

Quality Engineering

**to really manage QUALITY
we must change our culture to
ENGINEERING.**

Tom Gilb

at Quality Days, Vienna, January 17 2018,
45 minutes

<https://2018.software-quality-days.com/en/> #qualitydays2018

These slides will be at www.gilb.com downloads
at

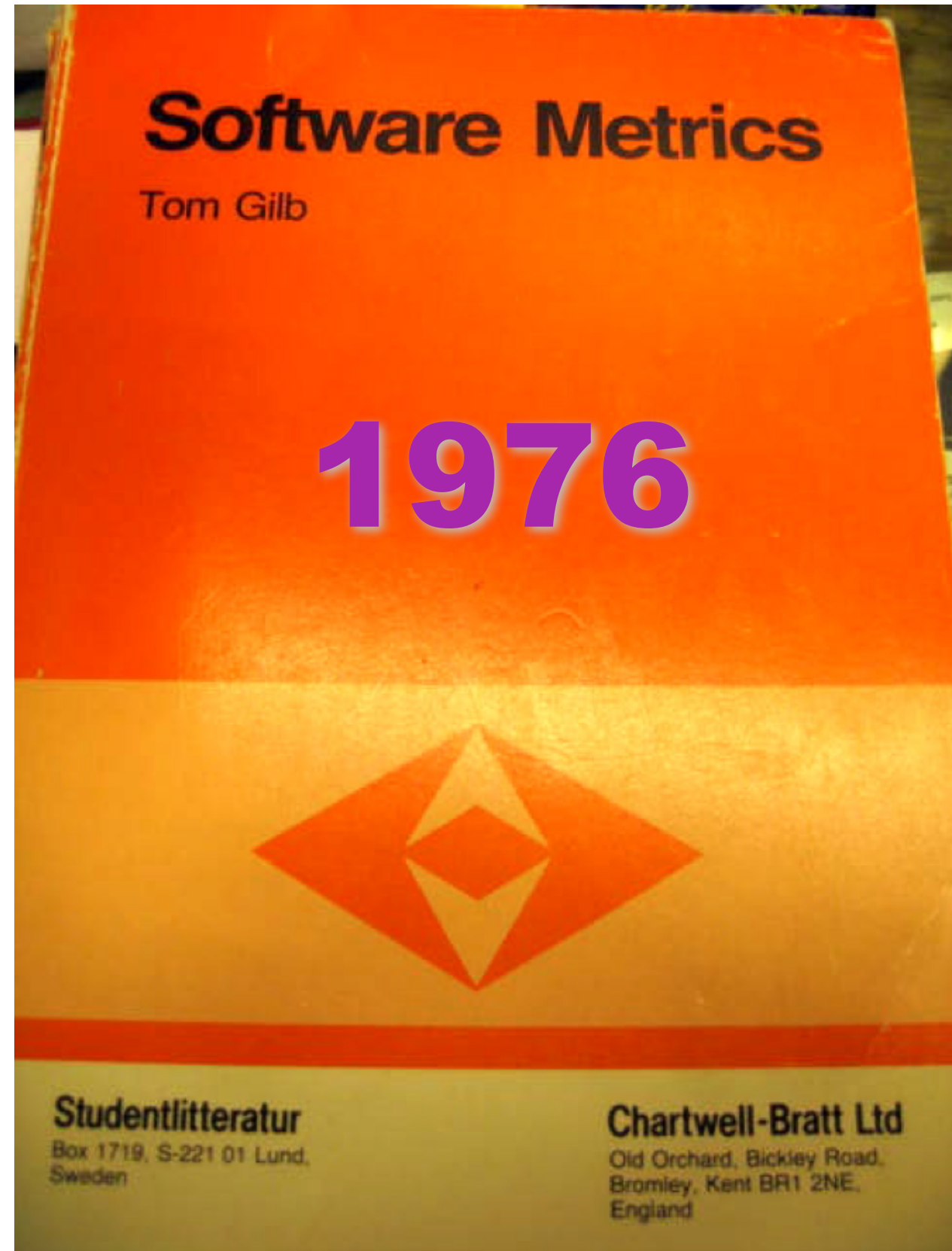
<http://concepts.gilb.com/dl923>

tom@gilb.com, www.gilb.com, @ImTomGilb

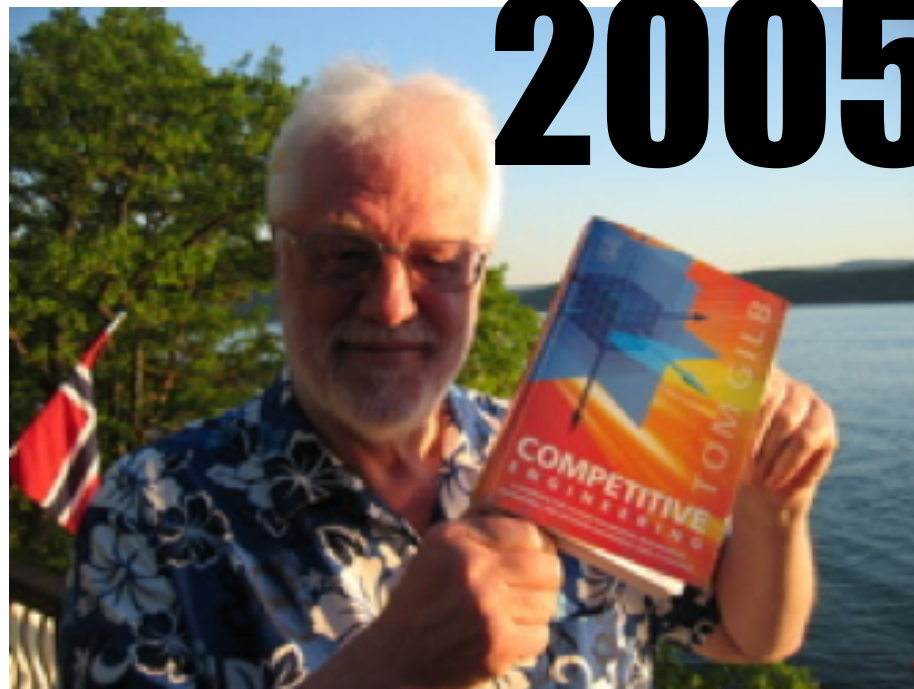
Quality Engineering Books 1976, 1977 and 2005, and 2018 (VP)



1977
USA



1976



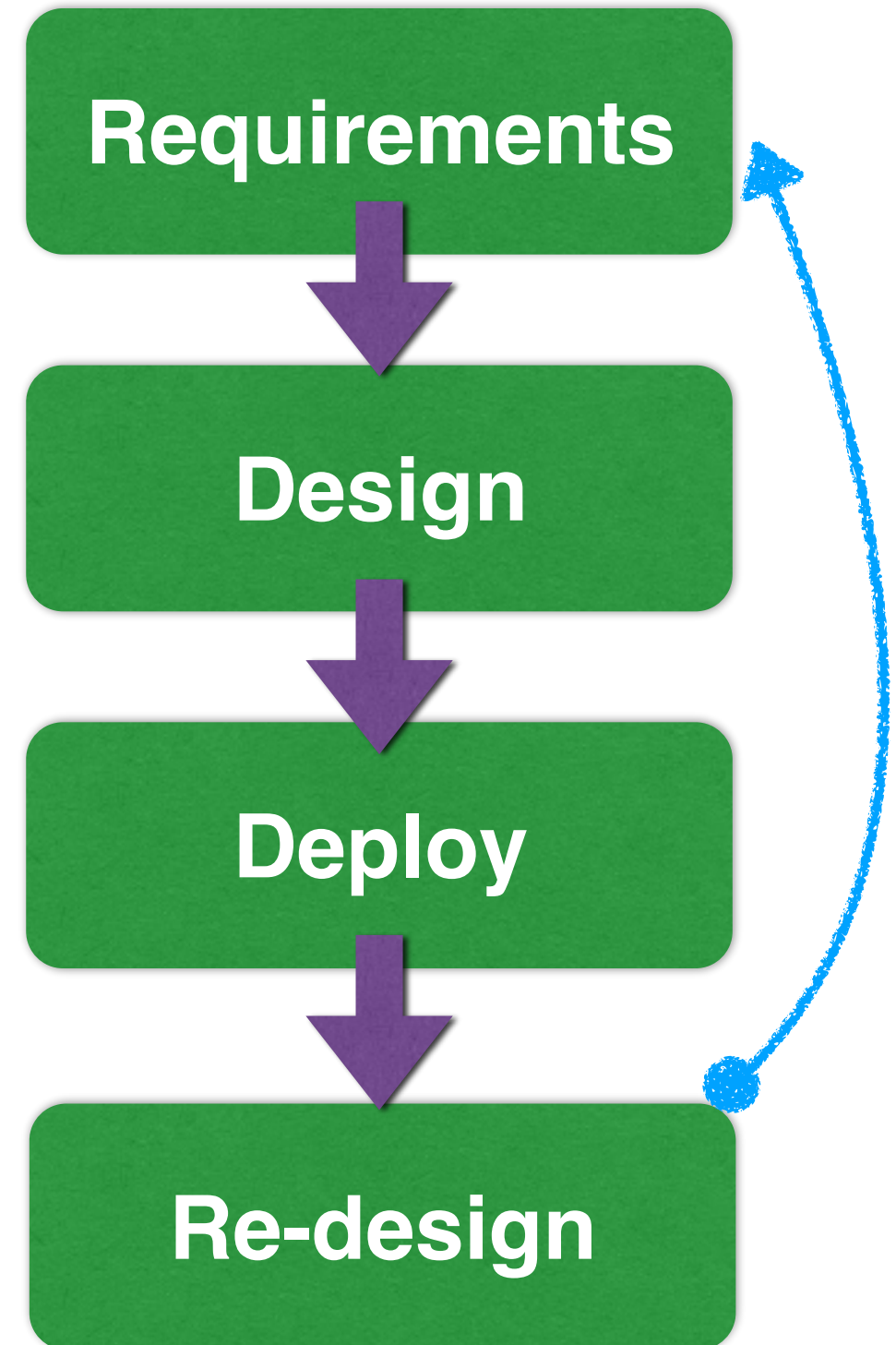
2005

Talk Outline:
Aspects of 'Quality Engineering'
How to be a 'Quality Engineer' in practice

- 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

LOGICAL STEPS OF 'QUALITY ENGINEERING'

1. **Environment Scope helps identify stakeholders.**
2. **Stakeholders have values and priorities**
3. **Values have many dimensions**
4. **Stakeholders determine value levels**
5. **Design hypotheses should be powerful and efficient ideas, for satisfying stakeholder needs**
6. **Design hypotheses can be evaluated quantitatively, with respect to all quantified objectives and resources**
7. **Designs can be decomposed, to find more efficient design subsets, that can be implemented early**
8. **Designs can be implemented sequentially, and their value-delivery, and resource costs, measured**
9. **Designs that unexpectedly threaten achievement of objectives, or excessive use of resources, can be removed or modified.**
10. **Designs that have the best set of effects on objectives, for the least consumption of limited resources, should generally be selected for early implementation.**
11. **A design increment can have unacceptable results, in combination with previous increments, and they, or it, might need removal or modification**
12. **When all objectives are reached, the process of design is complete: except for possible optimization of operational resources, by even-better design.**
13. **When deadlined and budgeted implementation-resources are used up, it might be reasonable to negotiate additional resources; especially if the incremental values are worth the additional resources.**
14. **When deadlined and budgeted implementation-resources are used up, it might be reasonable to negotiate additional resources; especially if the incremental values are worth the additional resources.**

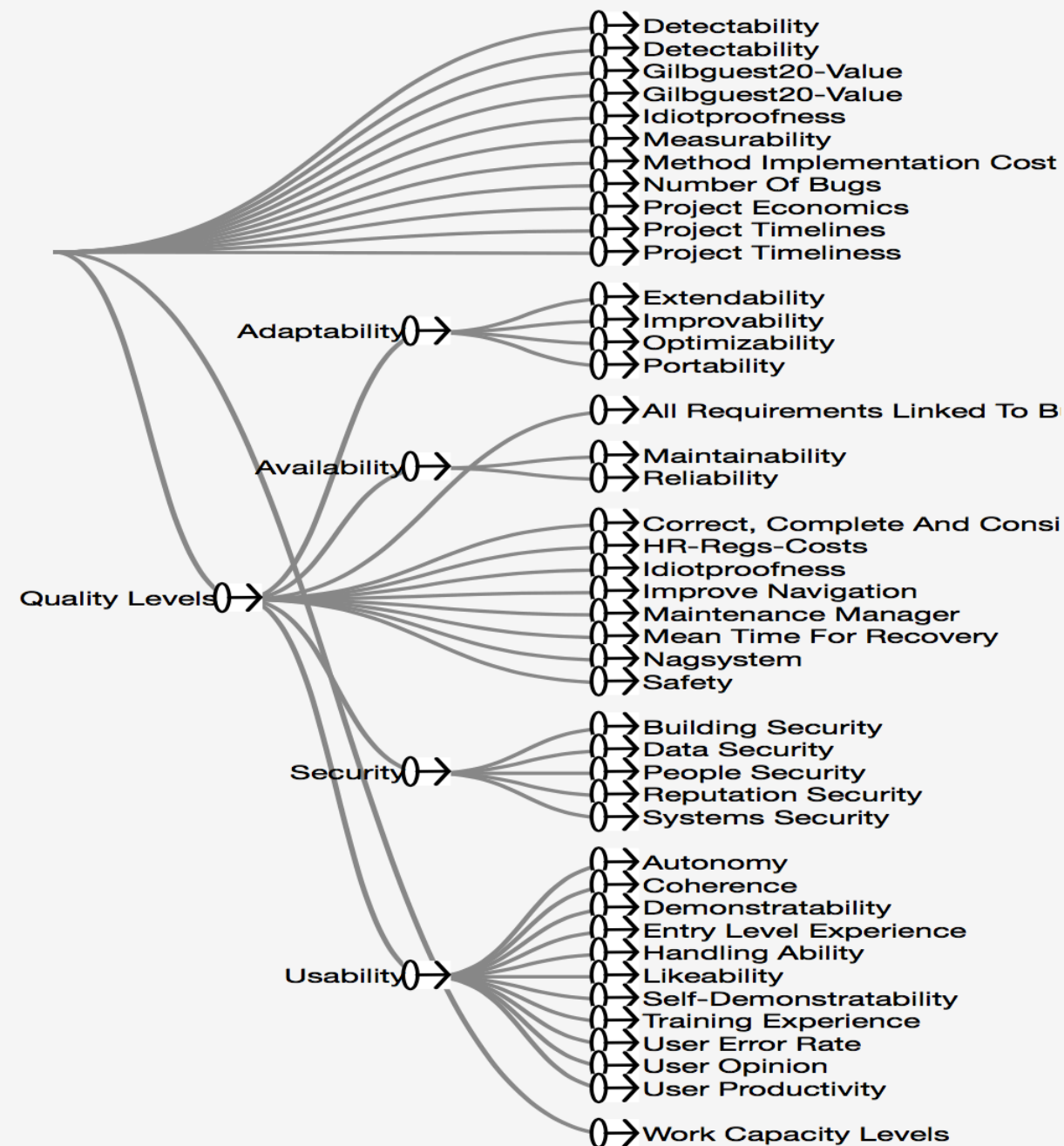


The Logic of Design: Design Process Principles.
Tom Gilb, 2016, Paper.
<http://www.gilb.com/dl857>

