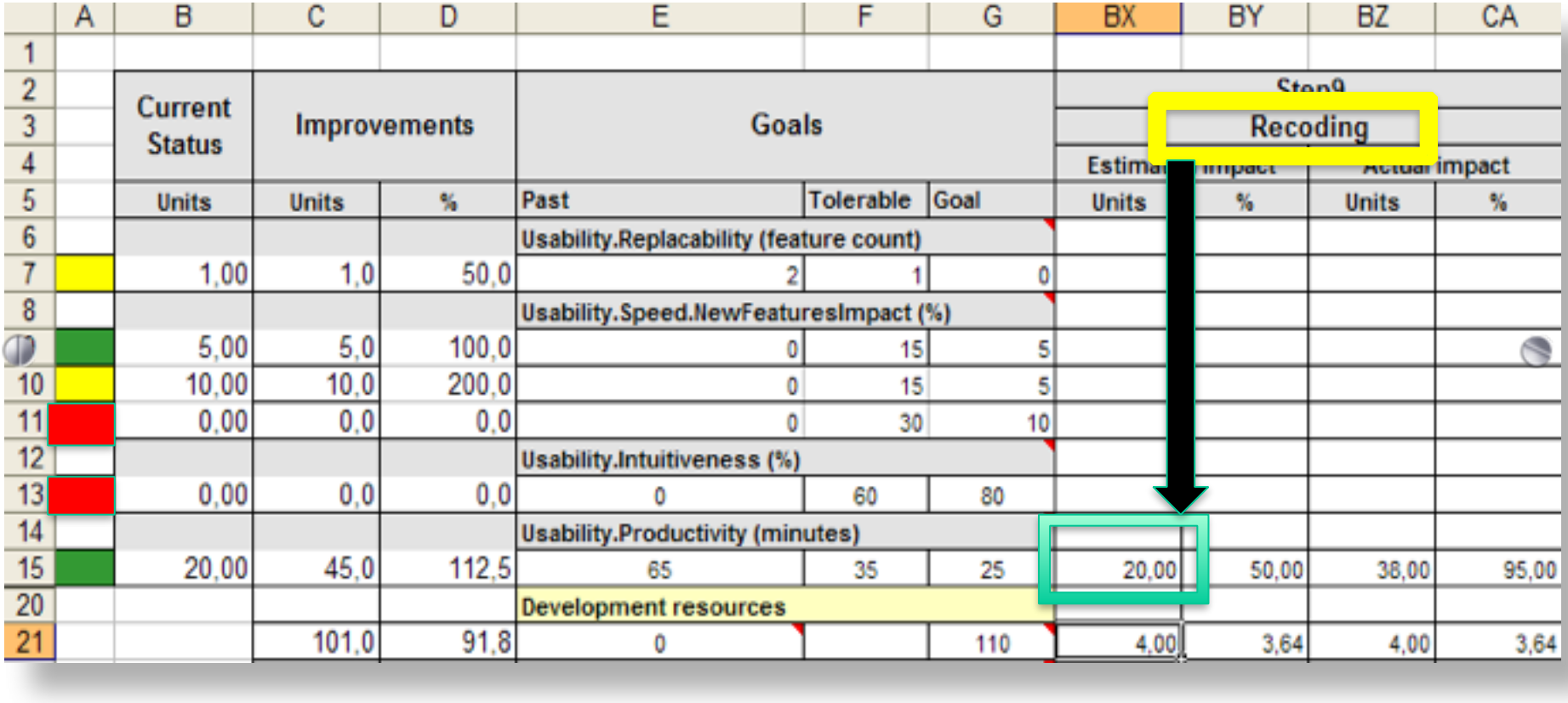


	A	B	C	D				BX	BZ	CA
1										
2		Current Status	Improvements		Goals			Step9		
3								Recoding		
4								Estimated impact	Actual impact	
5		Units	Units	%	Past	Tolerable	Go	Units	%	Units	%
6					Usability.Re cability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Sp l.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0			5				
11		0,00	0,0	0,0	0	30	0				
12					Usability.Int veness (%)						
13		0,00	0,0	0,0		60					
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

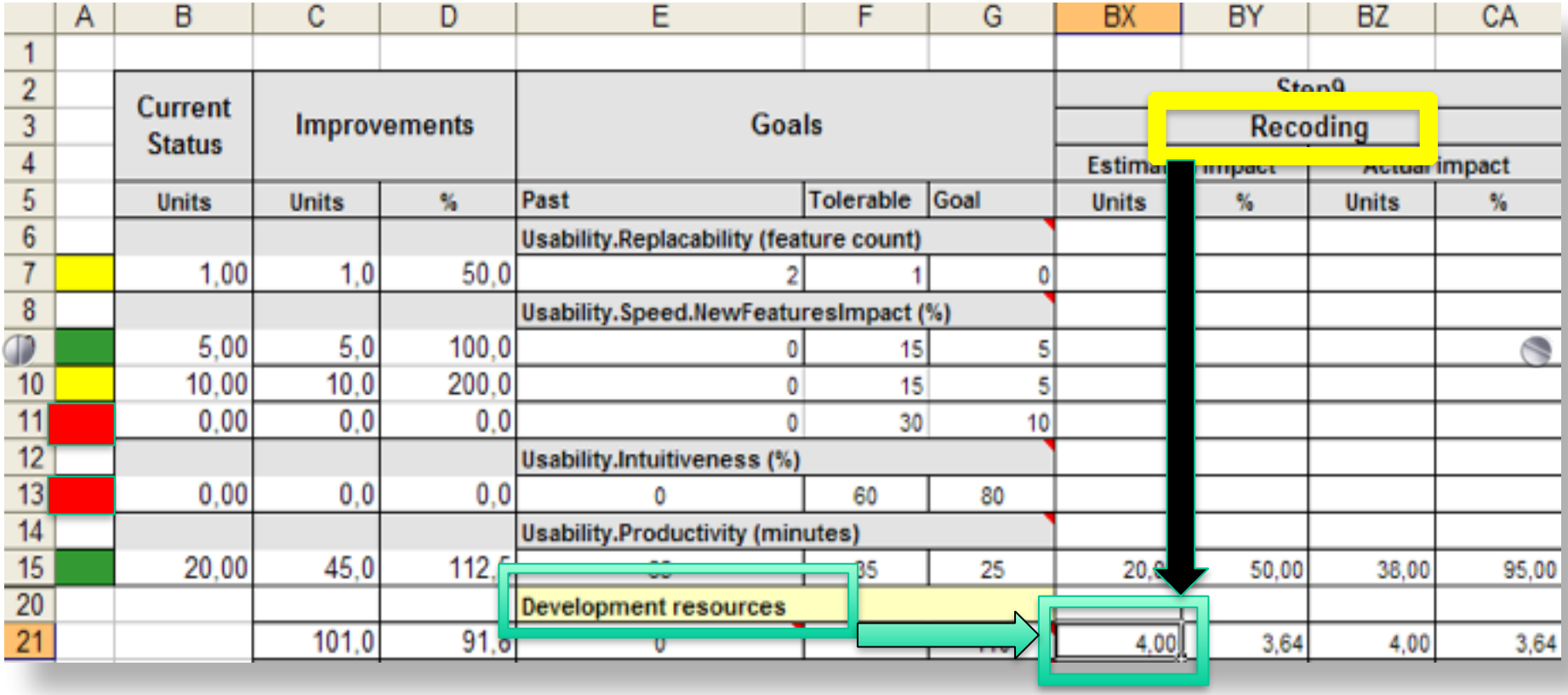
	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
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 ‘design 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

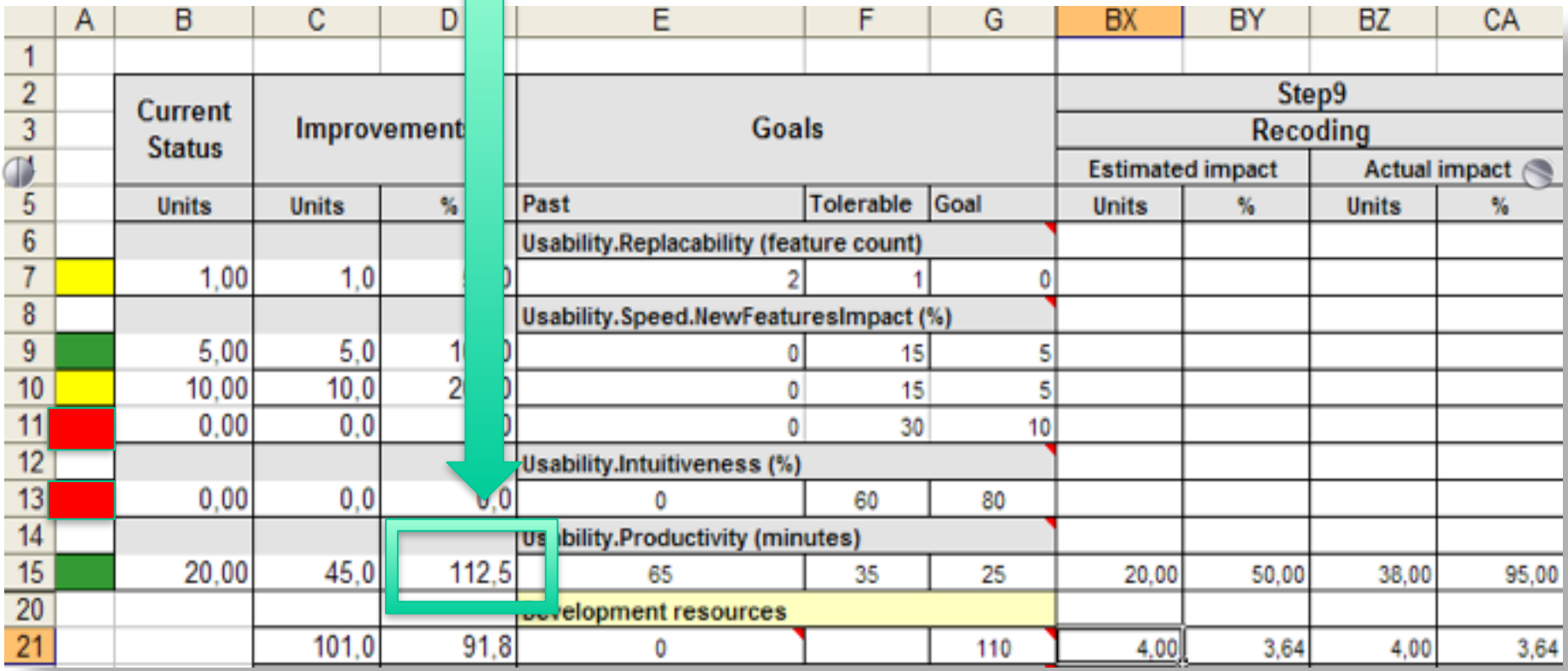
The team commits to the ‘Recoding’ solution. They code, test and handover to Microsoft usability Labs in Washington State, who volunteered to independently measure all the Usability designs.

	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

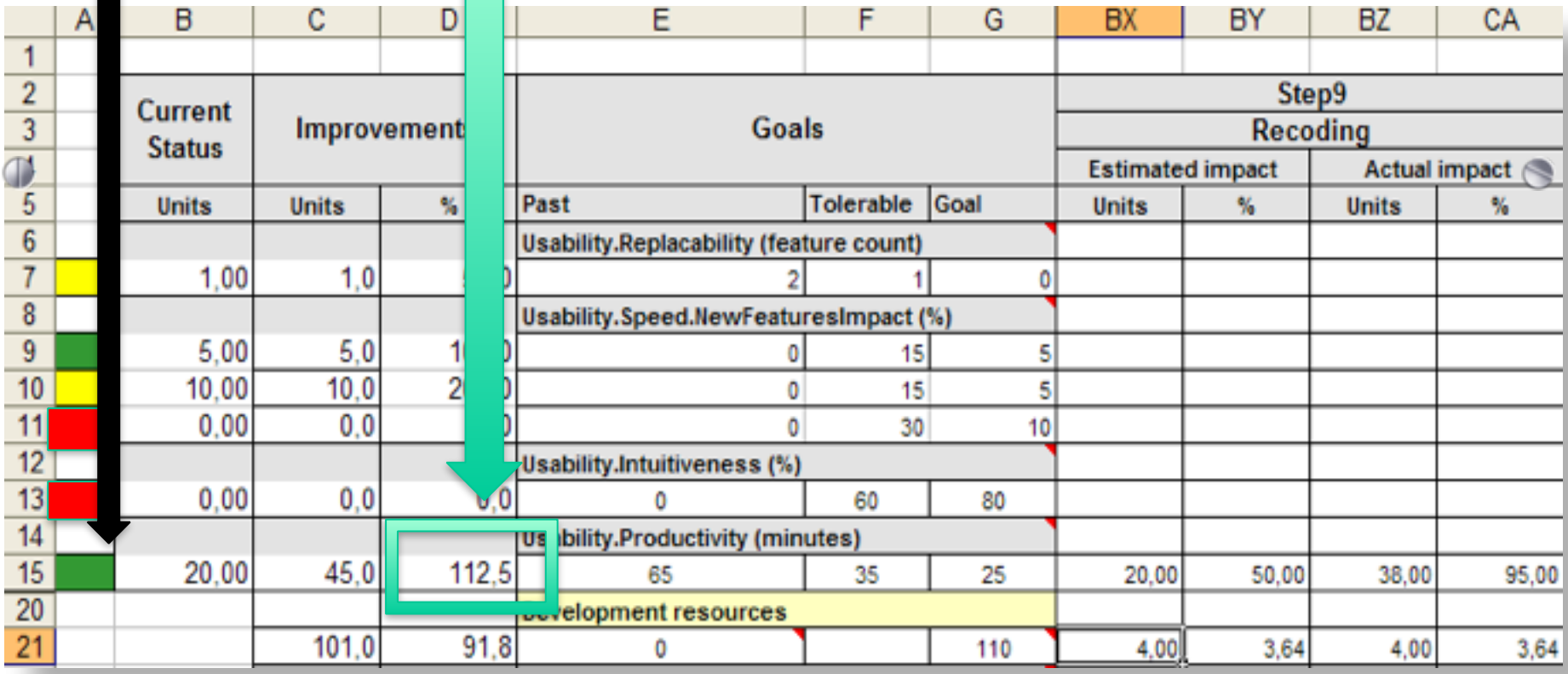
	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
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					Us bility.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					velopment 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					Us bility.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					velopment resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Requirements

Past	Tolerable	Goal
Usability.Replacability (feature count)		
2	1	0
Usability.Speed.NewFeaturesImpact (%)		
0	15	5
0	15	5
0	30	10
Usability.Intuitiveness (%)		
0	60	80
Usability.Productivity (minutes)		
65	35	25
Development resources		
		110

Benchmark

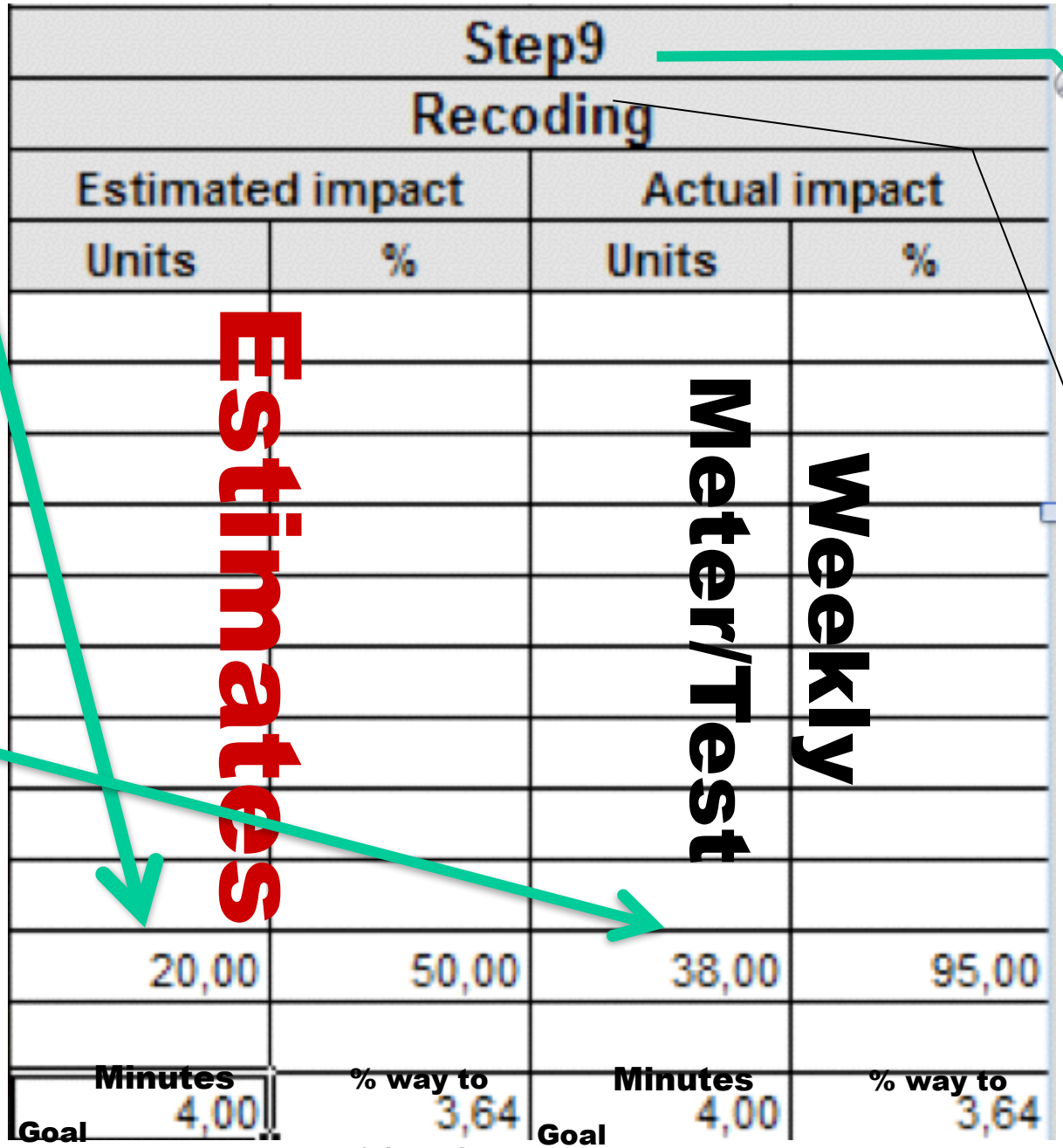
Constraint

Target

Cycle Resource

We estimate the 'design effect' at beginning of week

And measure the actual effect, at the end of the week



Week 9 of 12 Before Release

Tag of a 'design idea'

Work days % of Time to Release

Tracking Progress: after each Evo value delivery cycle

	Current Status	Improvements	
	Units	Units	%
	1,00	1,0	50,0
	5,00	5,0	100,0
	10,00	10,0	200,0
	0,00	0,0	0,0
	0,00	0,0	0,0
	20,00	45,0	112,5
		101,0	91,8

<- 50% of way to Goal level

<- All the way to the goal

<- Twice the way to the Goal level

<- No progress from Past level

<- 12.5 % over the Goal level

Computing Current Priority for next resources.
'Dynamic Prioritization'

	Current Status	Improvements	
	Units	Units	%
	1,00	1,0	50,0
	5,00	5,0	100,0
	10,00	10,0	200,0
	0,00	0,0	0,0
	0,00	0,0	0,0
	20,00	45,0	112,5
		101,0	91,8

Tolerable but not at Goal level

Not even Tolerable level
Give this highest priority next cycle

No priority. You reached or exceeded Goal

Snapshot End Week 9 of 12
for 1 of 4 4-developer teams

	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

Snapshot End Week 9 of 12
for 1 of 4 4-developer teams

	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

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

	Current Status	Improvements		Goals			Step9 Recoding			
							Estimated impact		Actual impact	
	Units	Units	%	Past	Tolerable	Goal	Units	%	Units	
				Usability.Replacability (feature count)						
	1,00	1,0	50,0	2	1	0				
				Usability.Speed.NewFeaturesImpact (%)						
	5,00	5,0	100,0	0	15	5				
	10,00	10,0	200,0	0	15	5				
	0,00	0,0	0,0	0	30	10				
				Usability.Intuitiveness (%)						
	0,00	0,0	0,0	0	60	80				
				Usability.Productivity (minutes)						
	20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
				Development resources						
		101,0	91,8	0		110	4,00	3,64	4,00	3,64

Weekly Testing Estimates

Priority Next week Warning metrics based

Cumulative weekly progress metric

Constraint

Target

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

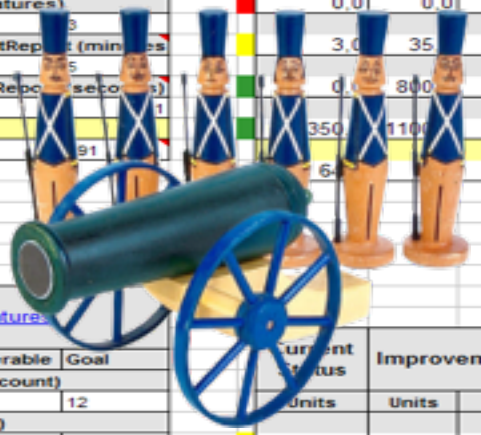
Impact Estimation Table: Reportal codename "Hyggen"

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	3
			Usability.ResponseTime.ExportRep (min		es
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	1100	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



The GREEN WEEK:

Agile *Technical Debt* **Engineering** beats 'Refactoring'

Tom Gilb

Tom @ Gilb . Com

www.Gilb.com

10 Minute Lightning Talk, 5 Nov 2013

Smidig 2013

Technical debt

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

Causes of technical debt include

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**

Conventional Refactoring

	Technique	Description
1	Code Refactoring (clean-up)	It is intended to remove the unused code, methods, variables etc. which are misleading.
2	Code Standard Refactoring	It is done to achieve quality code.
3	Database Refactoring (clean-up)	Just like code refactoring, it is intended to clean or remove the unnecessary and redundant data without changing the architecture.
4	Database schema and design Refactoring	This includes enhancing the database schema by leaving the actual fields required by the application.
5	User-Interface Refactoring	It is intended to change the UI without affecting the underlying functionality.
6	Architecture Refactoring	It is done to achieve modularization at the application level.

Refactoring – to Sustain Application Development Success in Agile Environm
by Narayana Maruvada
In agilerecord.com Nov 1 2013

Impact Software Qualities

- "Importantly, the underlying objective behind refactoring is to give thoughtful consideration and **improve** some of the **essential** ***<Quality> attributes*** of the software."

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In AGILERECORD.COM NOVEMBER 1 2013

Impact Software Qualities

"Key Benefits of Refactoring

From a system/application standpoint, listed below are summaries of the key benefits that can be achieved seamlessly when implementing the refactoring process in a disciplined fashion:

① Firstly, it improves the overall software **extendability**.

② Reduces and optimizes the code **maintenance cost**.

③ Facilitates highly standardized and organized code.

④ Ensures that the system architecture is improved by retaining the behavior.

⑤ Guarantees three essential attributes: **readability**, **understandability**, and **modularity** of the code.

⑥ Ensures constant improvement in the **overall quality** of the system. "

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In agilerecord.com Nov 1 2013

Impact Software Qualities

"Key Benefits of Refactoring

From a syst
summaries
when imple
fashion:

① Firstl

② Redu

③ Facili

④ Ensu
retaini

⑤ Guarantees three essential attributes: **readability**, **understandability**, and **modularity** of the code.

⑥ Ensures constant improvement in the **overall quality** of the system. "

sly

.

No numbers given to support this

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In agilerecord.com Nov 1 2013

There is a smarter way

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

- Not just- - - *softcrafters**

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

A bright idea: based on experience

- So, Confirmit was getting amazing results for the user, customer, and system level attributes they targeted
- And someone on the team realized…
  - **What about us devs and testers**
  - **We are stakeholders too!**
  - Refactoring (1 day a week) was NOT working well.
- Let us try **to engineer the qualities** that we need into the system
- The same way we engineer the **user** qualities into the system

Code quality – ”green” week, 2005
“Refactoring by Proactive Design Engineering!”

- In these ”green” weeks, some of the deliverables will be less visible for the end users, but more visible for our QA department.
- We manage code quality through an Impact Estimation table. T.I.

