How to succeed in your IT Project, when all others fail, more or less: the ‘secret' methods.

Tom Gilb
Hon. Fellow BCS

https://www.dropbox.com/s/qf3bfbc0wh81kbv/How%20to%20succeed%20in%20your%20IT%20Project%2C%20when%20all%20others%20fail%20more%20or%20less-%20the%C2%A0%E2%80%98secret%27%20methods.pptx?dl=0

Guest Lecture at London Metropolitan University
Friday 3rd Feb 2015
2PM

tom@Gilb.com
www.Gilb.com

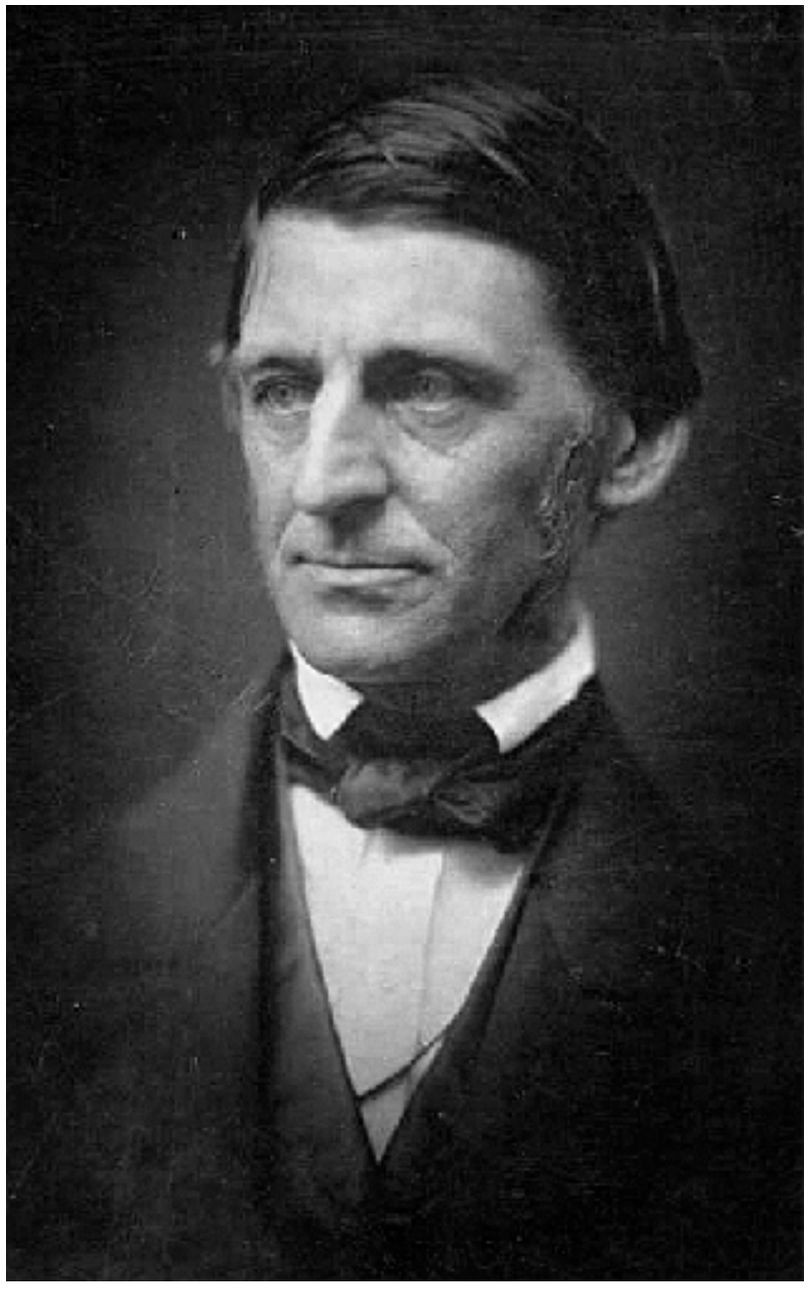
1: Being on time and under budget. 2: Delivering what stakeholders actually value, useful results; not just IT systems that do NOT deliver real value. The methods in a nutshell are; quantify all critical stakeholder values, as your main requirements. And, use Dynamic Design to Cost, like IBM did in Cleanroom projects, to be ‘agile' in correcting projects on a bad path.

