



Value Re- quirements



Figure Cover, Source: "To Catch a Butterfly: Epistemic Miracles of Serendipity. The.xel.io

<http://te.xel.io/posts/2018-03-04-to-catch-a-butterfly-epistemic-miracles-of-serendipity.html>

Introduction

This book will help you analyze and specify the most important requirements for your project, how to quantify success 'values', so you can manage them, and how to structure them, so as to reflect your complex reality.

This book will help you with the following work processes: stakeholder analysis, value requirement specification, requirement quality control, requirement prioritization, risk management, clear communication, and systems-level thinking.

It will help you set the stage for design, estimation, contracting, and project management.

The method is based on our advanced planning language, 'Planguage', which specializes in values, qualities and costs like no other alternative requirements method.

Simply the best, for those who must succeed, and cannot afford to fail to deliver real value.

<i>Introduction</i>	2
Chapter 1. What are `Value Requirements` ? (VR)	9
<i>What other kinds of requirements do we need to consider?</i>	10
<i>Part of the reason we are not treating these non-value requirements in detail here is:</i>	13
<i>What kinds of 'requirements' are NOT on our 'Value Requirements' agenda ?</i>	13
<i>Can User Stories be used?</i>	14
Chapter 2. More detail on the nature of Value Requirements (VR)	16
Chapter 3. The Scale In more detail.	19
<i>3.1 The essence of a 'Scale specification'. And a 'motivation' to use it.</i>	19
<i>3.2 Minimal Scales.</i>	24
<i>3.4 Developing a Scale from an Ambition Level.</i>	28
<i>3.5 Scale Parameters</i>	32
<i>3.6 Defining Other Terms in a Scale</i>	38
<i>3.7 Scale Libraries</i>	40
<i>3.8 Multiple simultaneous Value Requirements</i>	45
<i>3.9 Here is an example of a single complete Value Requirement Specification, with all the extra supporting detail we will discuss below.</i>	46
<i>3.10 Details of a Scale (Air Quality example), with Scale Parameter Definition.</i>	48
<i>3.11 More on Multiple Value Requirements</i>	49
<i>3.12 Knowing when to decompose a value into sub-values, using sub-scales.</i>	51
<i>3.13 Good Scales and bad Scales.</i>	53
Chapter 4. The 'Meter' Parameter	55
<i>4.1 The 'Meter' specification is a defined process for measuring the numeric value level, on a Scale.</i>	56
<i>4.2 The Meter as a high-level test process: why this is useful.</i>	58

4.3	<i>The multiple quality and cost attributes of a Meter</i>	60
4.4	<i>Sufficient Meter Accuracy for Purpose</i>	62
	Chapter 5. Benchmarks	64
5.1	<i>The purpose of 'Benchmarks' In a Value Requirement Specification.</i>	64
5.2	<i>'Past' as a Benchmark</i>	66
5.3	<i>'Status'-level Benchmark: real time value delivery tracking.</i>	69
5.4	<i>Record Benchmark</i>	70
5.5	<i>'Ideal-level' Benchmark</i>	72
5.6	<i>'Trend-level' Benchmark</i>	74
	6. Scalar Constraint levels.	75
6.1	<i>'Tolerable-level' Constraint.</i>	75
6.3	<i>Constraints are a Dynamic Prioritization Tool</i>	78
6.4.	<i>Several different Tolerable levels might be appropriate for different circumstances.</i>	80
6.5	<i>There are other types of 'Scalar Constraints' Defined</i>	81
	Chapter 7. Scalar Target-levels.	82
7.1	<i>Wish-level Target</i>	82
7.2	<i>Goal-level Target</i>	90
7.3	<i>Comment on the Multiple Goal Example Above (Fig. 1.43).</i>	92
7.4	<i>Stretch-level Target</i>	95
	Chapter 8. Background Specifications.	97
8.1	<i>The General Purposes of Background Specifications.</i>	100
8.2	<i>Risk Management with Background Specs</i>	102
8.3	<i>Prioritization using background specs</i>	106
8.4	<i>Responsibility and Motivation with Background specs</i>	111
	Chapter 9. Making Use of the Value Specification	114
9.1	<i>Dialogue with Stakeholders</i>	114
9.2	<i>Negotiating Priorities for Values</i>	118

http://news.mit.edu/2018/natural-resource-negotiations-for-mutual-gains-bruno-verdini-0621	119
9.3 Determining the higher-level value of value increments	120
Figure 1.56 : Gee I think I'll change my mind about the Solar Panels, and the Tennis Court	122
We need to use this kind of thinking about stakeholder value increments. Will it pay off?	122
Or 'don't even think about it!'	122
https://www.thesun.co.uk/money/7178628/home-renovations-affect-house-price/	122
9.4 Contracting and Proposals	123
9.5 Handover to architecture and design engineering.	125
9.6 Handover to testing and measurement, with Value requirements.	128
9.7 Presentation and approval for steering committees, based on Value Requirements.	131
9.8 Quality Control: of value requirements.	133
9.9 Defect Level Measurement and Exit Control	136
9.10 Management Reviews, in a Value Driven culture.	138
9.11 Estimation of resources to deliver Value levels	140
9.12 The 'Design to Value', and 'Design to Cost'.	142
Chapter 10. Presentation of Value Specifications	144
10.1 Presentation to Stakeholders	145
10.2 Value Requirements: Presentation To Project Managers	148
10.3 Value Presentation to Steering Committees	152
10.4 Value Presentation to Spec QC Teams	152
10.5 Value Presentation to Architecture	155
10.6 Value Presentation to Legal Team	157
10.7 Value Presentation to Sub-suppliers	159
Chapter 11. Levels of Value Specifications	160
11.1 Vision Levels, Vision Engineering	161
11.2 Fundamental Value Requirements	164

Chapter 12. Resource Requirements Specification	166
12.1 <i>The need to understand the incremental costs of value requirements before approving them.</i>	166
12.2 <i>Some basic categories of Resource Requirements</i>	168
12.3 <i>The difficulty of estimation of costs up front (ref. F)</i>	171
12.4 <i>The possibility of getting control over real costs by design to cost</i>	172
12.5 <i>The possibility of getting control of some costs by smart 'Dynamic Architecture' and design</i>	175
12.6 <i>The possibility of getting control of the value-to-cost ratio by decomposition; and then by prioritization of high-efficiency designs.</i>	178
12.7 <i>The possibility of getting control over costs, by negotiated reduction of initial Value level, and initial date ambitions</i>	180
12.7 <i>The possibility of getting control over costs by Subcontracting</i>	185
Chapter 13. Change Control of Value Specifications.	187
13.1 <i>The concept of a specification Owner.</i>	187
13.2 <i>Annotation of change source, and time stamps</i>	190
13.3 <i>Ways of controlling the whole of the specification</i>	194
Chapter 14. A Review of Requirement Methods Compared to Planguage	196
14.1 <i>General observations of methods for specifying Value Requirements</i>	196
14.2 <i>A Checklist for understanding capabilities of value requirement Specifications</i>	197
Chapter 15. SOME COMMENTS ON SPECIFIC METHODS	199
15.1 <i>User Stories (ref. H, a)</i>	199
15.2 <i>Use Cases</i>	201
15.3 <i>Earned Value Management (EVM)</i>	205
15.4 <i>Functional Requirements and Non-functional Requirements</i>	206
15.5 <i>Balanced Scorecard</i>	209
15.6 <i>Quality Function Deployment (ref. I)</i>	214
15.7 <i>Togaf</i>	216

15.8 Zachmann Framework	219
15.9 UML: Unified Modeling Language	220
15.10 Design Sprints (ref. g)	221
15.11 The 'Evo' Project Startup Week (ref. K) : Values Driven Start	222
15.12 OKR (K) Objectives and Key Results.	224
15.13 MoSCoW: Prioritization Method.	226
15.14 CMM: CMMI, Capability Maturity Model	229
Chapter 16. A Briefing on Use of Value Specifications for Design and Architecture	230
16.1 Architecture Engineering: disciplined and logical architecture.	233
16.2 Design Engineering: specialist areas with clear consideration and balance for all other values and constraints.	236
16.3 The Value Table for overview and synchronization	240
16.4 Design-attribute feedback incrementally, and adjustment of value levels, timings, and resource budgets, in mid-project.	241
16.5 The Evo Value Cycle	243
Chapter 17. Formal Standards for Value Specifications	244
17.1 Competitive Engineering as Planguage Standard.	245
17.2 ValPlan as a tool containing standards	246
17.3 Rules	248
17.4 Processes	249
17.5 Exit and Entry Processes	250
17.6 'Concept' Standards: several levels	252
17.7 Teaching and enforcing standards using Spec QC and Exit levels	254
Chapter 18. ValPlan: and apps for Value Specifications	255
18.1 Brief history of Planguage tools.	255
18.2 ValPlan the app.	258
18.3 Open Endedness for Tool Builders.	260
18.4 GraphMetrix	261

Chapter 19. Stakeholders and the planning environment	263
19.1 Determining your planning environment (ref. L)	265
19.2 Determining your stakeholders	269
19.3 Eliciting stakeholder needs and converting them into Requirements.	272
19.4 Understanding stakeholder priority, power, competence and motivation.	273
Chapter 20. Summary of Value Requirements	274
Appendix 21 Book References	275
Appendix 22. Papers References, with Free URL Links	277
Appendix 23. Slides and Talk Video References, with Free URL Links	280
Appendix 24. Concept Glossary	282
Appendix 99. Notes on editing the book and Versions.	289

Chapter 1. What are `Value Requirements` ? (VR)

Value Requirements are the most important requirements for any project. They are the main purpose, and main justification, for a project. They are the *stakeholder's* values.

Value requirements start life as value 'attributes' *needed* by 'stakeholders'. No project can deliver all 'needed' values, by a deadline. No project will find all stakeholder values to be *worth* delivering.

So all value requirements start life by being *acknowledged* as *possible* delivery candidates. But VRs need to go through an *evaluation process* to determine that we can prioritize them for real delivery.

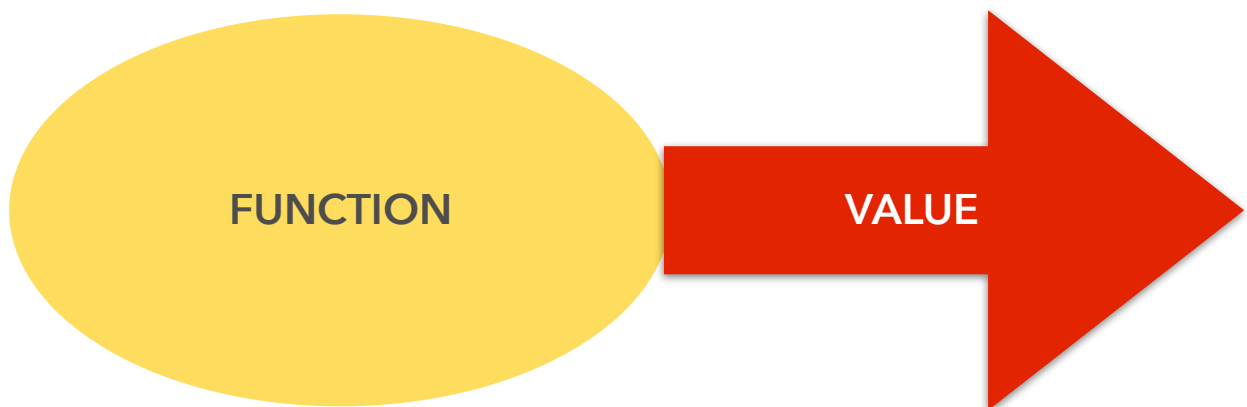


Figure 1.1 : A Value is a variable level of performance for a function. Represented graphically as an arrow emanating from a Function symbol. This is a simplified model, with a *single* value arrow. Reality is always that *multiple* values are needed concurrently.

What *other* kinds of requirements do we need to consider?

There are several other 'requirement types' which you will need to consider, but we will *not* treat these in detail here. They are treated elsewhere (Competitive Engineering, CE).

Here is a list of other (not 'value') requirement types.

Function Requirements: WHAT the system must DO.

The Function 'keyed icon' is: (any oval keyboard symbol) 'O'



Figure 1.2 : a system function, represented as an oval shape in Planguage icons.

Planguage icons are a formally defined set of symbols, like music notes, or maths symbols, which represent systems engineering concepts¹. And which are independent of human spoken languages.

¹ <http://www.gilb.com/DL386>, Full Planguage Concept Glossary. Including Icons.



Resource Requirements: limitations on any kind of resources (people, time, money).

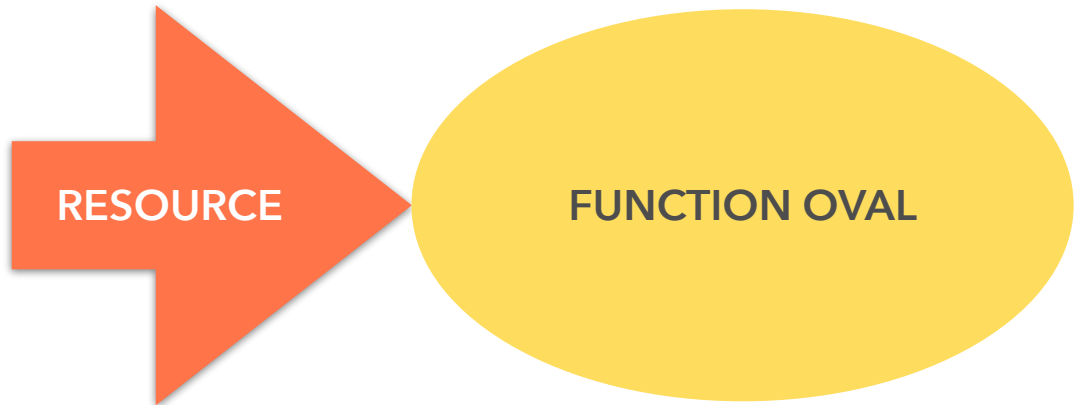


Figure 1.3 : a single resource arrow giving a function the potential of some values.

The 'Resource' 'keyed icon' is: ->O

Binary Constraints: legal constraints, design constraints, anything that must either be **done**, or **not done**.

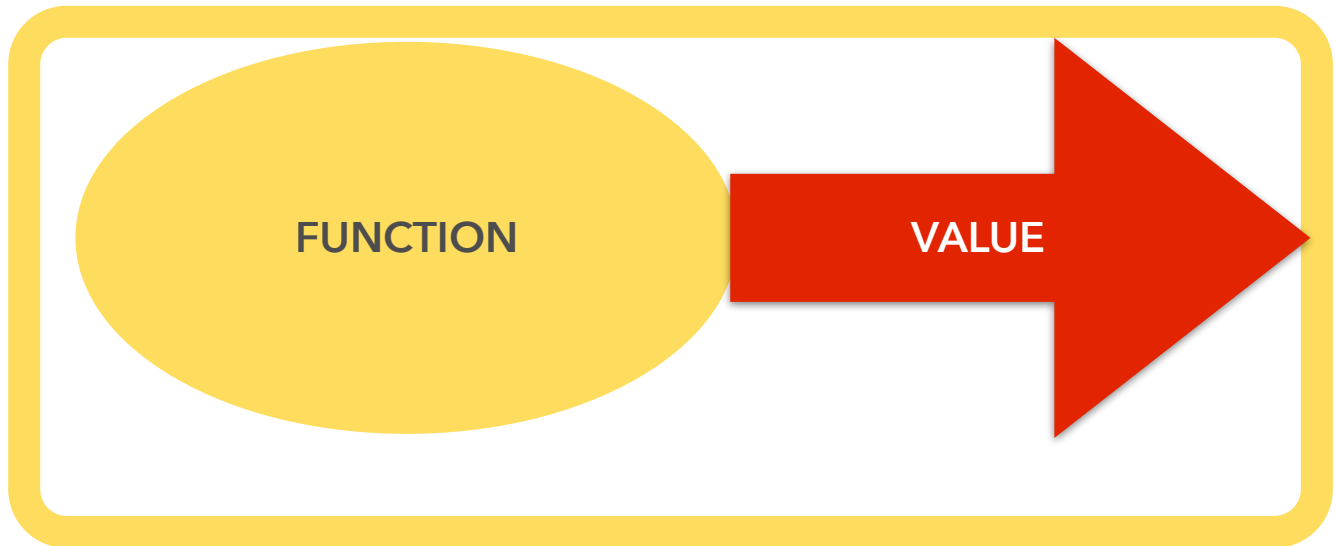


Figure 1.4 : A constraint (the rectangular shape) which the system must be 'within'.

Another type of constraint a 'Scalar Constraint' will be dealt with in this book. It is a numeric limit (not too hot, not too cold) on a Value Scale (at least 99.99% Availability) or a Resource Scale (finished absolutely latest ny end of year, no matter what).

Part of the reason we are *not* treating these non-value requirements in detail here is:

1. They are fairly well understood.
2. They are comparatively *simple*, compared to Value Requirements.
3. We want to focus on the less-understood, but extremely-decisive, Value Requirements. The main point of all projects.

What kinds of 'requirements' are NOT on our 'Value Requirements' agenda ?

1. User Stories (see Chapter 15.1)
2. Use Cases (See Chapter 15.2)
3. Simple written text (like 'better safety')
4. Designs *claiming* to be Requirements

Can User Stories be used?

User stories do not contain enough information to serve as value requirements. But you can add on the sort of information we present in this book, to make them serve as value requirements (ref.

A, US)

User Stories have a useful structure:

1. A stakeholder (narrowly called 'user')
2. A 'story', which functions as a sort of requirement. But is not detailed enough for serious purposes.
3. A justification for the story, which is good practice. But can be improved upon.

My initial advice for people who have to deal with User Stories is to write them up as a Planguage statement, and then proceed to derive more-useful detail from it.

For example:

Usability:

Type: Value Requirement.

User Story: As an expert user I want shortcuts to save me time. <-
US030719.

Scale: Average cycle time in minutes for a [Task] by a [User].

Pro Level: **Wish:** 6 minutes, Deadline = End Next Year, Task = Expert Complex Tasks, User = Expert.

Comment: in translating the user story we have carefully avoided the 'shortcuts' which is an amateur 'design' suggestion. We have focused on the stakeholder value of saving time, and left the detailed design, to achieve that end, to a professional UX designer.

Specification example 1A: the user story is cited, then translated into a value requirement (Scale and Wish statements). The 'scale parameters' [Task] and [User] are used to make a more general 'Usability' specification than the 'expert user' in the user story, and to specify a wider range of tasks than the unspecified tasks in the user story. The result is that we can specify a wide variety of Usability value requirements.

We can for example add a statement to the Usability specification above like:

Beginner Requirement: **Wish:** 10 minutes, Deadline = Beginning Next Year, Task = Beginner Frequent Tasks, User = Beginner.

Specification example 1B: We added a second 'Wish' value requirement, to the Usability specification above. This has several advantages. We can now prioritize one of them, based on value and cost, and deliver the value *early*; without waiting for the other Wish to be completed.

Chapter 2. More detail on the nature of Value Requirements (VR)

- VRs are often qualities, '-ilities' like reliability, security, usability: 'How Well' the function performs. Often called 'performance attributes'.

VALUE SPECIFICATION TYPES

Specifications	
	Requirements: Future Needs
	Value Requirements: How Good
	Qualities: How Well
	Other Values: How Much
	Functions
	Constraints

Figure 1.5 Value Specification Types.

- VRs are, in systems engineering, classed as *Performance* attributes. 'How good' the function is. 'How Good' includes: 'how

well' (Qualities): but also 'how **much**'. For example speed, volume, frequency, sales, market share and savings. Values which we would not call 'qualities'

- VRs are always a 'degree of system performance' which is 'actually *valued* by some stakeholder'. If not valued, then by definition, it is of no worth to *any* stakeholder.
- VRs are always defined as a desired 'numeric range' or a '*point*', on a 'scale of measure', which means the value is a *numeric* value.
- In addition to a value requirement being a numeric level, that level must *also* be achieved by defined *times* and *conditions*, for the total requirement to be fulfilled.

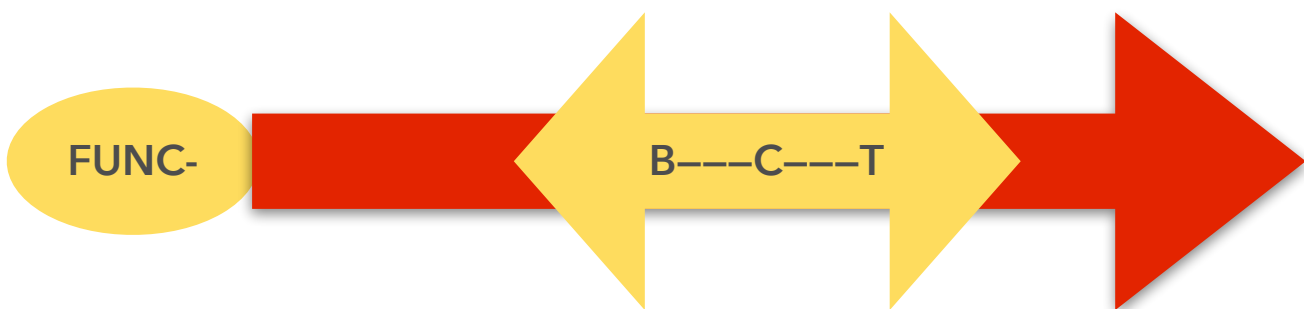


Figure 1.6 : on the Value Scale (Red Arrow symbol) a value requirement might be expressed by numeric constraints (C) and by numeric Targets (T). In addition a Benchmark level (B) might be added to a requirement specification to inform us of past and current levels of performance, for comparison with the required levels.

In our planning language, Planguage, this might be expressed like this.

Security:

Scale: % probability of detecting a hacker within 5 seconds.

Status: 10% last year. (Benchmark level)

Tolerable: 80% by End this year. (Constraint Level)

Wish: 98% by End Next Year. (Target Level)

Example 1C: a value specification.

Security is the reference tag for the entire specification.

Scale is a parameter in Planguage for defining a value variable, such as Security, so that the various levels of Security can be expressed numerically.

Status gives us the moving current change of status in the level.

Tolerable gives us the bare minimum level which is acceptable.

Wish is the stakeholder-desired, or stakeholder -needed, level of Security, on that Scale.

Chapter 3. The Scale In more detail.

3.1 The essence of a 'Scale specification'. And a 'motivation' to use it.

The most powerful, and useful, requirements method-detail that this book can inform you about is the 'Scale'.

This is because your 'Scale specification' moves you away from informal and fuzzy requirement specifications, and over to clear, logical, quantified methods of thinking about problems.

A Scale specification means you are moving your entire approach to projects from primitive and failure-prone communication with others, over to 'engineering' and 'science': over to a fact-based culture, to an evidence-based culture.

This takes a little more effort than, 'being lazy, and failing in your projects', but I assume you are reading this book because you want the skills to improve your capability and success, in your profession.

Another thing about the skill of 'defining values in terms of a Scale', is that you can use this skill for the rest of your life; on any kind of problem solving, and any kind of project. It is very good job security, in changing times.

Everything this book teaches is like that: it is based on universal ideas, which are quite independent of current technology, and independent of any profession you might undertake. I know from 60

Top Causes of Project Failure

Changes in organizational priorities: **41%**

Inaccurate gathering of requirements: **39%**

Change in project objectives: **36%**

Inadequate resources

Poor communication: **30%**



Figure 1.8 : Good value Requirements help us avoid project failure. All 5 of these failure causes are actually related to good 'value requirements'.

Source: <http://mobile.baselinemag.com/project-management/slideshows/why-some-companies-have-more-successful-projects.html>

years of professional experience, where I am still on top of my game; and able to impress top international professionals with these skills.

But this applies not only to me personally: thousands of professionals in major corporations have recognized the benefits of this skill, and chosen it.

One example is the over 21,000 Intel engineers, over about a 20 year period, who have voluntarily taken a 2-day training course in this way of specifying product values, and practiced their skills.

One simple measured result was 233% overall productivity increase (ref. Terzakis) (ref. A, Intel).

Figure 1.7 Intel

The good news for you personally is that, of the 100% of people who would benefit from these methods, there are probably less than 1% of them actually using them today. You will be more competent than the 99%, and hopefully you can help spread this culture of clear thinking?

Scale definition is the foundation, the core, of this quantified approach to value delivery.

This is true, as many of you are aware, in all sciences, and engineering disciplines. But somehow the business, politics, and planning disciplines have avoided quantification of many values and qualities, and just used 'nice-sounding words'.

We need to stop these immature practices, for serious projects, in order to improve our success rates.

The 'Scale' parameter specification is used to define success, and to define failure, so that we will know how far we are from success,

and how near we are to failure, at all times. And we can take real-time action to succeed, and to avoid failure.

The 'Scale' parameter is not only a clear definition of success and failure, but it is a tool to help us, as teams and groups of people, to communicate success and failure, so that all parties understand these ideas exactly the same; misunderstanding and misinterpretation should be near impossible.

This is important when projects are widespread, geographically, culturally and legally. And when change is the only assured constant.

The 'Scale' specification, is about an idea of *variability*: an idea for quantification: a platform for 'putting numbers on values', so as to express ideas of 'degree, of goodness' or 'badness', or 'improvement', or 'comparison'.

Scale is NOT an idea of 'measurement' (how to determine where you are now on that scale). We leave specification of *measurement* ideas to another parameter, the 'Meter', as in speedometer and voltmeter.

Once a Scale is defined, we reuse it for a very wide variety of purposes.

In addition, we can use more than one different measurement process (defined by Meter specs) for a single Scale. Quick and dirty, or more accurate and credible measurements, for example.

Our graphical symbol for a Scale is an arrow. *Value* Scales emerge from a system's function, and *resource* Scales point into the function - supplying a system with resources to drive the values to emerge.

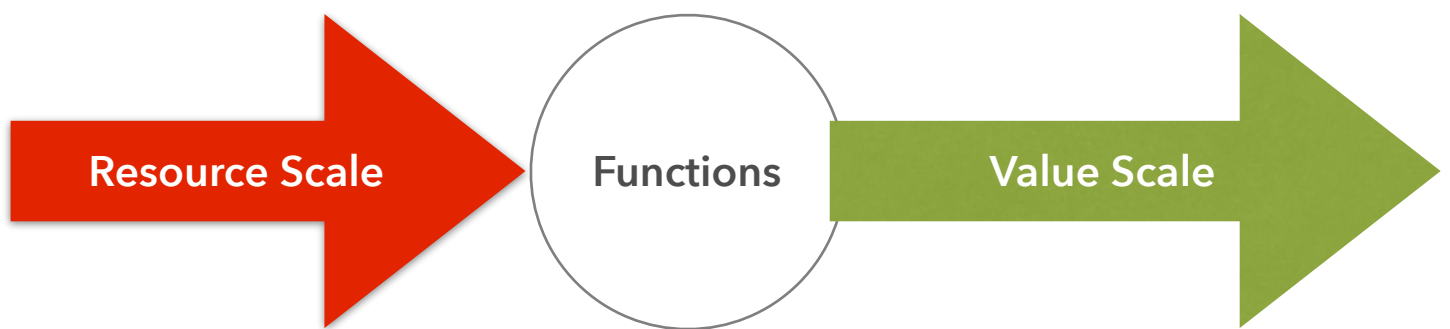


Figure 1.9: Functions (what a system 'does') need a supply of resources (like people, time, money) to produce values.

These *resources* are the prices we pay to develop and maintain 'values', and the 'value levels' we need, *when* we need them.

Initially, functions have no particular values associated with them.

Functions are independent of any particular values.

But for a function to be of use in a real world, it needs some value levels, of things like 'availability', 'usability', and 'work capacity'. If these values are zero or low, then the functionality would not be visible, or useful, in the real world.

If we improve value levels, for your product or service, to certain currently useful levels, we become 'competitive'.

If we improve value levels significantly beyond others in our market, then we can offer superior value to stakeholders, which might command their willingness to choose us, rather than competitors.

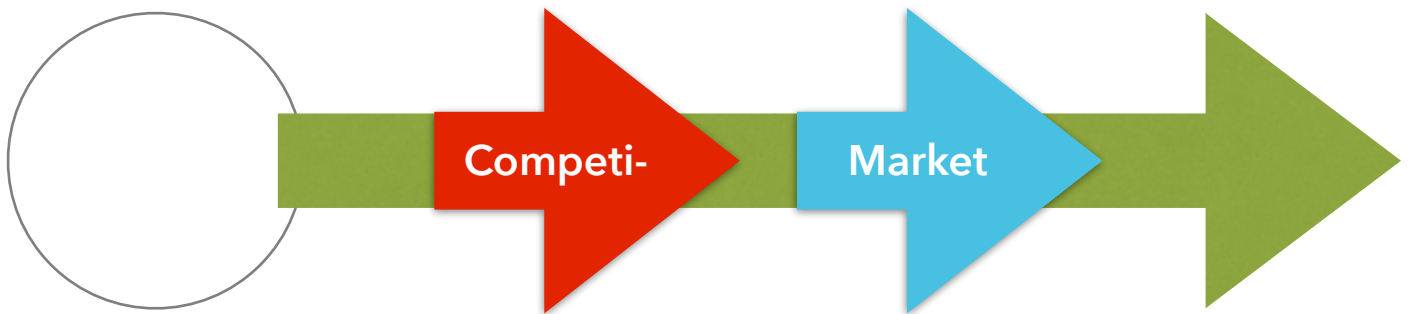


Figure 1.10 : for the same function, the basic market or business, like banking, or plumbing, or Yoga Training, you can plan to deliver a better level of one-or-more values (Market leading levels), to win the business against the competitors.

But this must be a clear idea, well-delivered in practice, and without sufficient immediate competitive response from your best competitors! This is a process for winners only: having clear and useful scales of measure, is a basic minimum tool, for this competitive action. 'Nice words and intentions only', are for losers.

3.2 Minimal Scales.

Here are *the minimum* attributes of a Scale parameter specification:

- Must allow scale numbers to have a 'useful meaning', when associated with the scale,
- Should not be so short a Scale-specification as to leave critical concepts undefined, or ambiguous to any reader.

Here are some desired attributes of any Scale specification:

- It should be intelligible to domain specialists,
- Should be a good reflection of the value, as perceived by the domain specialists, and other relevant stakeholders.

So here are some reasonable, and simple examples: (prefaced by a tag ('Usability', etc.), to give some context)

Usability:

Scale: % of users who can master the basics within first day of use.

Impressiveness:

Scale: % of people who took a test ride, who then joined a waiting list within a week.

Example 1D

I am not saying these are as good as we can make them, but they are ok for many purposes. They are not good enough for complex, large, critical systems. But they beat most non-quantified value requirement statements, like 'very impressive', or 'highly user-friendly'.

And here are some not so good examples:

Security:

Scale: Number of hacks.

Example 1E

Why? There is a failure to say more about 'who' did the hacking, what 'type' of hacks, the 'time period', the 'object hacked'. Just *too many* unspecified things.

Co-operativeness:

Scale: % of acceptances to join.

Example 1F

Why? Too many related conditions not defined here, like 'join what?', What kind of Invitation? Over which time period?

3.4 Developing a Scale from an Ambition Level.

An **Ambition Level** is an informal statement of a requirement, about one sentence long. It often comes from an 'official', attributed source. The problem is that it is filled with ambiguous terms, and does not lend itself to quantification. So it becomes our job to clarify and quantify 'His Masters Voice'.

We could of course complain that the source (our boss?) is sloppy. But that would be unnecessarily undiplomatic.

Instead we should joyfully accept the challenge of articulating what the *power that be*, said. After all, as I say:

He who taps the keyboard holds the real power.

Note that this Ambition translation process is essentially the same as the design of a Scale from a User Story, as explained above in 'Can User Stories be used?'

Here is an example of an ambition level:

Ambition: "before performance, Tesla prefers to focus on safety first" <- Elon Musk. 140319

Example 1G

And here is a Scale we can derive from that:

Scale: % average passenger safety rating by Euro NCAP

Example 1H. <https://www.euroncap.com/en/results/tesla/model-3/37573>

To derive that Scale we had to think:

- What kind of safety ratings give useful objective proof of safety?
- What units of measure are used in them (stars, % survival) ?
- Which units of measure, if there are several, serves our purposes in this project?
- Some searching on the the internet (Tesla, Safety Ratings) might give specific options of ideas.?
- Would the power-that-be (Musk) think this is a good scale of measure for *his* purposes?
- Is our suggestion broad enough for purpose, or is it unnecessarily narrow?
- Does it cover all market areas?
- Does it cover all types of the value (safety)?

The 'level' is missing!

There is a specification element missing in the **Scale** spec, which is 'exactly where on the Scale we are targeting for the future', our Ambition **Level**.

We have (Scale) derived a definition of the Safety value, suitable for applying (safety level) numbers. But we have not yet derived the *required levels* themselves. So we have only done the first half of the job of interpreting the Ambition Level. Let's say we did the 'Ambition definition' part (the Scale); and now we have to turn to specify interesting levels on that Scale.

We could add such a Level specification, and possibly derive it from the Ambition quotation. This is a subject we will look at below. But it might look something like this:

Goal 98% [Within 2 years, for Adult Occupants].

Example 11

There are several weaknesses with the Scale I suggested above. And I will discuss improvements below. But I wanted to make the point about 'deriving a scale from an Ambition Level' .

This derivation practice can be done in a much more detailed way, when the Ambition level is richer with various concepts (about *when, where, who, what*). We can return to that after the next section on Scale Parameters.

DOWNLOAD REPORT (PDF)

Adult Occupant



Child Occupant



Vulnerable Road Users



Safety Assist



Figure 1.11 : Tesla-3 %-ratings from euroncap.com (URL op cit). We note they use a % scale.

3.5 Scale Parameters

If we want:

- Accurate modeling of large and complex systems
- The possibility of separating critical value-deliveries from less-critical ones, which permits us to 'do critical stuff early'.
- The possibility of much better definitions, so nobody can possibly misunderstand (like contractors and suppliers, and even managers).

Then you will want to improve the 'resolution' of the Scale tool.

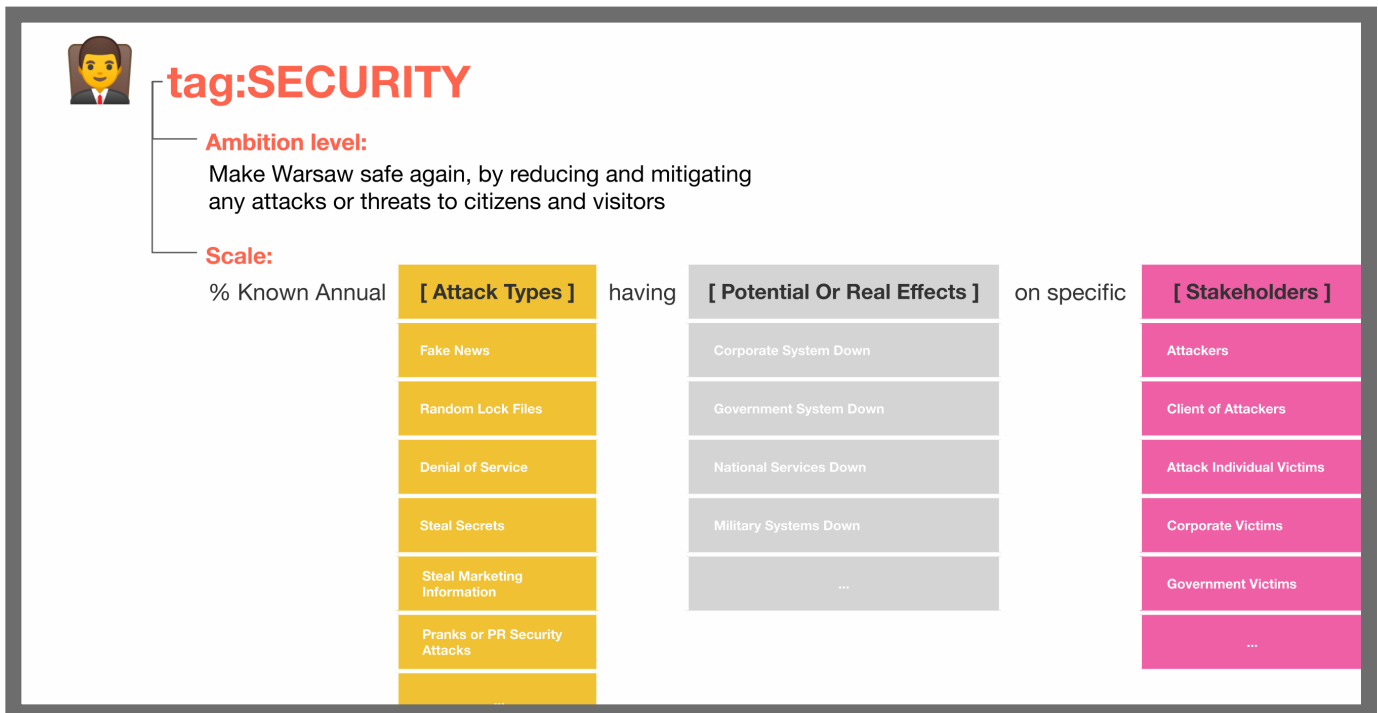


Figure 1.12 : Each Value Scale is one of many dimensions of the system's Value Set. Each Value spec can have several [Scale Parameters]. Each [Scale Parameter] can have several attributes which are used as requirement specification'conditions. I might call this Three-Dimensional Value Modeling. Example from May 2019 Master Class, Warsaw planning exercise. Graphic Design Source: anna@Karlowska.PL, 2019

This need for improved 'resolution', using Scale Parameters as a tool, is so common that I find I have to use it on almost every value Scale, on every project, even seemingly small projects.