Current Status		Improvement	Goals			Step 6 (week 14)		Step 7 (wee	
	Units		Past	Tolerable	Goal	Estimated Impact	Actual Impact	Estimated Impact	A
	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						

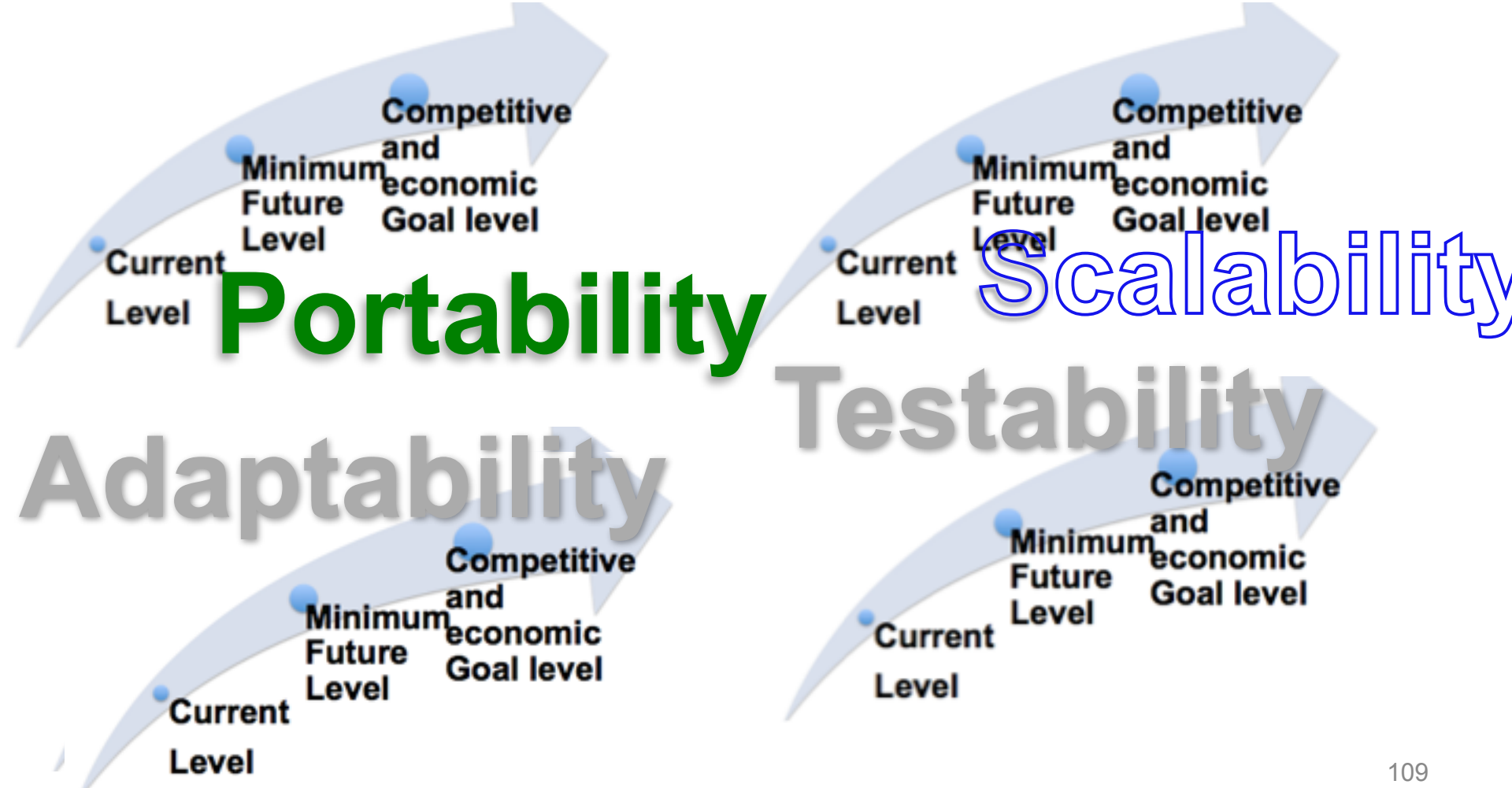
- Speed
- Maintainability
- Nunit Tests
- PeerTests
- TestDirectorTests
- Robustness.Correctness
- Robustness.Boundary Conditions
- ResourceUsage.CPU
- Maintainability.DocCode
- SynchronizationStatus

The Monthly 'Green Week'

User Week 1	User Week 2	User Week 3	**Developer Week 4**
• Select a Goal	• Select a Goal	• Select a Goal	• **Select a Goal**
• Brainstorm Designs	• Brainstorm Designs	• Brainstorm Designs	• **Brainstorm Designs**
• Estimate Design Impact/ Cost	• Estimate Design Impact/ Cost	• Estimate Design Impact/ Cost	• **Estimate Design Impact/**

Raising the Levels of Maintainability like 'Mean Time To Fix a Bug'

Raising the Levels of Maintainability
Multiple Attributes of Technical Debt



Broader 'Maintainability' Concepts
ALL *quantified*, with a defined Scale of measure in CE-5

Performance
- Quality
  - Availability
    - Reliability
    - Maintainability
    - Integrity
      - Threat
      - Security
  - Adaptability
    - Flexibility
      - Connectability
      - Tailorability
        - Extendibility
        - Interchangeability
    - Upgradeability
      - Installability
      - Portability
      - Improveability

Chapter 5: Scales of Measure:
http://www.gilb.com/tiki-download_file.php?fileId=26

1. **The Conscious Design Principle**:

- "Maintainability must be *consciously* designed into a system:
  - failure to **design** to a set of levels of maintainability
  - means the **resulting maintainability** is both *bad* and *random*."
  - © Tom Gilb (2008, INCOSE Paper)
    - http://www.gilb.com/tiki-download_file.php?fileId=138

The 'Maintainability' Generic Breakdown into Sub-problems

1. Problem Recognition Time.

   How can we reduce the time from bug actually occurs until it is recognized and reported?

2. Administrative Delay Time:

   How can we reduce the time from bug reported, until someone begins action on it?

3. Tool Collection Time.

How can we reduce the time delay to collect correct, complete and updated information to analyze the bug: source code, changes, database access, reports, similar reports, test cases, test outputs.

4. Problem Analysis Time.

   Etc. for all the following phases defined, and implied, in the Scale scope above.

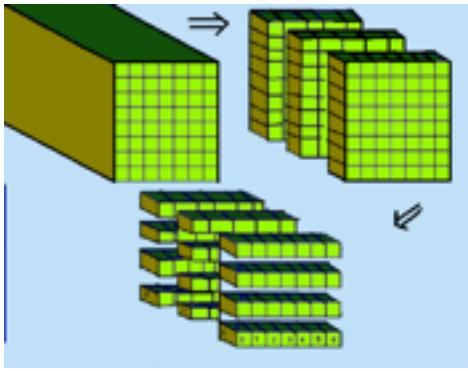
5. Correction Hypothesis Time

6. Quality Control Time

7. Change Time

8. Local Test Time

9. Field Pilot Test Time



10. Change Distribution Time

11. Customer Installation Time

12. Customer Damage Analysis Time

13. Customer Level Recovery Time

14. Customer QC of Recovery Time

Source: Competitive Engineering Ch 5
Chapter 5: Scales of Measure:
http://www.gilb.com/tiki-download_file.php?fileId=26
& Ireson (ed.) Reliability Handbook, 1966

An Example of Specifying 1 Attribute in 'Planguage'

***Restore Speed*:**

Type: *Software Quality Requirement.* ***Version***: *25 October 2007.*

Part of: *Rock Solid Robustness*

Ambition: *Should an error occur (or the user otherwise desire to do so), the system shall be able to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.*

Scale: Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.

Initiation: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps] 1 minute?

Fail [Initial and all subsequent released and Evo steps] 10 minutes. <- 6.1.2 HFA

Catastrophe: 100 minutes.

Let's Vote

1. How many of you would **prefer** to keep doing conventional 'softcrafter' refactoring; even if the results were not measurable

2. How many of you think you **ought** to try to engineer measurable software maintainability results into your systems
   - Even if your boss is not smart enough to ask you, or support you doing it?

Further Reading: AgileRecord.com
Collection is in tiny.cc/GilbMyths
many views on Agile and Quality metrics

Gilb's Mythodology Column

The Green Week: Reducing Technical Debt by Engineering

by Tom & Kai Gilb

Our client Confirmit.com has used our Evo Agile Method [2] successfully since 2003 [3]. They have adapted it, from the beginning, to their environment, and continued to innovate and learn. Their business success has been attributed to their remarkable product quality improvement, and that improvement specifically to the Evo Agile method, by them, on their website, and share offerings prospectus. Evo differs from other agile methods, in that it focuses on multiple, quantified, software and system qualities.

This column will focus on an innovation, the Green Week, that Confirmit, led by their method champion Trond Johansen, made and reported in 2005, two years after adopting Evo.

When we started in 2003, Confirmit had an 8 year old web-based system; a 'legacy' product that had grown, as most do, to meet rapidly emerging market demands. By 2005 there were the usual difficulties in enhancing the product, a web-based opinion survey tool, serving markets worldwide, to meet new opportunities, quickly and safely.

We recommended in 2003 that they spend 4 days a week on value delivery cycles to their customer base, and one day a week 'refactoring'. Their development team at the time was 13 plus 3 testers.

The 4 day value delivery cycle aimed at something like 25 distinct quality improvements (for example Usability: intuitiveness) or performance capacity improvements. The stakeholders aimed at were users and Confirmit's future market. The refactoring was aimed at their development team, as stakeholders. The team that did the development initially, also did the maintenance of the system for years, until today.

Let me be explicit, the people who had to 'suffer' bug fixing and long term enhancement were actually in full control of the architecture and design of the entire system. Maintenance was not farmed out to people who just had to suffer it. Most of the staff were not merely programmers, they had formal education in real engineering.

Well, the one day of refactoring was not a great success, while the 4 days of value delivery cycles, to quantified quality and performance requirements was a big success. To my knowledge there is nothing even near as good of quantified results, reported for any other Agile Effort! If you know of one, AgileRecord.com would like to hear from you! One possible reason for lack of success was that the refactoring was one day a week, and I suspect it was a Friday, where Norwegians want to sneak off early for a Cabin Weekend ("working off site"). But I really don't know.

They asked themselves, 'why should our customers get all the quality improvements?' What not, us hard working developers, get some systematic quality improvements too!'

So they decided to spend one week a month, using Evo [2] 'engineering' 'ease of maintenance' and 'testability' into their organization and their product. In other words: 3 weeks being customer oriented, and 1 week a month being internally oriented. Of course, improvements in maintenance capability also improve their ability to respond to customers!

User Week 1	User Week 2	User Week 3	Developer Week 4
Select a Goal	Select a Goal	Select a Goal	Select a Goal
Brainstorm Designs	Brainstorm Designs	Brainstorm Designs	Brainstorm Designs
Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost
Pick best design	Pick best design	Pick best design	Pick best design
Implement design	Implement design	Implement design	Implement design
Test design	Test design	Test design	Test design
Update Progress to Goal	Update Progress to Goal	Update Progress to Goal	Update Progress to Goal

Figure 1: The weekly development cycle, with the Green Week.

The key idea here is that we start by quantifying as requirements, all Confirmit system (the software product, the service product, the technical organization) attributes, related to ease of maintaining the system, in the widest sense of 'maintaining' [3].

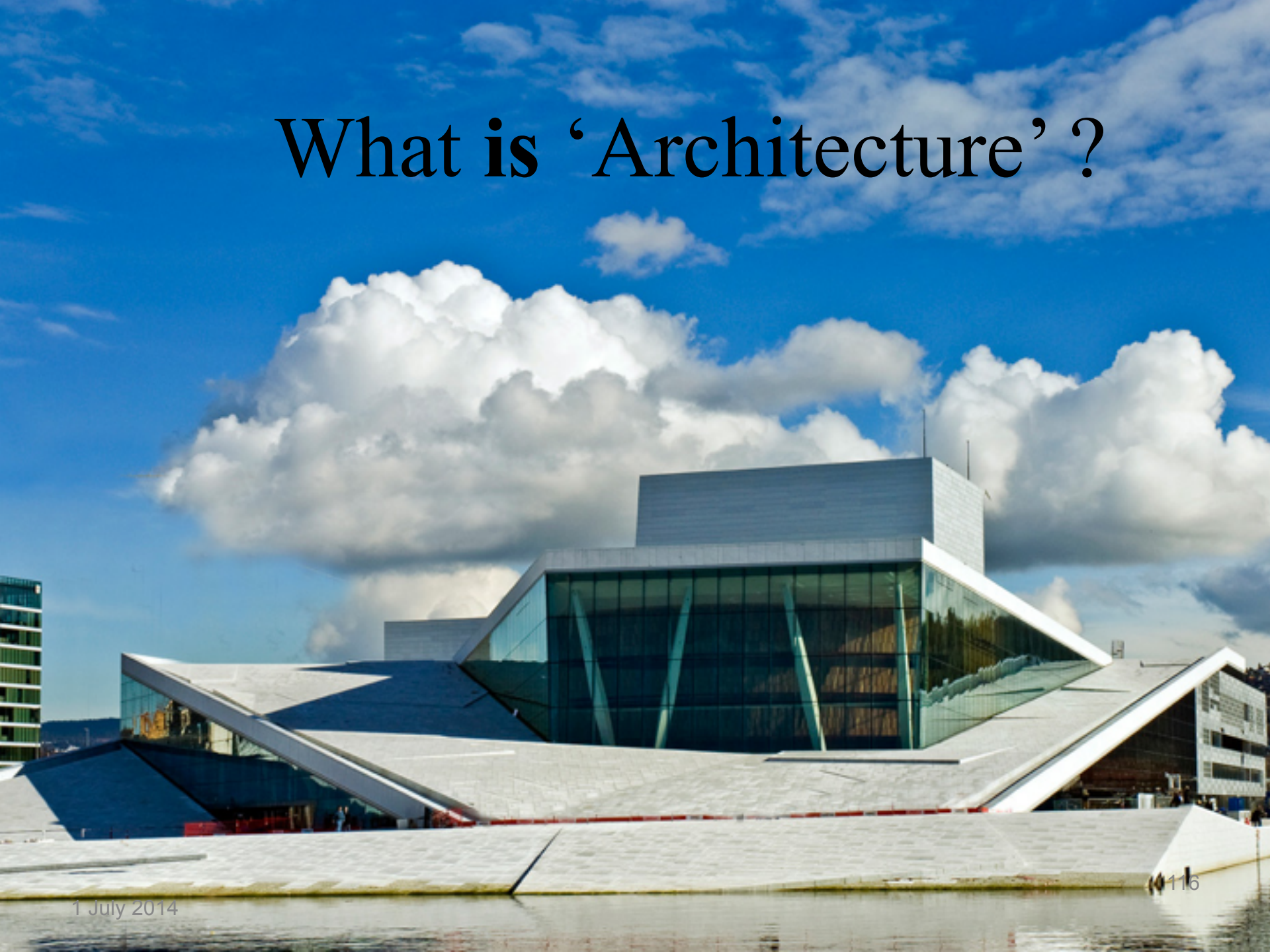
Here are the requirements they quantified as requirements initially:

Speed, Maintainability, Ayrit Tests, Peer Tests, Test Director Tests, Robustness.Correctness, Robustness.Boundary Conditions, Resource Usage CPU, Maintainability.OutCode, Synchronization Status.

Page 20 Agile Record - www.agilerecord.com

http://www.gilb.com/dl575
May 2013, In Agilerecord.com

What **is** ‘Architecture’ ?



Architect = Master Builder

Architect is from ‘Arc
Tecton,’

which means
‘Master Builder’.

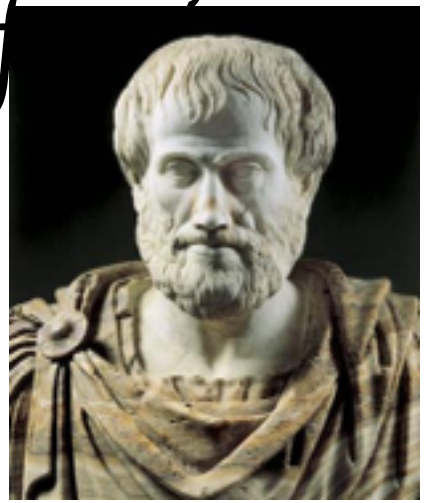
‘Archi’ is not from ‘Ar

but from ‘Arche’:
primitive, original,
primary.

The architecture is there to satisfy requirements

The closer an object is to fulfilling its purpose, the closer it is to perf

Aristotle's Belief



Oslo Opera house

- Qualities
- Costs
- Constraints

Oslo Opera house requirements (guess)

- Qualities
  - Impressive
  - Acoustics
  - Flexibility
  - Extendibility
  - Integratedness
  - Performance Visibility
  - National Symbol
  - Access to Fjord View
  - Comfort
- Costs
  - Building
  - Maintenance
  - Operational manpower
- Constraints
  - Legal Building
  - National Architecture
  - Archeological Site
  - Local Materials
  - Local Labour

The architecture is there to satisfy requirements

Architecture
that never refers to
necessary qualities,
performance characteristics,
costs,
and constraints

Is not really architecture

Of any kind

The architecture is there to satisfy requirements

The Architecture *process* is **driven** by requirements

Real (IT/Sw) Architecture

Real Architecture

- Has multidimensional *clear* design performance objectives
- Has *clear* multiple constraints
- Produces architecture ideas which enable and permit objectives to be met reasonably within constraints
- Estimates expected effects

***Pseudo* Architecture**

- Lacks dedication to clear **objectives** and **constraints**
- Does not **estimate** or articulate the expected **effects, on** objectives & constraints, of suggestions

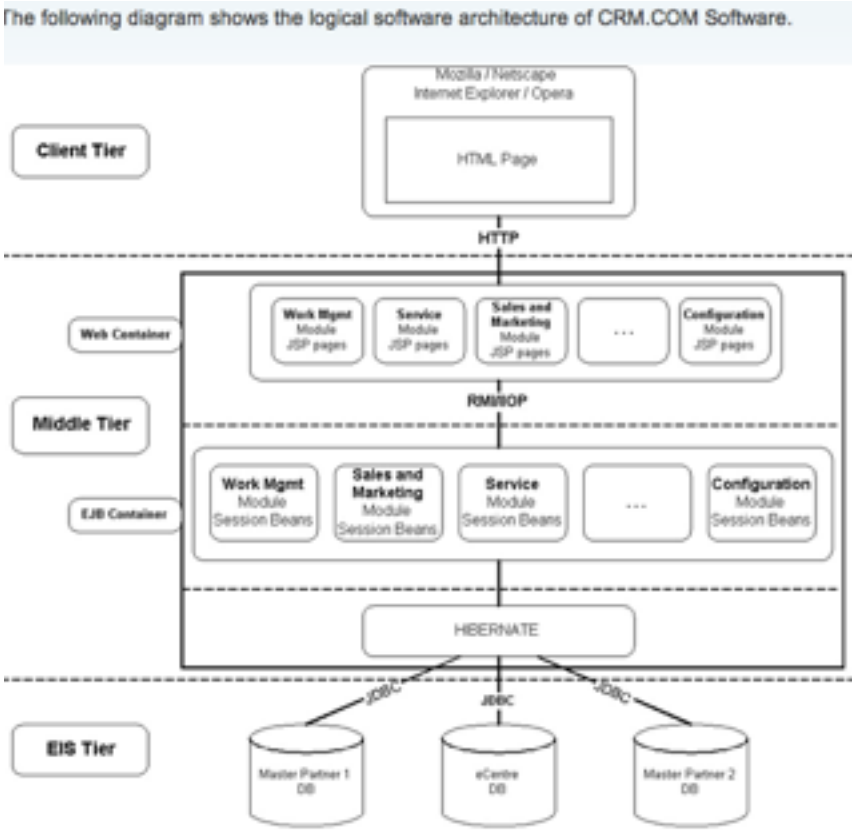
Pseudo Architecture
Does not mention goals and constraints

‘Bad’ ‘Arch.’ definitions

- Software architecture is a collection of software **components** unified via interfaces into decomposable system based on one or more technology platforms.
- Software Architecture shows the **structural** and **behaviour** of a system which is comprised of software **elements** and *exposing the properties* of those elements and relationships among them.

http://www.sei.cmu.edu/architecture/start/community.cfm

Uninformative diagrams

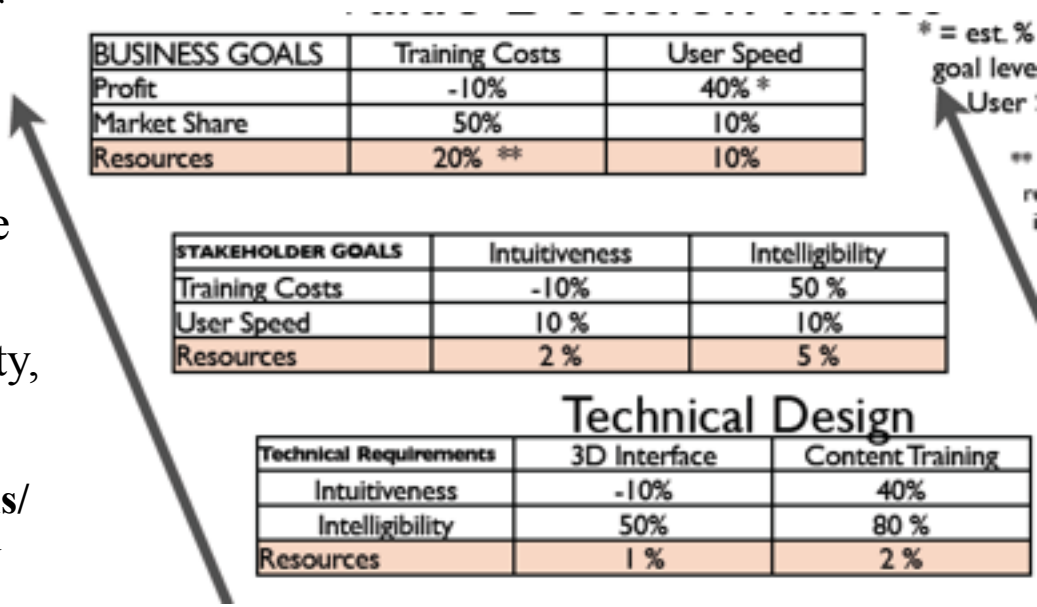


Better Architecture

Better definitions

- Software …needs to address the needs of business **stakeholders** within the organizational, technical and any other **constraints** to achieve the business, technical or any other **goals**.
  - It also needs to address software trustworthy characteristics like reliability, availability, maintainability, robustness, safety, security and survivability.
- System Architecture should contain **goals/ requirements** artifacts, and structure and behavior artifacts **based on** those goals.

Real Architecture diagrams



http://www.sei.cmu.edu/architecture/start/community.cfm

A Distinction

Architecture *Process*

- A continuous, and lifecycle long, **activity of finding means for ends**

Architecture *Specification*

- A specification of
  - a set of means
  - for a set of ends

We argue that the following are **absolute essentials** for 'real' architecture

Architecture *Process* has

- Clear multiple objectives
- Clear constraints
- A process of identifying and analyzing (estimating effects of) potential means
  - For reaching objectives, within constraints

Architecture *Specification* has

- Well defined components
  - Able to deliver predictable attributes
- Credible estimates of the multiple effects of each component, and the whole

Why are these Architecture essentials, essential?

Why?

- Failure to reach even one ‘critical’ objective can mean total system failure
  - Example: reliability
- Failure to respect even a single constraint can mean total system failure
  - Example: cost

And if they are missing…

- You cannot expect the specified architecture will reach objectives, within constraints
- You have lost architectural control

What a Difference

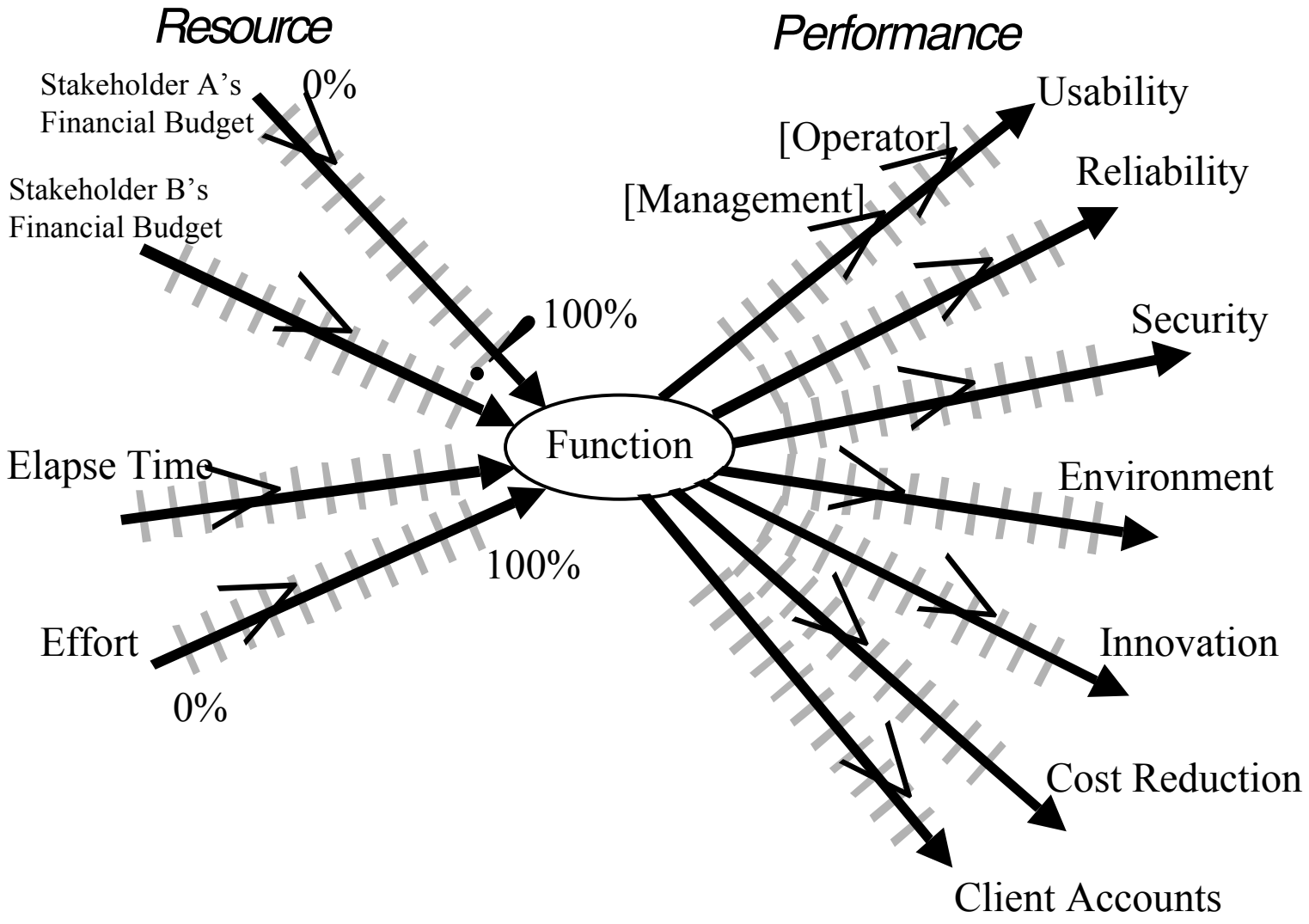
A Real Architect

- Can and does estimate resources needed for any suggested architecture
  - Capital Cost
  - Maintenance Cost
  - Skilled People hours to install and maintain
- Can and Does estimate the impact of each architecture component on the top level critical objectives
  - All '-ilities' (security etc)
  - All Performance (Capacity

A False Architect

- Does not even try to **estimate any costs**
- of any architectures
  - Does **not know how** to do so if asked
  - If they try to estimate they are at least 10x wrong
- Does not **even try to estimate the numeric impact** on even the most critical architectural objectives
- Does not even **realize** they need **quantified performance and quality objectives** to drive and justify architecture
- They have no specific verifiable idea of the impact their ideas have on numeric quality and performance levels.
- It is all 'smoke and mirrors'
- They take **no responsibilit**y for the performance and quality attributes or costs of their suggested architecture: no skin in the game.