For your FREE copy of
Competitive Engineering PDF
sign up to our email list at

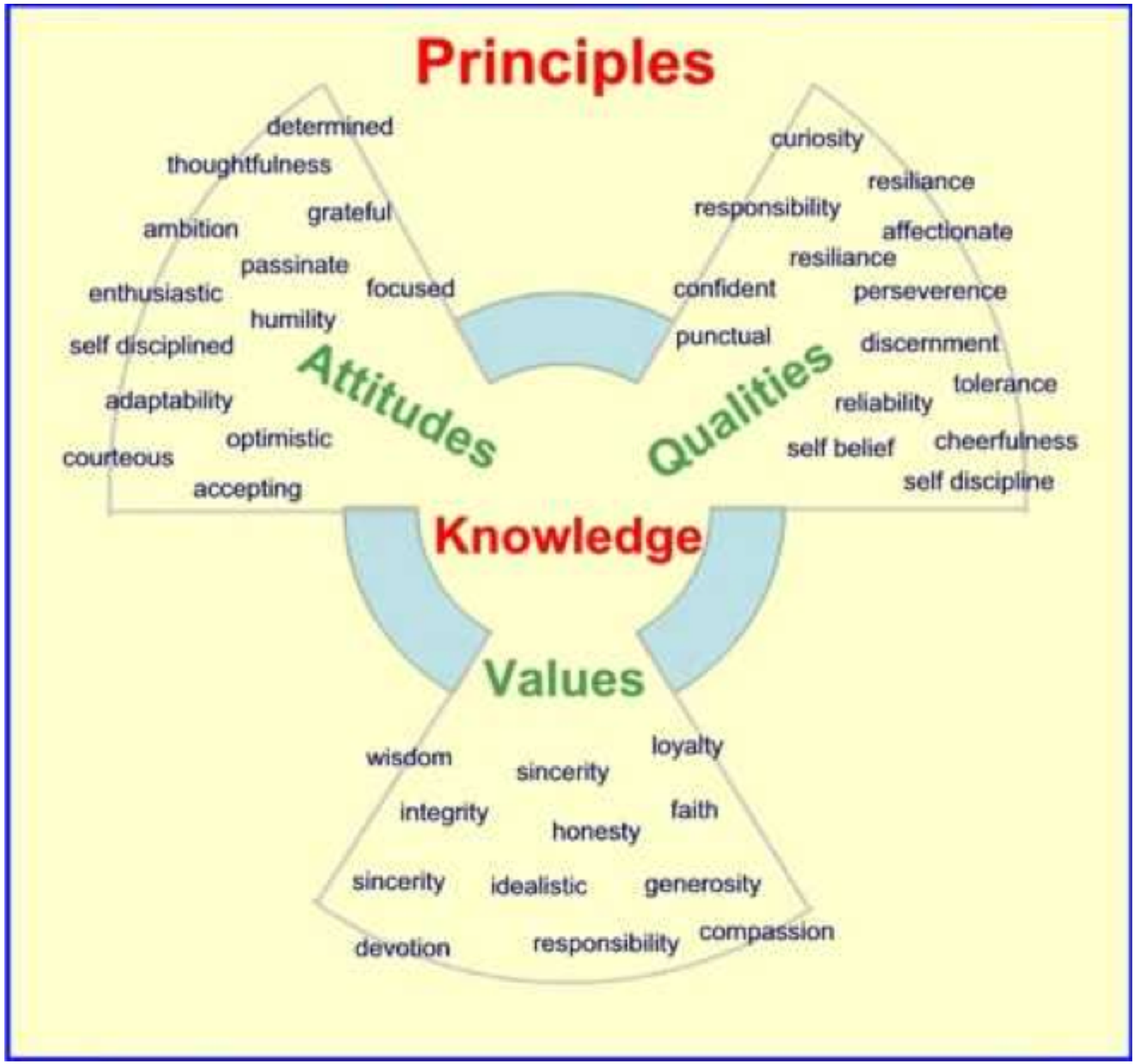
bit.ly/CompetitiveEngineering

The Principle that Principles beat methods

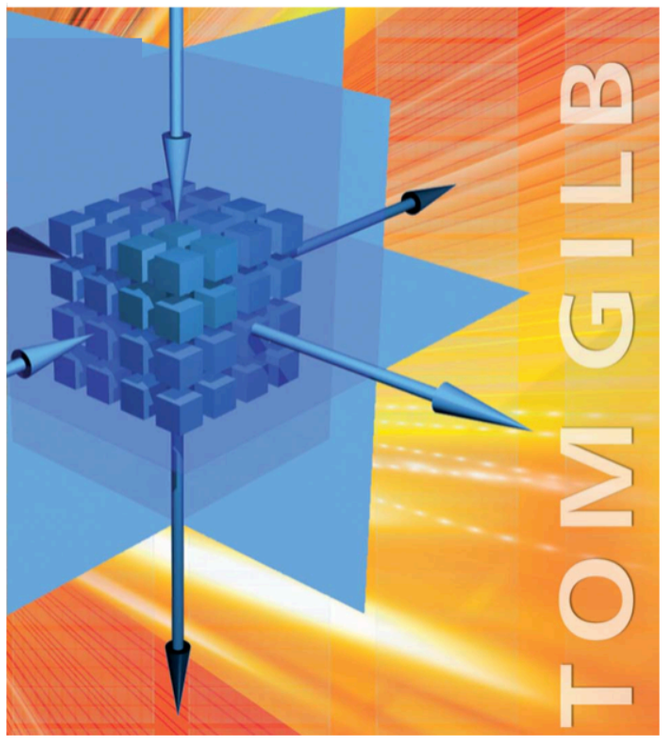
- “As to methods, there may be a million and then some, but principles are few.
- The man who grasps principles can successfully select his own methods”.
- - Ralph Waldo **Emerson,**
  – 1803-1882, USA