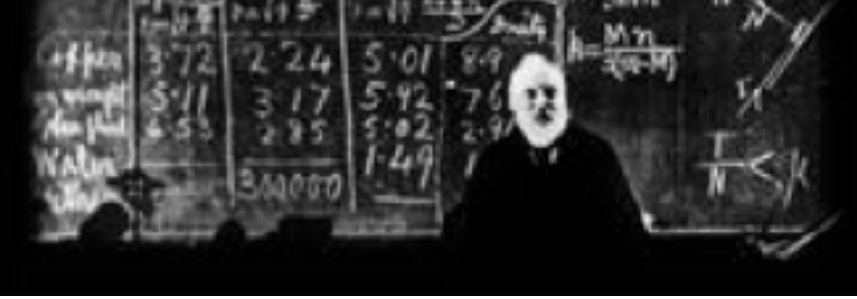
Tool Credit:

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>

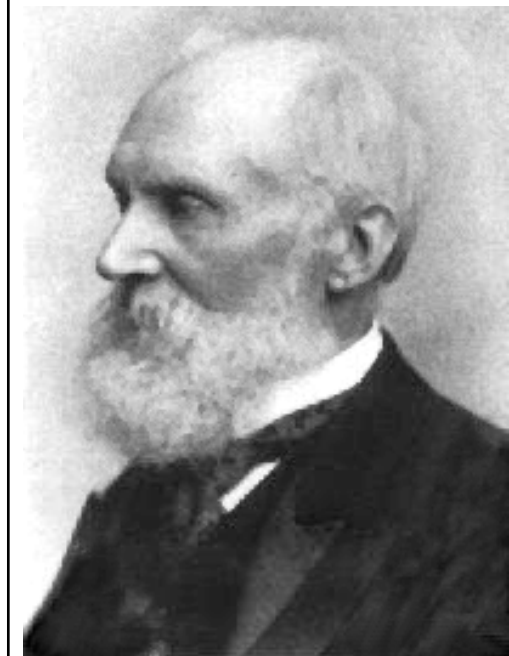
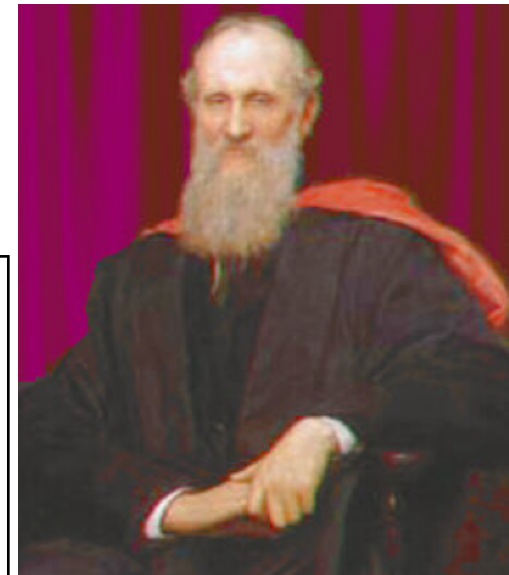
"In physical science the first essential step in the direction of *learning any subject* is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it.

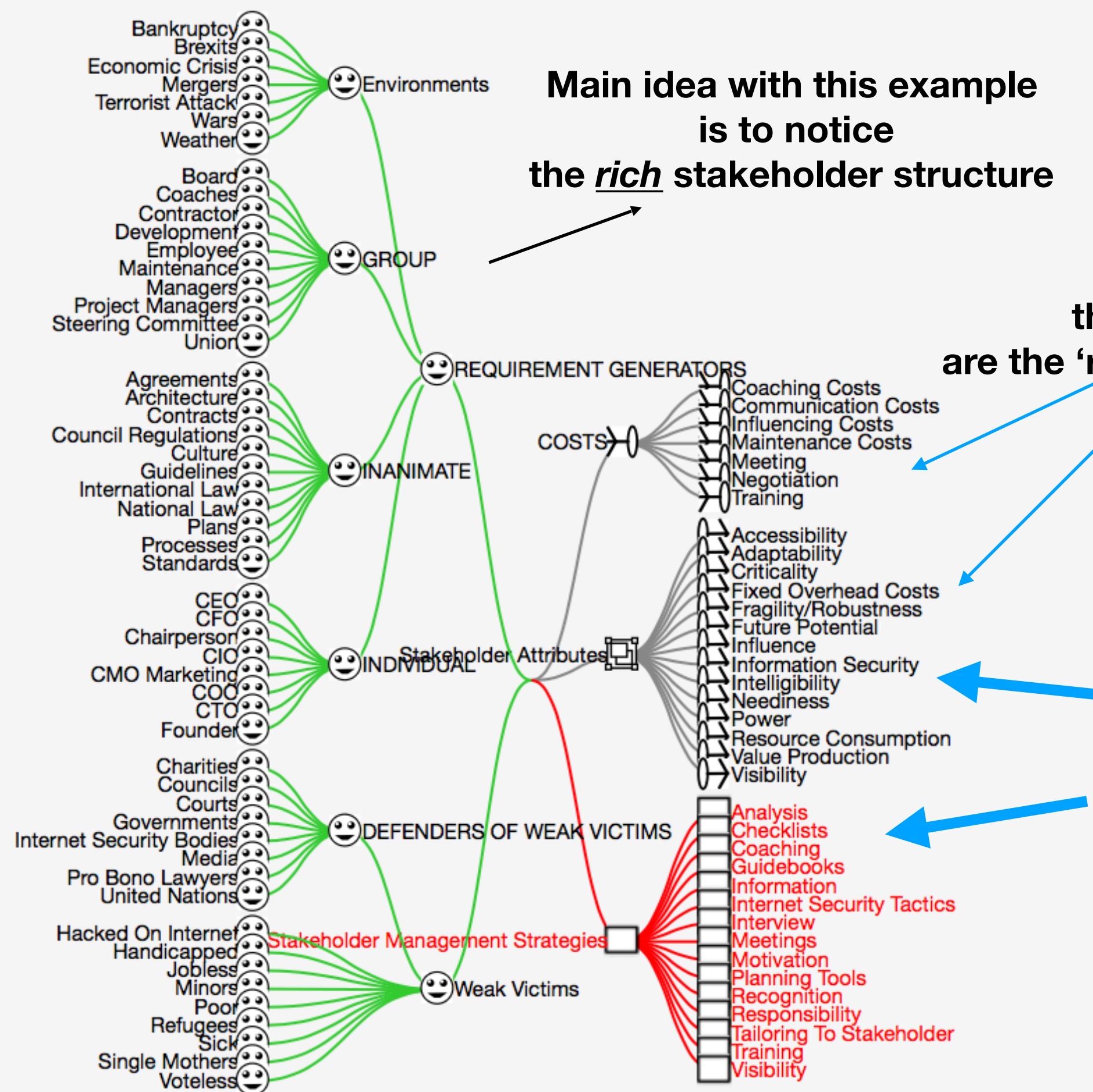
I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it;

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>





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

Next idea
is to notice
that stakeholders
are the 'requirement generators'

Stakeholder's
Needs &
Means diagram;
a 'Quality
Engineering'
framework

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.
is a 'Quality
Engineering' pre-
requisite



Security Value Quantification with Stakeholders

→ National Security

Business Value *Label?*

All values and qualities
can be expressed quantitatively

(✎ 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

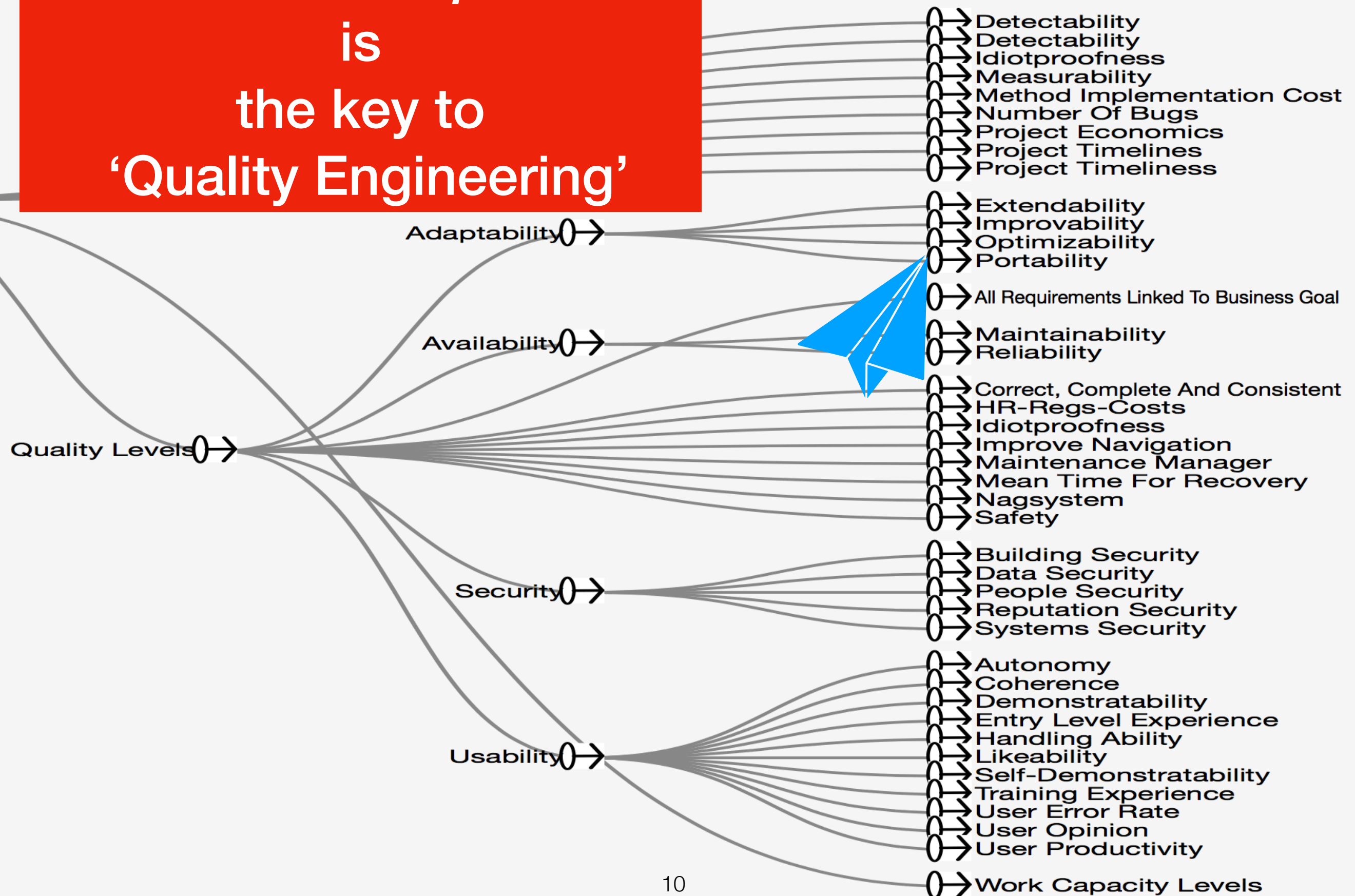
Status: Level: **150** Number Bad Stuff [Effects = { Death }, Stakeholders = { <All> }, Attack Types = { Vehicle Attack,Knife Attack,Gun Attack }, Conditions = { High A

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 A

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

‘Stakeholder Values’
All of which are *quantified*
is
the key to
‘Quality Engineering’



Portability

Example: Quantifying 'Portability'

[Permalink](#)

0.0.1

Stakeholder Value *Label?*

(✎ by tomgilb - 3 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: 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

11




Stakeholders —>

Requirement Sources

Example: Quantifying 'Portability'

Release by tomgilb 11 minutes ago

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

Stakeholder Value *Label?* (✎ by tomgilb - 3 minutes ago) 0.0

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:

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'

[Permalink](#)

0.0.1

 Portability

Stakeholder Value *Label?*

( by tomgilb - 3 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: 3rd Party Suppliers, Internal Project Team, Procurement Gilbquest 17, System Administrator Users, Support

Wish:

( 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 only to the defined Scale parameters below:
Requirements, Design... Method Tools.... PC Mac iPads Tablets ,, In house Support

Qualifiers: 

[Method Components] =

☐ Requirements ☐ Design

[Software] =

☐ Method Tools

[Devices] =

☐ PC ☐ Mac ☐ iPads ☐ Tablets

[Adapter Support] =

☐ In House Support

 Add additional qualifier

Source:

tom gilb

 Add Comment...