Multiple Required Performance and Cost Attributes
are the basis for architecture selection and evaluation



Planguage Glossary

(full glossary 650+ concepts download at www.gilb.com)
http://www.gilb.com/tiki-download_file.php?fileId=387

- **Architecture (collective noun):**
  - Concept *192. May 9 2005

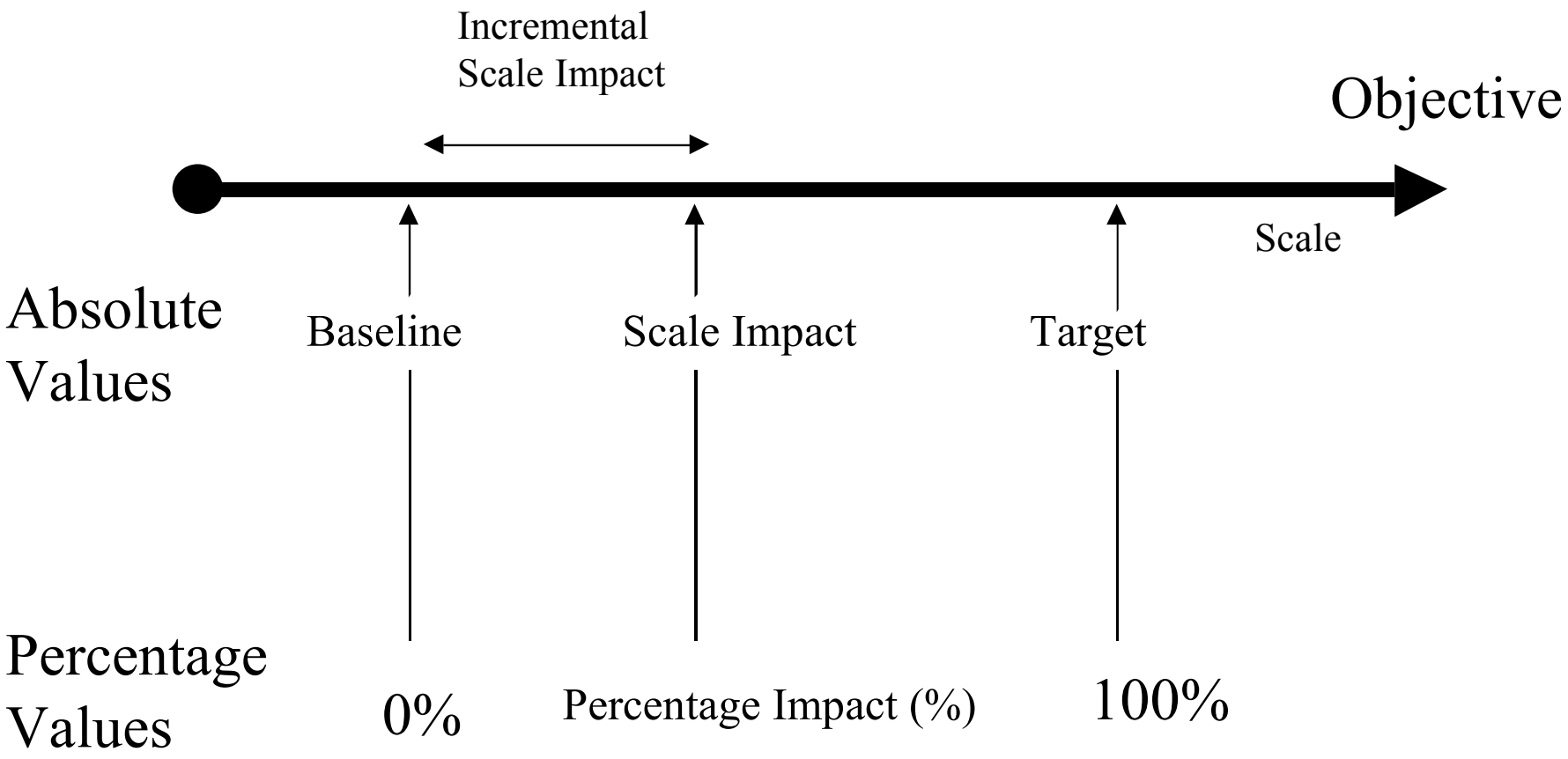
- The 'architecture' is
  - the set of entities that in fact exist
  - and impact a set of system attributes
  - directly, or indirectly, by
    - constraining,
    - or influencing,
      - related engineering decisions.

Requirement

- is a
- **stakeholder-valued system state,**
- **under stated conditions.**

- **Concept *026 (Planguage Glossary, 2012)**
- http://www.gilb.com/tiki-download_file.php?fileId=386

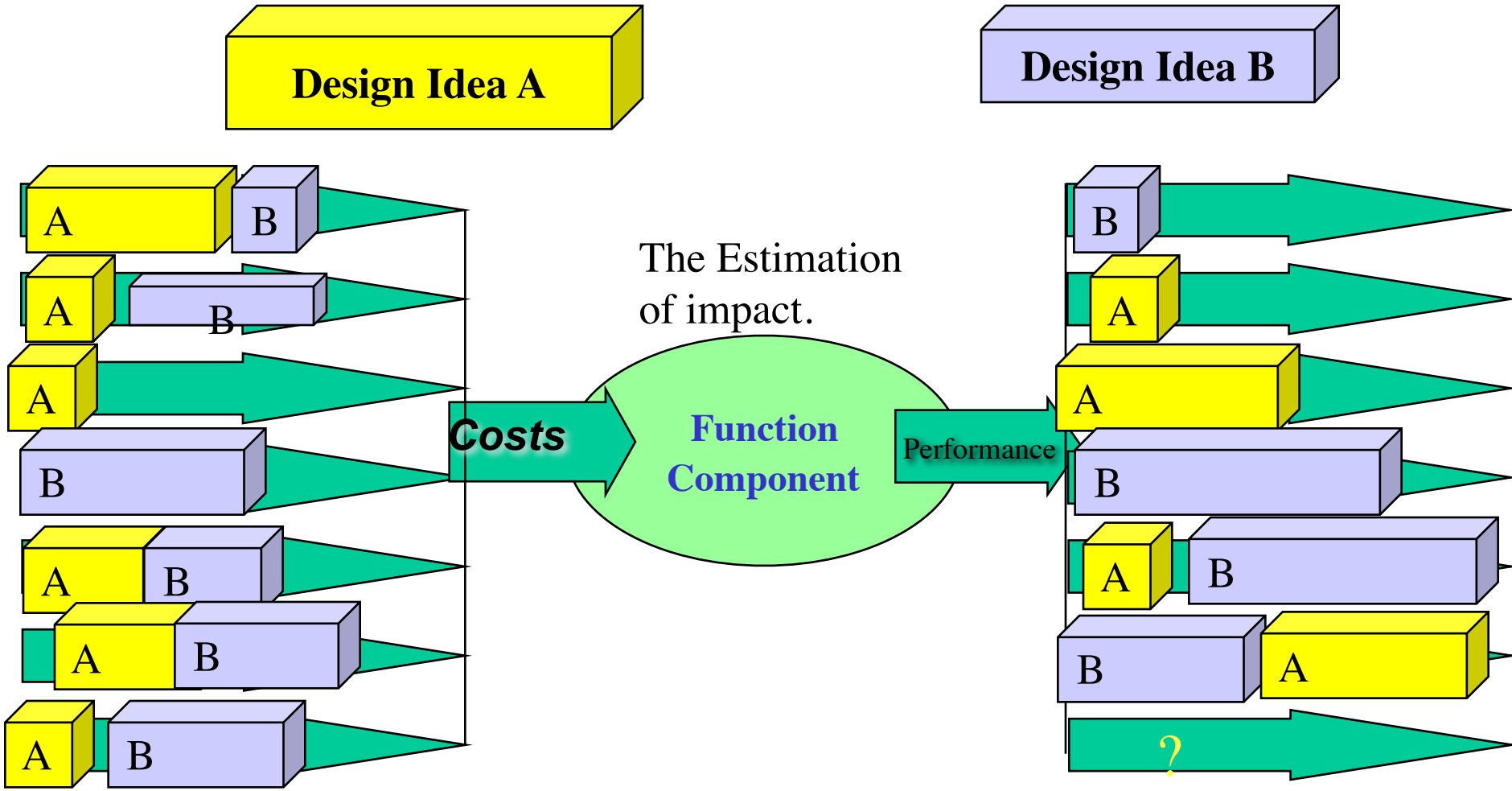
Impact Estimation Basic Concepts



Source: Lindsey Brodie, Editor of Competitive Engineering May 2000

Impact Estimation:

How much do designs impact all critical cost and quality attributes?

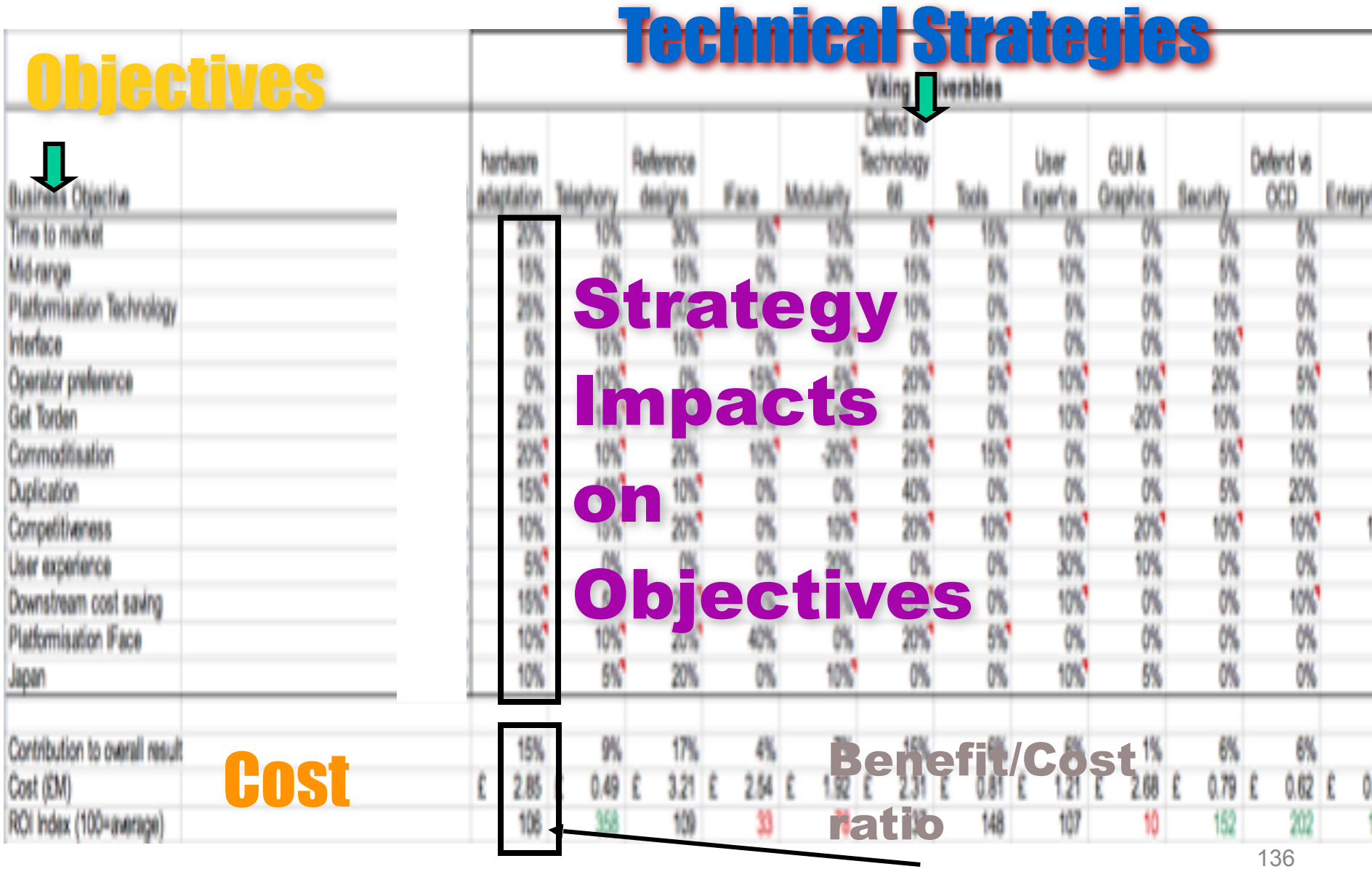


• Figure 1: Real (NON-CONFIDENTIAL version) example of an initial draft of setting the objectives that engineering processes must meet.

Business objective	Measure	Goal (200X)	Stretch goal ('0X)	Volume	Value	Profit	Cash
Time to market	Normal project time from GT to GT5	<9 mo.	<6 mo.	X		X	X
Mid-range	Min BoM for The Corp phone						X
Platformisation Technology	# of Technology 66 Lic. shipping > 3M/yr						X
Interface	Interface units	>11M	>13M	X		X	X
Operator preference	Top-3 operators issue RFQ spec The Corp			X		X	X
Productivity							
Get Torden	Lyn goes for Technology 66 in Sep-04	Yes		X		X	X
Fragmentation	Share of components modified	<10%	<5%		X	X	X
Commoditisation	Switching cost for a UI to another System						X
Duplication	The Corp share of 'in scope' code in best-selling device	>90%	>95%		X	X	X
Competitiveness	Major feature comparison with MX	Same	Better	X		X	X
User experience	Key use cases superior vs. competition	5	10	X	X	X	X
Downstream cost saving	Project ROI for Licensees	>33%	>66%	X	X	X	X
Platformisation IFace	Number of shipping Lic.	33	55	X		X	X
Japan	Share of of XXXX sales	>50%	>60%	X		X	X
	Numbers are intentionally changed from real ones						

Business Objectives Quantified

Strategy Impact Estimation



THE PRINCIPLE OF 'QUALITY QUANTIFICATION'

- **All qualities can be expressed quantitatively,**
- ***'qualitative'* does *not* mean unmeasurable.**

"In physical science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it.

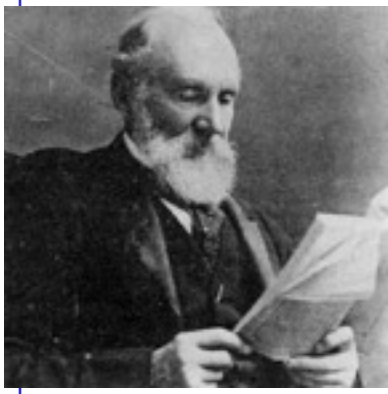
I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it;

but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind;

it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be."

Lord Kelvin, 1893

from
http://zapatopi.net/kelvin/quotes.html

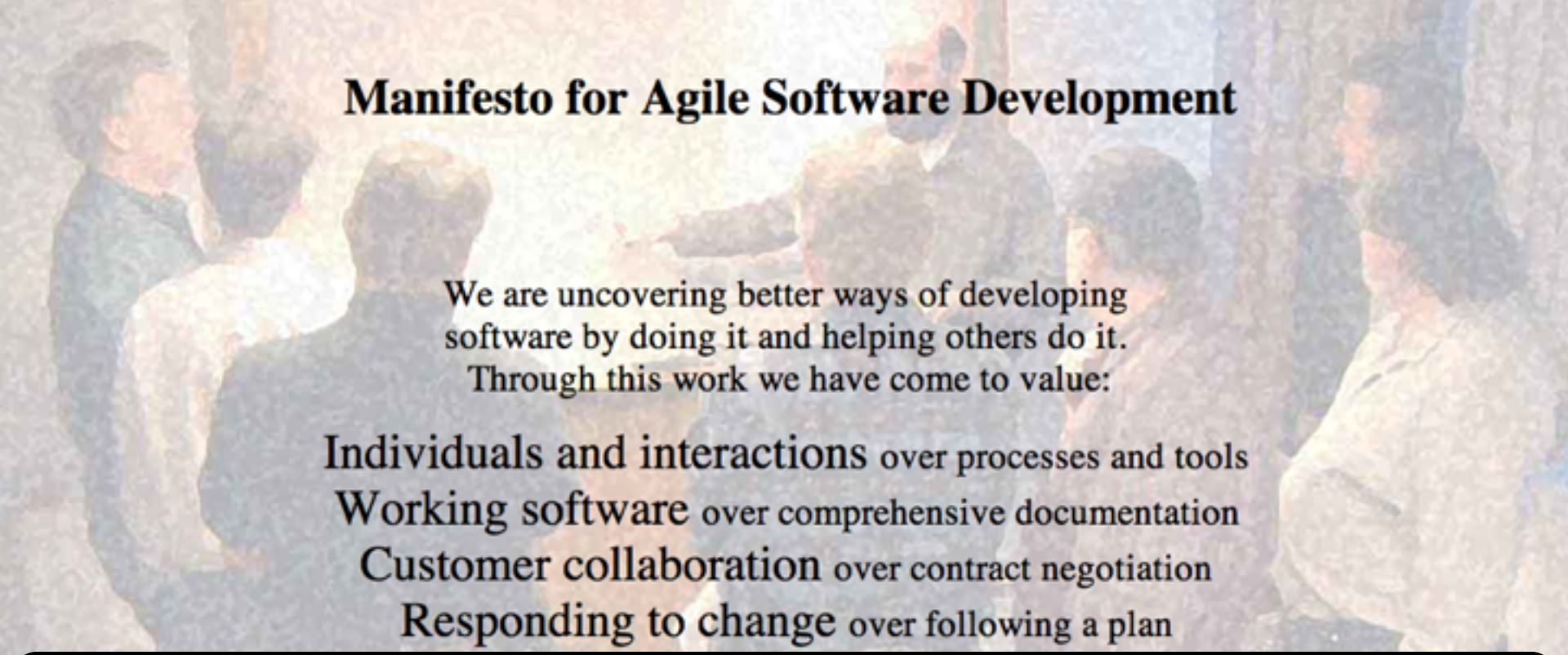


Value Management (Evo) with Scrum development

- developing a large web portal
www.bring.no dk/se/nl/co.uk/com/ee
at Posten Norge

We have a challenge ...

deliver
value to stakeholders,
within agreeable resources.



no external Value delivery?
not even a thought about Stakeholders?

It is all about YOU
"You, the developer, have become the center of the universe!"
<- Scott Ambler

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Working software is the primary measure of progress.

to maintain a constant pace indefinitely.

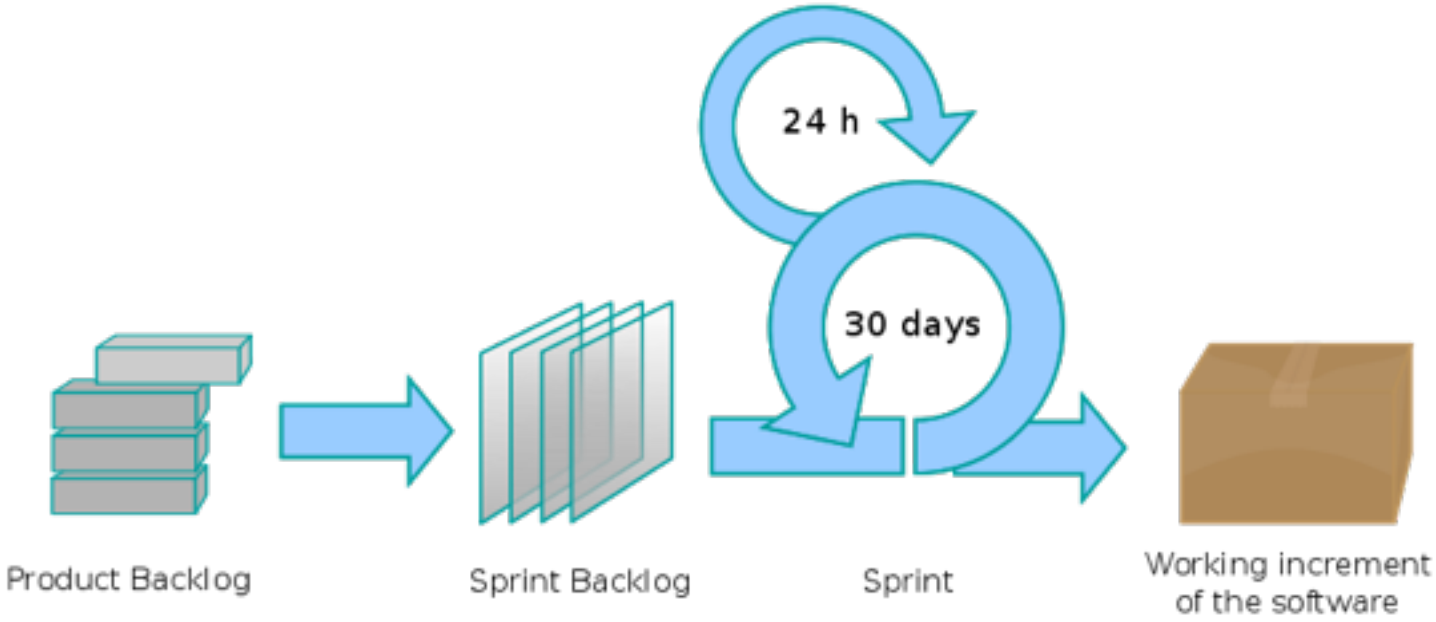
Continuous attention to technical excellence and good design enhances agility.

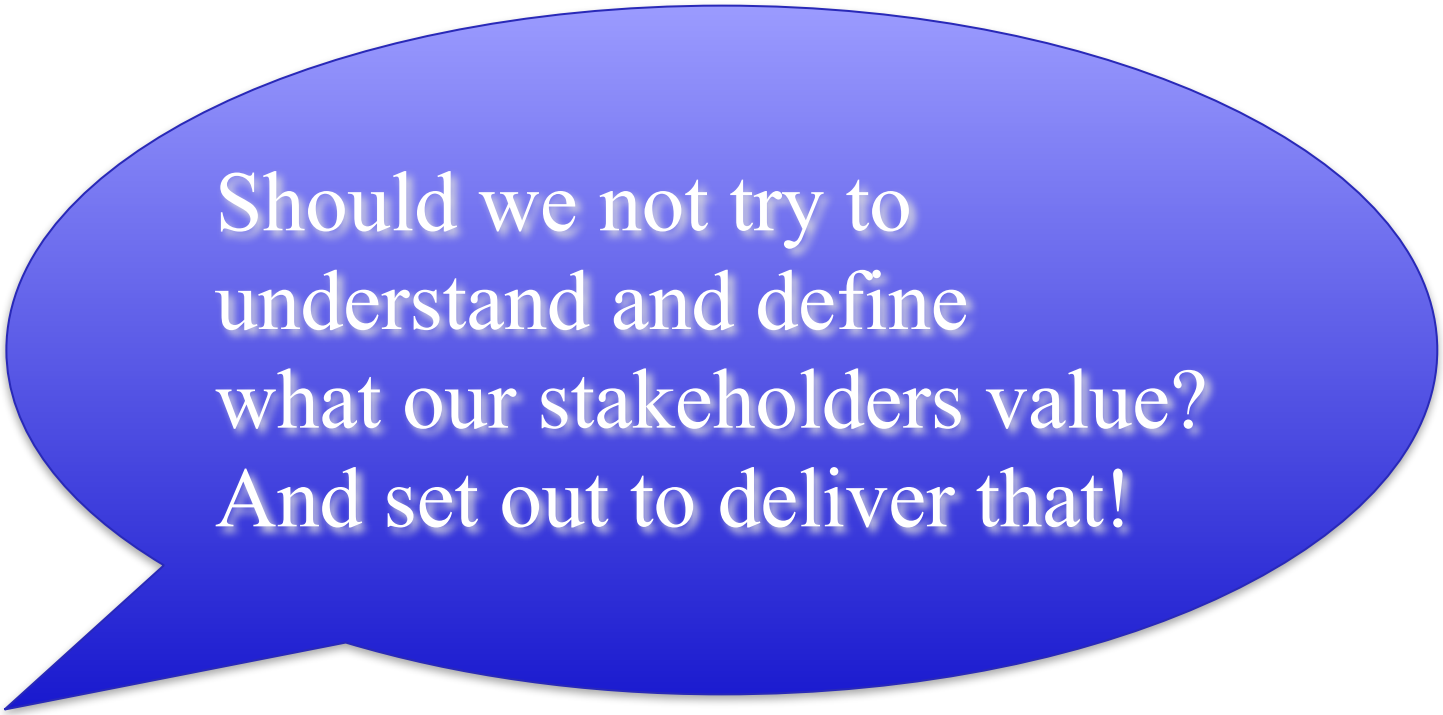
Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Scrum





"Our highest priority is to satisfy the customer
through early and continuous delivery of valuable software."

history

- **Posten Norge AS bought a series of companies**
  - within Logistics, Package transport, CRM and Storage
  - in Norway, Sweden, Denmark, Finland, UK, Holland and Estonia.

bring

VÅRE NETTSTEDER - OUR WEBSITES | POSTEN.NO | POSTEN KONSERN | Søk..

FORSIDE | PRODUKTER OG TJENESTER | KUNDESERVICE | JOBB I BRING NORGE | OM BRING

Velkommen til Bring

Hva kan vi hjelpe deg med?

- Basis brev- og pakkeprodukter
- Sende varer
- Lagre varer
- Postreklame og CRM
- Mange sendinger

Logg inn

Nyttige verktøy

Sporing

Sporing av pakker og brev på sending- eller kollinummer

Søk

Finn ut hvor din forsendelse befinner seg.