Role of Principles in Education



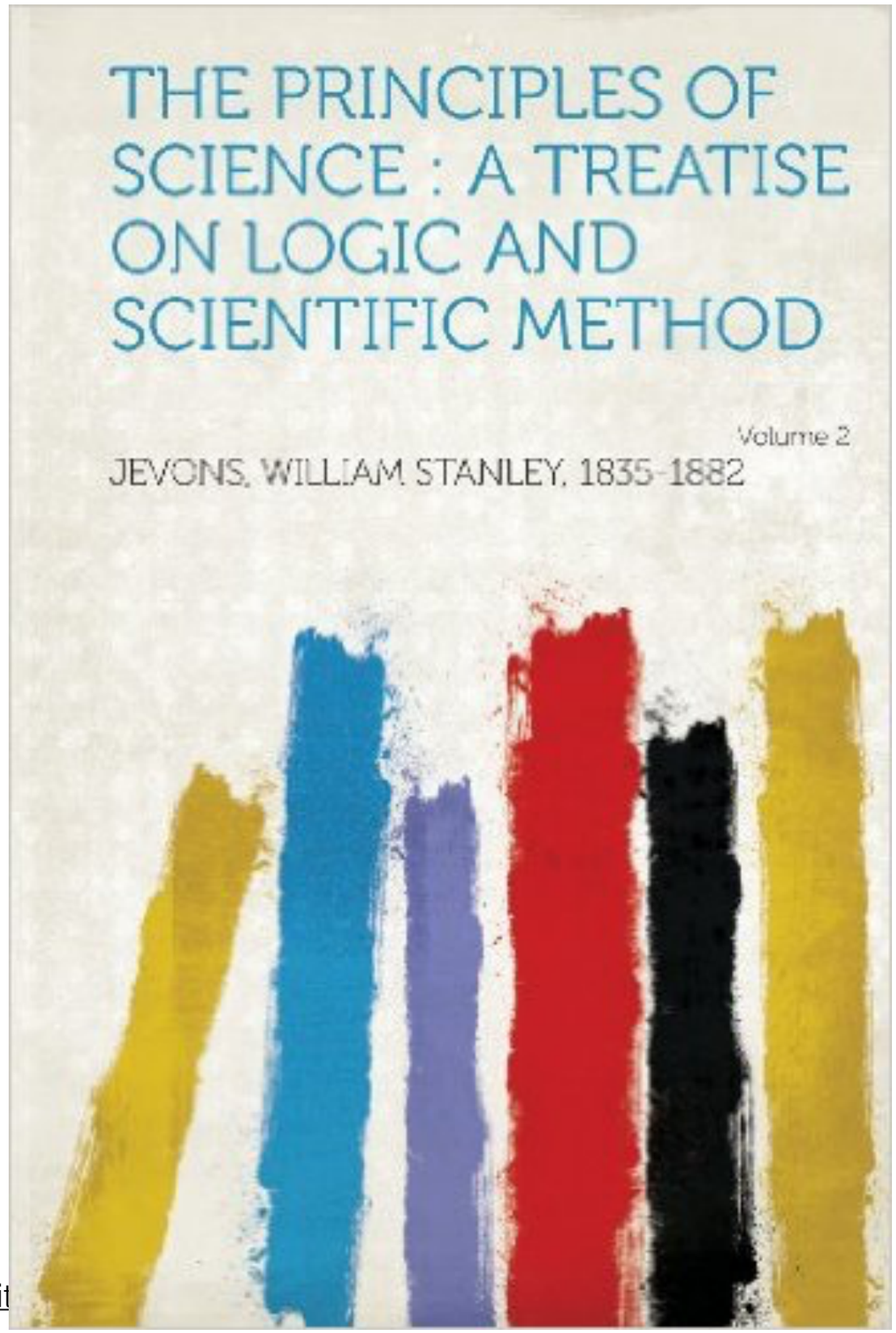
Value Planning



Practical Tools
for
Clearer Management Communication

leanpub.com/valueplanning

when I was 24 years old



Jevon's 1869 'Logic Piano' Machine

FINIS	+	d	D	c	C	b	B	a	A	COPULA	A	a	B	b	C	c	D	d	+	FULL STOP

Jevons' 1869 logic machine for doing Boolean algebra like truth tables.

http://www.eoht.info/page/Stanley+Jevons

(William) Stanley Jevons

bit.

Quotes

“There exists much prejudice against attempts to introduce the methods and language of mathematics into any branch of the moral sciences. Most persons appear to hold that the physical sciences form the proper sphere of mathematical method, and that the moral sciences demand some other method, I know not what.”
— Stanley Jevons (1871), *Theory of Political Economy* (pg. 3)

“We cannot weigh, or gauge, or test the feelings of the mind; there is no unit of labor, or suffering, or enjoyment.”
— Stanley Jevons (1871), *Theory of Political Economy* (pg. 9)

My Point

Some knowledge is ‘**eternal**’

Some knowledge is **more powerful** than other knowledge

BEING ON TIME

why do you think IT projects are often very late?

