Agile - The Next Level 'Value Engineering'

Tom Gilb, tom@Gilb.com

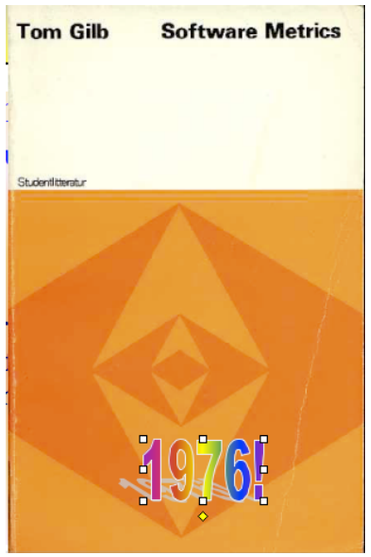
@ImTomGilb
These slides are at

GILB.COM DOWNLOADS LIBRARY

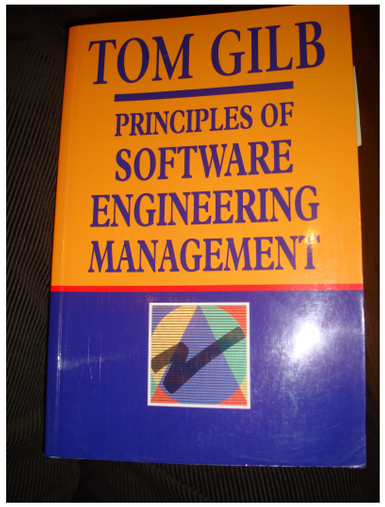
https://tinyurl.com/AGILEMEETUP18APRIL

Spring edition of the Masterclass
6 / 41

Agile Credibility

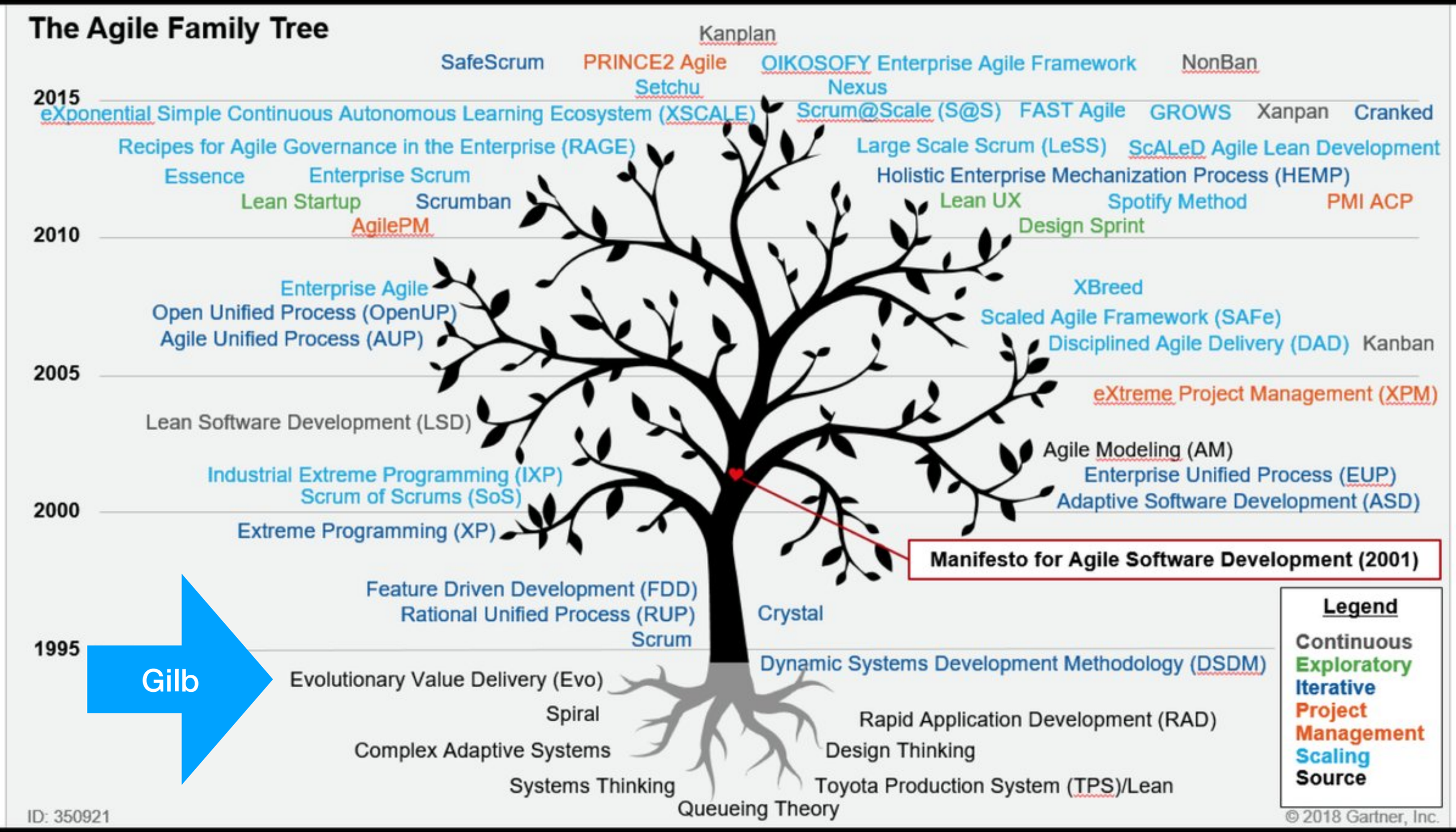


- Agile ‘Grandfather’ (Tom)
  - Practicing ‘Agile’ IT Projects since 1960 (Dobloug, Oslo)
  - Preaching Agile since 1970’s (Computer World column UK)
  - Acknowledged Pioneer by Agile Gurus and Research
    - See Presenters Notes to this slide for detail
- Agile Practice
  - IT: decades (Kai and Tom)
  - Organisations: Decades (
    - Citigroup, JP Morgan, Deutsche Bank, UBS, Intel, HP, Boeing)
- Books:
  - Principles of Software Engineering Management (1988)
  - Competitive Engineering (2005)
  - Evo: (Kai, evolving, 55 iterations)
  - Value Planning (2014-2019)
  - 5 Books in 2018 (see gilb.com):
    - LD, IC, 100 PPP, Technoscopes,Clear Communication



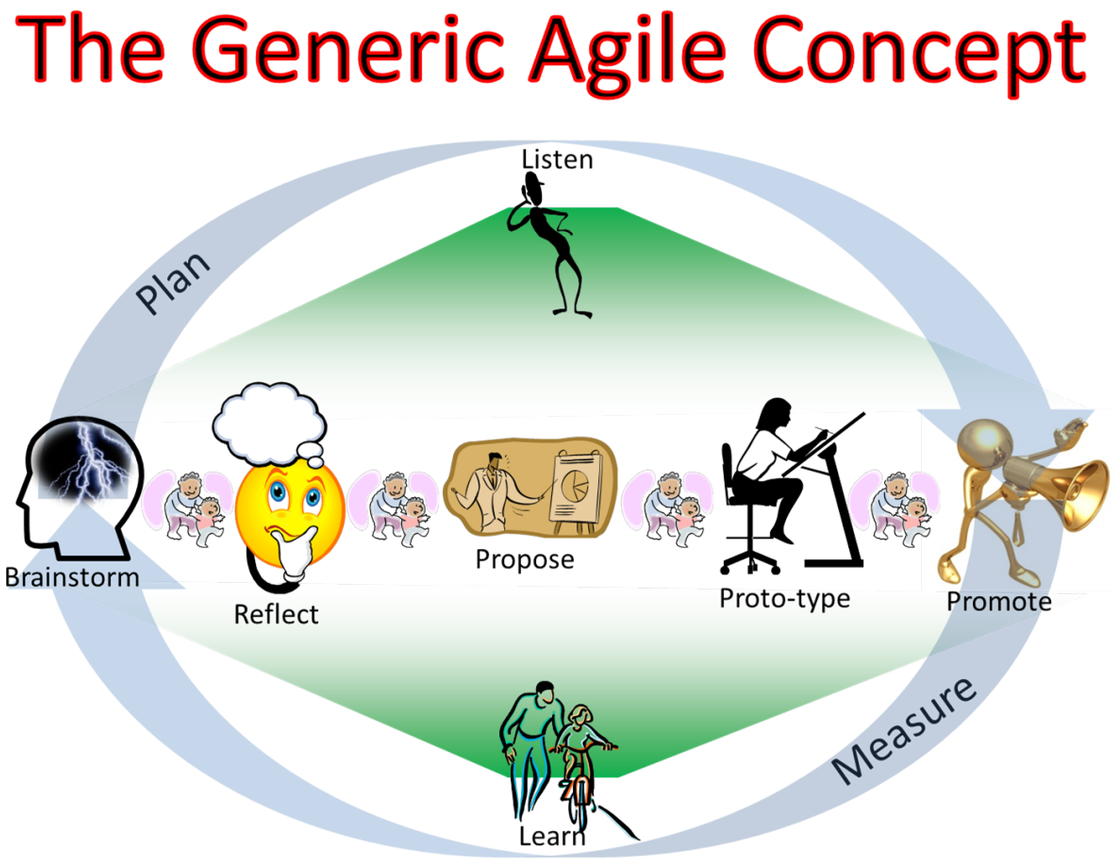
See this slide’s Presenter Notes for more detail, even 1976 SM book quotes)

© Gilb.com


The Agile Family Tree
Kanplan
SafeScrum
PRINCE2 Agile
OIKOSOFY Enterprise Agile Framework
NonBan
Setchu
Nexus
2015
eXponential Simple Continuous Autonomous Learning Ecosystem (XSCALE)
Scrum@Scale (S@S)
FAST Agile
GROWS
Xanpan
Cranked
Recipes for Agile Governance in the Enterprise (RAGE)
Large Scale Scrum (LeSS)
ScALeD Agile Lean Development
Essence
Enterprise Scrum
Holistic Enterprise Mechanization Process (HEMP)
Lean Startup
Scrumban
Lean UX
Spotify Method
PMI ACP
AgilePM
Design Sprint
2010
Enterprise Agile
XBreed
Open Unified Process (OpenUP)
Scaled Agile Framework (SAFe)
Agile Unified Process (AUP)
Disciplined Agile Delivery (DAD)
Kanban
2005
eXtreme Project Management (XPM)
Lean Software Development (LSD)
Agile Modeling (AM)
Industrial Extreme Programming (IXP)
Enterprise Unified Process (EUP)
Scrum of Scrums (SoS)
Adaptive Software Development (ASD)
2000
Extreme Programming (XP)
Manifesto for Agile Software Development (2001)
Feature Driven Development (FDD)
Rational Unified Process (RUP)
Crystal
Scrum
1995
Gilb
Dynamic Systems Development Methodology (DSDM)
Evolutionary Value Delivery (Evo)
Spiral
Rapid Application Development (RAD)
Complex Adaptive Systems
Design Thinking
Systems Thinking
Toyota Production System (TPS)/Lean
Queueing Theory
Legend
Continuous
Exploratory
Iterative
Project Management
Scaling
Source
ID: 350921
© 2018 Gartner, Inc.

Defining 'Agile'

- "Any set of tactics that **enable a prioritised stream of useful results**, *in spite of a changing environment*"
  - TsG 7 June 2013
- A focus on 'Agile', is the *wrong* level of focus.
  - Using agile tactics that 'work', is a good idea.
- Focus on *results*, no matter what.
- As Francis Maude emphasizes "Value for Money"



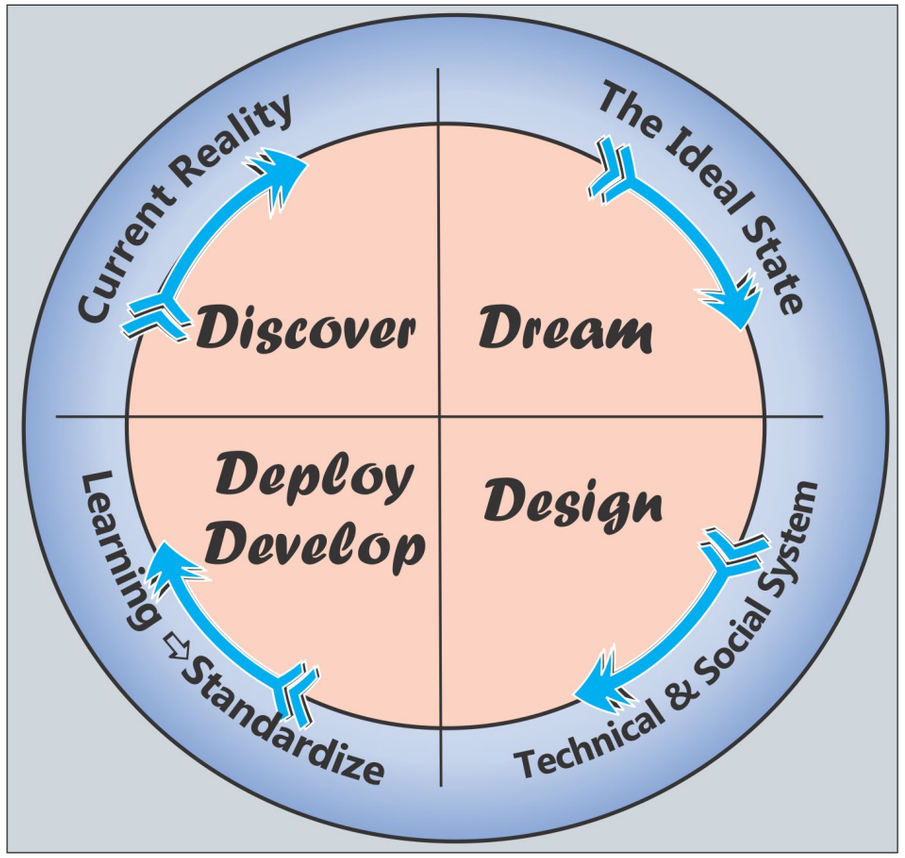
Advanced Agile Options

Specific Technologies to really deliver an agile measurable value stream to your stakeholders.

1. Quantifying all critical stakeholder values, incl. all qualities. A Scale of measure for all critical variables.
2. Estimation and measurement of all architectures and strategies. The Value Table. 3. Decomposition of architecture and strategies into sprint-sized value deliveries. Principles and Methods.

4. Advanced sprint prioritization strategies: based on values, costs and risks: computed.
5. Dynamic Design to Cost: the agility to adjust architecture in order to deliver all values on time, under budget every time. Zero Failures.
6. Specification QC: how to measure conformance to best practices for requirements, contracts and architectures.

These techniques are scale-free methods for dealing with large-scale and complex systems development. They apply to any type of systems. They are not a way to manage the 'coding process'. They are a way to manage the delivery of real organizational value to real organisations such as government, industry, innovation, and finance. You will not become a Master in 2 days, but you will succeed in your projects when others fail. Right now our IT and agile failure rate is an international shame.

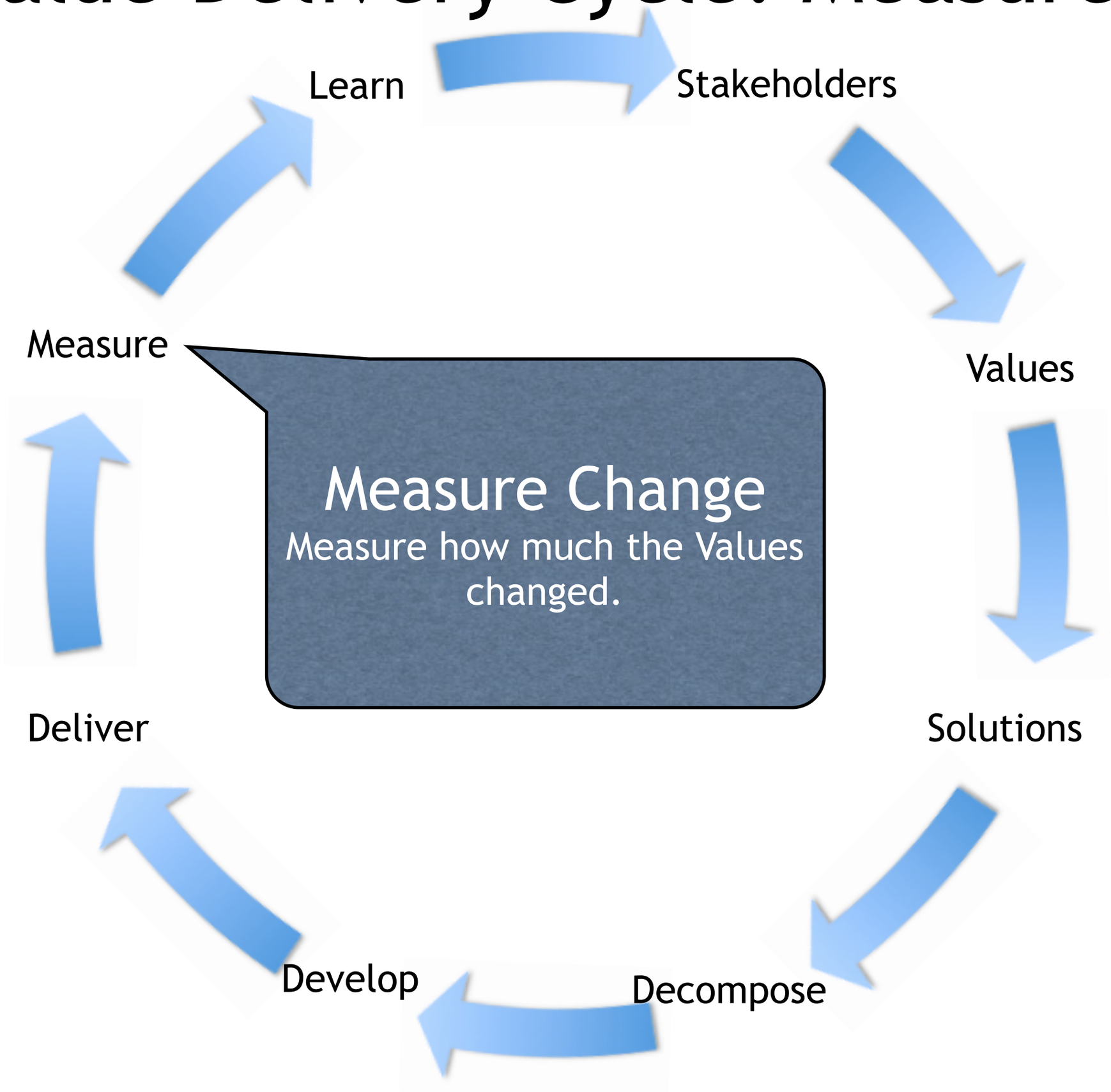


Agile as practiced today is perhaps good for delivering code functions faster.