Requirements		<input type="checkbox"/> Incentivise	<input type="checkbox"/> Tea Kiosk	<input type="checkbox"/> 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%	$\Sigma \Delta\%: 40 \pm 180$ %
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 $0 \pm \text{NaN}$ % 0 % (x 0.6) 0%	30 ± 10 -20 % Injury 50 ± 25 % 15 % (x 0.3) 50%	$\Sigma \Delta\%: 50 \pm 25$ %
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%	$\Sigma \Delta\%: 120 \pm 30$ %
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%: ?%:	$500k \pm 0$ 500k \$ 17 ± 0 % 34 % (x 0.0) 17%	$2m \pm 0$ 2m \$ 67 ± 0 % 134 % (x 0.0) 67%	$=:1m \pm 0$ Δ: 1m \$ Δ%: 33 ± 0 % ?%: 66 % (x 0.0) 33%	$\Sigma \Delta\%: 117 \pm 0$ %
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:
Engineering the design architecture for reaching the quantified
quality levels on time

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?**

**‘Engineering’ is researching risks and
unknowns**



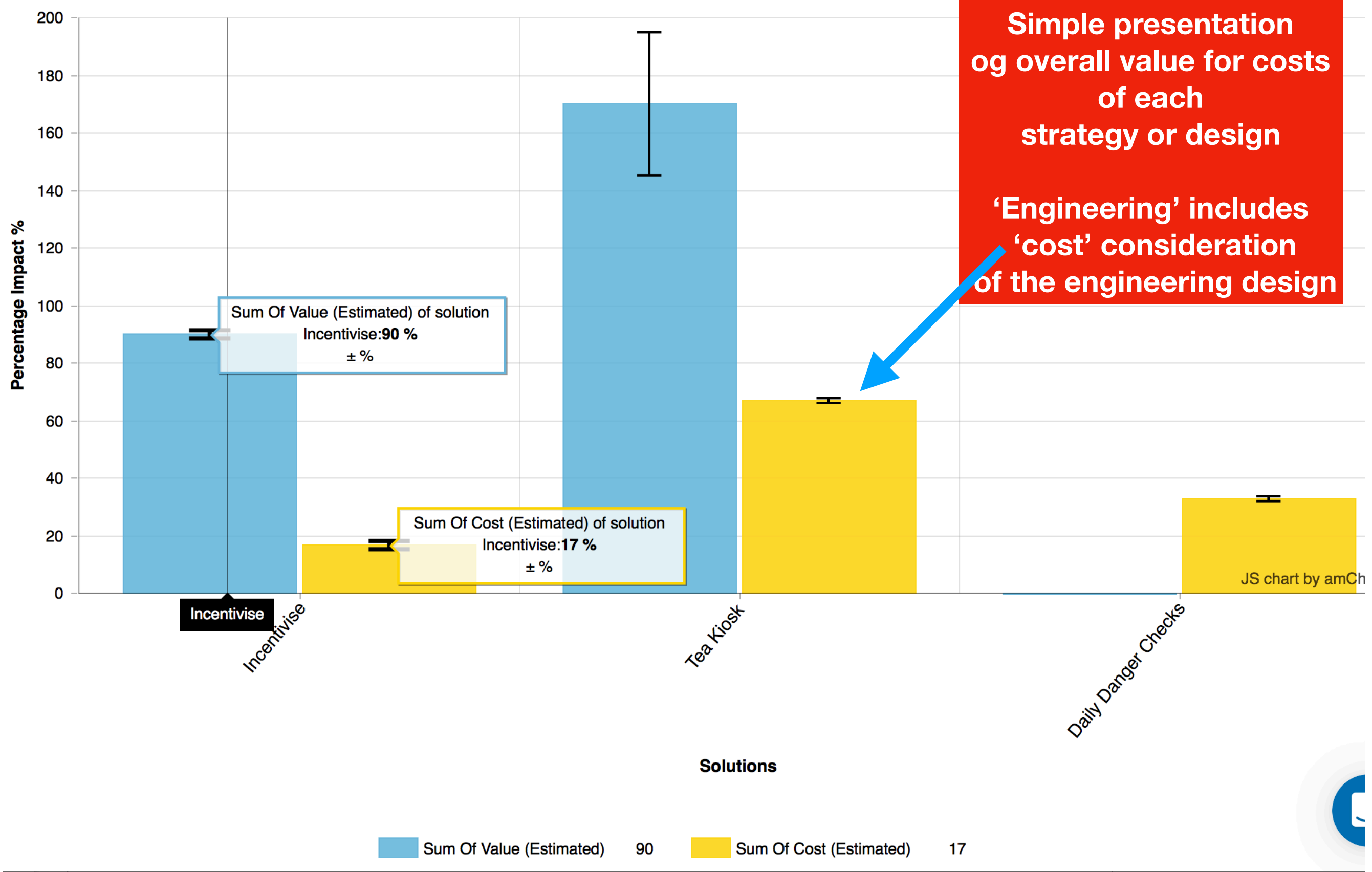
Designs ->		<input type="checkbox"/> Incentivise	<input type="checkbox"/> Tea Kiosk	<input type="checkbox"/> Daily Danger Checks	Sum
Requirements					
Project Timeliness Status: 10 → Wish: 5 % % time overrun necessary to deliver... Δ%: [Project Cost Size = { Medium (\$10k -...)] ?%: 30th June 2017		8 ± 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... Δ%: [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... Δ%: [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%	ΣΔ%: 120 ± 30 %
Sum Of Values: Credibility - adjusted:		90 ± 0 % 32 %	170 ± 50 % 106 %	-50 ± 185 % -65 %	
Method Implementation Cost Status: 0 → Budget: 3m \$ Total monetary cost in US Dollars fo... Δ%: [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: Credibility - adjusted:		17 ± 0 % 34 %	67 ± 0 % 134 %	33 ± 0 % 66 %	
Value To Cost:		5.30	2.50	-1.50	



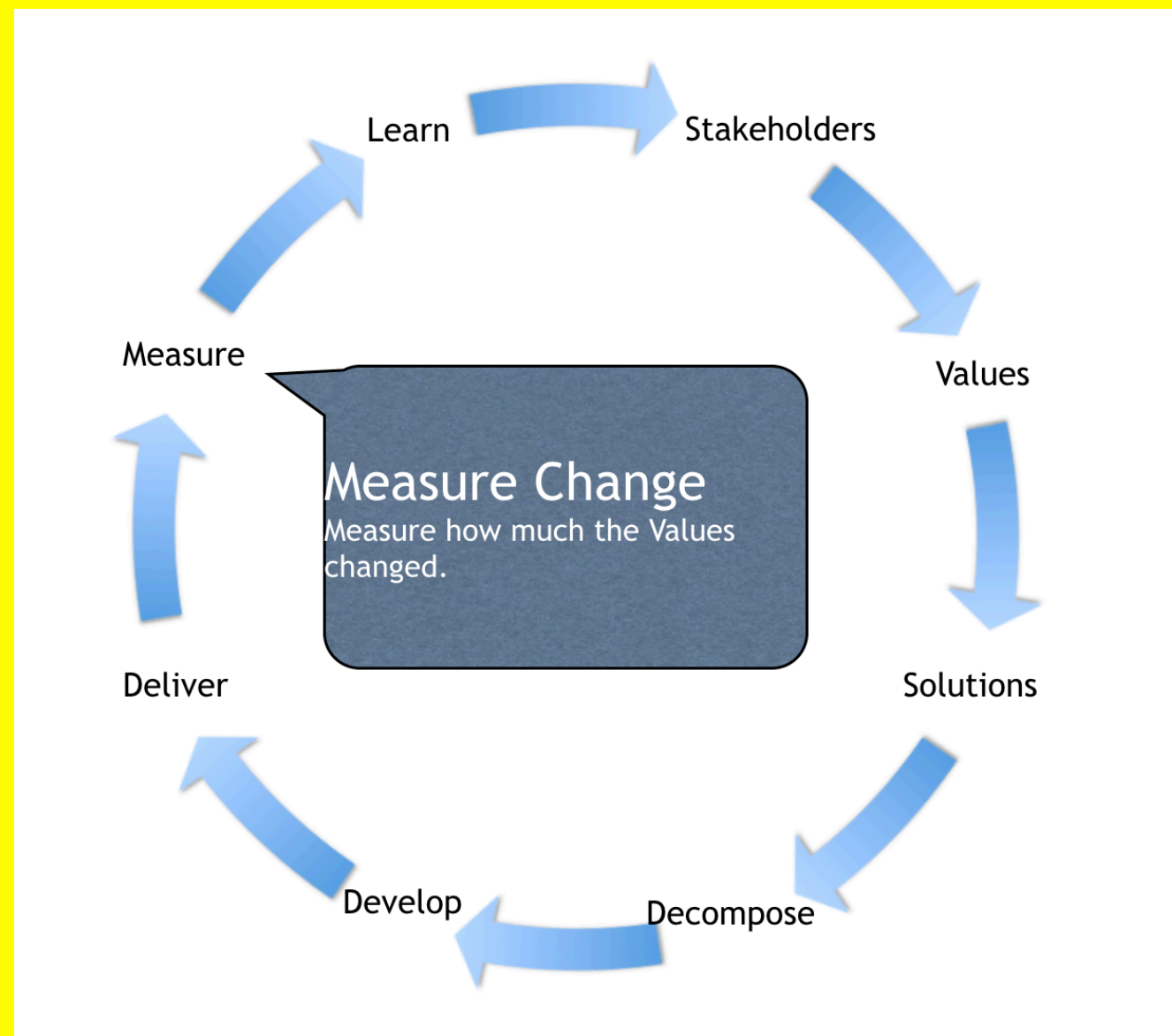
The numeric relation between ends and means: Engineering Analysis.

Basic Structure of an Impact Estimation Table

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



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)

Stakeholders

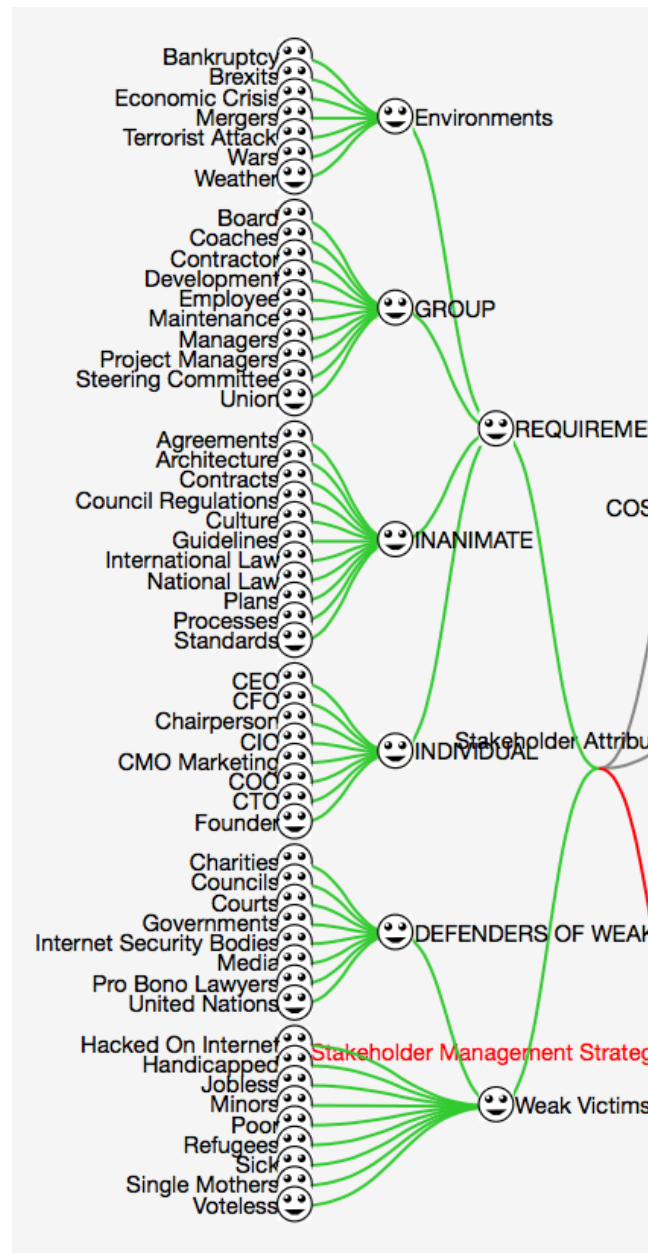
Identify your
critical stakeholders

the ones that have
one or more critical needs,

that if you fail to deliver *them*,

your project/product

might well fail



Values

Solutions

Decompose

Develop

Measure

Deliver

Learn

Learn

Stakeholders

Which numeric improvements
do stakeholders need,
critically?