- **Audience Opinions ?**
- My Opinions ?

why do you think IT projects are often very late?

- Audience Opinions ?

- **My Opinions ?**

1. **lack of motivation to deliver on time**
2. **lack of clear definition of what will be delivered on time**
3. **lack of easy and continuous feedback, about time and progress; with consequent adjustments to make sure the essentials**

Summary of Top ‘8’ Project Objectives

Real Example of ***Lack*** of Quantification in large Engineering Company Project

*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.*

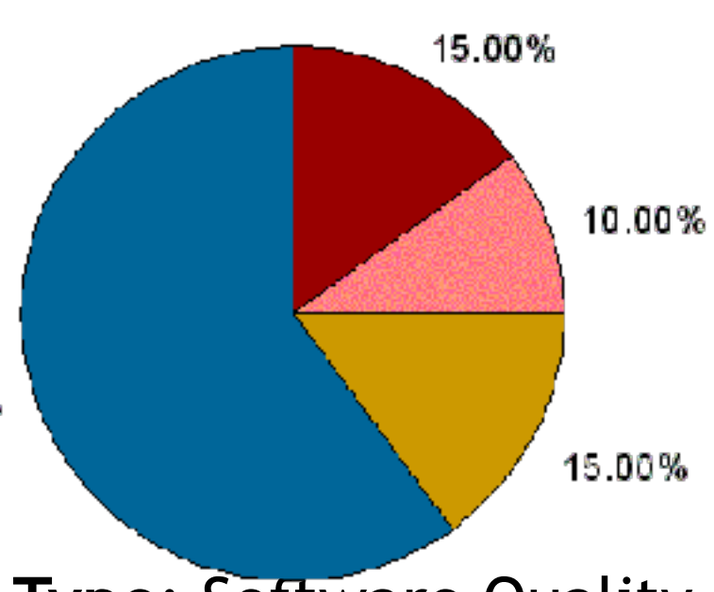
*7. **Robustness** is an essential system requirement*

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

This lack of clarity cost them over $100,000, 000. and 8 years delay

Rock Solid Robustness: *many splendored*

- **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:

Type: Software Quality Requirement. **Version**: 25 October 2007.
Part of: Rock Solid Robustness.
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

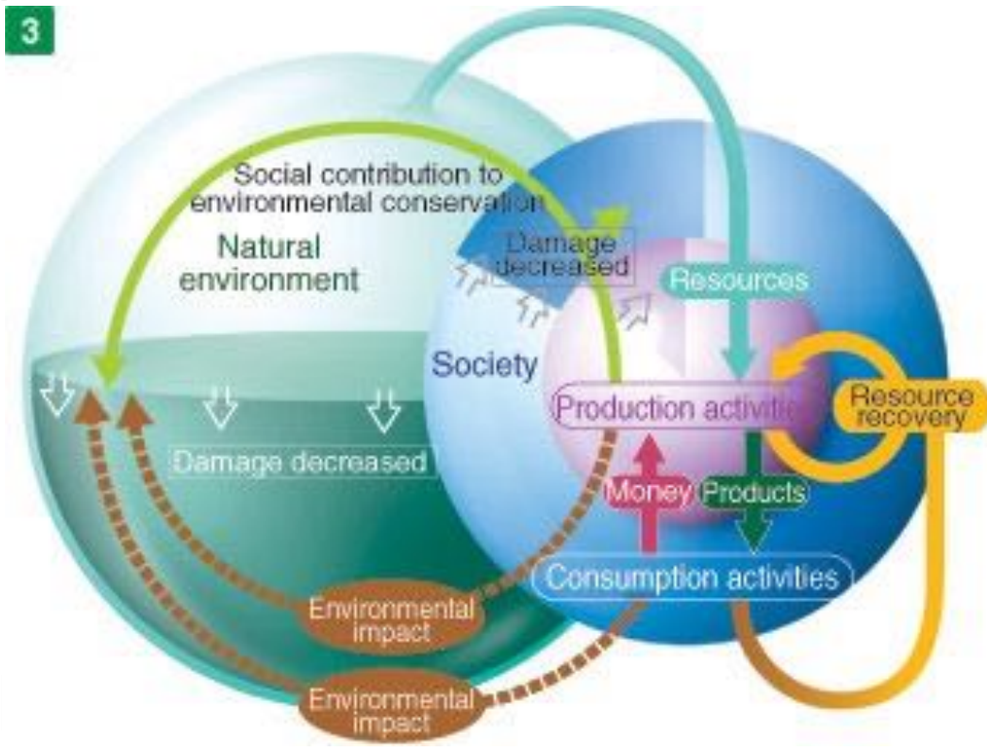
Goal [By 2008?, Activity = Data Acquisition, Intensity = Lowest level] : 300 days ??
Stretch: 600 days.

Restore Speed:

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. 3



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.

