Planguage

A Software and Systems Engineering Language, for Evaluating Methods, and Managing Projects for Zero Failure, and Maximum 'Value Efficiency'

Tom Gilb

Keynote at **IWSM Mensura** Conference, http://www.iwsm-mensura.org/about

13:00 Session 26th October 2017, Ericsson Lindholmen, Gothenburg

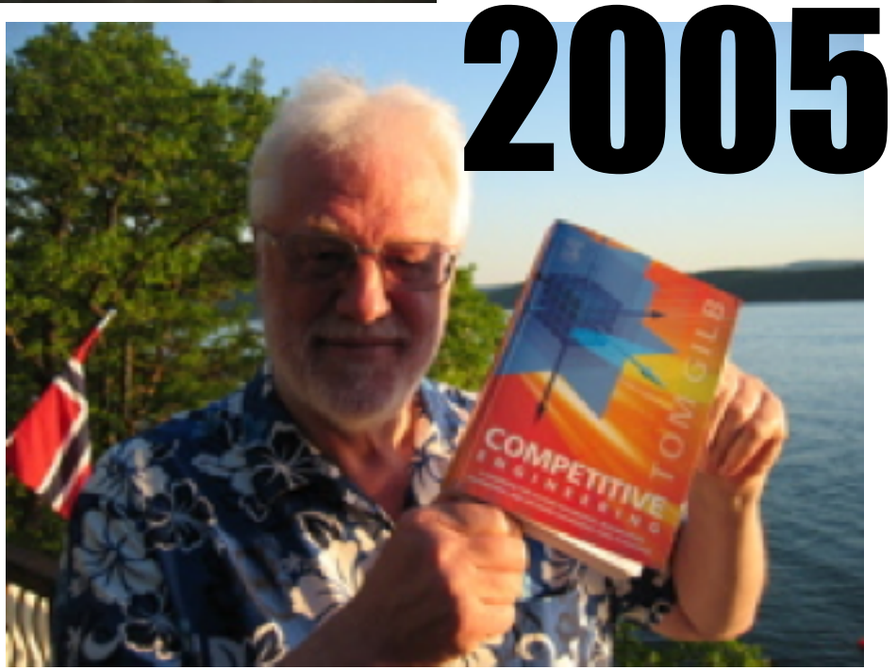
tom@Gilb.com, www.Gilb.com, @ImTomGilb

#IWSM Mensura 2017

‘SM’ Books 1976, 1977 and 2005

1977

USA



2005

Talk Outline

1. Quantification of Values and Qualities

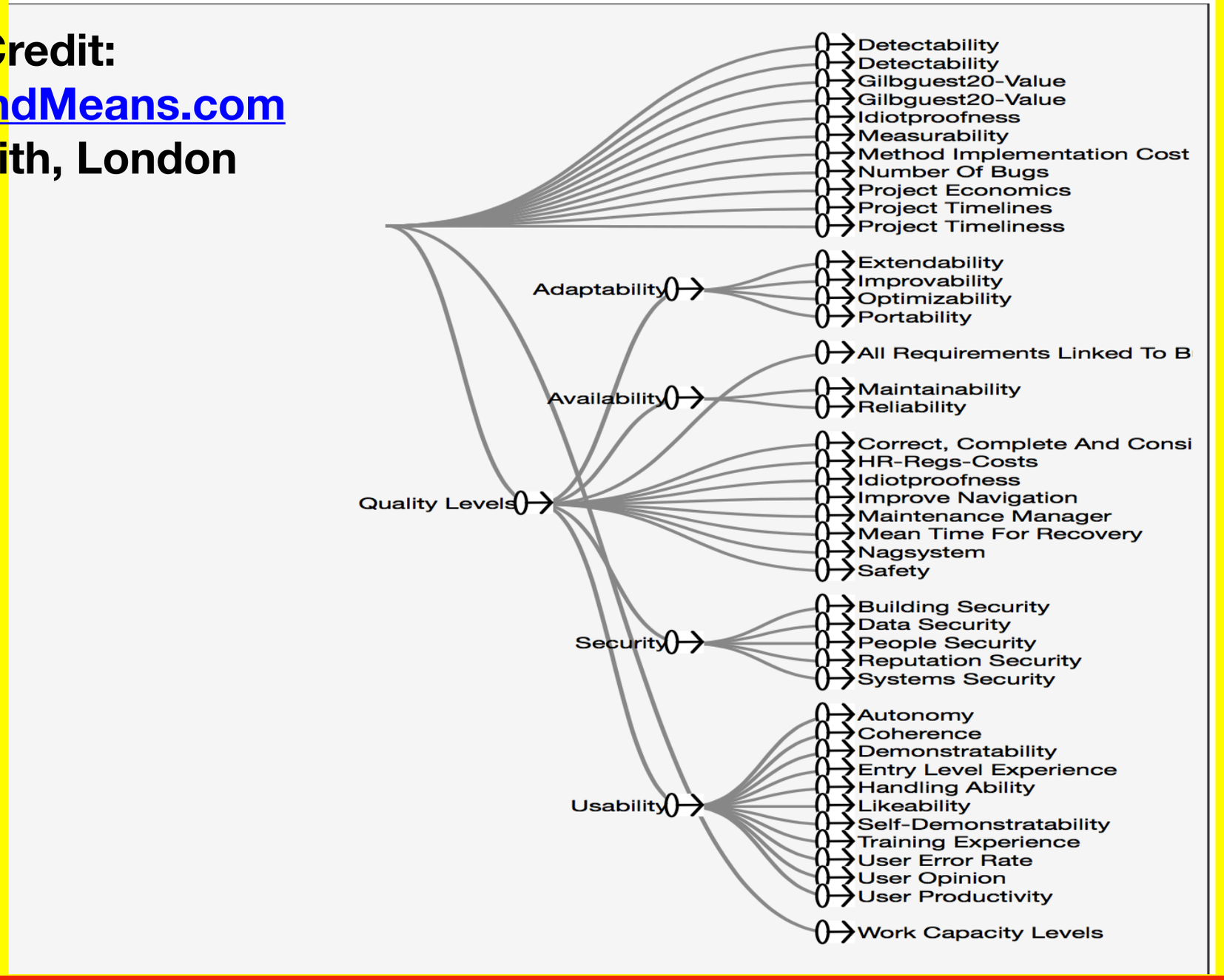
2. Estimation of multiple attributes of methods and strategies

3. Evo and Advanced Agile: Multiple Measures, and Dynamic Design to Cost Estimation

4. Measuring Development Specifications Quality:

Lean Quality Assurance

Tool Credit:
www.NeedsandMeans.com
Richard Smith, London



1. Quantification of Values and Qualities

The Principle Of 'Quality Quantification'
The Words of a 'Lord'

"All qualities can be expressed quantitatively,
***'qualitative'* does *not* mean unmeasurable". (Gilb)**

http://tinyurl.com/**GilbTedx**

"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.

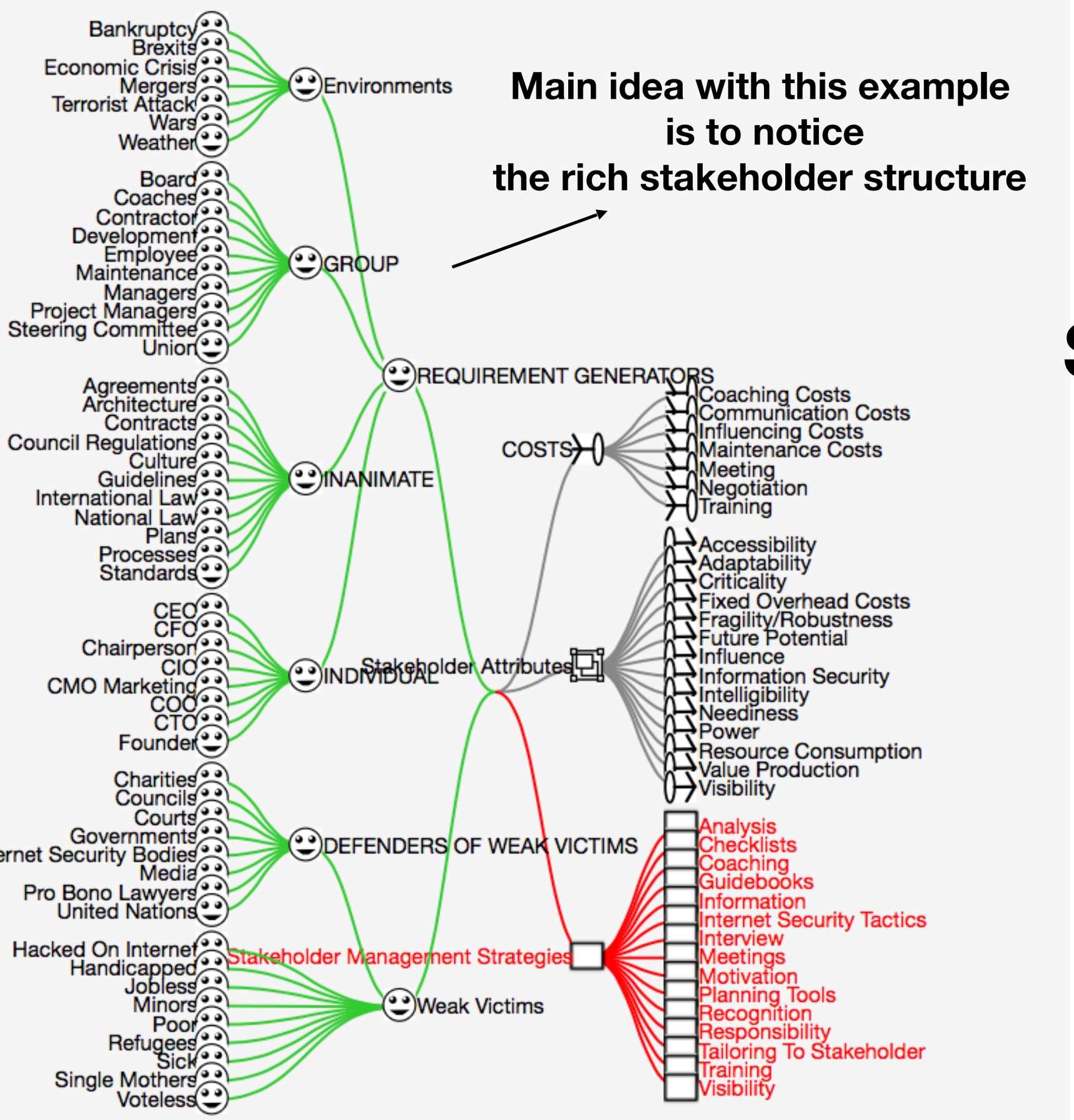
I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it;

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, *Lecture to the Institution of Civil Engineers, 3 May 1883* **From** http://zapatopi.net/kelvin/quotes.html

Born: 26 June 1824; Belfast, Ireland
Died 1907..

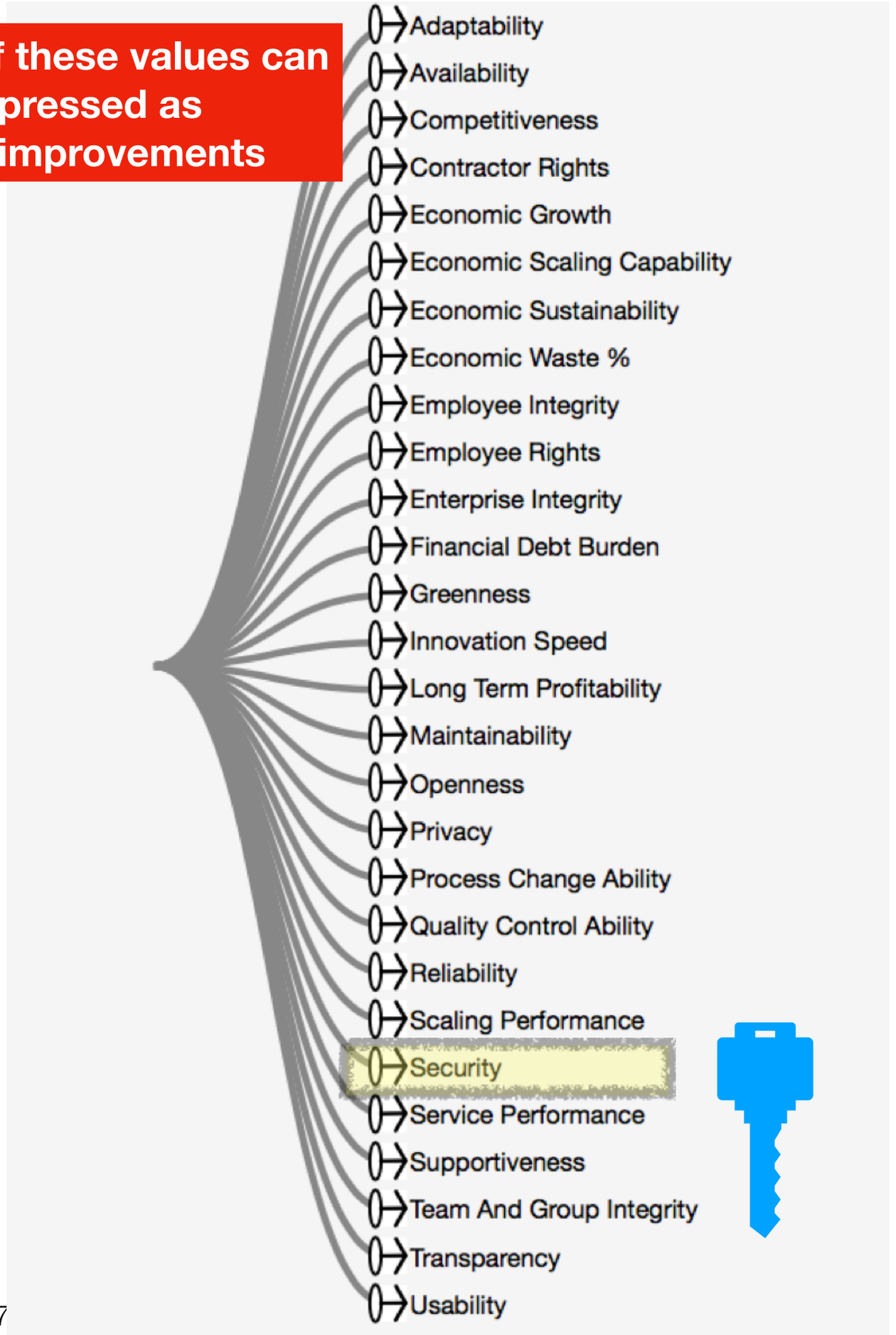


Main idea with this example is to notice the rich stakeholder structure

Stakeholders Needs and Means diagram

Every one of these values can be expressed as numeric improvements

Direct Quantification of all benefits, so they are unambiguous clear; and trackable in agile delivery steps.