- SPORING FOR AIR
- SPORING FOR SEA
- SPORING FOR STYKK OG PARTIGODS

Bestill

LOAD.09

Adresseendring og oppbevaring

Finn postnummer

Reklamehjelperen

Finn åpningstid og postkontor

Bring Mail

Brev og postreklame. Effektiv distribusjon av Post i Norge og Norden.

- BRING MAIL
- Bring bedriftskort
- Massebrev
- Postreklame Uadressert

Bring Express

Levering samme dag med ekspress bud i Norge og resten av Norden. Transport med bil, sykkel eller fly.

- BRING EXPRESS
- QuickPack
- VIP-bud
- Distribusjonstjenester

Bring Logistics

Transport av gods og frakt av pakker i Norge og utlandet. Din partner for logistikk og lager i Norden.

- BRING LOGISTICS
- Stykkgods
- Bedriftspakke Dør-Dør
- Lagringstjenester (3PL)

Bring Dialogue

Bring Frigoscandia

bring

OUR WEBSITES Search..

BRING LOGISTICS | CUSTOMER SERVICE | CAREERS | ABOUT BRING LOGISTICS

Bring Logisti

Delivering Nordic quality, speed and punctuality in the supply of cargo and parcel transport, warehousing and associated logistical services to the UK market.

Useful tools

Quotation Enquiry

The quotation enquiry form allows you to enter all the details of your shipment and have Bring Logistics respond with a quotation. To open the form, click here.

Shortcuts

- LATEST NEWS
- SCANDINAVIA SERVICES
- CUSTOMER SERVICE BRING LOGISTICS
- ABOUT BRING LOGISTICS
- CUSTOMER QUESTIONNAIRE

Groupage and partload national

SEE ALL GROUP OPTIONS

- Groupage
- Dangerous Goods

Groupage and partload international

SEE ALL GROUP OPTIONS

- Groupage
- Special cargo

Warehousing

SEE ALL GROUP OPTIONS

- Warehousing services
- Short term storage

Some Players

Posten

Webteam - Value Management Certified

Project Owner: Anne Hognestad anne.hognestad@posten.no

Product Owner: Terje Berget terje.berget@posten.no

Lin Smitt-Amundsen & Kristin Nygård

Many Business Groups and internal stakeholders.

Kjetil Halvorsen kjetil.halvorsen@posten.no

Bekk & Ergo Group

Scrum Master: Fredrik Bach fredrik.bach@bekk.no

Technical Architect: Stefan M. Landrø: stefan.landro@bekk.no

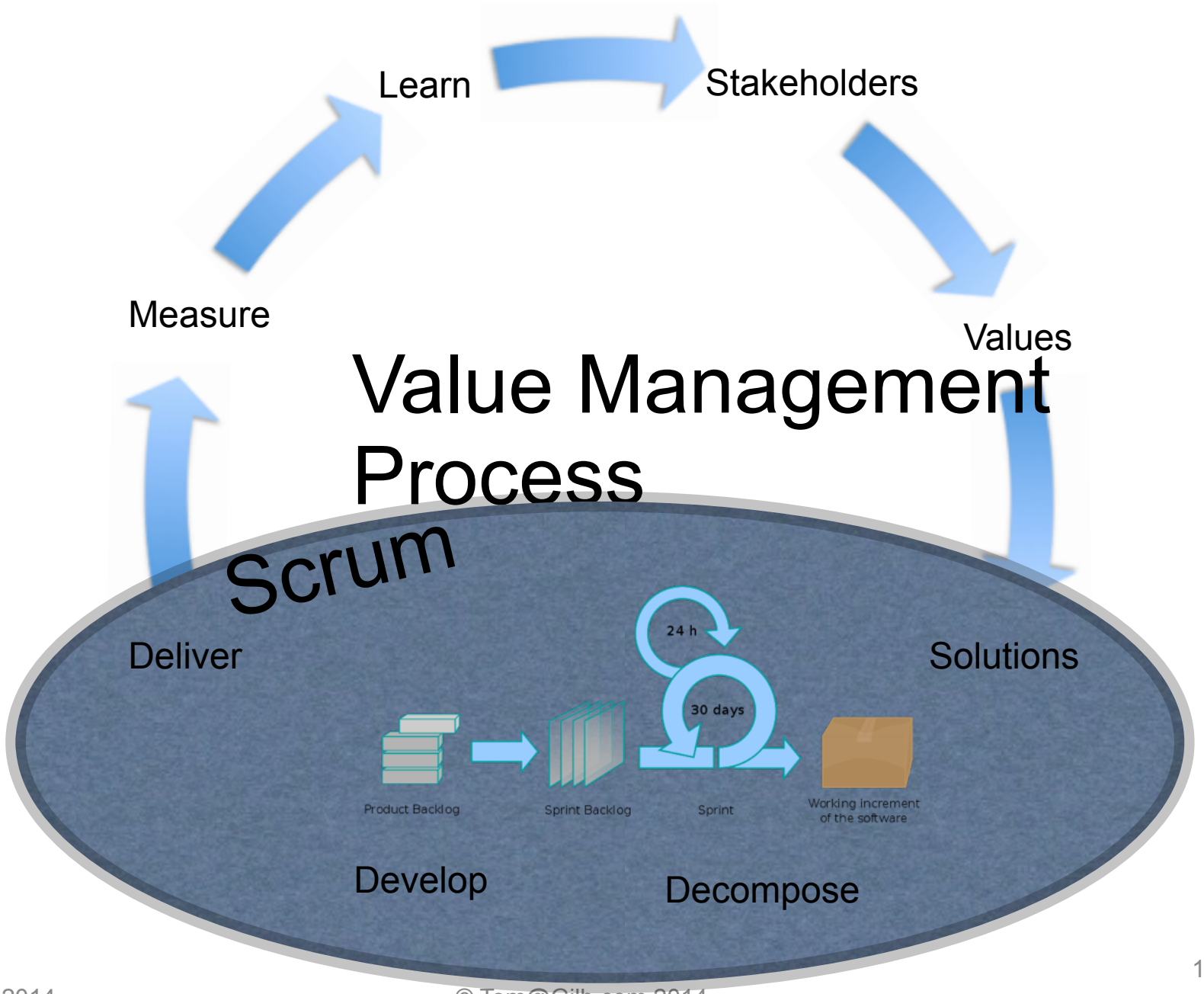
Graphics: Espen Satver

Morten Wille Johannessen, Markus Krüger, Dag Stepanenko

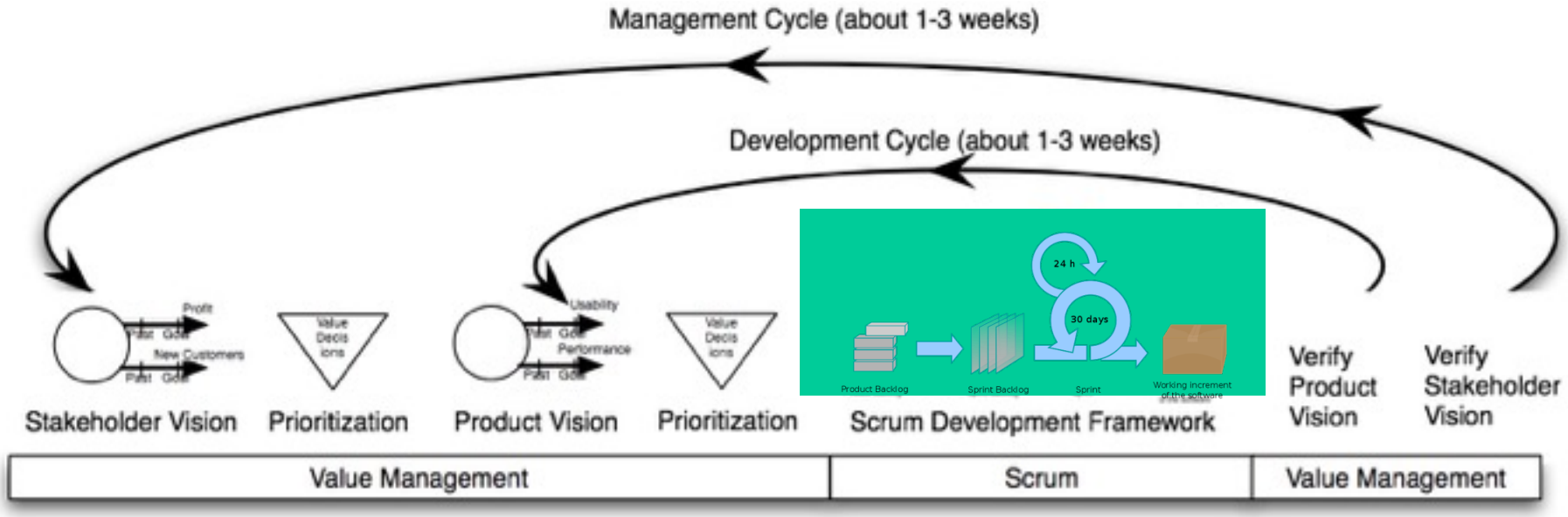
NetLife Research

User Experience: Gjermund Also gjermund@netliferesearch.com Kjell-Morten Bratsberg Thorsen

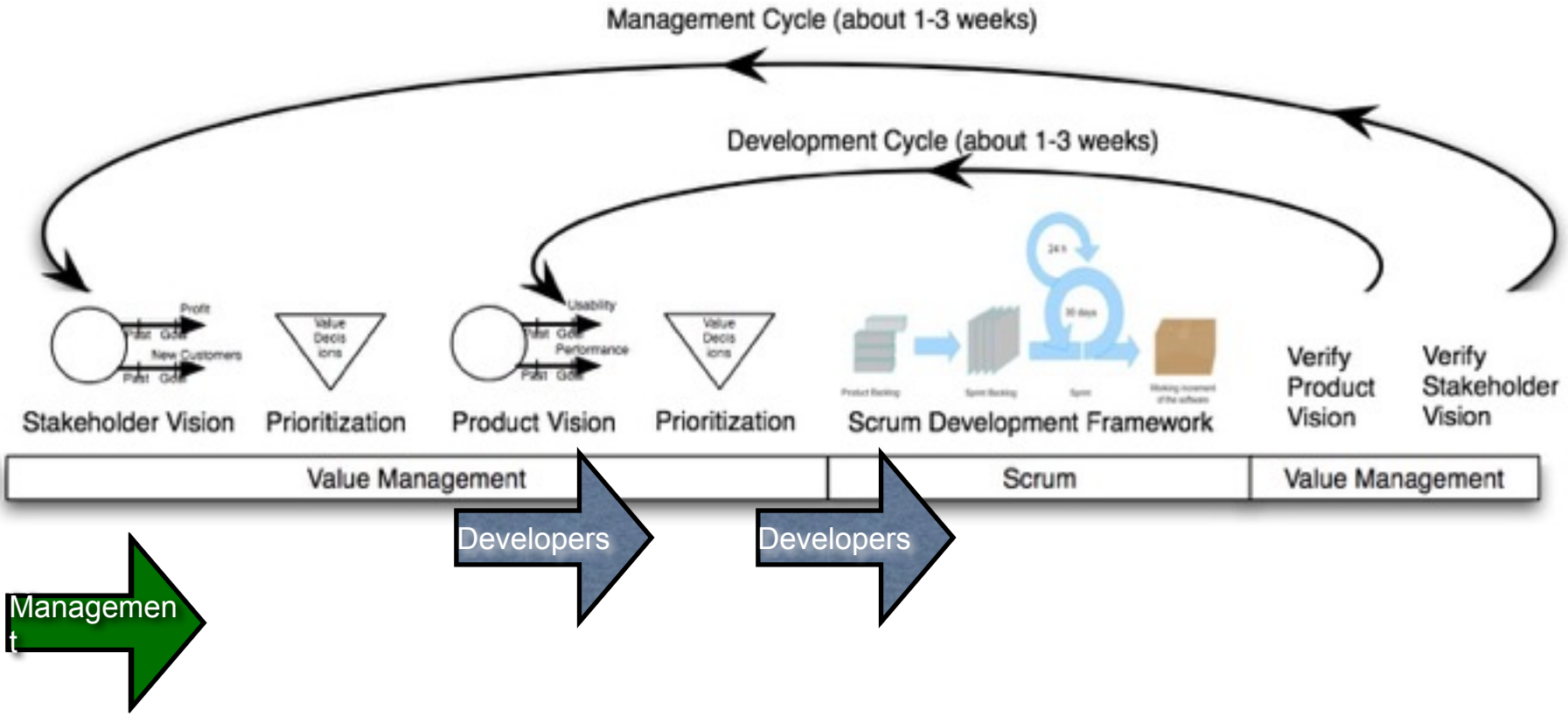
Kai Gilb: Management Coach: Kai



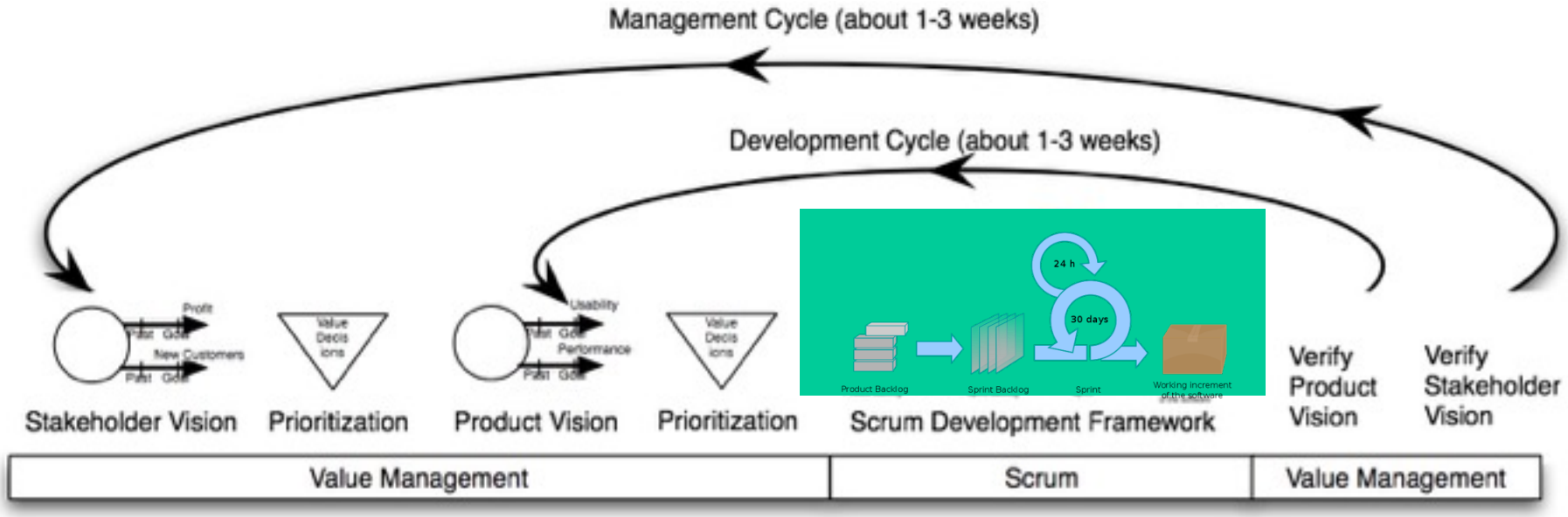
Value Management



Value Management



Value Management



Value Decision Tables

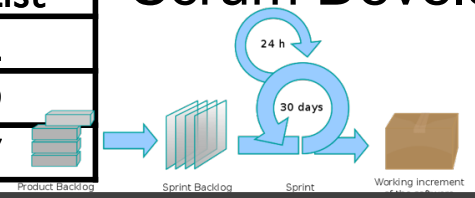
Business Goals	Stakeholder Value 1	Stakeholder Value 2
Business Value 1	-10%	40%
Business Value 2	50%	10%
Resources	20%	10%

Stakeholder	Product Value 1	Product Value 2
Stakeholder Value 1	-10%	50 %
Stakeholder Value 2	10 %	10%
Resources	2 %	5 %

Product Values	Solution 1	Solution 2
Product Value 1	-10%	40%
Product Value 2	50%	80 %
Resources	1 %	2 %

Prioritized List
1. Solution 2
2. Solution 9
3. Solution 7

Scrum Develops



We measure improvements
Learn and Repeat

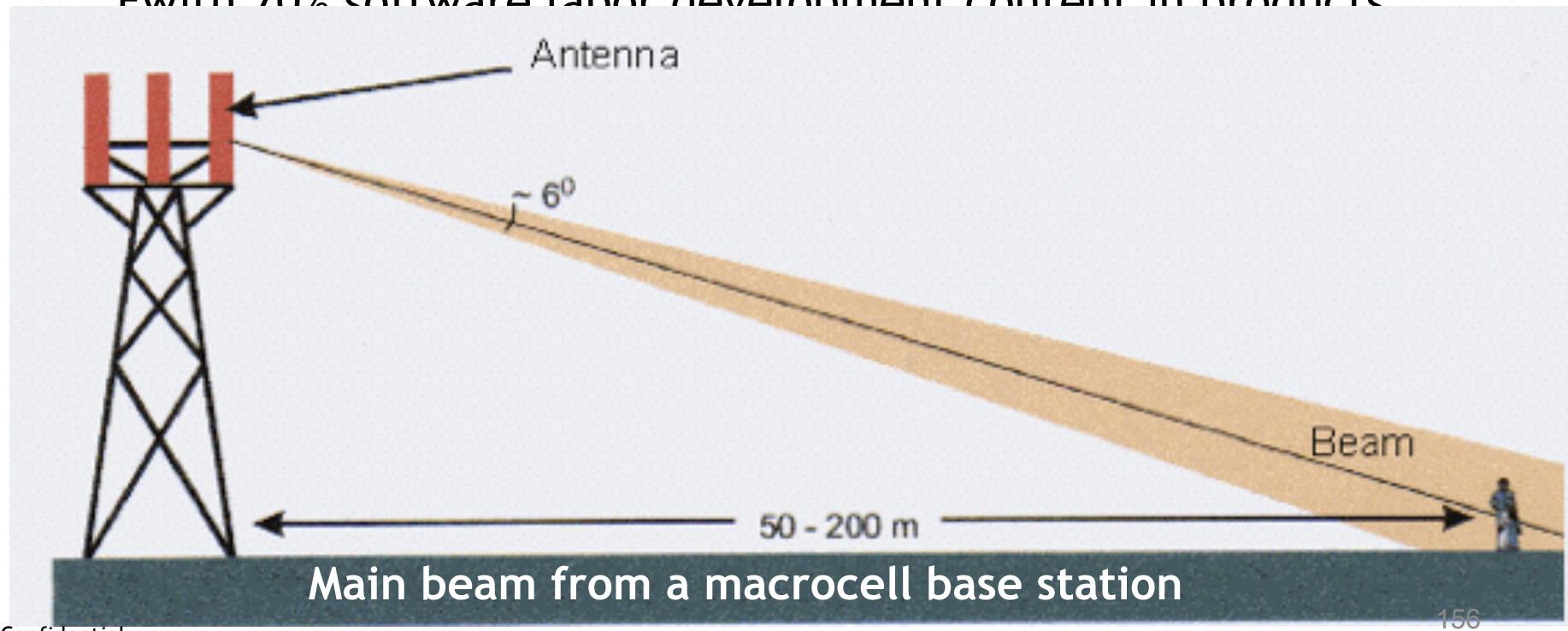
End of Bring Case

Software Engineering Productivity Study

ERICSSON

An example of setting objectives for process improvement

Ewith 70% software labor development content in products



Main beam from a macrocell base station

The problem

- Great Market Growth Opportunities
- Too Few Software Engineers

- Solution:
  - Increase productivity of existing engineers

The One Page Top Management Summary

(after 2 weeks planning)

The Dominant Goal

Improve Software Productivity in R PROJECT by 2X by year 2xxx

Dominant (META) Strategies

Continual Improvement (PDSA Cycles)
.DPP: Defect Prevention Process
.EVO: Evolutionary Project Management

Long Term Goal [2xxx+]

DPP/EVO, Master them and Spread them on priority basis.

Short Term Goal [Next Weeks]

DPP [RS?]
EVO [Package C ?]

Decision: {Go, Fund, Support}

The Ericsson Quality Policy:

ERICSSON

"every company shall *define* **performance** indicators (which) ..

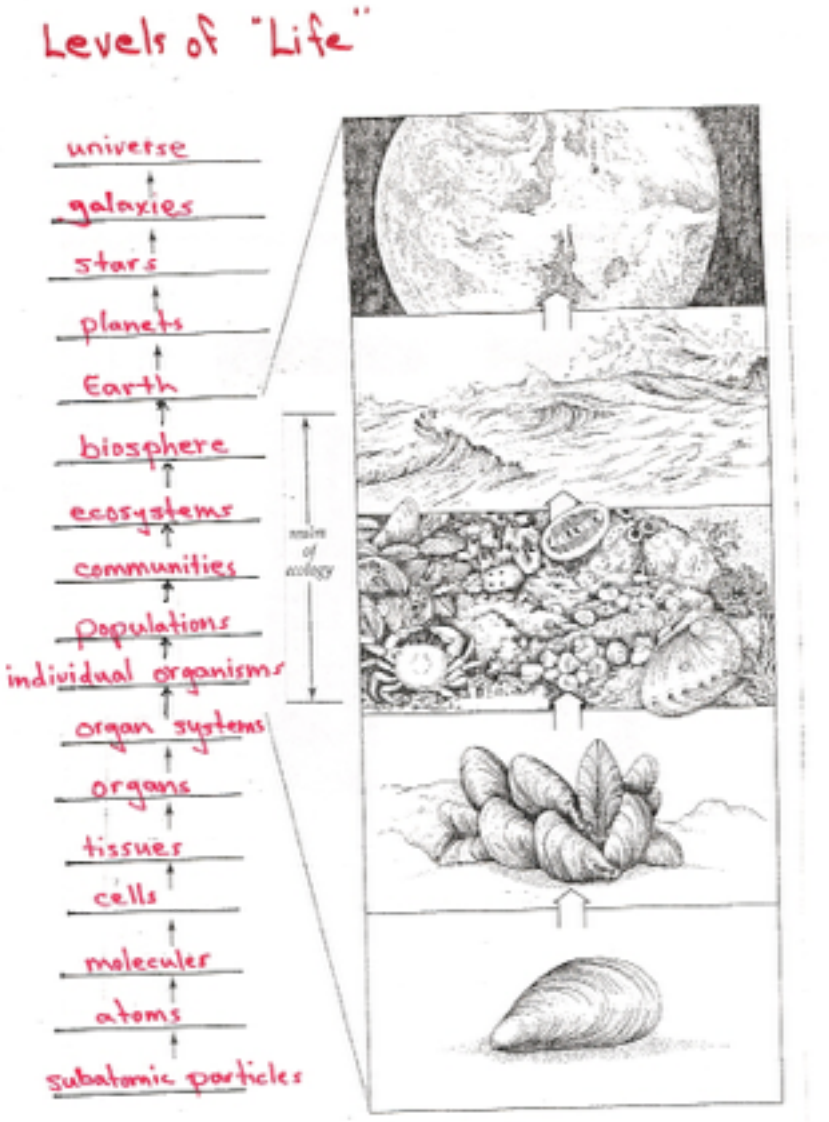
- reflect **customer satisfaction**,
- internal **efficiency**
- and business **results**.

- The performance indicators are used in **controlling** the operation."
- Quality Policy [4.1.3]

159

Levels of Objectives.

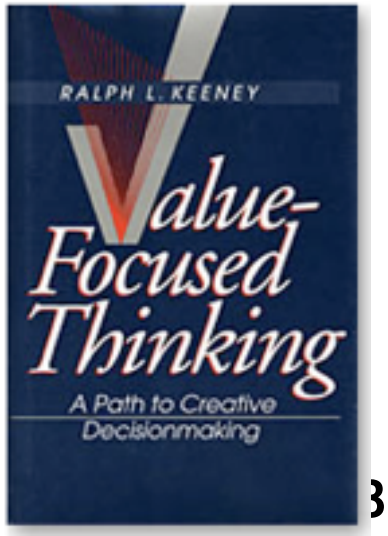
- **Fundamental Objectives**
- **Strategic Objectives**
- **Means Objectives:**
-
- **Organizational Activity Areas.**
  - Pre-study.
  - Feasibility Study.
  - Execution.
  - Conclusion.
- **Generic Constraints**
  - Political Practical
  - Design Strategy Formulation Constraints
  - Quality of Organization Constraints
  - Cost/Time/Resource Constraints



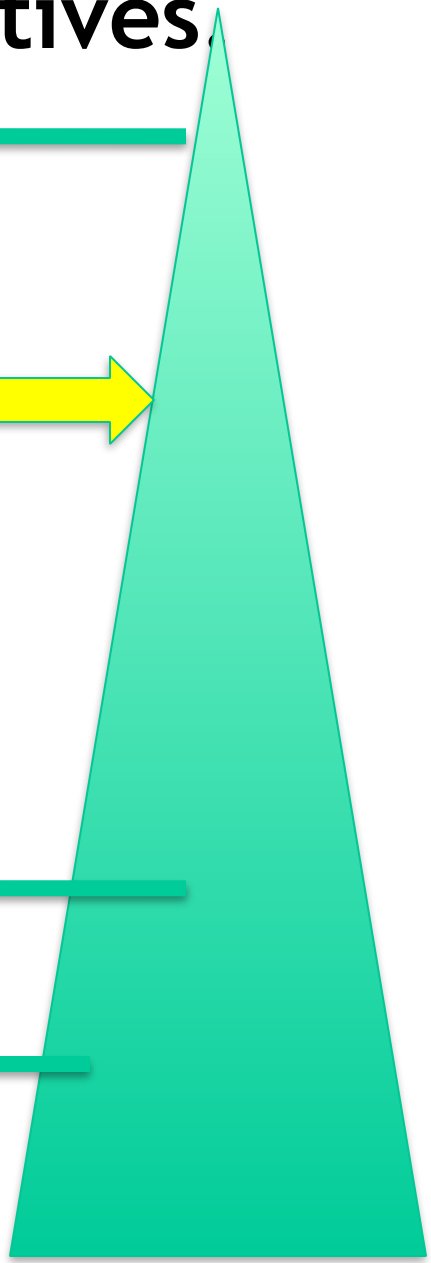
160

Keeney's: Levels of objectives

- 1. Fundamental Objectives
  - (above us)
- 2. Generic Constraints
  - (our given framework)
  - Political Practical
  - Design Strategy Formulation Constraints
  - Quality of Organization Constraints
  - Cost/Time/Resource Constraints
- 3. Strategic Objectives
  - (objectives at our level)
- 4. Means Objectives:
  - (*supporting* our objectives)



Constr aints



The Strategic Objectives (CTO level)

- Support
  - the **Fundamental** Objectives (Profit, survival)
  - **Software Productivity:**
    - **Lines of Code Generation Ability**
  - **Lead-Time:**
  - **Predictability.**
  - **TTMP: Predictability of Time To Market:**
  - **Product Attributes:**
  - **Customer Satisfaction:**
  - **Profitability:**

‘Means’ Objectives:

- Support the **Strategic** Objectives
  - ***Complaints:***
  - ***Feature Production:***
  - ***Rework Costs:***
  - ***Installation Ability:***
  - ***Service Costs:***
  - ***Training Costs:***
  - ***Specification Defectiveness:***
  - ***Specification Quality:***
  - ***Improvement ROI:***

"Let no man turn aside,
ever so slightly,
from the broad path of honour,
on the plausible pretence
that he is justified by the goodness
of his end.
All good ends can be worked out
by good means."
Charles Dickens

Strategies: (total brainstormed list) 'Ends for delivering Strategic Objectives'

-Evo [Product development]:

-DPP [Product Development Process]: Defect Prevention Process.