But the main point of our projects is to deliver critical factor improvements.

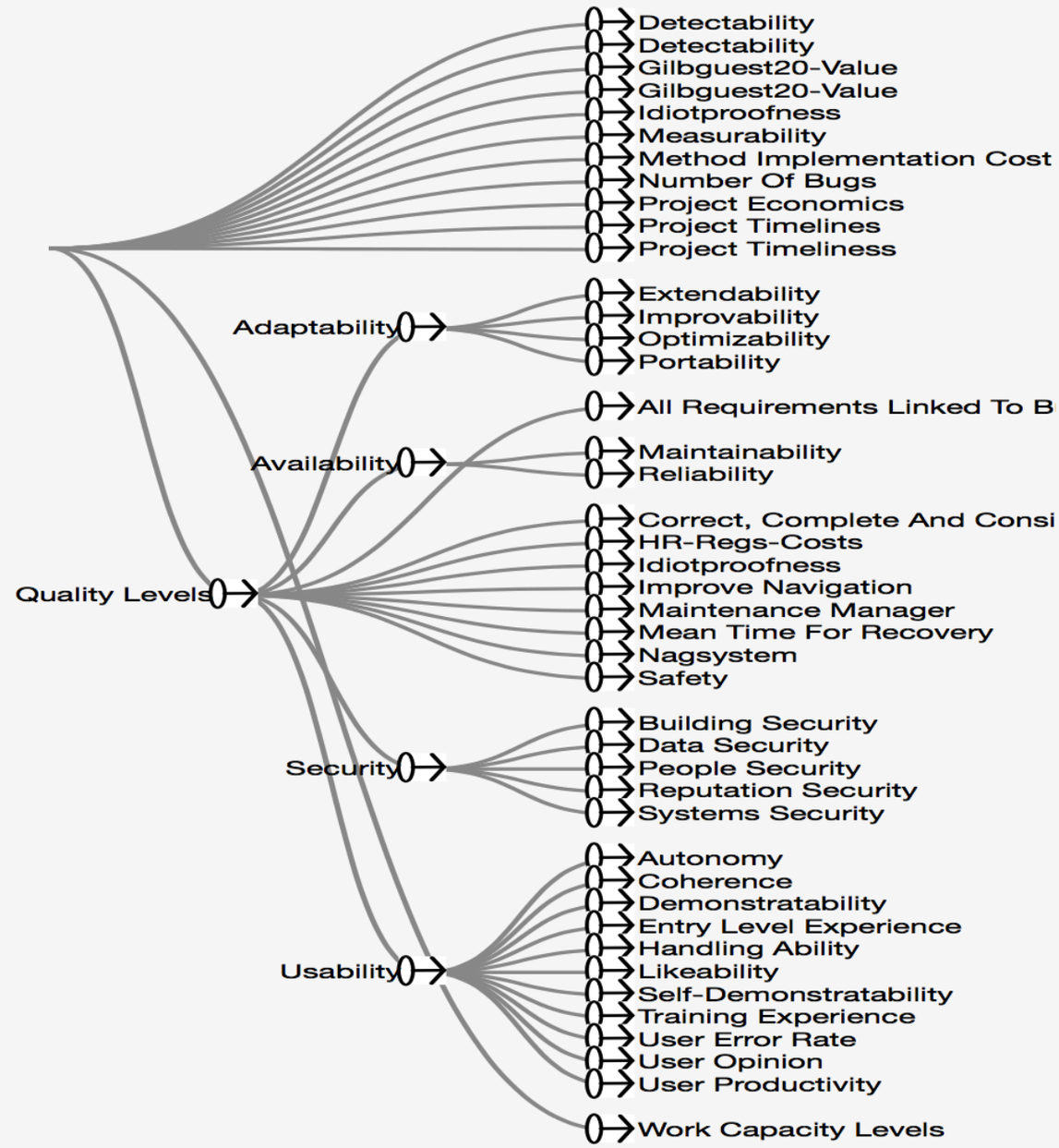
Not code!

Value Delivery Cycle: Measure



Copyright: Kai@Gilb.com

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;

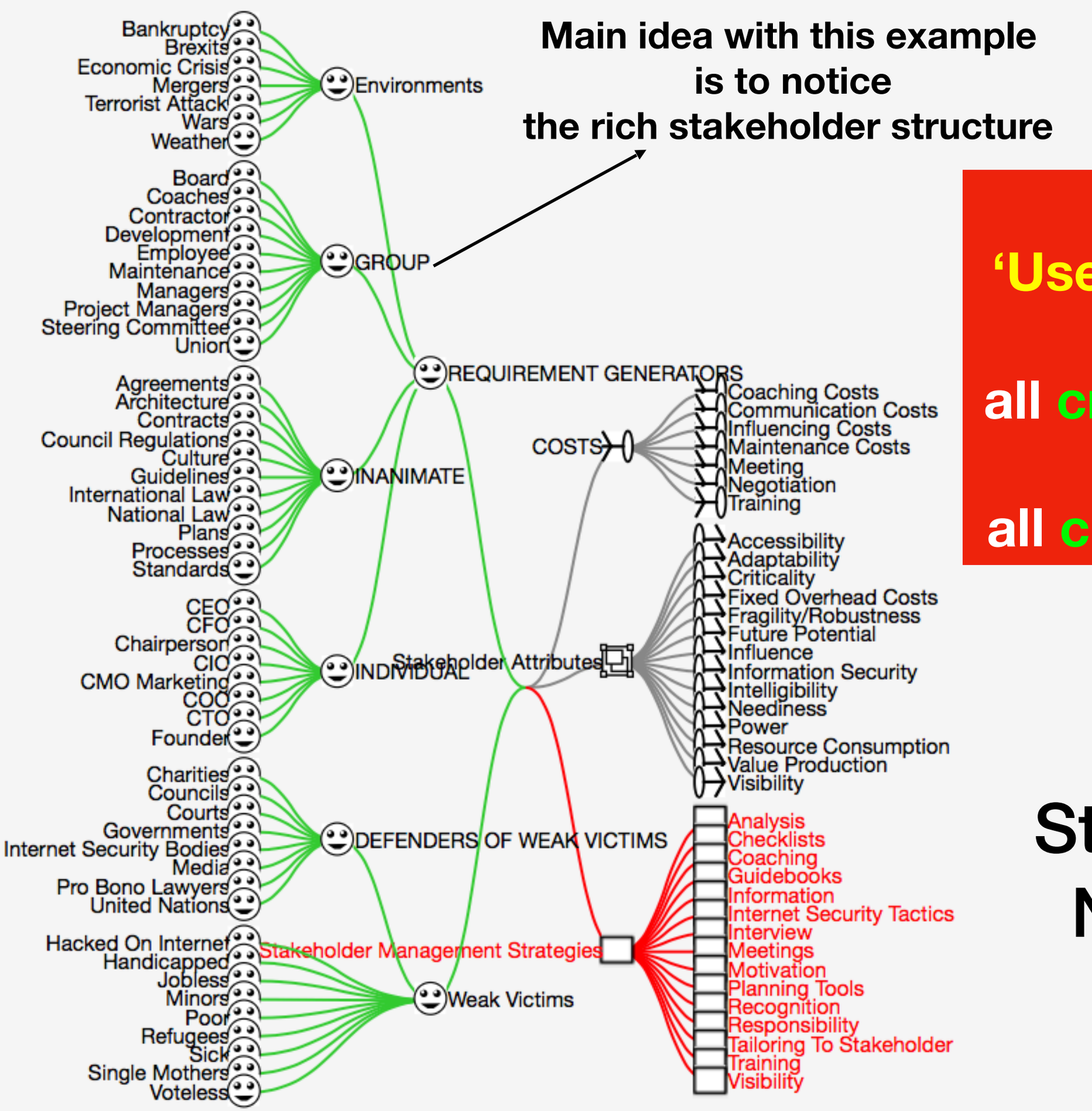
but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind;

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

© Gilb.com

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

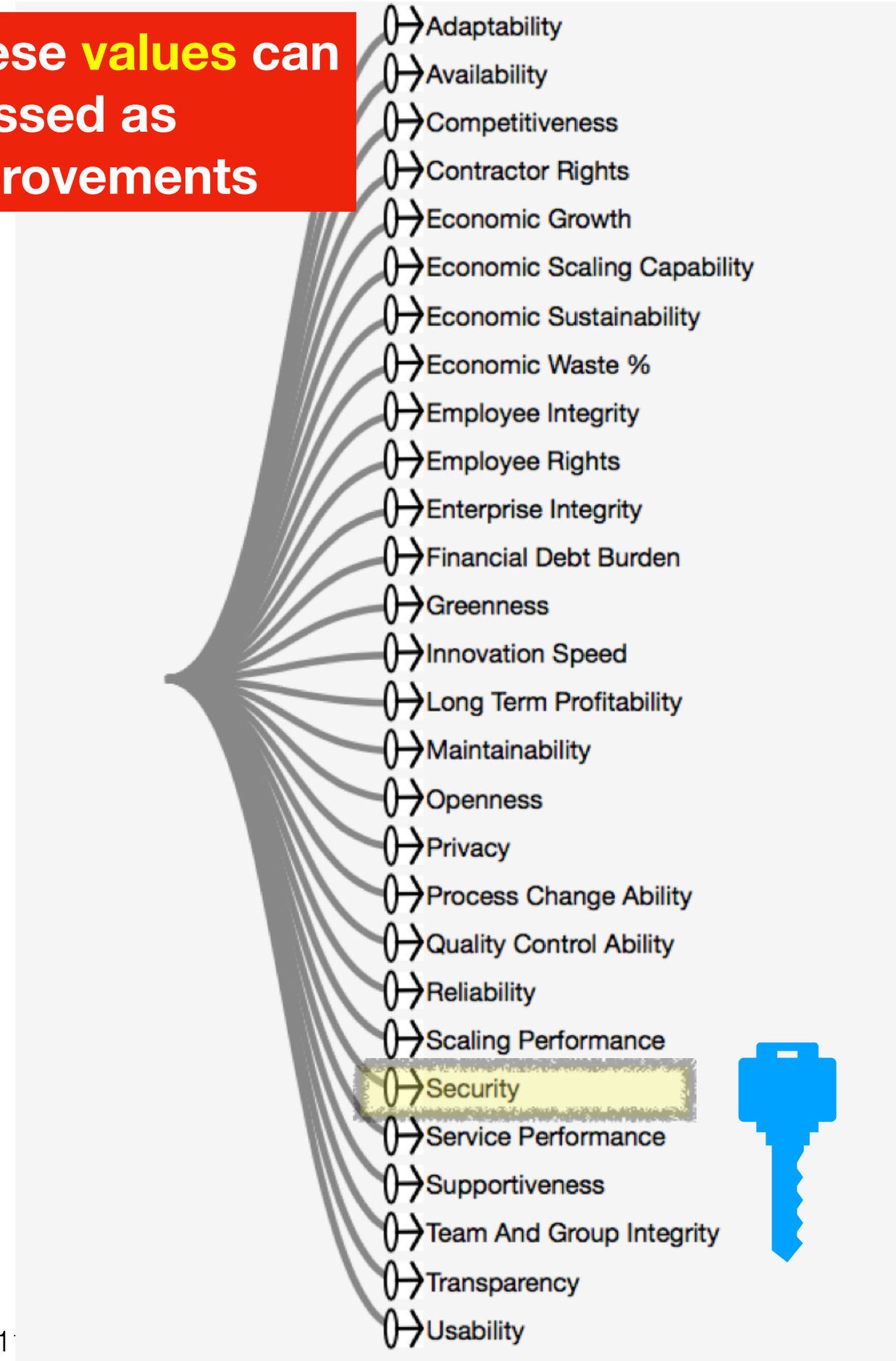


Not limited to
'Users and Customers'
but including
all critical requirements
from
all critical stakeholders

Stakeholders Needs and Means diagram

Every one of these values can be expressed as numeric improvements

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



Security Value Quantification with Stakeholders

National Security

All values and qualities can be expressed quantitatively

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	=: **8** ± 0 Δ: -2 % Δ%: **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 Δ: 0 % Injury Δ%: **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 Δ: -5 minutes Δ%: **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%: **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 Δ: 500k $ Δ%: **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%: **17** ± 0 % Σ?%: *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

When we quantify our critical 'values' we can take the next step of 'estimating and then tracking movement towards those value levels'

— Confucius, *Sayings of Confucius*

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

— Confucius, *Sayings of Confucius*

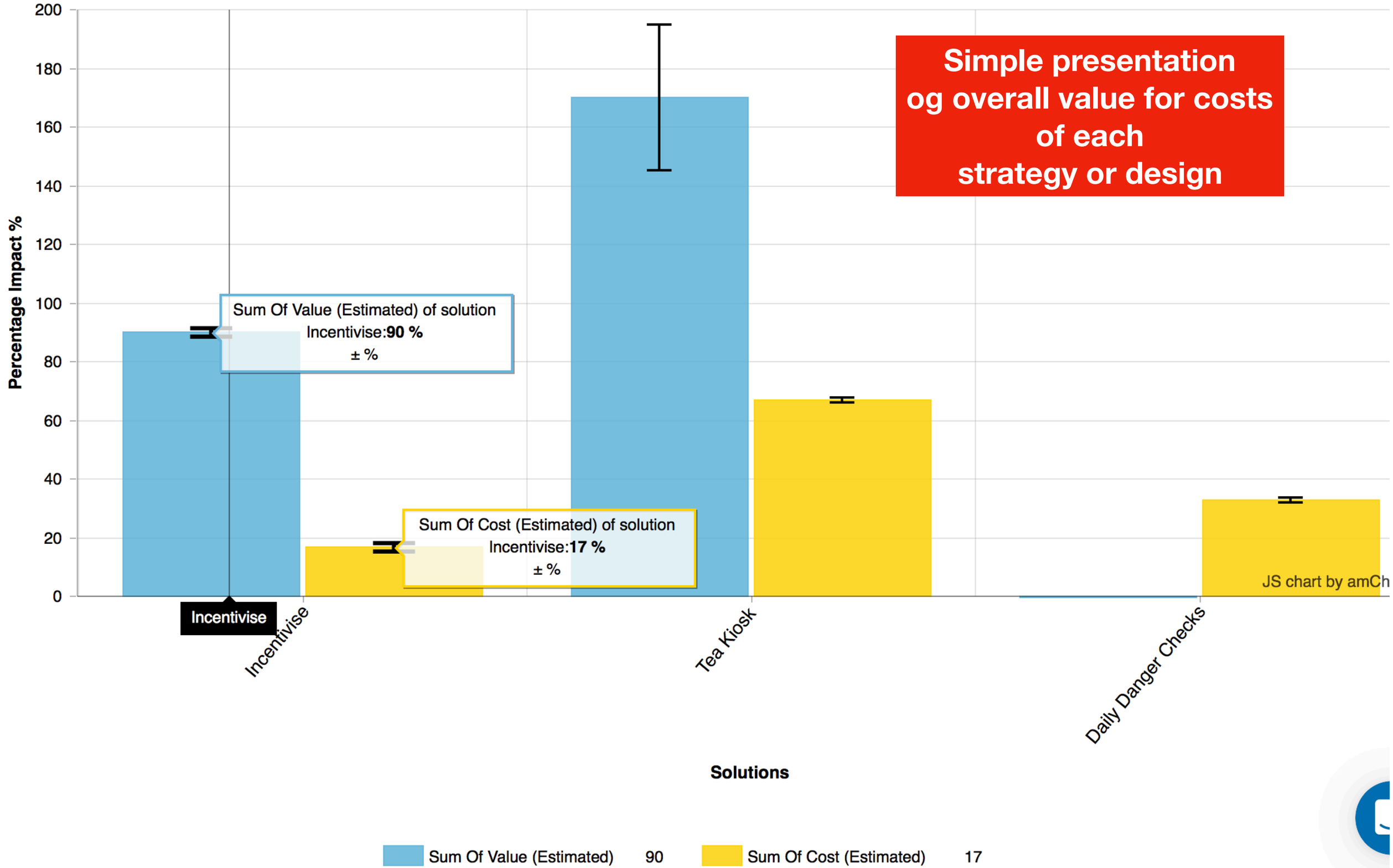
What intellectual tools do you have that will help you to be more conscious of exactly what you do NOT know enough about?