Security Value Quantification with Stakeholders

All values and qualities can be expressed quantitatively

National Security

Business Value ***Label?***

(✎ by tomgilb - 2 months ago)

Is Part Of: Stakeholder Values Value

Ambition Level: to reduce terrorist attacks, and identify potential terrorist attacks, and regulate cyber information

Bullshit level

Scale: Number Negative [Effects] on [Stakeholders] from [Attack Types] under [Conditions] in [Places] per year for given [Area]

Stakeholders: Prime Minister, Casualties, Council Representatives, Police, Relatives Of Victims, Volunteers

Status: Level: **150** Number Bad Stuff [Effects = { Death }, Stakeholders = { <All> }, Attack Types = { Vehicle Attack,Knife Attack,Gun Attack }, Conditions = { Hig

Wish: Level: **10** Number Bad Stuff [Effects = { Death }, Stakeholders = { <All> }, Attack Types = { Vehicle Attack,Knife Attack,Gun Attack }, Conditions = { High A

Record: Level: **1** Number Bad Stuff [Effects = { Death }, Stakeholders = { <All> }, Attack Types = { Vehicle Attack,Knife Attack,Gun Attack }, Conditions = { High

This structure of requirements is in 'Planguage'. Which is specified in books 'Competitive Engineering' and 'Value Planning'

REQUIREMENT

MANY DIMENSIONS

Requirements		Incentivise	Tea Kiosk	Daily Danger Checks	Sum
Project Timeliness Status: **10** → Wish: **5** % % time overrun necessary to deliver ... [Project Cost Size = { Medium ($10k -...] 30th June 2017	=: Δ: Δ%: ?%:	**8** ± 0 -2 % **40** ± 0 % *32 % (x 0.8)* 40%	**5** ± 1 -5 % **100** ± 20 % *50 % (x 0.5)* 100%	**15** ± 8 5 % **-100** ± 160 % *-80 % (x 0.8)* -100%	ΣΔ%: **40** ± 180 %
Building Security Status: **50** → Wish: **10** % I... % of [Emergency Types] which in fact... [Emergency Types = { Earthquake }, 30th June 2018	=: Δ: Δ%: ?%:	**50** ± 0 0 % Injury **0** ± 0 % *0 % (x 0.0)* 0%	**50** ± 0 0 % Injury **0** ± **NaN** % *0 % (x 0.6)* 0%	**30** ± 10 -20 % Injury **50** ± 25 % *15 % (x 0.3)* 50%	ΣΔ%: **50** ± 25 %
User Productivity Status: **15** → Wish: **5** minutes number of minutes for a [user] to co... [user = { adult }, task = { dri...] 30th June 2017	=: Δ: Δ%: ?%:	**10** ± 0 -5 minutes **50** ± 0 % *0 % (x 0.0)* 50%	**8** ± 3 -7 minutes **70** ± 30 % *56 % (x 0.8)* 70%	**15** ± 0 0 minutes **0** ± 0 % *0 % (x 0.0)* 0%	ΣΔ%: **120** ± 30 %
Sum Of Values: Credibility - adjusted:	Σ%: Σ?%:	**90** ± 0 % *32 %*	**170** ± 50 % *106 %*	**-50** ± 185 % *-65 %*	
Method Implementation Cost Status: **0** → Budget: **3m** $ Total monetary cost in US Dollars fo... [Project Cost Size = { }] 30th June 2017	=: Δ: Δ%: ?%:	**500k** ± 0 500k $ **17** ± 0 % *34 % (x 0.0)* 17%	**2m** ± 0 2m $ **67** ± 0 % *134 % (x 0.0)* 67%	=:**1m** ± 0 Δ: 1m $ Δ%: **33** ± 0 % ?%: *66 % (x 0.0)* 33%	ΣΔ%: **117** ± 0 %
Sum Of Development Resources: Credibility - adjusted:	Σ%: Σ?%:	**17** ± 0 % *34 %*	**67** ± 0 % *134 %*	**33** ± 0 % *66 %*	
Value To Cost:		5.30	2.50	-1.50	

2. Estimation of multiple attributes of methods and strategies

Quantifying Design/Architecture/Strategic Planning
Moving towards an engineering discipline.

— Confucius, *Sayings of Confucius*

"True wisdom is knowing what you don't know"

— Confucius, *Sayings of Confucius*

What intellectual tools do you have that will help you to be more conscious of exactly what you do NOT know enough about?

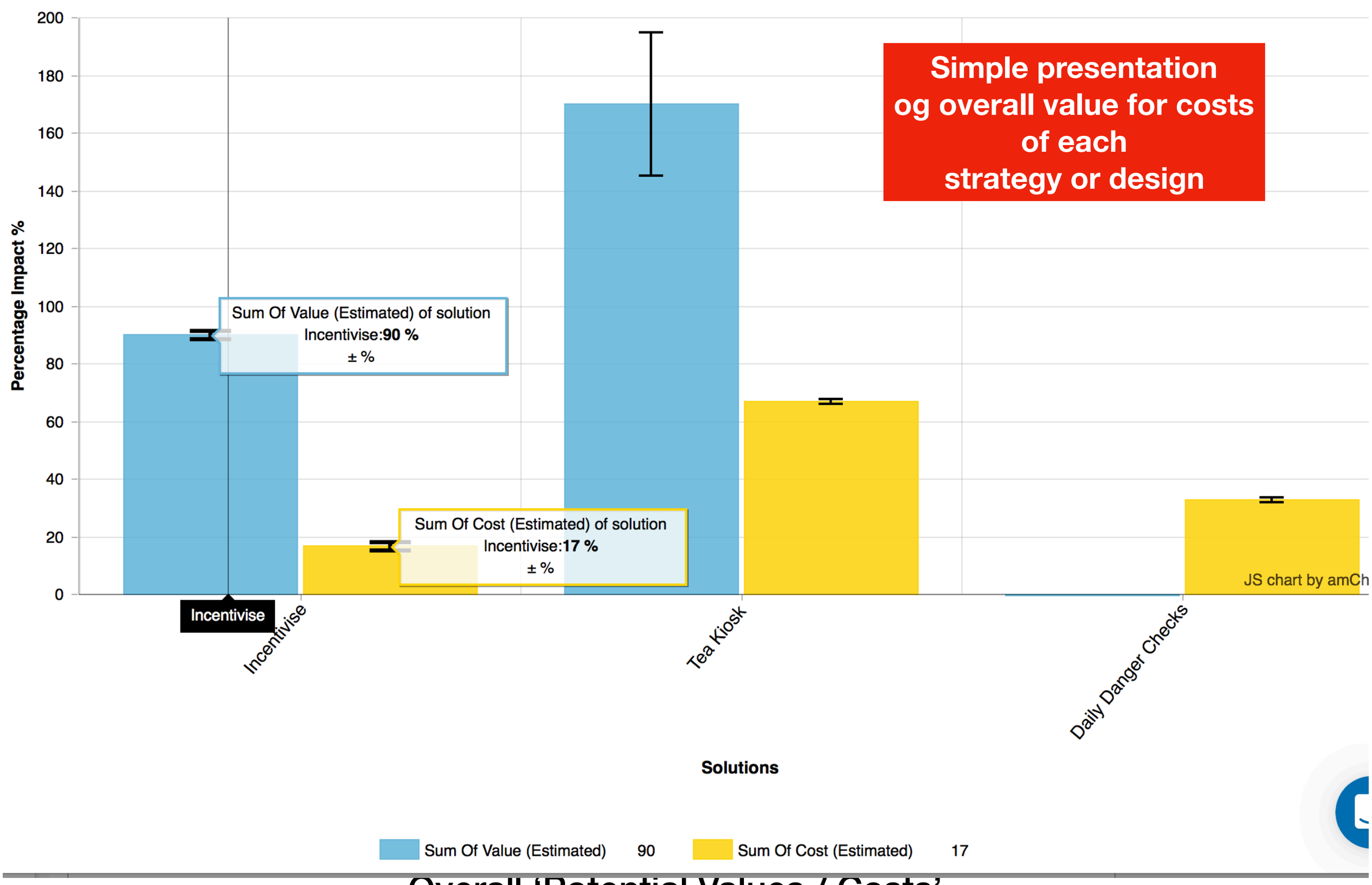
Designs ->	Incentivise	Tea Kiosk	Daily Danger Checks	Sum
Requirements				
Project Timeliness Status: **10** → Wish: **5** % % time overrun necessary to deliver ... [Project Cost Size = { Medium ($10k -...] 30th June 2017	=: Δ: Δ%: ?%:	**8** ± 0 -2 % **40** ± 0 % *32 % (x 0.8)* 40%	**5** ± 1 -5 % **100** ± 20 % *50 % (x 0.5)* 100%	**15** ± 8 5 % **-100** ± 160 % *-80 % (x 0.8)* -100%	ΣΔ%: **40** ± 180 %
Building Security Status: **50** → Wish: **10** % I... % of [Emergency Types] which in ...ct... [Emergency Types = { Earthquake }, 30th June 2018	=: Δ: Δ%: ?%:	**50** ± 0 0 % Injury **0** ± 0 % *0 % (x 0.0)* 0%	**50** ± 0 0 % Injury **0** ± NaN % *0 % (x 0.6)* 0%	**30** ± 10 -20 % Injury **50** ± 25 % *15 % (x 0.3)* 50%	ΣΔ%: **50** ± 25 %
User Productivity Status: **15** → Wish: **5** minutes number of minutes for a [user] to co... [user = { adult }, task = { dri...] 30th June 2017	=: Δ: Δ%: ?%:	**10** ± 0 -5 minutes **50** ± 0 % *0 % (x 0.0)* 50%	**8** ± 3 -7 minutes **70** ± 30 % *56 % (x 0.8)* 70%	**15** ± 0 0 minutes **0** ± 0 % *0 % (x 0.0)* 0%	ΣΔ%: **120** ± 30 %
Sum Of Values: Credibility - adjusted:	Σ%: Σ?%:	**90** ± 0 % *32 %*	**170** ± 50 % *106 %*	**-50** ± 185 % *-65 %*	
Method Implementation Cost Status: **0** → Budget: **3m** $ Total monetary cost in US Dollars fo... [Project Cost Size = { }] 30th June 2017	=: Δ: Δ%: ?%:	**500k** ± 0 500k $ **17** ± 0 % *34 % (x 0.0)* 17%	**2m** ± 0 2m $ **67** ± 0 % *134 % (x 0.0)* 67%	=:**1m** ± 0 Δ: 1m $ Δ%: **33** ± 0 % ?%: *66 % (x 0.0)* 33%	ΣΔ%: **117** ± 0 %
Sum Of Development Resources: Credibility - adjusted:	Σ%: Σ?%:	**17** ± 0 % *34 %*	**67** ± 0 % *134 %*	**33** ± 0 % *66 %*	
Value To Cost:		5.30	2.50	-1.50	

The numeric relation between ends and means.