Here is an example: derived from <https://www.euroncap.com/en/results/tesla/model-3/37573>

Vehicle Safety:

Scale: Star Rating number for [Person Type] and [Car Specs] for [Safety Equipment] with [Alternative Model Validity] for a [Publication Date] by a [Rating Agency].

Example 1J

Perhaps you can imagine roughly what is happening with this specification. It is intentionally quite readable for a domain specialist (car freak).

The terms in square brackets ([Car Specs]) are:

- Formally defined terms: The **C**apital **L**etters signal that they are defined, somewhere
- General Concepts: defined with a specific set of elements, which as a set, define the general concept.
- For example: People = {Babies, Children, Adults, Aged}. We are going to sometimes use the set '{...}' parenthesis, to list a set of things. But sometimes this is not necessary for clarity, so we drop it, for clarity.

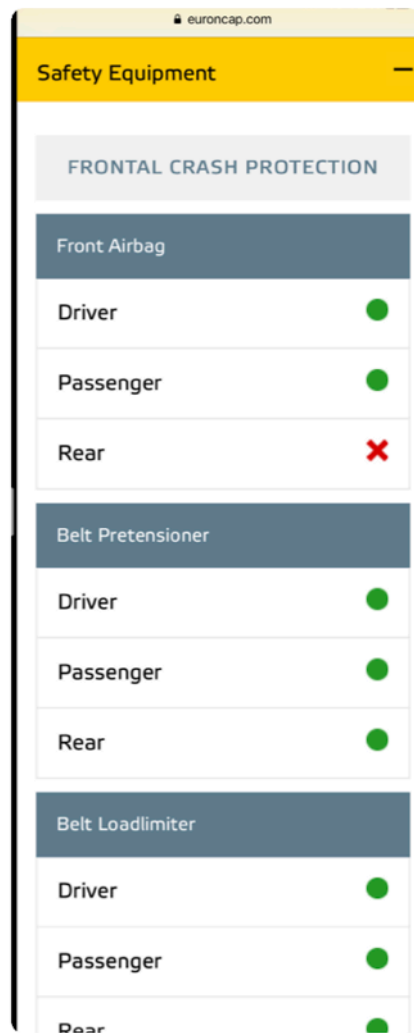


Figure 1.13 : a clip from euroncap.com safety report, URL cited above, which shows the actual structure of the 'Safety Equipment' concept in reality.

Here is an example of making use of the [Scale parameter] structure to articulate a target level requirement.

Wish: 5 Stars, by Next Year, Person Type = All, Car Specs= {Tesla 3, RWD, 4 Door, 2019}, Safety Equipment= {Front Airbag, Belt Pretensioner, Belt Load Limiter, Knee Airbag, Side Head Airbag, ...}, Alternative Model Validity= Dual Motor AWD Model 3, Publication Date=2019, Rating Agency= All.

Example 1K. A 'Wish' level is a stakeholder value target, which is not yet accepted by a project as prioritized, feasible and economic (that is called a 'Goal' level commitment).

Read it slowly, and parse it, decompose it. Or see an edit below.

Here is a more-structured format

Wish:

5 Stars,

by Next Year,

Person Type = All,

Car Specs= {Tesla 3, RWD, 4 Door, 2019},

Safety Equipment= {Front Airbag, Belt Pretensioner, Belt Load Limiter, Knee Airbag, Side Head Airbags, ...},

Alternative Model Validity=Dual Motor AWD Model 3,

Publication Date=2019,

Rating Agency= All.

Example 1L. Same as Ex. 1K, just spread out for readability.

We can specify any useful number of such statements, with any useful valid combination of Scale parameter dimensions we want.

We can home-in on the most critical subsets of Scale parameter dimensions, so that we can focus our energy on, and prioritize, exactly the ones that have the highest value for us, especially in the near term.

This might seem 'complicated' at first sight.

But it is in fact a way of *simplifying* very complex overall problems, by allowing us to carefully extract something simple that we can work on, and deliver some value improvements early, for critical subsets.

Early partial value delivery is about 'learning about complex realities', before we commit to scaling up.

3.6 Defining Other Terms in a Scale

A Scale specification, will use the **Scale Parameters**, to give pretty sufficient *definition* of the Scale Parameter terms.

For example (a definition of a Scale Parameter, in terms of a set of things):

Person Type = Adult Occupant, Child Occupant, Vulnerable Road Users, Safety Assist for Driver

Example 1M: The set of things that make up 'Person Type', serves as a definition.

But there may be *other terms* in a Scale specification which require formal definition in our specification, to avoid ambiguity, misinterpretation (intentional, or not), and consequent problems, delays and costs. But are not defined in terms of a set of things.

It is a necessary defensive practice, a risk-mitigation practice, to formally define these terms somewhere. In the specification, or in project-related glossaries.

The Planguage-agreed signal for formally-defined terms is, as is also the case with Scale Parameters, that we use **Capital Letters** in the words of the term, as a signal that a formal definition is available, or should be at some point. When tool support is used, such words will appear as hot link words, one click away from the formal definition.

Example:

Child Occupant: a person under 16 years of age.

Example 1N: Defining a 'Defined Term' with a straight definition; not using a set of things to define it.

My personal practice, when someone asks 'what does that word mean?' is to immediately and always, create a formal definition.

At least to Capitalize the term to indicate my intent to define later.

Merely answering orally is a poor practice, for obvious reasons.

3.7 Scale Libraries

The best scales of measure will be *highly tailored* to the local project environment.

It can however help you, to begin the tailoring process, from *previous* examples.

These Scale definition examples are stored in several ways.

1. Previous projects in the same environment are potentially rich with useful examples of Scales, and the experience of using them in practice (think of the 21,000 Intel engineers and the environment of *chip architecture*).
2. Some very common Scales of measure (examples Usability, Security, Maintainability) are published in books like my Competitive Engineering book: see examples Chapter 5 at <http://concepts.gilb.com/Free+Download+Competitive+Engineering+-+Chapter+5>.
3. Some of these are digitalized in tools, like ValPlan.net

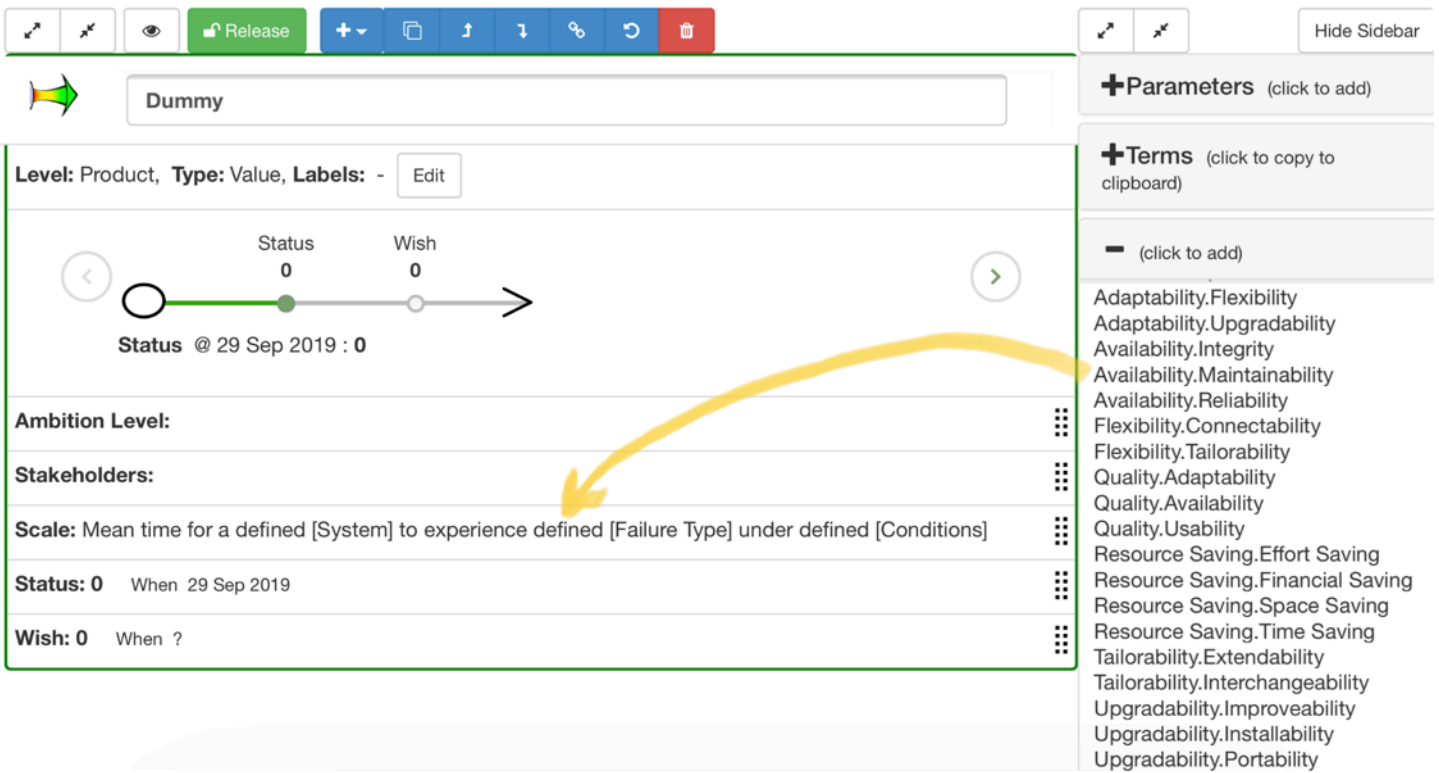


Figure 1.14 : a list of ready-made Scales of measure (ValPlan.net tool). The 'Maintainability' scale of measure was selected from the library of measures², and copied into the 'Scale' specification.

It can be modified if desired at this point. But at the least, with it's 3 [Parameters] it is quite general and can be applied for many detailed dimensions, which are up to us to define in detail. You can continue to add to this Scale library with your own Scales, for reuse later.

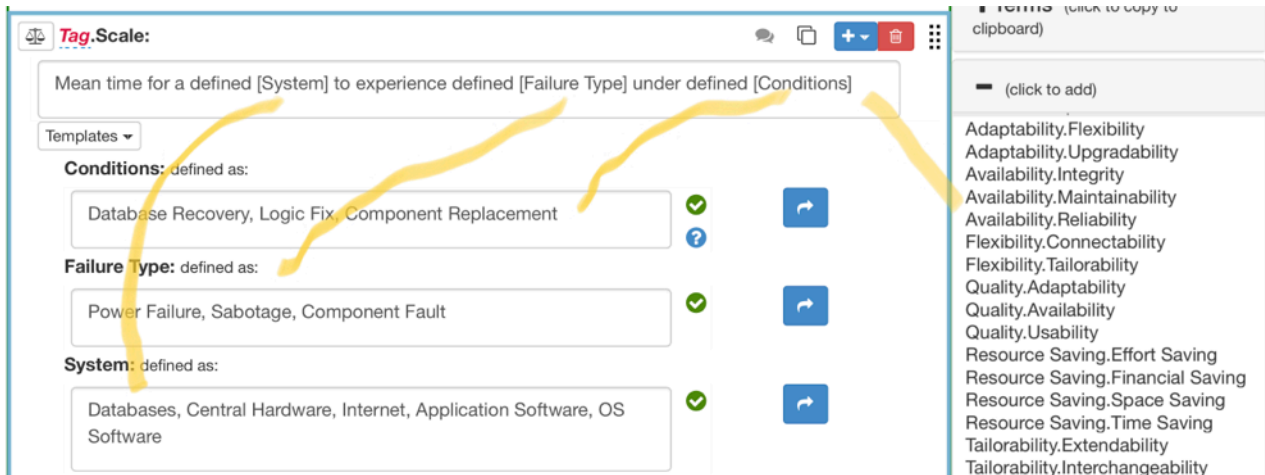


Figure 1.15: The [Scale Parameters] from the Scale Library Template can be defined as you wish and need. For example as above.

² This set of Scales was directly derived from Competitive Engineering Chapter 5. <http://www.gilb.com/DL26>

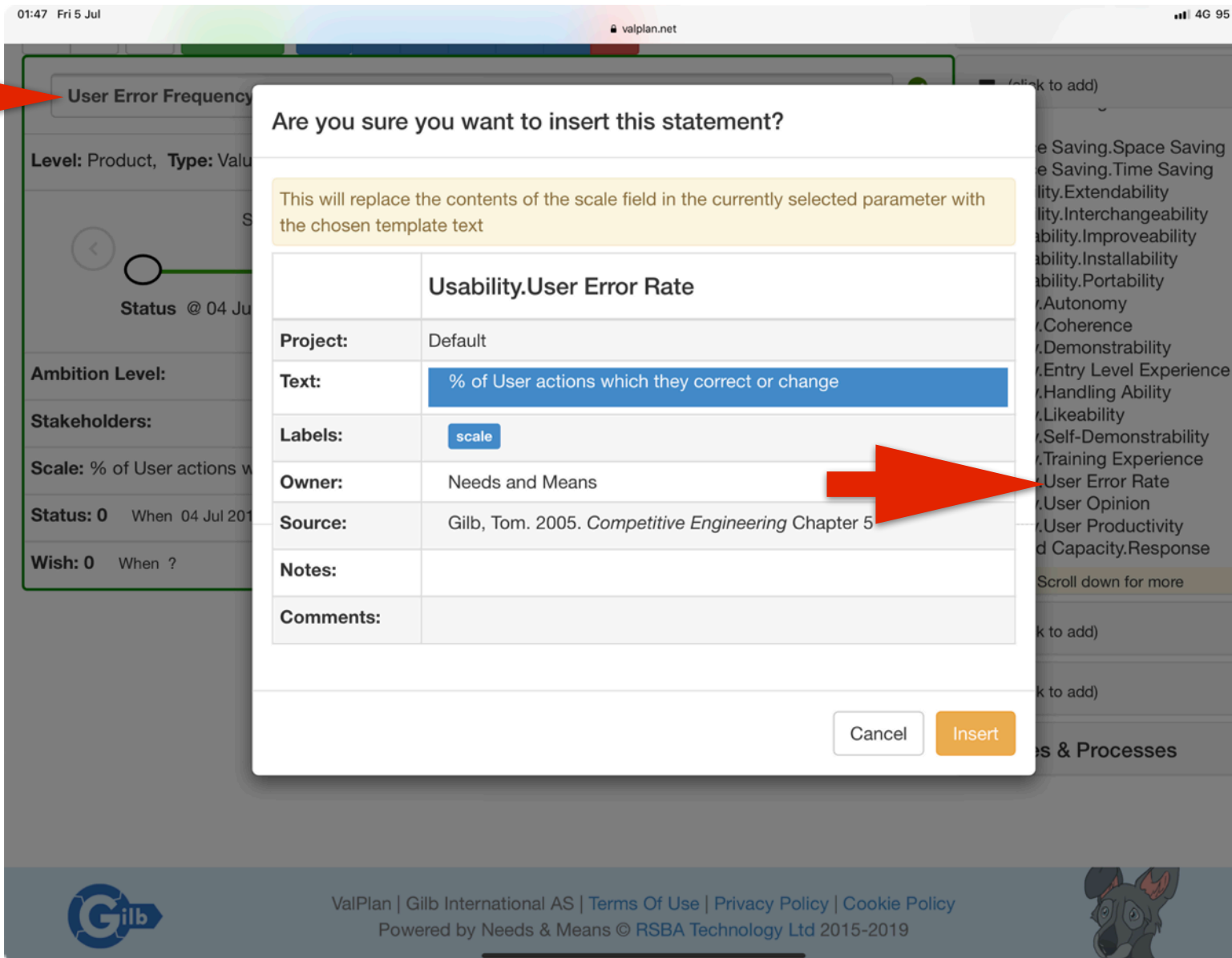


Figure 1.16: In this ValPlan tool example, a Value specification, tagged 'User Error Frequency' needed a Scale of measure.

We looked in the ValPlan library (copied from the *Competitive Engineering* book) and found a similar value called 'Usability.User Error Rate'.

The Scale looked good enough, so we Inserted it into the User Error Frequency specification window. (Left background, in the Scale).

We are now free to modify it to taste. For example by making 'User' a Scale Parameter [User] and then defining classes of User, like {Novice, Advanced, Coach}.

The Internet Library of Scales of Measure.

The internet has a huge collection of defined Scales of measure, for almost anything you can imagine.

Search your favorite Value + the word 'metrics'

Example 'ice cream taste metrics'

Here is what I found: the first hit,

Matters to Grocers

The one metric that grocers and the dairy industry use to determine the quality of ice cream is overrun, which, in the simplest terms, is how much air is in your ice cream. The lower the overrun, the lower the air content, and the better the quality of ice cream.

To get a bit more technical, overrun is a measurement of the volume of air relative to the initial volume of mix or base (typically milk, cream, and sugar). One hundred percent overrun, for example, means that every pint of mix or base will provide two pints of ice cream. In other words, half the content in your pint is air.

ADVERTISEMENT



In order to be called ice cream (vs. "frozen dessert"), the FDA requires an overrun of less than 100 percent, but the good stuff — in the dairy industry "Premium" or "Super Premium" — has an overrun of

Figure 1.17 : Quality of Ice Cream Metric

So, before you say

- There is no quantification
- It cannot be quantified
- It is a 'soft' value
- I do not know how to write a Scale for this

Search the web for a pretty-good starting-point Scale.

There are always lots of options out there, and you can tailor them for your use.

No excuses. **All** critical values can be quantified, with a defined Scale, easily.

Try *your* interesting value (+ 'metrics') on your phone browser now.

Lots of professionals and academics have struggled with quantification of the same values as you are interested in, and put their experience on the internet. Use it for inspiration.

3.8 Multiple simultaneous Value Requirements

Real problems are not nice to you, they are not simple to understand.

You cannot just focus on a single Value metric and forget about any others.

You are going to have to normally 'juggle' ten or more value metrics at the same time.

If they all have well-defined Scales of measure: you stand a chance of success.

If they are all in the fuzzy 'Ambition Level' condition: failure is guaranteed, in a stormy sea of confusion.

I recommend that you select a set of the **top-ten most-critical** value-requirements, to focus-on delivering, initially.

Keep all others, lower-priority values, on hold, until you have delivered the first group. (Reference B)

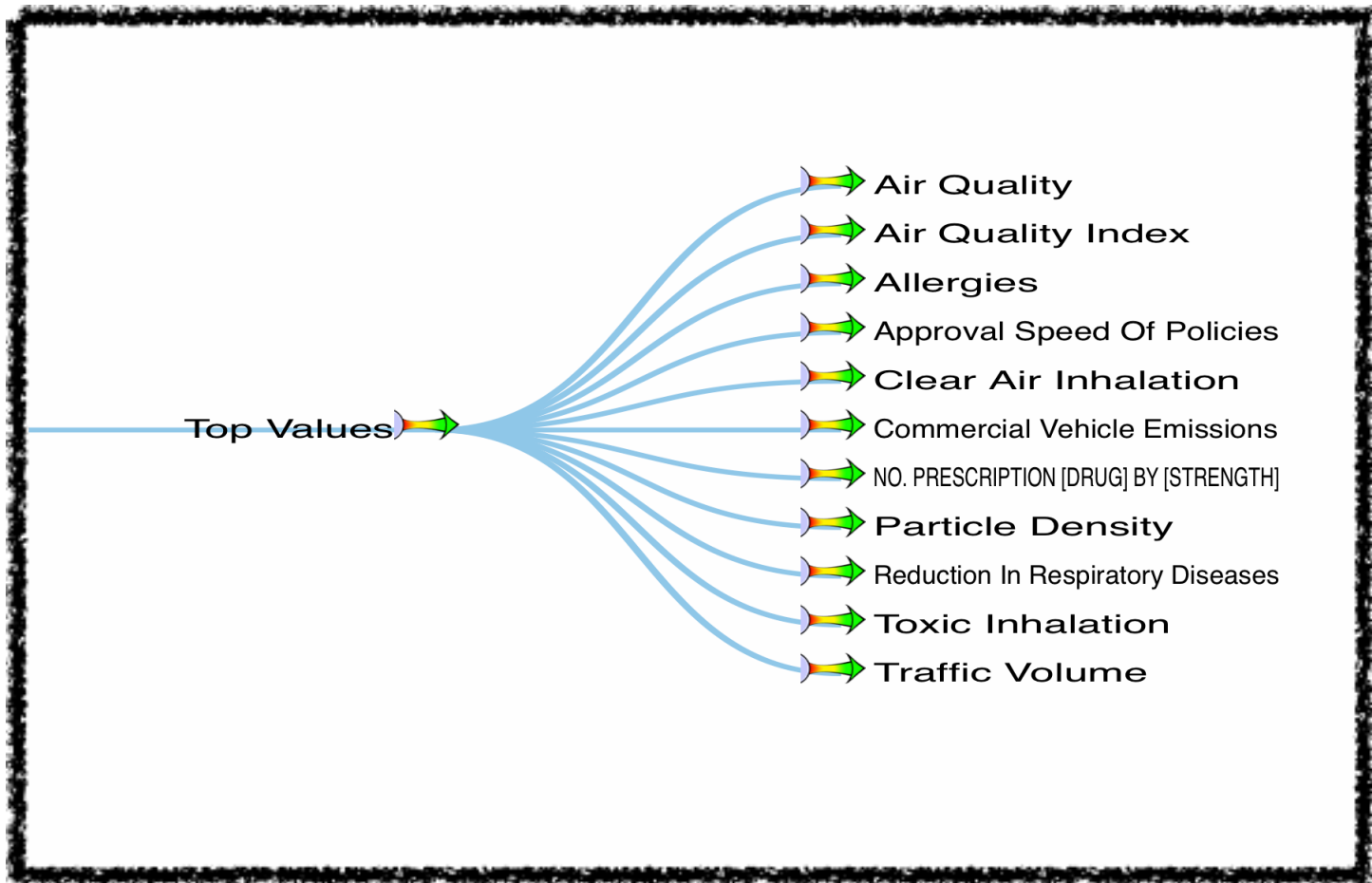


Figure 1.18 : Example of Top 11 Values.

Source BCS Exercise Sept 2017, 'London Congestion for Air Quality'. Notice that this is also a definition of 'Project Value' using the 11 decomposed different values, as the definition-by-subset.

3.9 Here is an example of a single complete Value Requirement Specification, with all the extra supporting detail we will discuss below.

Advance peek a real and complex example: to be explained in this book detail by detail.

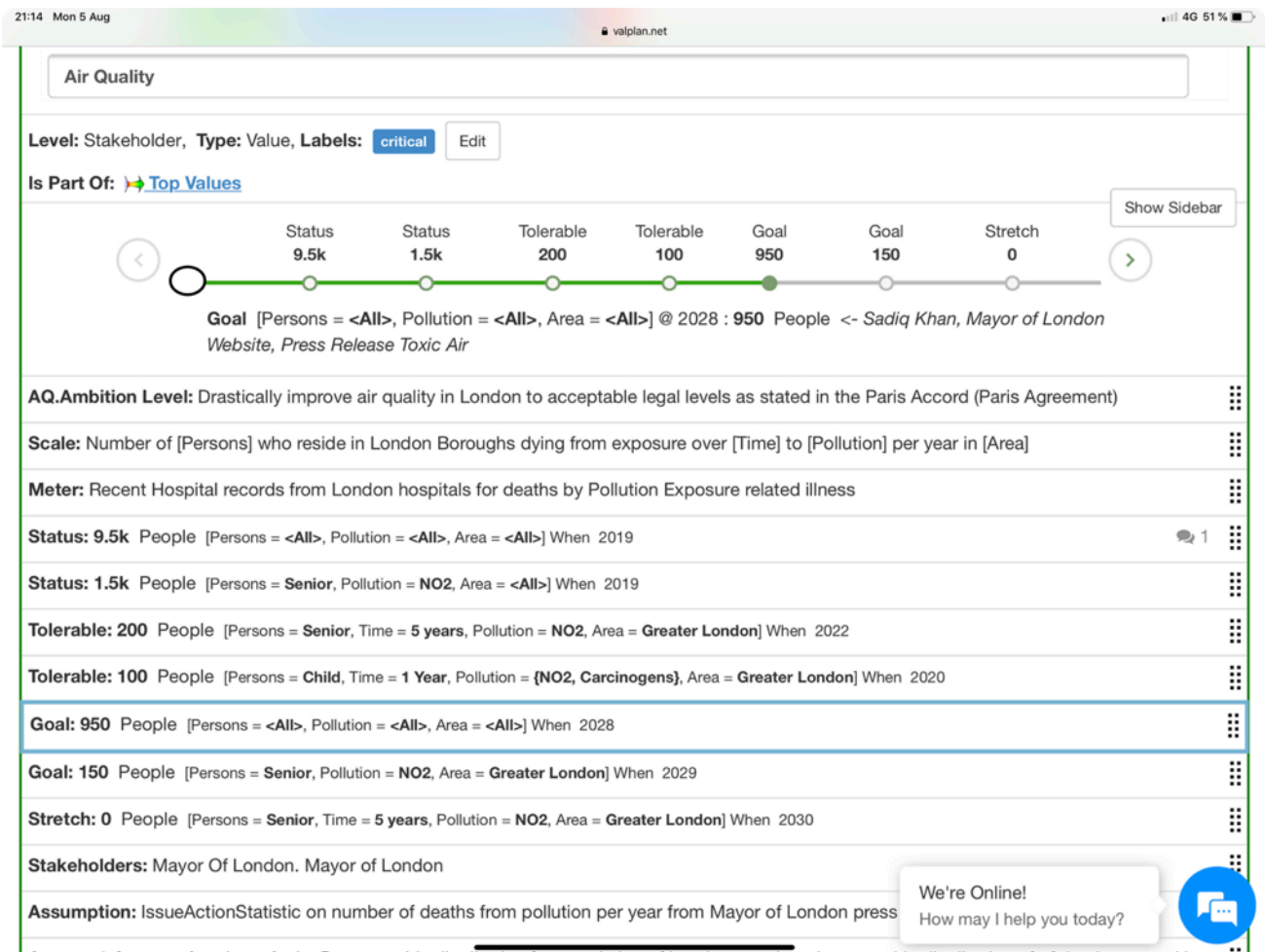


Figure. 1.19 : One of the 11 Values above (Fig, 1.18), Air Quality.

A Summary specification of 1-liners for each parameter. Using Planguage, and the ValPlan.net tool.

There are a few parameters in this example, which are not yet explained in the book. But they will be explained.

But you can guess, look them up in the book glossary, and read them with some understanding.

3.10 Details of a Scale (Air Quality example), with Scale Parameter Definition.

Air Quality

Level: Stakeholder, Type: Value, Labels: critical Edit

Is Part Of: ➔ [Top Values](#)

Status [Persons = <All>, Pollution = <All>, Area = <All>] @ 2016 : 9.5k People <- Sadiq Khan, Mayor of London Website, Press Release Toxic air

AQ.Ambition Level: Drastically improve air quality in London to acceptable legal levels as stated in the Paris Accord (Paris Agreement)

Tag.Scale:
🗨️ 📄

Number of [Persons] who reside in London Boroughs dying from exposure over [Time] to [Pollution] per year in [Area]

Templates ▾
Press **⌘+e** to show edit

Area: defined as:
Greater London. Within M25

Persons: defined as:
 ➔

Pollution: defined as:
 ➔

Time: defined as:

We're Online!

Figure

1.20 : Here is a view of the 'Air Quality' value spec with detail of the Scale. You can see that the Scale parameters are defined as a set of attributes.

The 'Area' Scale parameter was previously defined, and reused here. The colored 'Area' is a hot link to the glossary definition of 'Area'.

3.11 More on Multiple Value Requirements

You might be worried about the advice to put aside, for the moment, some of the critical values, in excess of about 10 of them.

They will not be not forgotten! They are usually specified at some level of detail in the overall planning. They are just intentionally delayed in time, so that we have a better chance to actually deliver some higher priority values first.

If we try to do everything at once, then nothing will be delivered early, and we increase the risk of total failure, by running out of resources, political or organizational change, or by failing to learn hidden lessons from the earlier deliveries.



Figure 1.21: I took this photo June 2019 of a Western Norway River. Value delivery needs to be an early and continuous stream of measurable value improvement. Quantifying our values is a necessary first step.

We are not even going to deliver the 'top ten' values, all at once. We are going to collect enough background information (like stakeholders, risks) on them, to further prioritize some of them.

We are going in the direction of decomposition of both 'values', and decomposition of their technical 'solutions', needed to deliver those values, so that we end up with very early, and frequent, small (2% of project resources at a time) value delivery steps. A 'value stream' to stakeholders.

We are in a hurry to deliver critical real stakeholder value very early, as a continuous stream of stakeholder value results.

We also want learn about the complex environment we are working in, so that we can apply those lessons forward; and to build up our credibility, with the powers that be, for real value-delivery,.

3.12 Knowing when to decompose a value into sub-values, using sub-scales.

Many of your attempts to find quantified value Scales, might be delayed by the fact that your value concept is in reality a set of quite different values scales, which have something in common' Love is a many-splendored thing, as the song goes (ref. b)

There is an engineering heuristic that says 'decompose the value you want to quantify until 'quantification is obvious'. This works well.

Sometimes there just seems to be no quantification available, because you are at 'too high' a level of abstraction.

Earlier we showed that the concept of 'value' needed to be decomposed, into many sub-values. Each with their own quite different scale-of-measure.

This is often true the next level down: some of those 10 values are going to need decomposition, before we can make sense of them quantitatively.

It is very common to need decomposition.

We seem to think in terms of complex value ideas: like 'love' or 'beauty'.

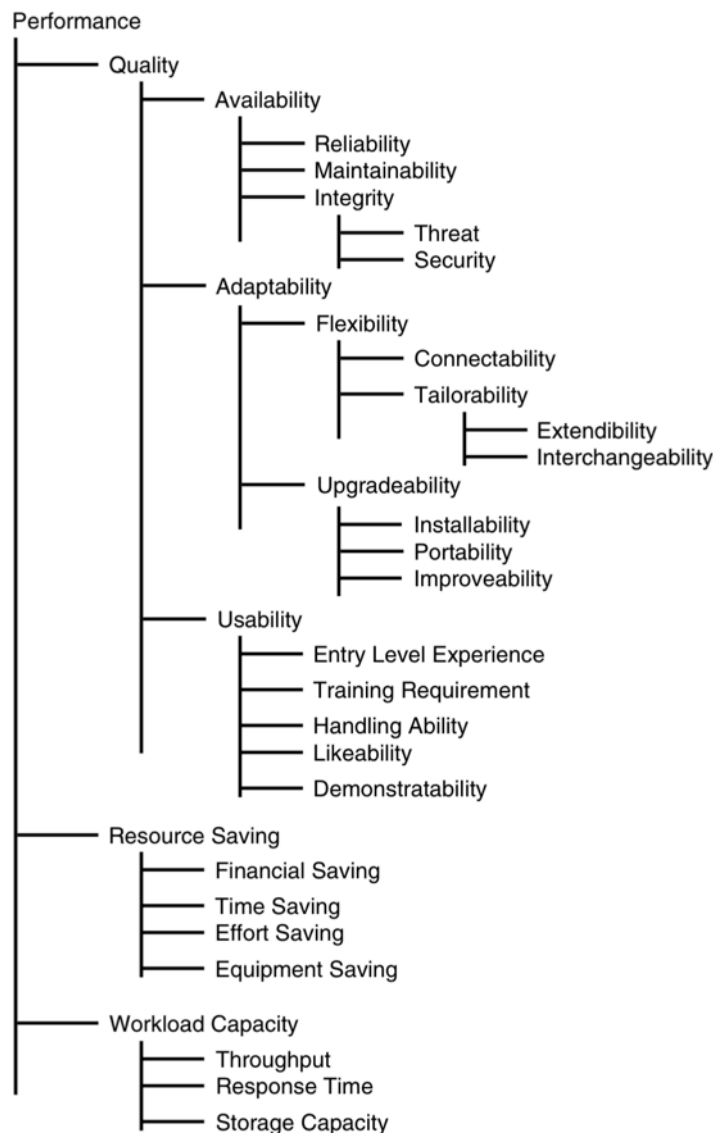


Figure 1.22 : Examples of decomposition of higher level values into different-Scale sub-values.

Detailed examples of potential Scales of measure for these sub-values are given in the CE book. (1)

Some 'values' are simply 'umbrella titles' for a set of different values.

Sometimes, you have an option to decompose into sub-values, or to use Scale parameters to combine the value ideas into one generic scale. Sometimes that is just a matter of taste or convenience. The result might be the same

3.13 Good Scales and bad Scales.

Just because you found a way to quantify a value, with a scale, does not mean you have a good-enough, and *useful* scale.

It means you have 'quantified clarity', and that the *clarity* may even help you understand that it is clearly a 'bad' scale!

Good and 'useful' Scales of measure:

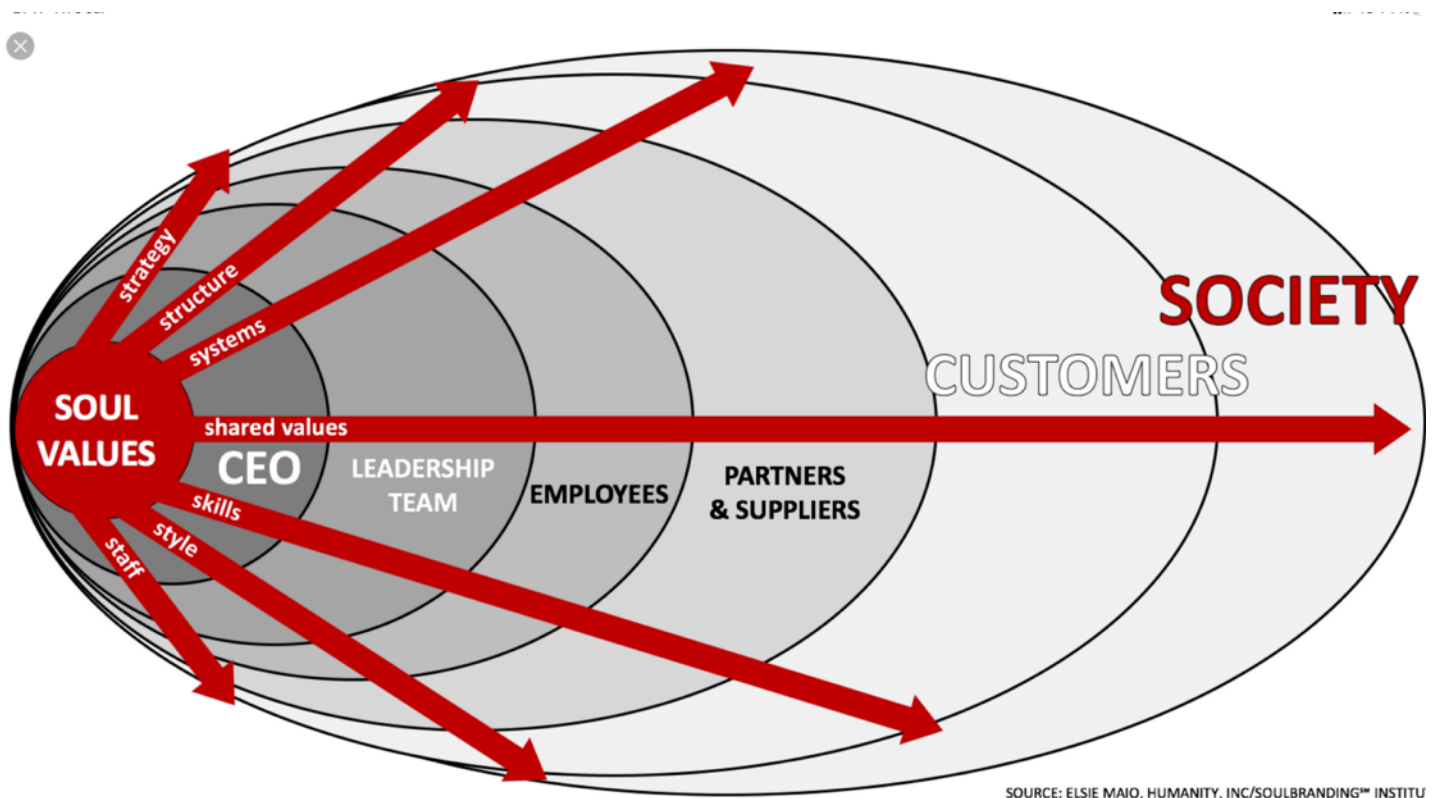
- Are strongly related to the 'values of the stakeholders' who care most about that value.
- Might be more difficult to measure in practice, but you should never choose a Scale *because* it is 'easier to measure'. You must first choose a *relevant* Scale, *then* try to reduce costs of measurement. Cheap measures of the wrong scale are a waste of time.
- Will be highly tied to the real environment, with plenty of necessary [Scale parameters] to specify realistic and critical dimensions of the value's application or environment.

Once a client of mine chose 'bugs' counts as their primary quality measure, because they were easy to measure: rather than system (software) availability for the phone system they were building: which they knew was far more critical ('but we do not know how to measure it' they said). This was part of the reason they were 2 years late to market. Micromanaging the wrong value. They got sorted out in time to reduce delay to only 6 months.

By the way, if you have clearly defined a relevant Scale, it will normally not be too difficult to design a reasonable measurement process to suit it. A 'Meter'

The right Scales will feel good and relevant to real stakeholders and domain specialists. Work with stakeholders until they are happy with the Scales.

This is all related to the management interest in 'alignment' of your plans with their values and objectives, at a higher level. Your values must align with the next level above you. Clear real alignment is the test of relevant value specifications. More later.



SOURCE: ELSIE MAIO, HUMANITY, INC/SOULBRANDING™ INSTITU

Figure 1.23 : alignment levels and related concerns.