Requirements	Incentivise	Tea Kiosk	Daily Danger Checks	Sum
Project Timeliness Status: **10** → Wish: **5** % % time overrun necessary to deliv... Δ%: [Project Cost Size = { Medium ($10k -...] ?%: 30th June 2017	=: **8** ± 0 Δ: -2 % Δ%: **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... Δ%: [Emergency Types = { Earthquake }, ?%: 30th June 2018	=: **50** ± 0 Δ: 0 % Injury Δ%: **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... Δ%: [user = { adult }, task = { dri...] ?%: 30th June 2017	=: **10** ± 0 Δ: -5 minutes Δ%: **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: Σ%: Credibility - adjusted: 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	=: **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: Σ%: Credibility - adjusted: Σ?%:	**17** ± 0 % *34 %*	**67** ± 0 % *134 %*	**33** ± 0 % *66 %*	
Value To Cost:	5.30	2.50	-1.50	

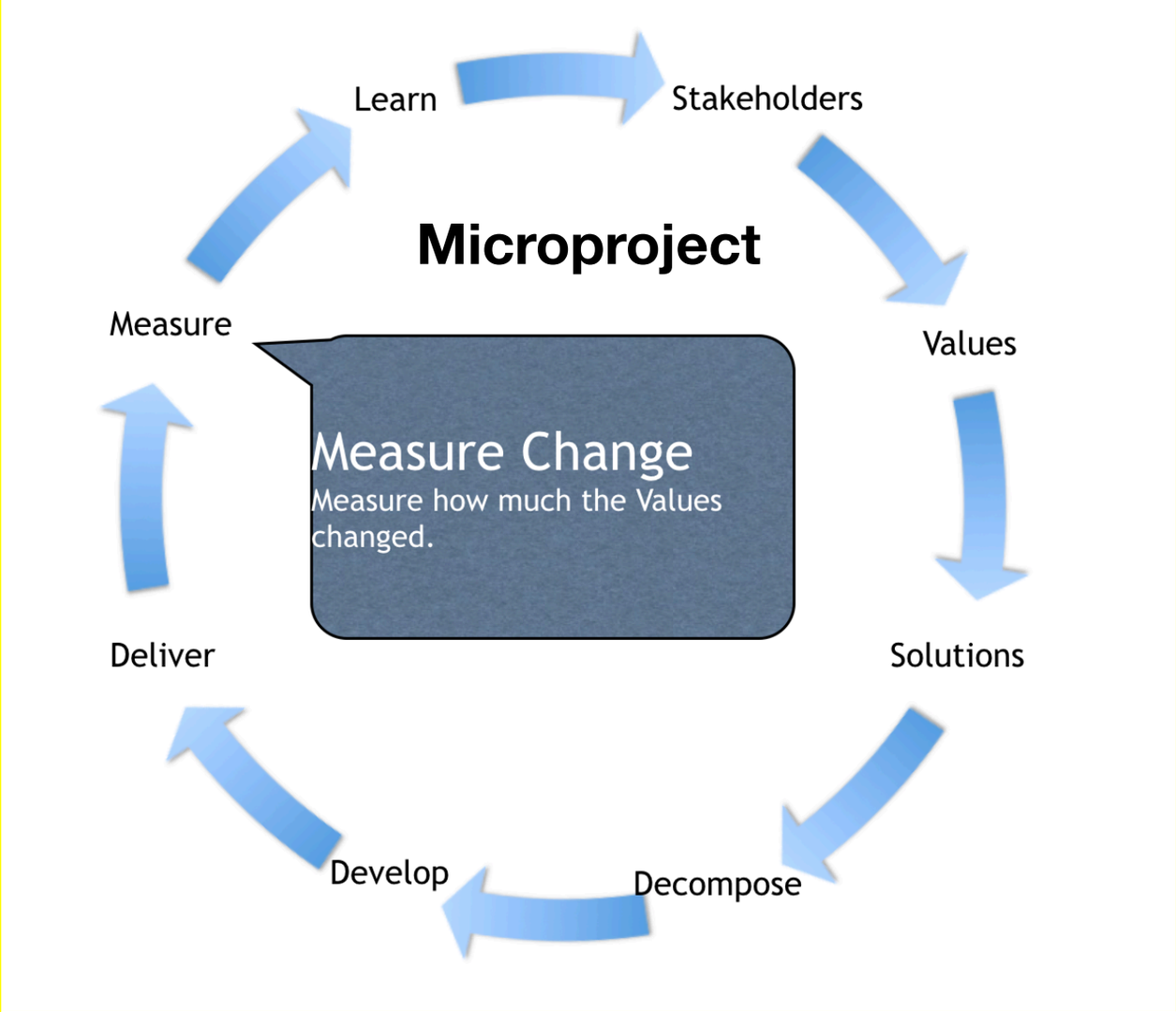
The numeric relation between ends and means.

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

Basic Structure of an Impact Estimation Table



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 each step is **about being 'numeric'**
('Engineering' not 'coding')
This is 'Evo' (Evolutionary Value Optimization)

Stakeholders

Learn

Values

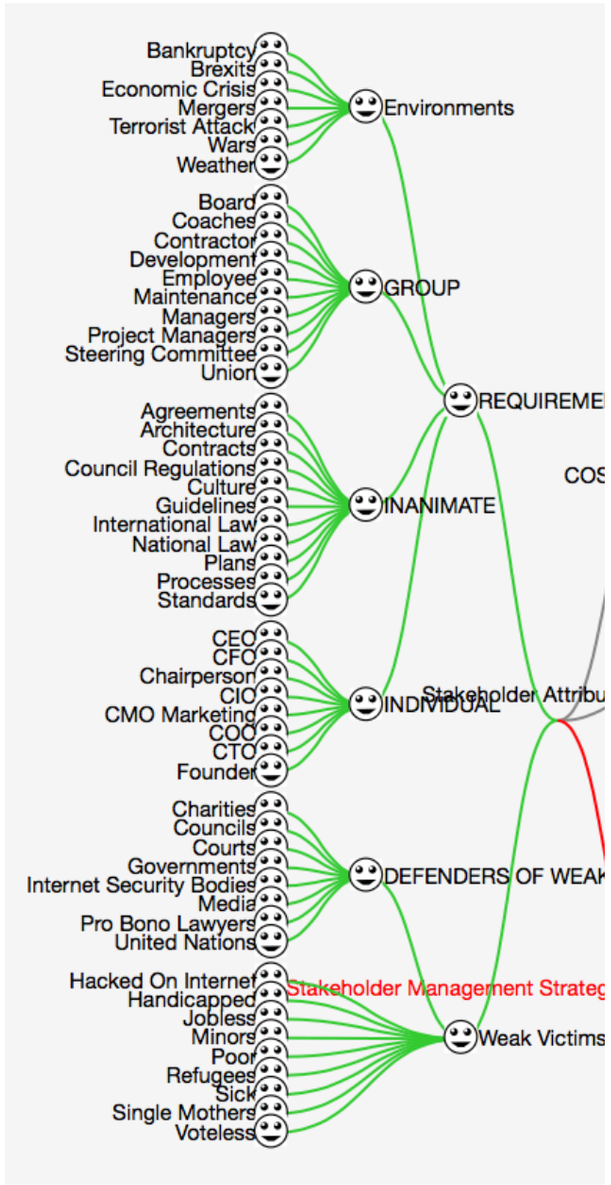
Solutions

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

Requirement Sources

Stakeholder Cases
Stakeholder Stories

What critical numeric improvements do stakeholders need?

**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**

Solutions (designs, architectures, strategies)

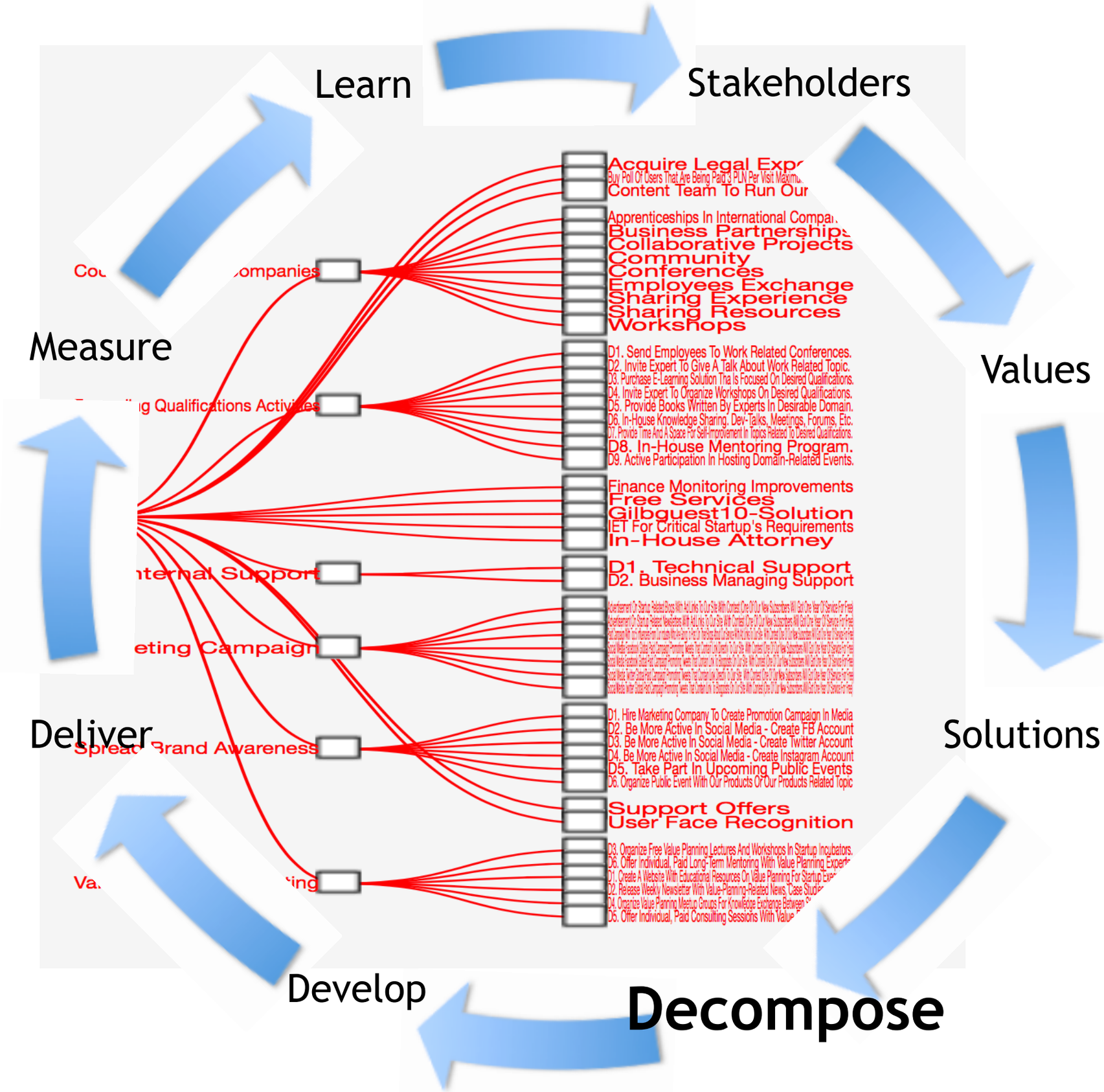
must be identified

and their total impacts on critical objectives and constraints

must be estimated reasonably

(order of magnitude)

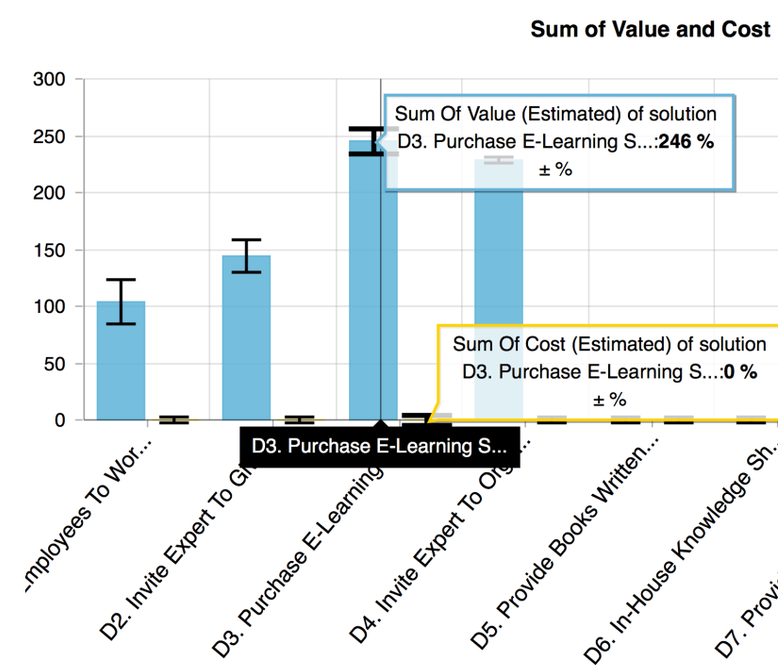
Value Decision Tables (Planguage) are a tool for doing estimates of potential solutions and how good they might be

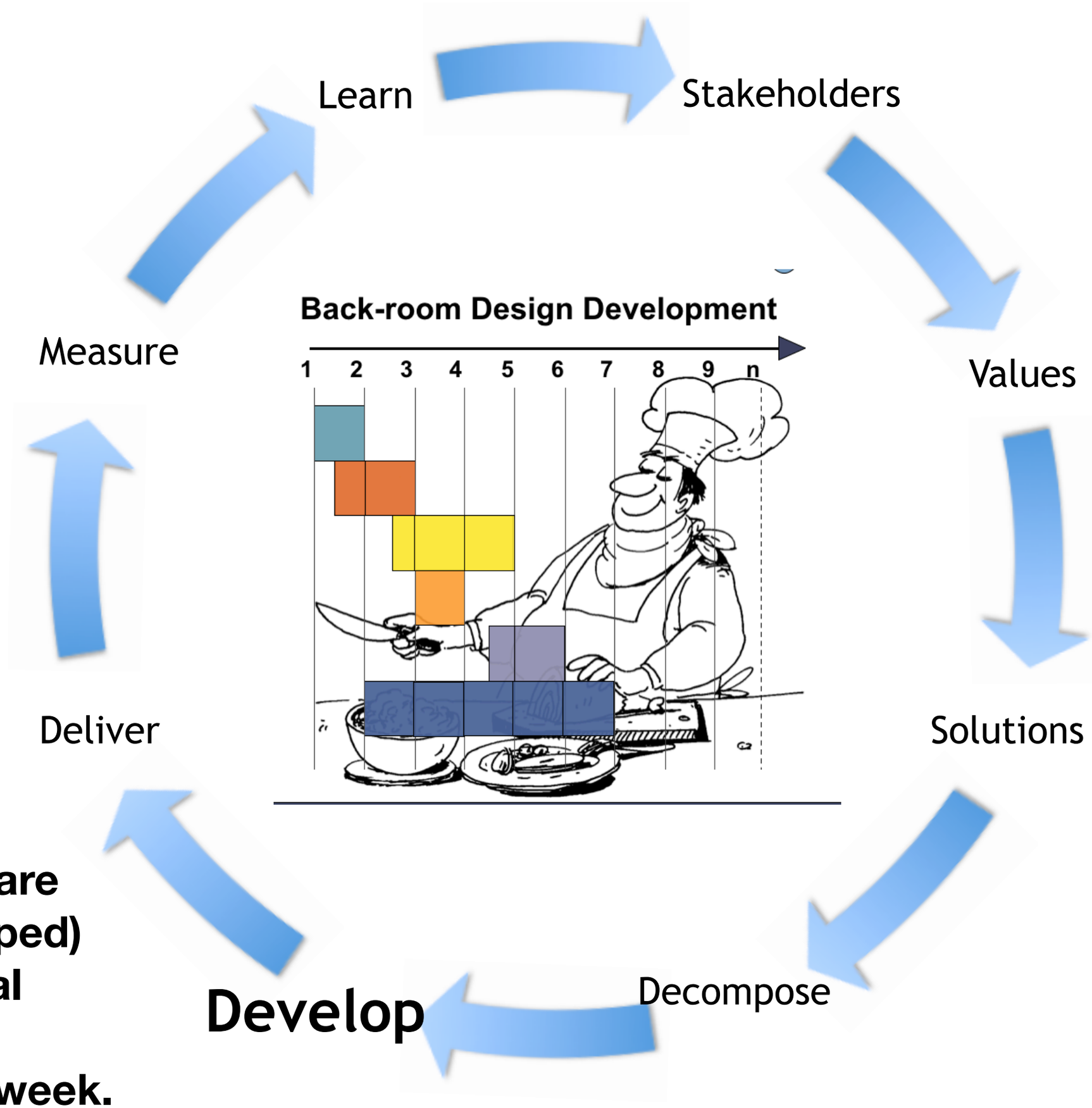


The solutions can be <u>decomposed</u> by 10x or 100x

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

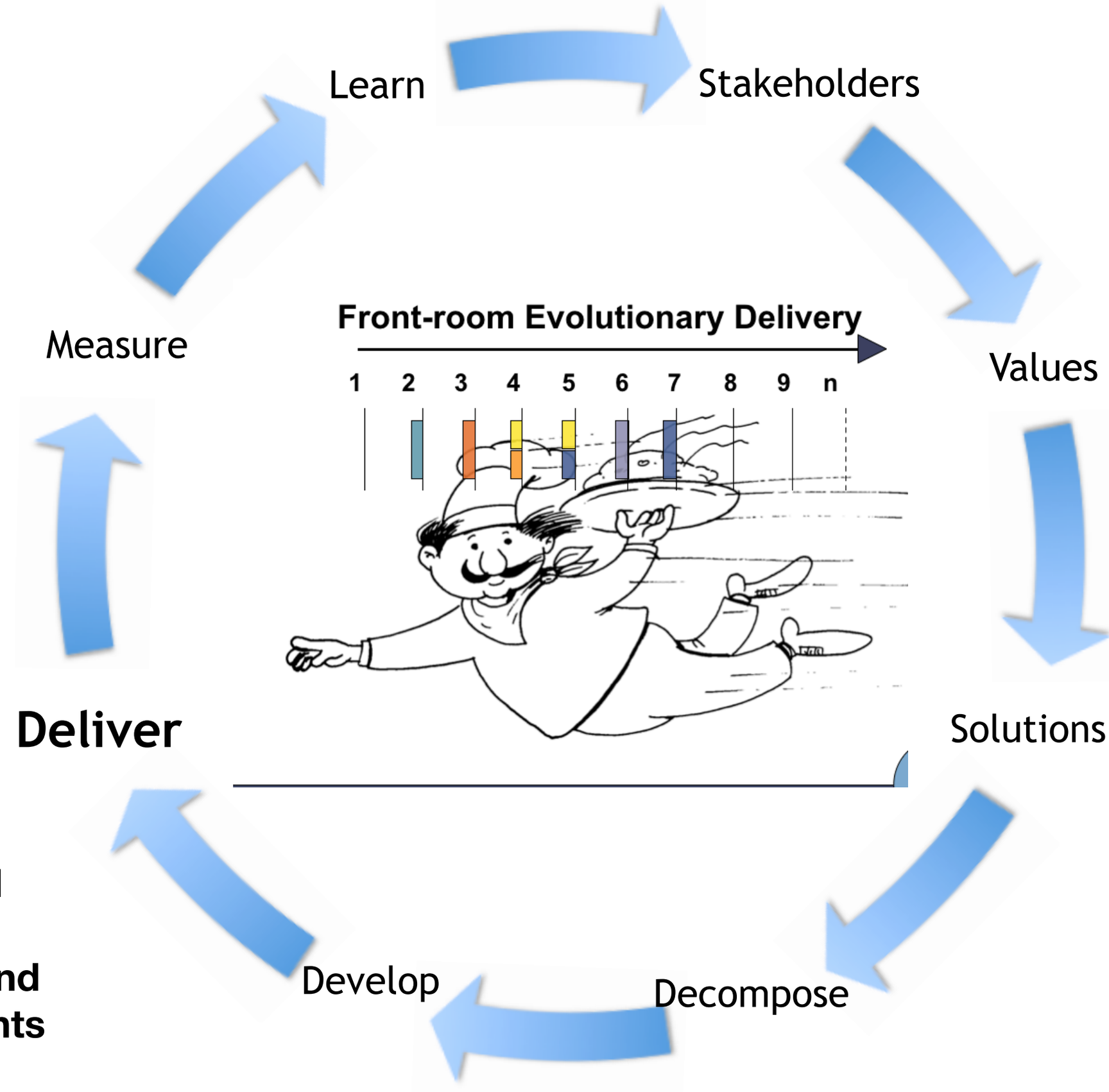
so as to prioritize the best value/cost for short term delivery