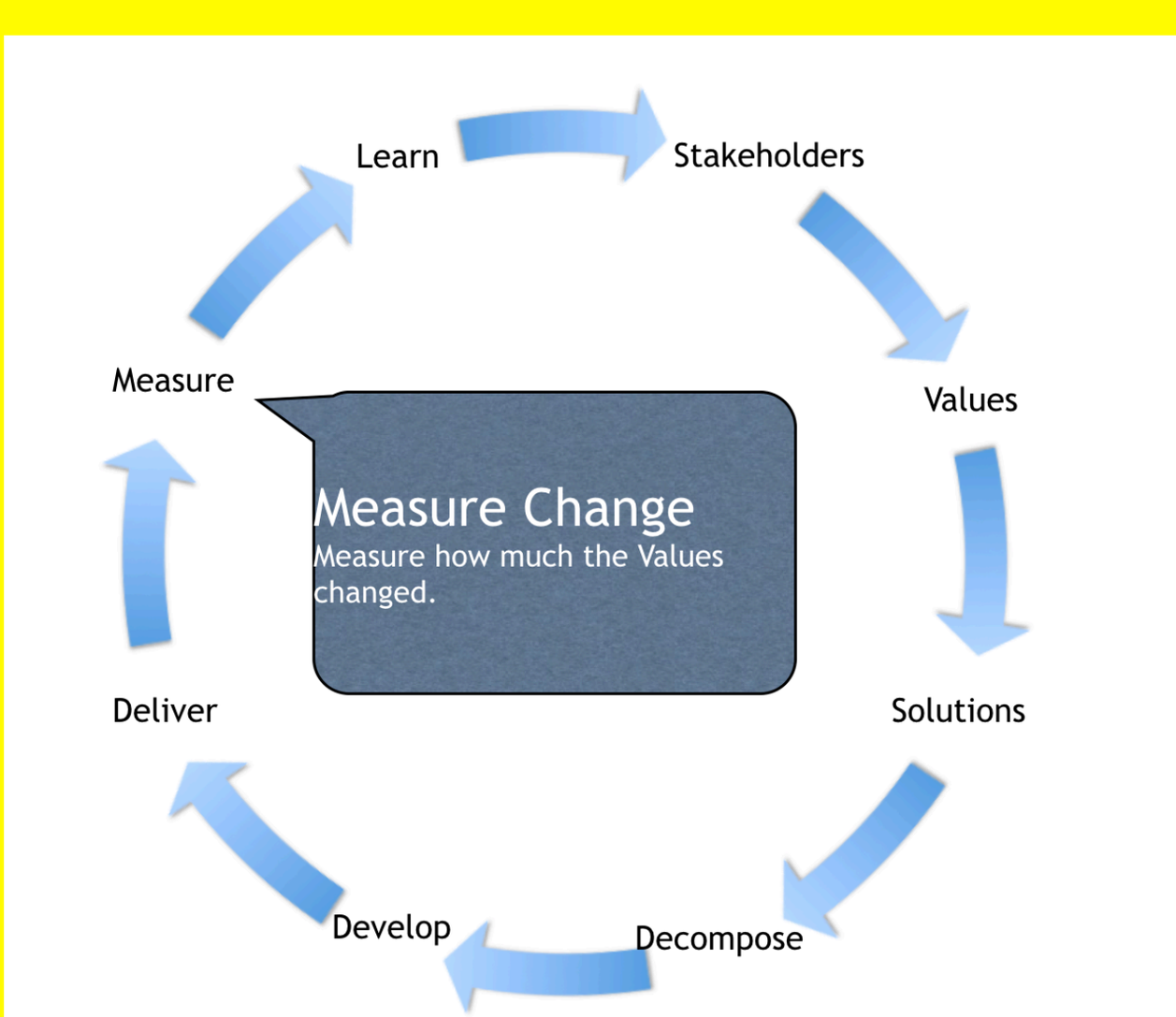
What items here help us to know what we do not know?

Basic Structure of an Impact Estimation Table

Simple presentation og overall value for costs of each strategy or design



Overall ‘Potential Values / Costs’
of 3 *options* or (if you need them all)
***complimentary* ‘benefit drivers’ = strategies = solutions = means’**



3. Evo and Advanced Agile: Multiple Measures, and Dynamic Design to Cost Estimation

An advanced, Deming, ‘Plan Do Study Act’ cycle
(Statistical Process Control)
and it is **all about numbers**
This is ‘Evo’ (Evolutionary Value Optimization)

Learn

Stakeholders

Values

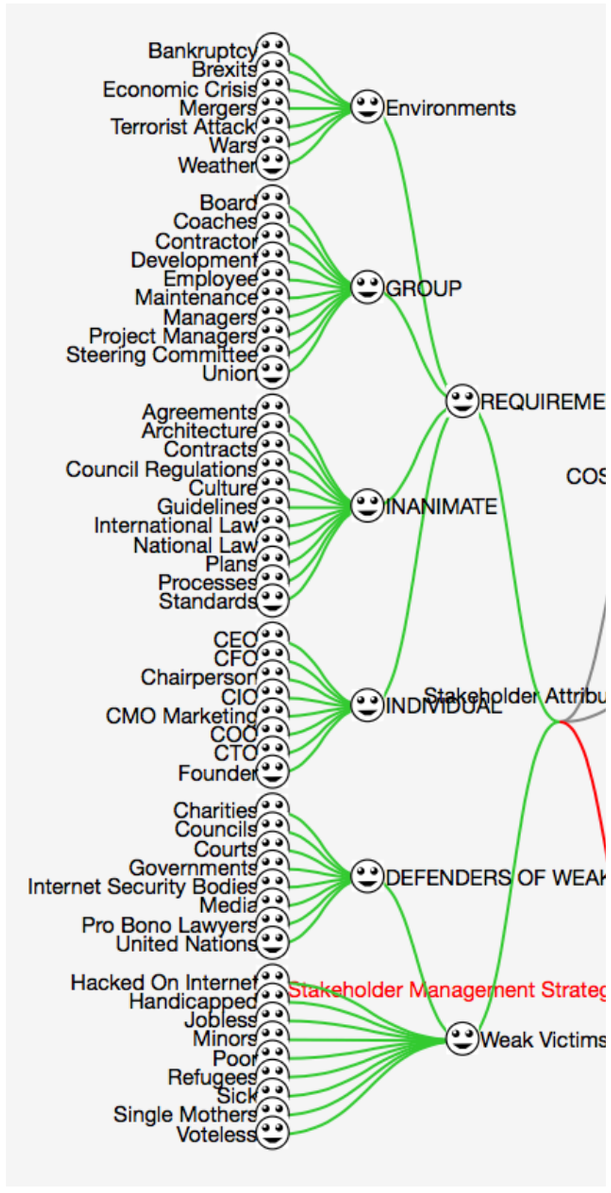
Solutions

Decompose

Develop

Deliver

Measure



Identify your critical stakeholders

the ones that have one or more critical needs,

that if you fail to deliver *them*,

your project/product

might well fail

Learn

Stakeholders

Values

Solutions

Decompose

Develop

Deliver

Measure

Requirements

Project Timeliness =:
Status: **10** → Wish: **5** % Δ:
% time overrun necessary to deliver ... Δ%:
[Project Cost Size = { Medium ($10k -...] ?%:
30th June 2017

Building Security =:
Status: **50** → Wish: **10** % l... Δ:
% of [Emergency Types] which in fact... Δ%:
[Emergency Types = { Earthquake }, ?%:
30th June 2018

User Productivity =:
Status: **15** → Wish: **5** minutes Δ:
number of minutes for a [user] to co... Δ%:
[user = { adult }, ?%:
task = { dri...]
30th June 2017

Sum Of Values: Σ%:
Credibility - adjusted: Σ?%:

Method Implementation Cost =:
Status: **0** → Budget: **3m** $ Δ:
Total monetary cost in US Dollars fo... Δ%:
[Project Cost Size = { }] ?%:
30th June 2017

Sum Of Development Resources: Σ%:
Credibility - adjusted: Σ?%:

Value To Cost:

Which numeric improvements do stakeholders need, *critically*?

**We can,
and must always,
express *their* values
with
well-defined numbers**

**Define both failure
and
success numerically**

and

**keep learning what
those
critical numbers are
continuously**

Learn

Stakeholders

Values

Solutions

Decompose

Develop

Measure

Deliver

Requirements	Incentivise	Tea Kiosk	Daily Danger Checks	Sum
Project Timeliness Status: **10** → Wish: **5** % % time overrun necessary to deliver ... [Project Cost Size = { Medium ($10k -...] 30th June 2017	**8** ± 0 -2 % **40** ± **0** % *32 % (x 0.8)* 40%	**5** ± 1 -5 % **100** ± **20** % *50 % (x 0.5)* 100%	**15** ± 8 5 % **-100** ± **160** % *-80 % (x 0.8)* -100%	ΣΔ%: **40** ± **180** %
Building Security Status: **50** → Wish: **10** % I... % of [Emergency Types] which in fact... [Emergency Types = { Earthquake }, 30th June 2018	**50** ± 0 0 % Injury **0** ± **0** % *0 % (x 0.0)* 0%	**50** ± 0 0 % Injury **0** ± **NaN** % *0 % (x 0.6)* 0%	**30** ± 10 -20 % Injury **50** ± **25** % *15 % (x 0.3)* 50%	ΣΔ%: **50** ± **25** %
User Productivity Status: **15** → Wish: **5** minutes number of minutes for a [user] to co... [user = { adult }, task = { dri...] 30th June 2017	**10** ± 0 -5 minutes **50** ± **0** % *0 % (x 0.0)* 50%	**8** ± 3 -7 minutes **70** ± **30** % *56 % (x 0.8)* 70%	**15** ± 0 0 minutes **0** ± **0** % *0 % (x 0.0)* 0%	
Sum Of Values: Credibility - adjusted:	Σ%: **90** ± **0** % Σ?%: *32 %*	**170** ± **50** % *106 %*	**-50** ± **185** % *-65 %*	
Method Implementation Cost Status: **0** → Budget: **3m** $ Total monetary cost in US Dollars fo... [Project Cost Size = { }] 30th June 2017	**500k** ± 0 500k $ **17** ± **0** % *34 % (x 0.0)* 17%	**2m** ± 0 2m $ **67** ± **0** % *134 % (x 0.0)* 67%	**1m** ± 0 1m $ **33** ± **0** % *66 % (x 0.0)* 33%	ΣΔ%: **117** ± **0** %
Sum Of Development Resources: Credibility - adjusted:	Σ%: **17** ± **0** % Σ?%: *34 %*	**67** ± **0** % *134 %*	**33** ± **0** % *66 %*	
Value To Cost:	5.30	2.50	-1.50	

Solutions (designs, architectures, strategies)

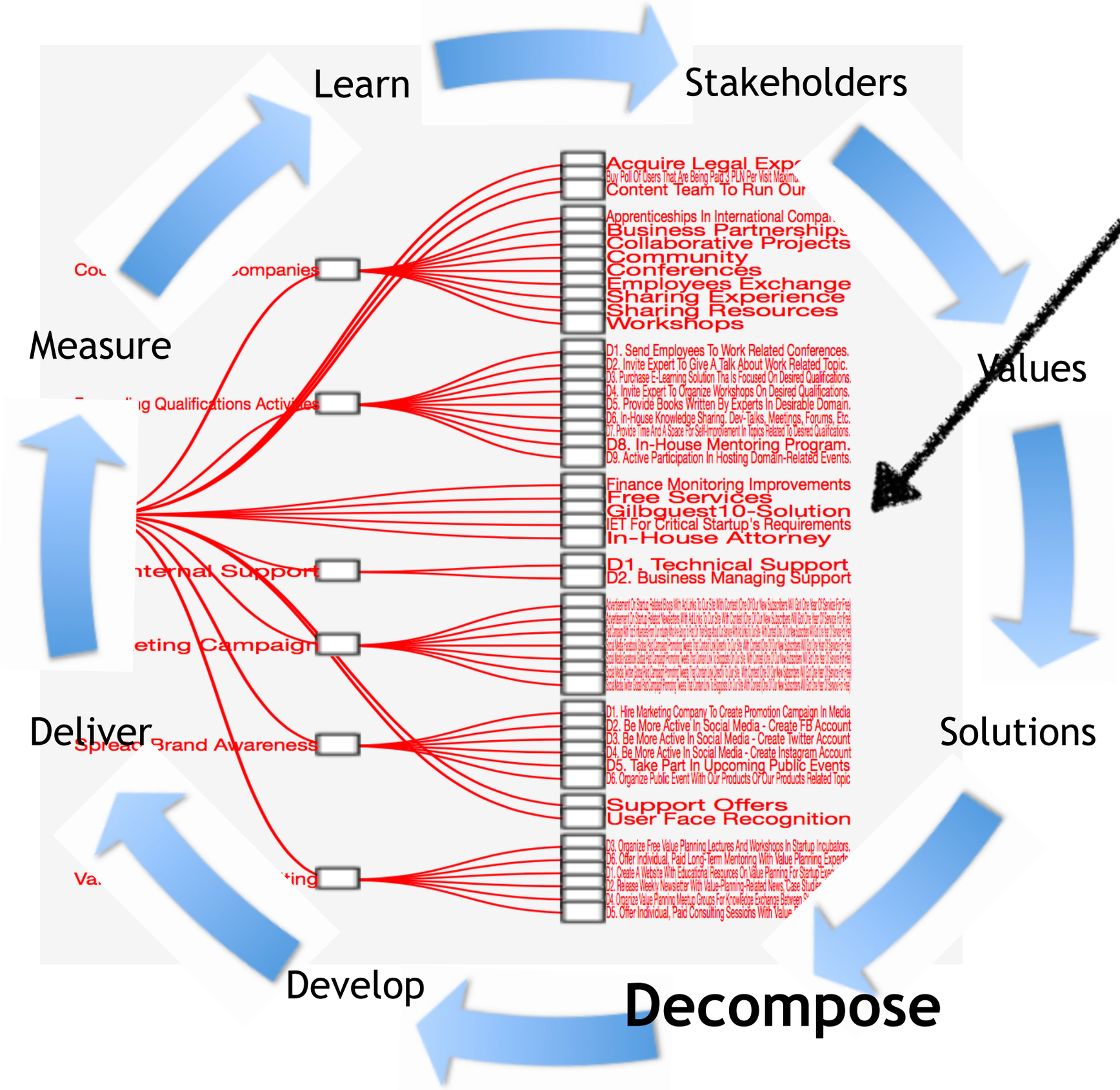
must be identified

and their total impacts on critical objectives and constraints

must be estimated reasonably

(order of magnitude)