Chapter 4. The 'Meter' Parameter



Figure 1.24 : Meters are sort of like this Weather Forecasting Stone. They tell it like it is.

With permission David Bishop (with beard), Photo, Tor Gilb, Hvitsten, Norway, 2019

4.1 The 'Meter' specification is a defined process for measuring the numeric value level, on a Scale.

A Meter Specification is not normally a 'requirement³' it is a way of measuring the delivery level of the requirement in a project.

The Meter is very directly connected to the defined 'Scale' of measure. The Meter must measure exactly what the Scale defines. That includes all its [Scale Parameters].

The 'Meter' question is not merely 'was the value level required finally delivered'?

The really useful Meter will give us incremental progress reports on the emerging value levels. It will be designed to give sufficient accuracy at a low-cost, consistent with frequent use.

There is not merely one single test process for a value. There may well be several for different purposes, with different qualities and costs.

³ a customer, in a contract, can require defined test or measurement processes. In that case they are required.

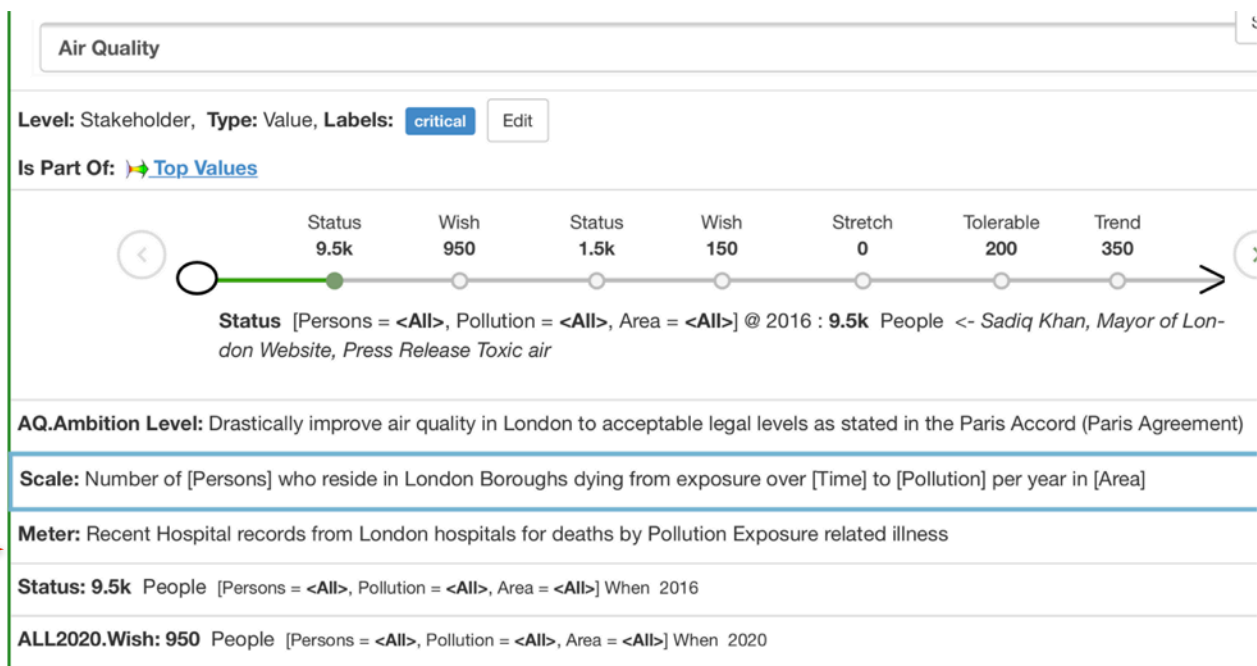


Figure 1.25 : the Meter is a direct reflection of the Scale. It measures along that exact scale.

The Meter has to deliver measurement of all aspects, including all defined [Scale Parameters] of a complex Scale, at a reasonable cost.

A Meter has to have necessary qualities such as acceptable levels of automation, accuracy, credibility, repeatability, setup costs, and legality.

At this 'requirements' level of specification, we might simply *outline* some major ideas of how to measure, and leave the final decision and detail to a professional test planner.

The most critical aspect of a requirement is the Scale and the future required levels.

It is not strictly necessary to define Meters immediately, unless they are contractually required. The measurement process can be worked out later when we need to measure the value created. But it is useful to sketch a reasonable possible process.

4.2 The Meter as a high-level test process: why this is useful.

I do not practice detailed test planning in the Meter specification. The details should be worked out by professional test planners. In fact they should be able to improve upon, and override a Meter specification.

The purpose of a Meter specification at this early, requirements stage is:

- To suggest that reasonable measurement methods exist at all for this value
- To suggest the possible accuracy, credibility and costs that the measurement process would give us
- To make it clear that we are seriously intending to measure the values delivered
- To give detailed test planners something to start with
- To make it clear if there are any mandatory constraints in the test as part of the system requirements
- Privacy concerns
- Contractual requirements regarding measurement for payment

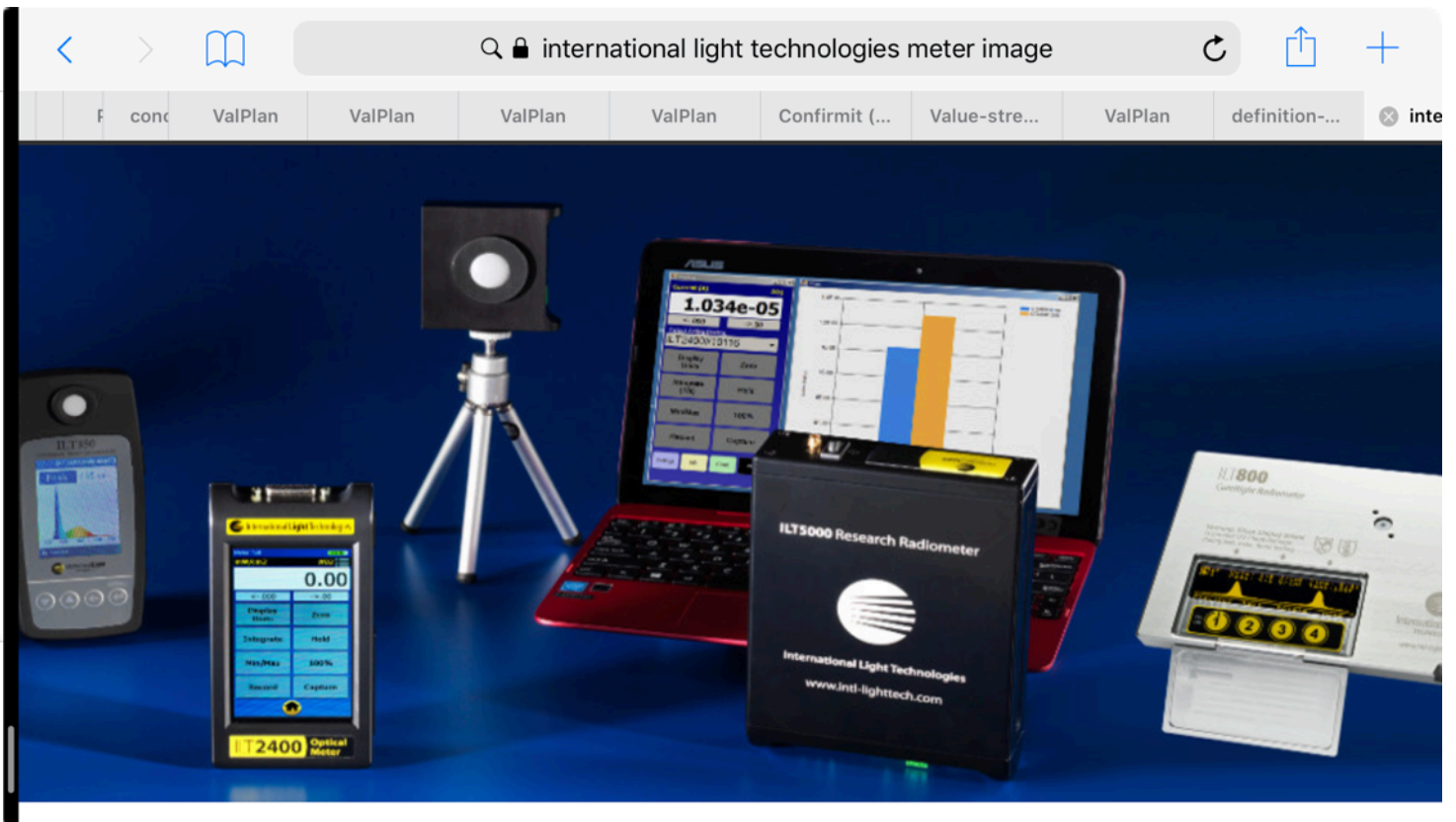


FIGURE 1.26: METERS.

4.3 The multiple quality and cost attributes of a Meter

A Meter, like any test process, has a number of interesting quality and cost dimensions.

The qualities must be sufficient for purpose, and the costs should be as low as possible, for a defined set of required Meter qualities.

In a sufficiently advanced project culture it might be useful to quantitatively state the Meter Value-Requirements (like 'accuracy'), and to *design* a Meter to be within them.

At least, there is always the possibility of designing the tests to use the least possible resources: for example by using sampling, or automation.

Here are some of the quality Aspects of a Meter:

- Accuracy (is it close enough to the truth?)
- Relevance to its Scale
- Repeatability (same results each time)
- Sensitivity (to disturbing factors)
- Credibility (will people believe it and buy in)
- Legality (will it break laws, customs, standards, contracts?)

- Automate-ability
- Privacy (permission to snoop?).

Here are some of the cost aspects of a Meter:

- Detailed Planning costs
- Execution Costs
- Result analysis costs
- Presentation Costs
- Permissions costs
- Travel costs

4.4 Sufficient Meter Accuracy for Purpose

In early incremental stages of value delivery, high quality measurement is not generally required.

It is sufficient to be pretty sure things are moving towards the required levels of the value, at a reasonable pace. At the extreme, 1-digit accuracy of the % value-improvement might suffice.

One client of mine dropped measurement of weekly increments, and left it to the very-experienced intuition of the system developers. At an earlier stage the same client decided to use no more than 30 minutes per weekly increment, to measure value delivery. For Usability factors they even got lucky when Microsoft Usability Labs offered to measure weekly, overnight, for free. (Ref. C,D). They did take release quarterly of product upgrades far more seriously for measurements.

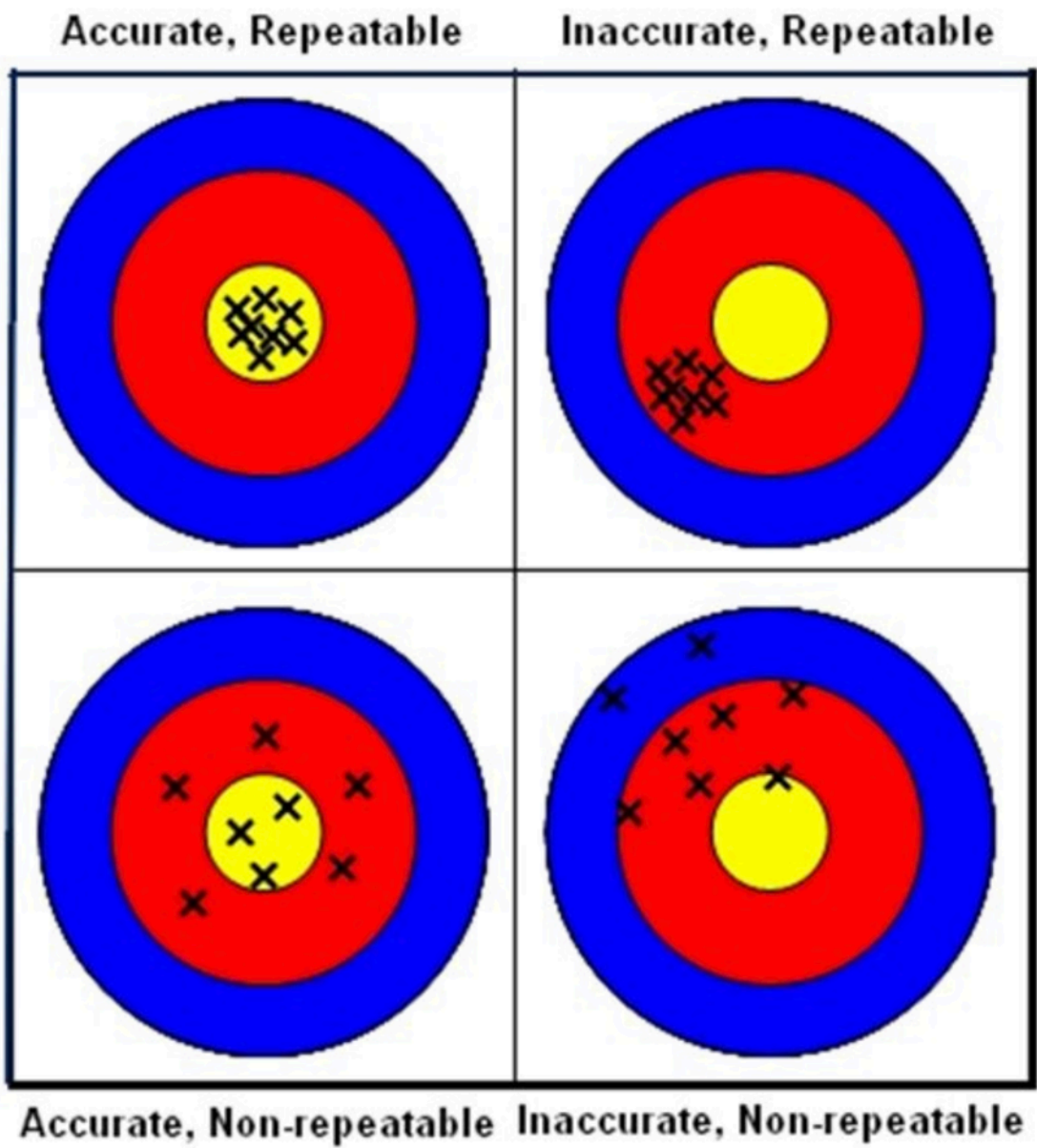


Figure 1.27 : Accuracy and other Meter concepts.

Chapter 5. Benchmarks

5.1 The purpose of 'Benchmarks' In a Value Requirement Specification.

A 'Benchmark' level of value, on a Scale of measure, is *background* information about a requirement level. It helps us decide if we have set the real requirement levels appropriately.

This is traditionally something a Business Analyst should look at, as a prelude to setting requirements.

But in Planguage, I decided that it was better to *integrate* Benchmark data with the requirements data.

- in order to make it possible for all reviewers and creators of a requirement object, to decide for themselves if the requirement levels are in reasonable proportion to the benchmarks
- To make it even clearer if the Benchmarks data is missing, or not particularly credible, or up to date.
- To support incremental delivery, where Benchmarks need to be updated, at each increment, not just in an initial Waterfall analysis phase.

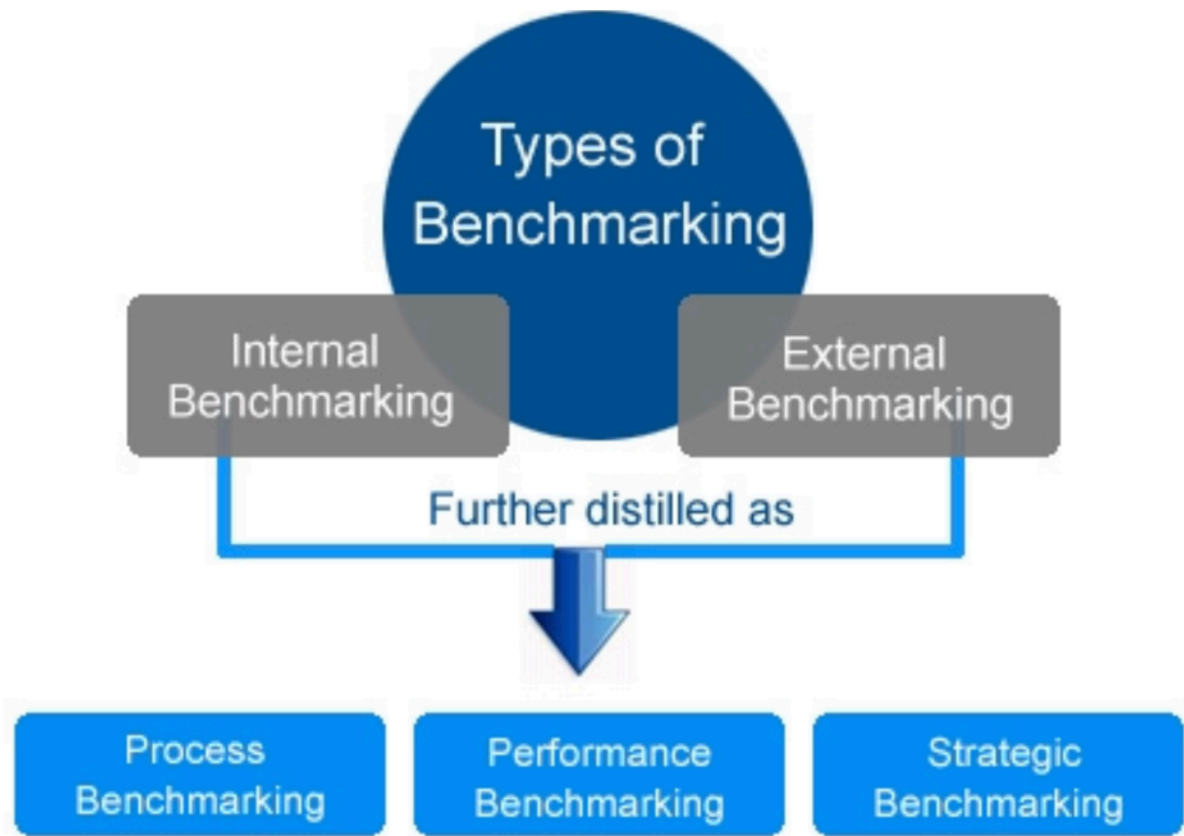
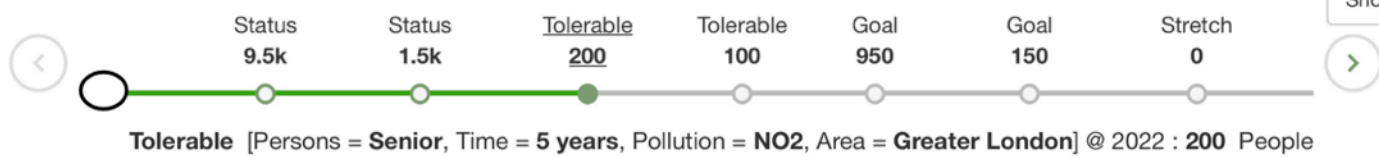


Figure 1.28 Types of Benchmarking.

Level: Stakeholder, Type: Value, Labels: **critical** EditIs Part Of: [Top Values](#)**AQ.Ambition Level:** Drastically improve air quality in London to acceptable legal levels as stated in the Paris Accord (Paris Agreement)**Scale:** Number of [Persons] who reside in London Boroughs dying from exposure over [Time] to [Pollution] per year in [Area]**Meter:** Recent Hospital records from London hospitals for deaths by Pollution Exposure related illness**Status: 9.5k** People [Persons = <All>, Pollution = <All>, Area = <All>] When 2019**Status: 1.5k** People [Persons = Senior, Pollution = NO2, Area = <All>] When 2019**Tolerable: 200** People [Persons = Senior, Time = 5 years, Pollution = NO2, Area = Greater London] When 2022**Tolerable: 100** People [Persons = Child, Time = 1 Year, Pollution = {NO2, Carcinogens}, Area = Greater London] When 2020**Goal: 950** People [Persons = <All>, Pollution = <All>, Area = <All>] When 2028**Figure**

1.29 : 2 different 'Status', which is a type of benchmarking. Benchmarks are 'background' information embedded in a requirement.

5.2 'Past' as a Benchmark

A Past level statement is a fixed result at a fixed date. History of a level which happened.

You can insert as many Past statements as are potentially useful, at any time in the process. As new data occur for example.

Using Past level information we can better decide if our requirement levels are appropriate.

- are we planning to be good enough in relation to our own Past levels ?

- And those of competitors ?
- Is updated Past level information sufficient to force us to reconsider planned requirement level specifications ?

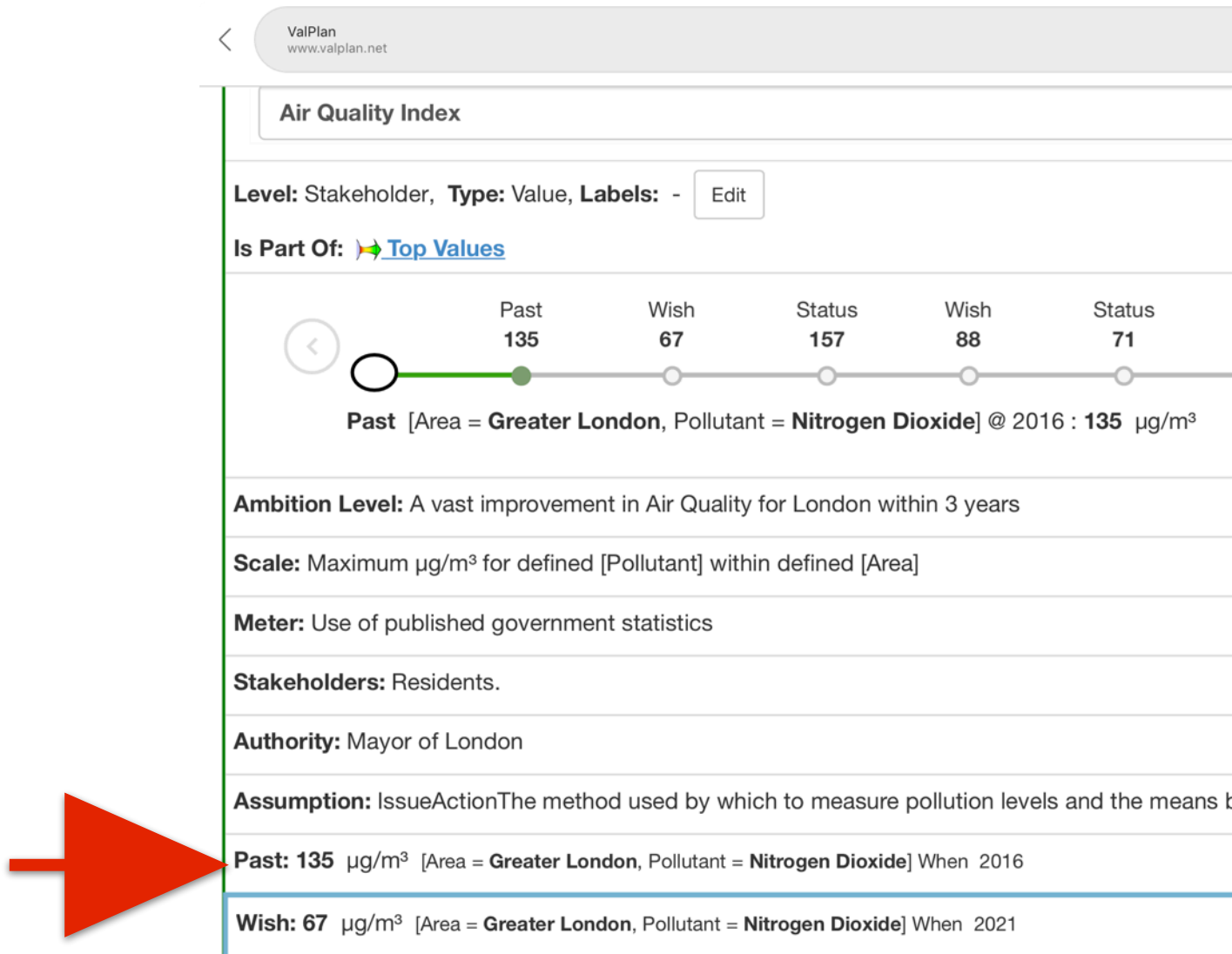


Figure 1.30 : The Past data here for Greater London and Nitrogen Dioxide, are directly comparable in the 2 Scale parameters, and the units of measure on the Scale. The requirement is for a 2x reduction over a 5 year period. As it is now 2019, we need to ask if the Past data is up to date (at least 2018) and if any progress has been made as a result of our project deliveries, if any. It is time to introduce a Status specification to track progress.

Note: I am using examples using the [ValPlan.net](http://www.valplan.net) tool. But this tool is NOT a prerequisite for using this method or Planguage. A Word Processor works fine (1, CE). Just more work.

'Past'-level data is not necessarily from our own systems. It can be from any system that might be useful to compare us with. Competitors, and other industries using similar methods or architectures.

5.3 'Status'-level Benchmark: real time value delivery tracking.

The 'Status' benchmark level is intended for use in incremental value delivery, to track our own project progress, or lack of it, towards required levels.

It can be used initially as a departure point, for tracking progress on your very own system: an incremental baseline in the continuous learning and re-planning process.

We can keep track of a series of Status, in the basic value requirement. But we can also track status as a graph line, based on feedback in increments after a value delivery for our system. Or both.

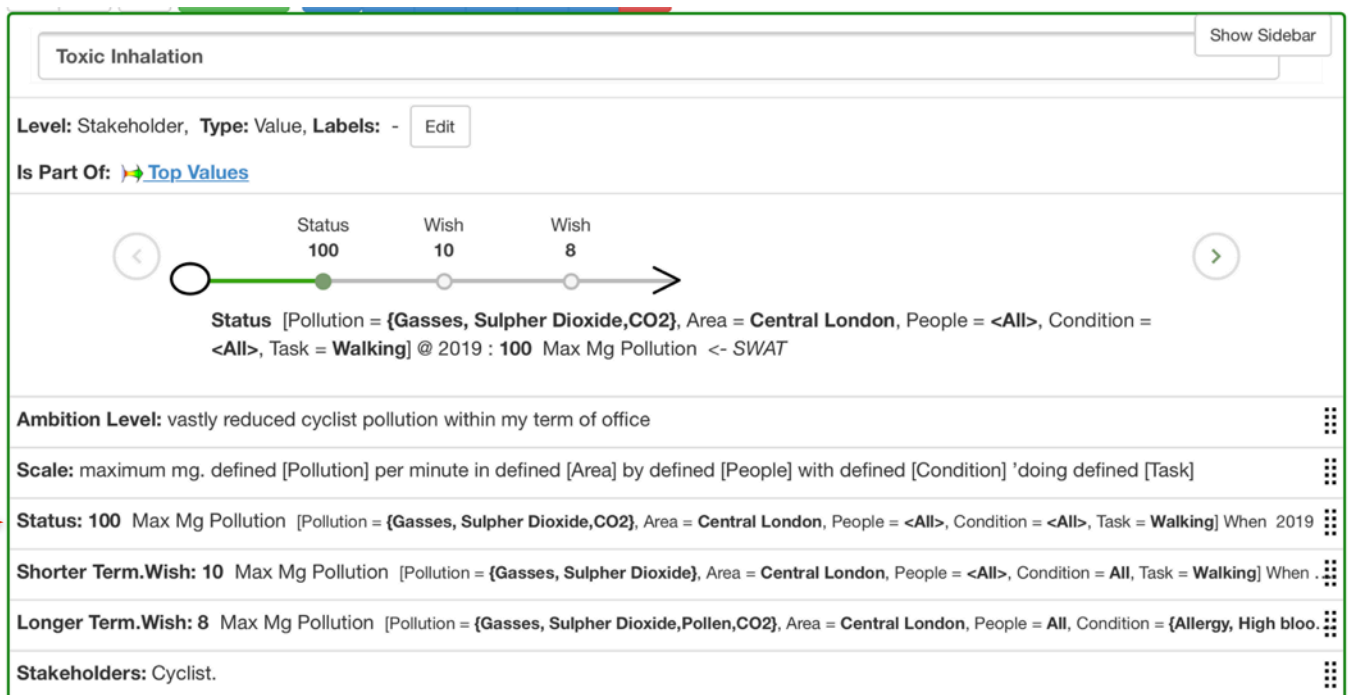


Figure 1.31 : here is Status used as an initial planning data, 'where our system is before we start delivering value increments'. It is followed by 2 different Wish levels, which have slightly different Scale parameter attributes, and different delivery dates. So the Wish levels are not completely comparable to the Status information. A signal that Status information might possibly be updated, to be comparable, for those Wish conditions, if possible.

5.4 Record Benchmark

A 'Record' benchmark is information about some extreme level of a value, good or bad.

It can be a Record level for us, or for others, like competitors.

The purpose is to stimulate us to be competitive with the best, both of our competitors, and with those in other domains using similar technology.

It is the sign of an expert that they know the Record Levels in their domain.

Keep in mind that Record setters do not stand still, but are probably trying to improve on their record. It is not sufficient to beat the old Record, you win by beating the new Record in the future.

We sometimes try to guess that using the 'Trend' parameter spec. (See Ch. 5.6 below)

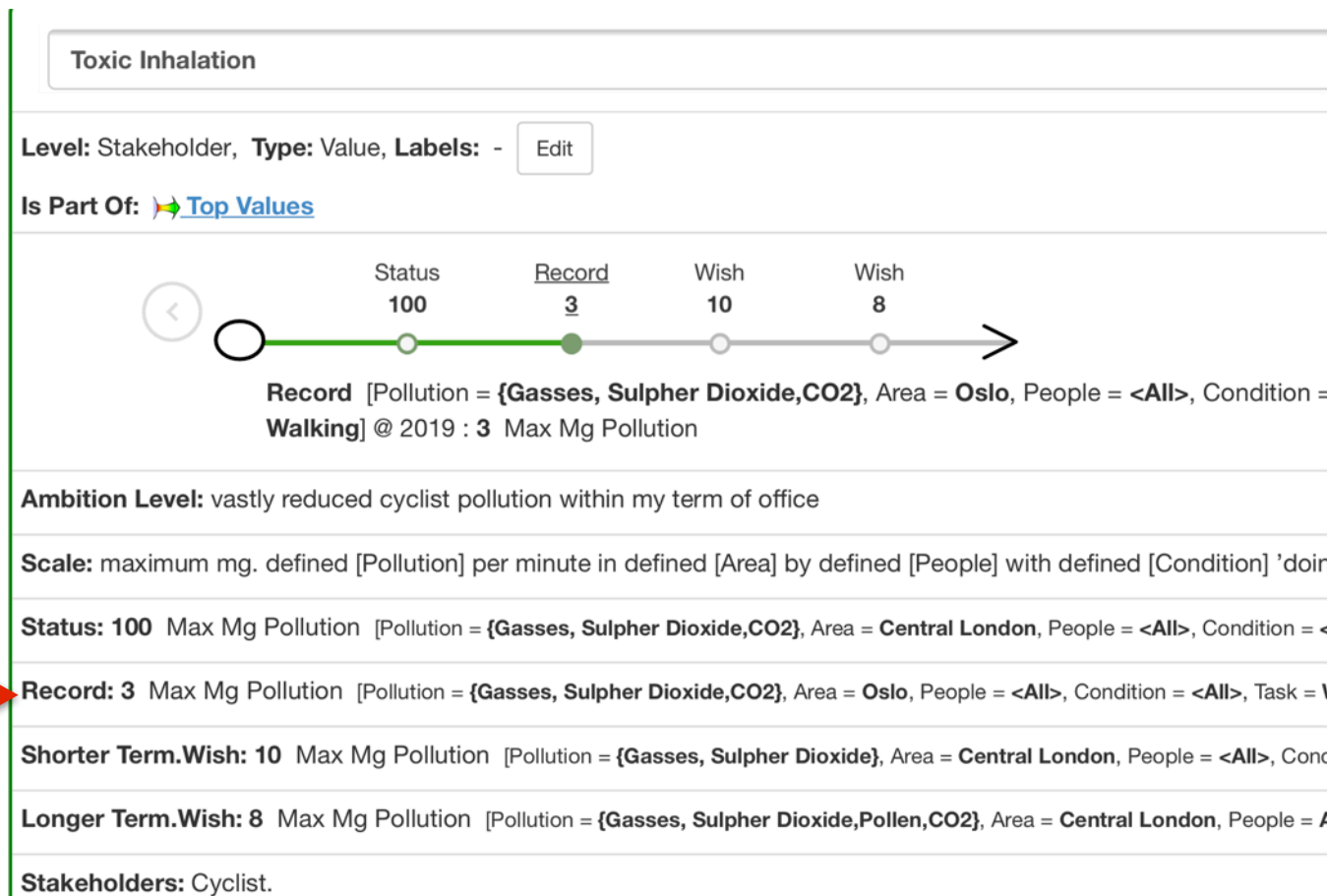


Figure 1.32 : We added a Record, comparable, but for 'Area = Oslo', and it shows that it is possible for a waterside city to get to a value level of 3. Well Oslo is not London, but what are they doing that we might learn from, and are our desired value levels ambitious enough?

The 'Record' levels can be particularly useful, because if you analyze the technology used to reach the Record level, and the costs incurred, you might come away with very useful insights.

5.5 'Ideal-level' Benchmark

The 'Ideal' benchmark is rarely specified, since it is rarely attainable. Perfection tends toward infinite costs. If ever attainable.

So I use this mainly in oral discussion to point out unrealistic ambitions. Unrealistic requirements.

Dangerous if they end up in a contract, as one of my Oslo Tech business clients CEO found out to their horror. They had contracted for 99.9999999% uptime for an airplane phone system with a big international supplier. The CTO when I asked, said no-one in the mother corporation had even done that, or knew how. So we had to 'adjust' the contract, or go bankrupt. They succeeded with the 100 person, 1 year, \$20 million project after that adjustment. Actually it was the first time they made a profit in several years. The marketing chief had had no problems saying yes to the customer's 'Ideal' . Salespeople get tempted to promise 'Ideals' which are unattainable. I assume they negotiated a more realistic availability level (like 99.98%).

Case 1N.

People are regularly specifying things like '24/7'. Which sounds like 100% availability to me.

Engineers know they can't do 100%. 99.998% is fine!

If necessary, specify Ideals *formally*, to erase all doubt.

Ideal: ZERO Pollution of any kind in London, ever and forever. Not planned yet!

Example 1O.

5.6 'Trend-level' Benchmark

The 'Trend-level' benchmark is an attempt to stop looking in the rear-view mirror, and look out at the road traffic, coming up, *ahead* of us.⁴

This is especially important in environments which experience high rates of unpredictable change, from competitors, enemies, nature, economics, technology, and politics. That is just about all of us today, I guess.

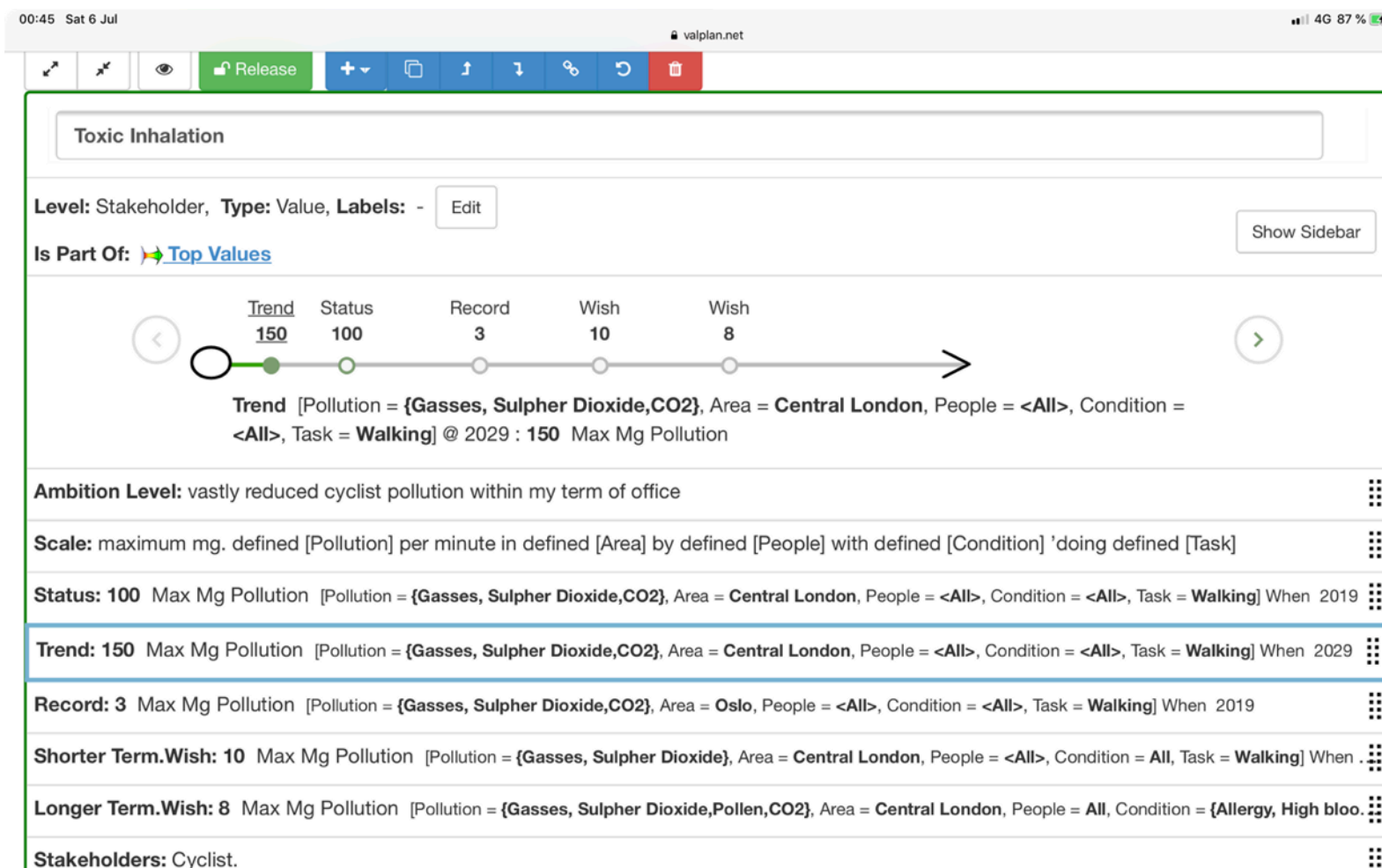


Figure 1.33 : The 10 year Trend, if we do not act, is 50% worse pollution. Useful to remind people of the alternative to funding and supporting your project. Don't assume people know such things. Research it and spell it out explicitly.