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

Values

Solutions

Define both failure
and
success numerically

and

keep learning what
those
critical numbers are
continuously

Requirements		
\rightarrow <u>Project Timeliness</u>	=:	
Status: 10 \rightarrow Wish: 5 %	Δ :	
% time overrun necessary to deliver ...	Δ %:	
[Project Cost Size = { Medium (\$10k -...)]	$\%?$:	
30th June 2017		
\rightarrow <u>Building Security</u>	=:	
Status: 50 \rightarrow Wish: 10 % l...	Δ :	
% of [Emergency Types] which in fact...	Δ %:	
[Emergency Types = { Earthquake },	$\%?$:	
30th June 2018		
\rightarrow <u>User Productivity</u>	=:	
Status: 15 \rightarrow 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 $\%?$:	
\rightarrow <u>Method Implementation Cost</u>	=:	
Status: 0 \rightarrow 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:		

Measure

Deliver

Develop

Decompose

Learn

Stakeholders

Solutions
(designs, architectures,
strategies)

Measure

Values

must be identified
and their total impacts on
critical objectives
and
constraints

Deliver

Solutions

must be estimated
reasonably

(order of magnitude)

Develop

Decompose

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

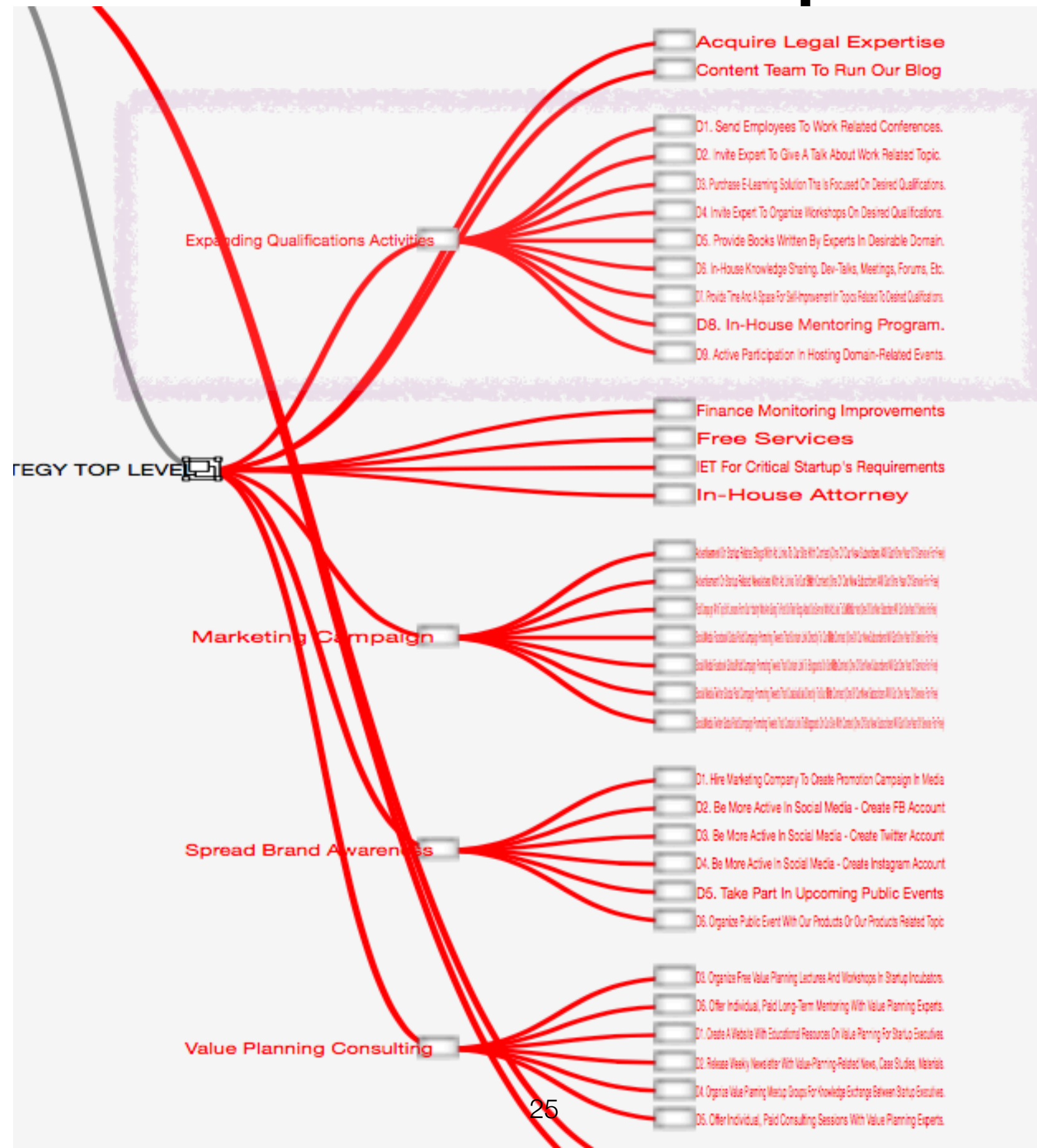
Requirements	<input type="checkbox"/> Incentivise	<input type="checkbox"/> Tea Kiosk	<input type="checkbox"/> Daily Danger Checks	Sum
(->) Project Timeliness Status: 10 -> Wish: 5 % % time overrun necessary to deliver ... [Project Cost Size = { Medium (\$10k -...)] 30th June 2017	8 ± 0 -2 % 40 ± 0 % 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 % Σ7%: 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) 134%	-1m ± 0 -1m \$ 66 ± 0 % 66 % (x 0.0) 33%	ΣΔ%: 117 ± 0 %
Sum Of Development Resources: Credibility - adjusted:	Σ%: 17 ± 0 % Σ7%: 34 %	67 ± 0 % 134 %	33 ± 0 % 66 %	
Value To Cost:	5.30	2.50	-1.50	

Solution Decomposition

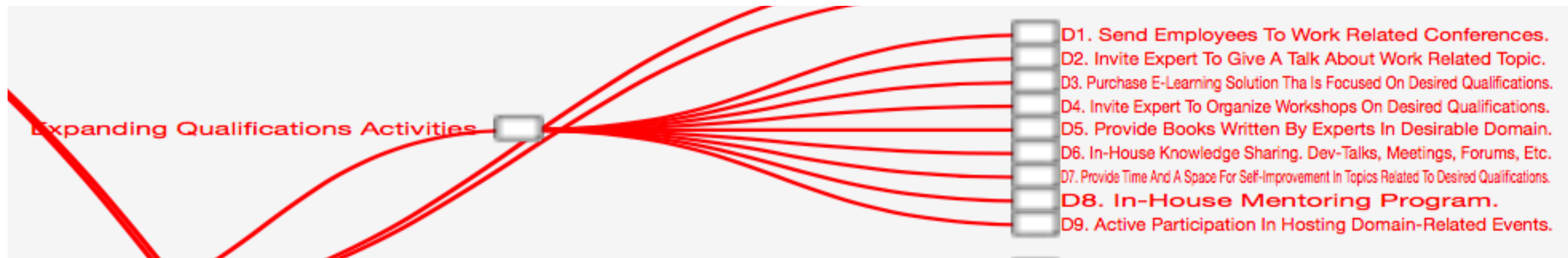
Example

2017 Polish Export Example

Several Solution Decompositions



Detail of 1 Solution Decomposition



Criteria for Decomposition

1. Each decomposition will deliver measure value to at least 1 stakeholder requirement
2. Any decomposition (D1... Dn) can be delivered independently of any other.

Expanding Qualifications Activities

[Solution Idea](#)

(✎ by gilbguest9 - 2 months ago)

Is Part Of: [STRATEGY TOP LEVEL](#) [Group](#)

Consists Of: [D1. Send Employees To Work Related Conferences.](#) [Solution Idea](#) [D2. Invite Expert To Give A Talk About Work Related Topic.](#) [Solution Idea](#) [D3. Purchase E-Learning Solution Tha Is Focused On Desired Qualifications.](#) [Solution Idea](#) [D4. Invite Expert To Organize Workshops On Desired Qualifications.](#) [Solution Idea](#) [D5. Provide Books Written By Experts In Desirable Domain.](#) [Solution Idea](#) [D6. In-House Knowledge Sharing. Dev-Talks, Meetings, Forums, Etc.](#) [Solution Idea](#) [D7. Provide Time And A Space For Self-Improvement In Topics Related To Desired Qualifications.](#) [Solution Idea](#) [D8. In-House Mentoring Program.](#) [Solution Idea](#) [D9. Active Participation In Hosting Domain-Related Events.](#) [Solution Idea](#)

Summary: A set of conferences, workshops and presentations lead by experts and other activities that aim t...

Description:

D1. Send employees to work related conferences.

D2. Invite expert to give a talk about work related topic.

D3. Purchase e-learning solution tha is focused on desired qualifications.

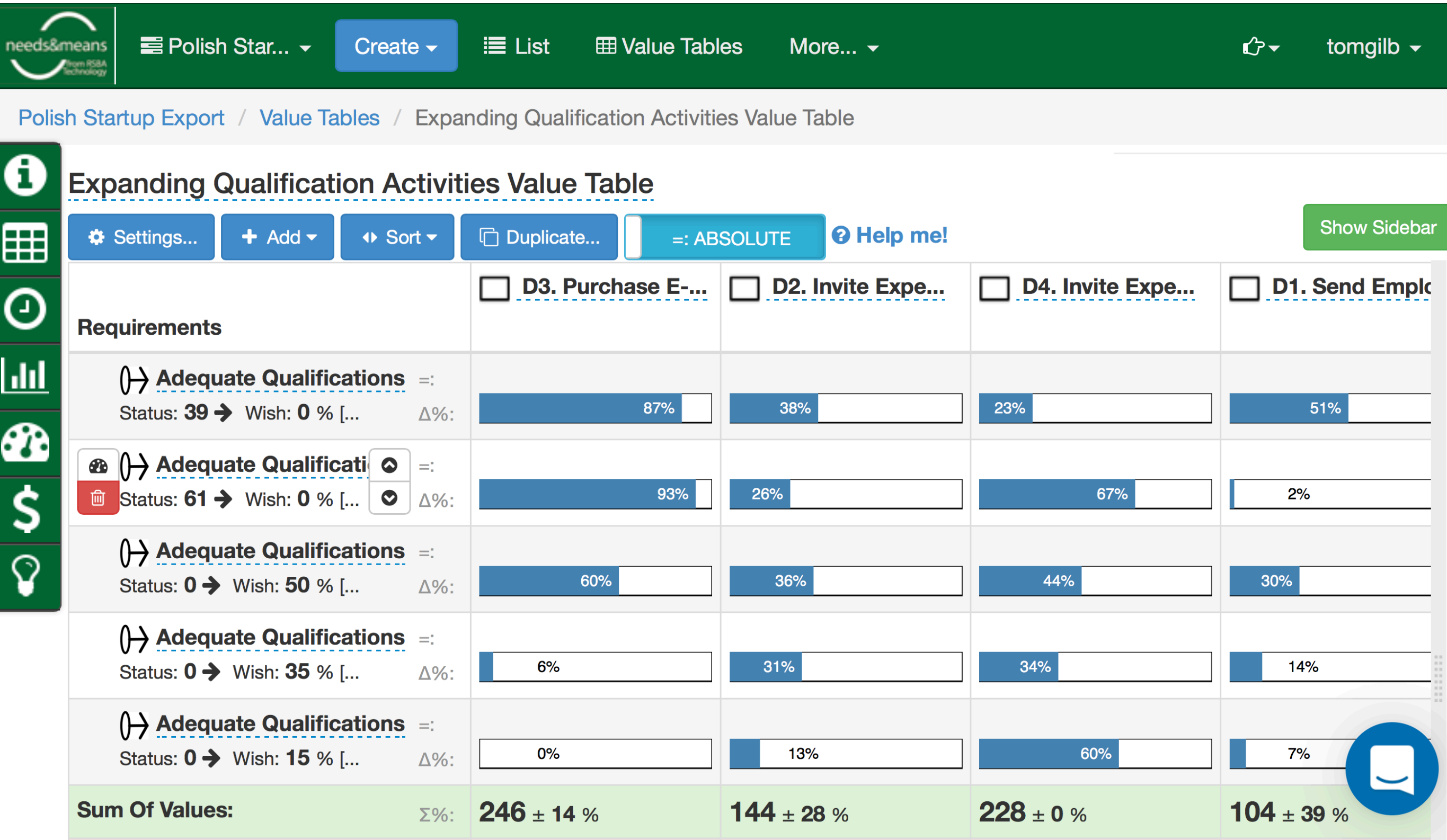
D4. Invite expert to organize workshops on desired qualifications.

D5. Provide books written by experts in desirable domain.

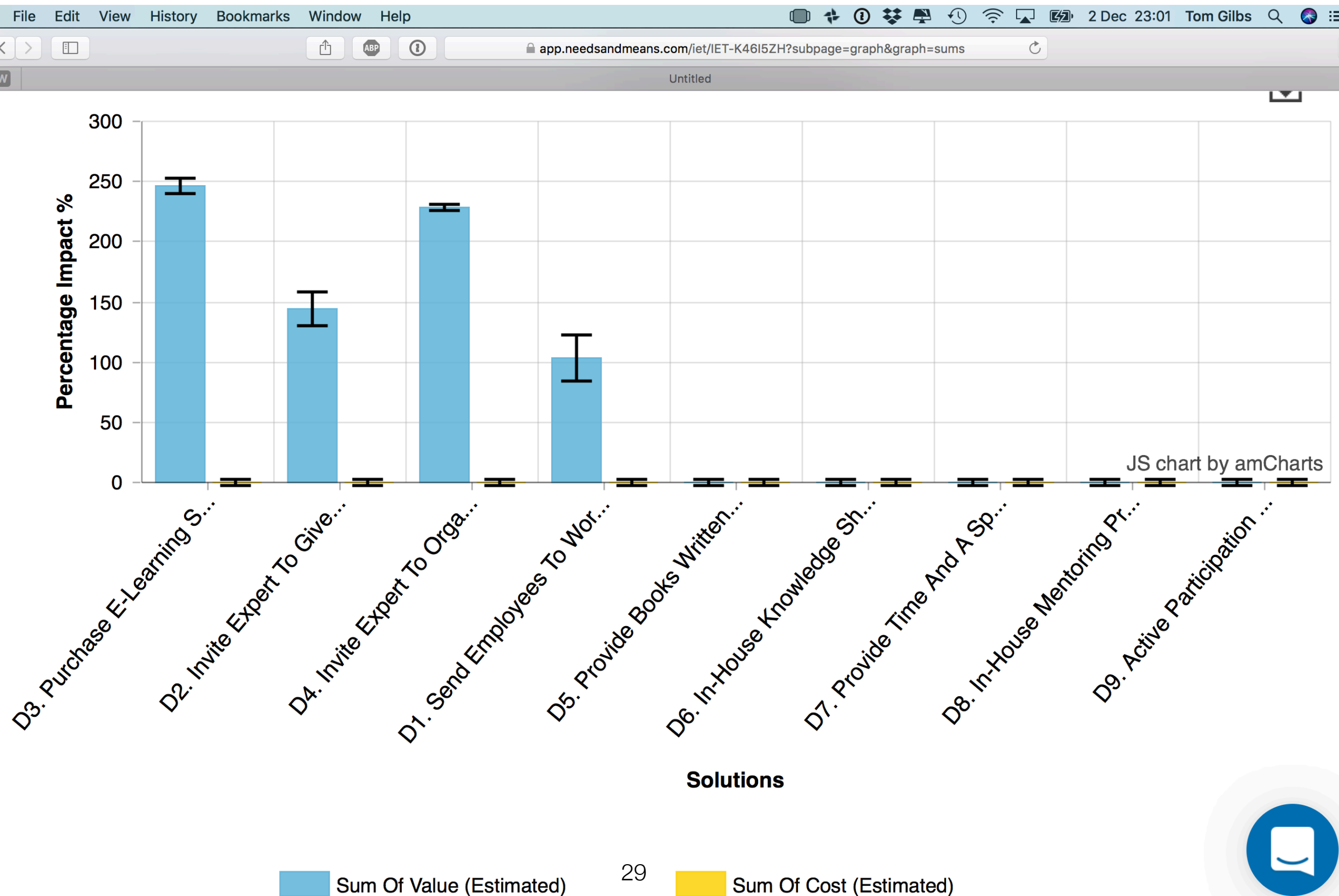
D6. In-house knowledge sharing. Dev-talks, meetings, forums, etc.