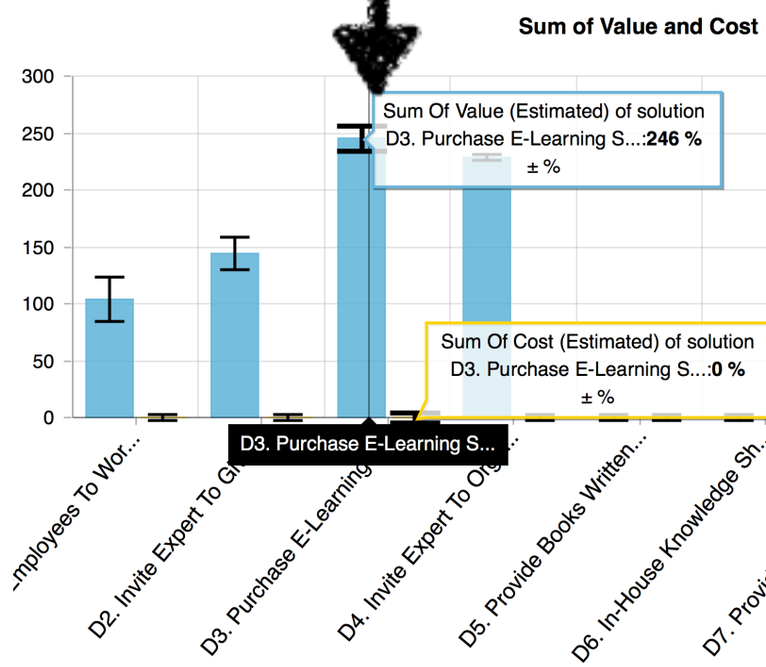
Impact Estimation Tables (Planguage) are a tool for doing estimates of potential solutions and how good they might be



The solutions can be <u>decomposed</u> by 10x or 100x

And we can <u>estimate</u> the solution <u>sub-component</u> value and cost,

so as to prioritize the best value/cost for short term delivery



Learn

Stakeholders

Values

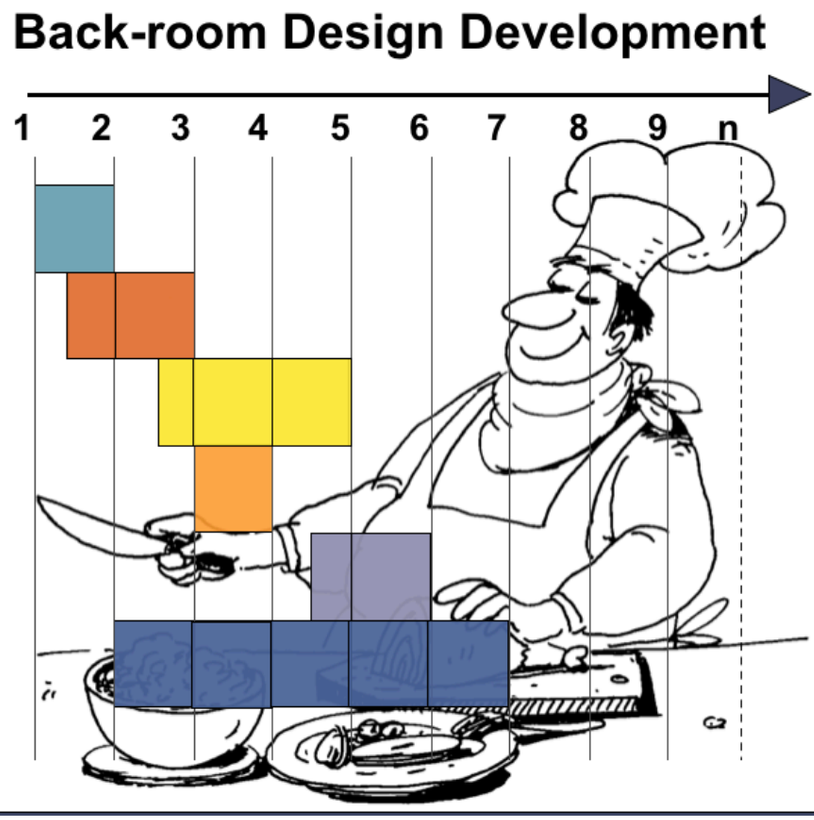
Solutions

Decompose

Develop

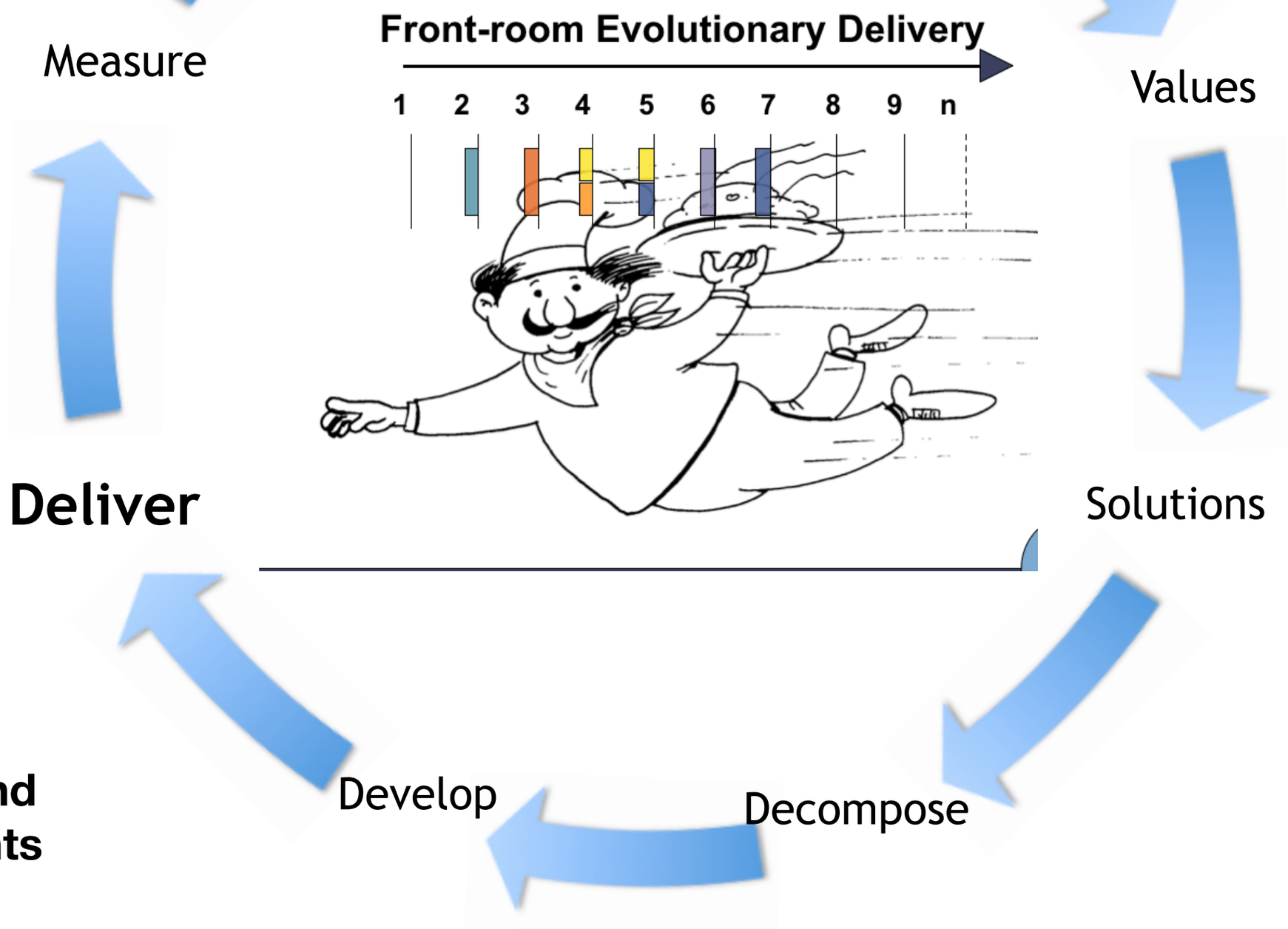
Deliver

Measure



The sub-solutions are made ready (developed) for delivery to real stakeholders, next week and every week. Or in about 2% of budget/ deadline increments

The sub-solutions are *delivered*
to real stakeholders,
in order to experiment,
to test, to pilot, to get reactions,
NUMERICALLY
and to allow for potential corrections in design, in implementation process, and in lower-priority requirements



The sub-solutions are
measured as to effect
on
all the
top
stakeholder
critical
objectives,
and
on their critical cost
increments,
with a view to improving
prediction of
final cumulative costs