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,

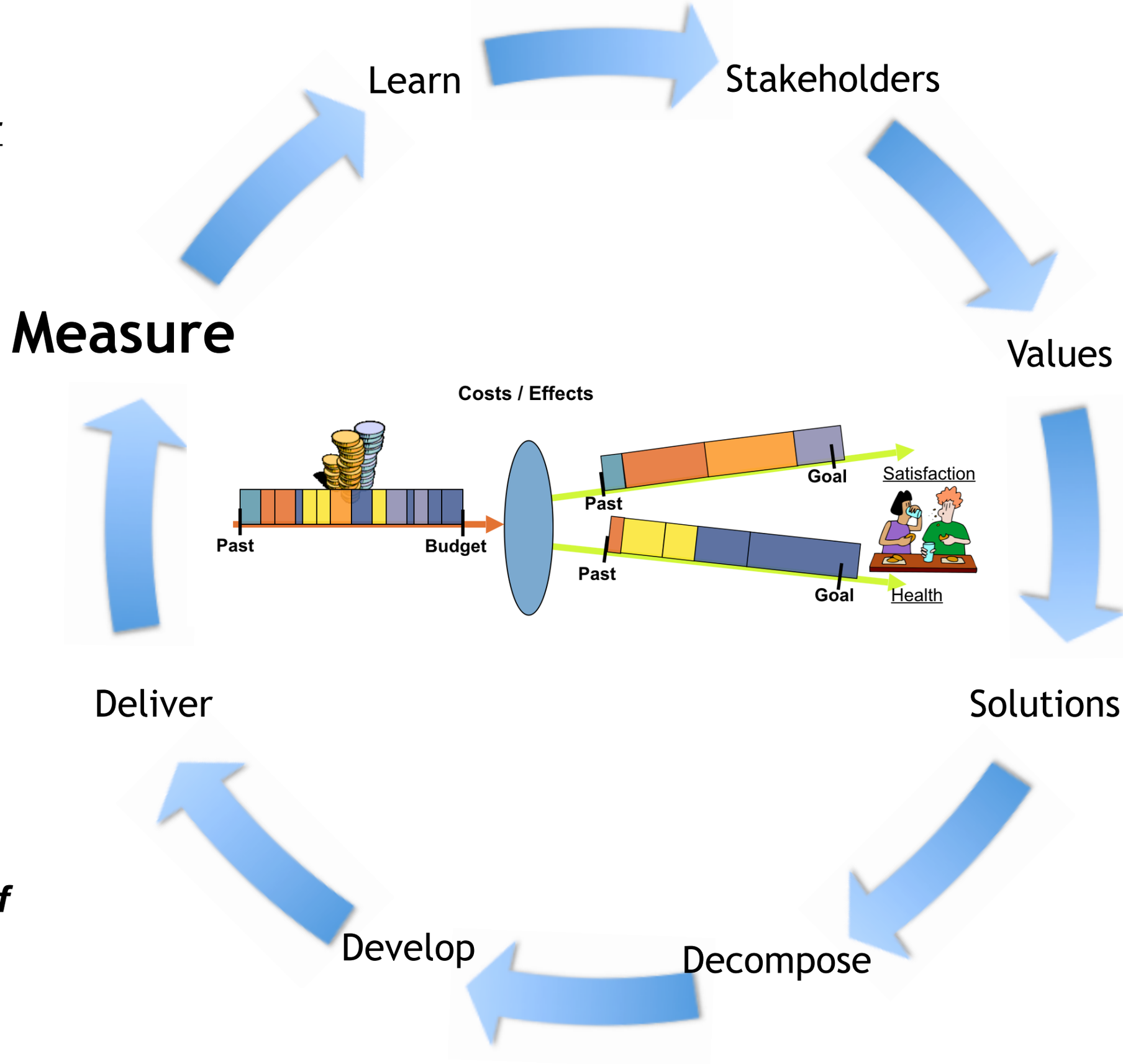
and

on their critical cost increments,

with a view to

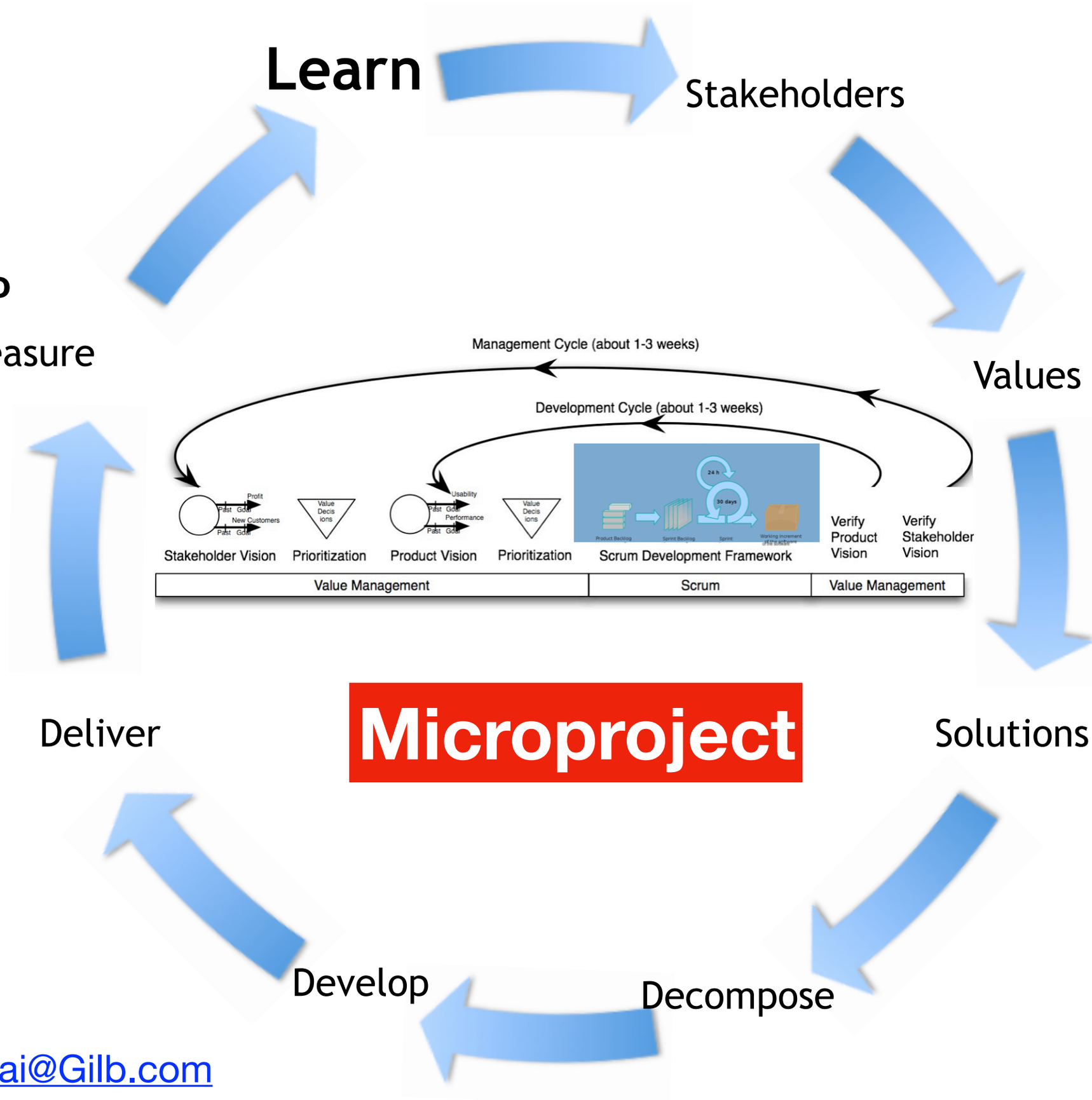
improving prediction of

final cumulative costs

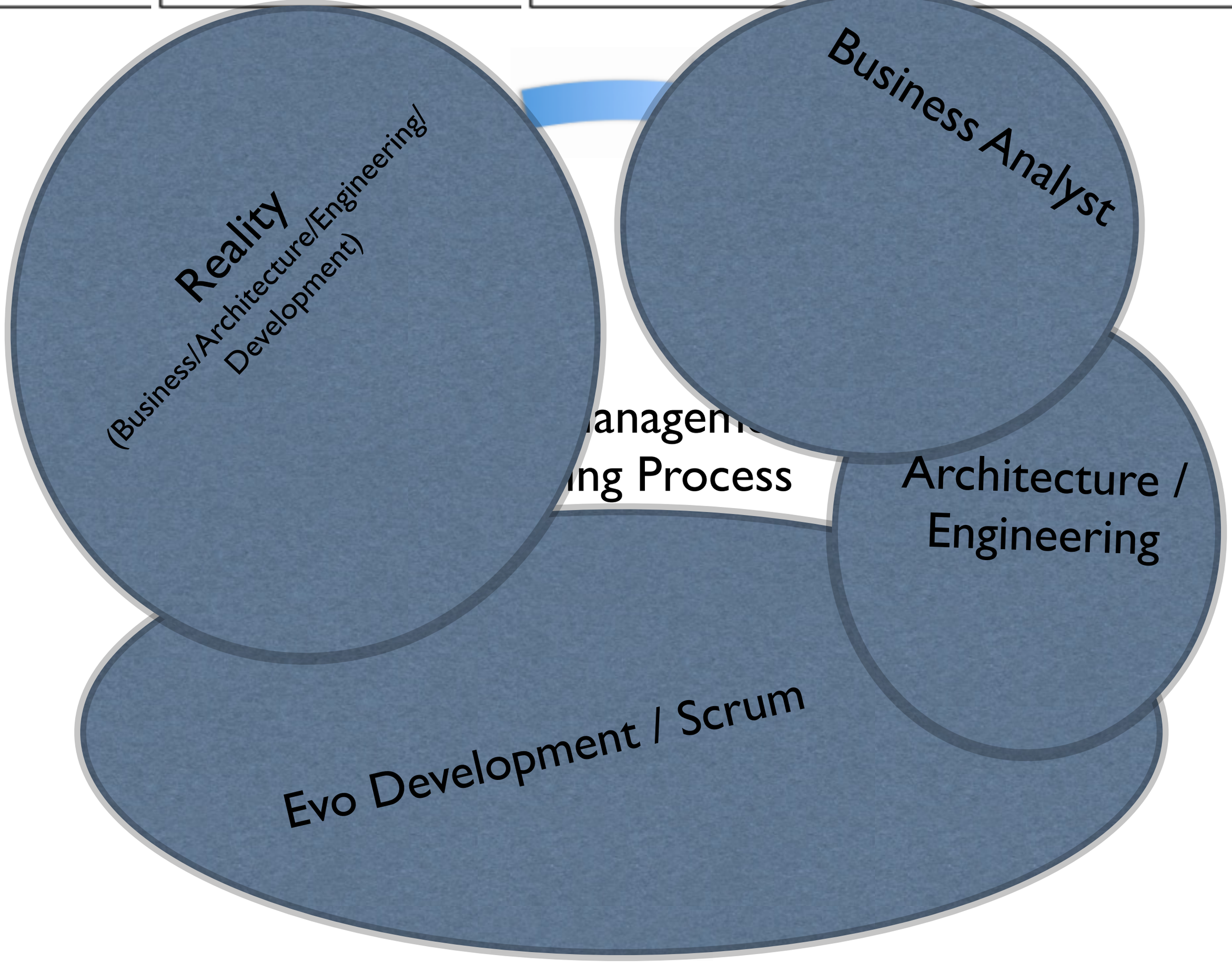


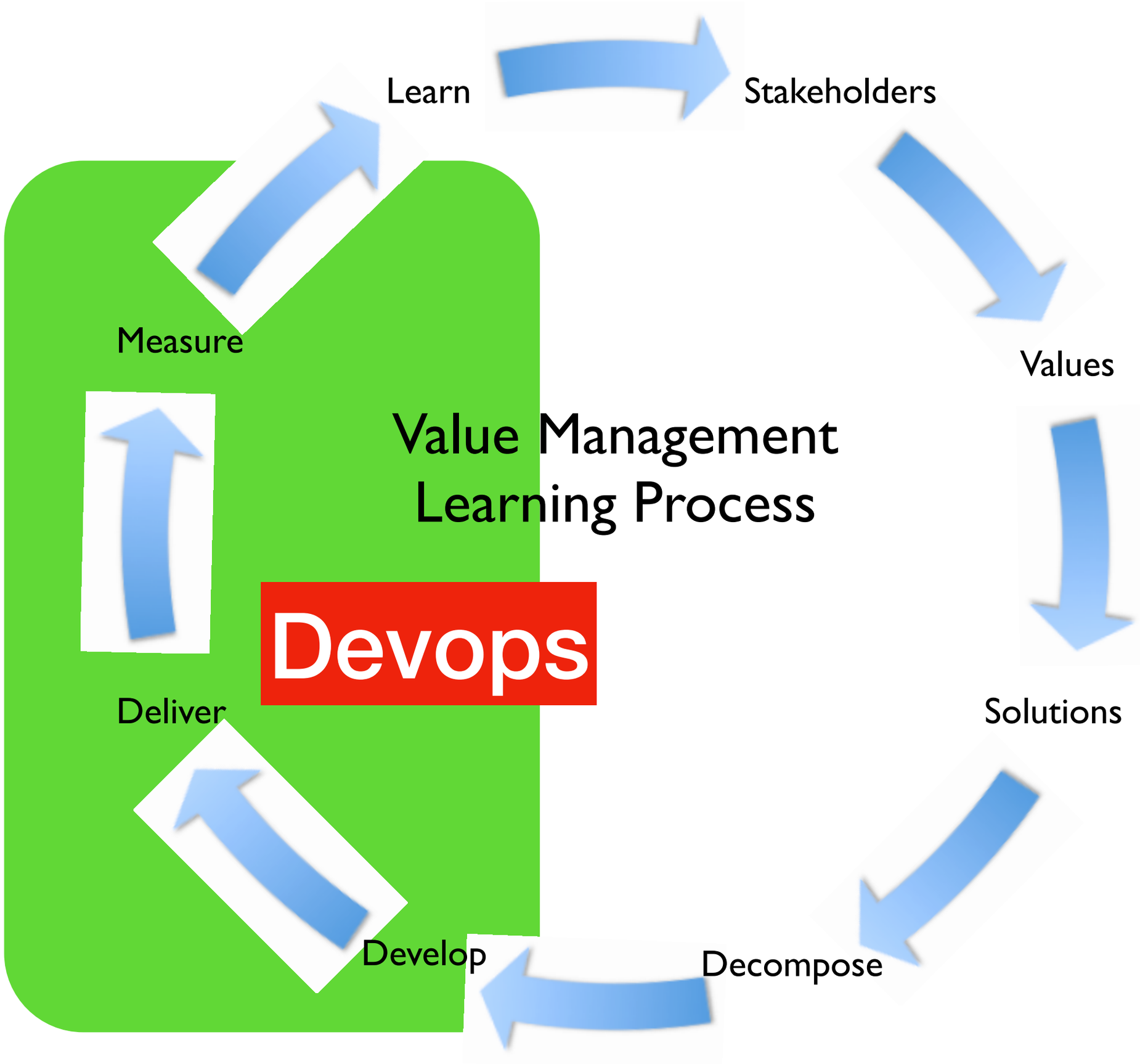
From the measurements,
and
other feedback
from stakeholders

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



These 2 diagrams are © kai@Gilb.com 2017, as well as several other illustrations used in this talk


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


Learn
Stakeholders
Values
Solutions
Decompose
Develop
Deliver
Measure
Value Management
Learning Process
Devops

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

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

Copyright Tom@Gilb.com 2013

Quality is designed in, not tested in
Our ‘Spec QC = ‘Inspection’)

“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

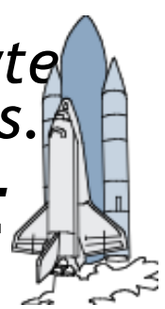
inIBM sj 4 80 p.419
In

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’ (Google it!), developed by IBM’s Harlan Mills (1970-1980) they reported:

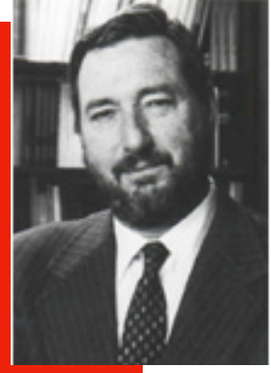
- *“Software Engineering began to emerge in FSD” (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) “some ten years ago [Ed. about 1970] in a continuing evolution that is still underway:*
- *Ten years ago general management expected the worst from software projects – cost overruns, 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 byte 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.”***

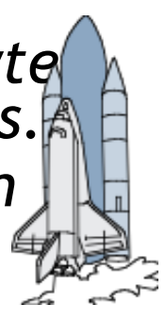


In the Cleanroom Method, developed by IBM's Harlan Mills (1970-1980) they reported:

(this is 'Agile' as it *should* be!)