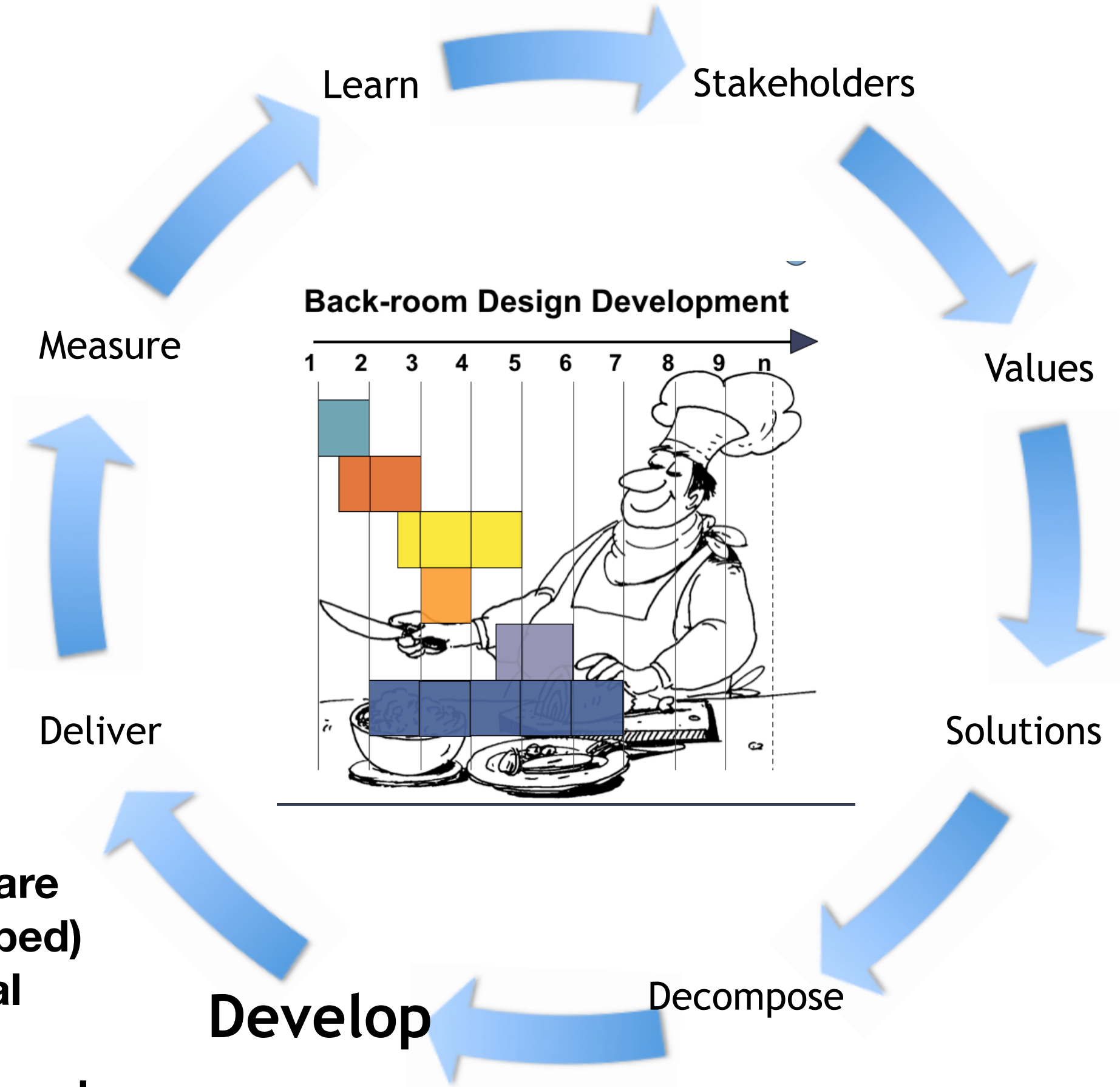
D7. Provide time and a space for self-improvement in topics related to desired qualifications.

We can estimate the value of the decomposed architecture, on different quantified requirements targets

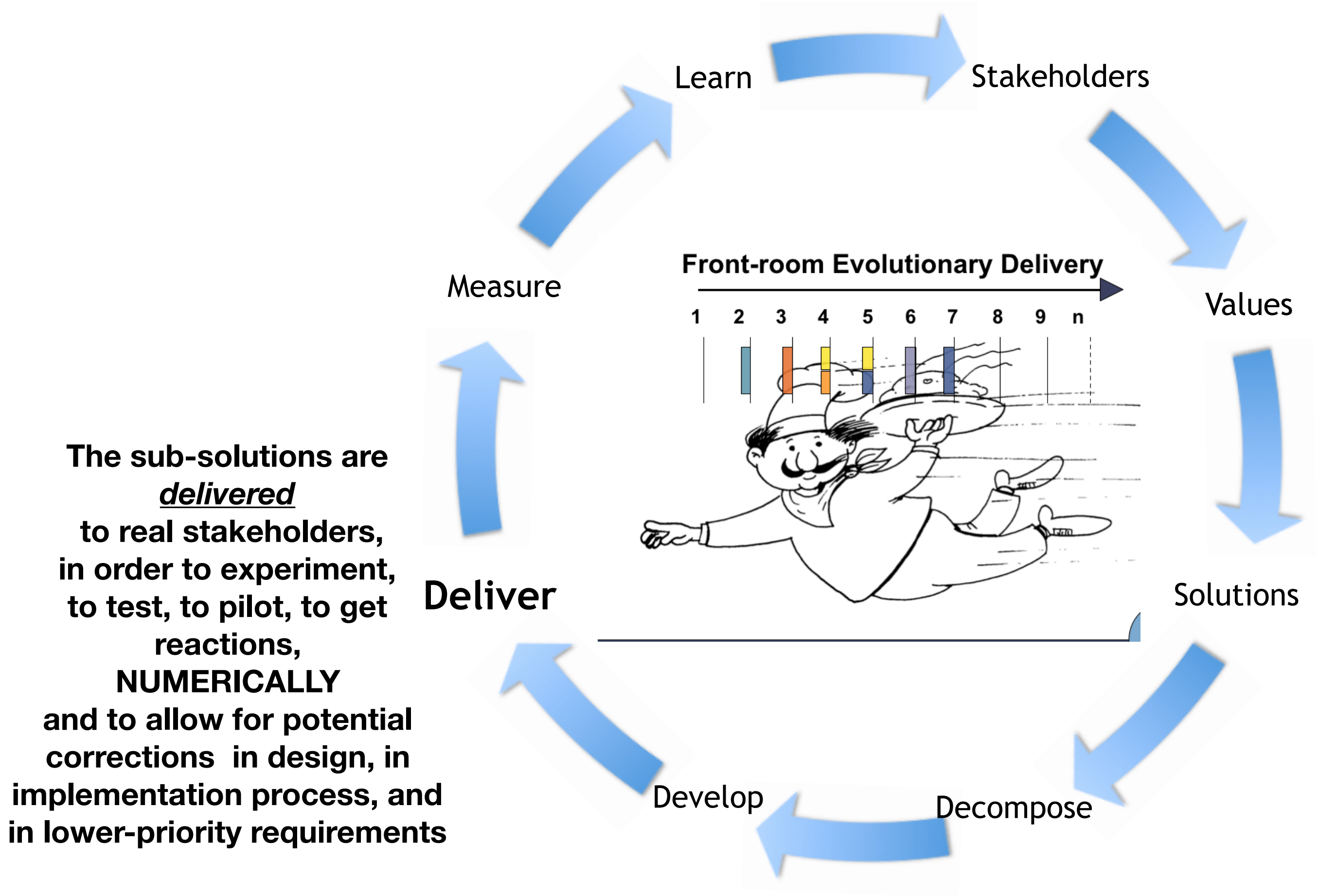


We can simplify presentation and even *automatically* sort design options into *delivery priority* (Product Owner Engineering)



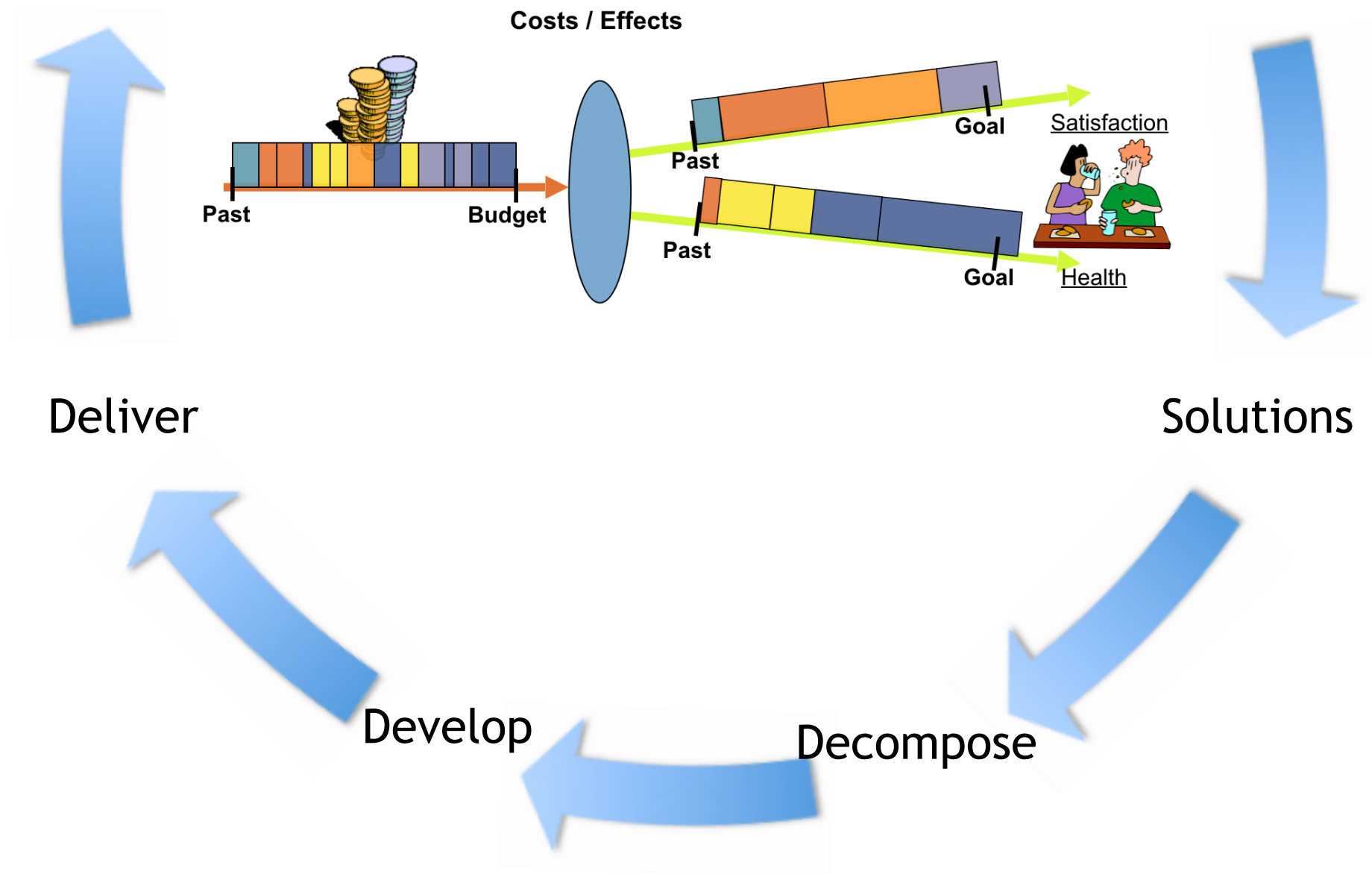


The sub-solutions are made ready (developed) for delivery to real stakeholders, next week and every week. Or in about 2% of budget/ deadline increments



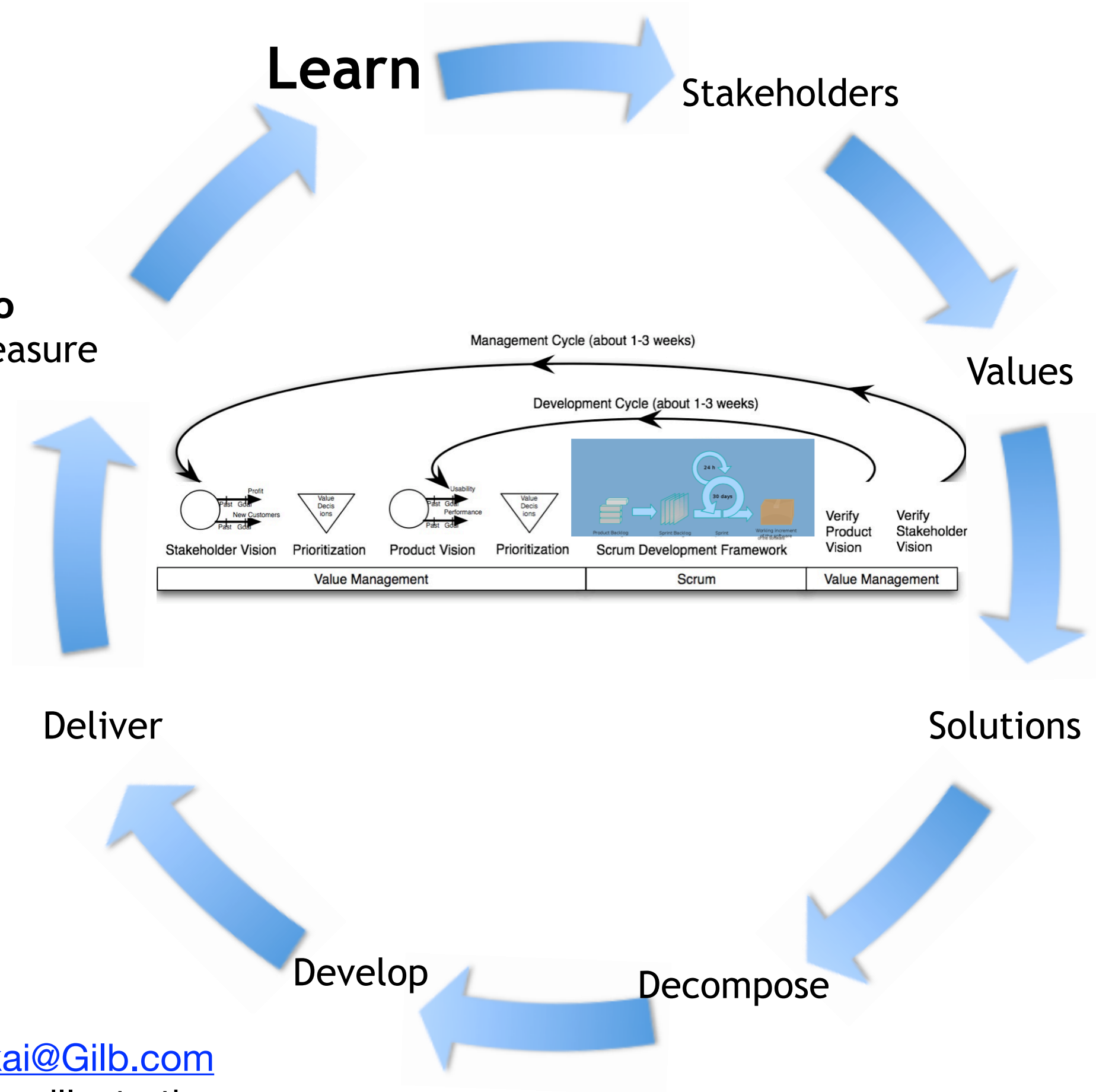
**The sub-solutions are
measured as to effect
 on
 all the
 top
 stakeholder
 critical
 objectives,
 and
 on their critical cost
 increments,
 with a view to improving
prediction of
 final cumulative costs**