Learn

Stakeholders

Values

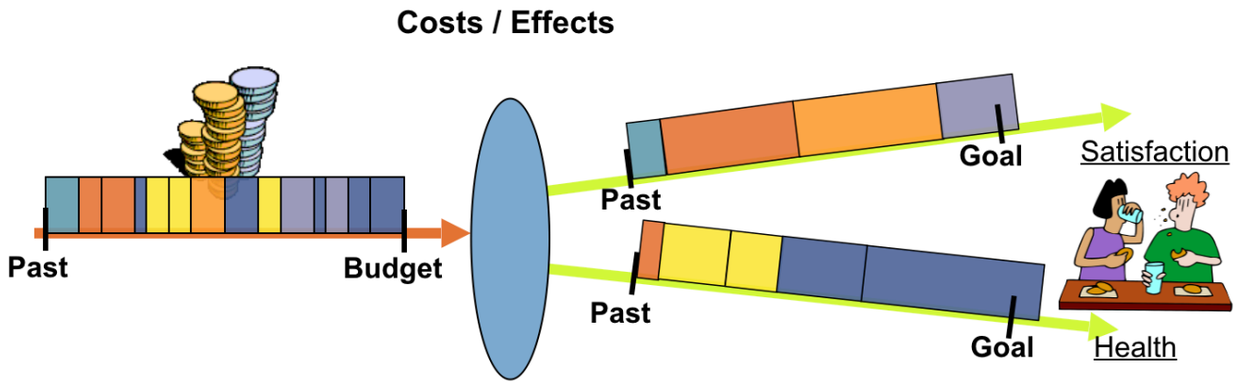
Solutions

Decompose

Develop

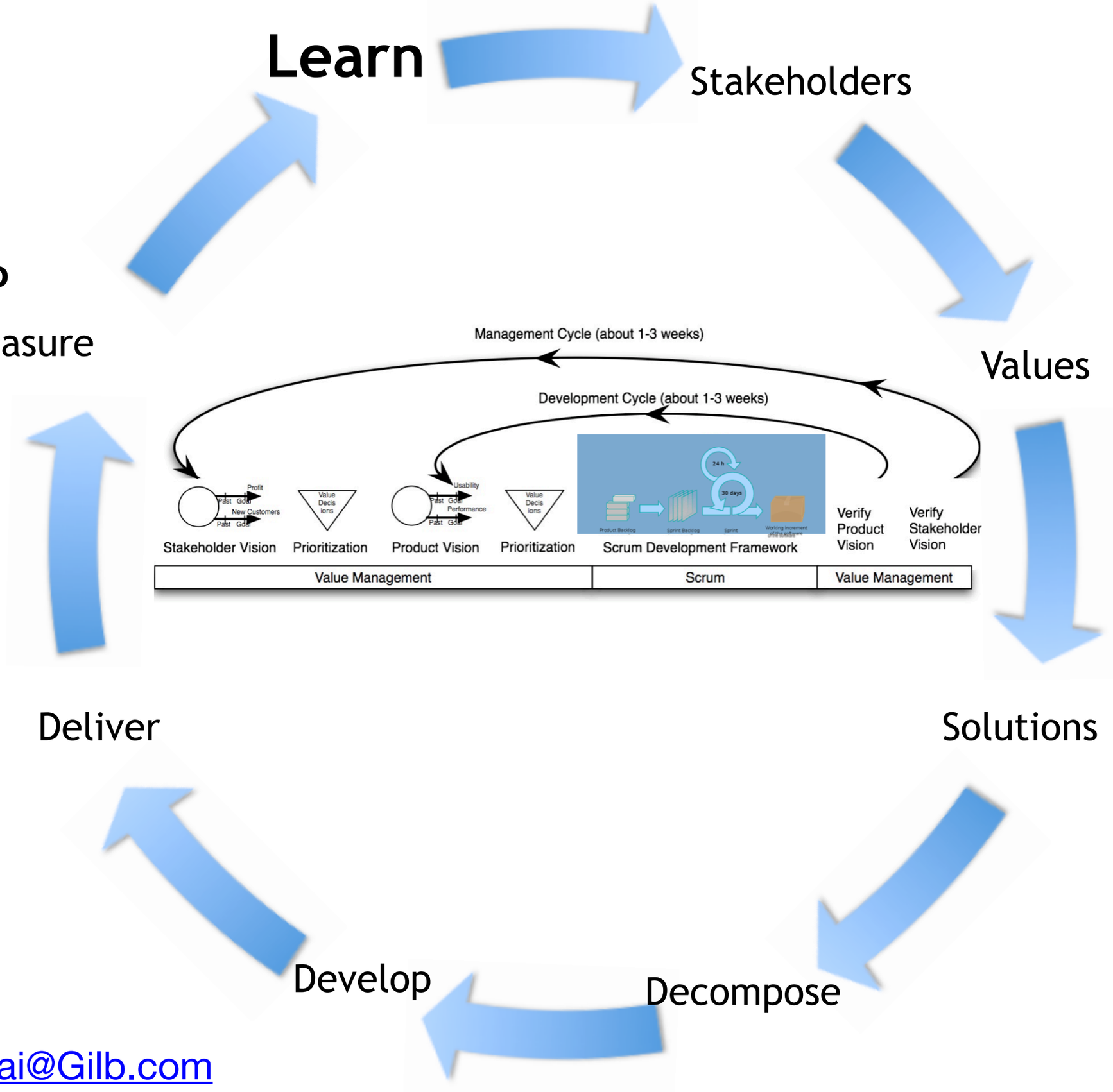
Deliver

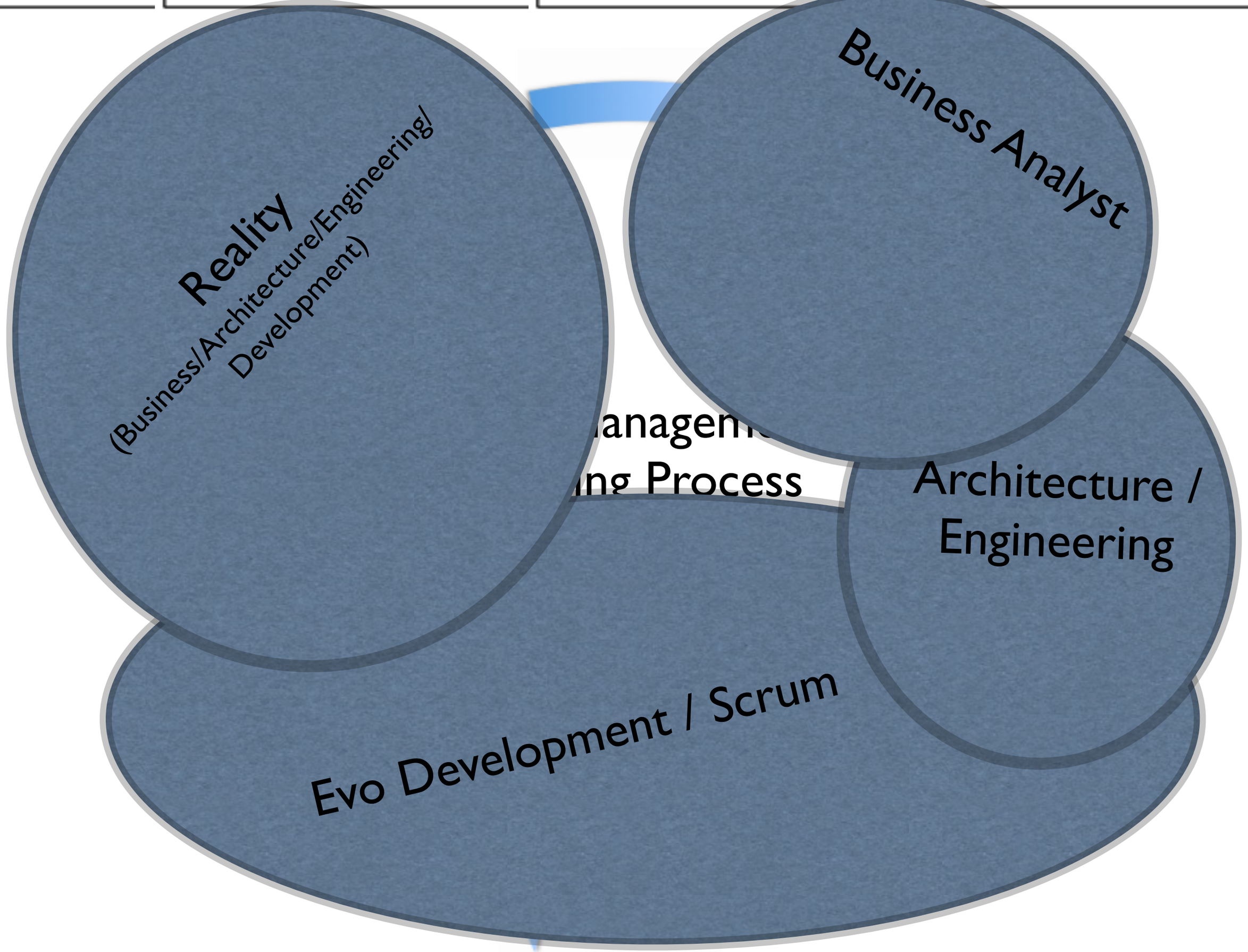
Measure



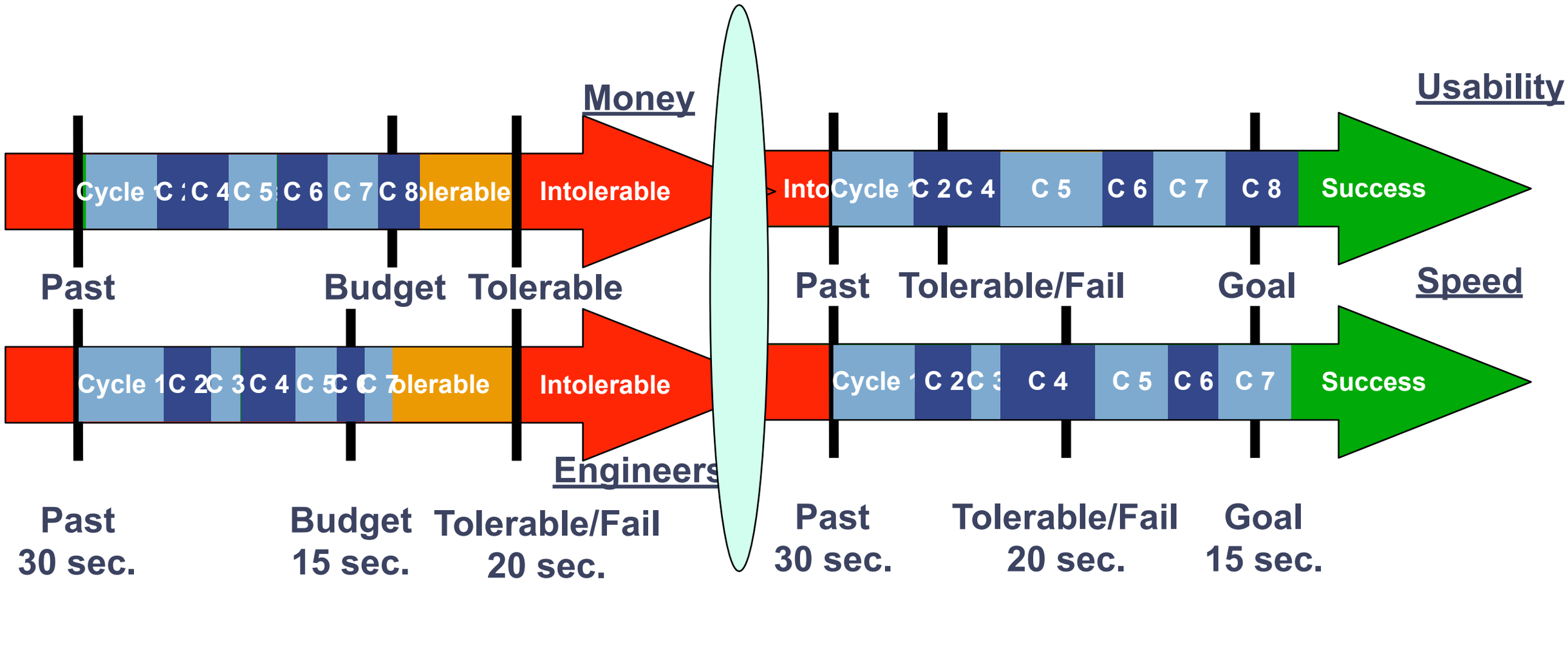
From the measurements,
and
other feedback
from stakeholders

Learn what you need to do
to avoid failure
and to succeed

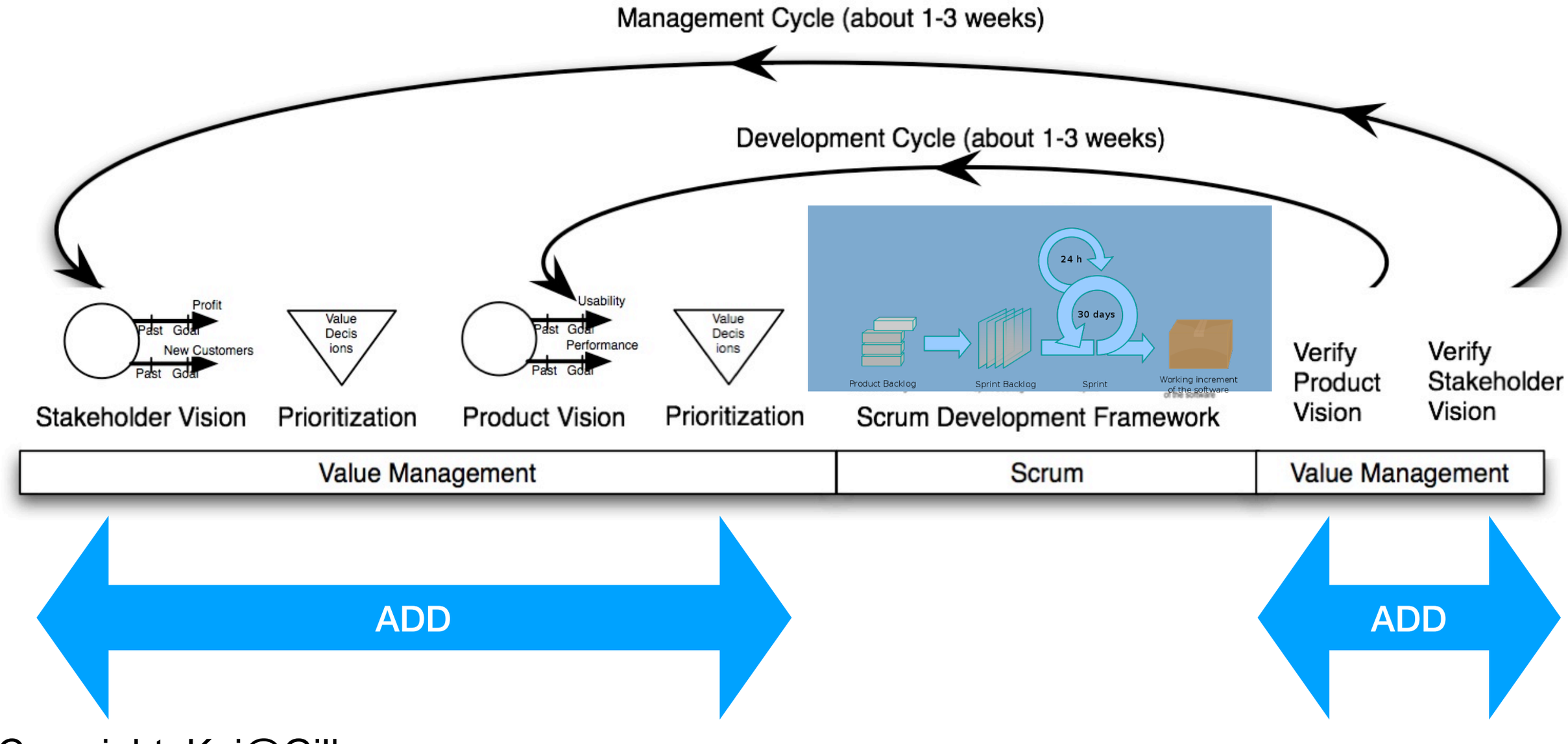



Reality
(Business/Architecture/Engineering/
Development)
Business Analyst
anagem
ng Process
Architecture /
Engineering
Evo Development / Scrum

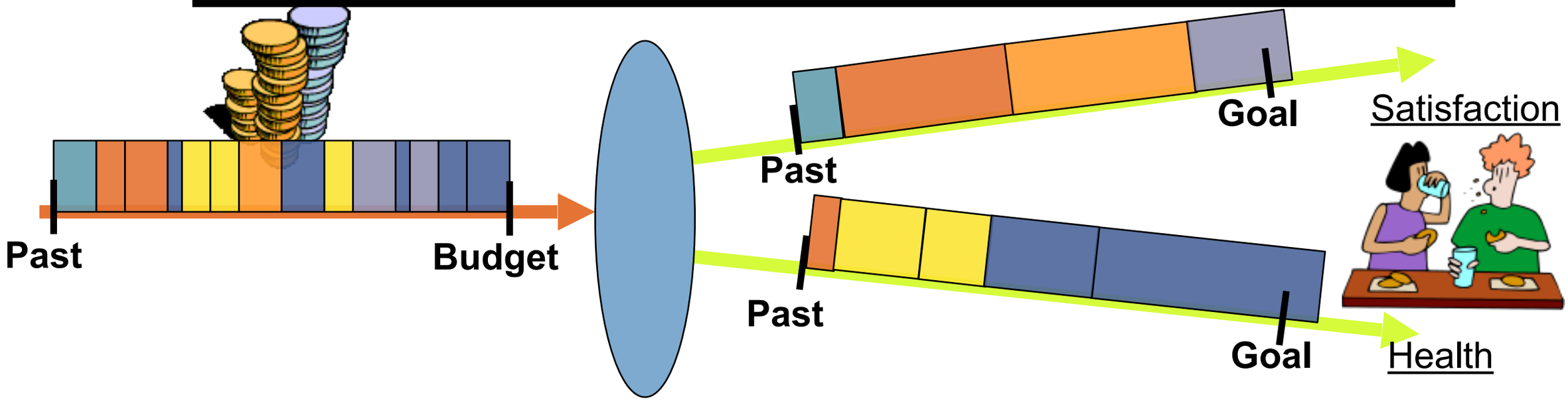
Each Evolutionary Cycle consumes a budget of Development Resources. We need to keep our eyes on something like 14 critical top-level value-and-resource requirements *simultaneously*. So we need tools, tables and numbers to help us to keep track of it all, both individually, and as scattered teams