A Complex Requirement
“Robustness”

Robustness

Software Downtime | Restore Speed | **Testability** | Fault Prevention Capability | Fault Isolation Capability | Fault Analysis Capability | Hardware Debugging Capability

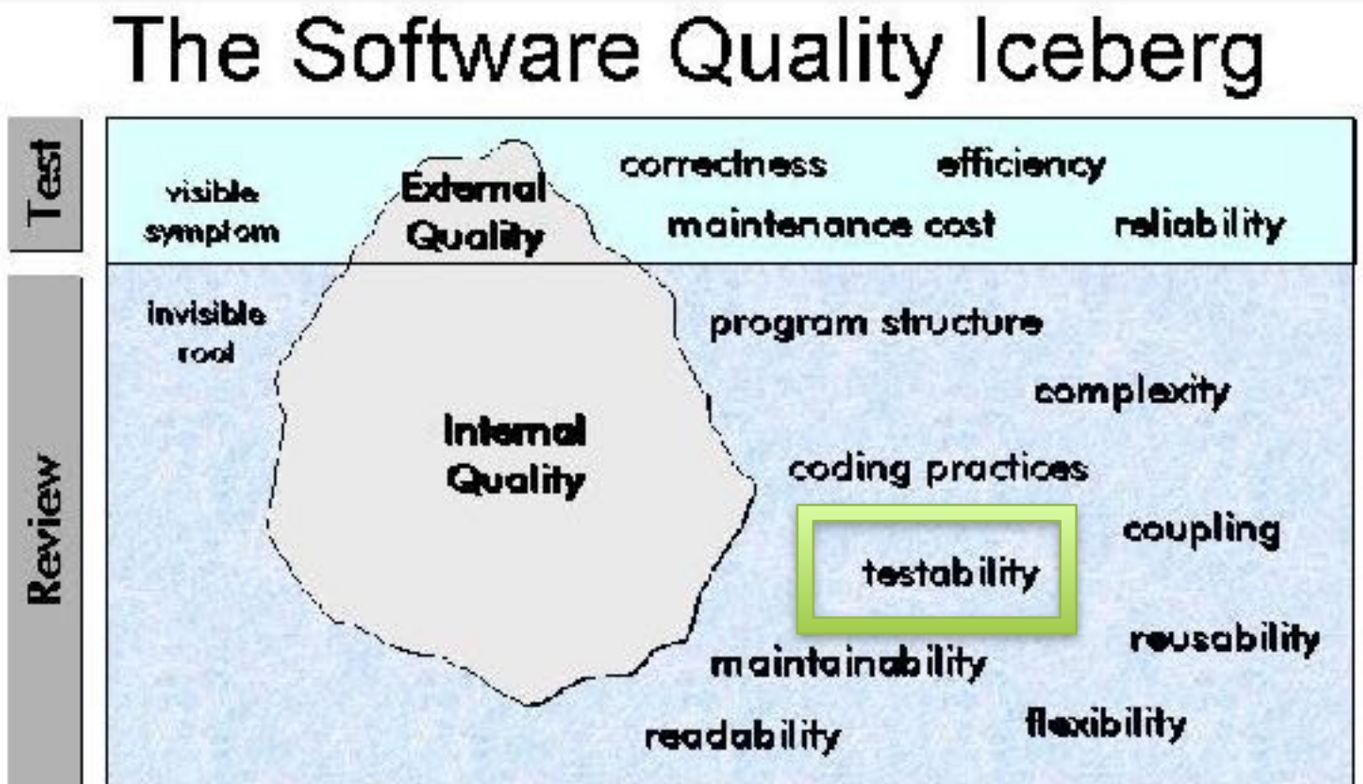
Testability:

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 the dump file, Application test harness console <-6.2.1 HFA*



BEING UNDER BUDGET

why do you think IT projects run over budgets?

- **Audience Opinions ?**
- My Opinions ?

why do you think IT projects run over budgets?

- Audience Opinions ?

- **My Opinions ?**

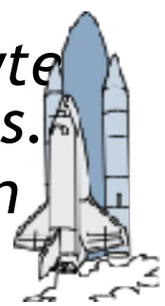
1. **‘somebody’ is earning a profit on the overrun (Greed)**
2. **the budget is not the projects personal money: it is taxpayer’s, the company (lack of responsibility)**
3. **we do not make ‘no cure no**

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

- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, f[illegible] ... [illegible]out*
- *T[illegible]cts – 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 s[illegible]rocessors distrib[illegible]veries [Ed. Note 2[illegible]dget*
- *A mor[illegible]'*
- *- Whe[illegible]-years of softwa[illegible]million byte of pro[illegible]en projects.*
- *- Ther[illegible]e at all in the pa[illegible]*

in 45 incremental deliveries

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



DELIVERING VALUE

why do you think IT projects fail to deliver impressive value?

- **Audience Opinions ?**
- My Opinions ?

why do you think IT projects fail to deliver impressive value?

- Audience Opinions ?