-Inspection?

-Motivation.Stress-Management-AOL

-Motivation.Carrot

-DBS

-Automated Code Generation

-Requirement -Tracability

-Competence Management

-Delete-Unnecessary -Documents

-Manager Reward:?

-Team Ownership:?

-Manager Ownership:?

- Training:?
- Clear Common Objectives:?
- Application Engineering area:
- Brainstormed List (not evaluated or prioritized yet)?
- Requirements Engineering:
- Brainstormed Suggestions?
- Engineering Planning:
- Process Best Practices:
- Brainstormed Suggestions?
- Push Button Deployment:
- Architecture Best Practices:
- Stabilization:
- World-wide Co-operation?

Principles for Prioritizing Strategies

- They are well-defined
  - Not vague
- The have some relevant predictable numeric experience
  - On main effects
  - Side effects
  - Costs
  - Risks - Uncertainty
- Not huge spread of experience

"Software Productivity" =

Lines of Code Generation Ability

-*"Software Engineering net production in relation to corresponding costs."*

-*Ambition: Net lines of code successfully produced per total working hours needed to produce them. A measure of the*

- *<u>efficiency</u> ('effective production/cost of production') of the organization in using its software staff.*

Scale: [Defined Volume, kNCSS or kPlex] per Software Development Work-Hour.

•**Scale: [Defined Volume, kNCSS or kPlex**

•**Software Development**: Defined:

•Productivity calculations include Work-H… …on Phase

• **Meter : <PQT Database and EPOS, CPA**

-*Comment: we <u>know</u> that real software … chosen this measure as it is available in our cu…*

-**P1: Past [1997, ERA/AR] < to be calcu…**

• **Past-R PROJECT: Past [1997, R PROJECT] < to be ca…ted when data available, available Volume/Work Hours >**

• **Past-EEI: Past [1997, Ireland, Plex] ___??__ kPLEX / Work-Hour.**

•**<add more like LuleÂ>**

•**Fail [end 1998, R PROJECT, <u>Same Reliability</u>] 1.5 x Past-R PROJECT <- R PROJECT AS 3 c " by 50%".**

-*"50% better useful code productivity in 1.5 years overall"*

•**Same Reliability: State: The Software Fault Density is not worse than with comparable productivity. Use official The Company Software Fault Density measures <- 1997 R PROJECT Balanced Scorecard (PA3).**

•**Goal [Year=2000, R PROJECT, Same Reliability] 2 x Past-R PROJECT,**

- **[Year=2005, RPL, Same Reliability] 10?? x Past-R PROJECT**

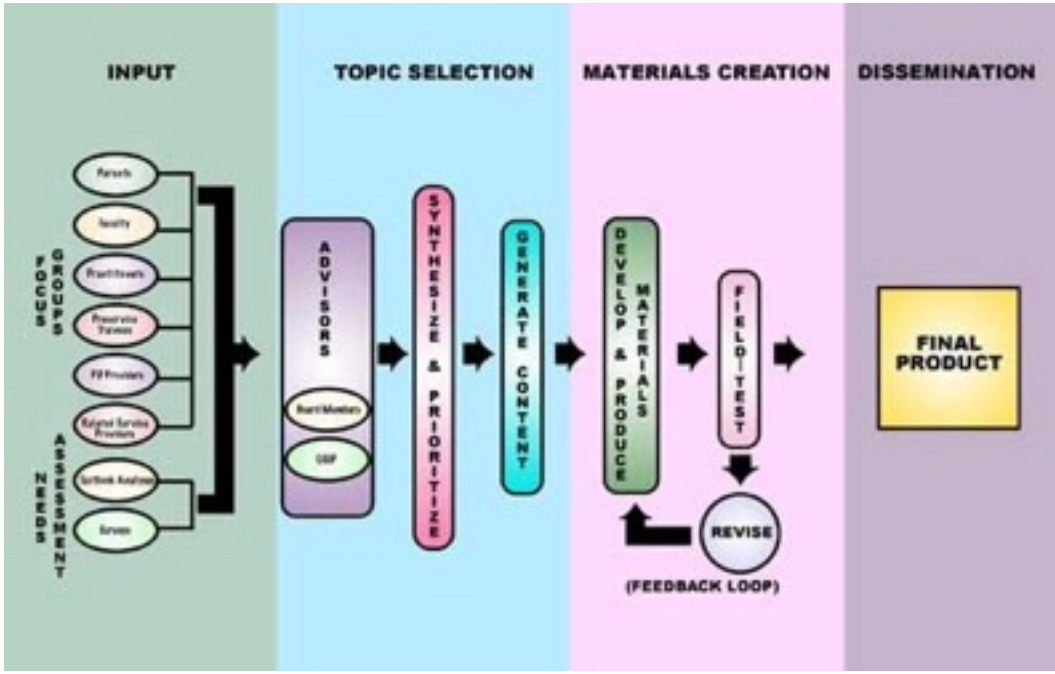
•**Wish [Long term, vs. D pack.] 10 x Past-R PROJECT "times higher productivity" <- R PROJECT 96 1.1 c**

•**Wish [undefined time frame] 1.5 x Past-R PROJECT <- R PROJECT AS 3 c " by 50%"**

-*Comment: May 13 1997 1600, We have worked a lot on the Software Productivity objectives (all day) and are happy that it is in pretty good shape. But we recognize that it needs more exposure to other people.*

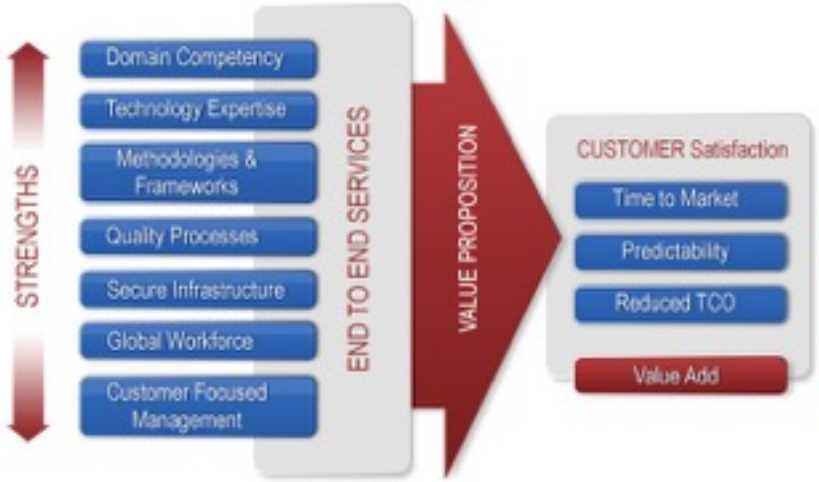
Lead-Time:

- **Lead-Time:**
  - *"Months for major Packages"*
- *Ambition: decrease months duration between major Base Station package release.*
- **Scale: Months from TG0, to successful first use for**
  - **major work station package.**
  - *Note: let us make a better definition. TG*
- Past [C Package, 1996?] 20? Months?? <-guess tg
- Goal [D-package] 18 months <- guess tg
- Goal [E-package and later] 10.8 Months <- R PROJECT 96 1.1 a "40% > D"
- Goal [Generally] ??? <- R PROJECT AS 3a
  - *"10% Lead-Time reduction compared to any benchmark".*



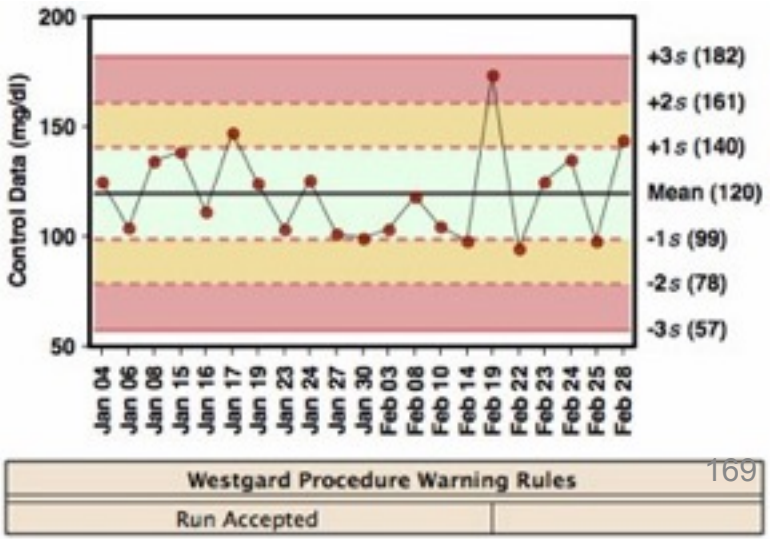
Predictability of Time To Market:

- **TTMP: Predictability of Time To Market:**
  - *Ambition: From Ideas created to customers can use it. Our ability to meet agreed specified customer and self-determined targets.*
  - **Scale: % overrun of actual Project Time compared to planned Project Time**
    - **Project Time: Defined:** time from the date of Toll-Gate 0 passed, or other Defined Start Event, to, the Planned- or Actually- delivered Date of All [Specified Requirements], and any set of agreed requirements.
    - **Specified Requirements: Defined:** written approved Quality requirements for products with respect to Planned levels and qualifiers [when, where, conditions].
      And, other requirements such as function, constraints and costs.
  - **Meter: Productivity Project or Process Owner will collect data from all projects, or make estimates and put them in the Productivity Database for reporting this number.**
  - **Past [1994, A-package] < 50% to 100%> <- Palli K. guess. [1994, B-package] 80% ?? <- Urban Fagerstedt and Palli K. guess**
  - **Record [IBM Federal Systems Division, 1976-80] 0% <- RDM 9.0 quoting Harlan Mills in IBM SJ 4-80**
  - ***"all projects on time and under budget"***
  - **[Raytheon Defense Electronics, 1992-5] 0% <- RDE SEI Report 1995 Predictability.**
  - **Fail [All future projects, from 1999] 5% or less <- discussion level TG**
  - **Goal [All future projects, from 1999] 0% or less <- discussion level TG**



Product Attributes:

- Product Attributes:
  - *"Keeping Product Promises."*
  - *Ambition: Ability to meet or bea agreed targets, both cost, time and quality. (except TTMP itself see above)*
- ***Scale*: % +/- deviation from [defined agreed attributes witl projects].**
- ***Past* [1990 to 1997, OUR DIVISION] at least 100% ???**
  - **<- Guess. Not all clearly defined and differences not**
    - **tracked. TSG**
- *Goal* [Year=2000, R PROJECT] **near 0%** negative deviation <- TsG for discussion.



Customer Satisfaction

Customer Satisfaction:
"Customer Opinion of Us"
Scale: average survey result on scale of 1 to 6 (best)
Meter: The Company Customer Satisfaction Survey
Past [1997] **4**
Goal [1998-9?] **5** <- R PROJECT 96 1.1 b

Profitability

- **Profitability:**
  - ***"Return on Investment."***
  - ***<u>Ambition</u>: Degree of saleable product ready for installation.***
  - **<u>Scale</u>: Money Value of Gross Income derived by**
    - **[All R PROJECT Production OR**
    - **defined products] for**
    - **[Product Lifetime OR**
    - **a defined time period]**
  - **Goal: <we did not complete this>**

‘Means Objectives’ Samples
Same *definition* process as higher level objectives

Means Objectives

- *"support Strategic Objectives"*

- Summary:
  - '**Means Objectives**' are
    - not our major Strategic Objectives
    - but each one represents areas which if improved
      - will normally help us achieve our Strategic Objectives.
  - Means Objectives have a lower priority than Strategic Objectives.
  - They must never be 'worked towards'
    - to the point where they reduce our ability to meet Strategic Objectives.

Complaints

Complaints:

"Customer complaint rate to us"

Ambition:

Means Goal: for Customer Satisfaction (Strategic).

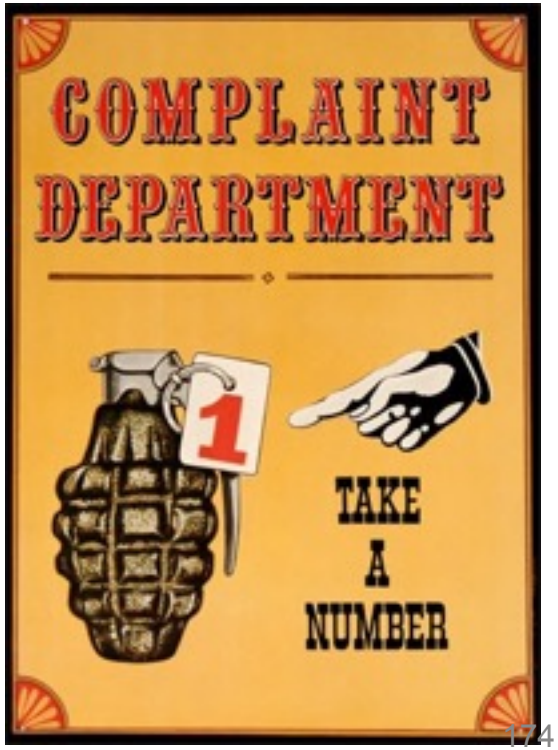
Scale: number of complaints per customer in [defined time into <operation>]

Past [Syracuse Project , 1997] ?? <bad> <- ML

Goal [Long term, software component, in first 6 months in Operation] **zero complaints** <- R PROJECT 96 1.1 b

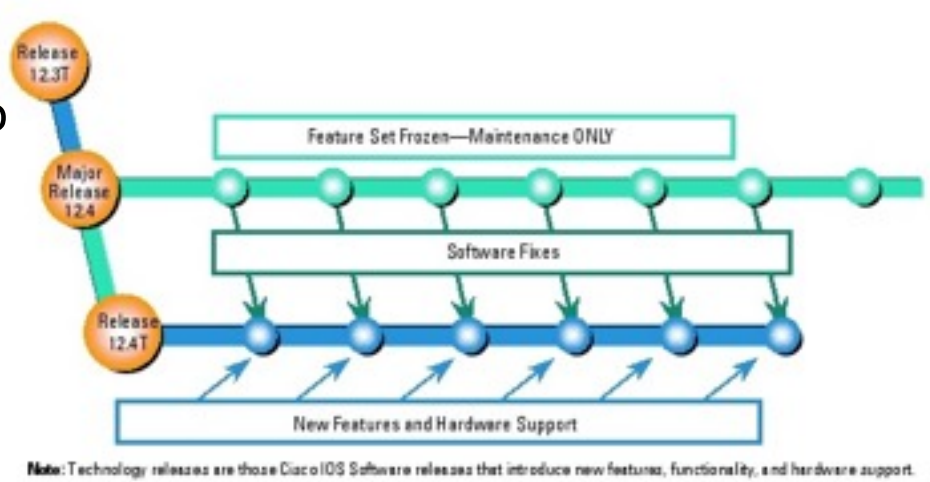
"zero complaints on software features"

Impacts: <one or more strategic objectives>



Feature Production:

- **Feature Production:**
  - *"ability to deliver new features to customers"*
  - ***Ambition**: reverse our decreasing ability to deliver new features <- R PROJECT AS 1.1*
  - **Scale**: Number of new prioritized <Features> delivered successfully to customer per year per software development engineer.
  - Too Little: **Past** [1997] ?? "estimate needed, maybe even definition of feature"
  - **Goal** [1998-onwards] **Too Little + 30% annually**?? <-For discussion purposes TsG.
  - "we need to drastically change our ability to effectively develop SW" <- R PROJECT AS 1.1



Improvement ROI:

Improvement ROI:

"Engineering Process Improvement Profitability"

Ambition: Order of magnitude return on investment in process improvement.

Scale:

The average [annual OR defined time term] Return on Investment in Continuous Improvement as a ratio of [Engineering Hours OR Money]

Note: The point of having this objective is to remind us to think in terms of real results for our process improvement effort, and to remind us to prioritize efforts which give high ROI. Finally, to compare our results to others. <-TsG

Record

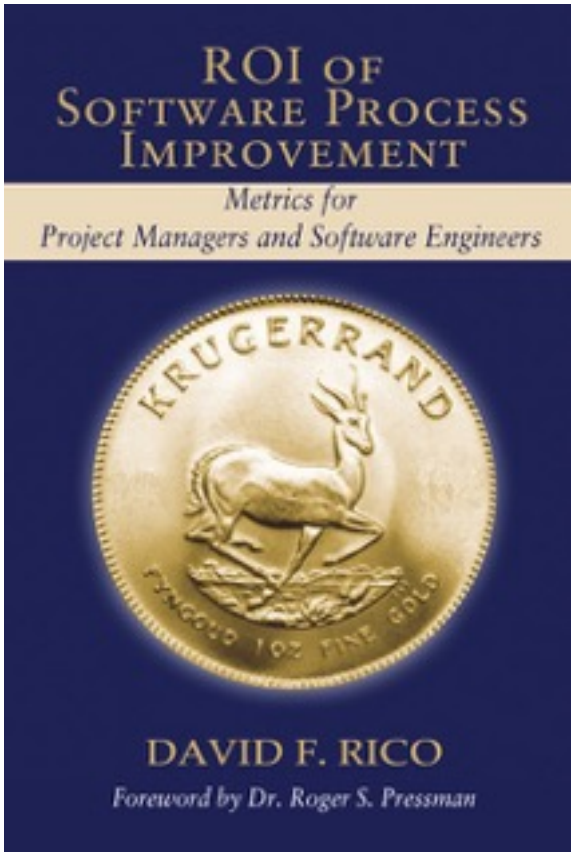
[Shell NL, Texas Instruments , Inspections] 30:1 <- Independently published papers TsG

Past

[IBM RTP, 1995, DPP Process] 13:1 <- Robert Mays, Wash DC test conference slides TsG

[Raytheon, 1993-5, Inspection & DPP] $7.70:1 <- RDE Report page 51 ($4.48 M$0.58M) Includes detail on how calculated. PK has copy.

[IBM STL, early 1990's] Average 1100% ROI (11:1) <- IBM Secrets pp32. PK has copy. NB Conservative estimate. See Note IBM ROI below.



2004

Simon Ramo (tRw)

"No matter how complex the situation,

good systems engineering involves putting value measurements on the important parameters of desired goals and performance of pertinent data,

and of the specifications of the people and equipment and other components of the system.

It is *not easy* to do this

and so, very often, we are *inclined to assume that it is not possible* to do it to advantage.

But skilled systems engineers can

change evaluations and comparisons of alternative approaches from purely speculative to highly meaningful.

If some *critical aspect* is not known,

the systems experts seek to make it known.

They go dig up the facts.

If doing so is very tough, such as setting down the public's degree of acceptance among various candidate solutions, then perhaps the *public can be polled*.

If that is not practical for the specific issue, then at least an attempt can be made to *judge the impact of being wrong in assuming the public preference*.

Everything that is clear is used with clarity:

what is not clear is used with clarity as to the estimates and assumptions made,

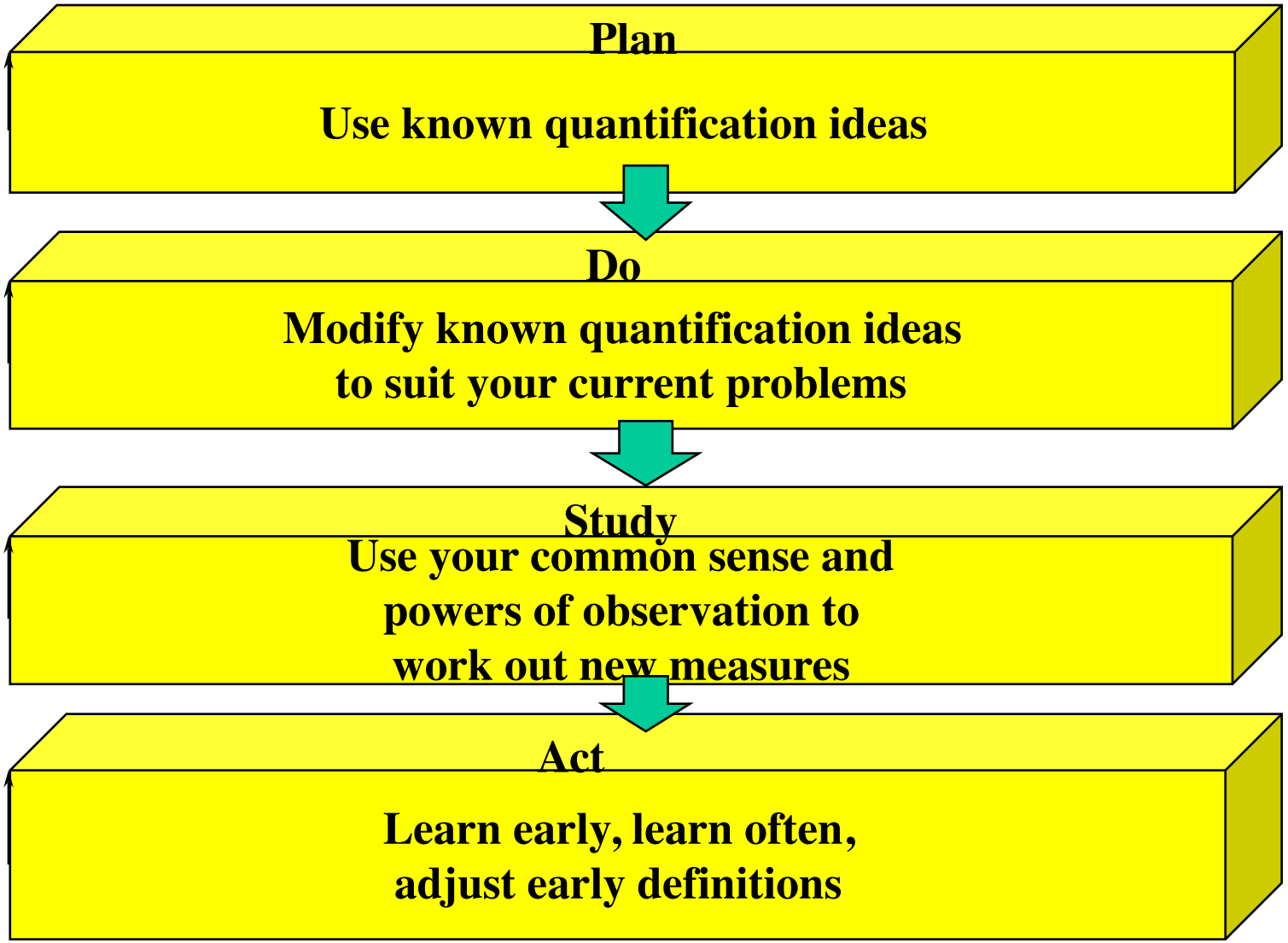
with the possible negative consequences of the assumptions weighed and integrated.

We do not have to work in the dark, now that we have professional systems analysis.