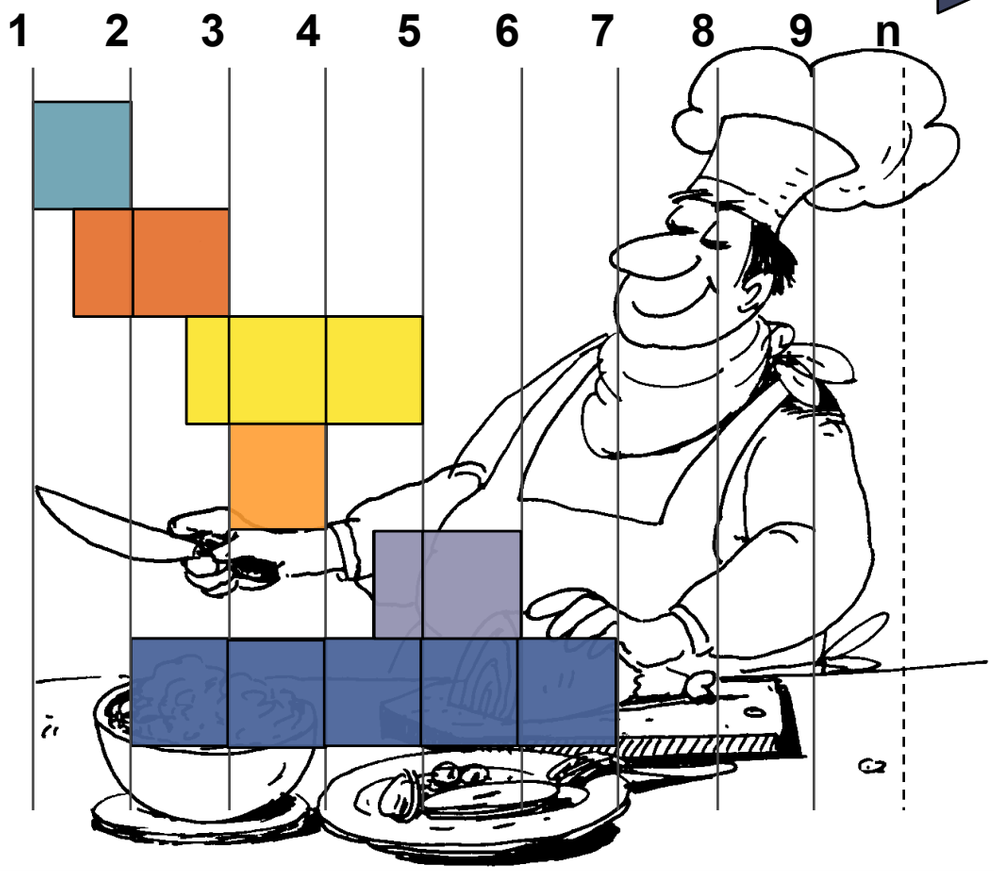
We need to add: ‘Value Management’:
Quantified, Engineering, Not just ‘coding’



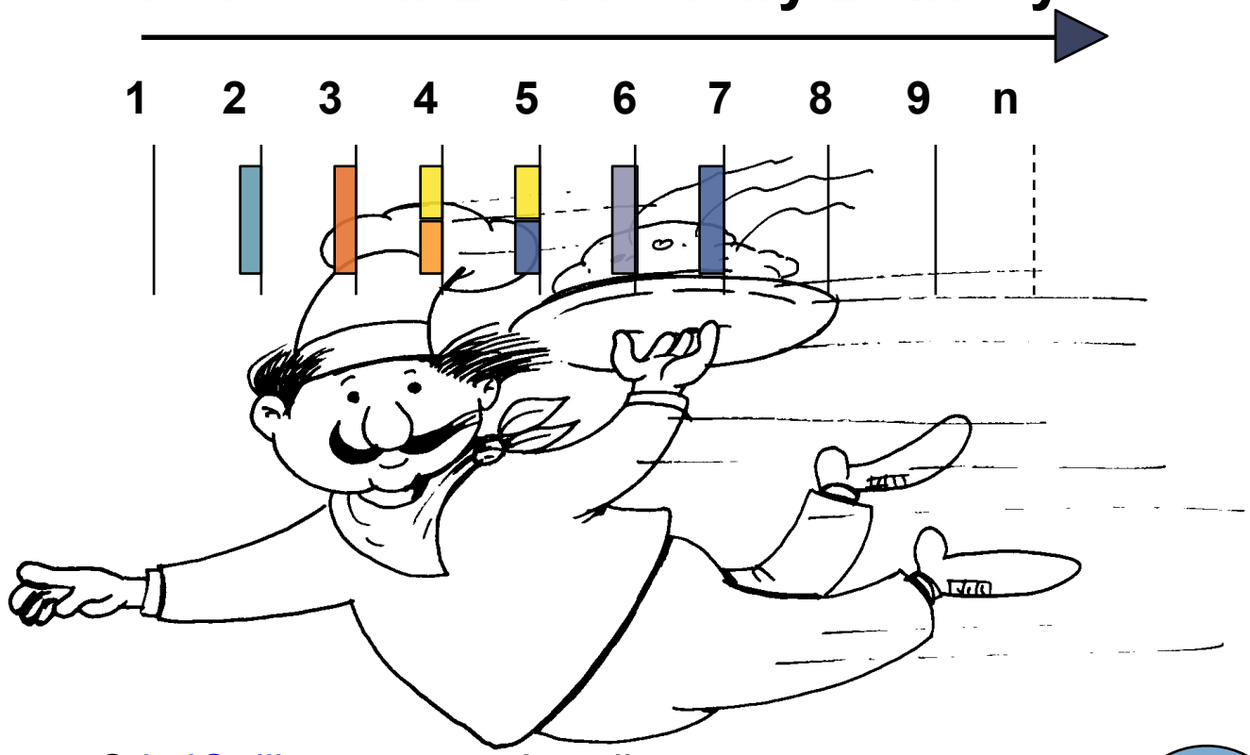
Sometimes 2% or weekly
decomposition is really impossible
so we develop long chunks in the Back-room
But we keep the value delivery frequency up in the Front-room, facing the stakeholders



Back-room Design Development



Front-room Evolutionary Delivery



‘Cleanroom Method’
at IBM Federal Systems Division (1980)

Dr. Harlan D. Mills
(May 14, 1919 – January 8, 1996)

quality is designed in, not tested in

“The first guarantee of quality in design is in well-informed, well-educated, and well-motivated designers.

Quality must be **built into designs, and cannot be inspected in or tested in.**

Nevertheless, any prudent development process **verifies quality through inspection** and testing.

Inspection by peers in design, by users or surrogates, by other financial specialists concerned with cost, reliability, or maintainability not only increases confidence in the design at hand, but also provides designers with valuable lessons and insights to be applied to future designs.

The very fact that **designs face inspections motivates even the most conscientious designers to greater care,** deeper simplicities, and more precision in their work.” Harlan Mills, IBM

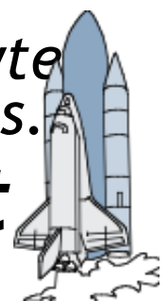
inIBM sj 4 80 p.419
In

Mills, H. 1980. The management of software engineering: part 1: principles of software engineering. IBM Systems Journal 19, issue 4 (Dec.):414-420.
Direct Copy
http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan
Library header
http://trace.tennessee.edu/utk_harlan/5/

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

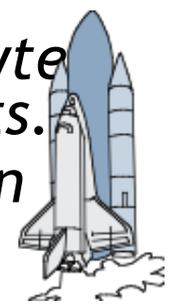
- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) "some ten years ago [Ed. about 1970] in a continuing evolution that is still underway:*
- *Ten years ago general management expected the worst from software projects – cost overruns, 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 deliveries [Ed. Note 2%!]s. Every one of those deliveries was on time and under budget*

- *A more extended example can be found in the NASA space program,*
- *- Where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million bytes of program and data for ground and space processors in over a dozen projects.*
- *- **There were few late or overrun deliveries in that decade, and none at all in the past four years.**"*



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

- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, f[illegible] ... [illegible]out*
- *T[illegible] ... [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] ... [illegible]rocessors distrib[illegible] ... [illegible]veries [Ed. Note 2[illegible] ... [illegible]dget*
- *A mor[illegible] ... [illegible]"*
- *- Whe[illegible] ... [illegible]-years of softwa[illegible] ... [illegible]million byte of pro[illegible] ... [illegible]en projects.*
- *- Ther[illegible] ... [illegible]ne at all in the pa[illegible]*



in 45 incremental deliveries

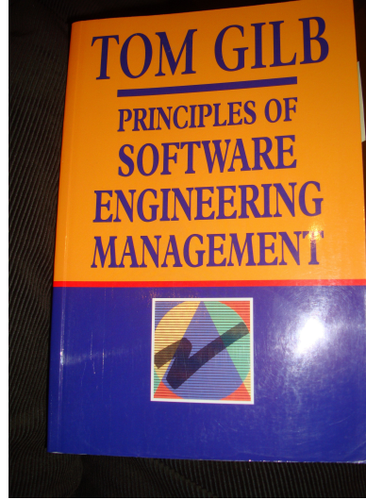
were few late or overrun deliveries in that decade, and none at all in the past four years

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 System Journal, No. 4 1980 p.420, see Links below

Mills, H. 1980. The management of software engineering: part 1: principles of software engineering. IBM Systems Journal 19, issue 4 (Dec.):414-420.
Direct Copy
http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan
Library header
http://trace.tennessee.edu/utk_harlan/5/

Robert E. Quinnan (-2015): IBM FSD Cleanroom *Dynamic Design to Cost*



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

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

-

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

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

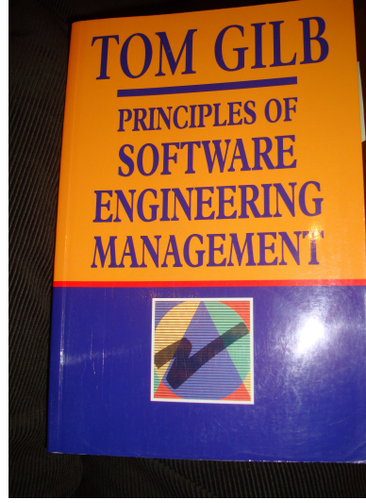
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability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

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

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

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

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

'Cost management. . . anagement farther by introducing design-t tegrated way to ensure that software technical m k by Figure 7.10] consists of developing a design,

-

He goes on to r by sacrificing 'planned capability.' When a sa t of each increment can proceed concurrently with the

'Design is an iterative

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

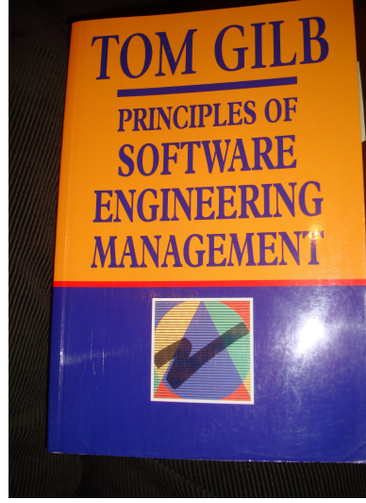
'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

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

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

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 concurrently with the program design of the others.'