⁴ Kai Gilb invented 'Trend', in connection with Ericsson assignments. Looking ahead is very important in fast-moving competition.

6. Scalar Constraint levels.

6.1 'Tolerable-level' Constraint.

Formal Planguage definition:

<http://concepts.gilb.com/definition-Tolerable-Limit>

The 'most critical' value requirement level is the 'Tolerable' level.

It defines the borderline between **failure** (below Tolerable, Intolerable) and **not-failure** (Tolerable).

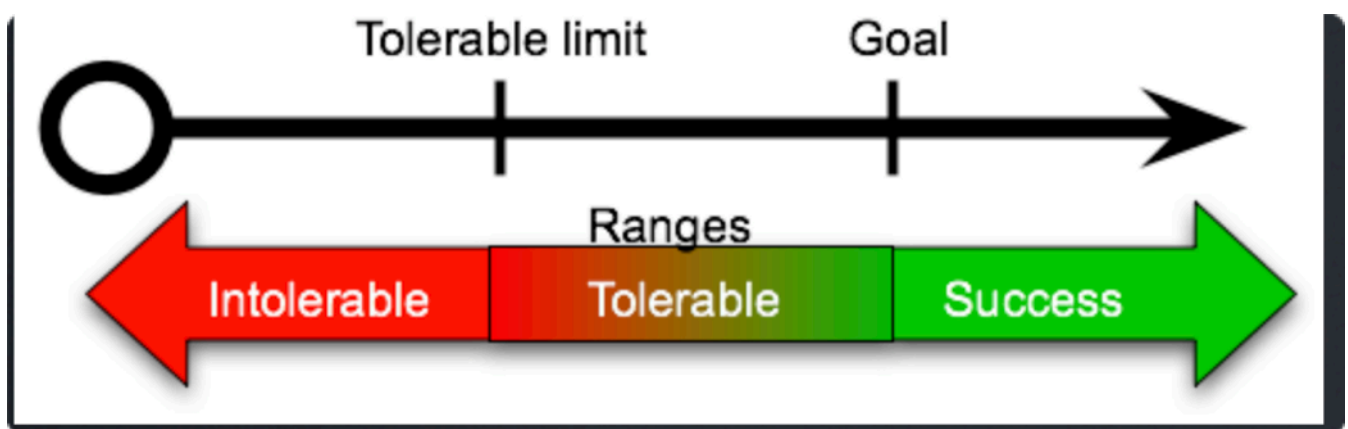


Figure 1.34 : the Tolerable Level, or Tolerable Range is just above the 'Intolerable level, and is a range extending until a 'success level' is defined. It is possible to have Intolerable levels and ranges at both extremes of a value scale, as in too hot and too cold.

Setting such constraints is mainly subjective. There is only rarely a 'cliff edge' at that point. But it is better to have clearly-defined fail/not-fail borders, than to leave your team in confusion about the borders.

This can easily have contractual implications, and you don't want to pay legal staff to argue in court about the meaning of 'sufficient' just because you did not make up your mind in the first place, in the requirement.

People would be wrongly motivated if they focused on just getting barely to the Tolerable level, at the edge of the border. Their main motivation *should* be:

- To get *well clear* of the Intolerable area, quickly, immediately.
- To create a safety margin by being *well-above* the borderline.
- To relax further efforts here, this particular value, until all other critical values, were also well clear of Intolerable dangers.
- Then to march on, towards target levels, like Goal, which define *success*.

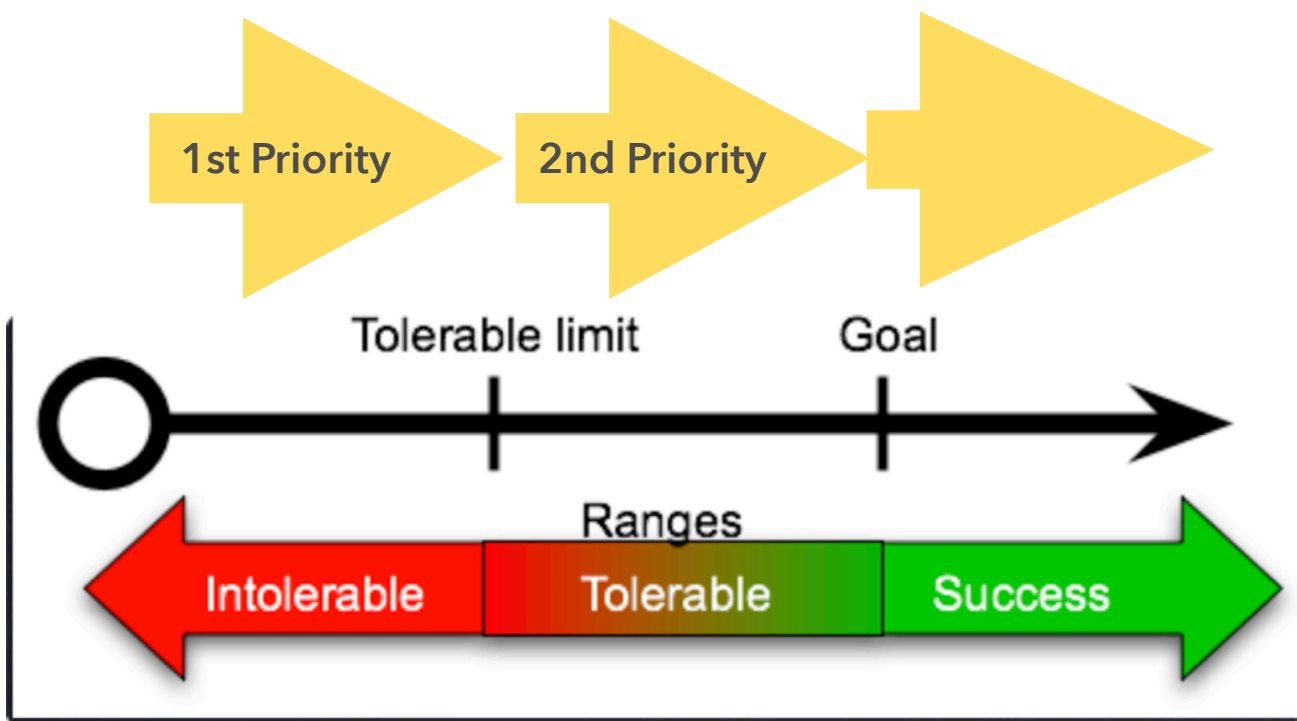


Figure 1.35 : Getting out of Intolerable levels of a critical value, is our first step. Then incrementing the value until we reach all success target levels (like a Goal). If any resources remain, we can choose to use them to increment values to more than ‘just barely’ success. Perhaps towards Stretch levels, or to longer term, and special conditions, success levels.

Because our top-level critical values are ‘critical’, meaning ‘critical for the entire system, product or service’, then as a rule, if even only one single top-level value requirement, fails to reach the Tolerable level, this probably implies failure of the *entire system*.

As a simple example, if all other top-level critical values are Tolerable or better, and the availability of the system is below the Tolerable level, say nearer 0%, then by definition none of the system functionality is available, most of the time. And none of the other value attributes, such as work capacity, usability, and security, are available. So this describes total system failure.

The determination of the Tolerable Level is a matter for the relevant stakeholders, and their practical needs and experience. At

what point do they throw up their hands and say “I give up”, and use alternative ways to satisfy their needs?

The Tolerable Level might also be set by other stakeholder needs such as legality, conformance to standards, economic profitability, or a first rough guess at the right level.

6.3 Constraints are a Dynamic Prioritization Tool

Another insight into applications of the Tolerable Level is that it is a powerful tool in helping us manage priority *dynamically*, that is, managing ‘step by step as we deliver value’.

Once we have reached a single Tolerable level, we need to ask ourselves (project management) if we should ease off on delivering more value, just yet, to this particular value.

We need instead to ask if any *other* critical values are still under their *own* Tolerable levels, and divert resources immediately to the task of getting *all critical value requirements* to at least Tolerable levels.

We need to get the system into ‘Tolerable conditions’ with respect to all critical values, *before* we plunge forward to satisfying Target levels for the critical values.



Figure 1.36 : Sailing re-

quires

dynamic prioritization. Source: "To Catch a Butterfly: Epistemic Miracles of Serendipity. The.xel.io

<http://te.xel.io/posts/2018-03-04-to-catch-a-butterfly-epistemic-miracles-of-serendipity.html>

6.4. Several different Tolerable levels might be appropriate for different circumstances.

One single value might well need to specify a variety of different Tolerable levels for different circumstances.

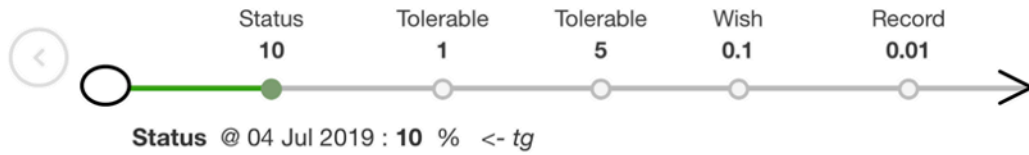
This avoids over-generalization of requirements, with consequent unnecessary costs for some circumstances.

And it supports our need to focus on particularly critical circumstances early, delivering value to those circumstances early.



User Error Frequency

Level: Product, Type: Value, Labels: -



Ambition Level: Reduce user errors when using our services

Scale: % of User actions which they correct or change

Status: 10 % When 04 Jul 2019

Tolerable: 1 When 29 Sep 2021

Tolerable: 5 When 29 Sep 2020

Wish: 0.1 % When 07 Jul 2027

Record: 0.01 When 29 Sep 2019

Figure 1.37: example with 2 Tolerable levels, with different deadlines.

6.5 There are *other* types of 'Scalar Constraints' Defined

In my books, (VP, CE) but this one is sufficient for most purposes.

Chapter 7. Scalar Target-levels.

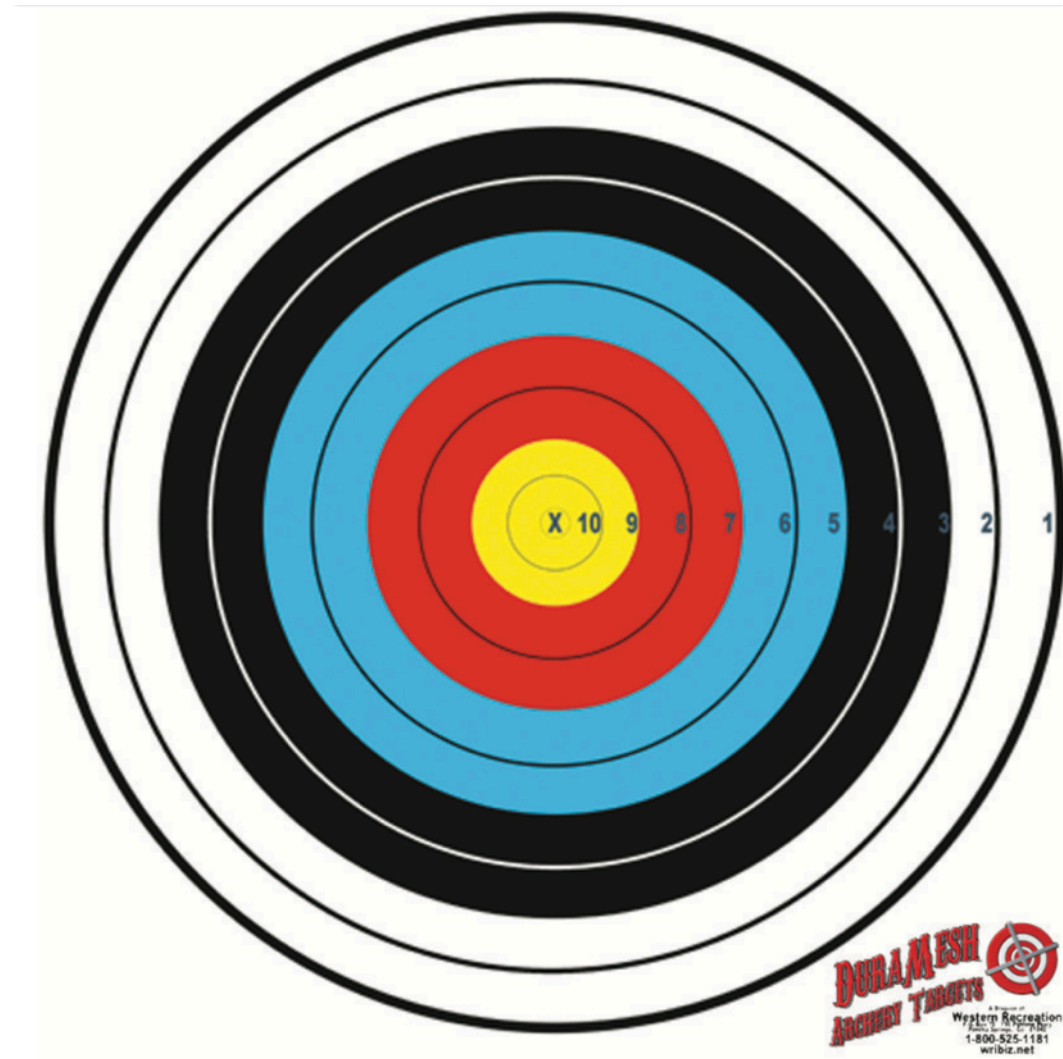


Figure 1.38 : a target level is a value level we positively aim to achieve. There are varying degrees of hitting the target. And there is such a thing as *not hitting the target at all*.

7.1 Wish-level Target

A 'Wish' specification is an expression of a *stakeholder desire*, based on *their* needs and values. It is a '*stakeholder target*', but not yet qualified as a '*project target*'.

'Wish' belongs to systems analysis: what should this project *consider* delivering? What would be *valued most* by the important stakeholders?

There can be serious problems with Wish statements, which means we cannot simply accept them as serious *project* requirements.

'The customer is always right', but they might not know state-of-the-art limitations or have infinite time and money.

Stakeholders are allowed to dream and be ambitious, but not every Wish is realistic, or is not consistent with other stakeholder needs of higher priority.

- They might be *unrealistic*, technically and economically.
- They might *steal resources* from other more-worthy requirements and stakeholders.
- They are usually expressed *without* the stakeholder having *any overview* of all *other* Wishes and constraints.

Wish statements are our formal acknowledgement that we have analyzed the stakeholder needs, and recorded their desires.

But they cannot simply be considered serious *project requirements*.

They need to be analyzed, for technical feasibility and economics.

Then they need to be prioritized together with all other Wishes (and Goal commitments), as part of the overall system, overall economics, and overall priorities.

When 'Wishes' pass all necessary tests, feasibility, economics, priorities - they can be converted to seriously committed requirements. Like 'Goal' specs.

To commit immediately to 'User Stories', and Customer Requirements, just because we want to respect them, is not wise.

It can lead to broken promises and hurt feelings, and even at the extreme, total project failure. That is not true respect. We have to be realistic, and we have to prioritize: *always*.

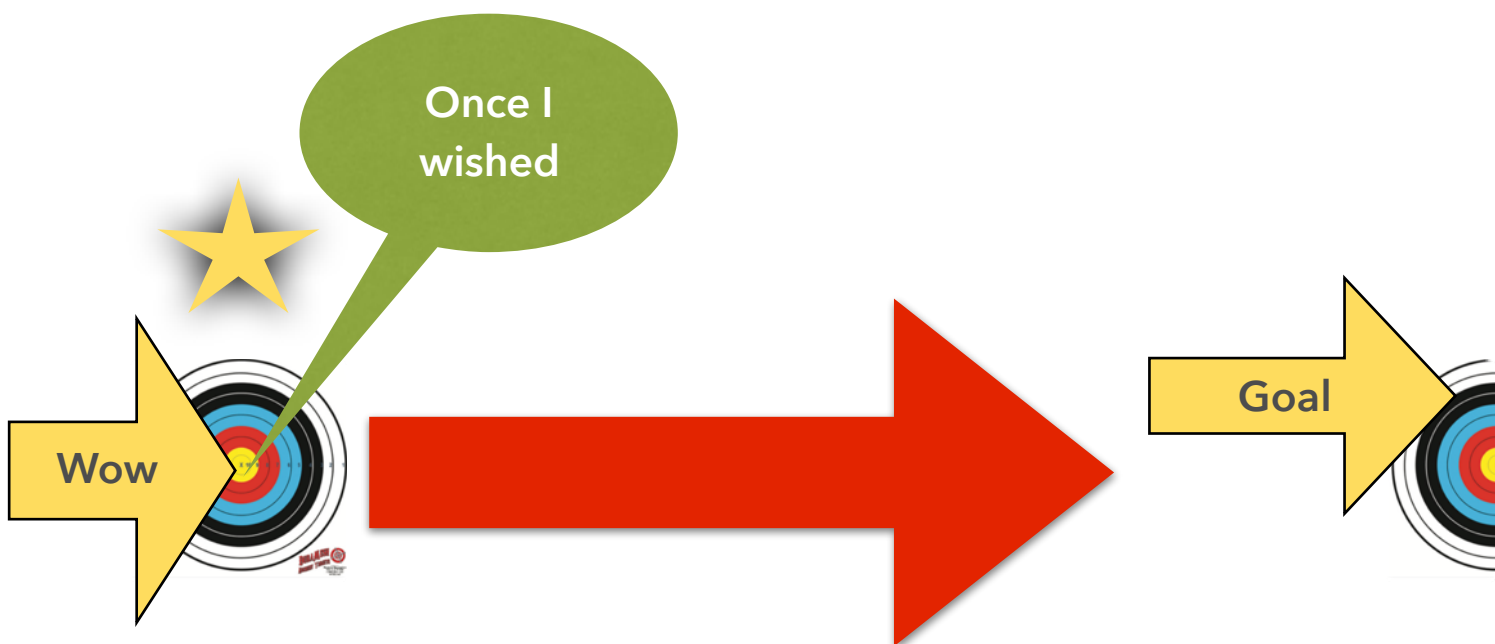


Figure 1.39 : "Santa, I want a real Tesla X for Christmas, not a toy."

The Wish must be a clear and detailed enough.

It is not good enough that the wish is almost always 'Wishy washy', highly ambiguous, like a typical Ambition-level statement, or a User Story. (see Ch. 15.1)

This is because you then, cannot really understand what is being asked for, and therefore whether it is *possible, economic*, and what its priority is.

So the Ambition Level still has to be translated into a numeric and well-structured Scale, as discussed above.

And that is not all. You have to decide exactly which Scale parameter attributes (who, what, where, and when) need exactly 'how much value level'. If not you still have a fuzzy question, and you are not going to get unrealistic answers.

For example: if the Stakeholder says:

Ambition: I want the best security, to fight hackers, and protect my customers and company.

Example 1P

Or

User Story: As a User I want good security, to fight bad guys.

Example 1Q

These are simply unacceptable statements:

Their possible range of value, and consequent technology interpretations, is far too wide. The cost range is roughly zero to infinity.

22:00 Sat 6 Jul valplan.net

Tag.Scale:

% of [Security Results] for [Attacks] carried out by [Attackers] on [Targets] with [Attack Results].

Templates ▾

Attack Results: defined as:

- No Damage, Data Stolen, Ransom Attempted, Data Corrupted, Data Spread Onward, Systems Down, Reputation Damaged, Future Business Damaged, Lawsuits From Customers, Opinion Swayed...

Attackers: defined as:

- Innocent Employees, Criminal Employees, Criminal Suppliers, Evil People, Evil Nations, Greedy Organizations, ...

Attacks: defined as:

- Denial of Service, Data Corruption, Logic Corruption, Enter Innards, Take Control of System, Steal Passwords, Steal Money, Steal Identities, ...

Security Results: defined as:

- Attack Attempt Detected, Successful Attack as Intended, Bad Results Thwarted, Perpetrator Identified, Perpetrator Reported to Authorities, Perpetrator Shut Down, Our Security Procedures Improved,

Targets: defined as:

- Individuals, Groups, Organization, National Interests, Data, System Control,

Figure 1.40 : in this case I took the Ambition Level statement (“I want the best security to fight hackers and protect my customers and company”), and created a Scale for Security with appropriate Scale Parameters (‘Attack Results’, etc.).






I then defined all 5 Scale parameters (Fig. 1.40) with a reasonable set of attributes. Anything forgotten can easily be added later, as we go.

Can you begin to see the need for detail in this Security problem?

The 'Ambition Level' hides all of it.

Ambition Level: I want the best security to fight hackers and protect my customers and company.


Scale: % of [Security Results] for [Attacks] carried out by [Attackers] on [Targets] with [Attack Results].

 **Tag.Wish:**    


dd/mm/yyyy


Qualifiers:

[Security Results] =

[Attacks] = 

[Attackers] =

[Targets] = 

[Attack Results] = 

Source: by tomgilb - Jul 6th 2019, 22:03

Figure 1.41 : after defining the Scale, I drafted my first Wish level. (Built on Fig. 1.40)

Note that it is a very small subset of all the Security Scale possibilities.

That is good. I can focus on this slice of the action, if it is high priority and critical.

I have a fair chance to understand it, and find security options and cost them.

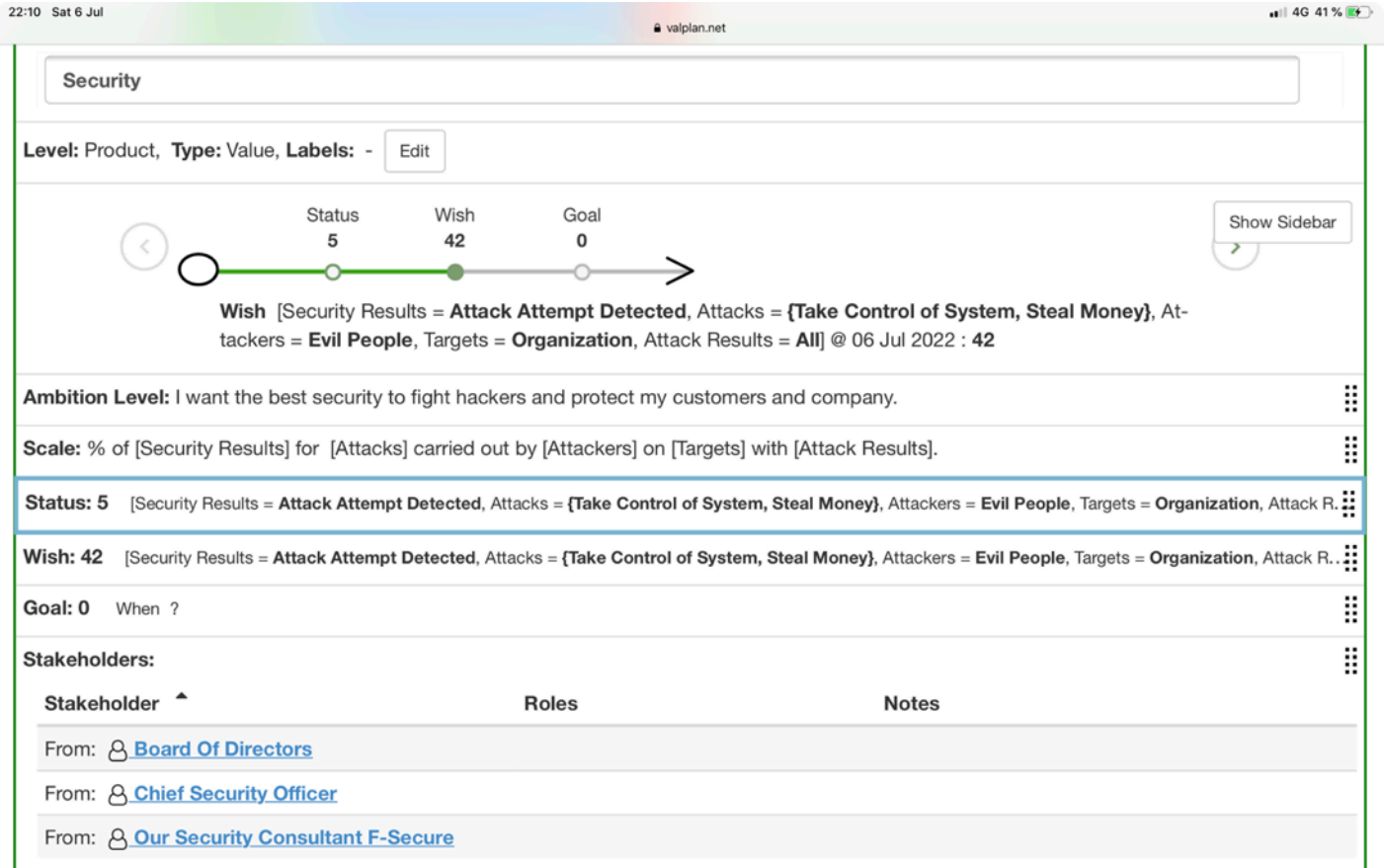


Figure 1.42 : So now I have my Wish specification. But I cannot possibly commit to a Goal (a 'firm committed promised value delivery from a funded project') because I have not identified and costed the necessary technology for delivering the Wish Level (42%) on time, 6 July 2022.

But at least the 'Wish' problem is much clearer.

We can understand, and discuss with our stakeholders on a much more realistic basis, than with the Ambition Level.

7.2 Goal-level Target

We have, in Planguage, defined a 'Goal' specification as a *project level commitment* to deliver that Goal value level, together with all concurrent project Goal levels, by their individual deadlines, with the *currently* allocated resources (money, time, equipment, etc.).

This is a specialized use of the term Goal, and not at all the same as the many loose uses of 'goal' in other situations.

Having well-defined concept terminology is important for clear and consistent thinking. One reason we write **Goal** with a **C**apital letter, is to signal that it is a defined-concept term.

You are not at liberty, using Planguage, to use it with an entirely different meaning. That would introduce confusion, and lead to failures. At least then, don't Capitalize it ! "We scored a goal". "My manager said he wants to have a meeting about his goals".

As with all other defined levels on a Scale, you can specify as many different Goal levels as is useful.

You can also add to the Goal specifications gradually, incrementally, as you get experience and insight.

Our Planguage culture is 'Evolutionary Incremental Value Delivery'. Things do not *have* to be specified all up front.

That does not work well.

Too much, gets badly done.

Security

Level: Product, Type: Value, Labels: -



C.Goal [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack Results = **All**] @ 06 Jul 2027 : **50**

Ambition Level: I want the best security to fight hackers and protect my customers and company.

Scale: % of [Security Results] for [Attacks] carried out by [Attackers] on [Targets] with [Attack Results].

Status: 5 [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack R...

A.Wish: 42 [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack...

A.Goal: 35 [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack...

C.Goal: 50 [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack...

B.Goal:

dd/mm/yyyy

Qualifiers:

[Security Results] = [Attacks] =

[Attackers] = [Targets] =

Figure 1.43 : Multiple Goal statements (tagged A, B and C).

7.3 Comment on the Multiple Goal Example Above (Fig. 1.43).

- I tagged both the Wish, and a corresponding Goal, with a tag 'A', to make it more visible that they were related.
- And by 'Tagging' any Planguage statement, I can more easily refer to it when presenting, or writing an analysis of the plans.
- Bullet points don't 'make the cut' (work well). Nor do 'yellow stickies' (kid stuff) in my world. Not even for early informal drafting of plans. We live in a complex integrated world and digital plans are the only reasonable tool for modeling it. Even just a Text editor!
- It looks like we cannot achieve the A.Wish on time, but only a level of 35%. (A.Goal)
- The good news is Goal 'C' (with exactly same Scale parameter attributes as A) we *can* achieve a higher than Wished for level of 55%, later, in 2027
- And Goal B, the ultimate 'All, All, All, All, All attributes' Goal, all security problems which are included in the Scale parameter definitions (see above Scale detail), can be achieved in the year 2030. With current funding! If you want *earlier* results, you can cross my palm with silver.
- The implication for all Goal statements is that we have done the necessary 'homework'. That means we have found a suitable ar-

chitecture or design, that will give the stated Goal level, by the stated deadline for the particular Goal level.

- We have in addition found that the 'technology solution costs' are within our 'resource budgets' (see later in the book, Chapter 12), when regard is paid to all *other* Goal commitments in the same time frame, or project. We know that because otherwise we cannot commit a Goal.

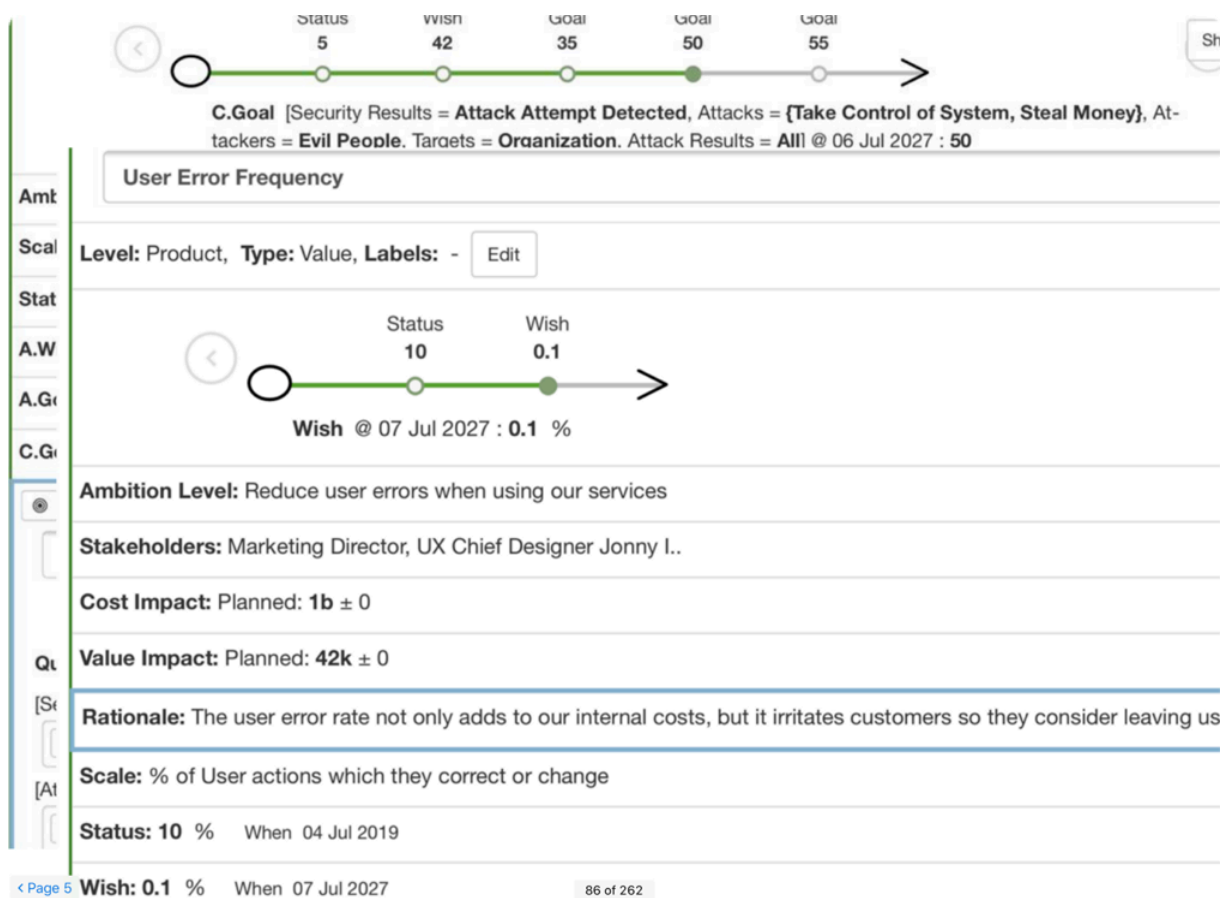


Figure 1.44 Simple example of a 100x better Wish level than the Status benchmark.

- B.Goal is presented here in a *detail* window, so we can see more detail.
- The others are summarized in a one-liner format. Except C.Goal which has also been selected for display in fuller detail, in the

'graphic summary arrow' at the top. Which summarizes all defined levels on the value Scale.

7.4 Stretch-level Target

A 'Stretch' parameter specification is an *even greater* level-of-success than we dared to commit (in a Goal = commit) to initially.

We are *not sure* if we have the resources, or even technology, to actually get to Stretch levels. But, Stretch means that some stakeholder, sees *some* specific value, in getting to that level, sooner rather than later.

So 'Stretch' is the 3rd Priority (1 = Survive (Tolerable), 2 = Succeed (Goal), 3 = Surpass (Stretch level)).

When all top-level critical factors have reached initial Goals, for this point in time, you have formal permission to choose a Stretch level of that value (or another value), and see if you can reach it, before money, or time, run out.

Stretch can be treated as a sort of 'Stakeholder Wish', that we *cannot* commit to (a 'Goal') in the current time frame. But we might just get lucky! Exceed formal planned expectations.

00:17 Sun 7 Jul valplan.net 4G 23%

Security

Level: Product, Type: Value, Labels: - Edit

Show Sidebar

A.Stretch [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Attack Results = **All**] @ 06 Jul 2023 : **38** %

Ambition Level: I want the best security to fight hackers and protect my customers and company. ⋮

Scale: % of [Security Results] for [Attacks] carried out by [Attackers] on [Targets] with [Attack Results]. ⋮

Status: 5 % [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Atta.. ⋮

A.Wish: 42 % [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Att. ⋮

A.Goal: 35 % [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Att. ⋮

A.Stretch: 38 % [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, . ⋮

C.Goal: 50 % [Security Results = **Attack Attempt Detected**, Attacks = **{Take Control of System, Steal Money}**, Attackers = **Evil People**, Targets = **Organization**, Att. ⋮

B.Goal: 55 % [Security Results = **All**, Attackers = **All**, Attack Results = **All**, Attacks = **All**, Targets = **All**] When 06 Jul 2030 ⋮

Stakeholders: Board Of Directors, Chief Security Officer, Our Security Consultant F-Secure. ⋮

ValPlan | Gilb International AS | [Terms Of Use](#) | [Privacy Policy](#) | [Cookie Policy](#)
 Powered by Needs & Means © RSBA Technology Ltd 2015-2019

Figure 1.45 : Adding a Stretch level for the 'A' value conditions.

Chapter 8. Background Specifications.

'Background' Planguage specifications are added to the core requirement (the 'really required' stuff), and intermixed with those 'core' requirements (Scale, Tolerable, Goal).

Background specs are part of a Planguage culture which believes that rich 'written specs' beat 'human memory'. **Digital** written information persists better, and allows 'smarter' apps.

This is true, as projects scale up, and become geographically decentralized. Simple 'local group' methods do not work any more, like 'stand up retrospectives'.

People have to find project information, and share their own project-information wherever they are, and whenever they are ready to do so.

The *written* info, and especially *digital* info is the right direction.

Specifications are not simply drafted, they accumulate, over time, from many sources, and much feedback and learning. We need to deal with the dynamics of this.

At the same time, even in the largest of projects there is a right time for video face-to-face meetings, to build trust, motivate, and ferret out project info, that needs writing down, to share with everybody, sooner or later, if useful.

One data-detail can be the difference between project or value failure and success.

Capturing *digitally* is cheap, compared to forgetting a critical detail.

I think it is important to distinguish between old-fashioned written cultures (paper, copies, issued infrequently, available to few), and a **digital written culture** (internet, app based, continuously updated, from anywhere on the planet, structured for automated analysis, connected to intelligible data sets).

Some of the prejudices against written bureaucratic cultures (and quite right these negative reactions were) were based on the 1990s pre-internet experiences. 'Written' is *not* what it was *then*.

Our Planguage requirements ideas grew up, digitally, but before the internet, but by looking at the capability and potential of the ValPlan.net app, we can see a powerful current capability, and also a potential, for well-structured, well-defined requirements (and other project specifications).

We detest unnecessary bureaucracy! But *some* degree of 'bureaucracies' payoff, and we have to know the difference.

Basis	Written communication	Oral communication
1.Record	It always has permanent record.	It does not have any permanent record.
2.Cost	Written communication is high cost.	Oral communication is less costly.
3.Feedback	Written communication it takes time to give feedback.	Oral communication it gives immediate feedback.
Basis	Written communication	Oral communication
1.Record	It always has permanent record.	It does not have any permanent record.
2.Cost	Written communication is high cost.	Oral communication is less costly.
3.Feedback	Written communication it takes time to give feedback.	Oral communication it gives immediate feedback.
4.Flesibility	Written communication is rigid or inflexible.	Oral communication is highly flexible.
5.Time taken	Written communication it takes more time to prepare and transmit message.	Oral communication it takes least time to prepare and transmit message.
6.Reliability	Written communication is most reliable.	Oral communication is not reliable.
7.Legality	Written communication is legal evidence.	Oral communication is not legal evidence.
8.Distortion	Written communication is not possibility or distortion.	Oral communication is high possibility of distortion.
9.Effectiveness	Written communication is not effective as oral communication.	Oral communication is most effective communication.
10.Significance	Most significant in all type of organizational context.	Less significant in the organizational context.
11.Relationship	Written communication is it establishes indirect relationship between parties.	Oral communication is it establishes direct relationship between parties.
12.Formality	It maintains formal communication relationship	It maintains informal communication relationship
13.Emotion	Written communication is seldom affected by emotion.	Oral communication is affected by emotion.
14.Preservation of information	Written communication it is possible to preserve and may be used in future.	Oral combination it is only face-to-face communication. So, preservation of information is quite impossible.
15.Educational qualification	Both parties must be educationally qualified.	If any party is illiterate, then oral communication is suitable.
16.Media	Its media are written in nature such as letters, memos etc.	Its media are oral in nature such as telephone, talks face-to-face discussion etc.
17.Response	Written communication quick response is impossible for this communication.	Oral communication quick response is possible for this communication.
18.Delegation of authority	Here, delegation of power is suitable.	Here, delegation of power is not suitable.