- **My Opinions ?**

1. **real stakeholder values are not explicitly used as primary project drivers**
2. **values are loose woolly bullshit (‘greater flexibility’)**
3. **Values are not quantified**

(65% by 100917)

4. **values are not contracted**

Incremental Value delivery at Philips

# Jobs	Week	[- 5%,+10%]	[-10%,+20%]	[-15%,+30%]	out of range
6	wk 8	1	5		
11	wk 9	3	1	7	
19	wk 10	6	3	7	3
25	wk 11	6	4	6	9
25	wk 12	17	3	5	
42	wk 13	31	3	2	6
55	wk 14	37	11	1	6
55	wk 15	39	9	1	6
55	wk 16	48	4	1	2
55	wk 17	50	4		1

Figure 5.6 Philips Value Delivery Cycles Results. The % is the accuracy of predicting a production run of electronic circuits, before that actual run. Green is good, red is bad.

Source Gilb: Value Planning, 5.6

Tracking Value Delivery 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

Source Value Planning
section 5.9
Confirmit

<- 50% of way to Goal level

<- Met goal

<- Twice as good as Goal level

<- No progress from Past level

<- 12.5 % over the Goal level

<- 91.8 average % to Goal in 9 of 12 weeks

EVO Value Tracking ‘Confirmit’ Version 8.5, in **Evo Step Impact *Measurement***
4 product areas were attacked in all: **25 Qualities** concurrently, one quarter of a year. Total development staff = 13

9

8

3

3

Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - C-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitiveness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
11,0	11,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	45,0	96,2	60	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReportExportFormats		
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.ExportReport (minutes)		
1,0	12,0	150,0	13	13	2
			Usability.ResponseTime.ViewReport (seconds)		
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
83,0	48,0	80,0	Backwards.Compatibility (%)		
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	69,0	100,0	69	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2304	500	100
			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	1
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0	2		1
			Runtime.ResourceUsage.CPU		
3,0	35,2	97,2	36	3	2
			Runtime.ResourceUsage.MemoryLeak		
0,0	800,0	100,0	800	0	0
			Runtime.Concurrency (number of users)		
300,0	110,0	146,7	150	500	1000
64,0			Development resources		
			0		64

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		88

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	61,0	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	186,0	97,4	170	50	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		40

QUANTIFYING STAKEHOLDER VALUE

what is the difference between stakeholder value’ and IT system Quality ?

- **Audience Opinions ?**
- My Opinions ?

what is the difference between stakeholder value' and IT system Quality ?,

example *Long term organisational flexibility*,
and *Software Portability*)

- Audience Opinions ?

- **My Opinions ?**

1. **Stakeholders care about their critical values deeply**
2. **IT qualities are merely a possible means to the Value 'ends'.**
3. **There are many ways to deliver the values, and many of them have nothing to do with IT**

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}.

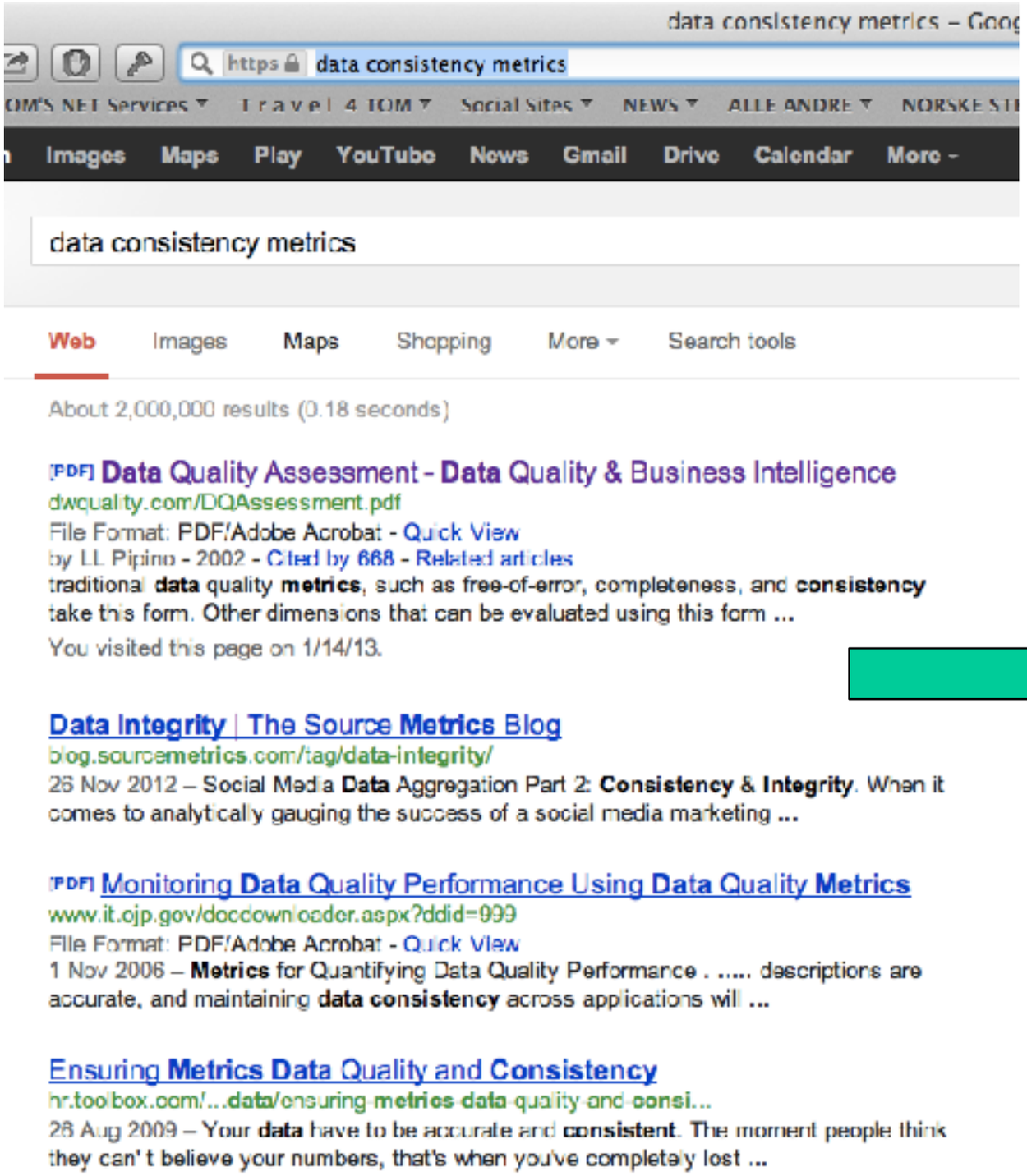
Problem Recognition:
Scale: C
system]
electroni
Adminis
Scale: C
assigned
Tool Co
Scale: C
acquire
and qua
Problem
Scale: C
toms an
Change
Scale: C
the nece
Note: Th
Quality
Scale: C
Modifica
Scale: C
correctio
Modifica
Unit T
Scale:
Integr
Scale:
Beta T
Scale:
releas
Syste
Scale:
Recovery:
Scale: Clock 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.