Measure

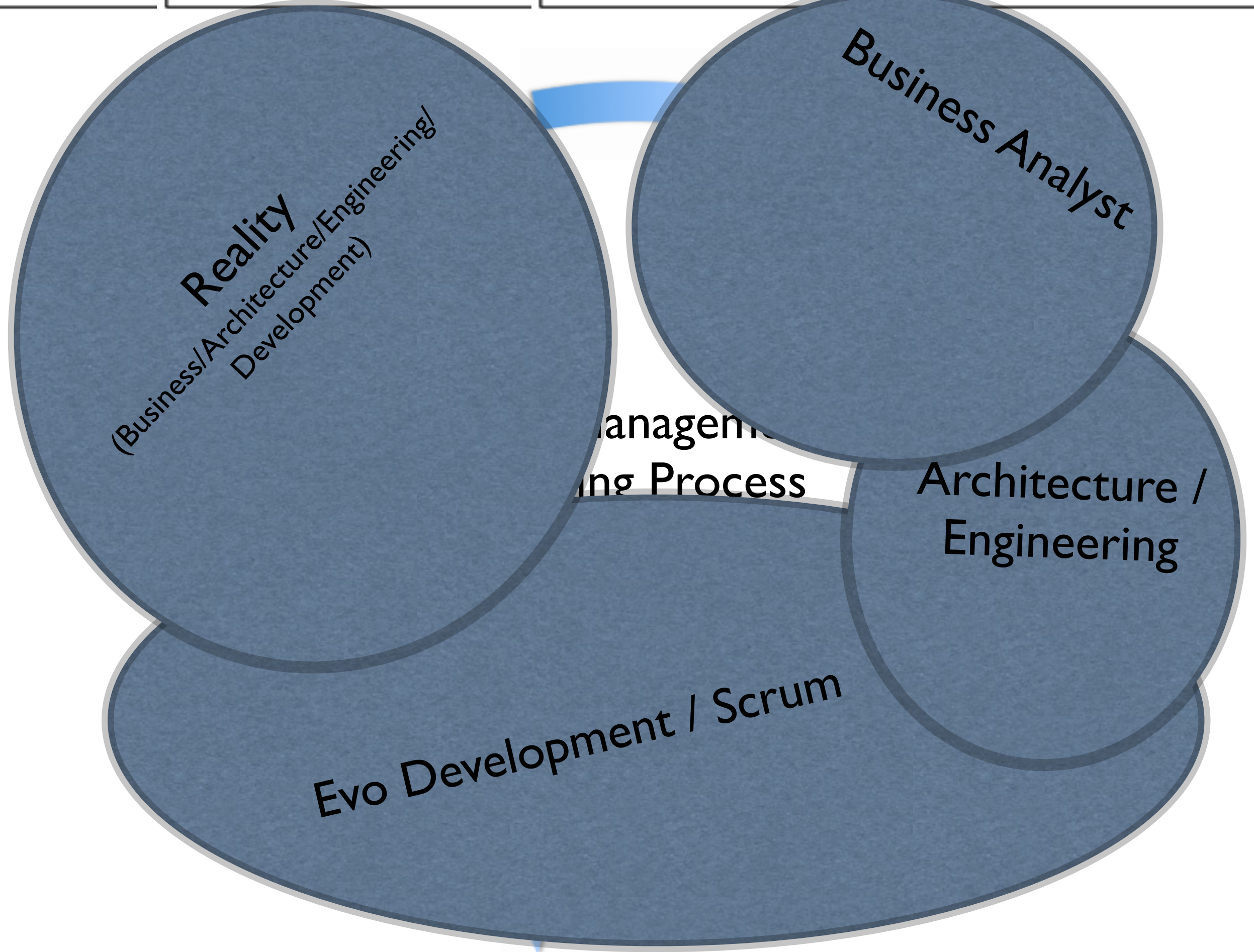


From the measurements,
and
other feedback
from stakeholders

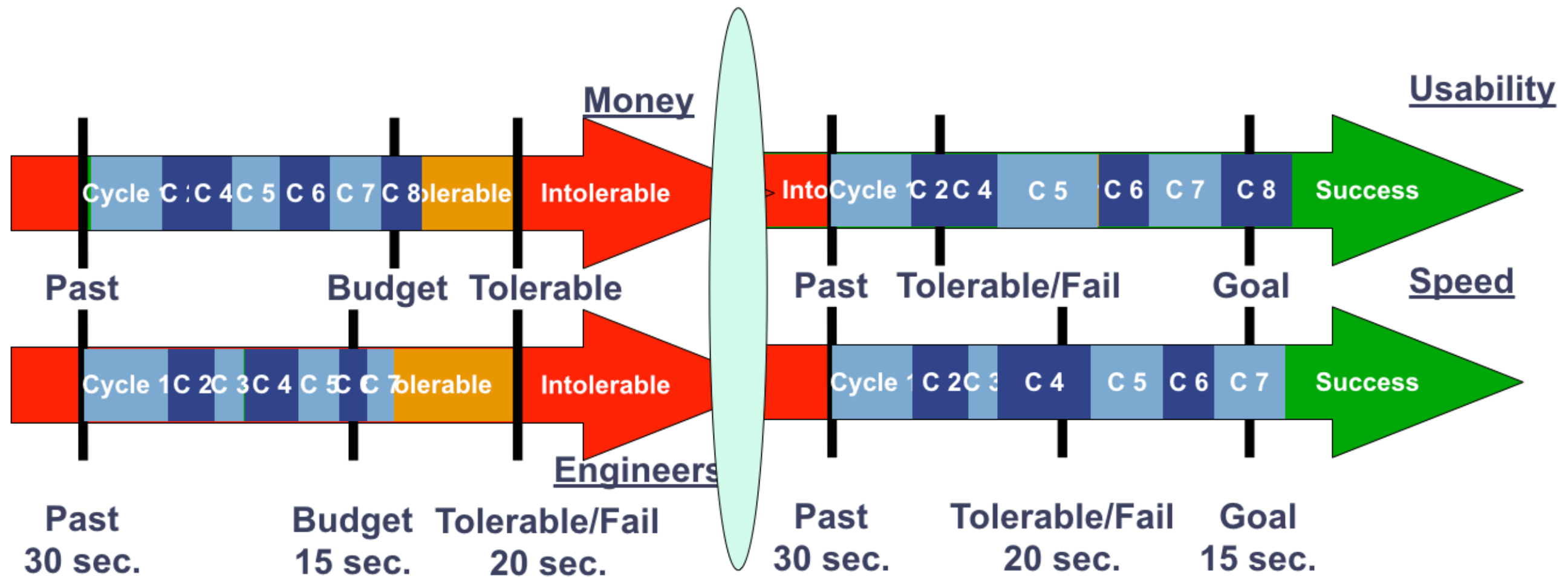
Learn what you need to do
to avoid failure
and to succeed



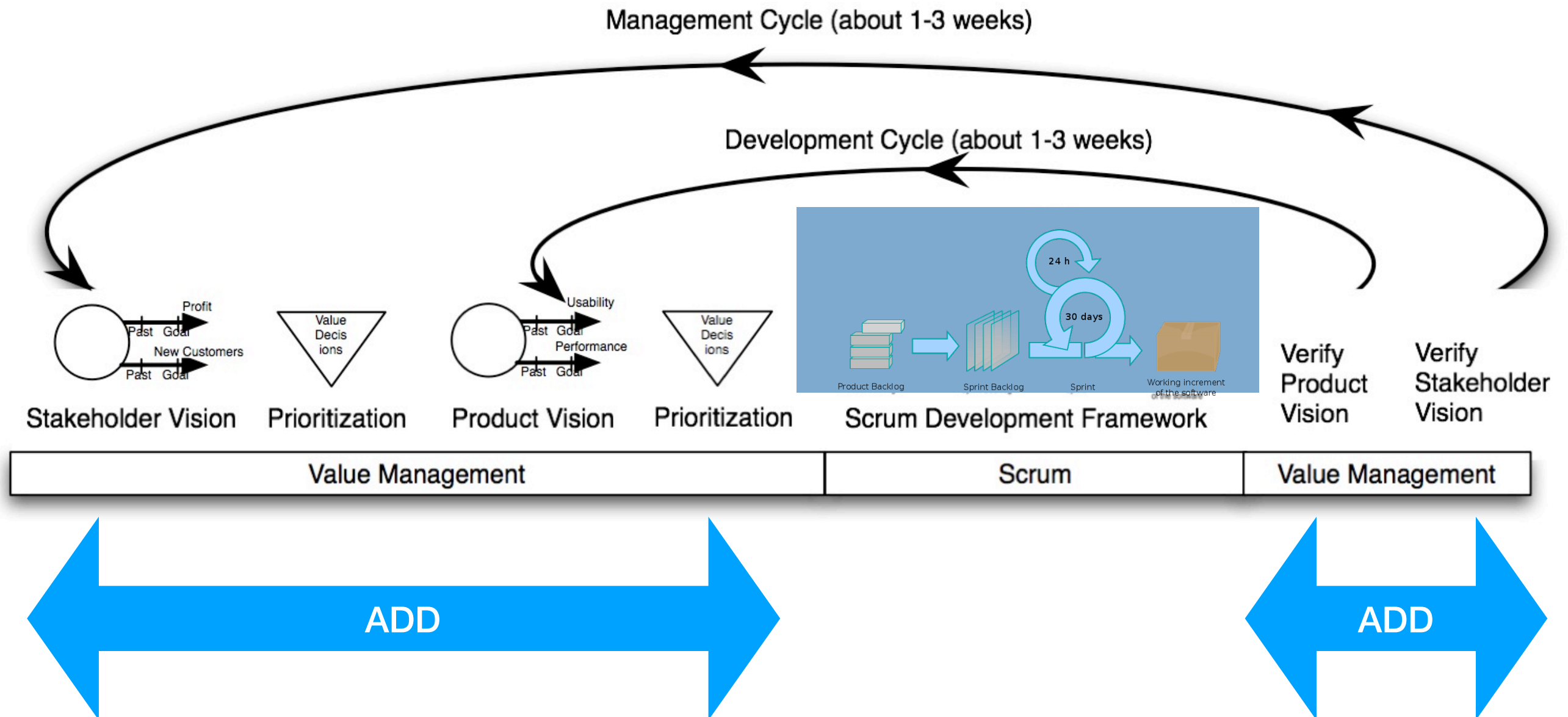
These 2 diagrams are © kai@Gilb.com
2017, as well as several other illustrations
used in this talk



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

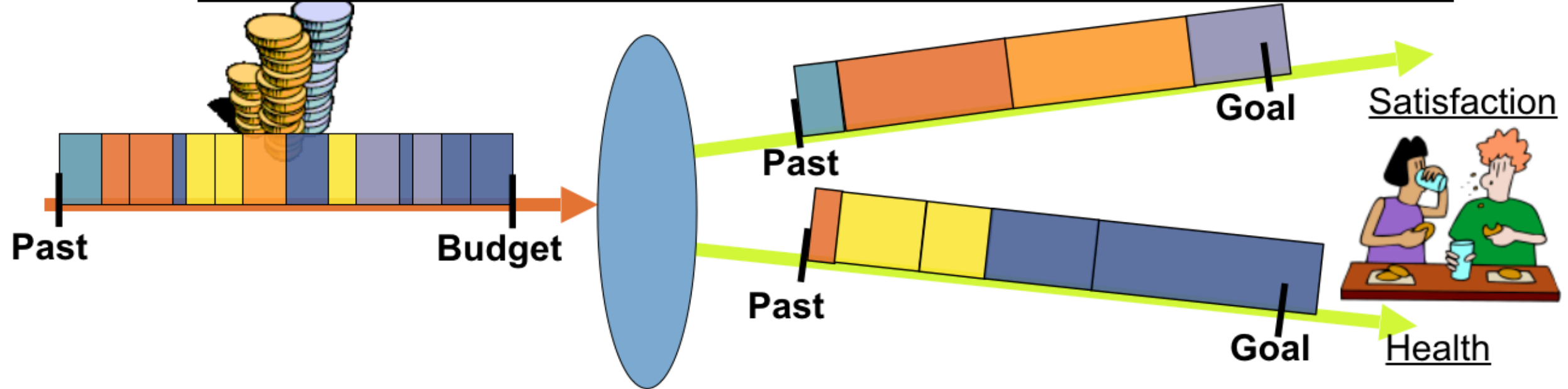


We need to add: ‘Value Management’: Quantified, Engineering, Not just ‘coding’