Figure 1.46 A Comparison of Written and Oral Communication Attributes.

Source: (This source is worth looking at for more detail).<https://thebusinesscommunication.com/difference-between-oral-and-written-communication/>

I excuse the poor grammar and spelling! It is a full set of ideas. And it makes the point that the written/oral comparison has many factors to consider. Even *before* we look at digitization.

Most of the 'Basis' attributes could do with a Scale definition. And then we could numerical-ly evaluate the modes of communication (after they too were better-defined and technologically updated).

8.1 The *General* Purposes of Background Specifications.

The 'Background Spec' purpose is to *enrich* the requirement spec with information, that

- Might *never* otherwise get specified
- Might be '*lost*' in earlier or later documents, like slide presentations and Business Analysis
- Might *not get used seriously* unless they are 'in your face' in the spec.
- Might be *difficult to retrieve* from other documents, or from human memory
- Might be well-known to some, but *unknown* to others
- Might be correct and updated for some people, but *incorrectly remembered and not updated for others.*
- Is needed for ongoing, real time, incremental steps, of value delivery *decision-making*
- Is needed for *risk management, prioritization, taking responsibility, motivation, reviewing efficiently:* and any other purposes on the path to successful value delivery, without being delayed by poor decisions, based on lack of correct information.

- Is needed to enable *automation* of certain aspects of requirements, such as Quality Control, prioritization, presentation, and risk detection. Yes AI. A complete Specification 'Object'.
- Help to manage the *updating and changes* to the spec.
- Help us to *follow our adopted defined Rules* (specification standards for requirements)

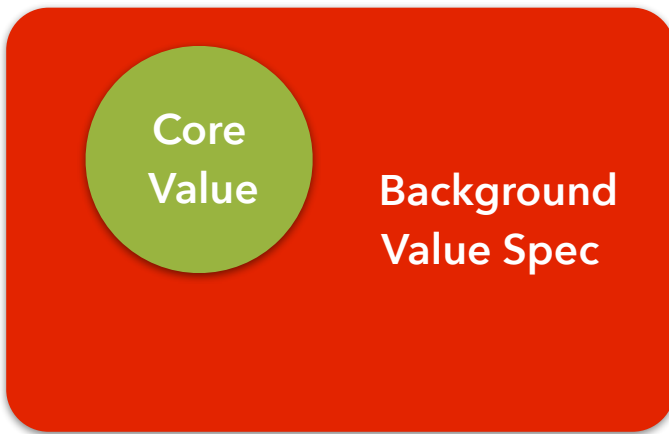


Figure 1.47 : Core Value specification, surrounded by supporting requirement information (background).

We have already encountered some Background Planguage specs earlier in this book. For example Ambition, Type, Tag, Stakeholders, Status, Level, Past.

User Stories have two kinds of background, built in to their structure: *who* is the stakeholder, and *why* do we need this requirement (Justification, or Rationale). Good, but not nearly enough (Ref. A, US). Simple, but 'too simple' for serious purposes.

8.2 Risk Management with Background Specs

Risk management means, reducing the losses in your scope of work, due to any causes which might somehow be dealt with by better planning, and by better plan specification.

Ericsson of Sweden had a deep insight when in their Quality Policy (see quote in VP Book) they declared that risk management is the job of every engineer, at all times.

I also believe that in the area of requirements, every little detail of specification has a potential of unleashing, or ignoring, risks.

Risks which are very much larger in consequence, than any cost associated with reducing the risks. We do this risk management by having more-solid requirements *craft*-capability. Attention to detail.

Let me be more direct. I believe that every detail in *this* book, about Planguage Value Requirements, are potential tools for reducing risks, systematically.

This is not a co-incidence. I am by nature a very risk-adverse person, so I *designed* Planguage to deal with requirements-and-systems risks.

Let me use as an example: the major idea in this book.

Quantification of the Value requirements.

- If a Value Requirement is **not quantified**,
- you *immediately* incur a huge and unnecessary **risk**. I. E.;
- Nobody can understand the requirement, in the same way.
- People will waste time and money working towards their private interpretation of the requirement ('better security').
- At worst, the entire project can fail for this one thing alone (plenty of projects fail now)

Some other scattered examples of Background spec details, related to Risks

- If the requirement is not **tagged**, giving it clear-long term identification, then it might be missed in test planning, and be delivered in failed condition.
- Too many value requirements are just bullet points on a slide
- Tagging a spec means we can reuse it, and avoid the risk of different copies or versions of the same spec.
- If we do not give the **Source** of a requirement, then quality control of its current validity, is less likely, and bad requirements might get implemented.
- If we do not capture the **Ambition Level** and its source, then we risk losing alignment with higher-priority objectives, and their possible changes, like from new management.
- If we do not assign a **Specification Owner** to each single requirement, there is a high risk that no one will be qualified and motivated, to keep the spec properly updated, and to keep it high quality. An 'orphan' specification, without an Owner.

Security

Level: Product, Type: Value, Labels: - Edit

Show Sidebar

C.Goal [Security Results = **Attack Attempt Detected**, Attacks = {**Take Control of System, Steal Money**}, Attackers = **Evil People**, Targets = **Organization**, Attack Results = **All**] @ 06 Jul 2027 : **50** %

Owner: David Bishop

Assumption: AssumptionConsequenceFull requested budget allocated by JanuaryIf not, delay project start

Issue: IssueActionWill Government pay its usual 50% for Security investments. We need to request written commitment and budget for this.

Dependencies: Projects cannot start without written approval of Security requirements here from Chief Security Officer.

Risk: RiskMitigationGovernment change or EU exit can remove financial support for EU security level.Make written agreement w current Ministers

Rationale: The level of security is required for government systems under EU rules and applies even after Exit, if we deal with the Common Market.

Ambition Level: I want the best security to fight hackers and protect my customers and company.

Scale: % of [Security Results] for [Attacks] carried out by [Attackers] on [Targets] with [Attack Results].

Status: 5 % [Security Results = **Attack Attempt Detected**, Attacks = {**Take Control of System, Steal Money**}, Attackers = **Evil People**, Targets = **Organization**, Atta...

A.Wish: 42 % [Security Results = **Attack Attempt Detected**, Attacks = {**Take Control of System, Steal Money**}, Attackers = **Evil People**, Targets = **Organization**, Att...

A.Goal: 35 % [Security Results = **Attack Attempt Detected**, Attacks = {**Take Control of System, Steal Money**}, Attackers = **Evil People**, Targets = **Organization**, Att...

A.Stretch: 38 % [Security Results = **Attack Attempt Detected**, Attacks = {**Take Control of System, Steal Money**}, Attackers = **Evil People**, Targets = **Organization**, ...

Figure 1.48 : some Background Specifications have been added to the Security specification. Owner, Assumption, Issue, Dependencies, Risk, Rationale

Hopefully you can guess how such specs help us to see and manage risks

8.3 Prioritization using background specs

In my 'Evolutionary Value Delivery' culture⁵, all Value requirements are not equal. We are going to start a stream of value improvement deliveries. So we have to figure out which Value deliveries are smartest to deliver early.

There is no simple method to help us decide what to do first or next, which will be realistic, and give the most satisfactory results. There are far too many different dynamically-changing factors, which influence a prioritization decision.

So, our best suggested approach today is to collect *prioritization*-information, directly in the *requirement* specification. This information can be used to help you decide *which* Value levels to prioritize.

A simple example might be that Value X has 3 Stakeholders interested, one of whom is the Government, and Value Y has 2 stakeholders interested, one of whom is your boss.

You cannot deliver *both*, in the coming time period, you must choose one. What would your boss advise?

Wait, it is never so simple! Value X delivery has an estimated return of 300% annually, and Value Y has only 120%.

⁵ The project management process known as Evo. See CE book Chapter 10: Evolutionary Project Management: <http://www.gilb.com/DL77>

The bad news is that there are many *more* factors to consider in this case.

You can always simplify, and ignore those factors. But sooner or later, somebody is going to ask why Factor XYZ was not considered ('Sales with Potentially large new customers', for example).

Prioritization can be complex⁶.

⁶Managing Priorities, A Key to Systematic Decision-making. With Mark Maier, 2005 (paper)

<http://www.gilb.com/DL60>

Automatic Value Delivery Prioritization

Right now we have enough *digital data specification parameters* in Planguage and ValPlan, to automate a 'delivery sequence prioritization' based on:

- a set of many required values - total value effect
- A set of budgeted resource costs, money-time-people, one time, recurrent
- The level of uncertainty of our estimates,
- This is *after* we have done a level of *design*, so we have real implementable action steps for the values.
- And we have really today automated the priority sequence choice, in our app. Anyone can do this on a spreadsheet too. You just need the clear, digital, value-data.

This does show the potential for automated prioritization calculation (AI). But this calculation does leave out a large number of factors, that should be used to influence prioritization

At the moment, these other factors can be considered, manually, if they are made visible and clear in the Background information of the requirement specification. We do this now, with an eye to further enhancing the information specified, to the point where it can be used to make even better automated prioritization decisions for us. (See Ref. E) My Deeper Priority Writings

TOP LEVEL KNOWLEDGE PROJECT IET

From Level: *Level?* To Level: *Level?*

Settings...
+ Add ▾
↔ Sort ▾
📄 Duplicate...
↶ Undo...
=: ABSOLUTE
🔗 Help me!

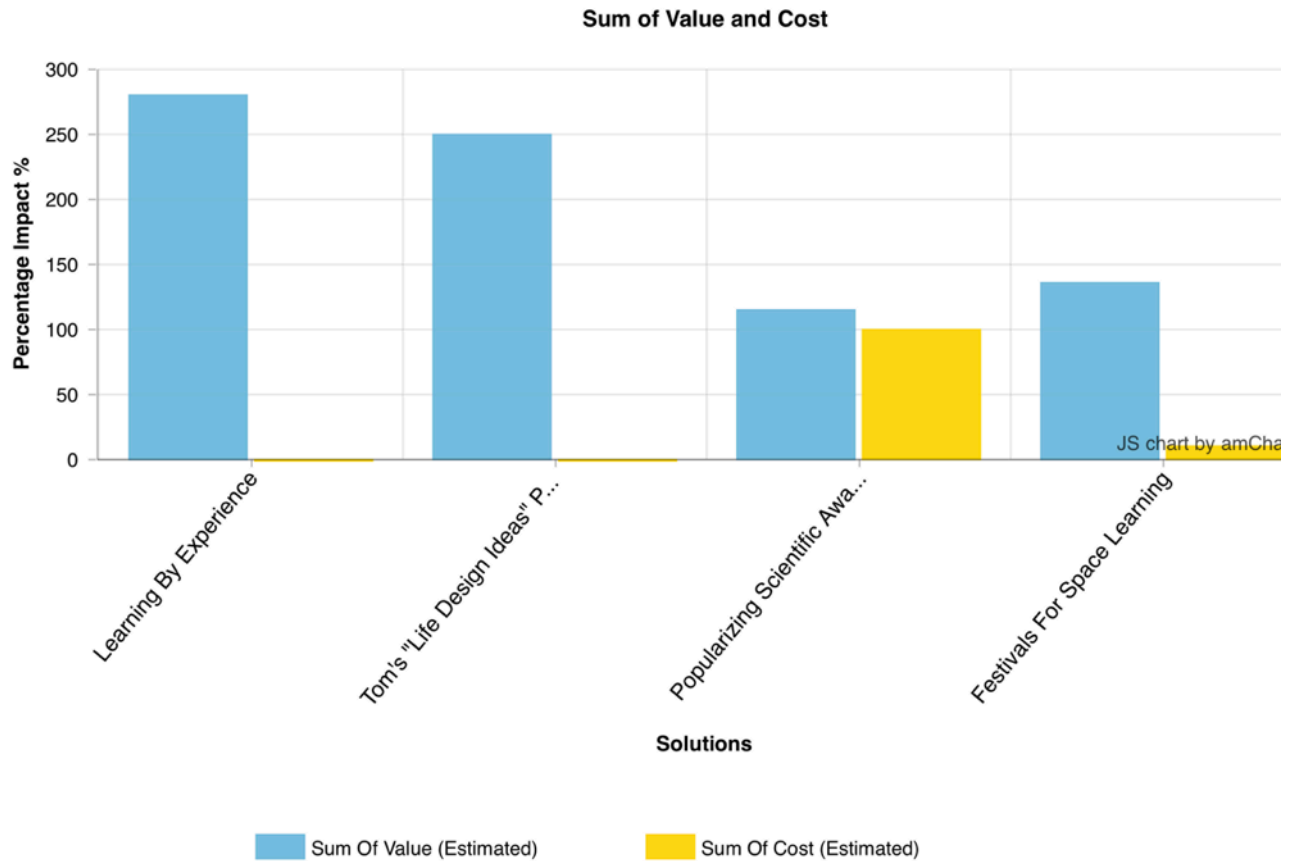


Figure 1.49 : an automated sort, best option at left, for delivering the greatest set of values, at the lowest set of costs, with regard to risks and uncertainty.

Source: Masterclass, Katowice, exercise on spreading knowledge nationally. 2018. This chart uses data in the Value Specification, along with a lot of other factors to optimize the flow of value for money.

23:31 Sun 29 Sep Status @ 04 Jul 2019 : 10 % <- tg

Ambition Level: Reduce user errors when using our services	⋮
Scale: % of User actions which they correct or change	⋮
Status: 10 % When 04 Jul 2019	⋮
Tolerable: 1 When 29 Sep 2021	⋮
Tolerable: 5 When 29 Sep 2020	⋮
Wish: 0.1 % When 07 Jul 2027	⋮
Record: 0.01 When 29 Sep 2019	⋮
Authority: The Financial Conduct Board looks at errors as a sign of misconduct.	⋮
Intended Readership: This spec must be intelligible to all manager stakeholders inside and outside, without help.	⋮
Owner: The Spec Owner is the Chief UX Designer.	⋮
Responsible: Responsibility for delivering this U E F value on time is the Project UX Designer.	⋮
Implementation Instance: Implementation will be carried out by our subcontractor Accent.	⋮
Dependencies: We are totally at the mercy of our Steering Committee abroad, which has little time for us.	⋮
Test: All testing to measure value delivery levels will be carried out by Tata Consultancy in Bangalore.	⋮
Stakeholders: Marketing Director, UX Chief Designer Jonny I..	⋮
Rationale: The user error rate not only adds to our internal costs, but it irritates customers so they consider leaving us for competitors!	⋮
Cost Impact: Planned: 1b ± 0	⋮
Value Impact: Planned: 42k ± 0	⋮

Figure 1.50 : we added 4 examples of background Parameters to the User Error Frequency Value spec. Hopefully you can see that each one might contribute to a prioritization decision.

See: Stakeholders, Cost Impact, Value Impact, and Rationale


8.4 Responsibility and Motivation with Background specs

22:55 Sun 7 Jul valplan.net

Ambition Level: Reduce user errors when using our services

Tag.Stakeholders: 🗨️ 📄 + 🗑️

🔗 Link to existing... + Link to new...

Stakeholder ^	Roles	Notes	Ac-tions
From: 👤 Marketing Dire... ▾	<ul style="list-style-type: none"> × Decision Maker × Funder 	<ul style="list-style-type: none"> ✓ Main power behind this need 	
From: 👤 UX Chief Desi... ▾	<ul style="list-style-type: none"> × Authority × Internal × Expert 	<ul style="list-style-type: none"> ✓ Main enable of improved value 	

Enter additional information

Source: by tomgilb - Jul 7th 2019, 22:47

Tg

Templates ▾

🗨️ Add Comment...

Cost Impact: Planned: 1b ± 0

Value Impact: Planned: 42k ± 0

Rationale: The user error rate not only adds to ur internal costs, but it irritates customers so they consider le...

Scale: % of User actions which they correct or change

Figure 1.51 : This is a *detailed window* for the above Stakeholders spec, for the 'User Error Frequency' value spec, summarized above.

We now can see some more information about the roles and power of the stakeholders involved for this one value.

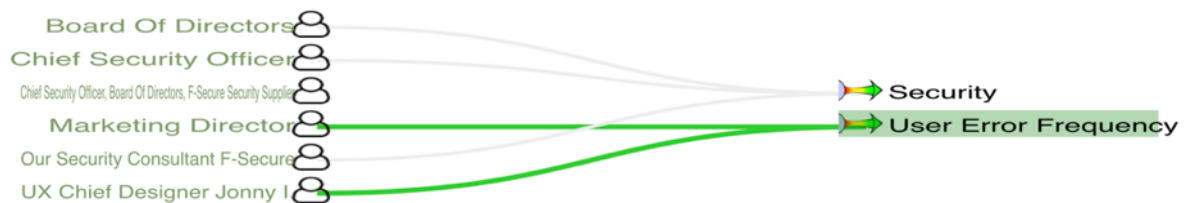


Figure 1.52 : the relationship between stakeholders and values is digital, so we can display it graphically (automatic thick lines to show relationships). Useful when things get voluminous and complicated.

That information is even more useful than just knowing the *names* of the stakeholders.

There is already enough digital information in the 'Roles' categories to help us prioritize this value better automatically, when we decide to write the code to use that data. We are moving in that direction, as you see.

Would you prioritize *Funders* over *Authorities*?

Here, below, is a set of Planguage parameters added to the *User Error Frequency* value requirements spec. These Background Value Spec Parameters clarify and assign formal responsibility for different aspects of the value requirement.

It is worth pointing out that we now have a practical tool for decentralizing authority, for delegating authority, to people and groups who are interested in having it, who have or *will make* the time to do things properly, and are specialist in this area.

I have observed that the moment you put someone's name on a responsibility voluntarily, it gets taken quite seriously. It is their baby and their honor! (See the Owner spec. Below)

00:02 Mon 8 Jul valplan.net

User Error Frequency

Level: Product, Type: Value, Labels: -

Status 10 Wish 0.1

Wish @ 07 Jul 2027 : 0.1 %

Ambition Level: Reduce user errors when using our services

Authority: The Financial Conduct Board looks at errors as a sign of misconduct.

Intended Readership: This spec must be intelligible to all manager stakeholders inside and outside, without help.

Owner: The Spec Owner is the Chief UX Designer.

Responsible: Responsibility for delivering this U E F value on time is the Project UX Designer.

Implementation Instance: Implementation will be carried out by our subcontractor Accent.

Dependencies: We are totally at the mercy of our Steering Committee abroad, which has little time for us.

Test: All testing to measure value delivery levels will be carried out by Tata Consultancy in Bangalore.

Stakeholders: Marketing Director, UX Chief Designer Jonny I..

Rationale: The user error rate not only adds to our internal costs, but it irritates customers so they consider leaving us for competitors!

Scale: % of User actions which they correct or change

Status: 10 % When 04 Jul 2019

Wish: 0.1 % When 07 Jul 2027

2 impact target parameters hidden. [Click here to show them.](#)

Figure 1.53 : adding specific responsibilities, as Background statements, to a Value requirement.

Chapter 9. Making Use of the Value Specification

The Value Requirement Spec is a set of data about the requirement, which can be exploited in many useful ways.

These are usually detailed in most of my other books and papers (ref. VP, CE).

So I will briefly point out the connection below, and refer you to other sources for *detail* (www.Gilb.com, VP)

9.1 Dialogue with Stakeholders

It is normal that *more than a single stakeholder* is interested in a particular Value Requirement. It is normal that they have different views of what is important about that requirement, such as delivery date and level.

There is an apparent 'conflict of interest' here. And we need to resolve matters so that one stakeholder is not accidentally harming the interests of another.

A good start here is that each Value systematically lists, as background information, all critical stakeholders for the value. Then we can see explicitly who is interested, and consult with them to avoid unnecessary conflict of interests.

In general, the more interested stakeholders are, for one Value, the more political and funding support for delivering that value early, and getting to some useful Value level.

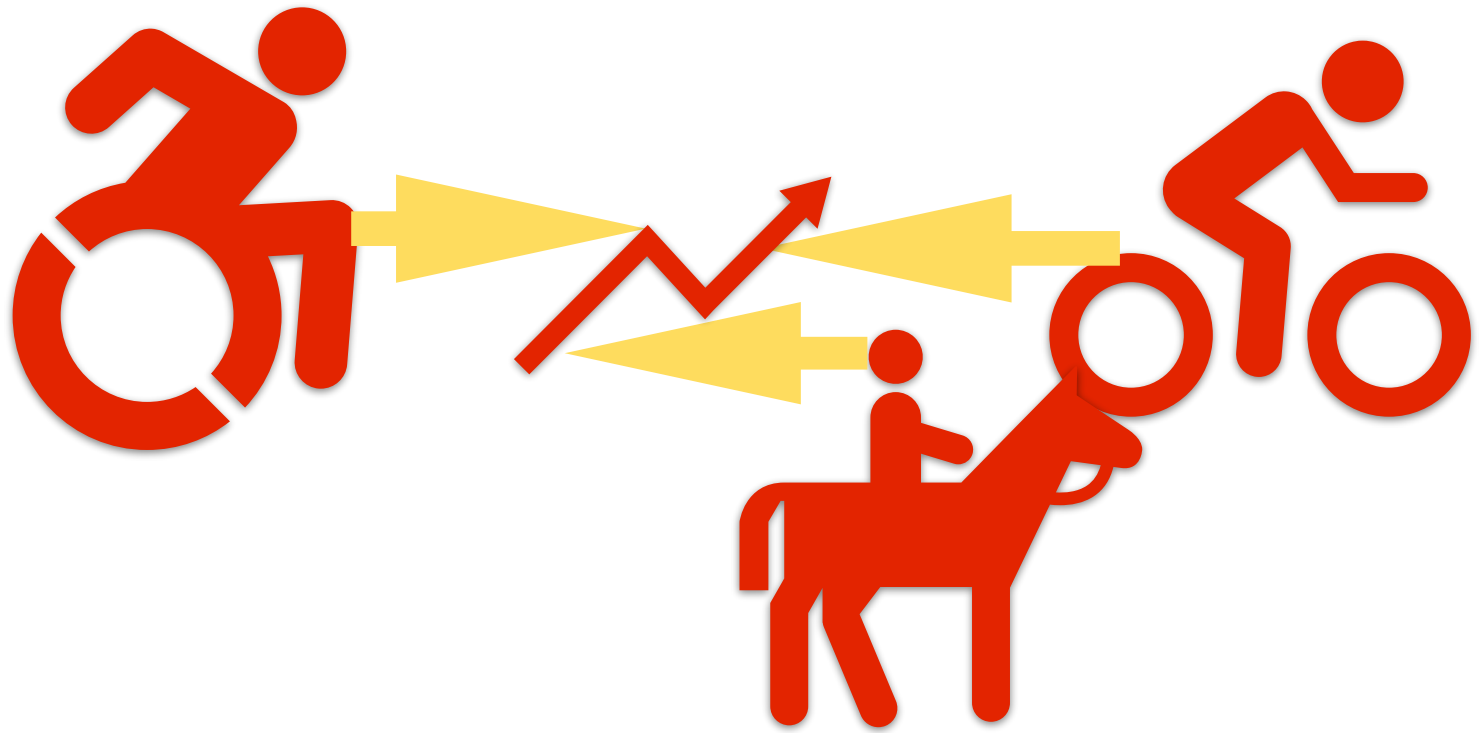


Figure 1.54 : 3 different stakeholders might want different levels of a value. The solution might lie in understanding more about each stakeholder's rationale. But the outlines of compromise lie in the fact that all stakeholders would welcome *some* increase in the value early.

As we all have experienced, people do not always know what value levels they will *really* need. They need more *experience* to know for sure, and potentially surprising future changes, to their environment, will force them to adjust their Wish levels.

So, we need to be flexible with regard to stakeholder real needs for the lifetime of the system. The beauty with numeric, and digital specification is that:

- It is easy to see exactly what was proposed and agree earlier
- It is easy to re-specify a change to the numbers, delivery dates, and Scale Parameter conditions
- It is easy, by comparing numbers and specified conditions, to see that something has changed
- Numbers do *not* lock you in to their digits, but they are the best way to understand value changes, and their cost consequences.

There is an implication here, that the Value Requirements are not simply a project plan, but are a **process** planning-and-change tool.

Probably useful as a **living map** of your system, for its entire lifetime. All the more reason to digitalize it, quality control it, and keep it updated.

The Value Specification becomes the visible marketplace for all the stakeholders to meet, and negotiate values and costs, for the system lifetime.

9.2 Negotiating Priorities for Values

As discussed above, the more we know about a stakeholder, and their reasons for Wishing for value levels, the better we can determine priorities: who gets how much first.

But it might happen that a stakeholder is not happy with a project decision. Here are some options for you in that case.

- ask or analyze the stakeholder for more information about why they need more value or value earlier, or in specific conditions (who, where, doing what). Document your findings in the Value spec Background statements, as detailed above.
- If the changed information deserves it, you can change priorities on that basis.
- but you might like to inform other competing stakeholders, and give them a fair chance to update their own arguments for value and fast service, in the specification.
- At the very least you need to inform them of why they got re-prioritized.
- Sometimes you need to point out to the stakeholder the means by which they can up their priority: special funding, powerful sponsorship, more active participation, better estimates of return on investment, etc.



Figure 1.55: negotiation with conflicting stakeholder interests is always a tricky business. But there are a number of systematic tools that can help you: and thorough specification of the values and the stakeholders, as suggested in this book, is a useful starting point.

<http://news.mit.edu/2018/natural-resource-negotiations-for-mutual-gains-bruno-verdini-0621>

9.3 Determining the higher-level value of value increments

Let us assume a stakeholder wants an increment of a value, any value, from 90% to 95% on any Scale.

We need to know two basic things about the increment, in order to understand if it is a reasonable idea, for implementation sometime.

- what is the lifetime economic-value range, preferably in money: but possibly in other dimensions like 'lives saved', 'months saved', and 'reputation improved'. ?
- Then in addition: what is it going to cost? The upper and lower range of lifetime costs (capital costs and operational costs).

This information will enable you to see if it is potentially worthwhile, 'values for costs', and how competitive it is with other options on your table.

There should be a clear advantage, even with lower range of values and higher range of costs.

If not, you need to refocus on more-promising options.

You can challenge proponent stakeholders to do another round of thinking about values and costs, and come back with improved numbers.

But you need to establish a clear policy of 'deciding by numbers', about the *profitability*, or the *efficiency*, of value improvement ideas.

Remind people that 100% perfect values will tend to cost infinity, and never be worth it.

So we are looking for the best 'deal', the best return on investment, both in private and government sectors.

This might seem obvious to you. But I am constantly confronted by managers and plans, that decide what to do, without any sort of incremental value analysis, and without any lifetime cost analysis at all.

They are of course usually 'spending someone else's money'.

But they are not really qualified to do so, without a little bit of common sense, or better education, or better top management, or whatever is lacking.

I refer back to the causes of failed Projects (Figure 1.8). Bad value requirements communication.

Something is very wrong here, and this lack of efficiency-estimation is part of it.

HOME IMPROVEMENTS IS IT WORTH IT?

IMPROVEMENT	AVERAGE COST	ADDED VALUE	PROFIT
NEW BATHROOM	£4250	£2,000	£-2,250
NEW KITCHEN	£7000	£4,000	£-3,000
NEW BOILER	£1995	£8,000	£6,005
CENTRAL HEATING	£4250	£8,000	£3,750
GARDEN UPGRADE	£1500	£2,000	£500
DOUBLE GLAZING	£8450	£8,000	£-450
EXTENSION	£48,000	£30,000	£-18,000
OPEN PLAN LIVING	£1750	£4,000	£2,250
SOLAR PANELS	£5000	£-4,000	£-9,000
ATTIC CONVERSION	£27,500	£20,000	£-7,500
EXTRA BEDROOM	£22,500	£6,000	£-16,500
CELLAR CONVERSION	£18,438	£20,000	£1,562
GARAGE	£18,620	£8,000	£-10,620
PAINT & DECORATE	£2500	£2,000	£-500
CONSERVATORY	£7500	£8,000	£500
NEW ROOF	£5622	£6,000	£378
NEW FLOORING	£2200	£0	£-2,200
SWIMMING POOL	£35,000	£2,000	£-33,000
TENNIS COURT	£36,000	£6,000	£-30,000
ENERGY SAVING	£475	£4,000	£3,525
SECURITY SYSTEM	£500	£4,000	£3,500
HOME CINEMA	£27,500	£2,000	£-25,500
CHARGING POINT	£800	£4,000	£3,200
OFF STREET PARKING	£2,000	£6,000	£4,000
WINE CELLAR	£46,075	£0	£-46,075

Figure 1.56 : Gee I think I'll change my mind about the Solar Panels, and the Tennis Court

We need to use this kind of thinking about stakeholder value increments. Will it pay off?

Or 'don't even think about it!'

<https://www.thesun.co.uk/money/7178628/home-renovations-affect-house-price/>

9.4 Contracting and Proposals

A well-specified Value Requirement is absolutely mandatory when soliciting proposals and estimates. No contractor can possibly estimate the cost of 'high Security'. They need more detail to begin to, even roughly (3X, 10X), estimate costs.

In fact if they make an estimate, and fail to ask you for more information about your exact value specifications, then you are in danger of severe failure.

- They are either **incompetent**, they do not understand the relation between value levels and costs, or
- They are *intentionally avoiding this*, to get your business, but plan to charge you much more when you reveal, after their first botched delivery, what your hidden quantified value requirements actually were.

So put that clearly-specified requirement in your proposal. In their face, up front. And make sure they understand that this is a legal condition for their estimates, for acceptable delivery, and for payment.



Figure 1.57 : Imagine if you provide something like this, in your request for a bid? Or maybe your top ten critical values ? All attributes (for example above, 'Evil People', 'Steal Money') need definition.

Imagine if you were to make a contract for this with 100% payment dependent of Goal level reached before the deadline.

Otherwise, 'proportional payment' applies.

For example, if 50% of increased value delivered in practice, then 50% of total payment for Security, is released to the supplier.

9.5 Handover to architecture and design engineering.

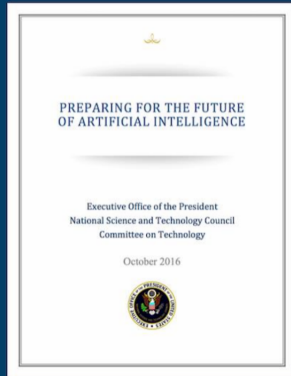
Requirements, as you know well, are the basic input, the problem formulation, for any process of design, or architecture, or design engineering.

Unfortunately there are several 'disciplines' that do not have 'clear quantified value requirements' as an input. These informal design disciplines I encounter in management planning (only time and money thinking), IT (we code, we found bugs, and have deadlines, not value). And in IT Enterprise Architecture.

More worryingly, I found a complete absence of quantification for the most important Artificial Intelligence International Standards, led by top professionals and academics (c, XAI). The good news is some of my friends there decided to listen!



Figure 1.58 : even IEEE manages to publish and accept vague fuzzy Values for deadly-serious subjects.



Proposal of Discussion toward Formulation of AI R&D Guideline

Referring OECD guidelines governing privacy, security, and so on, **it is necessary to begin discussions and considerations toward formulating an international guideline consisting of principles governing R&D of AI to be networked ("AI R&D Guideline")** as framework taken into account of in R&D of AI to be networked.

Proposed Principles in "AI R&D Guideline"

- 1. Principle of Transparency**
Ensuring the abilities to explain and verify the behaviors of the AI network system
- 2. Principle of User Assistance**
Giving consideration so that the AI network system can assist users and appropriately provide users with opportunities to make choices
- 3. Principle of Controllability**
Ensuring controllability of the AI network system by humans
- 4. Principle of Security**
Ensuring the robustness and dependability of the AI network system
- 5. Principle of Safety**
Giving consideration so that the AI network system will not cause danger to the lives/bodies of users and third parties
- 6. Principle of Privacy**
Giving consideration so that the AI network system will not infringe the privacy of users and third parties
- 7. Principle of Ethics**
Respecting human dignity and individuals' autonomy in conducting research and development of AI to be networked
- 8. Principle of Accountability**
Accomplishing accountability to related stakeholders such as users by researchers/developers of AI to be networked

Figure 1.59 : here extracted from an Obama White House paper on AI is a classical example of totally failing to define Values quantitatively.

I quantified them in anger (ref. c) using methods in this book.⁷

So there are too many planning cultures that are not mature enough to understand, when they need clear quantified Value requirements. You deal with them at your peril!

What can we do in practice?

You cannot hope to actually change immature planning cultures.

You can lead, for your own sake, and demand that critical values are quantified.

⁷ Another example is my 2019 'Sustainability Planning' book, based on bad goal statements for the UN Sustainability Goals. See gilb.com books. 'End Poverty' is Goal 1.

And demand that they are used in your planning and supply chain.
If not, you lose.

Just observe.

9.6 Handover to testing and measurement, with Value requirements.

Quantified and Scale-structured requirements should be a test planners dream. Finally they are getting a clearer idea of what to test.

But testing a variable Value will be new to some cultures of testers, like IT, where it is mainly 'test for presence or absence of functions'.

They might need to get some training and coaching.

In many cases a particular Value might be capable of having automated data collection for measurement, either from outside sources (official statistics) or by internal system measurements (user errors and response times for example). The more test automation the better.

You are invariably testing, or rather measuring, your system at a higher level of concern than technical design: the critical value levels.

There are fewer things to test (like 10 to 25 Values), and they effectively summarize all design component impacts.

The system you are improving values for, is usually a living system interacting with its environment.

In IT testing, if a function is there, it remains there mostly. So that kind of testing is relatively simple. Testing variable values is more challenging.



Figure 1.60 Value aspects.

In Value Testing there is a need for continuous life cycle monitoring of the critical Values. Over the course of years.

Generally, value levels are getting better all the time, but they can be disturbed by factors like load size, new stakeholders, new laws, and new management ambitions.

9.7 Presentation and approval for steering committees, based on Value Requirements.

No matter how detailed the basic Value requirement data is, there are a variety of ways of extracting the significant core ideas, for steering-committee decision-making, while holding open the door for 'drilling down' to get more supporting detail, meaning more *background*, when desired.

The *digital* requirement data can provide summarized graphics.

The requirement background data can provide a basis for recommendations as to priority.

To a degree today we can do some quality control, on the requirements specs, automatically, and more when we can afford to use effort to program more controls, like ValPlan.

We can easily print lists of all outstanding risks, issues, dependencies and assumptions, so they can be reviewed and dealt with. 'Automated risk ledgers' some would call it.

Steering committees can be supported when asking questions like:

- what problems are outstanding?

- What is the 'quality level'⁸ of our requirements documentation? Is it good enough to exit to the next processes yet? (Like architecture, estimation, test planning).
- Where are we overambitious in relation to our resources and needs?
- What can we expect of Value level improvements in the short term?
- Which stakeholders are giving us problems ?



Figure 1.61 : a steering committee needs data to see the whole picture, all requirements and their consequences, to make better decisions.

⁸ often measured in Defects per page, Defect = standards violation.

9.8 Quality Control: of value requirements.

Quality Control (QC) means checking a specification against our standards for best practices.

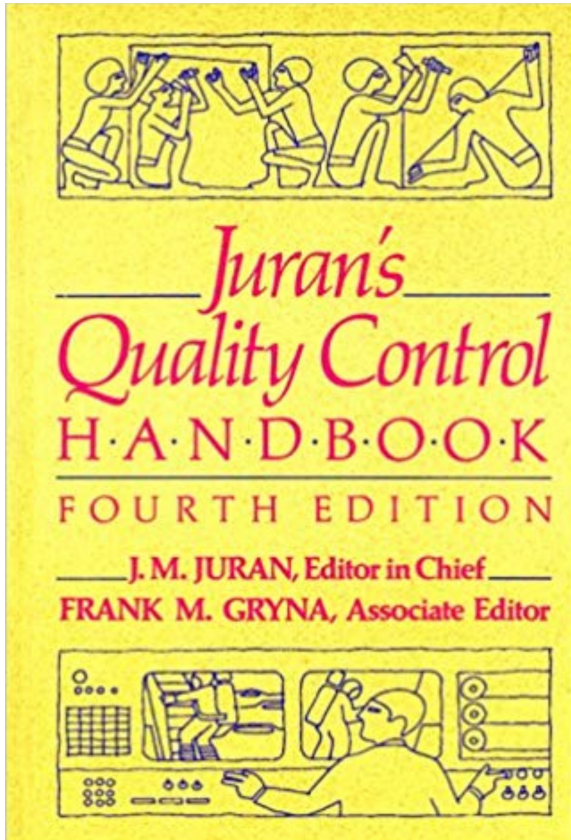


Figure 1.62 : the picture at the top of this book is from a real Egyptian wall. Thebes. It shows QC in relation to a standard. An ancient practice as you can see.

What Rules for Value requirement specification do you use? Many have been taught to you already, in this book (for example 'Define a Scale'. Many are available in other books (CE). Many are built in to the ValPlan app.