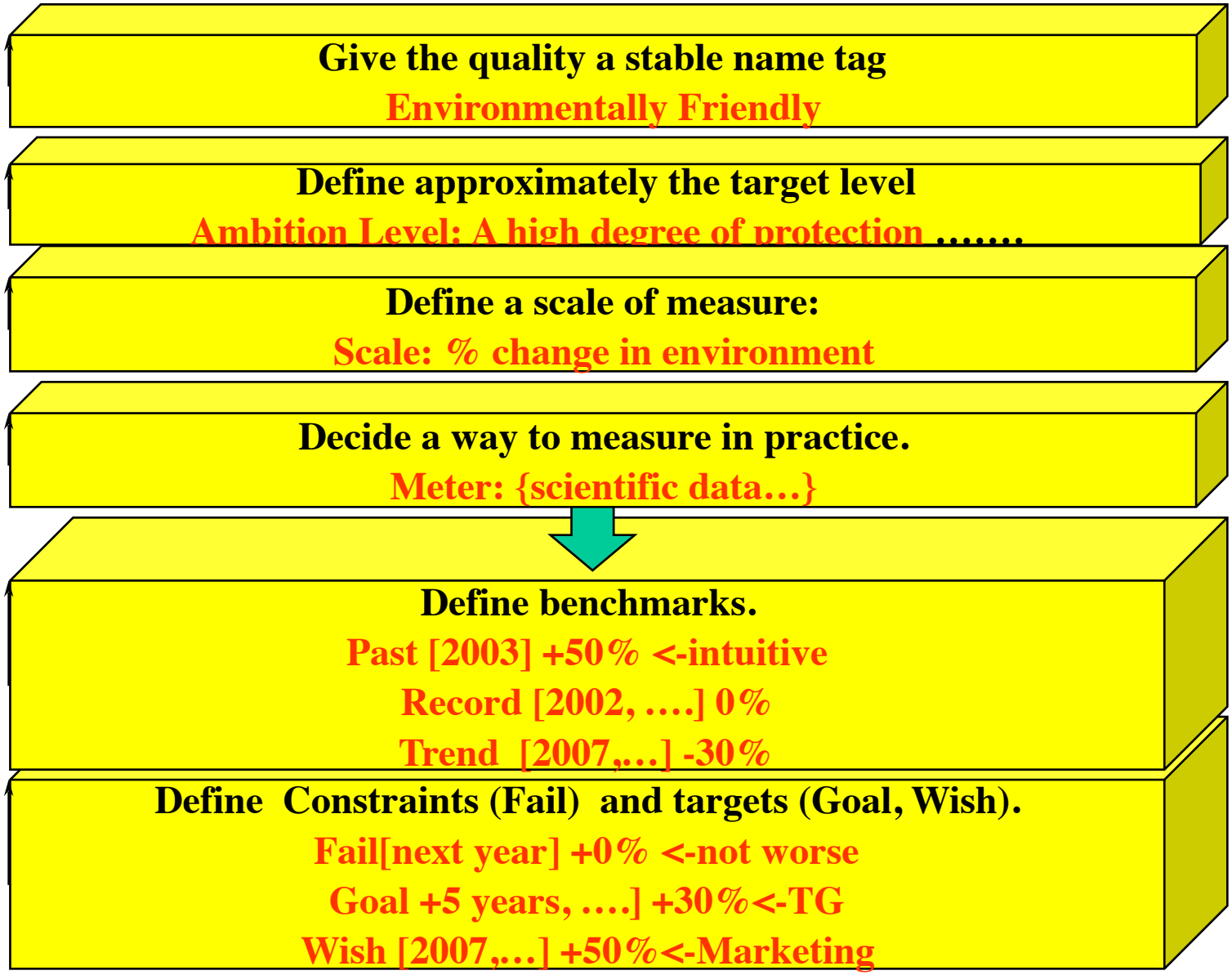
Ramo98 page 81

Simon Ramo and Robin K. St.Clair, The Systems Approach: Fresh Solutions to Complex Civil Problems Through Combining Science and Practical Common Sense, 1998, 150pp, © TRW, Inc., Manufactured in USA, KNI Incorporated, Anaheim CA. Free copy at TRW Stand at INCOSE conference 2002.

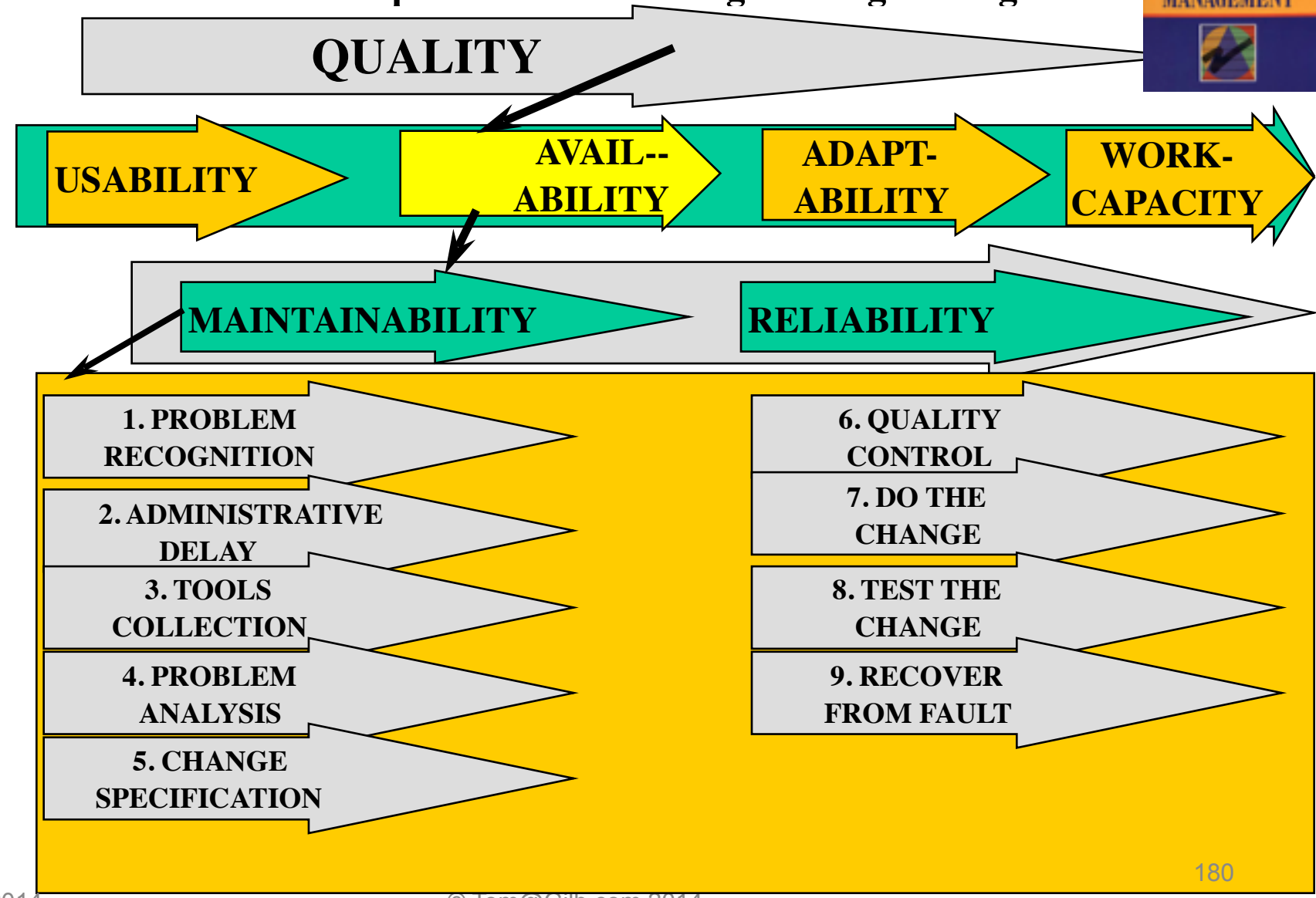
How to Quantify Quality



‘Environmentally Friendly’ Quantification Example



Devices to help quantify quality ideas: Standard Hierarchy of Concepts from Gilb: Principles of Software Engineering Management.



Using 'Parameters' when defining a Scale of Measure

Goal
[Users = NOVICES,
Components = USER MANUAL,
Tasks = ERROR CORRECTION]
60%

- ***Using [qualifiers] in the SCALE definition***
  - ***gives flexibility of detailed specification later***.
- Example
  - SCALE: the % of
    - **defined [Users]**
    - using **defined [system Components]**
    - who can successfully accomplish **defined [Tasks]**

[Scale Parameters]

Quality Quantification Process

(full detail 'Competitive Engineering', Scales chapter, & slide here later 'QQ')

Entry

E1. Do not enter if you can reuse existing standards.
E2.Do not enter if your source documents are poor.

Procedure

P1. Use applicable rules (GR, QR, QQ).
P2. Build list of quality ideas needing control.
P3. Detail qualities by exploding hierarchically.
- use evolutionary or pilot *feedback*.
P4. Revise your draft based on *design work*.
P5. Quality Control the specification.
P6. Get experience and then revise specifications.

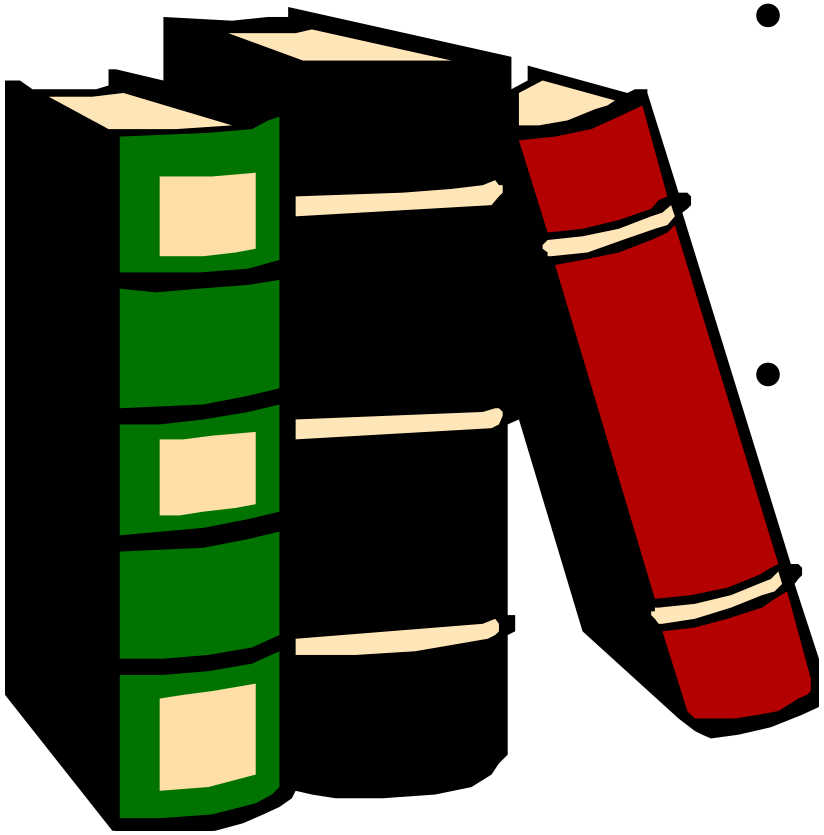
Exit

X1. Don't exit if *calculated remaining defects* are more than one per page.
X2. Unless you intentionally do so to learn more from experience.

Quantify for realistic judgements

- *"To leave [soft considerations] out of the analysis*
  - *–simply because they are **not readily quantifiable***
  - *–or to avoid introducing "personal judgments,"*
  - *– clearly biases decisions against investments*
    - *that are likely to have a significant impact on considerations*
      - *– as the quality of one's product, delivery speed and reliability, and the rapidity with which new products can be introduced"*
- ← *R. H. Hayes et al "Dynamic Manufacturing", p. 77 in MINTZBERG94: page124*

Principles for Quality Quantification.



- Some hopefully deep and useful guidelines
- to help you quantify quality ideas

0. THE PRINCIPLE OF 'BAD NUMBERS BEAT GOOD WORDS' (re-visited!)

- *Poor* quantification is more useful than none;
- at least it can be improved systematically.

State of the Art Flexibility

Not Clear! *Enhanced Usability*

Improved Performance

1. THE PRINCIPLE OF 'QUALITY QUANTIFICATION'

- All qualities can be expressed quantitatively,
- *'qualitative'* does *not* mean unmeasurable.

"If you think you know something about a subject, try to put a number on it. If you can, then maybe you know something about the subject. If you cannot then perhaps you should admit to yourself that your knowledge is of a meager and unsatisfactory kind.

Lord Kelvin, 1893

- Most quality ideas
  - are usefully broken into *several* measures of goodness.

Usability:

Entry Qualification: Scale IQ, …….

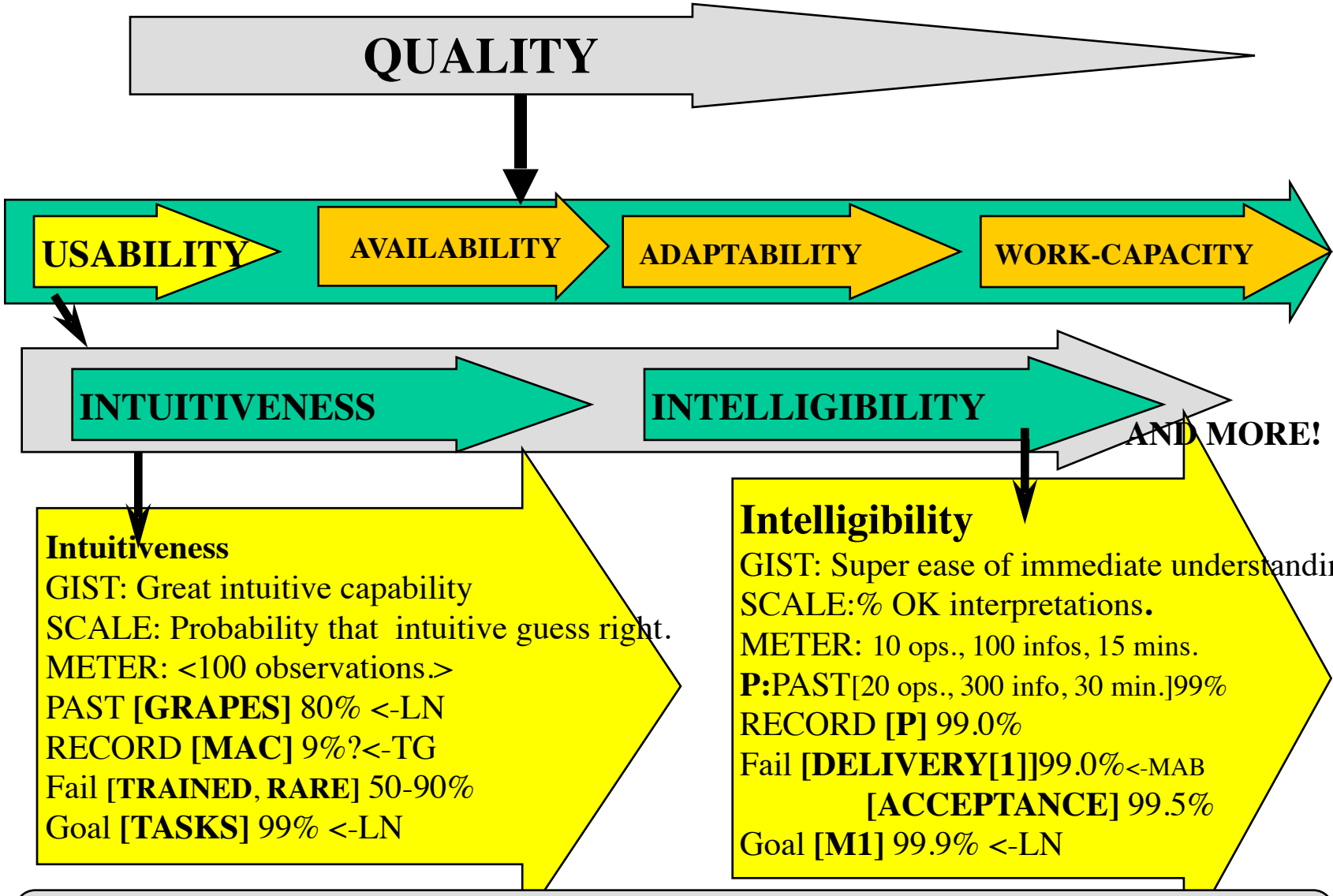
Learning Effort: Scale: Hours to learn, …..

Productivity: Scale: Tasks per hour,…….

Error Rate: Faults per 100 tasks, …..

Like-ability: % Users who like the system, ….

Quantifying Usability (Real C&C System)



TRAINED: DEFINED:C&Ctl. operator, approved course, 200 hours duration.
RARE: DEFINED: types of tasks performed less than once a week per op.
TASKS: DEFINED: onboard operator distinct tasks carried out.
ACCEPTANCE: DEFINED: formal acceptance testing via customer contract.
DELIVERY: DEFINED: Evolutionary delivery cycle, integrated and useful.

3. THE PRINCIPLE OF 'SCALAR DEFINITION'

- A *Scale of measure* is a powerful practical *definition* of a quality

Flexibility:

Scale: Speed of Conversion to New Computer Platform

HOW WE SPECIFY SCALAR ATTRIBUTE PRIORITY

<name tag of the objective>

Ambition: **<give overall real ambition level in 5-20 words>**

Version: **<dd-mm-yy each requirements spec has a version, at least a date>**

Owner: **<the person or instance allowed to make official changes to this requirement>**

Type: **<quality|objective|constraint>**

Stakeholder: **{ , , } "who can influence your profit, success or failure?"**

Scale: **<a defined units of measure, with [parameters] if you like>**

Meter **[<for what test level?>]**

====Benchmarks ========== the Past

Past [ ] **<estimate of past> <--<source>**

Record **[<where>, <when >, <estimate of record level>] <-- <source of record data>**

Trend **[<future date>, <where?>] <prediction of level> <-- <source of prediction>**

===== Targets ========== the future needs

Wish **[ ] <-- <source of wish>**

Goal **[…] <target level> <-- Source**

Value **[Goal] <refer to what this impacts or how much it creates of value>**

Stretch **[ ] <motivating ambition level> <-- <source of level>**

========== Constraints ==========

Fail **[ ] <-- <source> 'Failure Point'**

Survival **[ ] <- <source of limit> 'Survival Point'**

4. THE PRINCIPLE OF 'THREATS ARE MEASURABLE'

- **If *lack of quality* can destroy your project**
- **then you <u>can</u> measure it *sometime*;**
- **the only discussion will be 'how early?'.**

5. THE PRINCIPLE OF 'LIMITS TO DETAIL'

- **There is a *practical* limit to the number of facets of quality you can define and control,**
- **which is far less than the number of facets that you can *imagine* might be relevant.**

6. THE PRINCIPLE OF 'METERS MATTER'

Practical measuring instruments
improve
the *practical understanding*
and *application*
of ‘Scales of measure’.

Portability:

Scale: Cost to convert/Module

Meter [Data] measure/1,000 words converted

Meter [Logic] measure/1,000 Function Points Converted

Different quality-Scale *measuring processes*

will be necessary

for different *points in time*, different *events* and different *places*.

Availability:
Scale: % Uptime for System
Meter [USA, 2001] Test X
Meter [UK, 2002] Test Y

8. THE PRINCIPLE OF 'BENCHMARKS'

Past history and future trends *help* define words like "improve" and "reduce".

Reliability

Scale: Mean Time To Failure

Past [US DoD, 2002] 30,000 Hours

Trend [Nato Allies, 2003] 50,000 Hours

Goal [UK MOD, 2005] 60,000 Hours

9. THE PRINCIPLE OF 'NUMERIC FUTURE'

Numeric future requirement levels *complete* the quality definition of relative terms like 'improved'.

Usability:

Scale: Time to learn average task.

Past [Old product, 2003] 20 minutes

Wish [New product, 2007] 1 minute

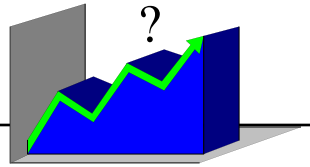
Stretch [End 2008, Students] 2 minutes

Goal [End 2005, Teachers] 5 minutes

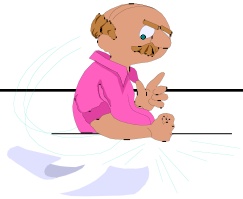
Some Planguage 'Quality Quantification' Concepts

PAST: any useful reference point. Your old product, a competitors organization, a quality achieved in same discipline but different branch of business.

RECORD: best in some class, state of the art. Something to beat. A challenge for you. An extreme PAST.



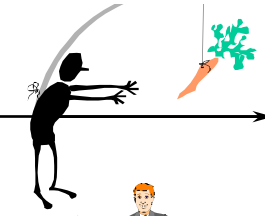
TREND: a future ? guess based on the PAST.



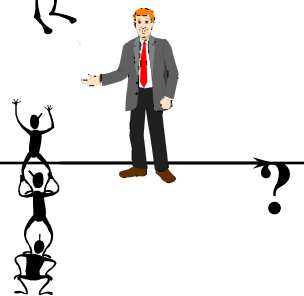
Survival : a level needed for survival of the entire system.

[-----]

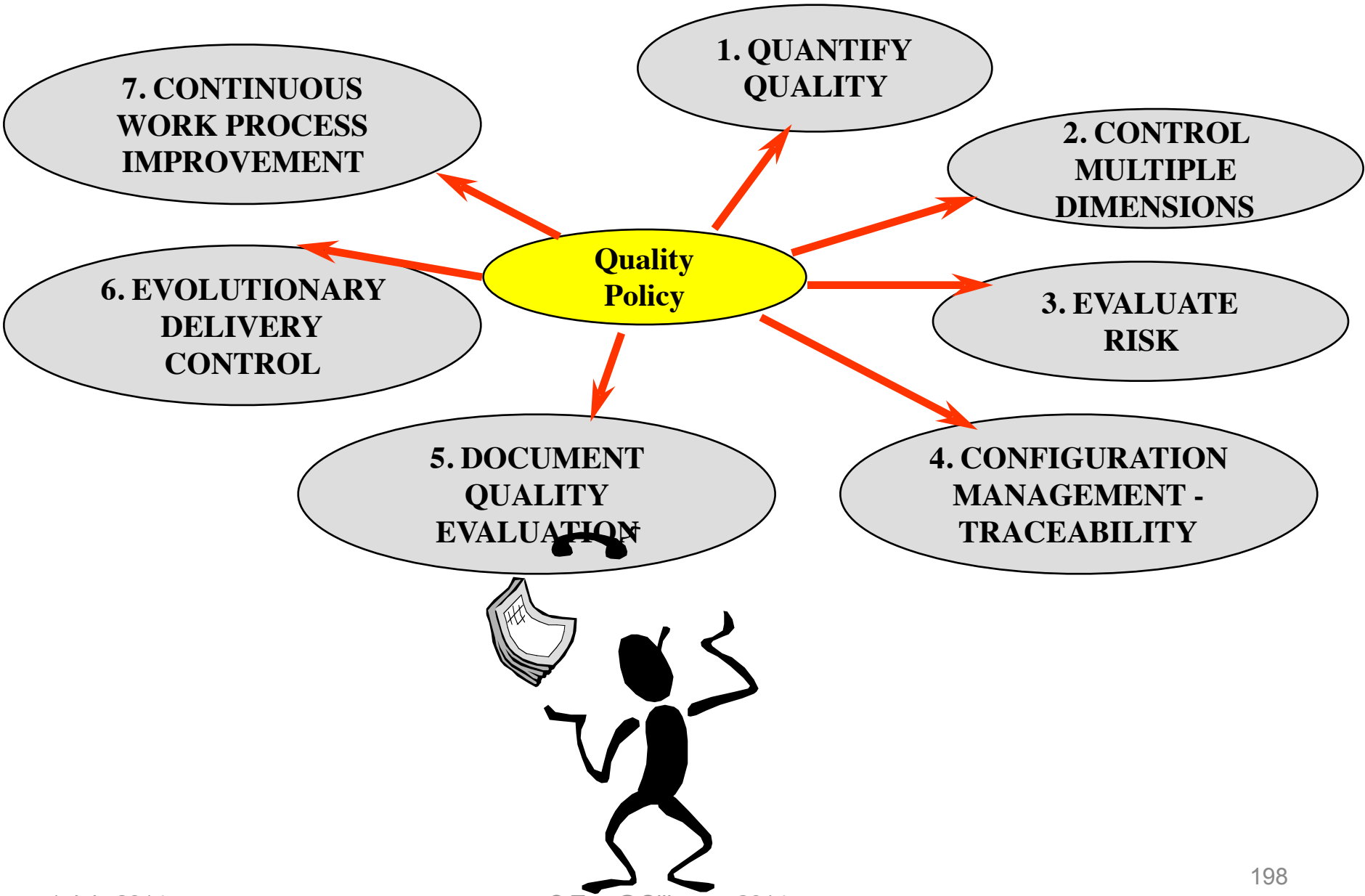
Goal: the level needed for satisfaction, happiness, joy and 100% full payment!



Wish: a level desired by someone, but which might not be feasible. Project is not committed to it.



A Corporate Quality Policy (Euro Multinational)



Policy on QUANTIFICATION, CLARIFICATION AND TESTABILITY OF CRITICAL OBJECTIVES:

"All critical factors or objectives
(quality, benefit, resource)
for any activity
(planning, engineering, management)
shall be expressed clearly, measurably,
testably and unambiguously
at all stages of consideration, presentation,
evaluation, construction and validation. "

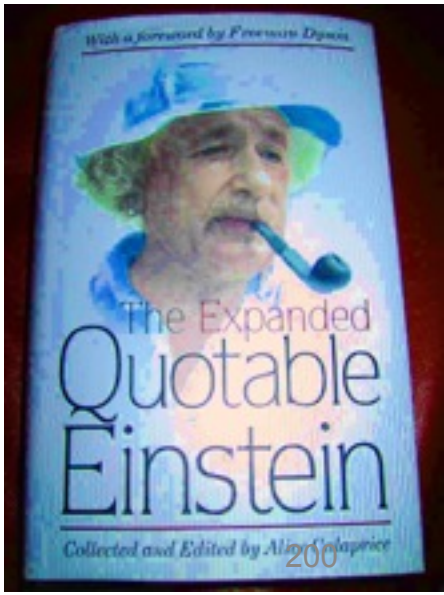
<- (Quality Manual Source is) 5.2.2, 4.1.2, 4.1.5, 5.1.1, 6.1, 6.4.1, 7.1.1, 7.3 and many others.

Einstein on Stretching

- "One should not pursue goals that are easily achieved.
- One must develop an instinct for what one can just barely achieve through one's greatest efforts." (1915)

"We have to do the best we can.

This is our sacred human responsibility" (1940)



Source detail in notes section of this slide. (Calaprice, 2000)

"Estimation: A Paradigm Shift Toward Dynamic Design-to Cost and Radical Management"

Scandinavian Developers Conference, Gothenburg, Sweden

March 4th 2013, 1140 to 1230 (50 mins.)