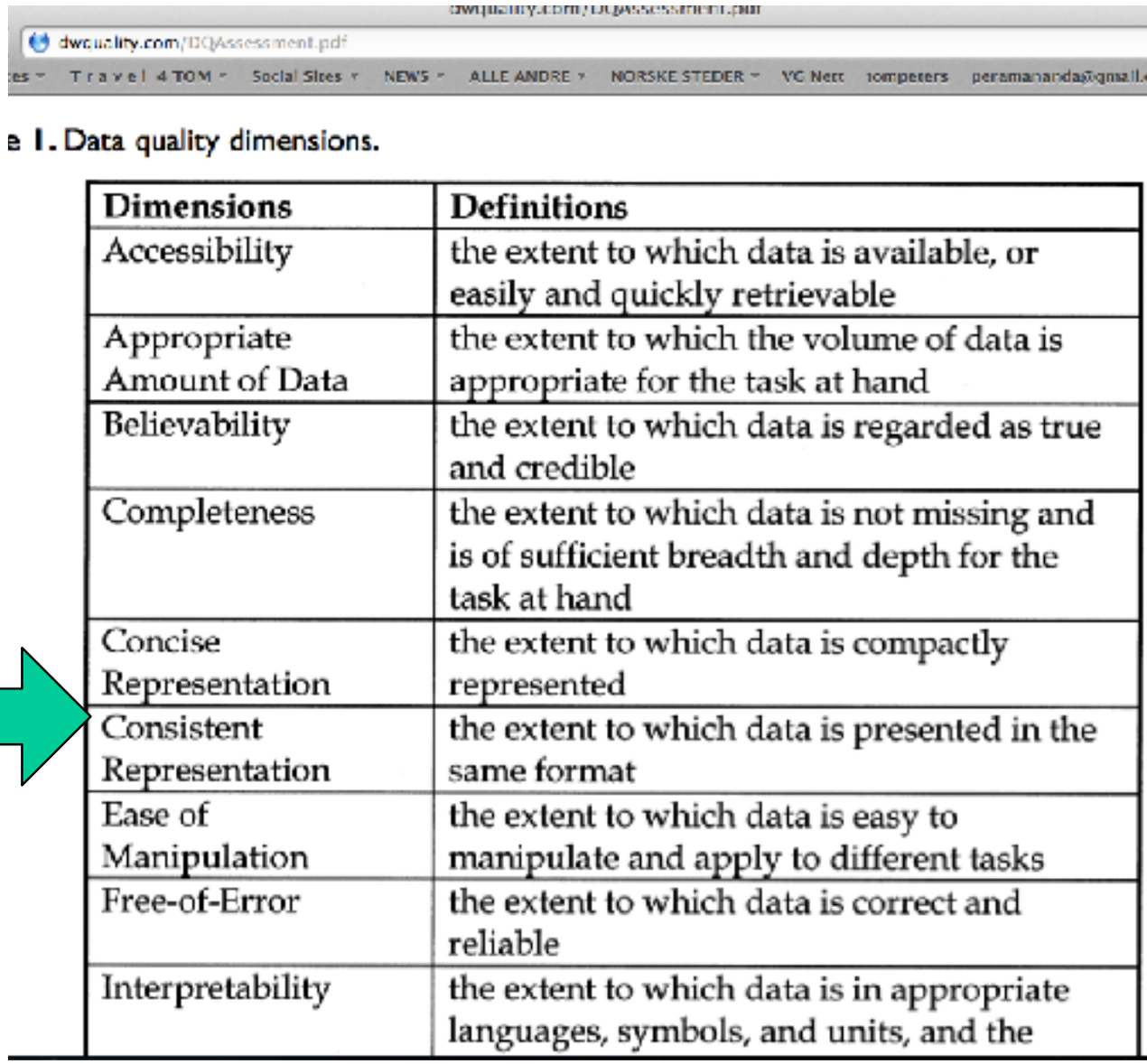
Source: The above is an extension of some basic ideas from Ireson, Editor, Reliability Handbook, *McGraw Hill, 1966* (Ireson 1966).