The most fundamental Rules for Value Requirements are:

1. They should be **clear** enough to test.
2. They should be **unambiguous** to the intended readership.
3. Value Requirements should be **quantified**.
4. Value Requirements should **not** contain **solutions** to achieving the value.

When checking requirements (doing QC) for compliance with your Rules, we call any violation of the Rules a 'Specification Defect'.

We now have a simple way of measuring Value Requirements quality levels: 'Defects Per Page', where a page is normalized to 300 words. In a planning tool we could use Average Defects per Planning Object, where a requirement is a Planning Object.

It turns out that defects in requirements will cause delays and bad specifications, and the problem gets 10x to 100x worse as time goes on (Capers Jones, namcook.com). So it pays off a lot to avoid them in the requirements phase. Remember the project failure table at the beginning of this book (Fig. 1.8). All the problems were requirements related.

By using Spec QC⁹ (VP, SI, CE books), a process we designed; and *measuring* the Defect Density of requirements, we have a chance to hold the problem under control. If we do not measure the defect density, the problem will be wildly out of control (100 to 300 defects per page).

Most organizations have no idea about this, and that is one reason why so many projects fail. The uncontrolled pollution of requirements, destroys or delays the entire project, when people encounter, and are forced to deal with the spec defects consequences downstream, at high costs.

⁹ Specification Quality Control. 'Agile Specification Quality Control, Shifting Emphasis from Cleanup to Sampling Defects', <http://www.gilb.com/DL264>, Paper in Text Experience.

9.9 Defect Level Measurement and Exit Control

How many defects is OK?

The answer varies depending on how high quality your products and services are.

A bank client of ours, had 82 Defects Per Page (DPP), and brought it down to 10 DPP within 6 months, thereafter aiming to get down to about 1 or 2 depending on what paid off. There can be a lot of money wasted, for a small defect, in a banking system.

The level you should be at (Defects Per Page) before you agree to exit downstream, next process, is an *economic* question. What pays off?

NASA and Intel have levels 10x better than 1 or 2 DPP at the bank. They have a standard of maximum 0.10 defects per 300 words. Mars landings, and Intel chips, have a lot of complexity and costs in common.

The main reason we *measure* defects, is not to correct them.

It is to *motivate* requirements writers to do their job right (almost no defects) in the first place.

This requires management leadership by insisting that these measures, and these exit levels, be taken seriously.

If management doesn't get serious, they are incompetent to manage, in my view. They are allowing chaos, and waste, to flourish.

Of course part of the problem is that people are not trained and informed about these things.

9.10 Management Reviews, in a Value Driven culture.

The Spec QC process assures us that the basics of requirement specification are 'in order'. Intelligibility, consistency, and purity of purpose (requirements, not '*design*' which is falsely called *requirements*', which is a common malpractice)

We could say that Spec QC is one type of review. And that it should be a basic requirement to pass this test, before moving on to a more sophisticated type of review, at a management level.

For example a Steering Committee review, or a review for presentation to a Steering Committee, or to management level.

Real World Review

A 'Real World' Review will be analyzing a fairly full set of specification, for example all of the critical top level Value requirements at once. And perhaps early stages of incremental delivery of those value levels, week by week.

We should be able to assume that a management review specification is not polluted with Rule violation defects, such as lack of clarity to the reviewers. We need Spec QC and Exit at low levels of Defects, before management tries to review.

Real World Reviewers have as their highest purposes:

- to buy in to, approve the requirements as officially sanctioned for further process, and progress
- To make sure all necessary elements are in the plan, so it has completeness. All critical stakeholders, all critical values, all critical constraints. All risks, assumptions, issues, dependencies documented.

9.11 Estimation of resources to deliver Value levels

There is a need for early estimation, and finally continuous re-estimation of resources needs (time, people, money).

There is a need for estimation of the value impacts, at a high level of planning, of the Values we plan to achieve.

For example what is the National Economic Effect of Saving Lives in Accidents ?

The clearer and more-detailed our requirements are, the better chance we have of making useful estimates.

But even then there are other factors, outside the Value requirements, that will determine the real resources needed.

We may still be wrong, about costs, by 3X or 10X, rather than 100X.

One initial factor, outside the direct control of the requirements, is the design process (architecture, engineering design).

This is a topic of many of my writings (CE, VP) and I am avoiding it here. But lets just say there is a thing called 'Design To Cost' and it means that a really smart designer/engineer/architect can reduce costs by 10X or more, just by coming up with a low cost solution!

So, you certainly cannot estimate real cost levels based on the Value **Requirements** alone. You can only begin to roughly get a picture of the costs.

Both design, and costing of specific **designs**, is necessary.¹⁰

Followed up by Dynamic Design to Cost, see Ch. 9.12 (Evo, IBM Cleanroom, see VP CE, and below and (reference d)) to gradually fine-tune real costs, and real value levels in the right directions.

¹⁰ See 'Value Design', 2019. Gilb.com

9.12 The 'Design to Value', and 'Design to Cost'.

If value levels are unambitious, and resources are plenty, then any fool can run a project without failing. But that is not why you are reading this book.

In everybody's real world, the ambition level for high values is pressuring us to go as high as possible, at the lowest use of resources.

The only method I know about, which really copes with this successfully is 'Dynamic Design To Cost'. (See ref. (d) IBM Cleanroom Method.)

That means you try to get off to a pretty good start, as outlined earlier in this book. But then you have to decompose your long term efforts into 50 or more Value Delivery Steps. I call this Evo.

At each step, we try to deliver a value improvement to the real world, your customers, the taxpayers, a potential market. Then we need to measure the value and cost results at each step. And compare them to our numeric estimates, our expectations.

If they are about the same, all OK, keep delivering more value steps. But, if there is a serious bad deviation in a value or cost, then we have to analyze and fix the problem immediately. Before we scale up, and compound the error.

This is one other place it is vital to have quantified and measurable Value Requirements to navigate by.

This process is very much like any navigation process, such as sailing the Seven Seas in 1492 or thereabouts. No GPS to help! Plenty of navigation readings and course corrections. But it worked.

'Get me to a nice place' is not a good enough Value requirement specification.



Figure 1.63: Egyptian navigator on sail ship. National Geographic.

Chapter 10. Presentation of Value Specifications

Value requirements need good presentation in different ways to different stakeholders.

One reason is that many professional cultures are not at all accustomed to Value Requirements, done so systematically, and measurably, as we are teaching here.

They are accustomed to the fuzzy requirements ('competitive agility', 'superior ease of maintenance').

You are probably going to be a pioneer in this area.

10.1 Presentation to Stakeholders

Stakeholders need to feel they are listened to, and understood correctly. They need to feel that we are really correctly analyzing the most critical sets of Scale Parameter attributes (who, where, when, what): the value sub-sets they need action on, in the short term.

One presentation method is to start with the Ambition Level, and its source or authority. Then show that the Scale is helping to clarify and structure that Ambition Level, quite directly. Listened to!

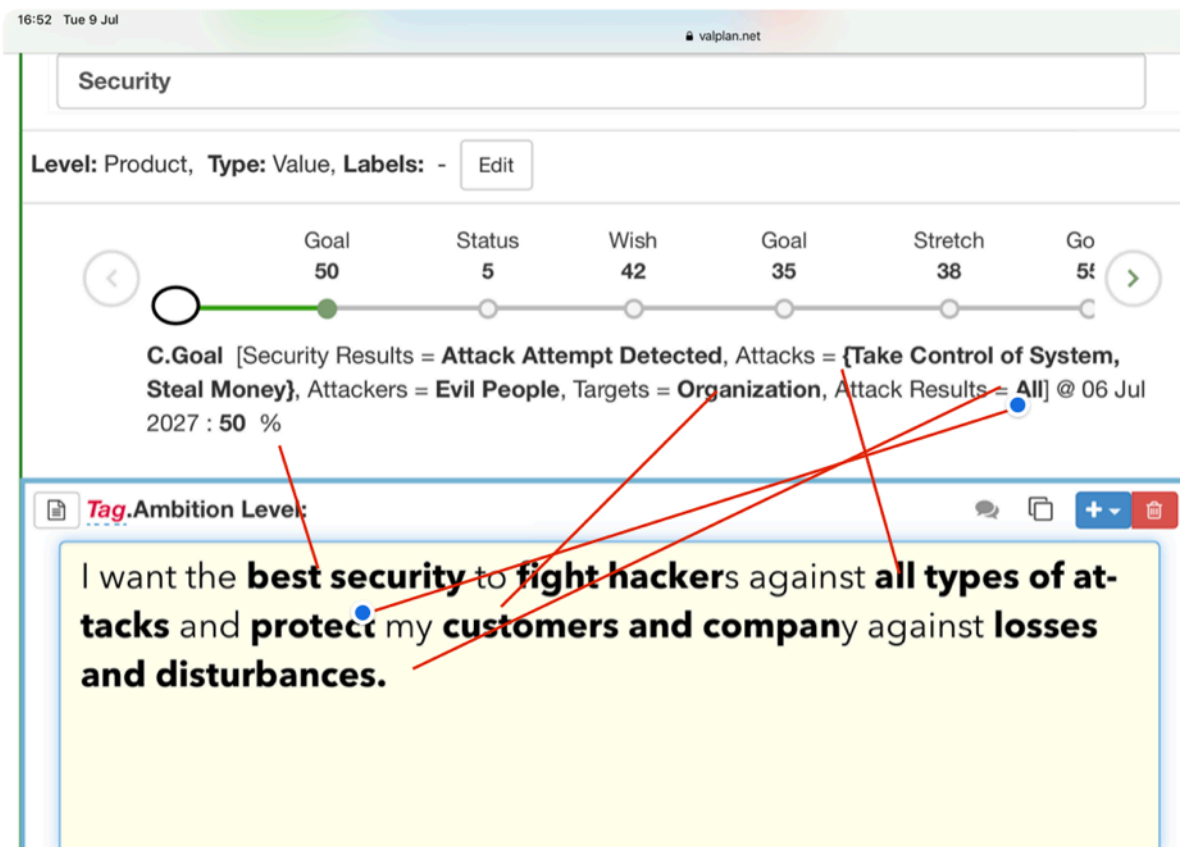


Figure 1.64 : the lines connect the original Ambition Level statement, with the specific Scale Parameters, and the Level

If you do involve some stakeholders directly, try some of the following:

- Ask if they would like to be the Owner of the specification.
- Ask if the required numeric level is good enough, by the specified date.
- Ask if the Scale Parameter attributes are a complete and useful set. Are we missing any critical Scale Parameter attributes?
- Are there specification terms which could do with a formal definition, because they could be misunderstood by some readers? what does 'Useful' really mean?

In other words, involve them to take ownership, at least by participation. Show genuine interest in their ideas: you are lucky to get

their ideas now!

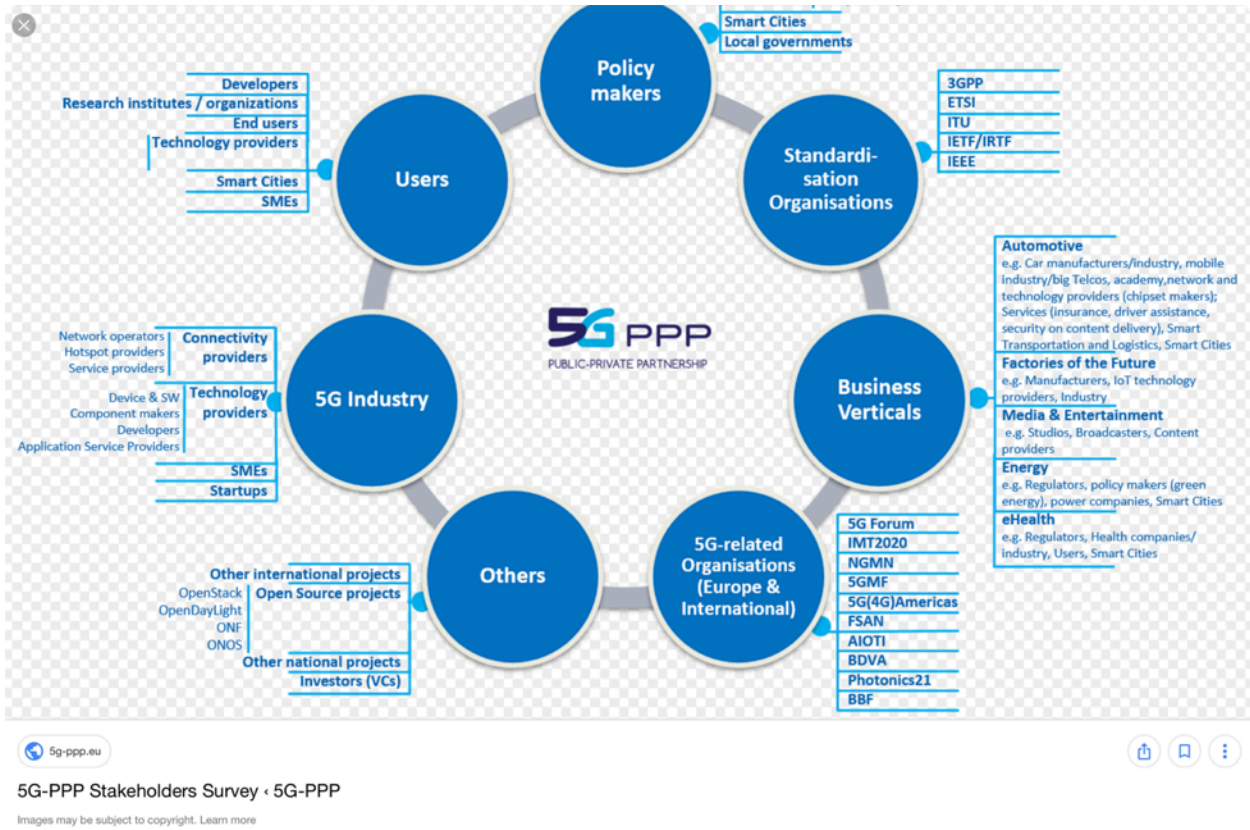


Figure 1.65 : notice how rich the stakeholder picture is just for '5G'? Oh yes, even 'users' are there!

10.2 Value Requirements: Presentation To Project Managers

The Value requirements should be presented to project managers as their **primary** focus. The main criteria upon which they will be **judged**, as project manager.

This might seem seem strange the first time they hear it.

Traditionally they have another focus.

In simple terms, get a set of tasks done and finish by the deadline.

But if you build the functionality (the 'building'), and your actual value levels are weak (security and safety are bad, and the elevators do not work), then you are in failure mode. The project is a failure.

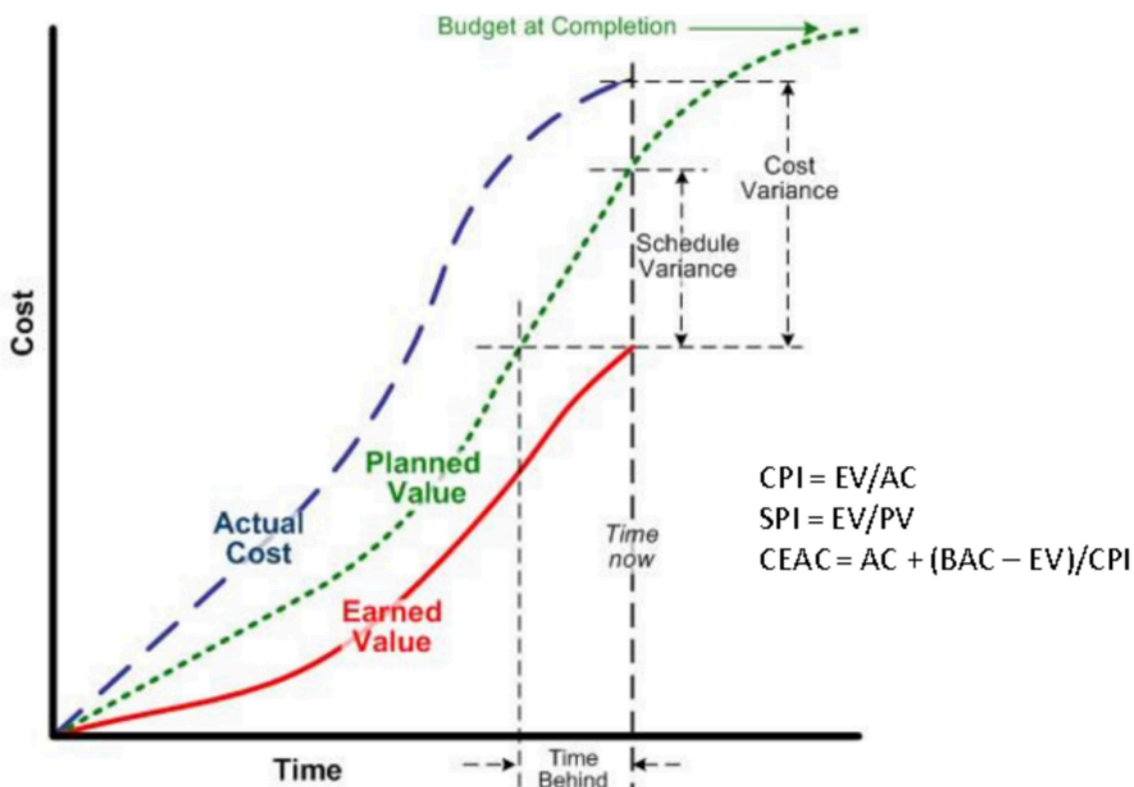
Building the functionality is very basic, you have to have it. But delivering the levels of **values** we planned (Tolerable, Goal) is the difference between *success* (All Goals met on time) and *failure* (any Value is under Tolerable level).

This implies that the project manager needs to have regular measurement, of the value delivery levels, and take action when they are not on track.

Somebody has to break the news to Old Guard project managers.

From now on they will be judged by their Value Delivery rate, **not** the 'speed of doing tasks'.

Their client or manager has to take a clear position on this, and ask if the project manager is up for the challenge. Manage Value.



Figure

Image source: www.pinterest.com

1.66
:
This
is a

chart from *conventional* Earned Value Management. (See Ch. 15.3)

I do not recommend it, because it is using a very *artificial* idea of value. EVM is a method that does not seem to understand how to define and quantify values, like 'Security', *directly* as we are teaching here. Direct measurement of 'Security' itself.

However we can use this graph example if we edit the words slightly.

Actual Cost: I would replace with **Resource Budgets Consumed.**

I would include all resource notions (people, time, money, etc.) and would especially budget and track future annual resource

consumption. Such as maintenance costs, and license costs. This gives a much more realistic picture of the economics of the real future system.

Planned Value: I would replace with **Planned Values**. Meaning a summary of the set of Values (Security, Usability, Market Image, and Employee Motivation, etc.). The sum of the stream of actual numeric individual value improvements. This gives more direct control over the critical values we want.

Earned Value: I would replace with **Delivered Values**, this would be the measured increase in various values, as a set. One method we use to give an overall single figure for this is to normalize all Values, on a Scale from 0% (Baseline, status, no increment) to 100% (Goal level reached on time).

When the Project Manager, who we might rather call the **Value Manager**, discovers negative differences in Values and Resources, they can call in the 'fixer'.

Like Robert Quinnan, Chief Architect in IBM Cleanroom method (see ref. d, Cleanroom). Who will perform 'Dynamic Design to Cost', and try to get the project back on track to delivering value within resource budgets. This actually works remarkably well, even on government projects, where EVM has a weak reputation, by comparison.

I just got an email from one of my professional students today (090719). Sad story. A Water Board project was determined to lie and falsify project progress reports. Rather than actually deliver a

good project. They asked my friend to do so, falsify progress. He refused and left them. A few years later, since the project never delivered the value that would have prevented it, the water got seriously mixed with 'bad other stuff'. And they were publicly fined £127 million in penalties. I wonder if even that fine would wake up such a corrupt management culture?

10.3 Value Presentation to Steering Committees

Steering Committees need the same Delivered Values / Resource Budget information, as the Project Manager needs to keep track of.

In addition they need to consider decisions to get back on track, when the Architecture's intervention is not working well enough.

That could be a matter of

- better architect
- More resources
- Trade offs of some values for others more critical

10.4 Value Presentation to Spec QC Teams

The Specification Quality Control team is usually the same level of people who are writing the requirement specification.

They do not need to take a position on the actual Value levels, or costs. Their job is to check that they are following reasonable best practice Rules, when writing specs. Clear, Consistent, Complete, etc. Intelligibility for others to read the specs. Good technical writing.

So they need to understand that their job is fearless self criticism, and to avoid exiting a spec, that will cause trouble, and embarrassment for them, downstream.

It is not actually so much a matter of fixing sloppy specs. It is more a matter of letting the measurements motivate requirement specifies to do incredibly much better work.

Intel Measures of Gilb Methods 2013



TABLE I: GEN 2 REQUIREMENTS DEFECT DENSITY

PRD Revision	# 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%

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), and time from project start to the second to the

II. PRODUCT BACKGROUND

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

Fig-
ure



1.67

: Source (A, Terzakis). An industrial example of Spec QC used on Planguage Value Requirements, at Intel.

In this example, a small Intel requirements team, using pretty much the basic ideas in this book (Scale etc.) measured their own deviation from their Requirement Specification Rules.

They started at a remarkably good 10 defects per page (600 words = page, I recall) and are refused exit to next process.

Finally on 6th attempt, they produce requirements of the extreme quality needed by Intel, and are allowed to exit. 50X better than

the initial submission. Overall project productivity improvement 233%.

10.5 Value Presentation to Architecture

Let us define the Systems Architect as whoever makes the technology selection, all the stuff needed to deliver the Values, within Resource Budgets. And other constraints.

Here is the brief to them: (See 'Value Design', 2019, gilb.com)

- find a set of designs (the 'architecture') which arguably will deliver the planned Value Levels, within the resources budgeted.
- document your ideas with estimates and evidence, on an Impact Estimation Table (CE, VP) also called Value Decision Table.

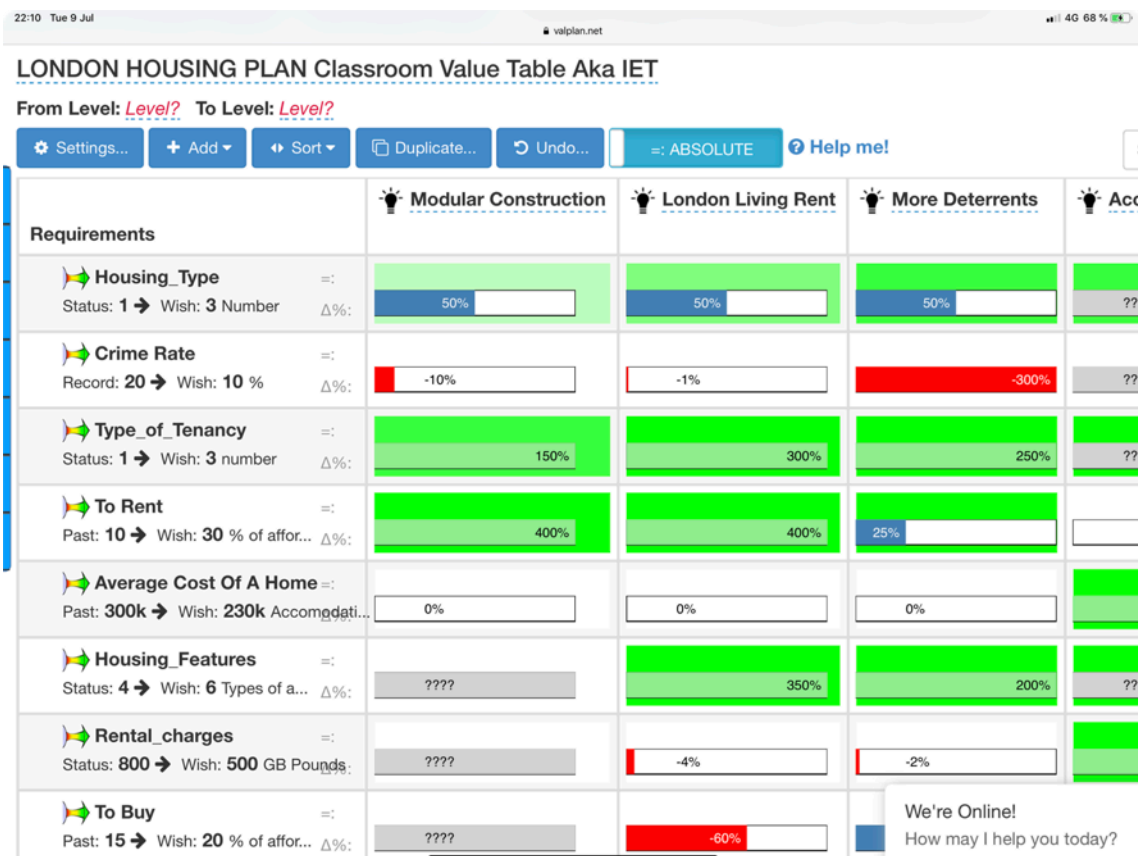


Figure 1.68 : An Impact Estimation Table: Architect keeping track of value. The colored bars in the middle are the estimated impact % of designs for value (lightbulbs) on the Value Requirements.

- Decompose the best options into independently implementable designs, and sequence them into small increments, which deliver real measurable value.
- Be available during the value delivery process, to adjust your designs, so they really do deliver value as needed, at resource levels we can afford. Like Quinnan IBM (reference d)
- Be responsible for maximizing values for resources, by design.

10.6 Value Presentation to Legal Team

Some of the Value specifications enter legal territory, when they are the basis for a bid, or a contract. As they should be.

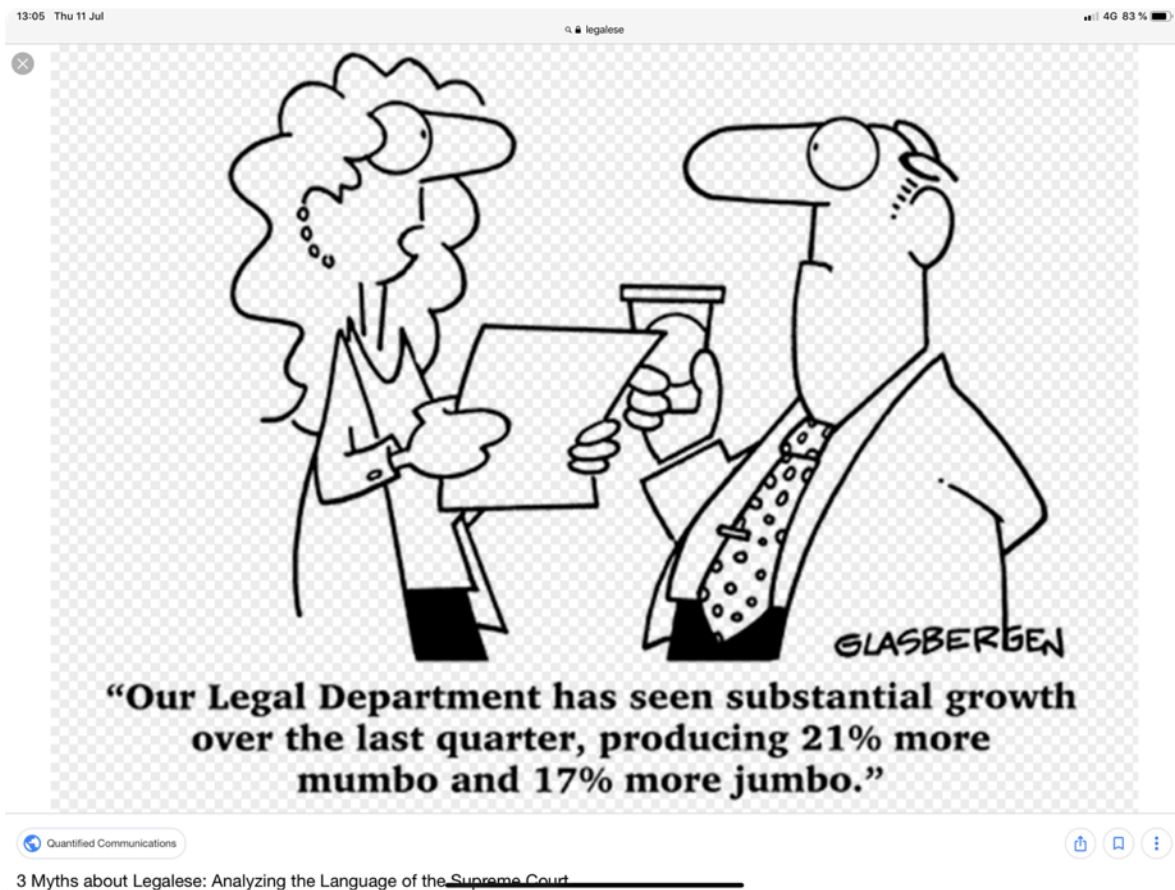


Figure 1.69: Legal Metrics.

Here is your message to your legal advisors.

- the Value numbers, and associated definitions (like Scale) are necessary clarification of what we are asking for
- The legal team should be clear that they expect, and insist on, such clarity for the critical Value aspects of the project.

- The legal team should point out (document it as a Risk in the Spec), whenever anything is specified, which might potentially cause legal trouble, in the event of a lawsuit.
- The legal teams interests should be indirectly represented by a set of Rules for specification, which direct the requirements writers to do, what is legally smart.
- Aside from clarity, I would expect a legal advisor to be interested in stakeholders, sources, and justifications.

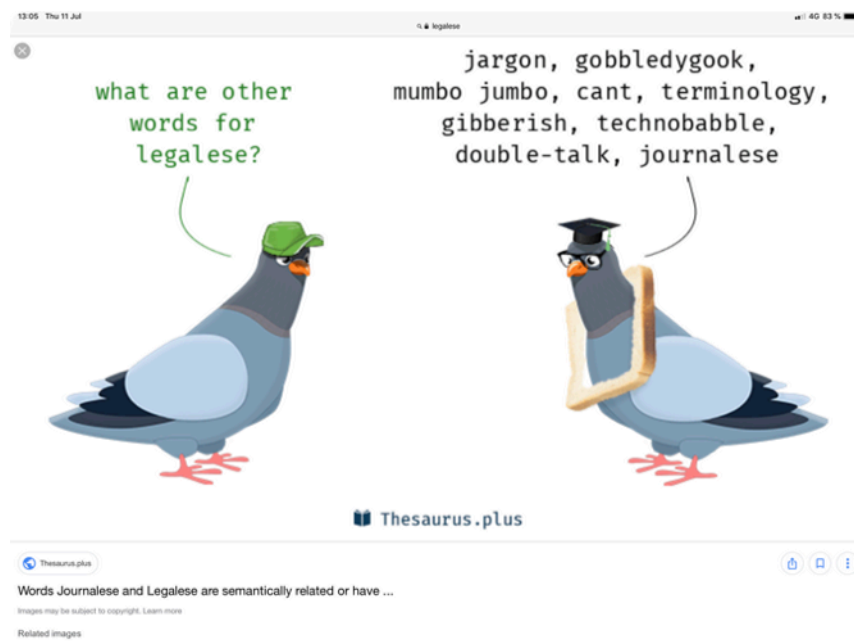


Figure 1.70. Legalese

10.7 Value Presentation to Sub-suppliers

Sub-suppliers need to be made aware of your priorities for getting the values delivered.

By connecting their payments to the progress of Value delivery, you can be sure of their attention and motivation.

50% of Value delivered should get 50% of agreed payment.

With a 'no cure, no pay' agreement, the sub-supplier, who is talented at delivering real measurable value, should be able to earn more, faster than any other method. If they cannot, earn more, maybe you need a better sub-supplier.

When evaluating competing sub-suppliers, a little contest to see who can really deliver measurable values, will sort out the winners from the losers.

In One case involving Accenture and a Nordic Government, Accenture earned so incredibly much, from a value-based contract that the government was forced to beg to renegotiate the terms.

No need to over-motivate!

Figure 1.71 No Cure.



Chapter 11. Levels of Value Specifications

A Value requirement is directly related to a particular defined level of a system. In general that means there are Values above it's level, to which it might contribute. There are Value levels below it's level, which might contribute to it's own value degree.

It is very useful to be conscious of these levels, and to document their relationships. It clarifies who is responsible for what.

This means taking *responsibility* for levels of value; for a specialist, a sub-supplier, a stakeholder, or a manager.

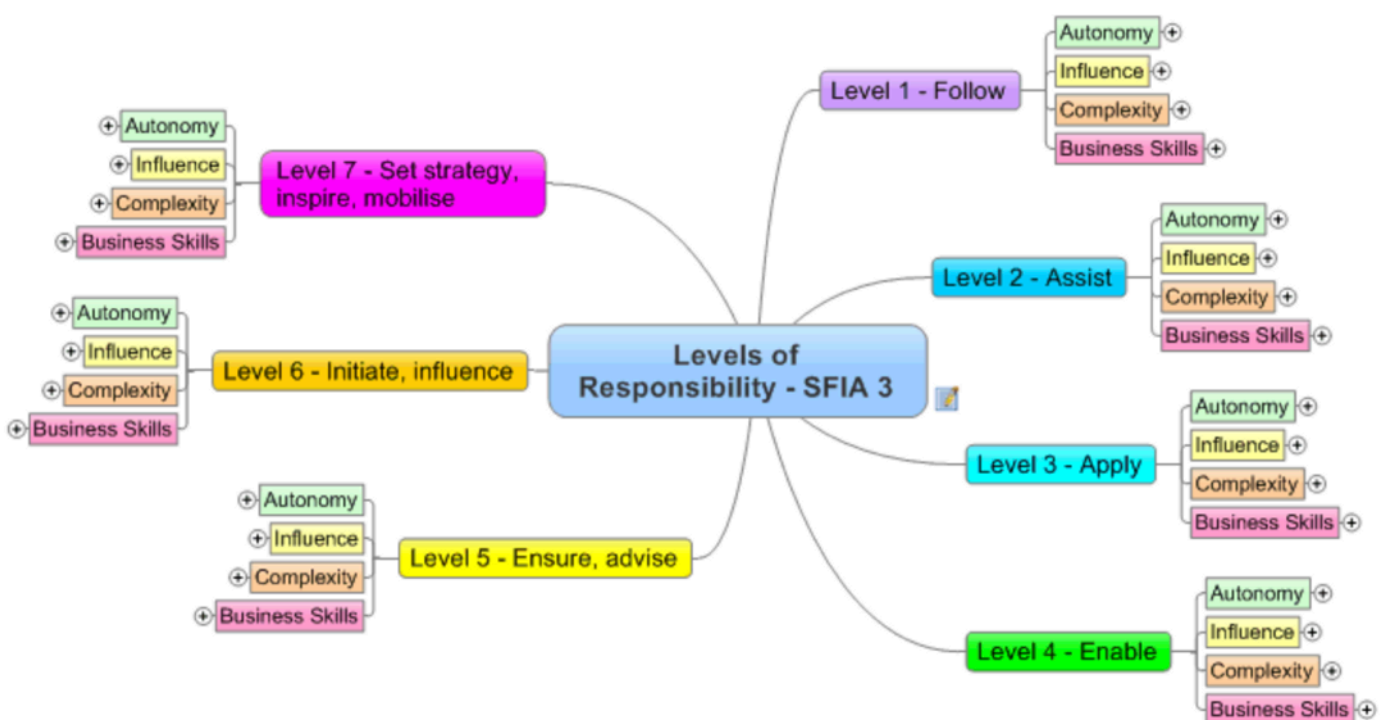


Figure 1.72 : some examples of levels of responsibility.

11.1 Vision Levels, Vision Engineering¹¹

Visions are the territory of top politicians and business , or society leaders. Think Musk, Kennedy, Martin Luther King.

We can certainly categorize *visions* as the top level of Value requirements. Everything which contributes to vision achievement is *related* to them.

You could, at one extreme, say that everything *below* that vision level is some kind of 'design' (or architecture, or solution) for achieving that vision.

At least that is true for the Visionary, they have value requirements, and everything that helps to achieve their vision, is clearly a *solution to meeting those requirements*.

However, just below the visionary is a hierarchy of vision-production stakeholders. Each one of them has a set of their own level of requirements, their own *value* requirements.

Each one of them has their own set of solutions to reach their requirements. And some of those solutions, are in fact perceived as 'requirements' by the next level below them, in the value chain.

Let me try to summarize and generalize.

¹¹ Value Planning: Top Level Vision Engineering" How to communicate critical visions and values quantitatively. Using The Planning Language. <http://concepts.gilb.com/dl926>. A 64 Page pdf book. Aimed at demonstrating with examples how top management can communicate their 'visions' far more clearly.

One stakeholder's requirements, are probably derived from the stakeholder level above them, in the value chain. These same next-level-down requirements are viewed as their solution, to fulfilling their level's vision (objectives, values) by the level above.

So, a planning specification, cannot really be categorized, once and for all, as either 'requirement' or 'design'. It is probably *both* at once. It simply depends on the point of reference. Whose requirement? Whose Solution?