By Tom Gilb

Tom@Gilb.com

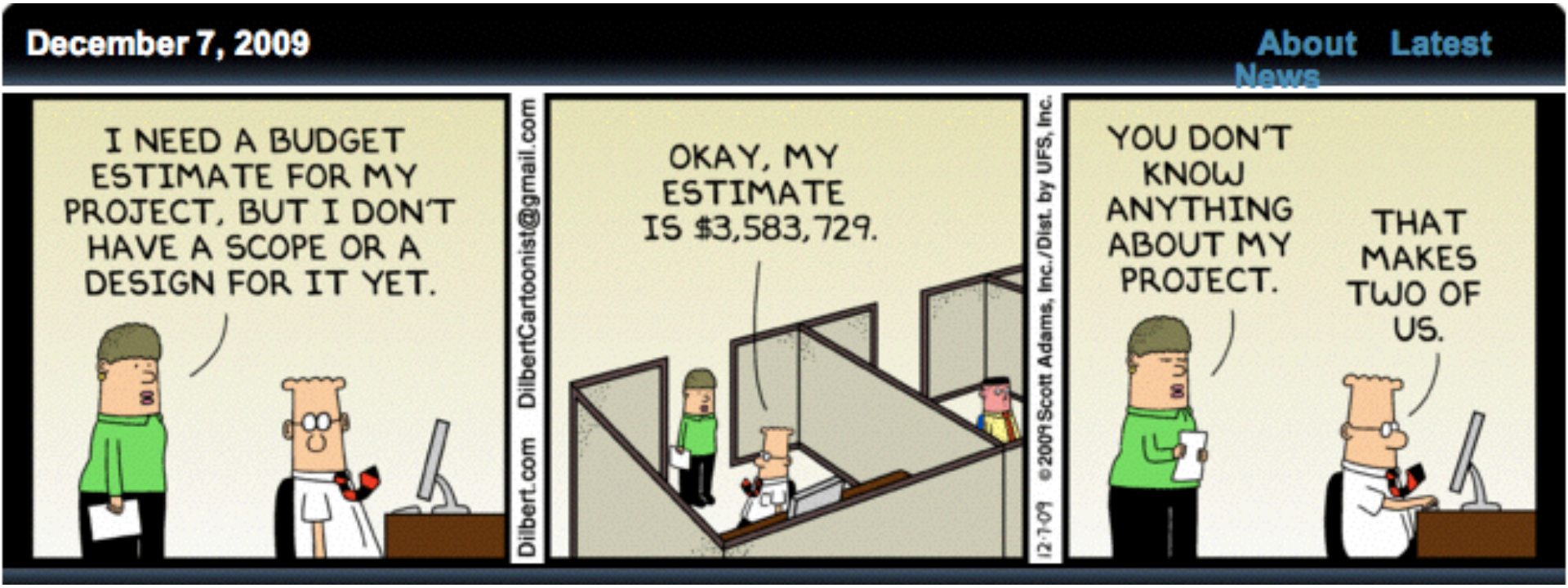
www.GILB.com

At Slideshare.com/tomgilb1 as of Mar 4 2013

Based On A Paper

- "Estimation: A Paradigm Shift Toward Dynamic Design-to Cost and Radical Management"
- Volume 13 Issue 2 of SQP journal - the March 2011 version.
  - Software Quality Professional, USA
  - The American Society for Quality (ASQ)
- http://www.gilb.com/tiki-download_file.php?fileId=460

The Obligatory Dilbert



The Risk Principles

- 1. DRIVERS: If you have not specified all critical performance and quality levels numerically – you cannot estimate project resources for those vague requirements.
- 2. EXPERIENCE: If you do not have experience data, about the resources needed for your technical solutions, then you cannot estimate the project resources.
- 3. ARCHITECTURE: If you implement your project solutions *all at once*, without learning their costs and interactions incrementally – you cannot expect to be able to understand the results of many interactions.
- 4. STAFF: If a complex and large professional project staff is an unknown set of people, or changes mid-project – you cannot expect to estimate the costs for so many human variables.
- 5. SENSITIVITY: If even the *slightest change* is made, after an 'accurate' estimation, to *any* of the requirements, designs or constraints – then the estimate might need to be changed *radically*. And – you probably will not have the information necessary to do it, nor the insight that you *need* to do it.

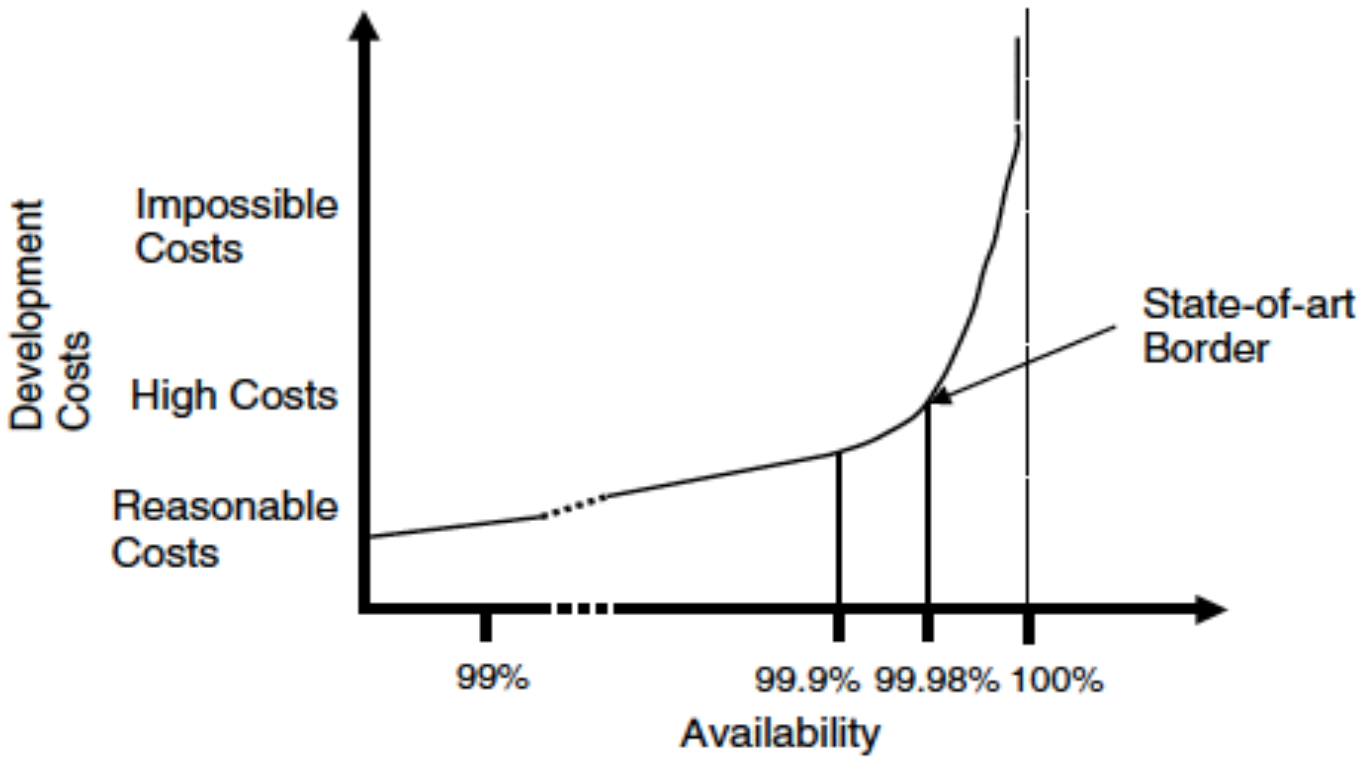
The Risk Principles (in Detail)

- The point being
  - that I want you to lose faith in convention notions of project estimation
  - The risk of being very wrong is very high!
  - The probability of being reasonably right is as big as you winning the Euro Lottery prize this week
  - In fact if you sometime experience being ‘right1, it is Not due to estimation
    - Just probably due to slamming on the brakes, when the resources are used up.

1. DRIVERS

- If you have not specified
  - all critical performance and quality levels numerically –
  - you *cannot* estimate project resources for those vague requirements.

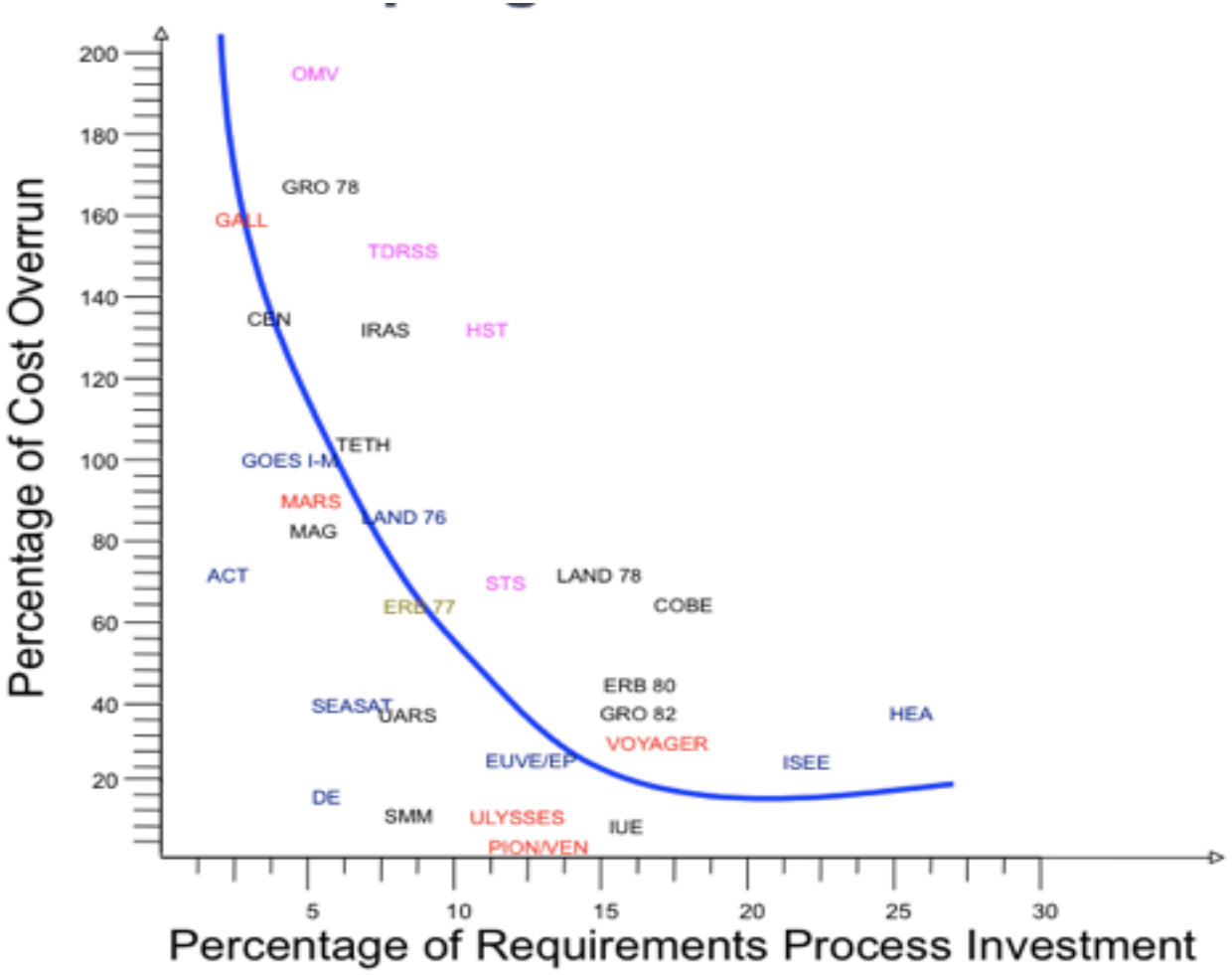
How much will ‘High Availability’ Cost?



2. EXPERIENCE

- If you do not have experience data,
  - about the resources needed for your technical solutions,
  - then you *cannot* estimate the project resources.

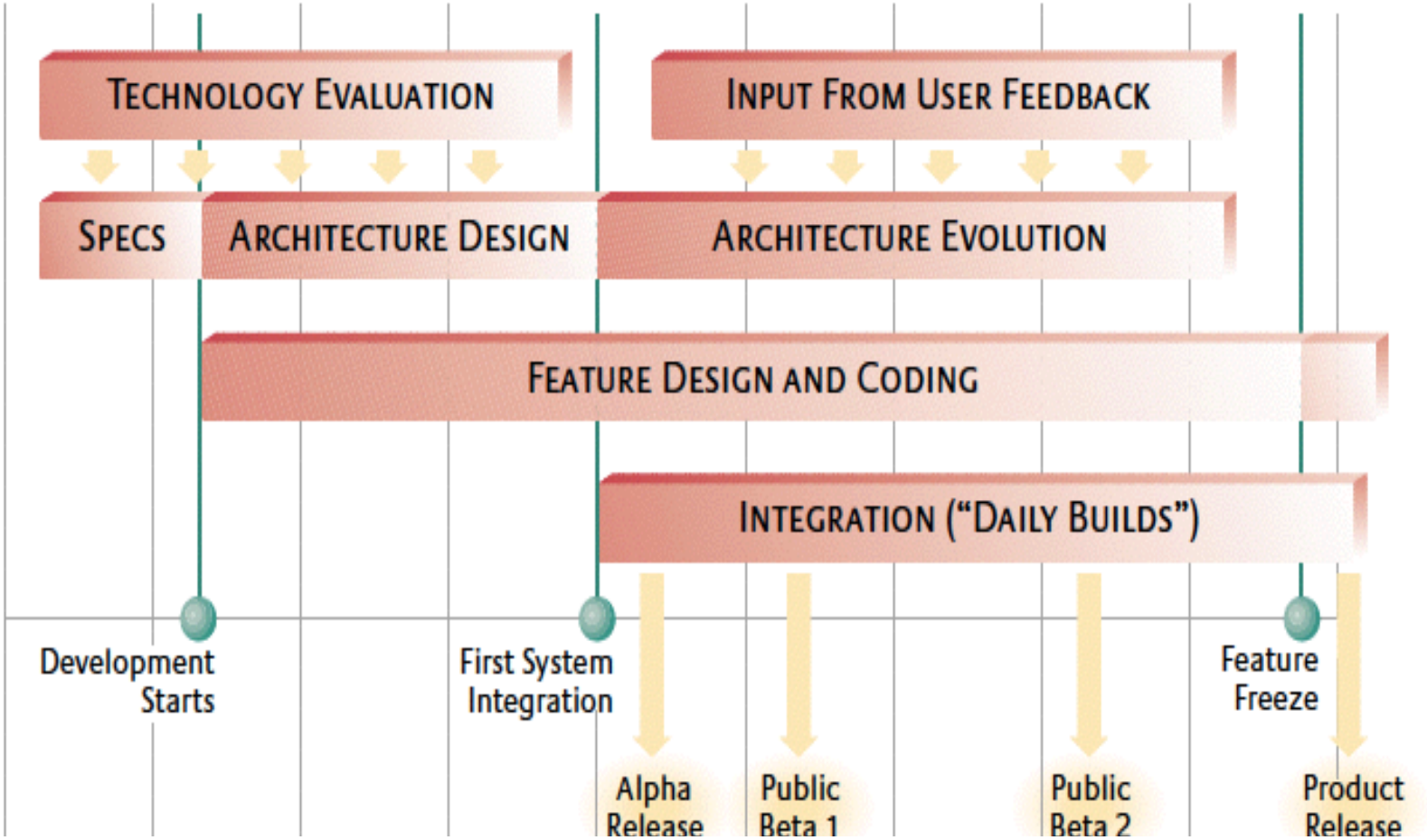
What is the cost difference if we use 5% for requirements, rather than 25%, if we are NASA?



3. ARCHITECTURE

- If you implement your project solutions *all at once*,
  - without learning their costs and interactions incrementally –
  - you cannot expect to be able to understand the results of many interactions.

Big Bang Fails: you don't know *exactly* why!



Small Delivery Steps
Give Better Control:
Cause and effect of failure is clearer

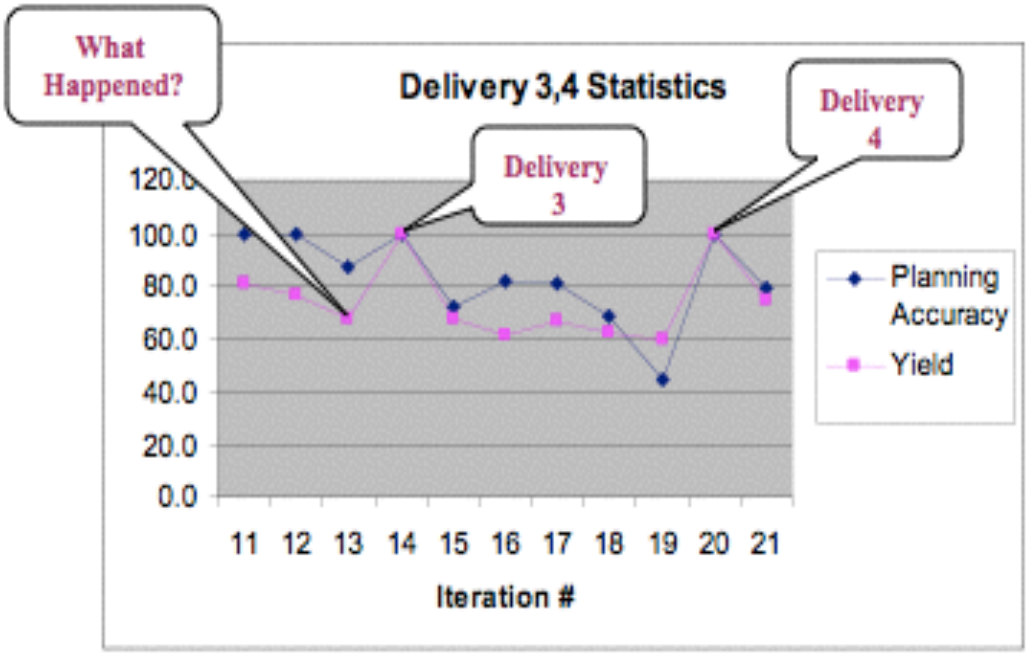
	Design Idea: Step 9 - Recoding			
Requirements	**Estimated Scale Impact**	**Estimated % Impact**	**Actual Scale Impact**	**Actual % Impact**
Objectives				
Usability.Productivity 65 <-> 25 minutes Past: 65 minutes. Tolerable: 35 minutes. Goal: 25 minutes.	**65 – 20 = 45 minutes**	**50%**	**65 - 38 = 27 minutes**	**95%**
Resources				
Development Cost 0 <-> 110 days	**4 days**	**3.64%**	**4 days**	**3.64%**

4. People

- If a complex and large professional project staff is
  - an unknown set of people,
  - or changes mid-project –
  - you cannot expect to estimate the costs for so many human variables.

Real Case:

Iterative measures, detected bad staff change (Honeywell, Berntsen)



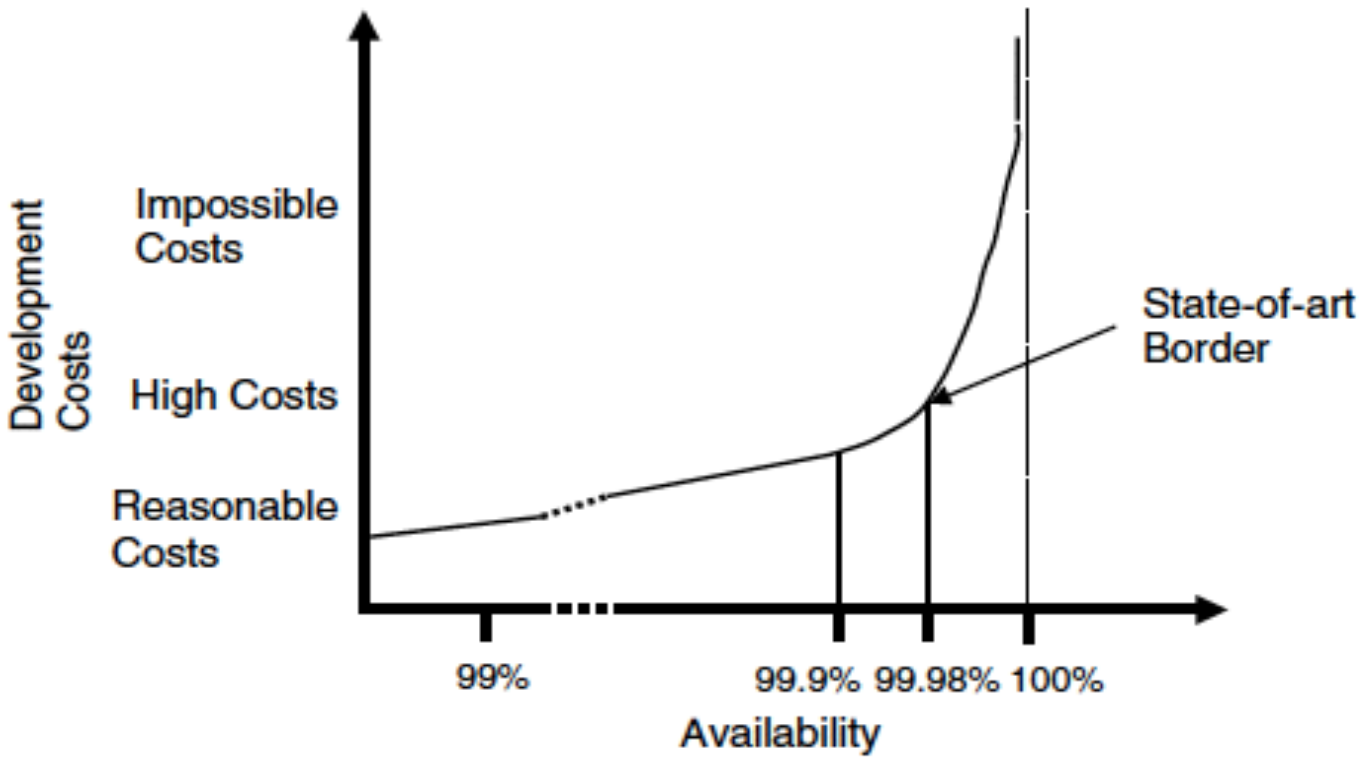
Measures

- Planning Accuracy - % of planned work that was completed.
- Build Yield - % of completed work that passed verification testing.

5. SENSITIVITY: to small changes in goals

- If even the *slightest change* is made,
  - after an 'accurate' estimation,
  - to *any* of the requirements, designs or constraints ,
  - then the estimate might need to be changed *radically*.
  - And – you probably will not have the information necessary to do it,
    - nor the insight that you *need* to do it.

99.98 – 99.90 = 00.08
80% to infinite costs

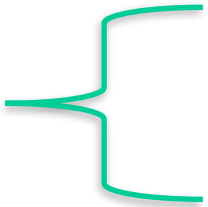


Real! : Primary Objectives for a £100 mill. Project

- Central to the Corporation's business strategy is to be the world's premier integrated <domain> service provider
- Will provide a much more efficient user experience
- Dramatically scale back the time frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to generate the desired products
- Make the system much easier to understand and use than has been the case for the previous system
- A primary goal is to provide a much more productive systems development environment than was previously the case
- Will provide a richer set of functionality for supporting next-generation logging tools and applications
- Robustness is an essential system requirement
- Major improvements in data quality over current practices.

Why COCOMO Estimation Method is doomed to fail

- Availability
  - Very High
    - 99.90%
    - 99.98%
  - High
  - Medium
  - Low



Why COCOMO Estimation Method is doomed to fail

- Availability
  - Very High
    - 99.90%
    - 99.98%
  - High
  - Medium
  - Low

8 years x 2 to 3,000 people
(AT&T Case 5 ESS)

The Control Principles: the Good News

6. LEARN SMALL: Carry out projects in small increments of delivering requirements – so you can measure results and costs, against (short term) estimates.

7. LEARN ROOT: If incremental costs for a given requirement level (and its designs) deviate negatively from estimates – analyze the root cause, and change anything about the next increments that you believe might get you back on track.

8. PRIORITIZE CRITICAL: You will have to prioritize your most critical requirements and constraints: there is no guarantee you can achieve them all. Deliver 'high-value for resources-used' first.

9. RISK FAST: You should probably implement the design ideas with the highest value, with regard to cost and risk, early.

10. APPLY NOW: Learn early, learn often, learn well; and apply the learning to your current project.

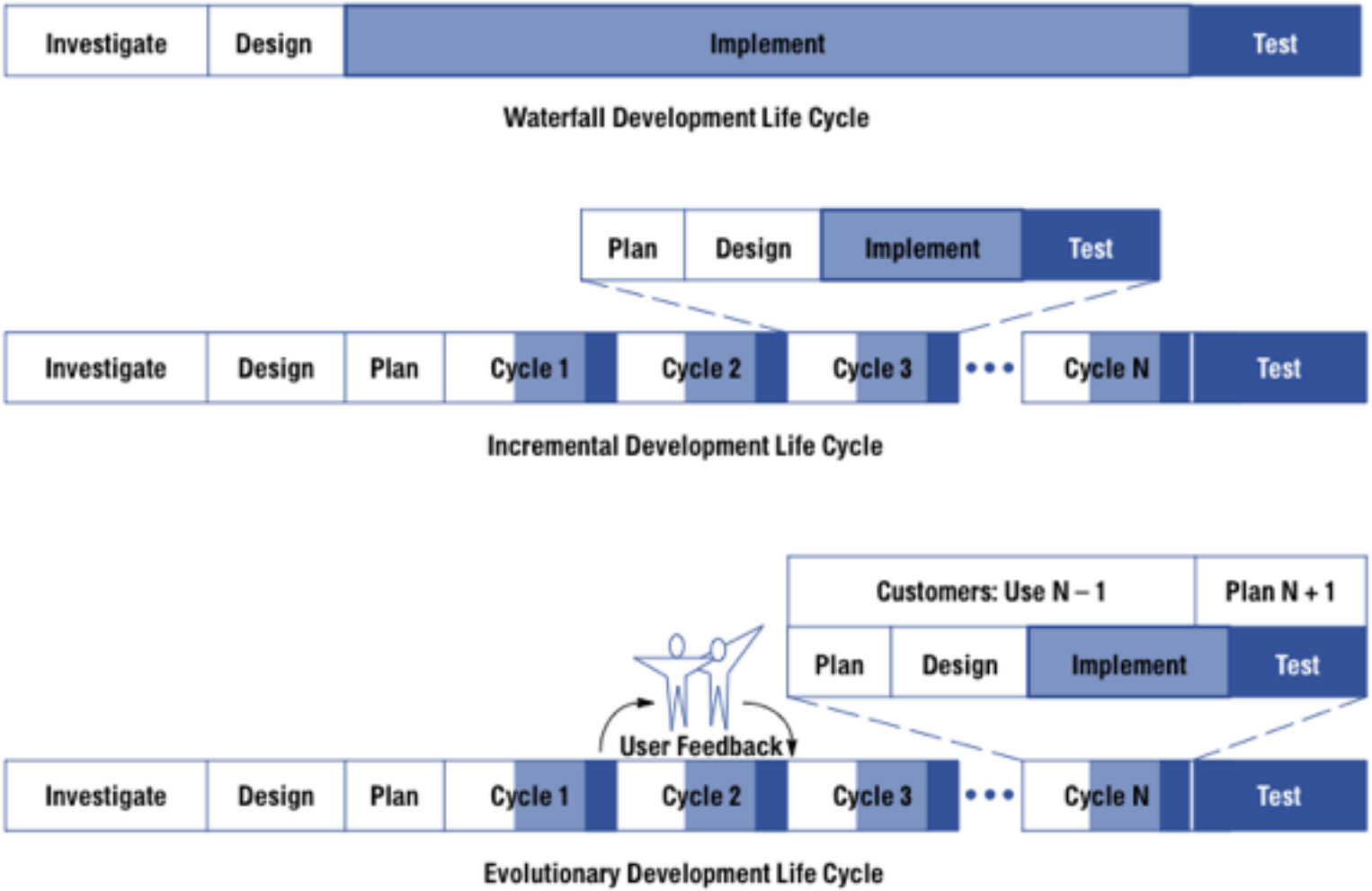
The Control Principles (shorter summary)

- The point here is that :
  - Given *any* arbitrary estimate of reasonable resources
  - You should be able to deliver so much prioritised value
  - that you will stay in business, forever (meaning)
    - People will want to feed you money!