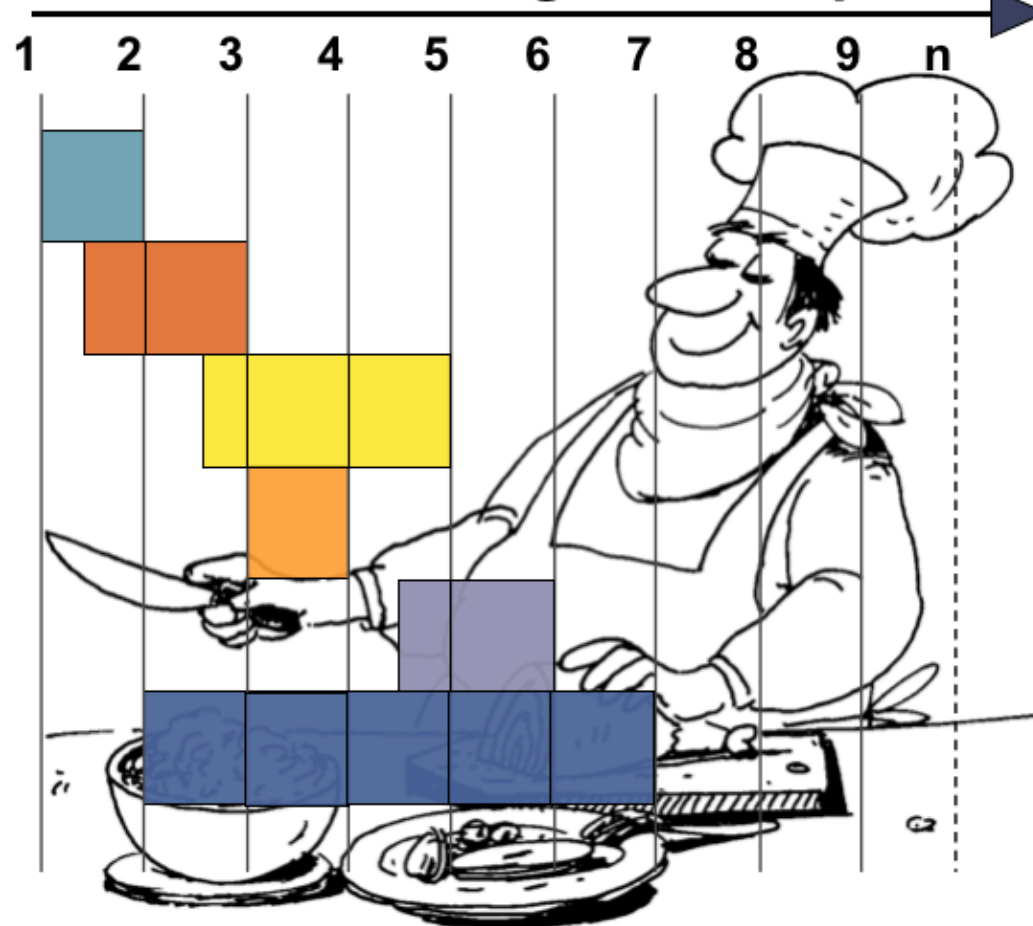


Copyright: Kai@Gilb.com

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

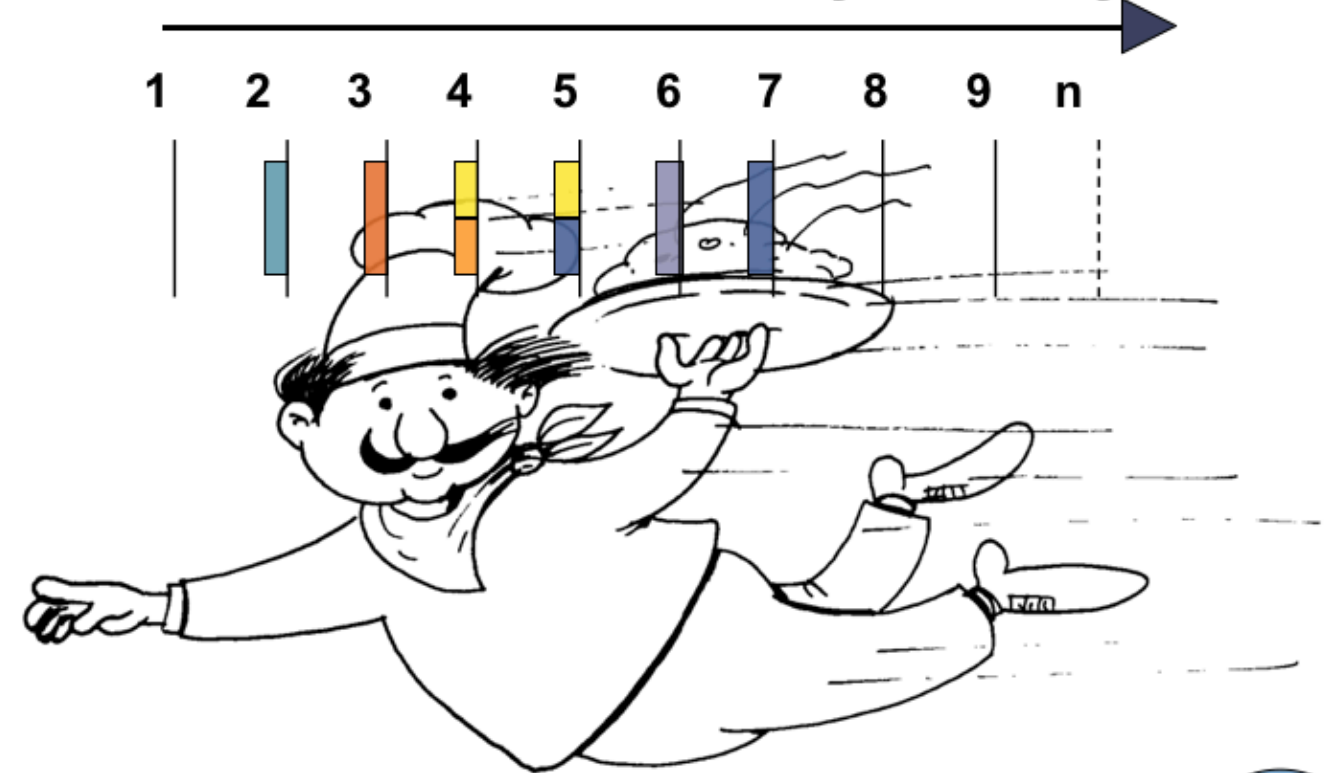


Diagram © kai@gilb.com 2017 & earlier



‘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

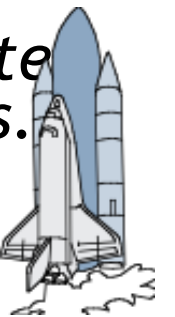
inIBM sj 4 80 p.419
In

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%!]. 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 (1980) they reported:



- “Software Engineering began to emerge in FSD” (IBM Federal Systems Division, 1980) “...without
- **in 45 incremental deliveries** ...cts -

cost overruns, late deliveries, unreliable and incomplete software

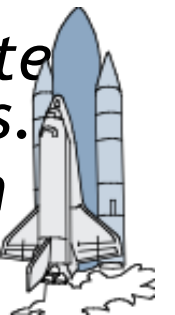
- Today [Ed. 1980!], management has learned to expect on-time, within budget deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over seven million words of program and data for eight different processors



- distributed over seven million words of program and data for eight different processors
- Note 2 ...eries [Ed. ...dget

- A more ...
- - When ... years of ... million byte ... en projects. ... ne at all in
- - There ... the po

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



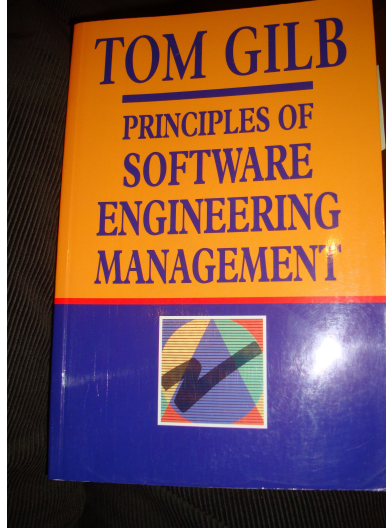
Mills on 'Design to Cost'

- “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





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)

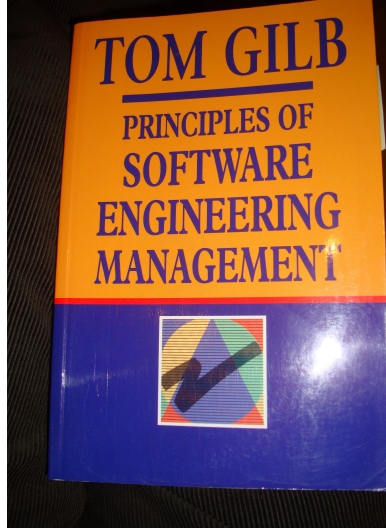
Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77

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. . . introducing design-to-cost software technical management. . . developing a design.

- He goes on to capability.' When a software increment is developed concurrently with the design.

'Design is an iterative

It is clear from the text that the process is iterative 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**

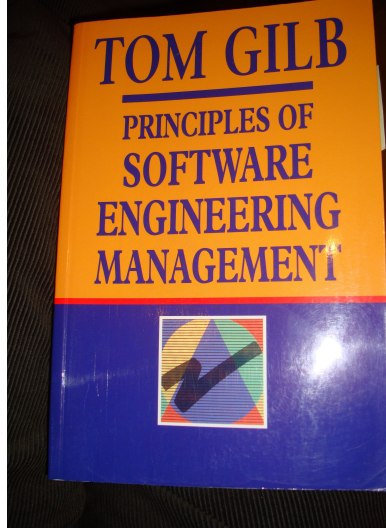
management farther by an integrated way to ensure that the process by Figure 7.10] consists of

by sacrificing 'planned of each increment can proceed



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
balance between cost
the task, and increase
increment becomes a

'When the development

Source: Robert E. Quinnan

This text is cut from C

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

in seeking the appropriate
thus reducing the complexity of
d as the true cost of the

increments is computed.' (p. 474)

1980, pp. 466-77



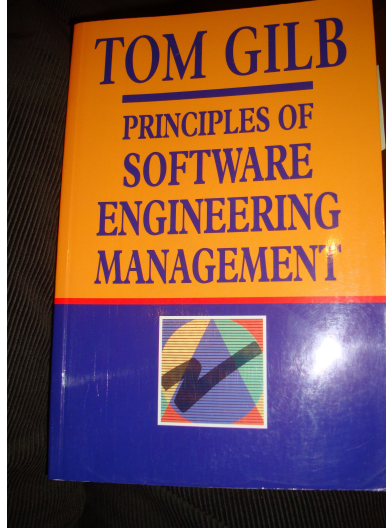
Quinnan: IBM FSD Cleanroom *Dynamic Design to Cost*