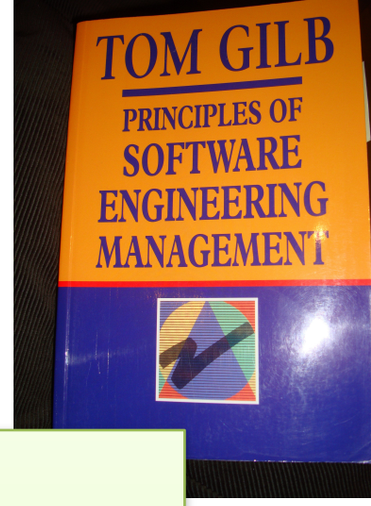
'Design is an iterative

It is clear from **in seeking the appropriate balance between cos** **thus reducing the complexity of the task, and increas** **d as the true cost of the increment becomes a**

'When the developme **crements is computed.' (p. 474)**
Source: Robert E. Quir 1980, pp. 466~77
This text is cut from G

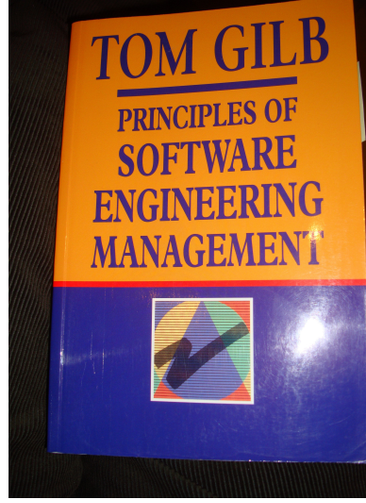
iteration process trying to meet cost targets by either *redesign* or by *sacrificing* 'planned capability'

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Design is an iterative process

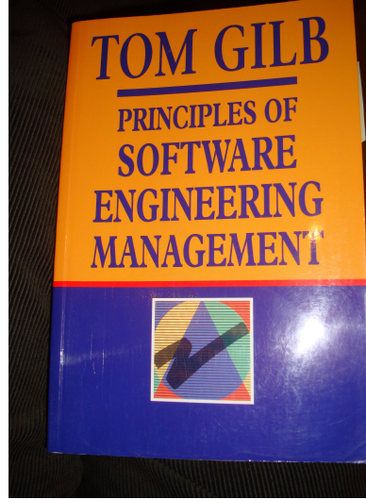
Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

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

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



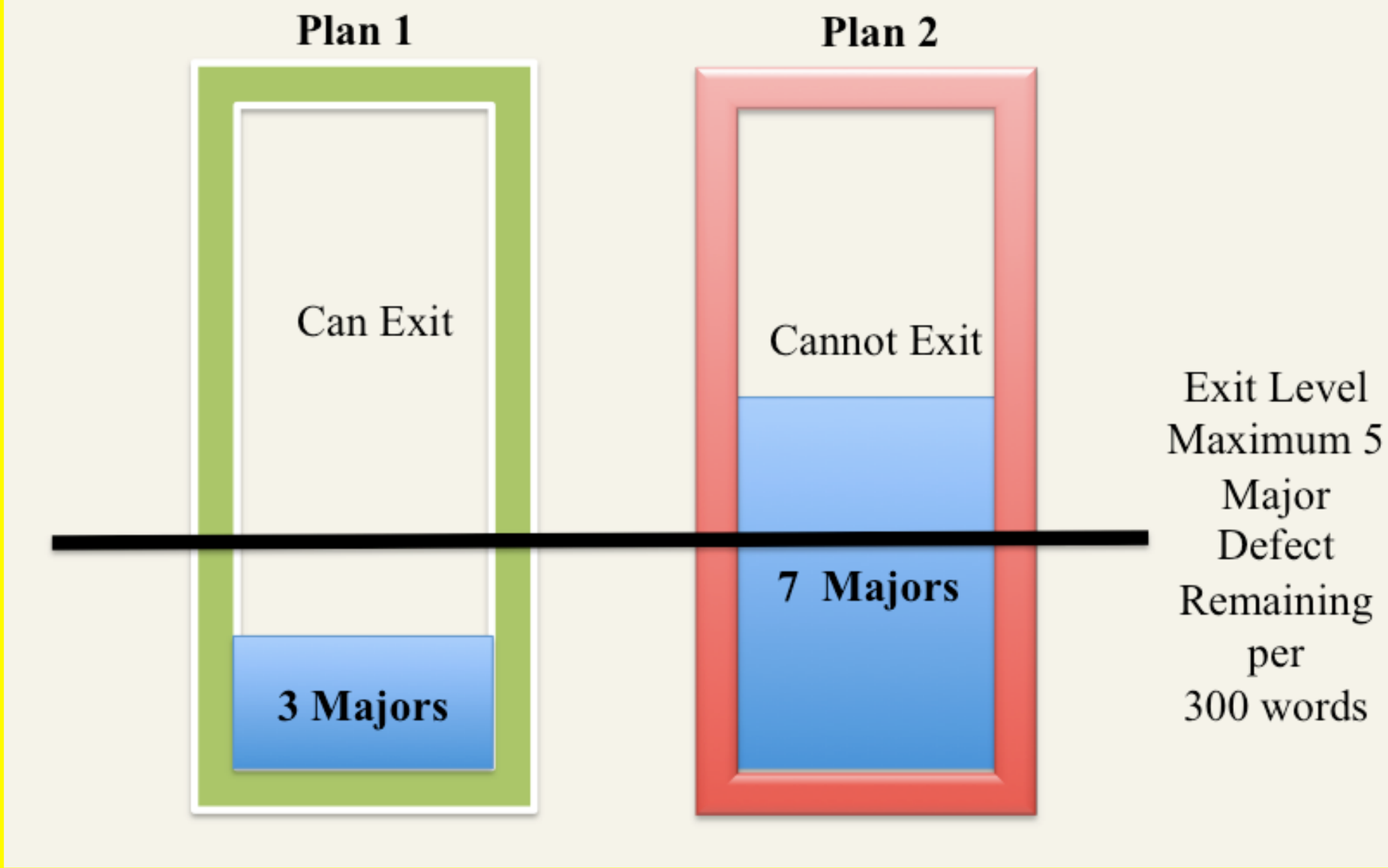
Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

an estimate to complete the remaining increments is computed.

ed

of

.)

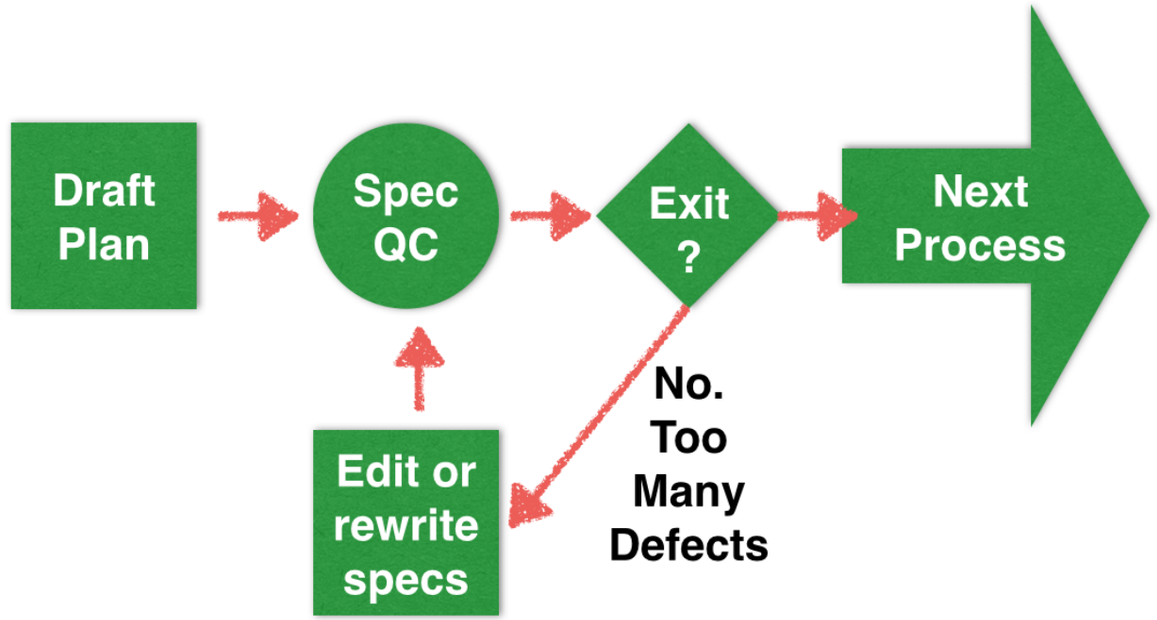


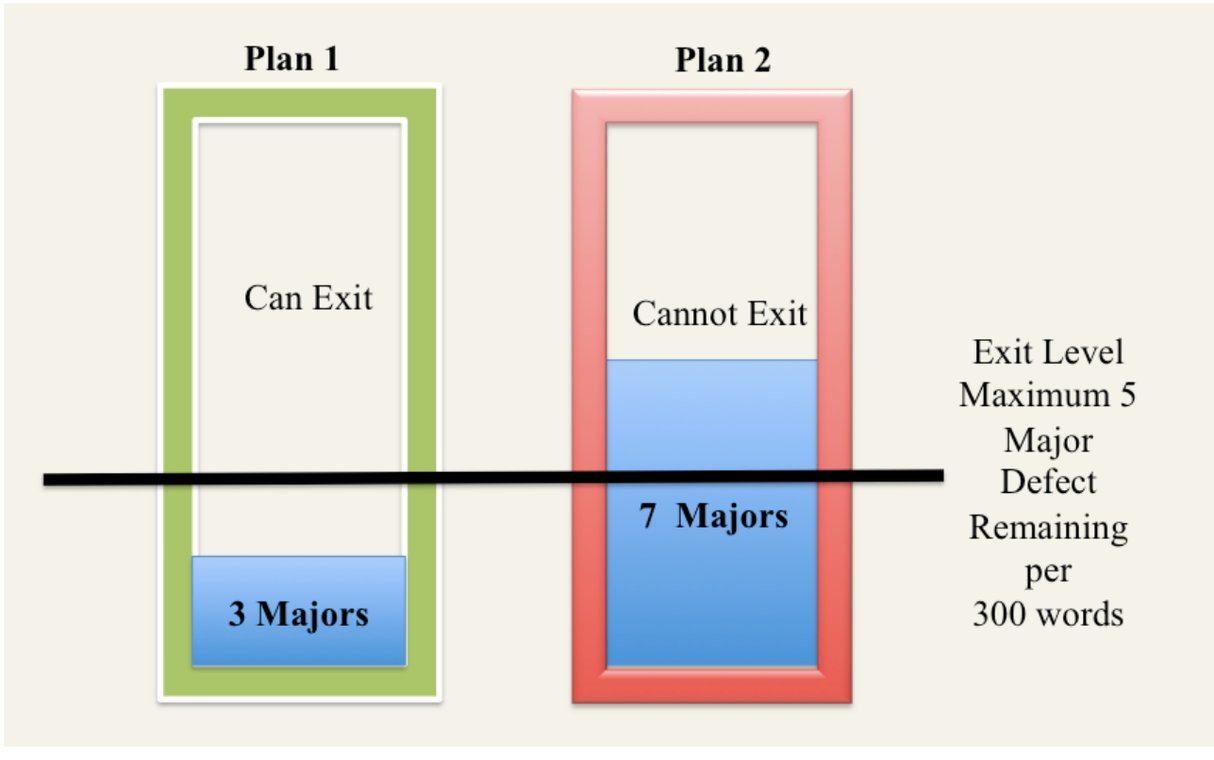
4. Measuring Development Specifications Quality: Lean Quality Assurance

The Agile Specification Quality Control process

for lean (early, prevents defect injection) measurement of quality of requirements, architecture specs, and contracts

- Our IT planning documents are heavily polluted
- with dozens of ‘major defects’ per page
- we need to measure defects by sampling
- and we need to refuse to ‘exit’ garbage out
- this lean approach can improve productivity 2x and 3x (Intel)





Source Eric Simmons, erik.simmons@construx.com
25 Oct 2011. See Terzakis research reports.

A Recent Example

Application of 'Specification Quality Control' (Gilb method) by an Intel software team, resulted in the following defect-density reduction, in requirements over several months:

Rev.	# of Defects	# of Pages	Defects/ Page (DPP)	% Change in DPP
0.3	312	31	10.06	
0.5	209	44	4.75	-53%
0.6	247	60	4.12	-13%
0.7	114	33	3.45	-16%
0.8	45	38	1.18	-66%
1.0	10	45	0.22	-81%
Overall % change in DPP revision 0.3 to 1.0:				-98%

Downstream benefits:
- Scope delivered at the Alpha milestone increased 300%, **released scope up 233%**
- SW defects reduced by ~50%
- Defects that did occur were resolved in far less time on average

Industrial Studies of Planguage and SQC to measure quality of requirements

The Impact of Requirements on Software Quality across Three Product Generations

John Terzakis

Intel Corporation, USA
john.terzakis@intel.com

***Abstract*—In a previous case study, we presented data demonstrating the impact that a well-written and well-reviewed set of requirements had on software defects and other quality indicators between two generations of an Intel product. The first generation was coded from an unorganized collection of requirements that were reviewed infrequently and informally. In contrast, the second was developed based on a set of requirements stored in a Requirements Management database and formally reviewed at each revision. Quality indicators for the second software product all improved dramatically even with the increased complexity of the newer product. This paper will recap that study and then present data from a subsequent Intel case study revealing that quality enhancements continued on the third generation of the product. The third generation software was designed and coded using the final set of requirements from the second version as a starting point. Key product differentiators included changes to operate with a new Intel processor, the introduction of new hardware platforms and the addition of approximately fifty new features. Software development methodologies were nearly identical, with only the change to a continuous build process for source code check-in added. Despite the enhanced functionality and complexity in the third generation software, requirements defects, software defects, software sightings, feature commit vs. delivery (feature variance), defect closure efficiency rates, and number of days from project commit to customer release all improved from the second to the third generation of the software.**

***Index Terms*—Requirements specification, requirements defects, reviews, software defects, software quality, multi-generational software products.**

I. Introduction

This paper is a continuation of an earlier short paper [1] that presented quality indicator data from a case study of two generations of an Intel software product. The prior case study

II. Product Backgrounds

The requirements for Gen 1 that existed were scattered across a variety of documents, spreadsheets, emails and web sites and lacked a consistent syntax. They were under lax revision and change control, which made determining the most current set of requirements challenging. There was no overall requirements specification; hence reviews were sporadic and unstructured. Many of the legacy features were not documented. As a result, testing had many gaps due to missing and incorrect information.

The Gen 1 product was targeted to run on both desktop and laptop platforms running on an Intel processor (CPU). Code was developed across multiple sites in the United States and other countries. Integration of the code bases and testing occurred in the U.S. The Software Development Lifecycle (SDLC) was approximately two years.