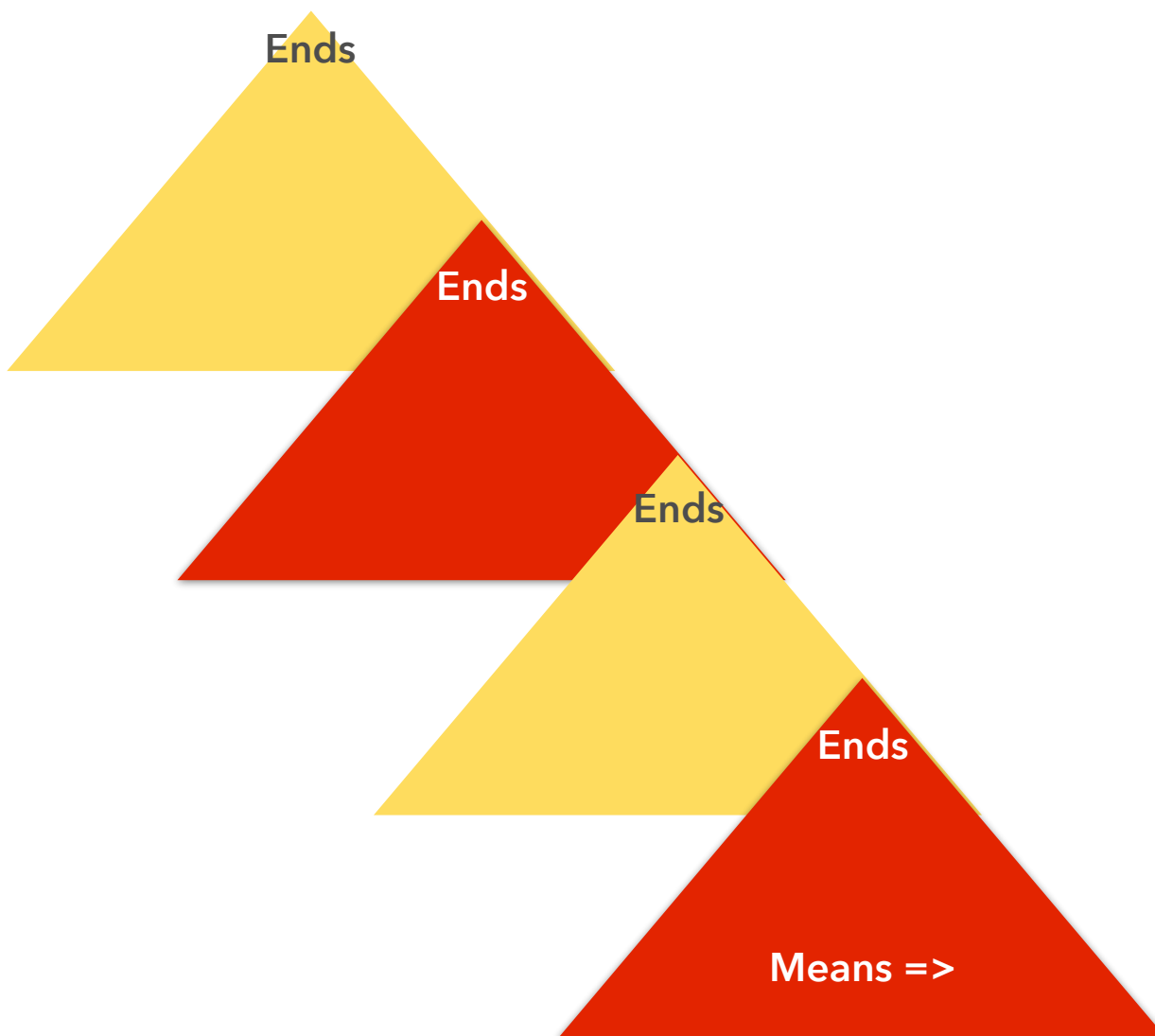
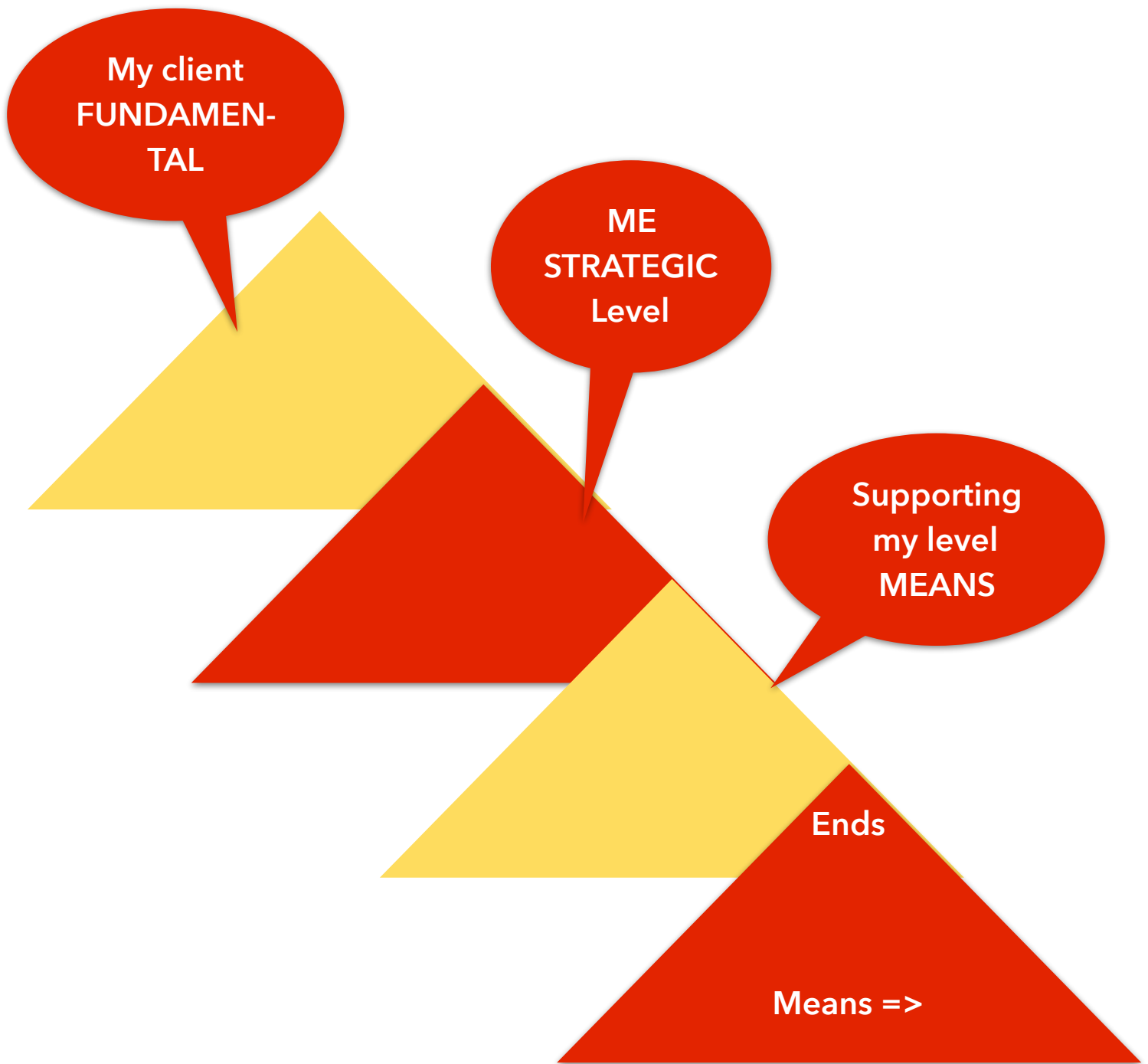


Figure 1.73 : 'One mans meat is another man's poison'. Each level of concern, needs to find *their* means to their ends. And typically that 'means' becomes the 'ends' (requirements) for the supporting level.



11.2 Fundamental Value Requirements

Figure 1.74 : Ralph Keeney's¹² Levels Hierarchy.

Upstream from you, are your Fundamental-level demands. They *become* your Strategic requirements, your *own* level of requirements.

There is the real danger that the Fundamental Means, *your* requirements, are badly thought out, badly formulated. What should you do? Blind obedience, or constructive change, with your help?

Downstream of your responsibility is the 'Means Objective' of someone. Your helper's task. If they reach it, that means your solution (*their* means objective) is 'in the box'. Your value requirement levels will be met.

12 Keeney, Ralph L. 1992. Value-focused Thinking: A Path to Creative Decision-making. Cambridge, MA and London: Harvard University Press. ISBN 0-674-93197-1.

<http://www.fuqua.duke.edu/faculty/alpha/keeney.htm/>. The diagram is my interpretation of his ideas of levels of concern.

Chapter 12. Resource Requirements Specification

12.1 The need to understand the incremental costs of value requirements before approving them.

You cannot accept or approve a Value requirement, just because a client *asks* for it.

Nor can you accept it, just because it *can* be achieved *technically*.

The Value requirement level must also be within *your economic* constraints.

And it must be *consistent* with all concurrent other Value commitments you have made, with the same total-project budgeted resources.

So, as difficult as it is, and as untraditional as it is, we need to estimate the 'resource use consequences' of any specified Value requirement level. What will it cost in time, money, people? What will be its recurrent maintenance costs?

Most people today skip this Value-level cost-estimation entirely, and irresponsibly. But this explains why we have budget and deadline overruns as 'normal'. Surprise! Shall we grow up yet?

As we shall see below, Value requirement estimation is quite difficult to do, accurately. But total lack of trying to understand such

costs, is not a responsible option. And, there are some reasonable solutions, for responsible planners, which we shall discuss.

If you have no idea what the costs of a Value Level are going to be, then maybe you need to make this clear to your client, up-front (that you are incompetent and irresponsible).

For example:

“ We have no idea what your requested Value level will cost. We are not even going to try to estimate it. With really bad luck it will far exceed all existing budgets, and threaten your job, and cause a public scandal. We can try to cover up, and blame someone else, as usual. But we thought we’d let you know. We do have some interesting less-risky options, like delivering value in smaller increments, and deciding to continue when the real costs emerge.”

Example 1R



I do like Confucius’ observation, that real wisdom is knowing what you do not know.

Figure 1.75 : By Wu Daozi, 685-758. Tang Dynasty

<https://i.pinimg.com/originals/77/95/19/7795198c-c338e4642d2ce7ac965458b2.jpg>

12.2 Some basic categories of Resource Requirements

Resources are needed to pay for the future Value improvements.

Limited resources need to be *budgeted*, and we need to make sure we do not plan to use more resource than we are allocated.

And that we do not use more resources than would 'pay off', be profitable, be a good and prioritized use of scarce, limited, resources. Those same resources are needed by other value requirements, other stakeholders, and other projects.

Here are some interesting resource categories:

Up Front Costs: *before* producing Value

- Capital Costs Money
- Calendar Time
- Work Hours

Lifecycle Costs

- Money for recurrent lifetime operational costs
 - System Maintenance, bug fix, port to new platforms
- Software Licenses

- Premises and Hardware rental
- Training Costs
- Internet Costs



Capital Cost £ Status: 0 → Budget: 3.584m £ Pounds to deliver the initial set of... No qualifiers 31st December 2020	500k ± 60k 500k £ 14 ± 2 % 14 21 % (x 0.5) 14%	???? ± 0 0 £ 0 ± 0 % 14 0 % (x 0.0) ????	90 ± 1 90 £ 0 ± 0 % 14 0 % (x 0.0) 0%	45 ± 30 45 £ 0 ± 0 % 14 0 % (x 0.0) 0%	256k ± 32k 256k £ 7 ± 1 % 21 11 % (x 0.4) 7%	EA%: 21 ± 3 % Show Sideb...
Calendar Cost, Days Status: 0 → Budget: 1k days days, the time taken to implement... [Defined Tasks = All...] 30th June 2022	1k ± 50 1k days 100 ± 5 % 100 150 % (x 0.5) 100%	???? ± 0 0 days 0 ± 0 % 100 0 % (x 0.0) ????	60 ± 5 60 days 6 ± 1 % 106 12 % (x 0.0) 6%	400 ± 200 400 days 40 ± 20 % 146 80 % (x 0.0) 40%	60 ± 20 60 days 6 ± 2 % 152 12 % (x 0.0) 6%	EA%: 152 ± 28 %
Full Time Equivalents 4.5 Status: 140k → Budget: 200k Pounds £ No qualifiers 2018	5k ± 500 145k Pounds 8 ± 1 % 8 12 % (x 0.5) 8%	15k ± 2k 155k Pounds 25 ± 3 % 33 38 % (x 0.5) 25%	30k ± 10k 170k Pounds 50 ± 17 % 83 100 % (x 0.0) 50%	25k ± 50k 165k Pounds 42 ± 83 % 125 84 % (x 0.0) 42%	50k ± 1k 190k Pounds 83 ± 2 % 208 168 % (x 0.0) 83%	EA%: 208 ± 106 %
Maintenance Costs £k Status: 0 → Budget: 1m ann... £ cost per Year No qualifiers 2022	0 ± 0 0 annual... 0 ± 0 % 0 0 % (x 0.5) 0%	200 ± 7 200 annual... 0 ± 0 % 0 0 % (x 0.2) 0%	13 ± 5 13 annual... 0 ± 0 % 0 0 % (x 0.0) 0%	220 ± 10 220 annual... 0 ± 0 % 0 0 % (x 0.0) 0%	52.8k ± 10k 52.8k annual... 5 ± 1 % 5 8 % (x 0.4) 5%	EA%: 5 ± 1 %
Sum Of Development Resources: Worst Case: Credibility - adjusted: Worst Case Cred. - adjusted:	Σ%: 122 ± 8 % 122 Σ±%: 130 % Σ7%: 183 % Σ±7%: 65 %	Σ%: 25 ± 3 % 147 Σ±%: 28 % Σ7%: 38 % Σ±7%: 14 %	Σ%: 56 ± 18 % 203 Σ±%: 74 % Σ7%: 112 % Σ±7%: 0 %	Σ%: 82 ± 103 % 285 Σ±%: 185 % Σ7%: 164 % Σ±7%: 0 %	Σ%: 101 ± 6 % 386 Σ±%: 107 % Σ7%: 197 % Σ±7%: 6 %	

Figure 1.76 : Example of 4 different resources being estimated, and then computing an overall 'Sum of Development Resources'. Each of the 5 columns is a different design, being considered. Source: Technoscopes book, 47.

There are more resource types than listed above. The point is that the higher our ambition level for value, the more likely we are to drive up these costs for the system. So we have to keep track of them. The purse is not bottomless.

The correct place to keep track of resources is during the design phase, where the 'costs of designs' needs to be estimated (see above Figure), and designs with 'low and controllable' costs need to be prioritized.

The design you choose contains, like it or not, a set of immediate and lifecycle costs. We cannot remain totally unaware of cost consequences of our designs and strategies; until it is too late and we are irreversibly committed to expenditure, by contract or other decisions.

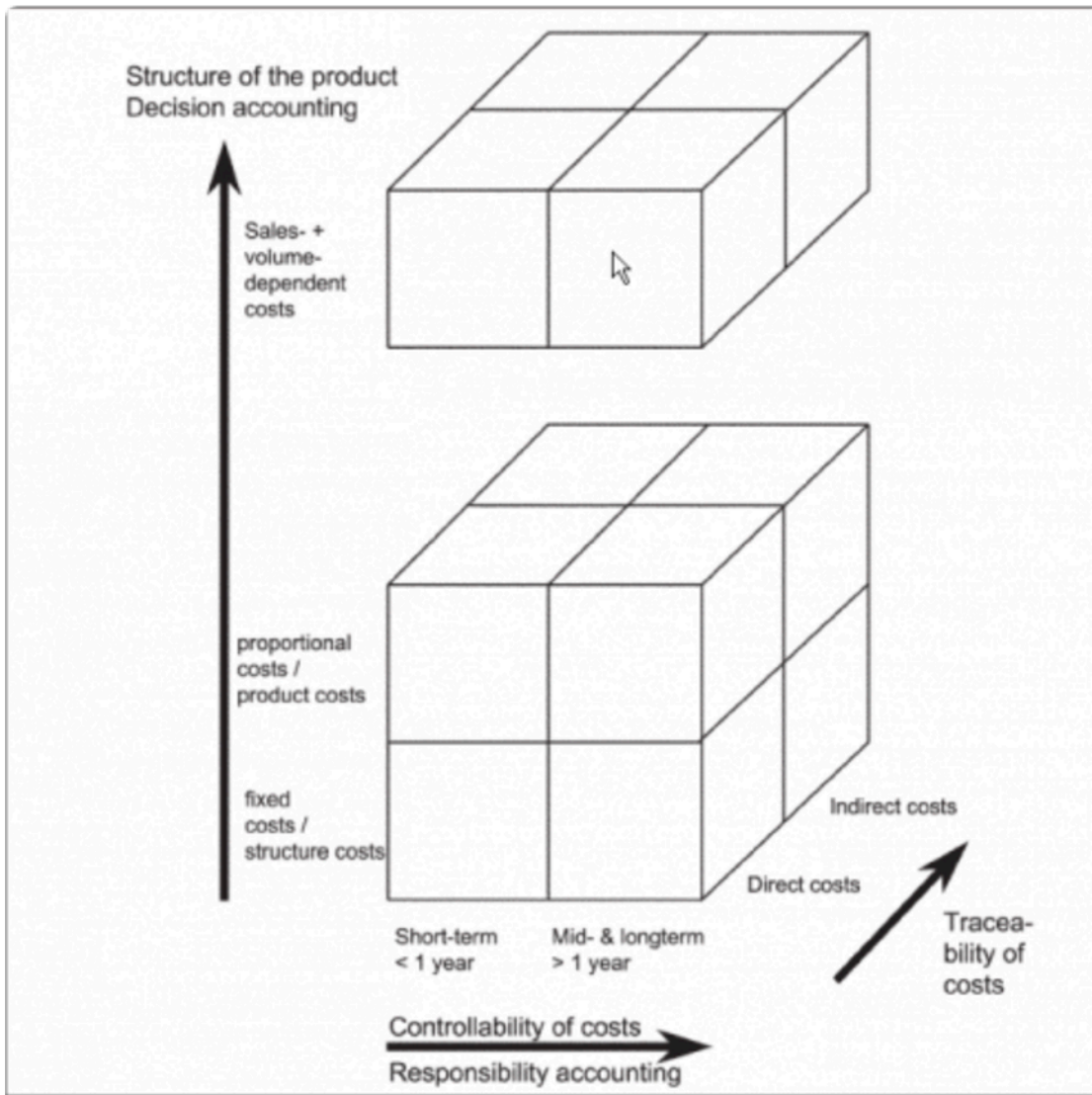


Figure 1.77 : there are many dimensions to understanding cost impacts of your Value requirements.

12.3 The difficulty of estimation of costs up front (ref. F)

If the availability level of a system goes up from 99.90% to 99.98%, a mere 00.08% increase: how might that impact the building costs? If I added, that it was for a 'large national telephone switching system' (AT&T Electronic Switching System) ? The answer was 8 years project duration, with between 2 to 3,000 people.

If you have no experience at that level, with that type of system, it is quite difficult to make accurate estimates. But you have been warned!

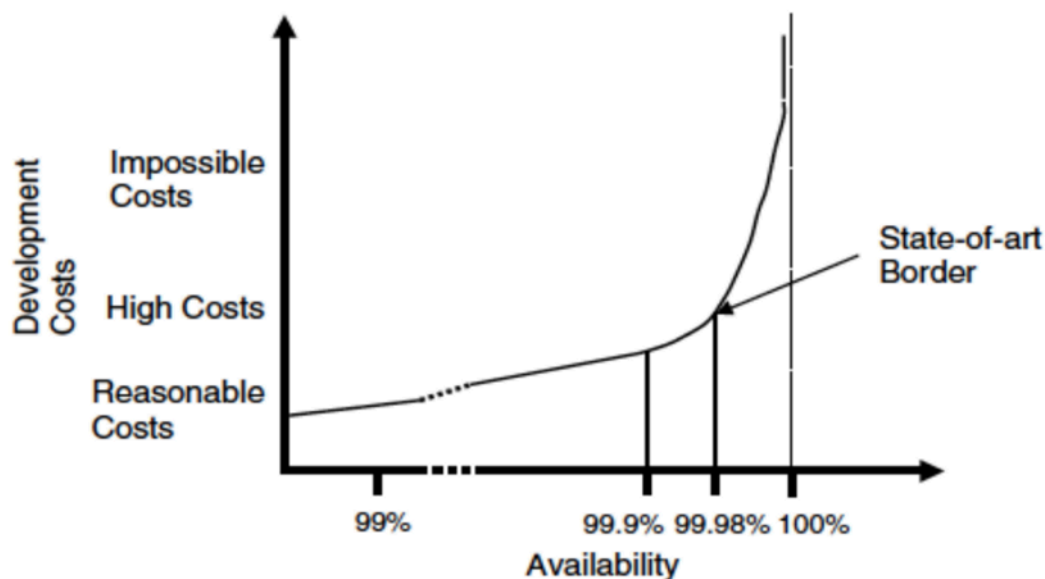


Fig-

ure 1.78 : The AT&T Case. Sources: Communications of ACM, and Former employees. PS 99.998% is State of the Art nowadays the people involved tell me.

The interesting thing is that as we approach 100% availability, the costs approach infinity, as all good engineers know.

In Barry Boehm's COCOMO cost estimation method, there were about 60 factors which were shown to be software project 'cost drivers'. This one, Availability, was only 1 of the 60. How many of the 60 do you have historical data, for the cost levels of them?

12.4 The possibility of getting control over real costs by design to cost

So for large, complex, hi-tech projects, the simple answer is, nobody has a good answer for estimating it all in advance. If they think they do, they have a secret they should share with the rest of us.

More likely they are ignorant of the estimation difficulty, especially when Value level requirements, in competitive environments, in arms races, and beyond the state-of-the-art, are concerned. The safe estimate is 'near infinity'. Or very much more than you can imagine.¹³

There is however one alternative approach, which is more likely to be good management, and safe politics for you. It is the engineering paradigm '**Design To Cost**'.

That means we do **not** specify a system, and *afterwards* ask for a cost estimate ('near infinity', is not an acceptable estimate).

¹³ Volume 13 Issue 2 of SQP journal - the March 2011 version.

<http://www.gilb.com/DL460> 'Estimation: A Paradigm Shift Toward Dynamic Design-to Cost and Radical Management'

We need to get a very smart architect/engineer/designer who can figure out how to bring the critical cost elements to 'within your acceptable budgets': by choosing cost-effective designs.

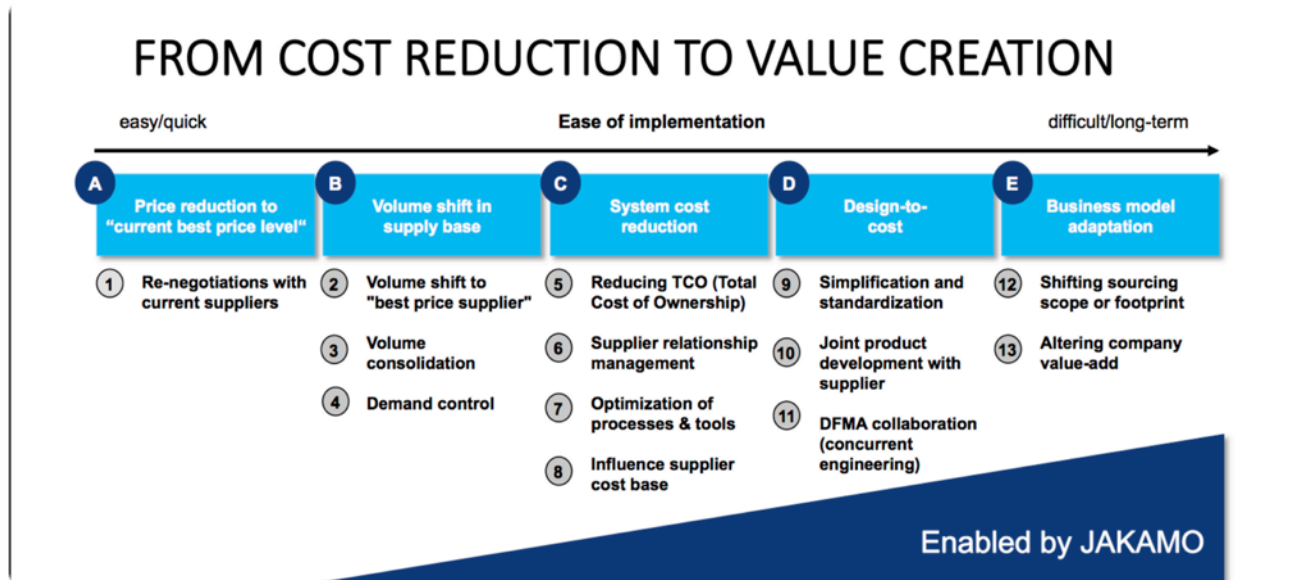


Figure 1.79 : 'Design To Cost', as one possible process in cost management. All 5 processes are a form of Design to Cost. Source <https://jakamo.net/cutting-costs-driving-value-creation/>

This can be as simple as 'contracting outside' at a *really* fixed price.

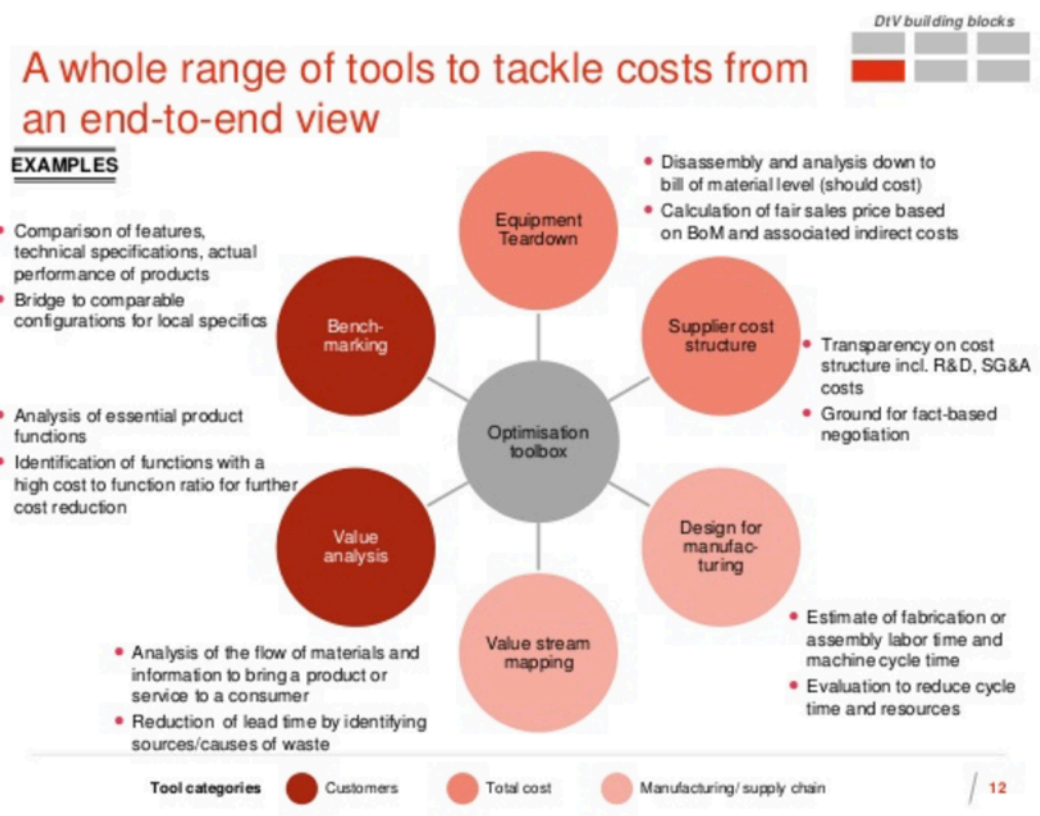
But it can also be done by reusing existing systems, rather than re-inventing the wheel; or done by using radically cheaper new technologies, and there are quite a lot of *other* cost-effective options.

The purpose of estimation, asking for a cost estimate, is to try to keep within 'economically acceptable' cost boundaries.

Design-to-cost will do *that* same job, better.

It is surprising:

- how many planning processes do not consider costs, or *all* types of costs, at all (see Chapter 16 for examples)
- How many cost processes do not think about design-to-cost at all. They do not assume that smart designers have great control over various costs, and assign the designers to do the cost reduction task.



Design-to-Value Approach for Telcos - JP Romieu

Figure 1.80 : More practical examples of tools to manage costs. Including 'Design for Manufacturing'. Source JP Romieu

12.5 The possibility of getting control of some costs by smart 'Dynamic Architecture' and design

The best single cost management method I know about is the Cleanroom Method at IBM Federal Systems Division (ref. d). Which is, in terms of iterative learning, the same as my own 'Evo' method (ref. Books CE, VP).

This was applied to military and space Software projects.

But I think you will find that it is the *same* as Elon Musk uses producing Teslas, where he releases to production, about 20 changes a week, half hardware.¹⁴ Half uploadable to my car at home.

The result at IBM was radically improved costs management from previous methods. Instead of being always late, and over fixed-price budgets, IBM was *always* on time and under budget, for years on end, for named real space and military projects.

Basically IBM were using 'design to cost', except it was not 'once off at the beginning'. I call that 'Dynamic Design to Cost'. It was *regularly* at every increment, for 50 or more increments. In Tesla's case, I assume, weekly.¹⁵

¹⁴ as quoted from him in a biography. Ashlee Vance, Elon Musk: Tesla, SpaceX, and the Quest for a Fantastic Future, Hardcover 2015, Paperback 2017.

¹⁵ Musk uses weekly increments at the factory production level, but software release to car owners is about monthly. For new hardware, I had to buy a new Model S. But it had about 50x20 hardware increments, like 4WD, and autopilot, and many smaller tweaks, like to the music interface.

At every, real system, increment, costs and values are measured, as well as possible, in the short term. If there is negative deviation from plans and expectations, the architect (If necessary, Musk and team, at IBM the Architect Robert Quinnan) dives in and fixes things immediately, whatever the root cause of the deviation.

As a 2nd time Tesla S owner, I can tell you - whatever they do, it works, on the incremental quality of the car. Half software, half hardware. Detroit never dared do things like that.

Tesla calculator estimates the exact monthly costs of owning a Model 3



by MIX — Sep 6, 2017 in APPS



Figure 1.81 : From my detailed years-long study of Musk and Tesla, it is clear that there is both an initial Design-to-Cost for the main architecture, followed by 'weekly-in-production 20 changes'. With value improvement tuning, and learning rapidly what works.

The Tesla is only 'half software'. So I get my car updates over WiFi. And I must have gotten about 1,000 hardware upgrades when I traded in my old Tesla for a New One.

Value Requirements at Tesla?

- Safety
- Build Quality
- Performance
- Owner Maintenance Costs
- And several more! (Comfort, ease of driving, emissions, styling, scaling up production, automatic driving)

12.6 The possibility of getting control of the value-to-cost ratio by *decomposition*; and then by prioritization of high-efficiency designs.

By decomposing values, functions, and design components into a more-detailed set-of-things, we can select a small fraction of the total product or system, to implement incrementally.

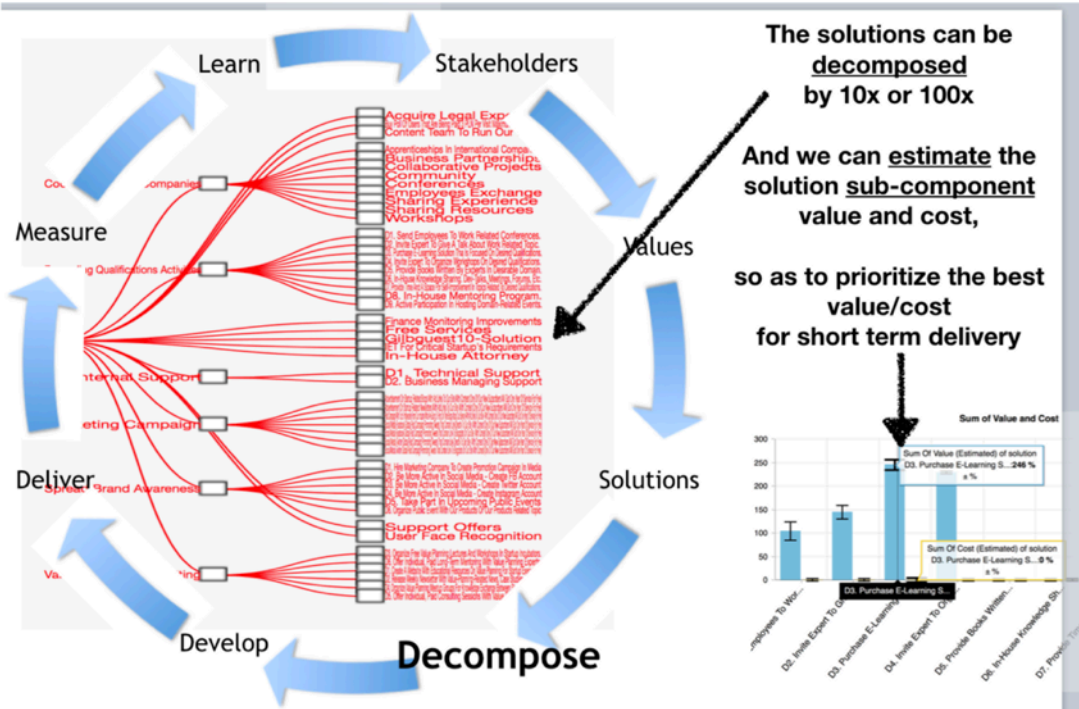
This will give us real 'value delivery' to *some* stakeholders, for *some* value improvements. But, at least as important, it will give us *measurable feedback* about a large number of factors, so we can better see what we need to improve, before we scale up, when implementing the rest of our agenda.

Value Decomposition Methods	Details		
Design Decomposition	Independently Implementable	Measurable Value Delivery ability	Urgent subset
Value Decomposition	Multiple Values	Scale Dimensions	Scale Dimension Sets
Deadline Decomposition	Date Due Start Date	Event Synchronization like 'Law in Force', 'Official Holiday'	Condition Synchronization like 'Interest Rates over 5%', 'Heat Wave'
Priority Type Decomposition	Wish (Desired Level)	Tolerable (Minimum Level)	Goal (Committed Level)

23. DIFFERENT TYPES OF DECOMPOSITION INTO COMPONENTS OR SUB-SETS

Source: Value Planning book, Chapter 5 Decomposition

Figure 1.82 : various decomposition methods, so that we can learn reality early, and credibly, before scaling-up size or volume.



24. HERE IS A REAL EXAMPLE OF MY CLIENT'S DECOMPOSITION OF DESIGN SOLUTIONS INTO INDEPENDENTLY IMPLEMENTABLE COMPONENTS. ON THE RIGHT LOWER CORNER IS AN ESTIMATE OF THE EFFECTIVENESS OF EACH SUB-COMPONENT, SO THAT WE CAN CHOOSE TO GO IMPLEMENT THE MOST EFFECTIVE ONE EARLY

Source: ADVANCED AGILE SOFTWARE ENGINEERING Turkey, April 2018, Polish Case

Figure 1.83 : real planning example of decomposition and values/costs estimation, of the decomposed design ideas.

12.7 The possibility of getting control over costs, by negotiated reduction of initial Value level, and initial date ambitions

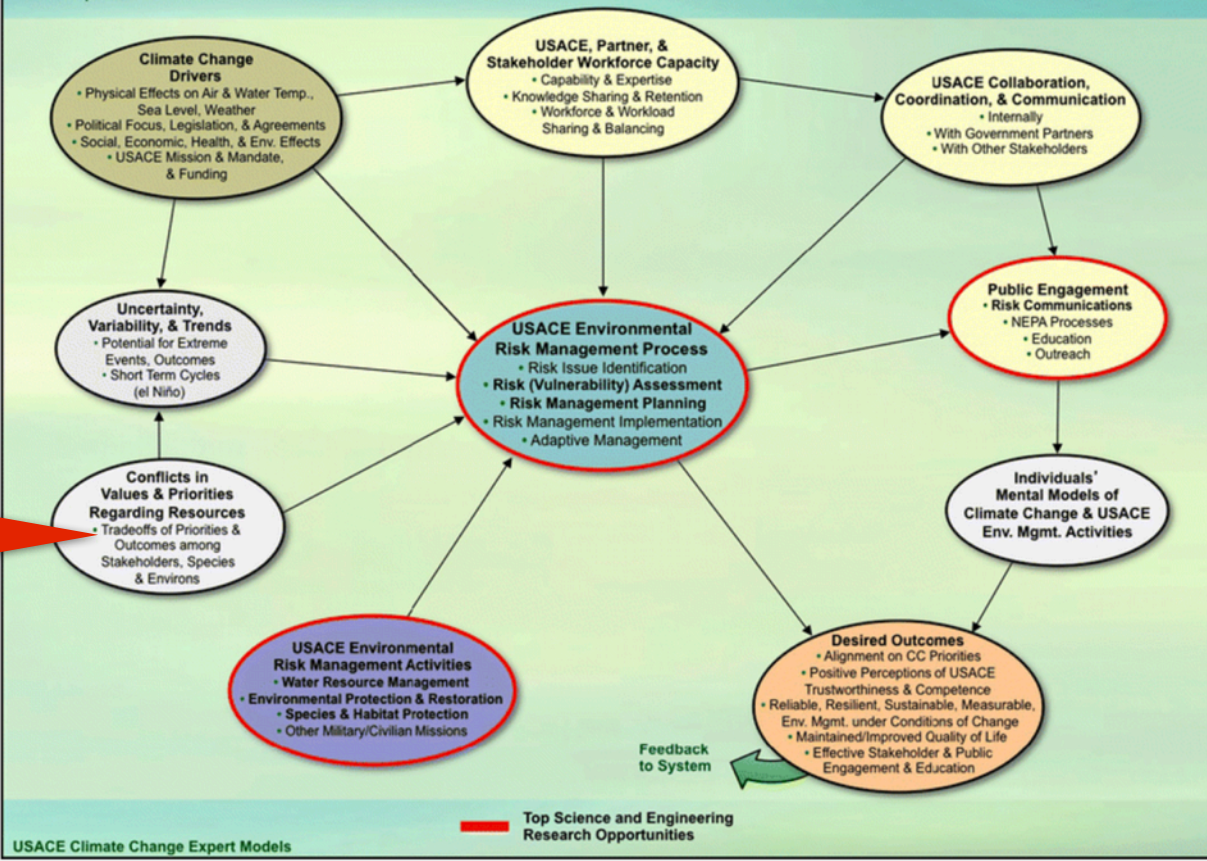
Sometimes the best way to get a cost reduction is the reduce the Value level *itself*.