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



in 45 incremental deliveries

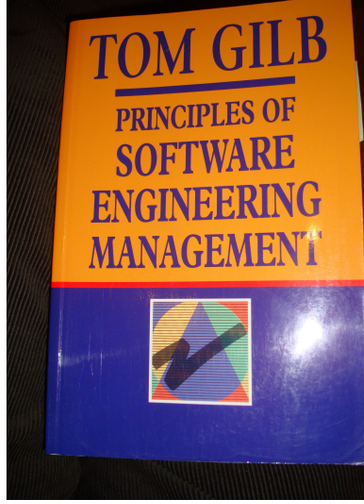
were few late or overrun deliveries in that decade, and none at all in the past four years

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.

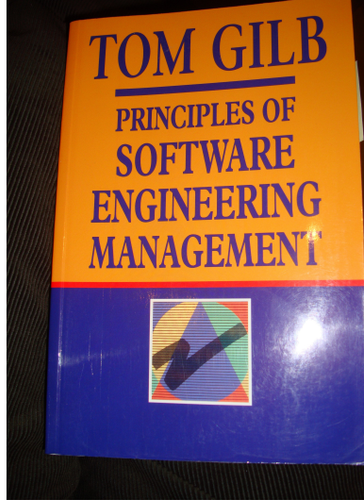
'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

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

Copyright Tom@Gilb.com 2017

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 s t of each increment can proceed concurrently with the

'Design is an iterativ

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.

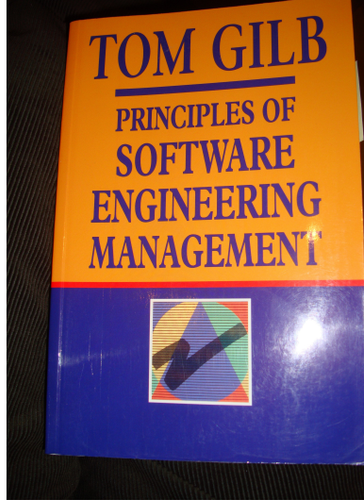
'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

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.'

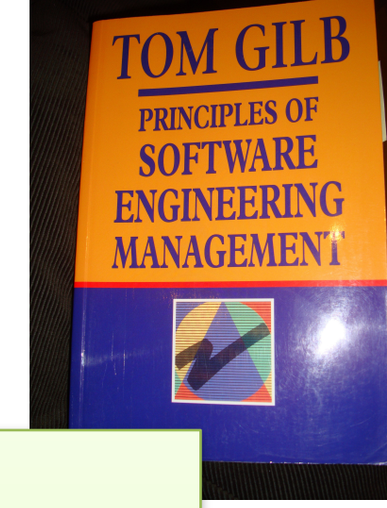
'Design is an iterative

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. Quin 1980, pp. 466~77
This text is cut from G

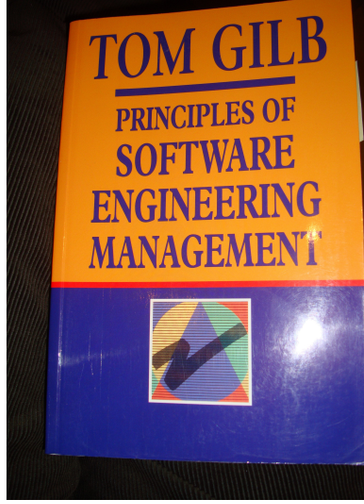
iteration process trying to meet cost targets by either *redesign* or by *sacrificing* 'planned capability'

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Design is an iterative process

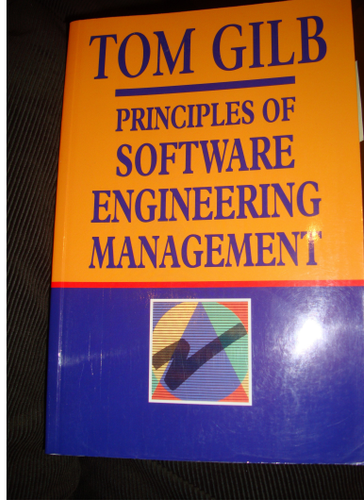
Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



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

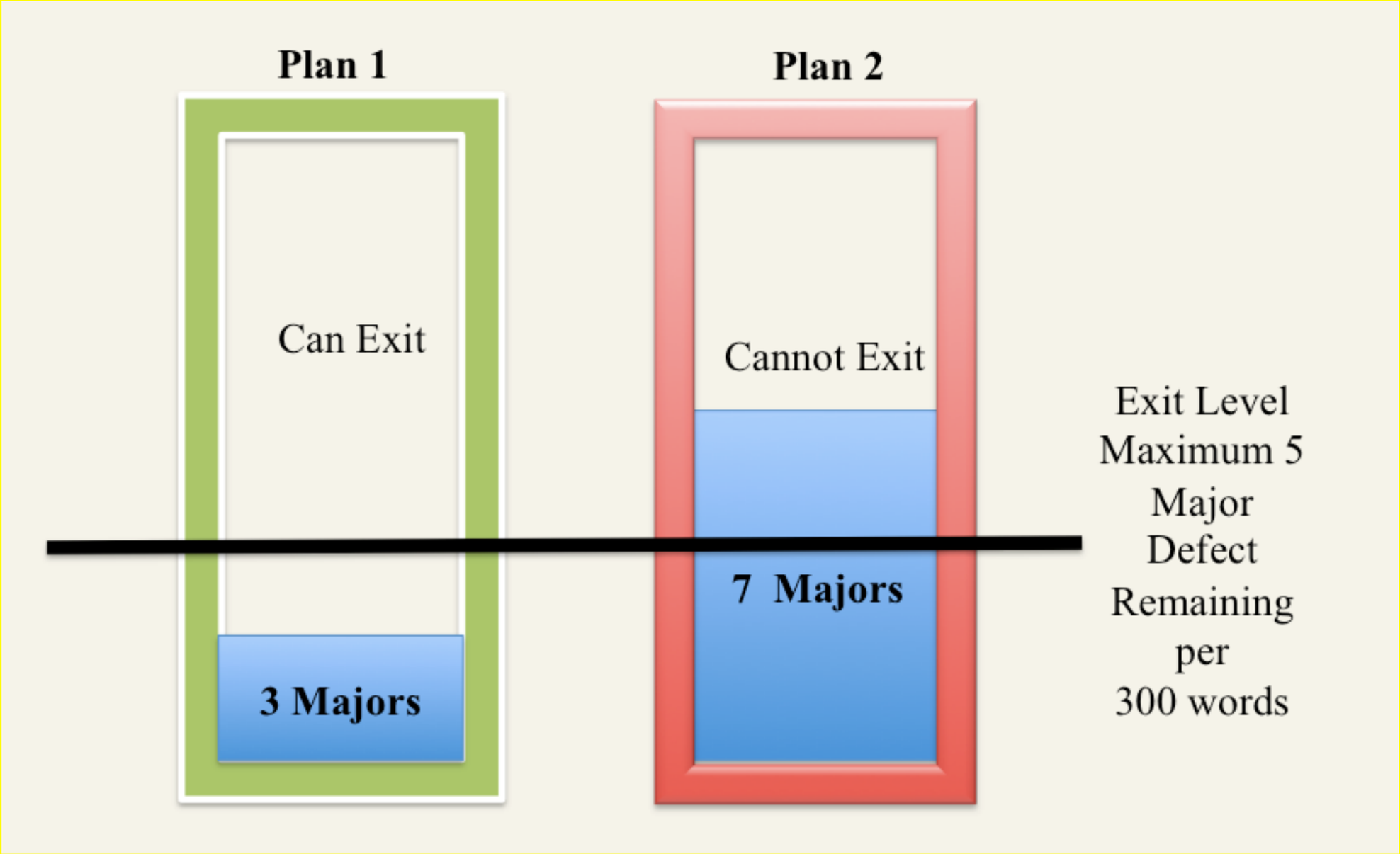
but they iterate through a series of increments,
thus *reducing the complexity of the task*,
and *increasing the probability of learning from experience*

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.

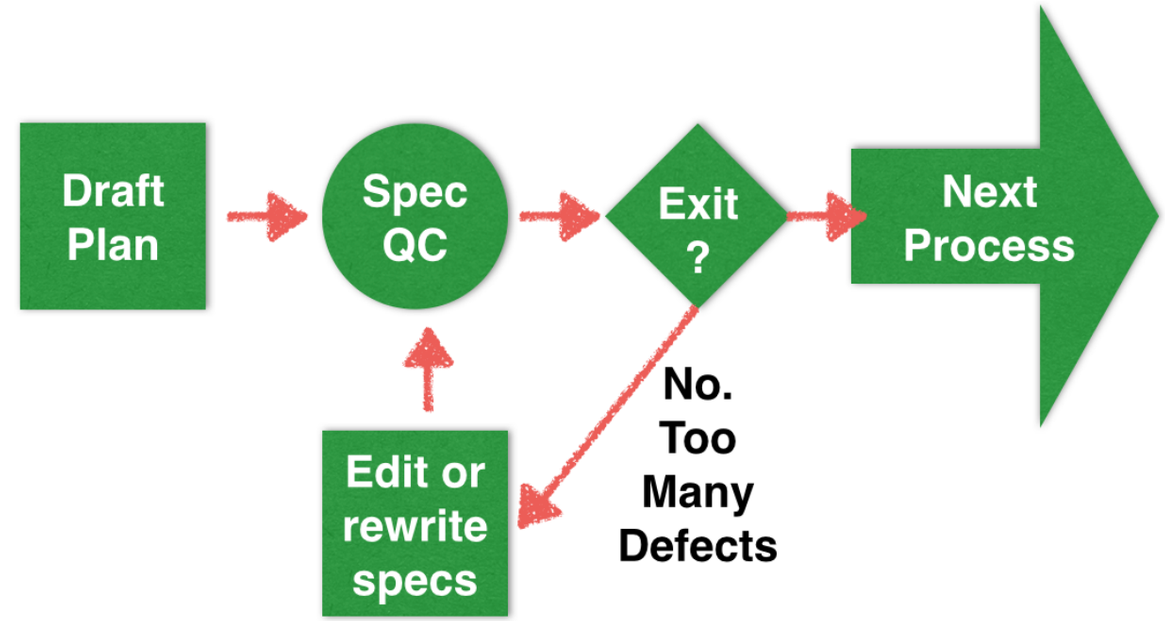


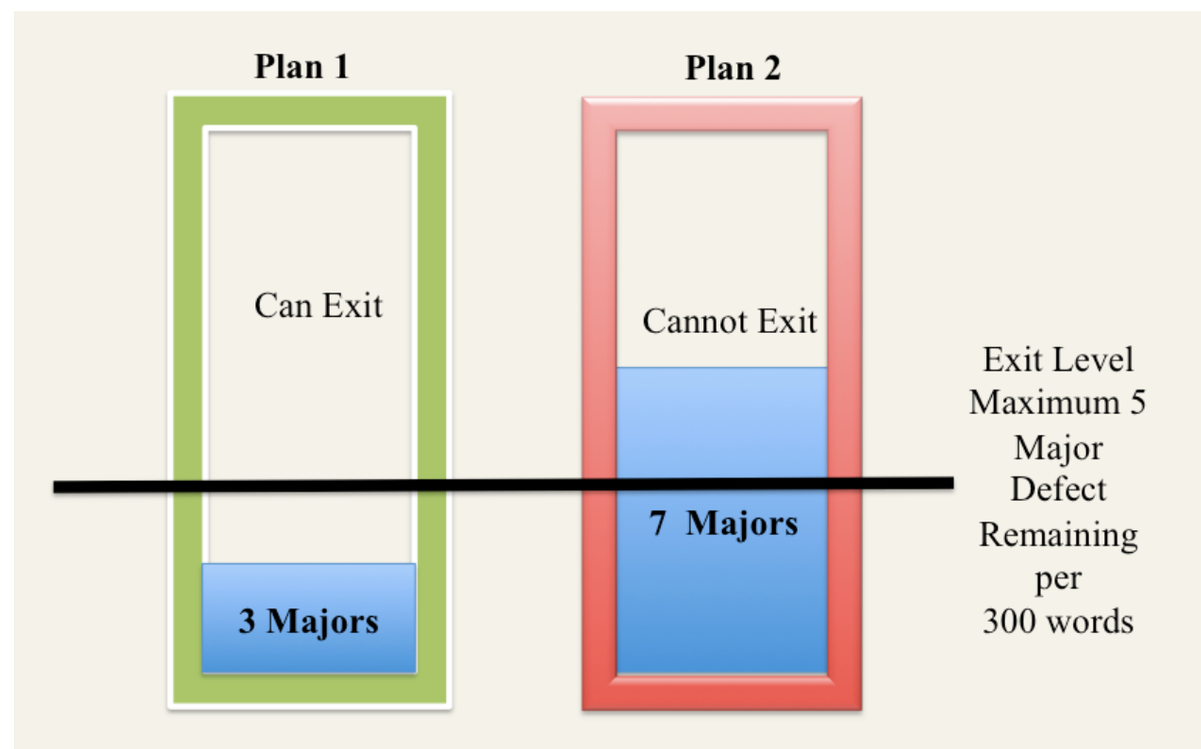
4. Measuring Development Specifications Quality: Lean Quality Assurance

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 Practical Industry 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%

50:1 !

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

results from a third generation product ("Gen 3") that was

characteristics of the first product: it ran on similar platforms,

6. Advanced Product Owner Responsibilities and Capability: much better requirements and design than conventional agile offers.

http://www.gilb.com/dl799

Gilb's Mythodology Column

Advanced Product Owners

by Tom & Kai Gilb

We are going to argue that the normally defined role of Product Owner (PO) is inadequate for projects that have serious multiple quality requirements, and consequent architecture processes, to deliver the necessary levels of performance and quality.

This includes all large serious projects, such as government or corporate projects. We do not want to argue that the Product Owner role as conventionally defined is inadequate for small projects, nor for projects that are not dependent on multiple state of the art quality, performance, and cost levels – and the consequent architecture to meet them. However, a point is reached where a project is so demanding that methods adequate for smaller projects will fail. The methods of a homeowner-built house will not work for a 200-story skyscraper. The Scrum project failure rate (about 19%) may be better than waterfall (more like 40%) [3] due to better feedback. But killing only one out of five pedestrians at a crossing, as opposed to two out of five, is still not good enough. We need to get closer to zero project failures in IT development. The myth is that the conventional PO process is *universal*, and that it scales up to any type of project. I believe we need to become more aware of the limits of today's dogma, and to identify practices appropriate for Agile methods in the more demanding projects.

The myths of the process owner are in italics below (http://www.mountaingoatsoftware.com/agile/scrum/product-owner). Our comments will briefly indicate another point of view for the more demanding projects. This column will not allow us to argue in detail, but the references will help those who need more depth. The purpose of this column is to open up the debate for a less dogmatic and less oversimplified presentation.

1. The Scrum product owner is typically a project's key stakeholder.

[1] The PO in reality needs to perform at least three very distinct functions, which are well understood in larger scale software engineering, and enterprise IT.

Requirements Engineering (RE): this is NOT a matter of User Stories. This is primarily a matter of quantified specification of the top-level primary drivers of a project [5]: qualities such as security, usability, and adaptability [4]), project constraints, and performance requirements. There is no such concept as this in Scrum or other similar Agile variations at present.

Page 63 Agile Record – www.agilerecord.com

Advanced: = 'Evo' = Agile Method *

<u>*Advanced*</u> Product Owner

- Value Focussed
- Real Engineering
- Requirements = Value
- Stakeholder Focussed (all 50+ !)
- Qualities Focussed (all 30)
- Measurable Value Stream
- Architecture Engineering

Conventional 'Product Owner'

- Code Focussed
- Craft ('Softcraft')
- Reqts = Function, Story
- User Customer Focussed (all 2)
- Bug Focussed (not even MTBF)
- Code Stream
- No clear design concept

* CE book, Chapter 10: Evolutionary Project Management: Chapter 10: Evolutionary Project Management: http://www.gilb.com/DL77

Copyright Tom@Gilb.com

The Policy

- Advanced Product Owner' Policy:
  System 'Requirements Engineer' (RE).
  - **<u>Background</u>**: this policy defines the expectations for a 'Product Owner' (PO) for serious, critical, large, and complex systems.
    - This implies that it is not enough to manage a simple stream (Backlog) of 'user stories' fed to a programming team.
    - It is necessary to communicate with a systems engineering team, developing or maintaining the 'Product'.
      - *System* implies management of all technological components, people, data, hardware, organization, training, motivation, and programs.
      - *Engineering*: means systematic and quantified, 'real' engineering processes, where proactive design is used to manage system performance (incl. all qualities) attributes and costs.