After analyzing the software defect data from the Gen 1 release, the Gen 2 team identified requirements as a key improvement area. A requirements Subject Matter Expert (SME) was assigned to assist the team in the elicitation, analysis, writing, review and management of the requirements for the second generation product. The SME developed a plan to address three critical requirements areas: a central repository, training, and reviews. A commercial Requirements Management Tool (RMT) was used to store all product requirements in a database. The data model for the requirements was based on the Planguage keywords created by Tom Gilb [2]. The RMT was configured to generate a formatted Product Requirements Document (PRD) under revision control. Architecture specifications, design documents and test cases were developed from this PRD. The SME provided training on best practices for writing requirements, including a standardized syntax, attributes of well written requirements and Planguage to the primary authors (who were all located in United States). Once the training was complete, the primary author submitted early samples of his requirements

2013 Rio Paper
https://www.thinkmind.org/download.php?articleid=iccgi_2013_3_10_10012

Tool Credit:

www.NeedsandMeans.com

Richard Smith, London

Requirements		Incentivise	Tea Kiosk	Daily Danger Checks	Sum
Project Timeliness Status: **10** → Wish: **5** % % time overrun necessary to deliver ... [Project Cost Size = { Medium ($10k -...] 30th June 2017	=: Δ: Δ%: ?%:	**8** ± 0 -2 % **40** ± 0 % *32 % (x 0.8)* 40%	**5** ± 1 -5 % **100** ± 20 % *50 % (x 0.5)* 100%	**15** ± 8 5 % **-100** ± 160 % *-80 % (x 0.8)* -100%	ΣΔ%: **40** ± 180 %
Building Security Status: **50** → Wish: **10** % I... % of [Emergency Types] which in fact... [Emergency Types = { Earthquake }, 30th June 2018	=: Δ: Δ%: ?%:	**50** ± 0 0 % Injury **0** ± 0 % *0 % (x 0.0)* 0%	**50** ± 0 0 % Injury **0** ± NaN % *0 % (x 0.6)* 0%	**30** ± 10 -20 % Injury **50** ± 25 % *15 % (x 0.3)* 50%	ΣΔ%: **50** ± 25 %
User Productivity Status: **15** → Wish: **5** minutes number of minutes for a [user] to co... [user = { adult }, task = { dri...] 30th June 2017	=: Δ: Δ%: ?%:	**10** ± 0 -5 minutes **50** ± 0 % *0 % (x 0.0)* 50%	**8** ± 3 -7 minutes **70** ± 30 % *56 % (x 0.8)* 70%	**15** ± 0 0 minutes **0** ± 0 % *0 % (x 0.0)* 0%	ΣΔ%: **120** ± 30 %
Sum Of Values: Credibility - adjusted:	Σ%: Σ?%:	**90** ± 0 % *32 %*	**170** ± 50 % *106 %*	**-50** ± 185 % *-65 %*	
Method Implementation Cost Status: **0** → Budget: **3m** $ Total monetary cost in US Dollars fo... [Project Cost Size = { }] 30th June 2017	=: Δ: Δ%: ?%:	**500k** ± 0 500k $ **17** ± 0 % *34 % (x 0.0)* 17%	**2m** ± 0 2m $ **67** ± 0 % *134 % (x 0.0)* 67%	=:**1m** ± 0 Δ: 1m $ Δ%: **33** ± 0 % ?%: *66 % (x 0.0)* 33%	ΣΔ%: **117** ± 0 %
Sum Of Development Resources: Credibility - adjusted:	Σ%: Σ?%:	**17** ± 0 % *34 %*	**67** ± 0 % *134 %*	**33** ± 0 % *66 %*	
Value To Cost:		5.30	2.50	-1.50	

End Game

So, what are my main messages to you?

- You can expand your current use of metrics to **include QUALITY, and VALUE metrics**

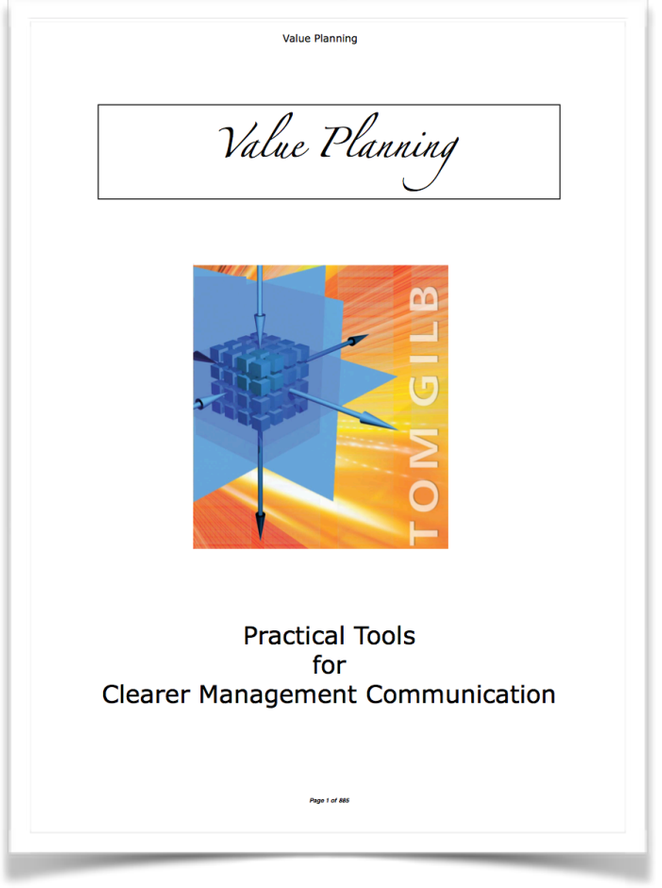
- **Quantification** of values is useful, even *<u>without</u> measurement. Quantification itself* is useful for **clearer communication** about critical objectives

- **Estimation** of 'multiple critical impacts' of any design/architecture/strategy, is useful for intelligent **prioritization** of value delivery, and for considering **risks**

- You can manage **costs and deadlines** by agile **feedback and correction**; the 'dynamic design to cost' process

- We can and should **measure the quality of upstream planning,** and code, specs, in order to motivate people, to follow high standards of specification, and to avoid downstream bugs and delays

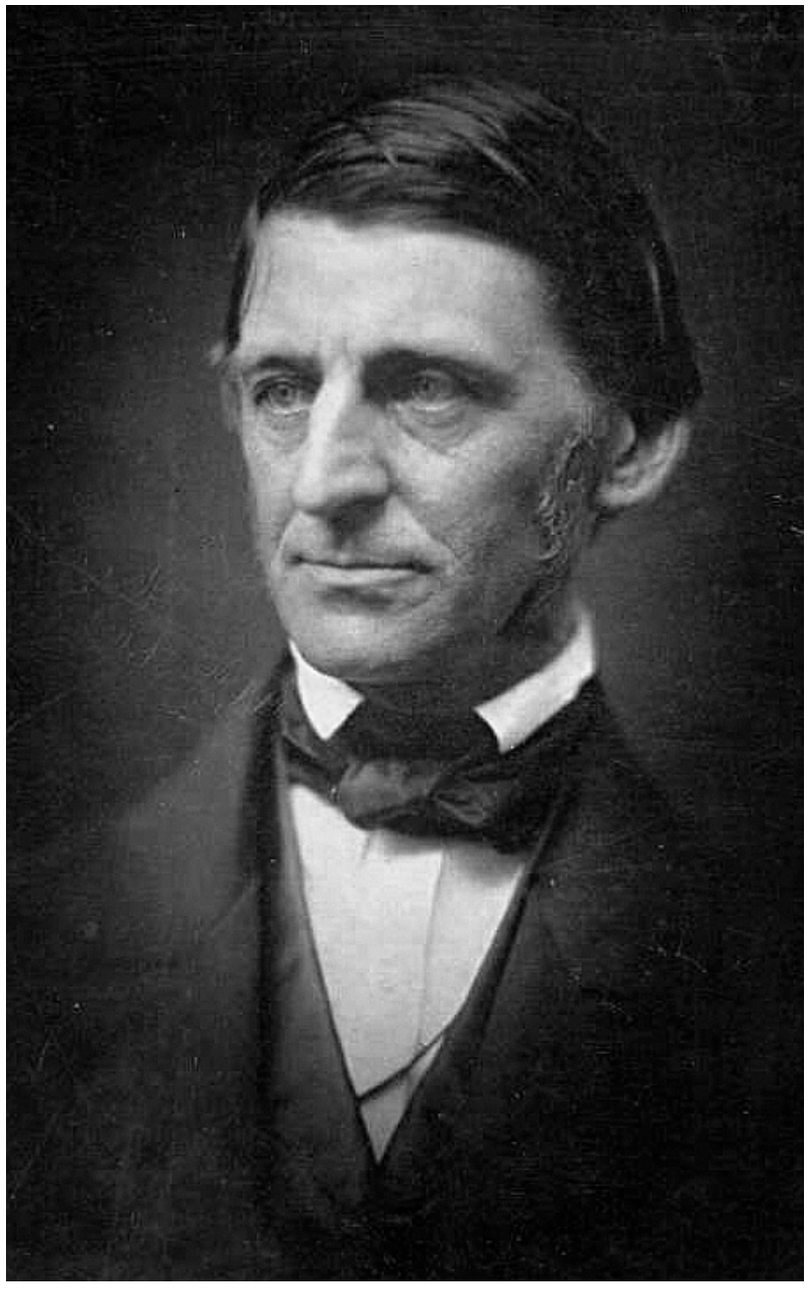
Get a free e-copy
of 'Competitive Engineering' book.
https://www.gilb.com/p/competitive-engineering



Link to book: https://www.gilb.com/store/2W2zCX6z
ALMOST FREE Coupon Code: FIRE gives £9 discount on £10 price – £1

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



My ‘Planguage’ Requirements Concepts <-CE book

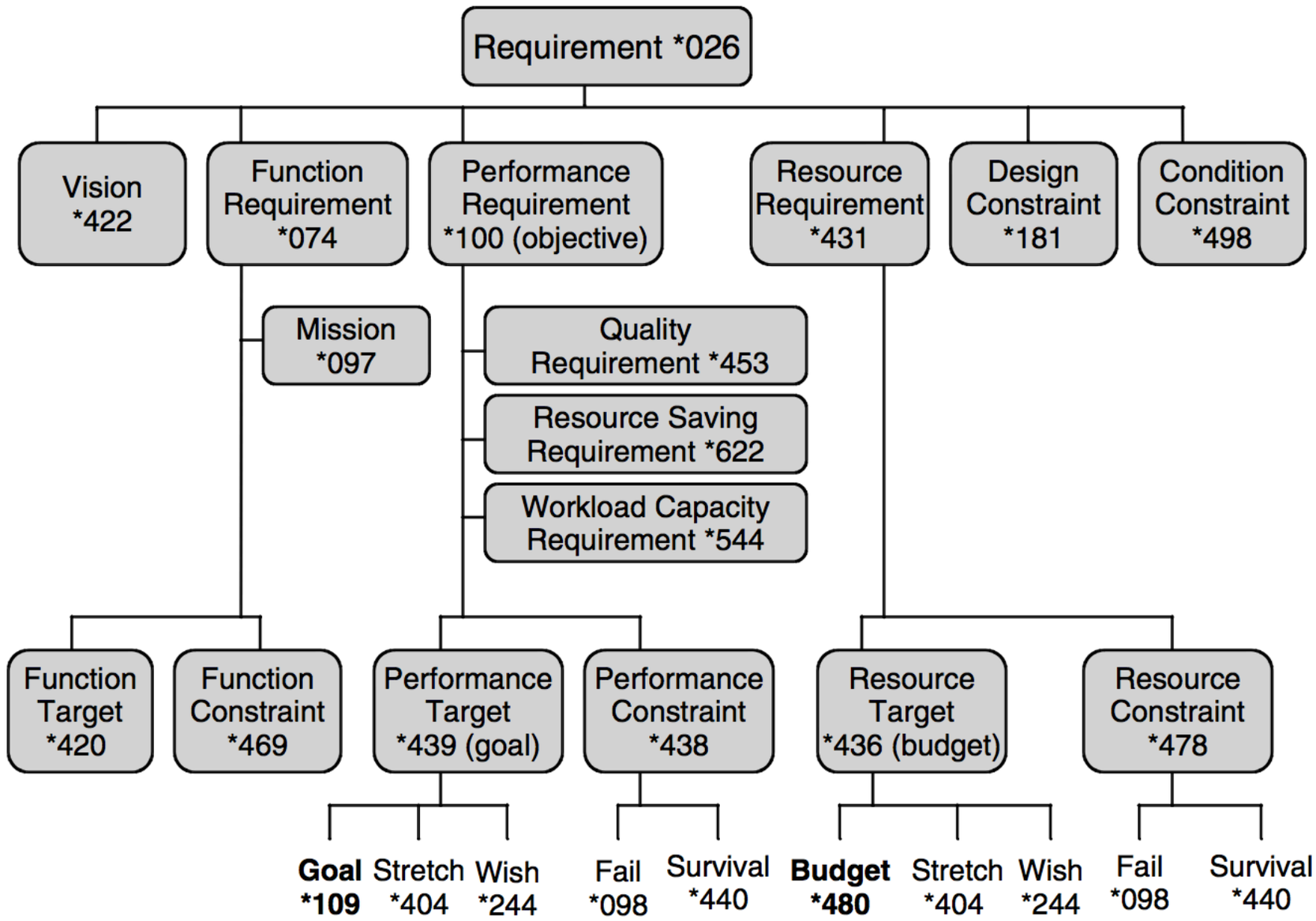


Figure G20