**Design is an iterative
process**



Quinnan: IBM FSD Cleanroom *Dynamic Design to Cost*

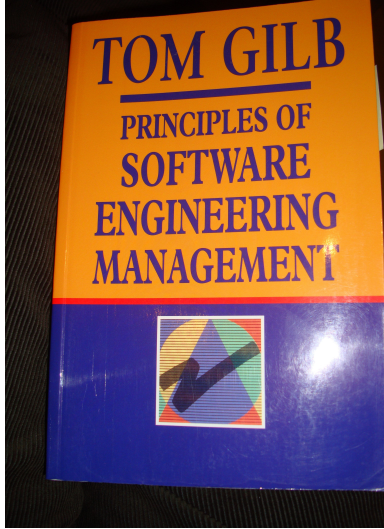


Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

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

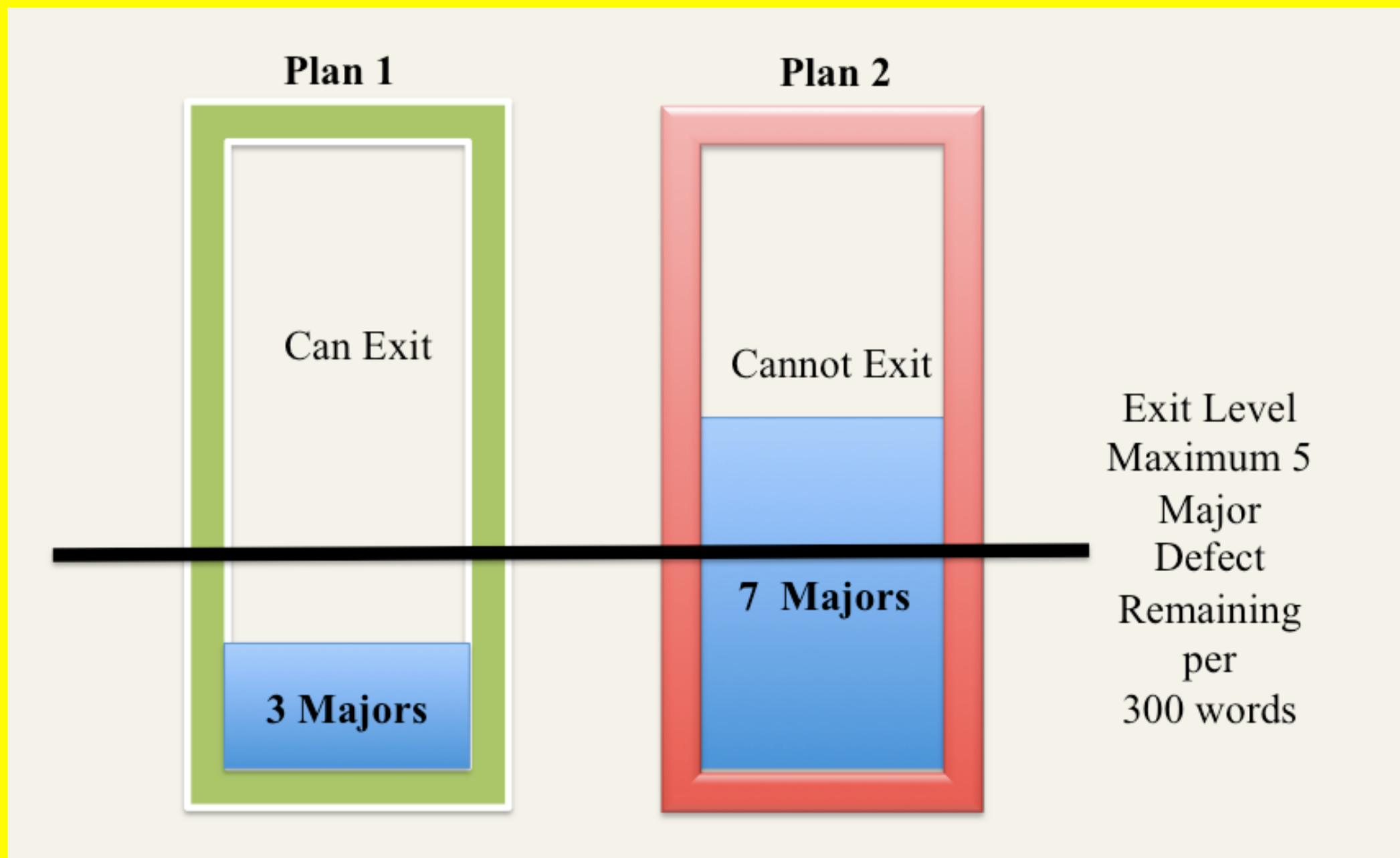


Quinnan: IBM FSD Cleanroom *Dynamic Design to Cost*



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.**

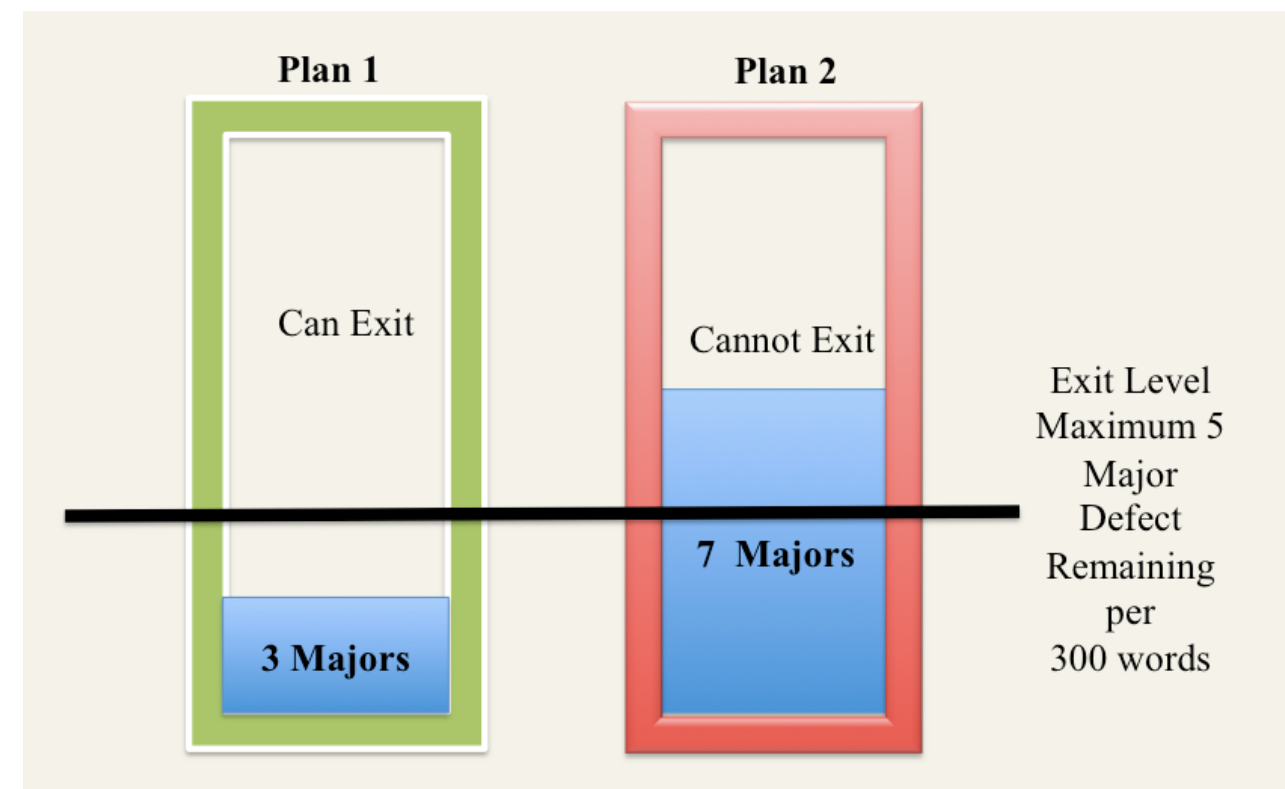
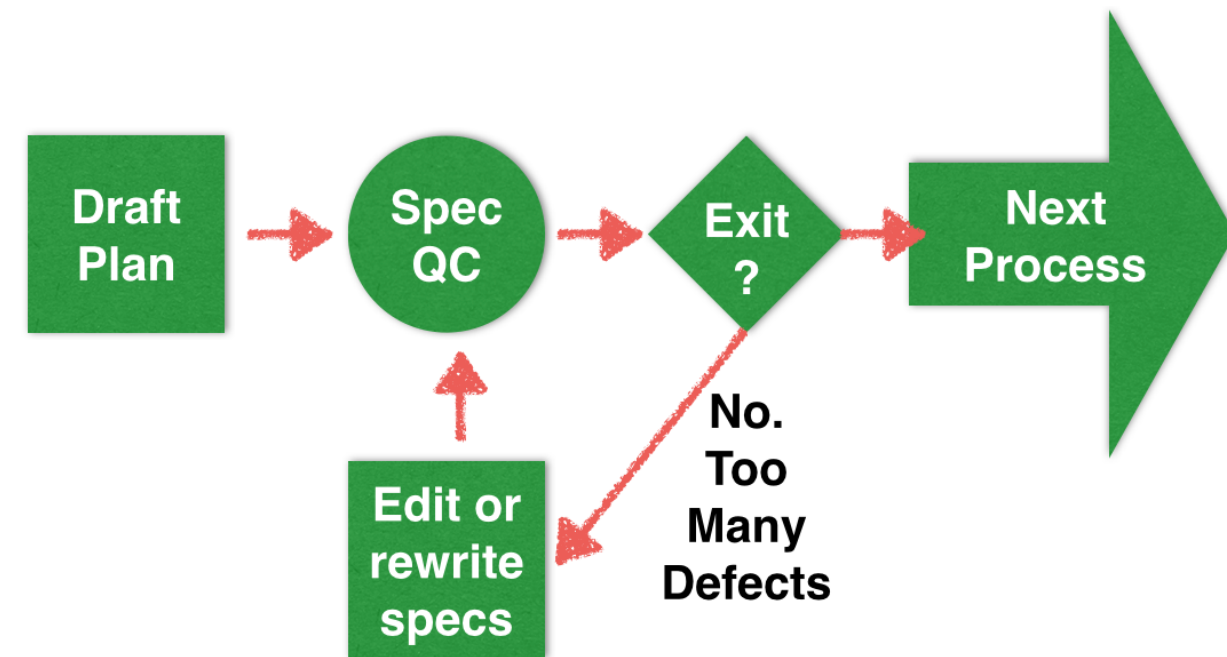


4. Measuring Development Specifications Quality: Lean Quality Assurance

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)



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

Our 'Quality Engineering' in practice at Intel for 20,000 engineers and 17 years

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=iccg_i_2013_3_10_10012

foundation. This paper includes the background and validation results from a third generation product ("Gen 3") that was characteristics of the first product: it ran on similar platforms,

We need to 'engineer' quality into software

- You can expand your current use of metrics 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



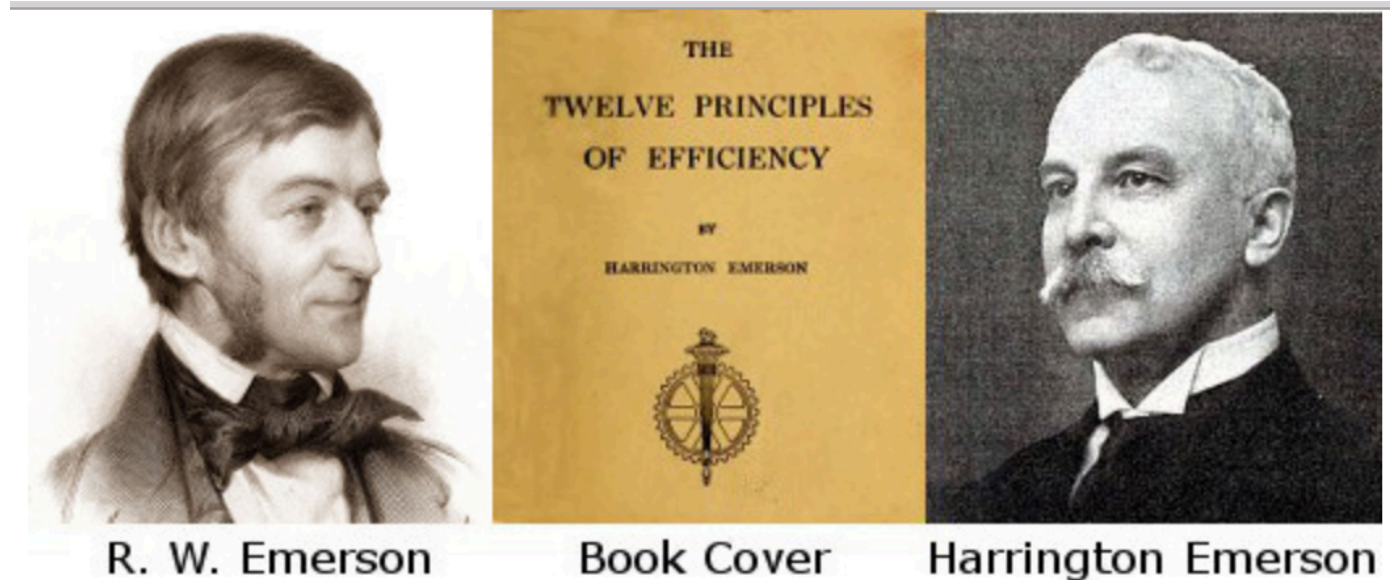
Get a free e-copy
of 'Competitive Engineering' book.
<https://www.gilb.com/p/competitive-engineering>



Get 50% discount on Value Planning
Use this link: <https://goo.gl/MB6kaR>
Coupon Code: CONNECT

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”.
- - Emerson, Harrington
- (Not as thought, R W E)



My 'Planguage'

Requirements Concepts <-CE book

Planguage Concept Glossary **401**

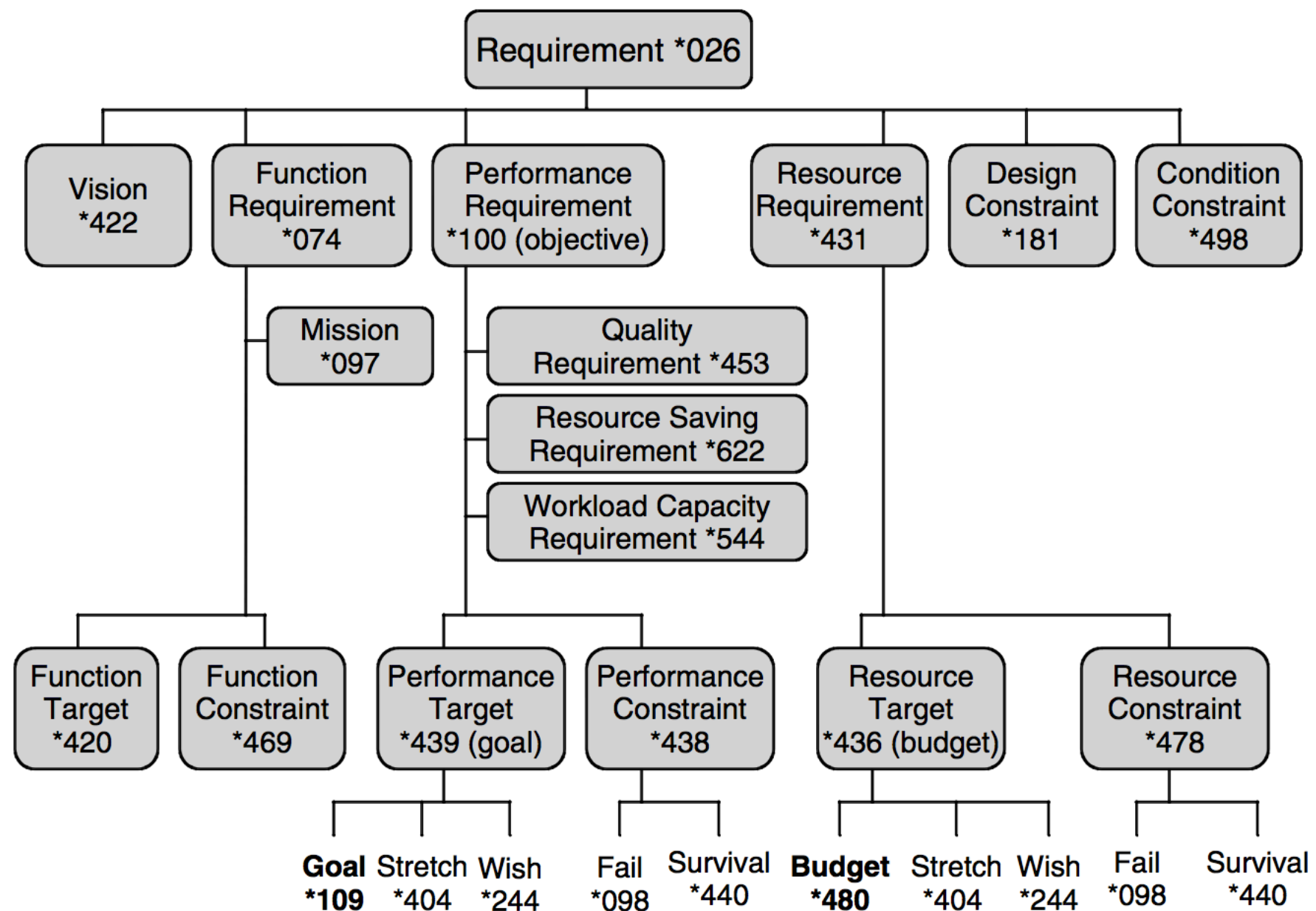


Figure G20