1. COMPLETE REQUIREMENTS:

- The RE (Requirements Engineer) is responsible for **absolutely all requirements specification** that the system must be aware of, and be responsible for to all critical or relevant stakeholders.
  - In particular, the RE is
    - not narrowly responsible for requirements from users and customers alone.
    - They are responsible for all other stakeholders,
      - » such as operations, maintenance, laws, regulations, resource providers, and more.

2. **QUALITY REQUIREMENTS:**

– The RE is **responsible for the quality level**, *in relation to official standards*, of all requirements they transmit to others.

- They are consequently responsible for making sure the quality of incoming raw requirements, needs, values, constraints etc. is good enough to process. No GIGO.
- If input is not good quality,
  - they are responsible for making sure it is better quality,
  - or at least clearly annotated where there is
    - » doubt, incompleteness, ambiguity and any other potential problems, they cannot resolve yet.

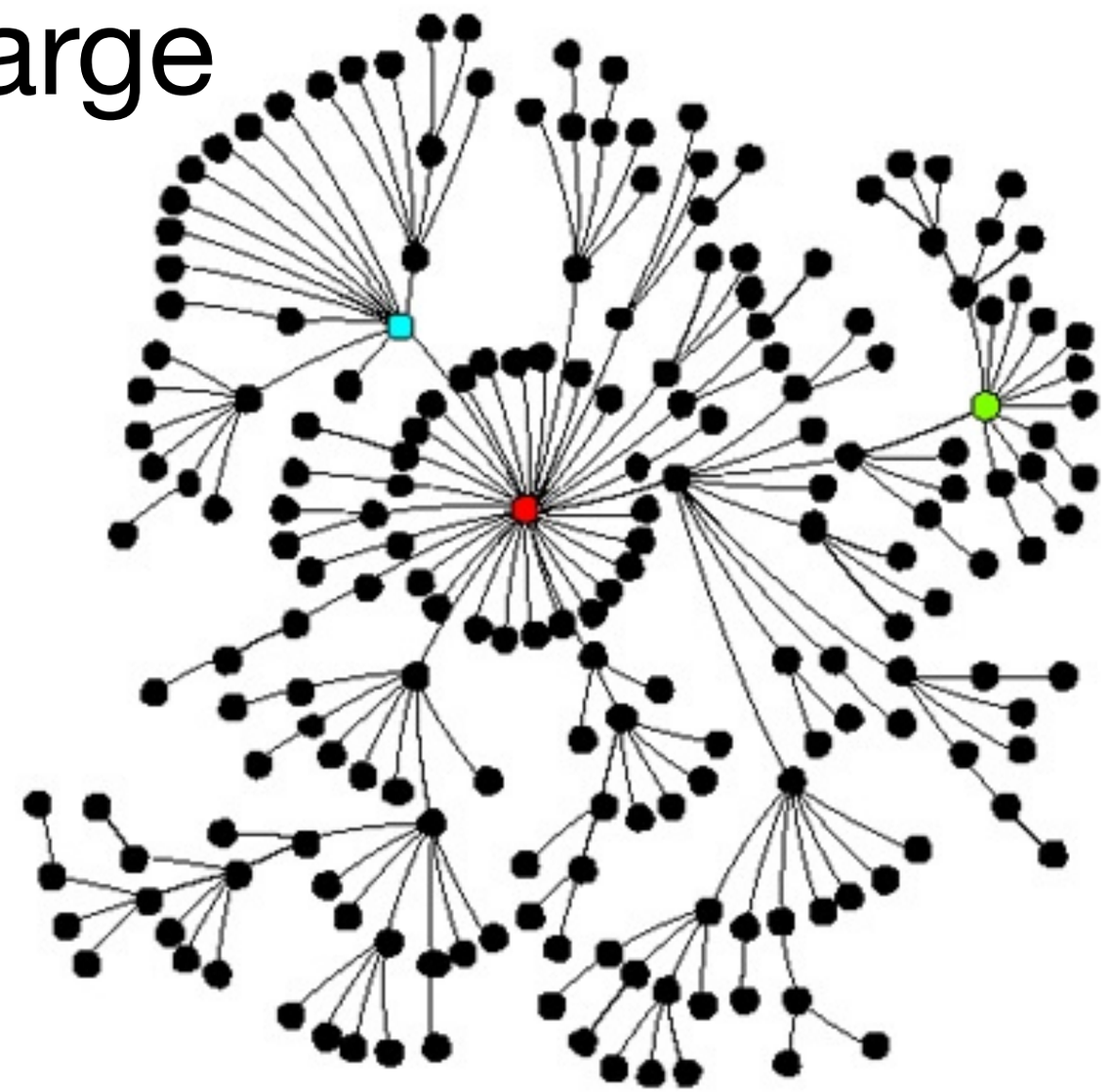
3. ARCHITECTURE:

- The Requirements Engineer is NOT responsible for any architecture or design process itself.
  - This will be done by professional engineers and architects.
- They are however very much responsible for a *complete* and *intelligible quality* set of requirements,
  - transmitted to the designers and architects.
- The are also responsible for transmitting quality-controlled architecture or design specifications to any relevant system builders.
  - These are the designs which are input requirements to builders. Effectively they are ‘design constraints requirements’.

4. Priority Information:

- The Requirements Engineer is NOT responsible for *prioritization* of requirements.
- Prioritization is done dynamically
  - at the project management (PM) level,
  - based on prioritization signals in the requirements,
  - and on current feedback and experience in the value delivery cycles (Sprints).
- The primary responsibility of the Requirements Engineer,
  - is to *systematically and thoroughly collect and disseminate* all relevant priority signals, into the requirement specification;
  - so that intelligent prioritization can be done at any relevant level, and at any time.

7. Scale-Free Agile: Planguage works at all scales large and small.



Erik Simmons, Intel Scaling

On 08 Jan 2016, at 19:30, Simmons, Erik erik.simmons@construx wrote:
Just a couple of things come to mind after reading this:

(Gilb:

Beyond Scaling: Scale-free Principles for Agile Value Delivery - Agile Engineering.

© tom@Gilb.com 2016, Posted at gilb.com resources/downloads/papers
http://www.gilb.com//dl865
Version March 14 2016, Modified April 11 2016 (XP)

Cheers,
e

Erik Simmons, Intel Scaling

I've not been a fan of the scaling movement since it started.

There are very **few things that scale well**, and economies of scale are often pursued without adequate understanding of the accompanying **diseconomies of scale.**

SW development does not scale well

- *because* of the **diseconomies of complexity**,
- such as the number of communication pathways,
- cognitive load on programmer brains, etc.
- That is among the core reasons for Brooks' Law.

What makes us think that scaling Scrum, which is successful in small teams and projects, is a good idea?

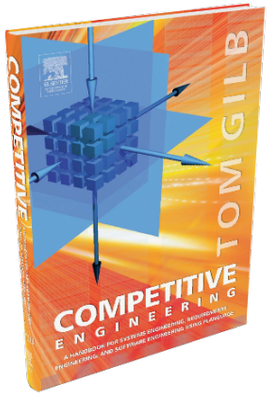
A grown-up is not a scaled baby.

Scaling as a concept is selling a lot of books, consulting, and certifications right now. But **I don't think it is a valuable concept.**

erik.simmons@construx.com

Erik Simmons, Intel Scaling

- Instead, I believe that the **majority of what you have** included for ideas, principles, etc. from CE and VP are in fact **scale-free**.
- They are **not dependent on *project* or *organization* size.**
- They are **good heuristics for almost any project,**
- and **nearly universally applicable**
  - (nearly universal because I hear Koen in my head, and all is heuristic).
- So, CE and VP are not *about* scaling
  - so much as they should be taught and understood as **scale-free.**
- Size is not a reason to choose (or not choose) to use Competitive Engineering, Evo, Planguage, etc.
- As you quoted me in the paper – **this stuff works**.
  - It works on **small** projects. It works on **large** projects.
- Evo on a 5-person team is not really much different than Evo on a 100-person team, except there are more people.
- The principles apply **without alteration** (or "scaling").
- Anyone who sees a random page of your new paper would probably not guess the topic is scaling (unless you happen to mention that in the text on that particular page).
- **'Competitive Engineering' does not scale. It doesn't need to.**



erik.simmons@construx.com

Erik Simmons, Intel Scaling

- There's no doubt that large projects are *different*.
- There's no doubt that we should approach them *differently*.
- We still don't have a recipe for large projects, and probably never will.

- But all that does *not* lead me to think that the answer to large projects can be found in scaling successful practices for small projects.

- Instead, **it must be found** in use of **principles and practices that are scale-free**,
  - coupled with use of particular practices that are effecting on large projects.

- If something that works on small projects also works on large projects, then I'd propose we call it a **scale-free practice**, not a scaled practice.

- erik.simmons@construx.com

Erik Simmons, Intel Scaling

- I'm deeply interested in scale-free practices.
- I'm also interested in specific practices tuned to large, small, complicated, and complex projects,

- but I find **particular power in scale-free practices.**

- Your work for decades has been focused on a very good set of these.
  - **SQC, for example, works on any size specification. It does not (need to) scale.**
  - SQC: (Specification Quality Control).see next slide
-
- BTW, I think the agile principles are also quite scale-free. But most **Scrum practices are definitely not.**
-
- So, perhaps you can chart a better course by advocating for use of scale-free core practices,
  - augmented with a set of specific, tailored practices
  - that are effective for the size of the project in question.

erik.simmons@construx.com

Scale-free Principles

1. Keep focus on measurable delivery of critical values and their costs. [3, 4, 5, 6, 9, 10, 12, VP (20) Part 1**, VP 10.6**]
2. Deliver value early, quickly and regularly: in roughly 2% increments. [14, 11, **VP Ch.4,** 2**,** 5]
3. Do NOT focus on code delivery; focus on overall system value and costs. [**VP Ch.4,** 10D, 10F, 13, VP 3.4, VP 2.10, VP 9.8, 4, 12]
4. Focus on quantified *critical stakeholder* values. [**19**, VP 3.4, VP 3.7, VP 3.9, VP 3.10 VP 4.2, 10]
5. Synchronize all teams in terms of measurable value delivery. [VP 3.3, VP 3.4, VP Part 1, VP 3.6, VP 3.8, VP 8.4 , 11, 12, 13]
6. Solve big problems through ingenious architecture; not through coding faster. [VP 4.5, VP 5.1, VP 5.3, VP 7.2, 15]
7. Decompose the large problems by incremental value deliveries: not code deliveries. [**7**, VP Ch. 5, VP 5.1, VP 5.6 , 10, 11, 13, 15]
8. The software component needs to be integrated into the total system of hardware, data, people, culture. [VP 5.2, 10]
9. If your team cannot deliver small increments of real value early, frequently, and predictably; they are incompetent and need to be abandoned for those who can deliver. [7, VP 2.8, 10]
10. Never commit to contacts for *work done* or *code delivered* alone: there must always be a sufficiently large contractual protection, of paying for measurable value delivered. [12, 15].

Methods

1. Quantification of Values [10, VP 1.1].
2. Quantification of short term and long term costs [VP 3.4, VP 4.5, VP 6.7].
3. Design to Cost: Top Level Architecture [VP 7.9, 10].
4. Dynamic Design to Cost: Each Delivery Cycle [12 C, VP 4.5, VP 2.5, VP 2.3, 5, 10, 12].
5. Quality Control of Plans, Contracts, Code and all written artifacts [VP Part 2, VP Part 4, VP 7.7].
6. Flexible Contracting [12, VP 4.5].
7. **Value delivery Cycle Measurable Feedback, Learning and Change [4, VP 7.3, VP 9.8, VP 6.7, VP 8.6, 2, 9, 10, 11, 14].**
8. **Value Decision Tables (Impact Estimation Tables) [9, VP 2.3, VP 4.4, VP 5.3, 13].**
9. Risk Management in all aspects of planning and Management [VP Ch. 7], 12.
10. Intelligent Prioritization Policies: for short term and long term [VP Ch. 6, 12, 13, 14].

Engineering Tools

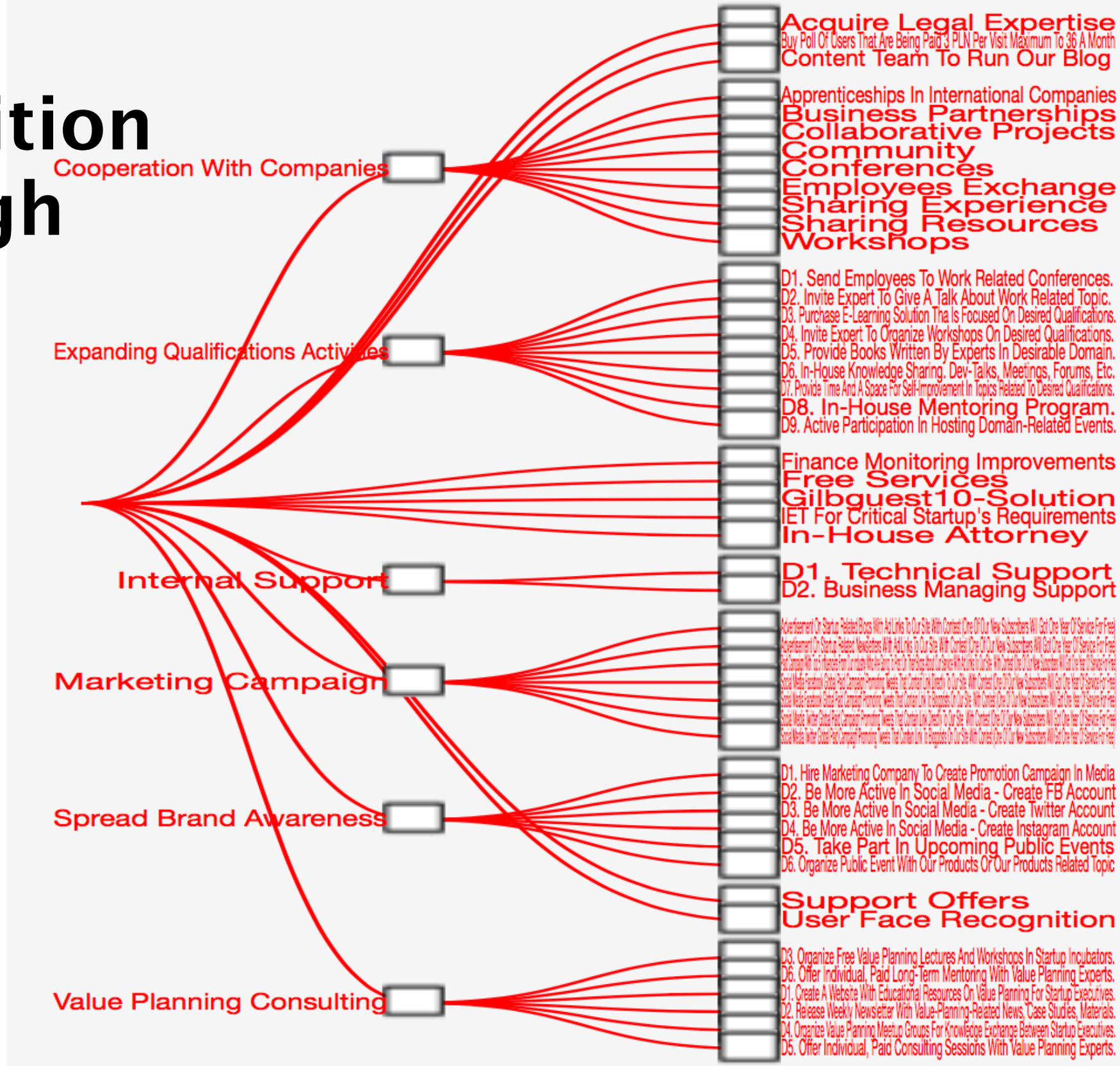
1. The Planning language: ‘Planguage’ [22, VP, 8, 9].
2. The 111111 Decomposition Method [7B, 7C, 3].
3. Flexible Contracts [12].
4. **The ‘Needs and means Planning’ tool [16, 9].**
5. Quantification of Values processes: Scales, Meters, Past, Tolerable, Wish, Goal. [VP 10.7].
6. The Agile Spec QC measurement process, Exit Processes, Rules [VP 10.4, VP Part 4].
7. Multiple Relationship Management technology [9, VP Ch.3, VP Ch. 6, 13].
8. Continuous Architecture adjustment based on delivery cycle feedback (Cleanroom) [5, 14, 8].
9. Graphic Visibility of Values, Costs, and Risks [16].
10. Design to Cost Practices: initially and continuously [14, 12 C, VP 4.5, VP 2.5, VP 2.3, 5].

Why do these scaling ideas work?