6. LEARN SMALL

- Carry out projects in *small increments* of delivering requirements –
  - so you can measure *results* and *costs,*
  - against (short term) estimates.
  - And see cause and effect in useful detail

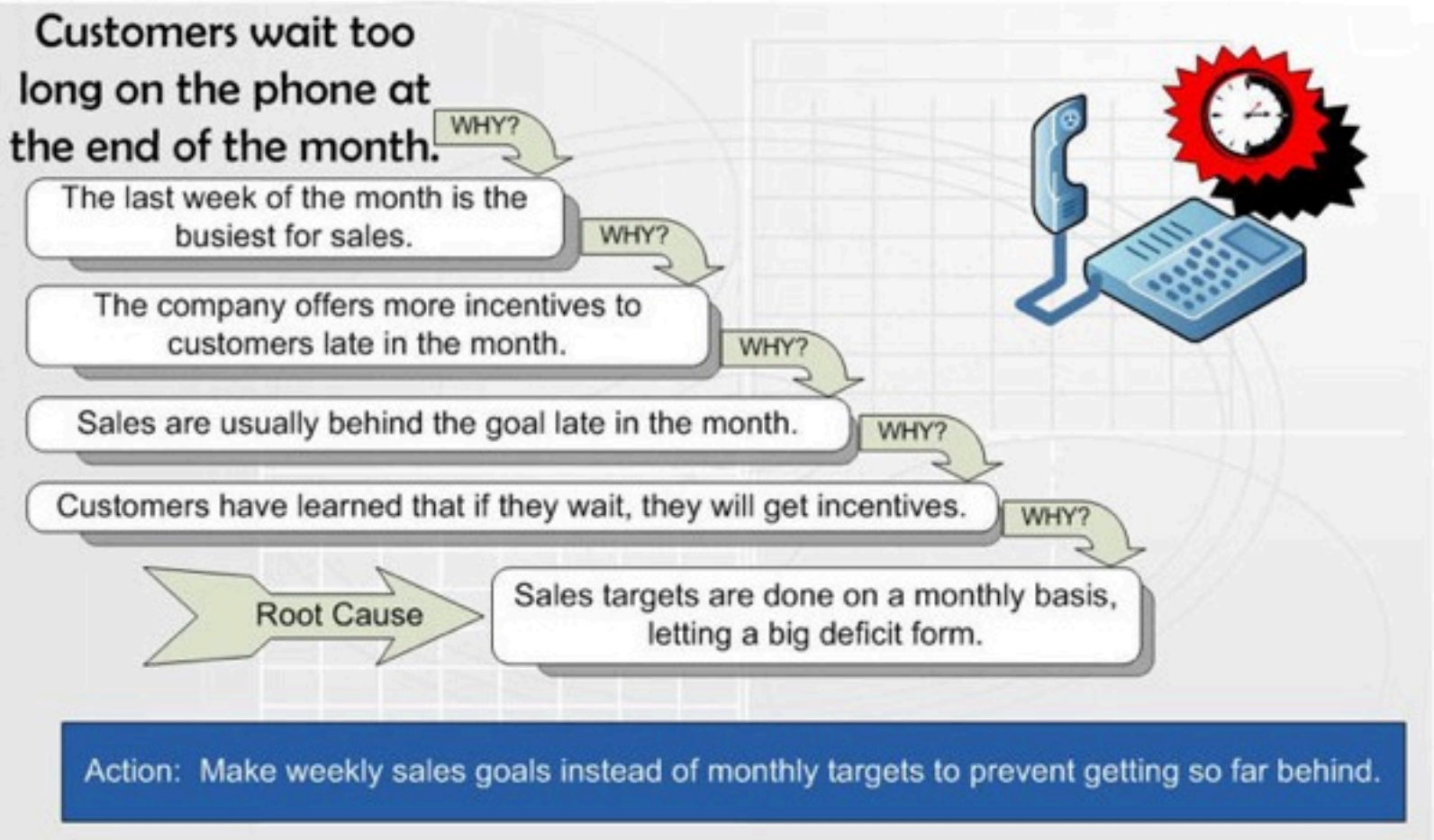
Breaking Result Deliveries into Small Chunks (Evo, HP, 1988 on)



7. Learn the Root Cause
(not unlike 'Lean Startup' !)

- If incremental costs for a given requirement level (and its designs) deviate negatively from estimates –
  - analyze the root cause, and
  - change anything
    - about the next increments
    - that you believe might get you back on track.

5 'Why's find roots

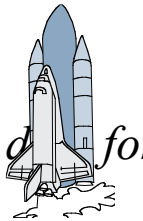


8. Prioritize the Critical Value Deliveries

- You will have to
  - *prioritize* your most critical requirements ('deliveries')
  - and respect your resource constraints:
    - there is no guarantee you can achieve them all.
- Deliver:
  - 'high-value for resources-used'
  - *first*.

In the Cleanroom Method, developed by IBM's Harlan Mills (1980) reported:

- *"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 deliverie [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 d 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."*

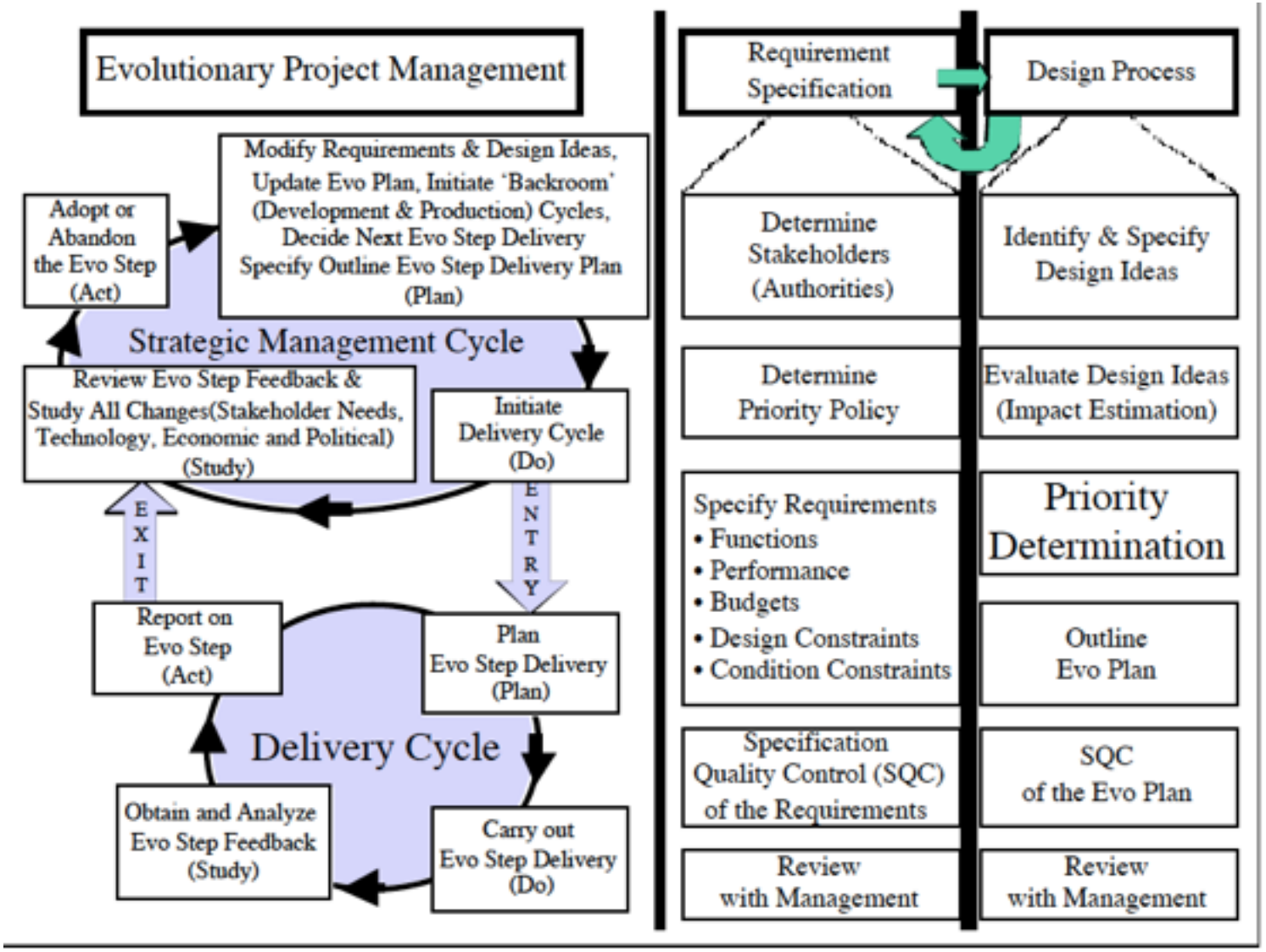


In the 'Cleanroom' Method, developed by IBM's Harlan Mills (1980) : Early 'Agile' in practice! (1970's)

- *"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 deliverie [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."***

Dynamic Prioritisation



9. Deliver Highest Value Early

- You should probably implement the design ideas (architecture components)
  - with the highest value,
  - with regard to cost and risk,
  - early.

Which Designs are 'Risky' ?

Design Ideas

On-line Support: Gist: Provide an optional alternative user interface, with the users' task information for defined task(s) embedded into it.

On-line Help: Gist: Integrate the users' task information for defined task(s) into the user interface as a 'Help' facility.

Picture Handbook: Gist: Produce a radically changed handbook that uses pictures and concrete examples to *instruct*, without the need for *any* other text.

Access Index: Gist: Make detailed *keyword indexes*, using *experience* from *at least ten* real users learning to carry out the defined task(s). What do *they* want to look things up under?

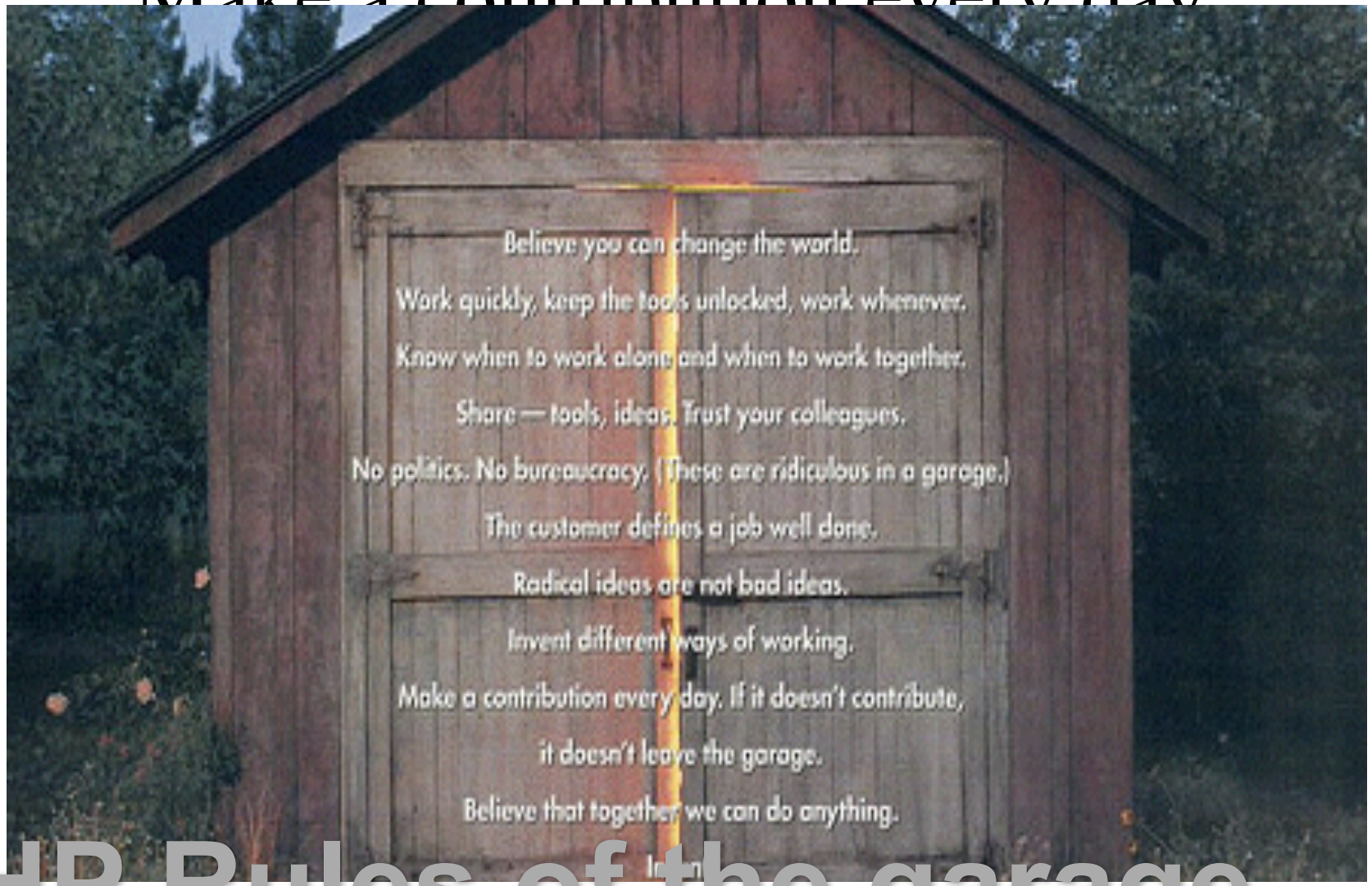
‘Impact Estimation’
Making ‘Risk’ Visible

	On-line Support	*On-line Help*	*Picture Handbook*	*On-line Help + Access Index*
Learning 60 minutes <-> 10 minutes				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3 min.	±5 min.	±10 min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%?	±10%
Evidence	Project Ajax: 7 minutes	Other Systems	Guess	Other Systems + Guess
Source	Ajax Report, p.6	World Report, p.17	John B	World Report, p.17 + John B
Credibility	0.7	0.8	0.2	0.6
Development Cost	120 K	25 K	10 K	26 K
Performance to Cost Ratio	110/120 = 0.92	100/25 = 4.0	60/10 = 6.0	104/26 = 4.0
Credibility-adjusted Performance to Cost Ratio (to 1 decimal place)	0.92*0.7 = 0.6	4.0*0.8 = 3.2	6.0*0.2 = 1.2	4.0*0.6 = 2.4

10. APPLY NOW
(does this sound like 'Lean Startup' ?

- Learn early,
  - learn often,
    - learn well;
      - and apply the learning to your *current* project.

“Make a contribution every day”



HP Rules of the garage

HP Garage Rules

(does this sound like 'Lean Startup' ?

- Believe you can change the world.
- Work quickly, keep the tools unlocked, work whenever.
- Know when to work alone and when to work together.
- Share tools, ideas. Trust your colleagues.
- No Politics. No bureaucracy. (These are ridiculous in a garage).
- The customer defines a job well done.
- Radical ideas are not bad ideas.
- Invent different ways of working.
- Make a contribution every day.
- If it doesn't contribute, it doesn't leave the garage.
- Believe that together we can do anything.
- Invent.

Simplified
'Control Principles'

- 1. Do valuable stuff quickly
- 2. Measure values & costs
- 3. Adjust plans, if necessary
- Repeat 1-3 , until no net value

Advantages with Control Principles

- 1. You can*not* waste much time or money before you realize that you have false ideas
- 2. You *can* deliver value early, and keep people happy
- 3. You are forced to think about the *whole* system, including *people* (not just code)
- 4. So you are destined to see the true costs of delivering value – not just the code costs
- 5. You will learn a general method that you can apply for the rest of your career.

Disadvantages
Control Principles

- 1. You cannot hide your ignorance from yourself any longer
- 2. You might have to do something not taught at school, or not taught in textbooks
- 3. There will always be people who criticize anything different or new
- 4. You cannot continue to hide your lack of ability to produce results, inside a multi-year delayed project.

Estimation ?

- Estimate, and re-estimate In small increments
- Make the most of *value* delivery
  - What does value actually cost?
- If you cannot deliver incremental value, stop
- A large estimate, or budget, is NOT important
  - But delivering value for money is far more important

Tack Takk Talk

If you request by email,

Subject: 'Estimation Books/Papers'

Tom@Gilb.com

I'll send you 2 free books and some papers

PS 6-7 March 2013, Wed-Thurs This week

Tom will hold a 2 day Course on Requirements (Vinnande kravdesign) in Gothenburg in Scandinavian Language, arranged by

www.inceptive.se/tomgilb/

- www.Gilb.com

Free Digital Book on Quality Quantification

- REQUEST "BOOK" in subject from
  - TOM @ GILB .com
- Tom Gilb,
  - Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage
  - and I will also send links to related papers on requirements and estimation.

Free BCS Courses in London (or £40 for coffee for non BCS members)

- 29-30 September 2014 **Requirements Engineering**, BCS London Details Later
- 1-2 October **Architecture Engineering**, BCS London. Details Later
- 6-7 October, **Lean Quality Assurance**, BCS London. Details Later
- 13-14 October **Project Management**
- see http://www.gilb.com/CourseSchedule for upcoming courses, conferences in all countries

LAST SLIDE

SEE

WWW.Gilb.COM

FOR MORE DETAIL

"Competitive Engineering" at www.gilb.com

(or via memory stick here at conference from presenter):

Supporting Standards for Quality Quantification

These following slides contain supporting Standards in detail which I do not expect to have time to show in my lecture

A
Process for
Quality Quantification. (PROCESS.QQ)

ENTRY: (ENTRY.QQ)

- 1. Do not enter if company files or standards already have adequate quantification devices.
  - Use existing quantification SCALES and METERS preferably.

- 2. Enter only if your process input documents
  - (contracts, marketing plans, product plans, requirements specification for example)
  - are Quality Controlled,
  - and have *exited* at a known and acceptable standard of defect-freeness
    - *(default standard; less than 1Major defect/page estimated remaining).*

Procedure for the Quality Quantification Task (PROCEDURE.QQ)

NOTE: these following steps cannot be simply sequentially. They need to be repeated many times to evolve realistic quality quantifications.

1. Use applicable rules {RULES.GR, RULES.QR, RULES.QQ}

2. *Build a list* of all quality concerns from your process input documents. Include *implicit* quality requirements *derived from* design requirements. Include any recent practical experience such as from evolutionary steps (of this project, pilot experiences or prototypes.

3. *Detail* the specification to a useful level. Include any recent practical experience such as from evolutionary result delivery steps of this project.

4. Revise these specifications when some design engineering/planning work is done on their basis. Only through design work can you know about the available technology and its costs.

5. Perform Quality Control (Inspection method) calculating remaining Major defects per page for the exit control. Apply valid rules {RULES.GR, RULES.QR, RULES.QQ}

6. Get experience using these specifications and revise specifications to be more realistic.

7. Repeat this process until you are satisfied with the result.

8. Cumulate your improved idea experiences and make available to others.

EXIT: (EXIT.QQ)

1. Calculated remaining Major defects/ page less than 1.

2. or exit condition “1.” above is <u>waived</u>

   with the intent of getting experience or opinions

   so as to <u>refine</u> it

   for official exit and more-serious use.

Specific Rules for Quality Quantification (QQ)

- 4.3. Rules: Quality Quantification. (RULES.QQ)

- The following rules would be
  - appropriate for a culture which was intent on raising quality specifications to a high level
  - and to systematically learn as a group,
  - in the long term,
  - from the experiences of themselves and others.
- The rules are guidance to the any writer or maintainer of quality specifications.
- Violations of these rules would be classed as *'defects'* in a quality control process on the document.

Rules for Quality Quantification:(RULES.QQ) 1of2

0:RULES: Rules for technical specification (RULES.GR) apply. This may be used in *addition* to the Quality Requirement Specification Rules (RULES.QR) or whenever serious emphasis on quality definition is required.

1:STANDARD: The Scale shall wherever possible be derived from a standard SCALE (in named files or referenced sources) and the standard *shall* be source referenced (←) in the specification.

2:SCALENOTE: If the Scale is not standard, a notification to Scale owner will inform about this case. "Note sent to <owner>" will be included as comment to confirm this act.

3:RICH: Where appropriate, a quality concept will be specified with the aid of *multiple* Scale definitions, each with their own unique tag, and appropriate set of defining parameters.

4: Meter : a practical and economic Meter or set of Meter s will be specified for each Scale. Preference will be given to previously defined Meter s in our Quantification archives.

5: Meter. NOTE: When 'essentially new' (no reference to previous case in generic archives) Meter specifications are made a Notification to Meter owner will notify about this case. "Note sent to <owner>" will be included as comment.

Continued next slide

Rules for Quality Quantification:(RULES.QQ) 2of2

6:BENCHMARK: Reasonable attempt to establish 'baselines' (Past, Record, Trend) will be made for our system's past, and for relevant competition.

7:TERMS: Future-priority requirements (Fail, Goal) will be made with regard to both *long* and *short* term.

8:DIFFERENTIATE: A distinction will be made, using qualifiers, between those system components which must have significantly higher quality levels than others, and components which do not require such levels. "The best can cost too much".

9:SOURCE: Emphasis will be placed on giving the exact and detailed source (even if a personal guess) of all numeric specifications, and of any other specification which is derived from a process input document (like a Meter which is contractually defined).

10:UNCERTAINTY) Whenever numbers are uncertain, we will have rich annotation about the degree (plus/minus) and reason (a comment like "because contract & supplier not determined yet"). The reader shall *not* be left to guess or remember what is known, or could be known, with reasonable inquiry by the author.

Generic Rules for Technical Specification (including Quality Quantification)
GR

- Here are some formal **rules** which could serve as a **standard** for how to communicate such ideas.
- We call this standard '**Generic**' because it applies to many types of **specification**.
- 'Rules' are a 'best practice' procedure for writing a document. Violation of rules constitutes a formal '**defect**' in that document.
- Rules are the local law of practice, and violation of them is an 'illegal' act.

GENERIC RULES FOR TECHNICAL AND MANAGEMENT DOCUMENTATION

Tag: RULES.**GR**

- **1:CLEAR Statements should be clear and unambiguous to their intended reader.**
  2:SIMPLE: Statements should be written in their most elementary form.
  3:TAG. Statements shall have a unique identification tag.
  4:SOURCE: Statements shall contain information about their detailed source, AUTHORITY and REASON/Rationale.
  5:GIST: Complex statements should be summarized by a GIST statement.
  6:QUALIFY: When any statement depends on a specific time, place or event being in force then this shall be specified by means of the [qualifier square brackets].
  7:FUZZY: When any element of a statement is unclear then it shall be marked, for later clarification, by the <fuzzy angle brackets>.
  8: COMMENT: any text which is secondary to a specification, and where no defect could result in a costly problem later, shall be written in *italic text statements, or/ and headed by suitable warning (NOTE, RATIONALE, COMMENT) or moved to footnotes*. Non-commentary specification shall be in plain text *Italic* can be used for emphasis of single terms in non-commentary statements. Readers shall be able to *visually* distinguish critical from not critical specification.
  9: UNIQUE: requirements and design specifications shall be made one single time only. Then they shall be re-used by cross reference to their identity tag. Duplication is strongly discouraged.

In addition to the general rules,
we can specify some special rules
for the specific types of statement
we are dealing with.

For example SR (below), QQ (above), QR (above).

REQUIREMENTS SPECIFICATION RULES. SPECIFIC RULES.**SR**

- **0:GR-BASE: The generic rules (RULES.GR) are assumed to be at the base of these rules.**
  1:TESTABLE: The requirement must be specified so that it is possible to define an unambiguous test to prove that it is later implemented.
  2:METER: Any test of SCALE level, or proposed tests, may be specified after the parameter METER.
  3:SCALE: Any requirement which is capable of numeric specification shall define a numeric scale fully and unambiguously, or reference such a definition.
  4:MEET:The numeric level needed to *meet requirements fully* shall be specified in terms of one or more [qualifier defined] target level {PLAN, MUST, WISH} goals; mainly the PLAN level here.
  5:FAIL: The minimum numeric levels to *avoid system, political, or economic failure* shall be specified in terms of one or more [qualifier defined] 'MUST' level goals.
  6. QUALIFY. Rich use of [qualifiers] shall specify [when, where, special conditions].

Very last slide