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





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

Exercise on Value Quantification

- what is your most critical stakeholder's most critical non-financial value?

- be sure it is their real value, not an iT product quality (like security, usability). nOt a solution to getting their real values (like an IT system)

- can you write down a quantified requirement for that value, that cannot be misunderstood?

CORRECTING BAD DESIGN, AGILE

can bad architecture or design be corrected in time to prevent IT project failure?

- **Audience Opinions ?**
- My Opinions ?

can bad architecture or design be corrected in time to prevent IT project failure?

- Audience Opinions ?

- **My Opinions ?**

1. **Yes, as for example Confirmit, and IBM Cleanroom have proven for years. Supported by similar recent Lean Startup methods**

2. **Yes. If we decompose all implementation into small short term incremental value delivery**

Mills on Design to Cost

- “To meet cost/schedule commitments based on imperfect estimation techniques, a software engineering manager must adopt a **manage-and-design-to-cost/schedule process.**
- That process requires a **continuous and relentless rectification of design objectives** with the cost/schedule needed to achieve those objectives.”
- in IBM Systems Journal, 4/80 p.420

Quinnan: IBM FSD Cleanroom: *Dynamic Design to Cost*

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

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

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

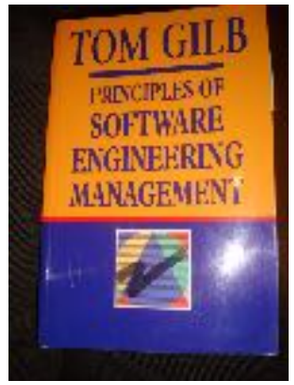
'Design is an iterative process in which each design level is a refinement of the previous level.' (p. 474)

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

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

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

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

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

-

_____He goes on t gn or by sacrificing 'planned capability.' When a s ment of each increment can proceed concurrent

'Design is an iterativ 74)

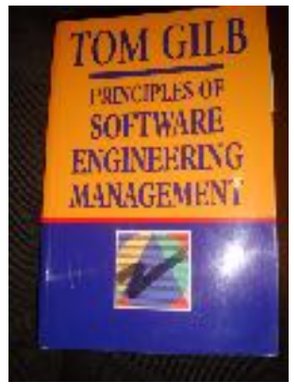
It is clear fro erate in seeking the appropriate balance s of increments, thus reducing the complexity of th ncrement develops, and as the true cost of the incr

'When the developm g increments is computed.' (p. 474)

Source: Robert E. Qu o. 4, 1980, pp. 466-77
This text is cut from

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

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

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

-

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

'Design is an iterati **74)**

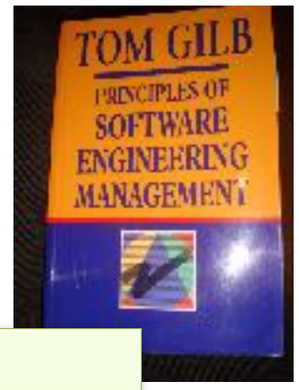
It is clear fro **erate in seeking the appropriate balance** **s of increments, thus reducing the complexity of th** **ncrement develops, and as the true cost of the incr**

'When the developm **g increments is computed.' (p. 474)**

Source: Robert E. Qu **o. 4, 1980, pp. 466-77**
This text is cut from

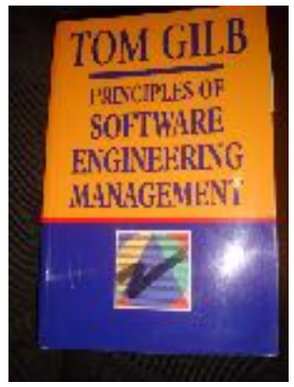
iteration process trying to meet cost targets by either *redesign* or by *sacrificing* 'planned capability'

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Design is an iterative process

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

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

ng
he

(p.

45

decomposing architecture

- think of a big strategy for IT
- or architecture idea for IT

- name 5 ways to decompose one of these ‘solution ideas’ so it can be delivered in weeklyincrements

decomposing architecture

- think of a big strategy for IT
- or architecture idea for IT
- name 5 ways to decompose one of these ‘solution ideas’ so it can be delivered in weeklyincrements

examples
do it one town at a time,
do it one employee, one department at a time
one major function at a time

For your FREE copy of
Competitive Engineering PDF
sign up to our email list at

bit.ly/CompetitiveEngineering