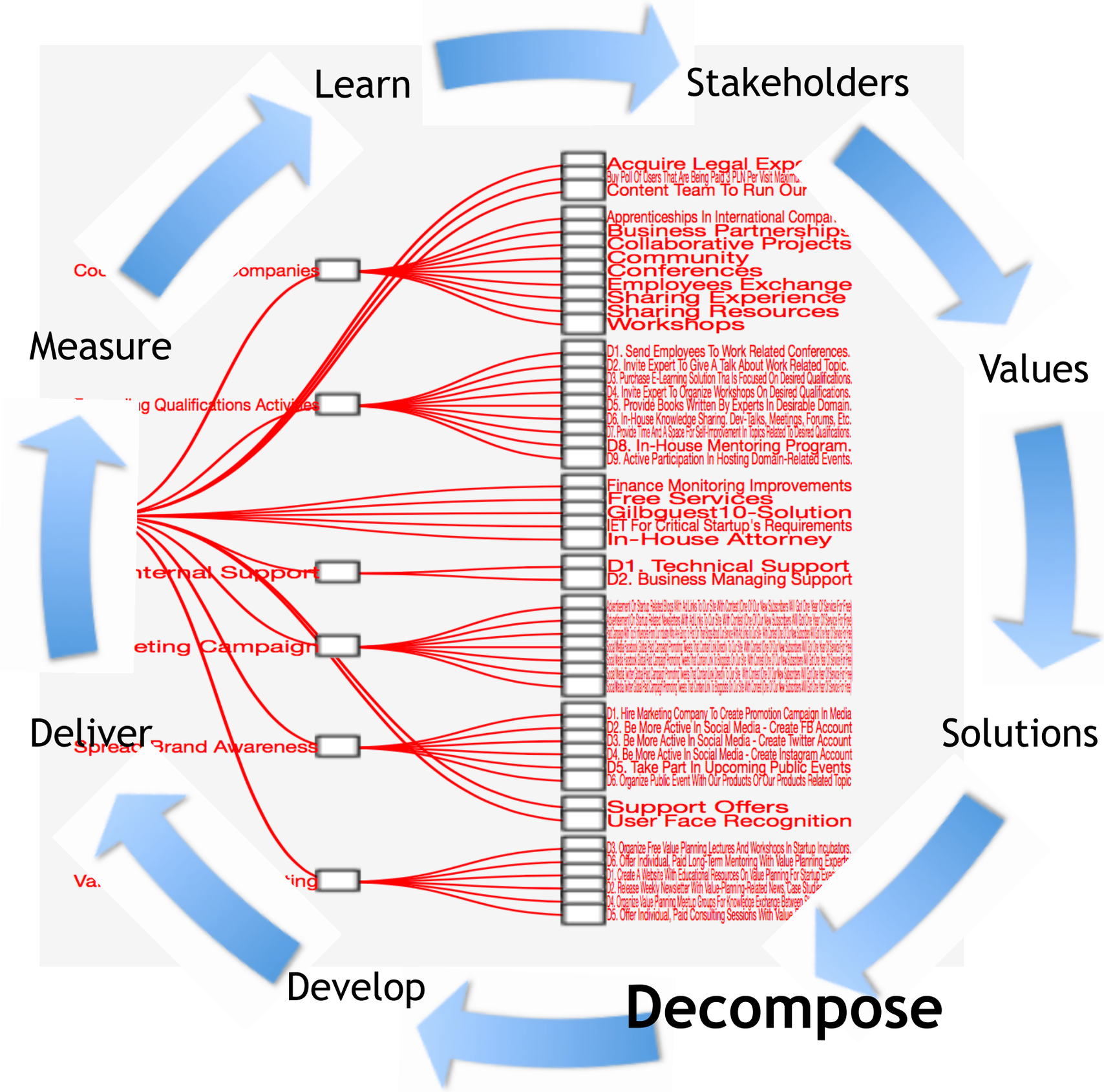
1. **Value quantification** allows us to focus on the stakeholder results, the main objectives of any project. All other activity, below this level should be contributing to delivery of the planned values. This means we can delegate the activity to any combination of specialist teams of any size and complexity: yet we can judge whether things are 'working'. We keep our eyes on measured value delivery. We can judge whether both our organization and our architecture are delivering as expected and needed. If not we can adjust (**dynamic design to cost**) and go with things that are actually delivering necessary value.
2. **Contracting for value** relates to the above explanation, with the added benefit that outside contractors are now motivated to focus on value delivery, not just 'doing work', or 'programming'. It does not matter so much about the underlying complexity. That underlying complexity either works (delivers contracted value measurably) or not. If not, we change it until it does, or give up if we cannot change to satisfy value delivery needs.
3. **Decomposition by small 2% deliverable value architecture components**: this is a very basic attack on large size and consequent complexity. We can see the incremental impact of each step on the whole system, regarding both value delivery and costs. If it is not good enough we try new ideas. If we run out of ideas that work, we need to stop.
4. **Risk Management:** our methods, including 1-3 above, are really all about managing the risk of failing to deliver value for money, on time. In addition we have suggested a number of additional risk management ideas. For example estimating the ± uncertainty of a design impact on values and costs [9]. For example asking for specific evidence [9] that any given design, or strategy will deliver the values and costs we need. The more engineering effort we put in to planning for risk up front, the less likely we are to get nasty surprises later (and then blame them on 'project size and complexity'; rather than our own lack of decent engineering planning).
5. **Delegation of decision-making [23]. Delegating the power to make decisions to a grass roots level, and in addition to do so incrementally while keeping any eye of their level of concern (in terms of value and costs), should obviously help us make better decisions, in an evidence-based situation.**

I have personally used these methods, with remarkable success, on projects involving for example 1,000 programmers and 1,000 hardware engineers (example HICOM (which was in total failure mode after 2 years, at Siemens. Boeing Aircraft projects [thousands of employees involved. To mention just a couple of many). There is no doubt for me that they work, and why they work.

8. Decomposition into small high value result deliveries



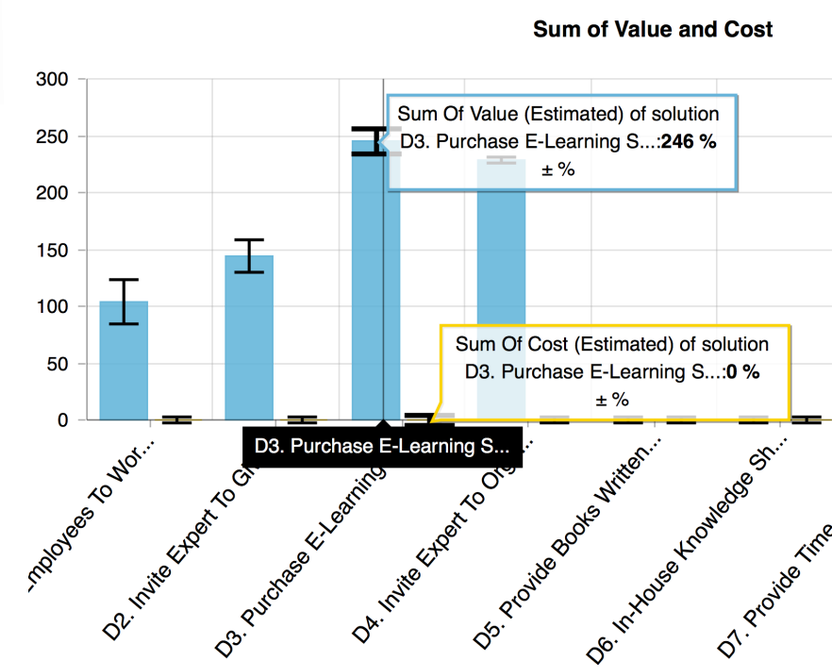
1. See the Chapter 5 Decomposition chapter in Value Planning book (leanpub.com/Valueplanning)
2. or https://www.dropbox.com/sh/dc7v636m7w7vvgx/AABfMAW_FnJny23XZKQZQkF4a?dl=0

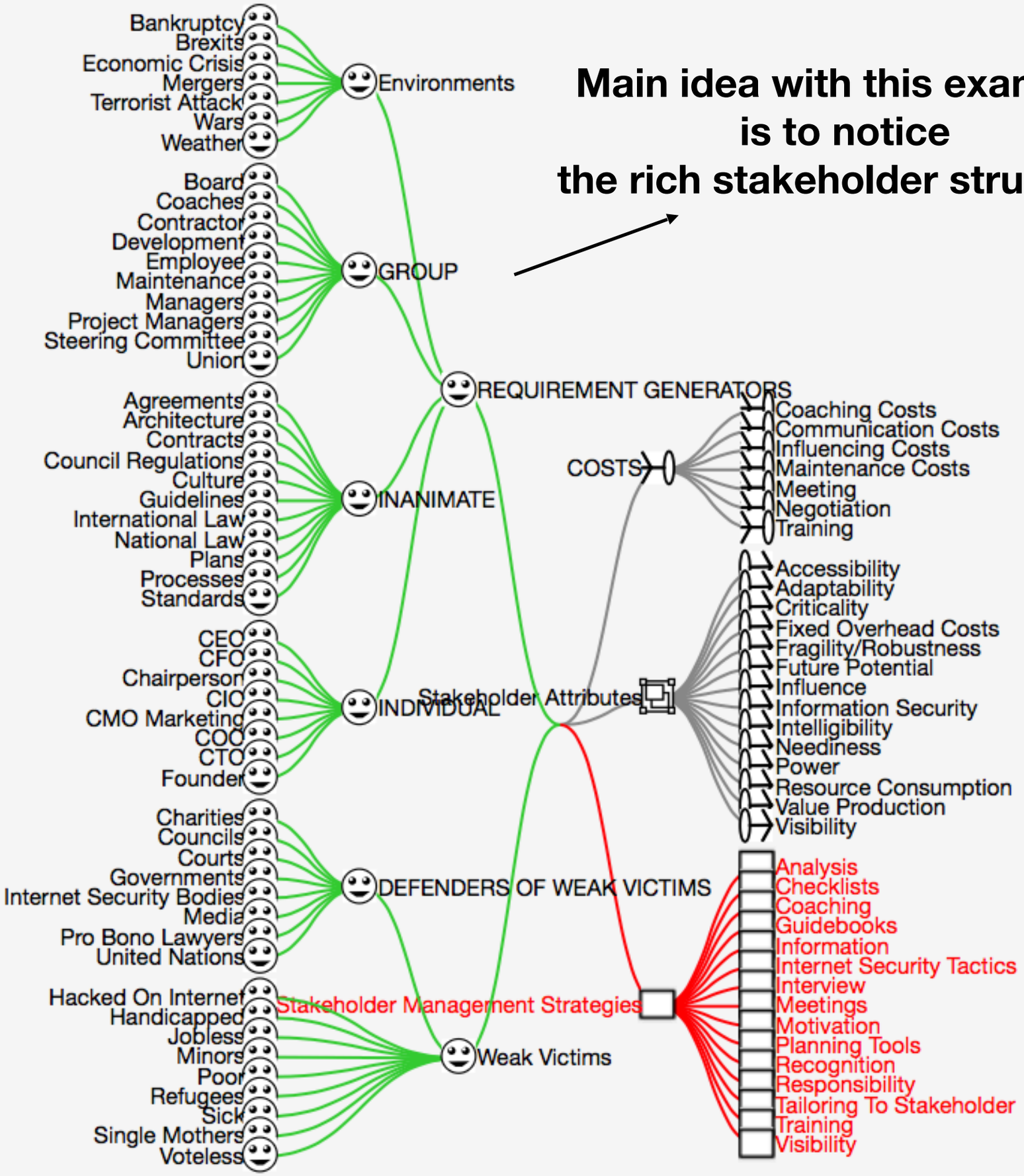


The solutions can be <u>decomposed</u> by 10x or 100x

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





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

Here are some other complimentary forms of decomposition

1. stakeholders
2. Values
3. Costs
4. designs

Security Value Quantification with Stakeholders

National Security

All values and qualities can be expressed quantitatively

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

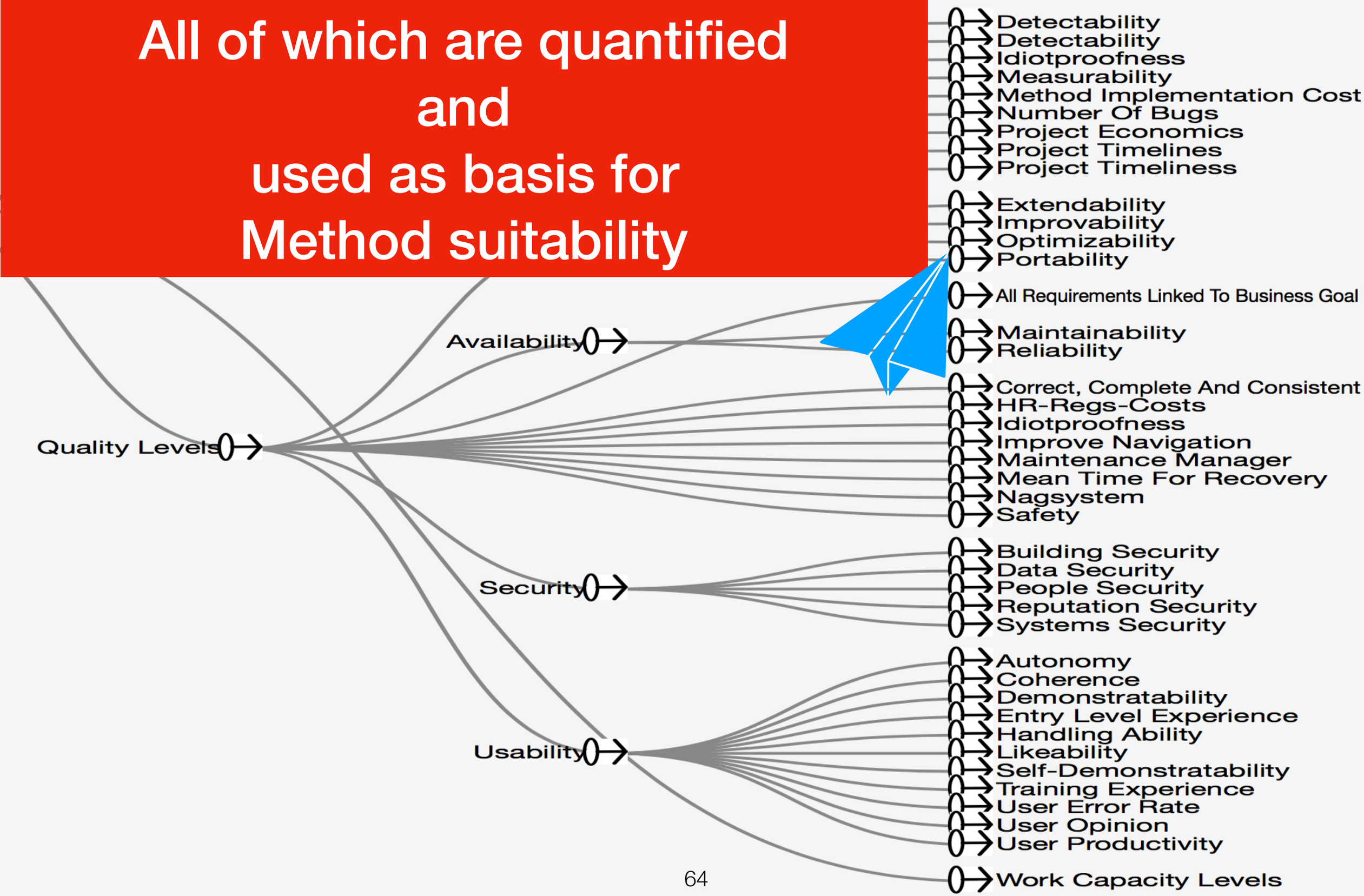
Wish: Level: **10** Number Bad Stuff [Effects = { Death }, Stakeholders = { <All> }, Attack Types = { Vehicle Attack,Knife Attack,Gun Attack }, Conditions = { High A

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

REQUIREMENT WITH MANY DIMENSIONS

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

Decomposition by Stakeholder Values
All of which are quantified
and
used as basis for
Method suitability



Example: Quantifying 'Portability'

Portability

Stakeholder Value *Label?*

(by tomgilb - 3 minutes ago)

Permalink

0.0.1

Is Part Of: Adaptability Value

Ambition Level:

Scale: % of [Method Components] that can Immediately be moved to [Devices] and [Software] by [Adapter Support].

A scale states the fundamental and precise operational definition for a specific scalar attribute.

Stakeholders: 3rd Party Suppliers, Internal Project Team, Procurement Gilbguest 17, System Administrator Users, Support

Wish: Level **90** % Portable [Method Components = { Requirements,Design }, Devices = { PC,Mac,iPads,Tablets }, Software = { Method Tools }, Adapter Support =...

Status: Level: **0** % Portable [Method Components = { <All> }, Devices = { <All> }, Software = { <All> }, Adapter Support = { }] When 18th September 2017

Portability

Stakeholder Value *Label?*

(by tomgilb - 6 minutes ago)

Is Part Of: Adaptability Value

Ambition Level:

Scale: % of [Method Components] that can Immediately be moved to [Devices] and [Software] by [Adapter Support].

Stakeholders: Change...

(by **tomgilb** - 6 minutes ago)

+ Link to Stakeholder

Tag	Actions
3rd Party Suppliers	
Internal Project Team	
Procurement Gilbguest 17	
System Administrator Users	

Stakeholders —>

Requirement Sources

Example: Quantifying 'Portability'

Release

Portability

<- The 'Portability' is the name or 'tag of the specification'

Permalink

Stakeholder Value *Label?*

(by tomgilb - 3 minutes ago)

0.(

Is Part Of: Adaptability Value

This documents where in a hierarchy the spec belongs and what *type* of spec (Value) it is

Ambition Level: Superior ease of moving methods software to new environment.

Management BS Level

Slogan or Headline

Many specs stop at this level.

We use this as a platform to develop much more precise requirements

Quantified, and
Decomposed to varied-value components

Ambition Level: Superior ease of moving methods software to new environments without human effort

Scale: Change...

Example: Quantifying 'Portability' THE SCALE DEFINITION with [Scale Parameters] decomposition: 2 levels

Scale Description:

% of [**Method Components**] that can Immediately, with little or no effort, be moved to [**Devices**] and [**Software**] by [**Adapter Support**].

[Scale Parameters] decomposition: 1st level

Adapter Support: defined as:

In House Support, External Specialists, Users Themselves

Devices: defined as:

PC, Mac, iPhone, Android, iPads, Tablets, Apple Watch,

Method Components: defined as:

Requirements, Design, Architecture, Quality Control, Project Management, Prioritization, Risk Management

Software: defined as:

Spreadsheets, Word Processors, Method Tools, Operating Systems, Mac OS, iOS, Windows

Second-Level Decomposition <— — — — — very detailed 'modelling' of the system

Example: Quantifying 'Portability'

Portability

Stakeholder Value *Label?*

(by tomgilb - 3 minutes ago)

Permalink

0.0.1

Is Part Of: Adaptability Value

Ambition Level:

Scale: % of [Method Components] that can Immediately be moved to [Devices] and [Software] by [Adapter Support].

Stakeholders: 3rd Party Suppliers, Internal Project Team, Procurement Gilbguest 17, System Administrator Users, Support

Wish: Change...

(by **tomgilb** - 3 minutes ago) 0

Scale Level: % Portable

By When:

90

<- Wish level (90) expresses a need or desire of a stakeholder

The 'Wish level' here, refers <u>only</u> to the defined Scale parameters below:
Requirements, Design… Method Tools…. PC Mac iPads Tablets ,,, In house Support

Qualifiers:

[Method Components] =

× Requirements × Design

[Devices] =

× PC × Mac × iPads × Tablets

[Software] =

× Method Tools

[Adapter Support] =

× In House Support

+Add additional qualifier

Source:

tom gilb

Add Comment...

Devops?

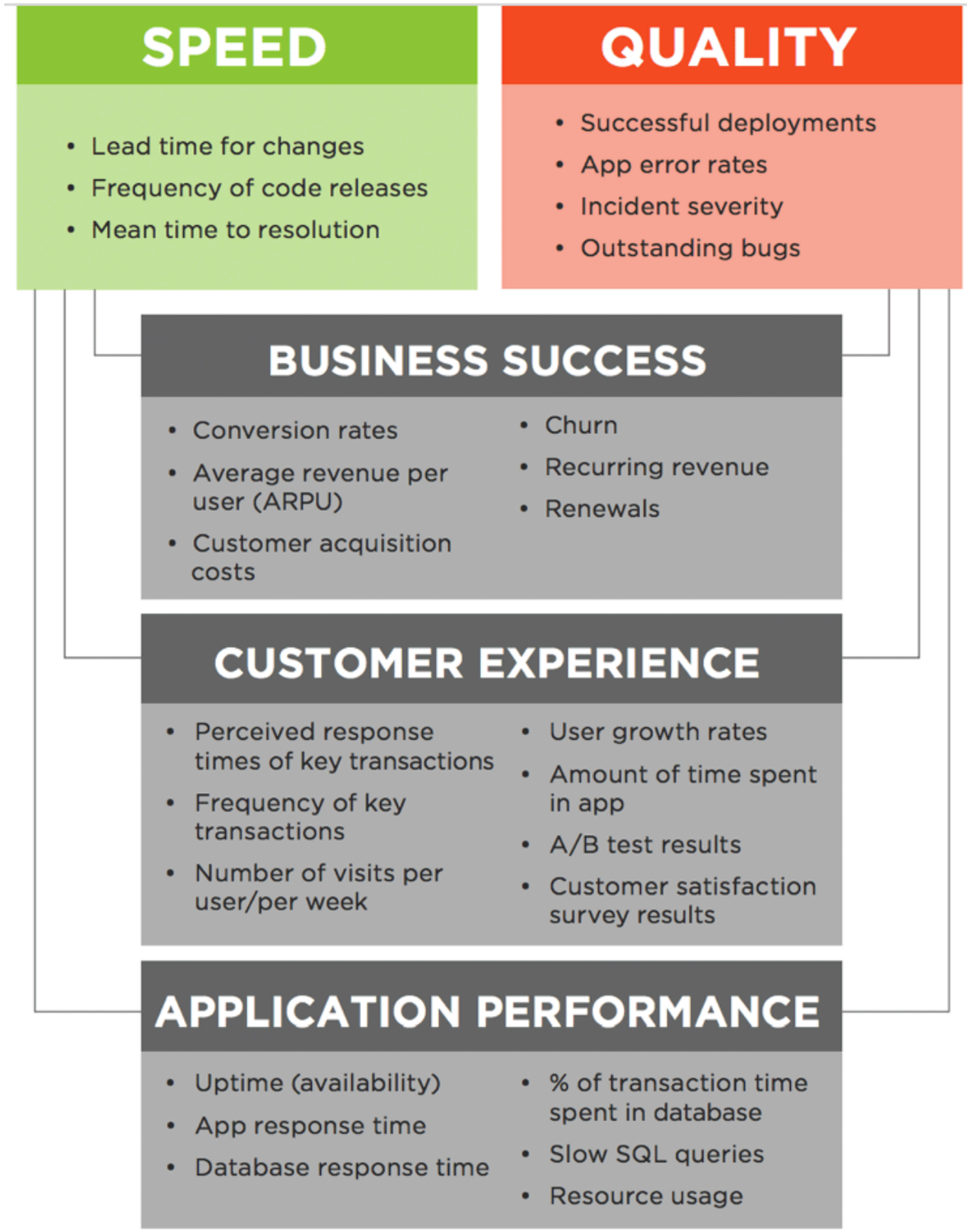
Devops 'heart' is in the right place.

- **Plenty of realtime multiple metrics to control *operations* and *change***

- **BUT**
- **Devops does not even try to seriously cover the problems outside and 'above' healthy operations and change**

- **For example Devops lacks**
  - **Serious deep stakeholder analysis**
  - **Serious quantification of business and organizational objectives for system development** (the Business success factors in the diagram are not good enough)
  - **Serious Understanding of technical qualities, like usability, security, maintainability** (quality is far more than 'bug absence')
  - **Serious architecture or strategy planning to meet the business objectives and constraints (IET etc.)**
  - **Systems Engineering (people, motivation, culture, data, hardware: Not just code!!)**
  - **Quality control (SQC/Inspection) of requirements, code, changes, test plans**

- **so Devops is missing the stuff I described in my talk as <u>things missing from 'popular' agile</u> !**



The laudable,
but limited, metrics categories
of Devops.
The illusion of 'business' metrics.

https://newrelic.com/how-to-measure-the-success-of-devops?content=eBook

User Story analysis

A Sample of Real User Stories April 2019

Not a very specific stakeholder name

These are NOT 'Goals' in any remote sense. They are Functional Requirements

'So that' implies a result I value will happen if we do the function

7	ser Type	Goal **(not really'Goals'. They are designs for reaching the higher requirement, badly stated in the ')**	Reason. THIS IS THE REAL VALUE OF STAKEHOLDER (USER TYPE). BUT IT IS SO VAGUE AND UNQUANTIFIED THAT NOONE CAN UNDERSTAND IT OR DESIGN FOR IT OR CHECK THE DESIGN, OR TEST, OR QA	THIS IS CALLED 'FAILED PROJECT.'
8	s a customer	I want to be able to add, edit and remove items in my tracking list	So that I can easily get updates on my delivered and undelivered items	NA
9	s a customer,	I want to be told when my parcel has been delivered to a neighbour,	So that I know where it has gone, and whom has accepted delivery -> **Customer Traceability**	Given a customer is on the RM app, When they have a parcel being delivered to a entered the tracking ID in Track and Trace, Then the app will display deliver to neighbou
10	s a customer,	I want to be given a parcel delivery window, **PROBLEM: ONE DIMENSIONAL THINKING. TRACEABILITY FORGETTING NEED FOR PERSONAL PRIVACY**	So that I'm not waiting all day to eceive it. -> **Delivery edictability**	Given a customer is on the RM app, When they have a parcel being delivered and tracking ID in Track and Trace, Then the app will display estimated delivery
11	s a customer,	I would like to print my Proof of Delivery,	at I can use it to make a nplaint. -> **Documentation ccessibility**	Given a customer is on the RM app, When they access their Proof of Delivery slip, Then they will be able to print it as an image

Value back to stakeholder

Design Impact

All the ‘so that’ reasons, are really badly specified Quality requirements

Not a very specific stakeholder name

These are NOT ‘Goals’ in any remote sense. They are Functional Requirements

‘So that’ implies a result I value will happen if we do the function

		Goal (not really’Goals’. They are designs for reaching the higher requirement, badly stated in the ’)	Reason. THIS IS THE REAL VALUE OF STAKEHOLDER (USER TYPE). BUT IT IS SO VAGUE AND UNQUANTIFIED THAT NOONE CAN UNDERSTAND IT OR DESIGN FOR IT OR CHECK THE DESIGN, OR TEST, OR QA	THIS IS CALLED ‘FAILED PROJECT.’
7	ser Type			
8	s a customer	I want to be able to add, edit and remove items in my tracking list	So that I can easily get updates on my delivered and undelivered items	NA
9	s a customer,	I want to be told when my parcel has been delivered to a neighbour,	So that I know where it has gone, and whom has accepted delivery -> **Customer Traceability**	Given a customer is on the RM app, When they have a parcel being delivered to a entered the tracking ID in Track and Trace, Then the app will display deliver to neighbour
10	s a customer,	I want to be given a parcel delivery window, **PROBLEM: ONE DIMENSIONAL THINKING. TRACEABILITY FORGETTING NEED FOR PERSONAL PRIVACY**	So that I'm not waiting all day to receive it. -> **Delivery Predictability**	Given a customer is on the RM app, When they have a parcel being delivered and tracking ID in Track and Trace, Then the app will display estimated delivery
11	s a customer,	I would like to print my Proof of Delivery,	So that I can use it to make a complaint. -> **Documentation Accessibility**	Given a customer is on the RM app, When they access their Proof of Delivery slip, Then they will be able to print it as an image

Value back to stakeholder

Design Impact

A defined ‘scale of measure’ for the ‘so that’ of
“easily get updates on my delivered and undelivered items”

Safari File Edit View History Bookmarks Window Help 100 % 12 Apr 13:46 Tom Gilbs

app.needsandmeans.com/requirements/SPEC-15K9J38

Untitled

Tag.Scale: 0 Hide Sidebar

Scale Description:

% [Parcel Customer] gets [Updates] for specific classes of [Delivery Items] which are available within 5 minutes of asking

Press ⌘+e to show editing toolbar.

Delivery Items: defined as:

Package, Bulk Mailing Packages, Dangerous Cargo, Fragile Cargo, Secret Cargo, Secure Cargo, ..

Parcel Customer: defined as:

Any **[Parcel Recipient]**, **[Parcel Sender]**,

Updates: defined as:

Scan Events, <Acceptance>, Sent To Mail Centre, To Delivery Office, Final Mile, Delivered Intermediary, In Hands of Final Recipient, ...

Time Units:

Calendar Date

Display Units: (optional, max. 10 characters)

Enter units for display

Source: by **tomgilb** - Apr 12th 2019, 11:55

Type something

Add Comment...

Today At Our Org.Status: 30 % updates ±10% [Parcel Customer = **Any Parcel Recipient**, Updates = **{To Delivery Office, Final Mi..**

Wish: 50 [Parcel Customer = **Any Parcel Recipient**, Updates = **{To Delivery Office, Final Mile}**, Delivery Items = **Package**] When 02 Apr ..

Parameters (click to add)

Scalar

- Meter
- Scale
- Stakeholders

Benchmarks

- Past
- Record
- Status

Targets

- Budget
- Goal
- Stretch

Scroll down for more

Terms (click to copy to clipboard)

Scale Templates (click to add)

Statement Templates (c add)

Here are all the ‘so thats’ as a list of ‘all quality requirements’

Safari File Edit View History Bookmarks Window Help 100 % 12 Apr 15:58 Tom Gilbs

app.needsandmeans.com/iet/IET-

Untitled

Customer Traceability Status: 60 → Wish: 85 % [Deliver	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Delivery Predictability Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Documentation Accessibility Status: 0 → Wish: 0		???? 0 % ????	???? 0 % ????
Identity Correctness Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Security Usability Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Story 1 Status: 30 → Wish: 50 % [Parcel	Δ: Δ%:	-100 -500 % -500%	???? 0 % ????
Updatability Status: 0 → Wish: 0	Δ: Δ%:	999 Infinity % Infinity%	???? 0 % ????
Sum Of Values:	Σ%:	Infinity %	0 %

Show Sidebar

USER STORIES TEST RISK ASSESSMENT MATRIXS RM App - TRAM

Add Category Insert Table Chart Text Shape Media

Sheet3 SetUp

C	D
Goal (not really’Goals’. They are designs for reaching the higher requirement, badly stated in the ’)	Reason. THIS IS THE REAL VALUE OF STAKEHOLDER (USER TYPE). BUT IT IS SO VAGUE AND UNQUANTIFIED THAT NOONE CAN UNDERSTAND IT OR DESIGN FOR IT OR CHECK THE DESIGN, OR TEST, OR QA
I want to be able to add, edit and remove items in my tracking list	So that I can easily get updates on my delivered and undelivered items **UPDATABILITY**
I want to be told when my parcel has been delivered to a neighbour,	So that I know where it has gone, and whom has accepted delivery **-> Customer Traceability**
I want to be given a parcel delivery window,**PROBLEM: ONE DIMENSIONAL THINKING. TRACEABILITY FORGETTING NEED FOR PERSONAL PRIVACY**	So that I'm not waiting all day to receive it. **-> Delivery Predictability**
I would like to print my Proof of Delivery,	So that I can use it to make a complaint. **-> Documentation Accessibility**

Text So that I'm not waiting all day to receive it. -> Delivery Predictability

l ‘Goals’ (Functional Requirements) potentially impact all ‘So that’ requirements
By how much, on what evidence and sources? with what ? certainty

‘Goal’ 1

‘Goal’ 2

		‘Goal’ 1	‘Goal’ 2
Customer Traceability Status: 60 → Wish: 85 % [Deliver	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Delivery Predictability Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Documentation Accessibility Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Identity Correctness Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Security Usability Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????
Story 1 Status: 30 → Wish: 50 % [Parcel	Δ: Δ%:	-100 -500 % -500%	???? 0 % ????
Updatability Status: 0 → Wish:	Δ: Δ%:	999 Infinity % Infinity%	???? 0 % ????
Sum Of Values:	Σ%:	Infinity %	0 %

Show Sidebar

D	E
VALUE OF STAKEHOLDER (USER TYPE). BUT IT IS SO VAGUE AND UNQUANTIFIED THAT NOONE CAN UNDERSTAND IT OR DESIGN FOR IT OR CHECK THE DESIGN, OR TEST, OR QA	(TG:THIS IS CALLED ‘FAILED PROJECT.’ IS WE remark: this iso as test planning guidance for t of the technical design ((Goal). BUT IS DOES N THAT THE REAL GOALS (LIKE Customer Trac Predictability) have improve the real business.
So that I can easily get updates on my delivered and undelivered items **UPDATABILITY**	T
So that I know where it has gone, and whom has accepted delivery -> **Customer Traceability**	T
So that I'm not waiting all day to receive it. -> **Delivery Predictability**	Given a customer is on the RM app, When they have a parcel being delivered and tracking ID in Track and Trace, Then the app will display estimated delivery
So that I can use it to make a complaint. -> **Documentation Accessibility**	Given a customer is on the RM app, When they access their Proof of Delivery slip, Then they will be able to print it as an image
So that I can easily access the	

So that I'm not waiting all day to receive it. -> Delivery Predictability

Known Unknowns:
fill in the ???? if you know what the impacts, of your ‘Goals (Functional Requirements) are, on all your ‘so thats’ (Quality requirements)

Requirements		I Want To Be Able...	I Want To Be Give...	I Would Like To P...	I Want To Be Able...	I Want To
Ap Accessibility Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	? 0 % ????
App Credibility Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????
App Reusability Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????
App Security Status: 0 → Wish: 0	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????
Customer Traceability Status: 60 → Wish: 75 % [Deliver.	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????
Customer Traceability Status: 60 → Wish: 85 % [Deliver.	Δ: Δ%:	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ????	???? 0 % ?

Show Sidebar

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 agile processes to **include QUALITY, and VALUE metrics**

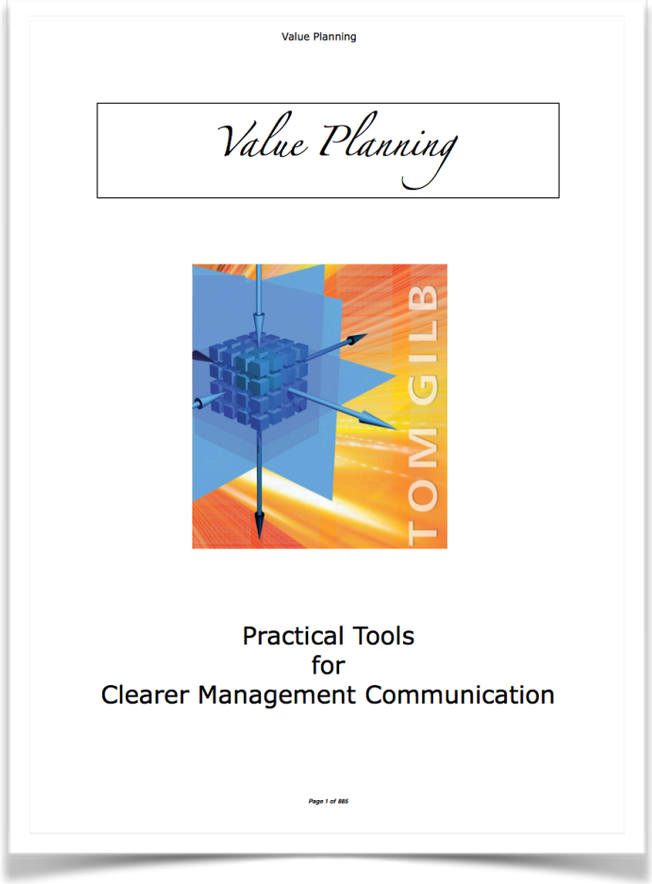
- **Quantification** of values is useful, even *without measurement. Quantification itself* is useful for **clearer communication** about critical objectives

- **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



Geta copy: leanpub.com/ValuePlanning, or at gilb.com