This does not have to be a permanent arrangement. The deal can be that you will be able to raise the level later:

- when additional resources are available
- When a real customer is willing to pay for it
- When new market entry demands it
- When cheaper technology makes it more cost-effective.
- And it will never, you can promise, be lower than the improved level stated in a 'Tolerable' level requirement.

This is called an engineering trade off.

Influences of Climate Change on USACE Environmental Risk Management
Base Expert Model



Springer

Climate change risk management: a Mental Modeling application

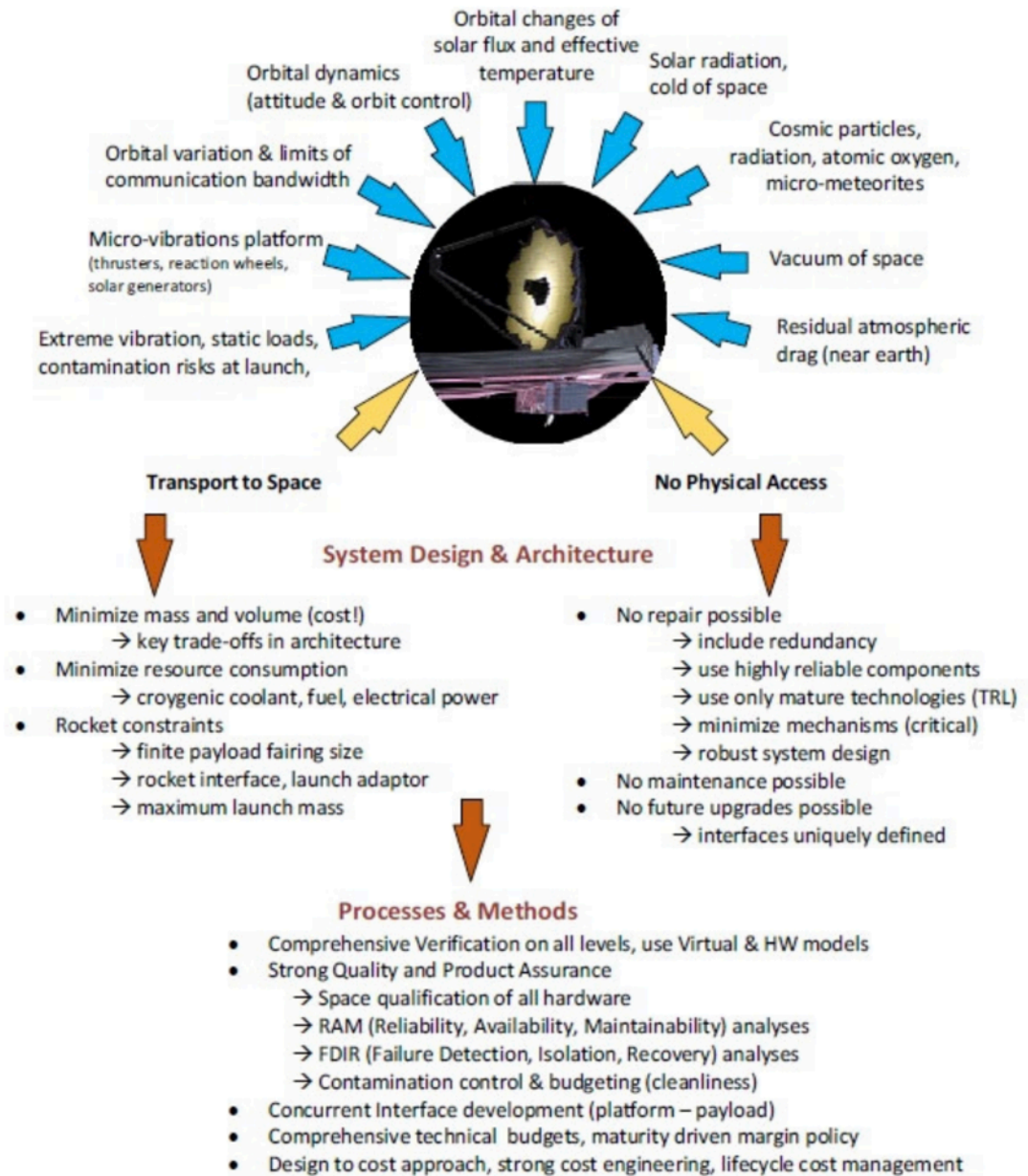
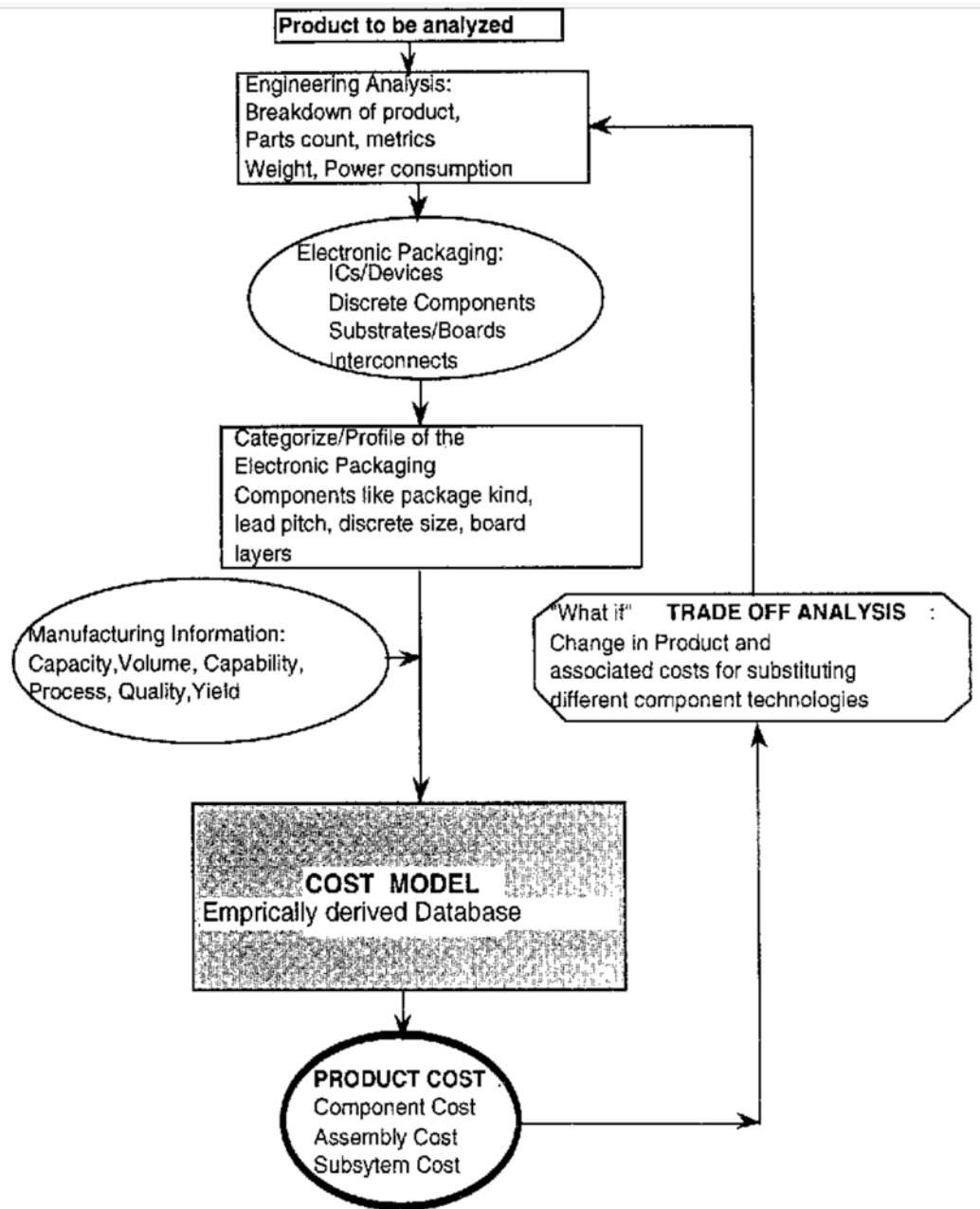


Figure 1.84 : a 'space' environment view of multiple Values, driving multiple design options with multiple constraints, to find satisfactory balance.



Trade-off analysis on cost and manufacturing technology of an _____

Figure 1.85 : With very many values, and very many international stakeholders, the trade-off process is in play, in a risk management context.

Figure 1.86 : here is a simple flow chart showing an iterative design and trade-off process, until satisfactory costs are reached.

12.7 The possibility of getting control over costs by Subcontracting

In the diagrams above, the possibility of getting control over costs, by evaluating cost-competitive suppliers is hinted at.

This can be done by direct bidding, contracting, competition, and asking *them* to do design-to-cost.¹⁶

But, perhaps the most important tool to do this with subcontracting, and really save costs is, to *use most of the advice about numeric value requirements in this book. To protect your project against cost reduction by means of undesirable value reduction.*

Anybody can cut costs, if they are not constrained by real measures of values and qualities expected.

In addition, this Value Delivery needs to be proven incrementally, rather than 'all at once at the bitter end'. Avoid big surprises.

Sub-contractor cost control:

- all quantified and specified Value Requirements are in the contract
- Payment is released when Values are achieved

¹⁶ Agile Contracting for Results The Next Level of Agile Project Management: Gilb's Methodology Column Agilerecord August 2013. concepts.gilb.com/dl581

- Work is done incrementally, so there is early and continuous proof of capability to deliver value for expected costs.
- Bonus for more-than-expected cost reductions.

Chapter 13. Change Control of Value Specifications.

Just because we quantify and structure Value requirements, does not mean they are chiseled in stone.

Change is inevitable and necessary. And quantified structured Value requirements are *ready* for systematic controlled change.

Here are some methods of managing change to Value Requirements.

13.1 The concept of a specification Owner.

A 'Specification Owner', or more precisely a Specification Object Owner is a person or group given sole power to change a specification object, such as a single Value Requirement.

The screenshot displays a web-based interface for managing user experience requirements. At the top, there is a title 'User Experience Aka Usability' and a search bar. Below this, the current level is set to 'Product' and the type is 'Value'. A progress bar shows three stages: 'Status' (33), 'Wish' (50), and 'Wish' (99). A red arrow points to the 'Owner: Eugene' field in the list of requirements. The list includes details such as 'Ambition Level', 'Scale', 'Stakeholders', and 'Today's Level Of Quality'.

Requirement	Status	Wish	Wish
Long Term One Year Requirement .Wish [User Experience = Ordinary Average Frequent Things, User Type = {Child, Pensioner}, Understanding = {Can Do Most Daily needs, Can Circumvent and Trick It}, Skill Level = {Random Results Unpredictable, All}, Constraint = No Formal Training] @ 03 Jan 2019 : 99 ? % Users Get It <- tsg	33	50	99
Owner: Eugene			
Ambition Level: new users should be able to understand their system quickly			
Scale: % [User Experience] for [User Type] who attain an app [Understanding] and [Skill Level] within a [Constraint]			
Stakeholders: European Union, Local Authority, Regulators, Residents, The Law.			
Today's Level Of Quality. Status: 33 ?? % Users Get It [User Experience = Ordinary Average Frequent Things, User Type = {Child, Pensioner}, Understanding.			
Extreme Short Term Sprint. Wish: 50 ±49 ?? % Users Get It [User Experience = Ordinary Average Frequent Things, User Type = {Child, Pensioner}, Underst.			
Long Term One Year Requirement .Wish: 99 ? % Users Get It [User Experience = Ordinary Average Frequent Things, User Type = {Child, Pensioner}, Und.			

Figure 1.87 : 'Eugene' is the designated spec object (the User Experience Aka Usability Value requirement) Owner. 'Source BCS April 2018, Waste Management'

The spec Owner should:

- have accepted the Owner role voluntarily
- Be more than usually knowledgeable in the specific requirement
- Be interested in making the spec the best possible, over time; motivated.

The spec Owner is responsible for:

- receiving any hints from any sources, like stakeholders, of the need for corrections, updates, and changes
- Being password-enabled to actually do, and publish, any change
- Informing all instances, documented in the specification object, all relevant stakeholders, of the pending change, and the actual change (according to corporate guidelines for changes)
- Quality controlling, and reviewing changes, personally, or using others, and using Rules for specification best practices.

Notice what this means:

- we have *decentralized* change control to motivated people

- Control is no longer at a *committee* level, a level that does not really have time or interest in the many individual planning specification objects.
- You can use this decentralized responsibility to activate many people, including juniors and trainees, to grow in experience and motivation, into the larger planning system.

13.2 Annotation of change source, and time stamps

I am pretty clear that we need to annotate the 'Source' of each individual element of a plan. At the same time it is a good idea to get a time stamp for exactly when changes are made.

There are two change sources:

- the Spec Owner, or whoever actually keys in the change
- The information Source: 'who exactly said 64%?', or 'London?'

Owner: Eugene

Tag.Ambition Level:

new users should be able to understand their system quickly

Templates ▾

Source: by tomgilb - Jul 14th 2019, 15:09

Eugene, edit by Tom g July 2019 for Value Requirements book

Templates ▾

tomgilb added a comment - 2 minutes ago - edited by tomgilb - a minute ago

There was a typo in the sentence from Eugene, which I corrected for book presentation.

Add Comment...

Scale: % [User Experience] for [User Type] who attain an app [Understanding] and [Skill Level] within a [Constraint]

Stakeholders: European Union, Local Authority , Regulators, Residents, The Law.

Today's Level Of Quality.Status: 33 ?? % Users Get It [User Experience = Ordinary Average Frequent Things, U

Extreme Short Term Sprint.Wish: 50 ±49 ?? % Users Get It [User Experience = Ordinary Average Frequent Th

Figure 1.88: A detail window of the 'Ambition' parameter specification. Sources and change details are there.

A simple way of noting the source of any statement, is to use the keyed icon '<-'

For example:

Wish: 99% <- Tom

Example 1S.

And then there is the question of exactly WHY a change was. Made, its justification, or background.

This justification is important because:

- We need to make sure the change is really justified.
- We need to explain to other stakeholders why the change is being made.
- Other stakeholders need to be able to argue about that justification.

A simple way of adding justification information can be:

Wish: 99% <- Tom

Rationale: this level is necessary to beat competitors.

Example 1T

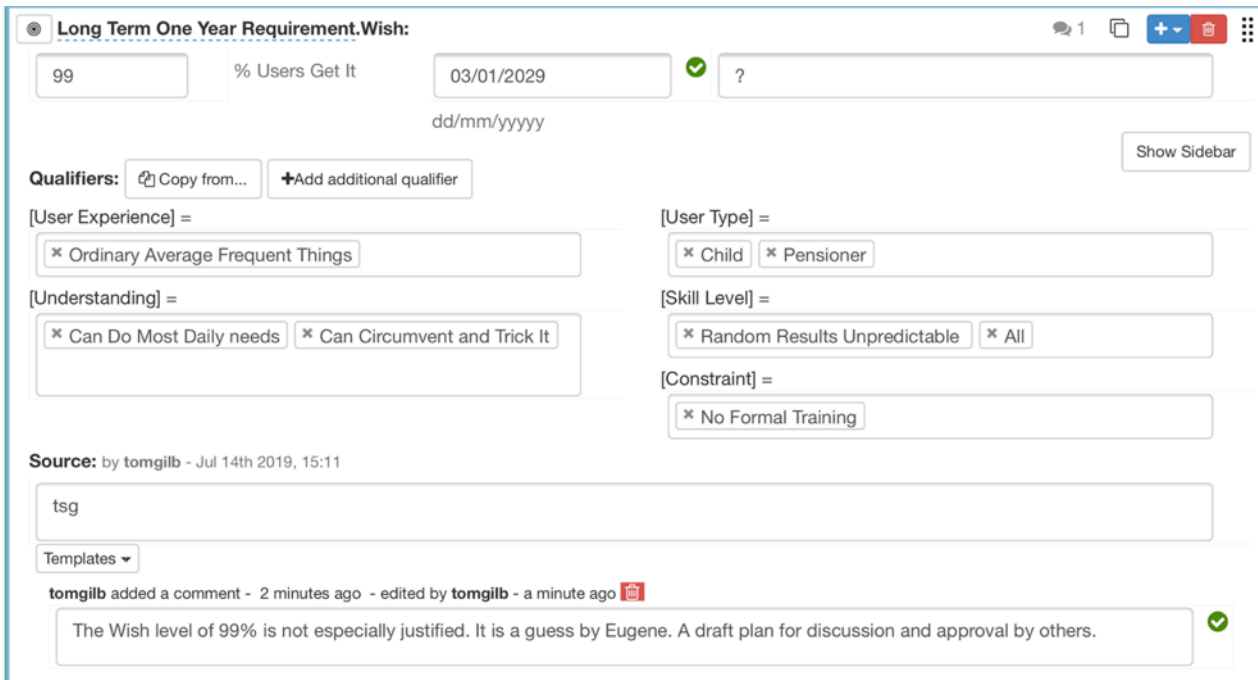


Figure 1.89: I used the Comment, in the Wish window to explain why the 99% level was set. An app can support and remind people to document 'Source' and 'Justify', in more detail

In this case the example is a Strawman, an initial draft subject for discussion, and improvement. Clearly not to be locked in and taken seriously, yet. Hopefully it is obvious to the reader why this is important to know, and know in writing, near the specification.

These *background* information can be backed up by specification Rules, like

1. The actual responsible Source of all critical specifications will be noted by personal name, position, or a group name.
2. All critical specification details shall be connected directly and locally to a justification, or Rationale, for why it is specified exactly they way it is. Even if the answer is that there is no good reason yet. It is a wild guess or strawman. Be explicit about that.

Example 1T

13.3 Ways of controlling the whole of the specification

How can we exercise control over the *entire* Value Requirements specification ?

There are a great many processes discussed here for ensuring the overall quality of the total set of Value requirements. And each component.

Examples of QC-Supporting Processes:

- Specifying 'background' things which allow us to verify and understand a specification (sources, stakeholders, justifications, comments)
- Giving power and responsibility to people, with *their name* on it (Owners, Sources)
- Leadership: showing that you *really* care to do things well, and knowing when not to overdo it, so it seems like a silly bureaucracy. A 'balance'.
- Retrospectives, root cause analysis, DPP Defect Prevention Process (ref. G) will all bring out reasons for problems. Hopefully 'root causes', including not yet taking the quality of requirements specs seriously enough. These analysis are your potential 'war stories' to remind people of the value. of 'doing it right the first time'.

- Motivation and culture change takes time, and leadership.

Chapter 14. A Review of Requirement Methods Compared to Planguage

14.1 General observations of methods for specifying Value Requirements

I am quite disappointed in the prevailing culture of dealing with Values, and Value requirements.

That is why I have had to invent my own way.

The current unhealthy requirements culture is very widespread, and new bad methods seem to spring up quickly and spread widely.

But our projects continue to fail, and part of that is bad requirements.

My central criticism is that most methods do not **quantify** the Value requirements at all. And the few that do so, do not do it *well*.

The following material, is for people who would like more-specific background.

They might have to attack some Holy Cows in their 'Temple', in order to deal with these problems.

14.2 A Checklist for understanding capabilities of value requirement Specifications

Here is a basic checklist, I do mentally, to **compare any requirements method with Planguage.**

1. Is the Value Quantified (or is it just nice words?). (“Highly efficient”)
2. Is a re-usable **Scale of measure defined well**, or is an over-simplified badly-defined scale only hinted at, together with the numeric level (“35% agree”)
3. Is the requirement **tagged** in some way, or is it just a bullet point, a sentence, or sub-clause?
4. Is there any systematic way used to **define terms** used in the spec, or are we left guessing at clarity and ambiguity?
5. Is there any **structure** in the Scale similar to our Scale Parameters? How is this variation and definitions of (whom when, why, where) dealt with?
6. Is there any way to annotate or capture the **justification** for a requirement?
7. How do they capture **sources** of requirements ideas?
8. Is there any set of **Rules** for requirement specification which could be the basic for Spec Quality Control: the defect level?

9. Is there any concept of **measured Defect Density**, which could give a basis for Exit from the requirements process?
10. Does the process simply capture a raw ambition level requirement, and leave it at that, or is there an attempt to analyze it and come up with a **better clearer requirement**.
11. Does the requirements process actually **permit 'designs' to sneak in as requirements**, when the real requirement is unstated, implied, or badly formulated? ('We want a password for Security')
12. Is there any concept of **stakeholders** for the requirements?
13. How good is the capture of **background information**, to help understand quality, risks, relations, priorities?
14. Are **Benchmark** levels systematically captured (Past, Status, Record, Trend)
15. Are the requirements suitable for **digital automation**? Can you program visual presentations, and analyze the specs?
16. Is there a **well defined classification and definition of requirements types**? (Function, Resource, Value, Mandatory Design, Constraint, Scalar Constraint, Scalar Target).
17. There is more, but this list should separate *strong* Value spec methods from weaker methods.

Chapter 15. SOME COMMENTS ON SPECIFIC METHODS

Not all these following methods below are 'requirements' methods, as such. But they are *related* to Value Requirements in interesting ways, and I want to share my observations with the reader, so that they themselves in turn, can argue with others better about the methods.

In some cases I have written a special paper in more depth about the method, and I shall refer to it for detail, and just give the 'highlights' here.

15.1 User Stories (ref. H, a)

I commented early in this book about User Stories. They are at the level of an Ambition Level, and we can use User Stories to start the process of deeper understanding of the implied Value requirement. But User Stories do not pretend to go into depth themselves.

My good friend Mike Cohn (Mr. User Stories) specifically referred to our Planguage methods, when asked on his website what to do about qualities and quantification.

As I said, I like the fact that the User Story does not merely have a 'requirement' idea, but that it specifically includes information about the 'stakeholder' (ok, 'User' only), and the justification (because)

As a method for 'generating ideas about requirements', and possible values and qualities, for small and less-critical systems (no state of the art competition levels, no huge national health systems) user stories are quite OK.

My problem is, that I see user stories being used way beyond their 'level of competence', and I think user stories, as a primary requirements culture, are probably one initial cause of project failure.

Success and failure are not defined by user stories; they are more of a detail. But as we have pointed out earlier, the Value level 'Tolerable level' defines a failure border, and Goal level defines success.

User stories just do not deal with values and qualities, so we need something more, operating at a higher level of controlling the system stakeholder results, values, and qualities.

My advice, if you are committed to using them, is to use them as an Ambition Level, a simplified departure point, and then analyze what the real, but implied-only, 'value level' has to be (derive a Scale and a Wish for example).

15.2 Use Cases

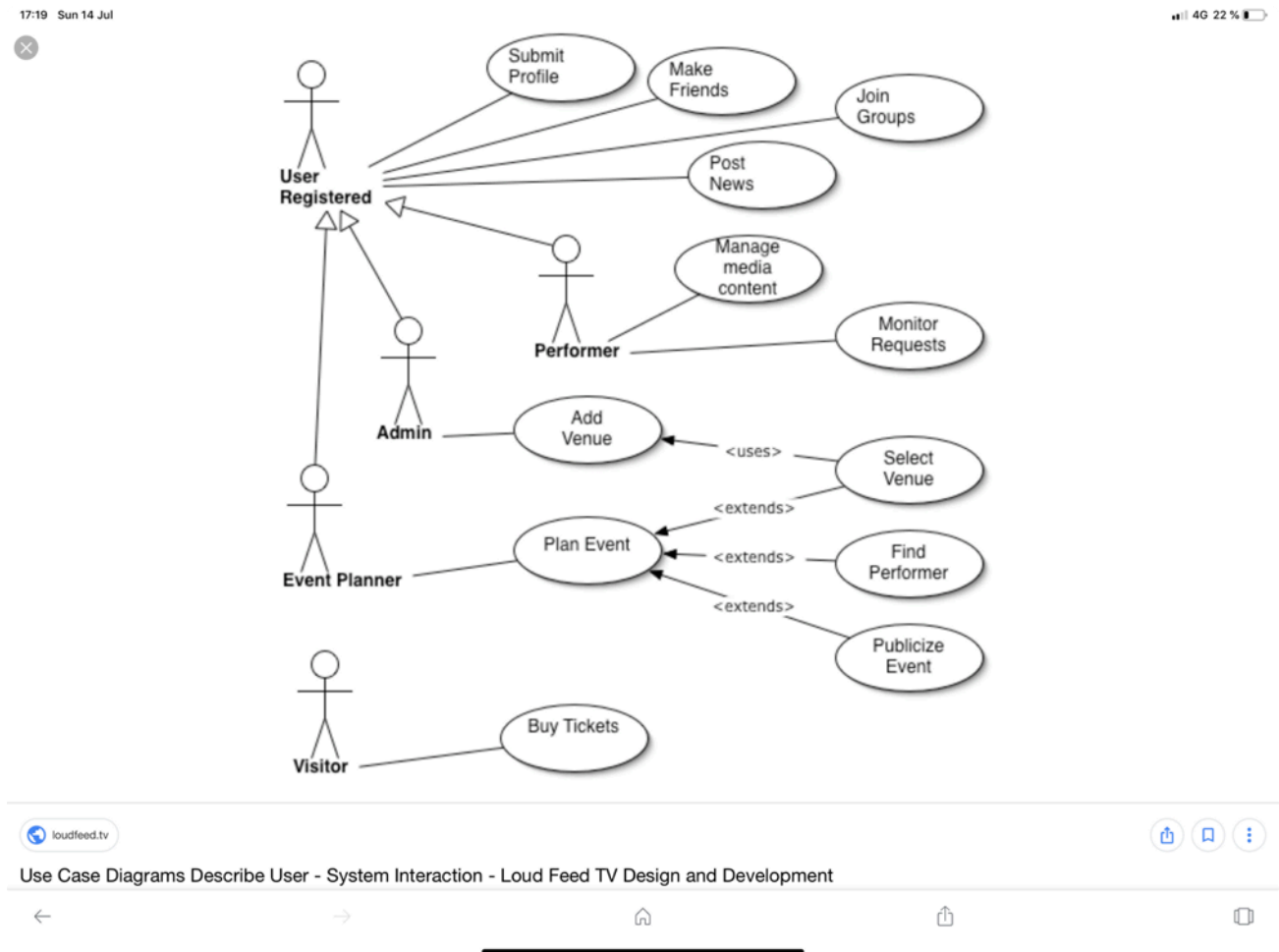


Figure 15.2.1: Use Case Diagram. Notice the 'Actors' (Admin etc) which we would prefer to make more general as 'Stakeholders'

Use cases are of course not complete requirements, nor Value requirements.

They are in fact very close, but not identical to, what I call 'Scale Parameter Attributes'.

Use Case Success Rate

Level: Stakeholder, Type: Value, Labels: - Edit

Status 80 Wish 95

Wish [Stakeholder = User Registered, Use Case = {Submit Profile, Make Friends,Join Groups,Post News}, Actions = None] @ 14 Jul 2029 : 95 %

Ambition Level: Do the use cases correctly

Scale: % Successful Completion of [Stakeholder] [Use Case] [Actions].

Target Time Units: Calendar Date

Actions: defined as:
Uses, Extends

Stakeholder: defined as:
User Registered, Performer, Admin, Event Planner, Visitor

Use Case: defined as:
Submit Profile, Make Friends, Join Groups, Post News, Manage Media Content, Monitor Requests, Add Venue, Select Venue, PPlan Event, Find Performer, Publicize Event, Buy Tickets

Source:
Tom Gilb for Use Case example in Chapter 14 Value Requirements book

Status: 80 % [Stakeholder = User Registered, Use Case = {Submit Profile, Make Friends,Join Groups,Post News}, Actions = None] When 14 Jul 2019

Tag.Wish:

95 14/07/2029 %
dd/mm/yyyy

Qualifiers: Copy from... Add additional qualifier

[Stakeholder] =
* User Registered

[Use Case] =
* Submit Profile * Make Friends * Join Groups
* Post News

[Actions] =
* None



Figure 15.2.2: Compare this directly to the Use Case figure above. The 3 arrows point to Scale Parameters, each of which has 'Space Cases'.

So we can now more clearly see what Use Cases are. They are essentially Scale Parameter attributes, or for fun 'Space Cases'.

So my 'Space Cases' (a term I just invented to express the broader scope than mere *Use cases*) are digitally integrated into the Requirement Spec., and can cover a broader *space* category.

For example we could add such Scale Parameters as:

- Places (where, city, country, area, groups like EU, NATO)
- Situations (War and Peace, Recession, Brexit, Natural Catastrophe)
- Experience and education levels of stakeholders
- Event Conditions (ordered, confirmed, attempted delivered, delivered, for example)
- And any other dimension spaces you need to express as conditions, for a requirement.

So my preference would be to not use the Use Case method, but instead to integrate the basic idea of Use Cases into Planguage with broader 'Use' Cases. In other words by using '[Scale Parameters]'. :)

15.3 Earned Value Management (EVM)

I recommend this EVM overview

https://en.m.wikipedia.org/wiki/Earned_value_management

I would have hoped that EVM would deliver exactly what the name implies. But it does not.

It does not deal with a set of critical Value requirements, at all.

It does assume Big Bang waterfall model pretty much, and 'value' is really just 'work done', or 'tasks', sometimes simply '% of budget spent' !

I recommend the blogs of a professional friend who spends his time fighting for non-corrupted, honest versions of EVM in US Government Projects, <https://www.pb-ev.com>. Paul Solomon, who has written a book on the subject with another friend that I have worked on several US Government Projects with, Ralph Young.

These guys are honest idealists, so you can trust what they say about EVM.

Of course when a desperate Government, *dictates* EVM, in an attempt to control greedy, and technically incompetent subcontractors, it gets used, and abused.

There is little EVM interest outside of those circles.

15.4 Functional Requirements and Non-functional Requirements

My definition of 'Function', and 'Functional Requirements' (which I call 'Function Requirements') is 'what a system does'.

Similar to the Use Case Actor actions. What they *do*.

But I observe that there is little agreed discipline in using the Function term. It can easily cover any type of requirements. And as often as not, 'function' can be applied to what really are 'design'. So the situation is messy.

A 'Function' can be programmed by a programmer. So can some designs. Both, functions and designs, are *binary*, present or absent. Nothing in between. Both are testable, for presence or absence. Both are in some sense, therefore, simple. 1:0.

People *outside* of IT do not seem to have a problem here.

When programmers were reminded that there were some qualities they were not good at, like *Usability*, they observed that this was 'not a function' or design they could program. So, they solved their dim understanding (of a Value) by calling it a 'non-functional' requirement or attribute.

I have seen what they then do with this requirement category. They specify it as 'TBD', to be determined, someday, when we figure out what it is.

One problem is that although people mean 'qualities' (and Values) when they say 'non-functional': things are not so simple.

There are very many other requirement attributes to consider.

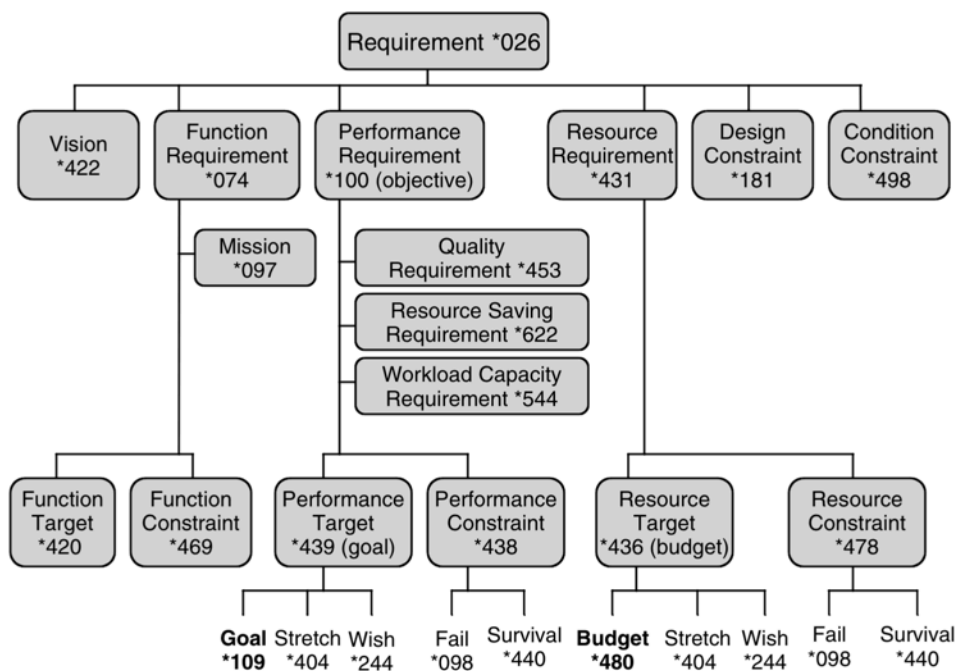


Figure G20 Requirement Concepts.

Figure 15.4.1: Planguage requirement concepts. From the 'Competitive Engineering' book. There is Function, then Quality, and all the others which are not functions!

From this 'Value Requirements' book point of view, we are interested in all those system attributes which stakeholders value.

That is pretty much everything, including Functions. That is why they are called requirements, I guess.

But this book has chosen to focus on the more-complicated value requirement, because it is variable, and people have a problem with variables. They don't stand still.

PS the use of the term 'non-functional' is a dead give-away that people have no real understanding of requirements.

15.5 Balanced Scorecard



Figure 15.5.1: a BSC with emphasis on Key Performance Indicators (KPI). This is the best of a lot of bad examples. It does try to quantify, and has benchmarks and targets.

Source: Datapine.com

The original Harvard Business School, 'Balanced Scorecard' failed in my opinion because it

- recognized there was an imbalance between financials quantification, and non-financials
- But it failed to quantify the non-financials (as a *norm*, not an exception)

Later efforts have tried to be better at quantifying the non-financials, such as the example above.

But (BSC improvement efforts):

- they still fail at tackling the really critical values
- They admittedly prioritize things, which they find easy to quantify (still 'unbalanced')
- They do not depart from the 'really critical values', and find a way to quantify them
- Notice in the example above, the total lack of qualities, or anything ending in in '-ility' ?
- They are avoiding the issue, because they do not know how to deal with it.
- Notice there are absolutely no product or service qualities in the example at all.
- Notice there are no well-developed Scales of measure at all, just highly-ambiguous ones. "Sustain Customer Retention". OK as an Ambition Level, but not as a well-defined Scale of measure.
- Some points for having both a Benchmark and a Target level. But notice no constraint, worst-acceptable-case level (Tolerable). No dates set on obtaining the target levels
- And notice a mystical "Likelihood of Reaching Target", done how? By whom? Any validation that it works?

Financial Performance		Non-Financial Performance	
1	Risk-adjusted ROE/ROA/EPS	1	Regulatory performance
2	Delinquencies/Losses	2	Employee satisfaction
3	Risk-adjusted customer/product/organizational profitability	3	Customer satisfaction/Net Promoter
4	Efficiency	4	Employee turnover
5	Non-risk adjusted ROE/ROA/EPS	5	Customer engagement
Operational			
1	Operational loss/fraud	4	Cybersecurity metrics
2	Forecast (budget/risk) accuracy	5	Time to decision
3	Strategic initiative execution		

CenterState Correspondent Bank



How To Create Effective Metrics for Your Bank | CenterState ...

Images may be subject to copyright. Find out more

RELATED IMAGES

SEE MORE

Figure 15.5.2: example of someone's idea of bank metrics in a BSC context. At this level they are just name Tags. I wonder what real Scales of Measure would look like?

Focused on finances? Here are 68 more financial KPIs your bank might want to measure.

Quality

9. **Client Survey Score:** Bank performance as measured by customer feedback. Many banks send out client surveys to gather performance-related feedback; tracking these responses with some type of internal scorecard is helpful. You can even create categories for response types (e.g. employee communication, variety of products/offers, speed of service, etc.) and track them individually, as well as your overall customer satisfaction score.

10. **Average Time To Close Issues:** Length of time from when a problem is identified to when it is solved. Issues may originate internally (operations, technology, etc.) or externally (customers).

11. **New Account Setup Error Rate:** The total number of new customer accounts created containing an error (e.g. typo or incorrect address, name, account type, etc.) divided by the total number of new customer accounts set up at the same point in time, shown as a percentage. This metric will ultimately link to the previous "Average Time To Close Issues" KPI.

12. **Accounts Opened With Insufficient Documentation:** The total number of new accounts opened with insufficient documentation divided by the total number of new accounts opened over the same period of time, shown as a percentage. This is similar to the previous KPI for banks, but in this case, the information is missing versus incorrect.

Figure 15.5.3: This example is getting closer to specified Scales of Measure.

Big US Government Bank Case (e)

I consulted with a large US bank, one that was key in the recession of 2008. The top management were trying to use Balanced Scorecard.

They were tearing their hair out in frustration to try to make it work. I saw their problem, and helped them solve it.

They were trying to communicate about a lot of 'soft' management objectives, that were not defined well, no scales, and so everybody had to 'make up a definition' in their own mind.

Their relief when I showed them how to do that, was immense.

The details are in the reference (ref. e). But it is the old story, no knowledge or teachings yet, of how to define a concept with a scale.

This is not merely a BSC problem. It is a widespread cultural problem.

In this case it was at Harvard Business School that BSC was developed and published. But, I find that no business schools, which I can identify, teach managers the skills of this book: *how to define any Value Scale.*

Yet if we do as advised earlier, just search the internet for things like "Bank Employee Efficiency METRICS"

We get far more interesting metrics and insights than the BSC above. <https://www.clearpointstrategy.com/bank-kpis/amp/>

15.6 Quality Function Deployment (ref. I)

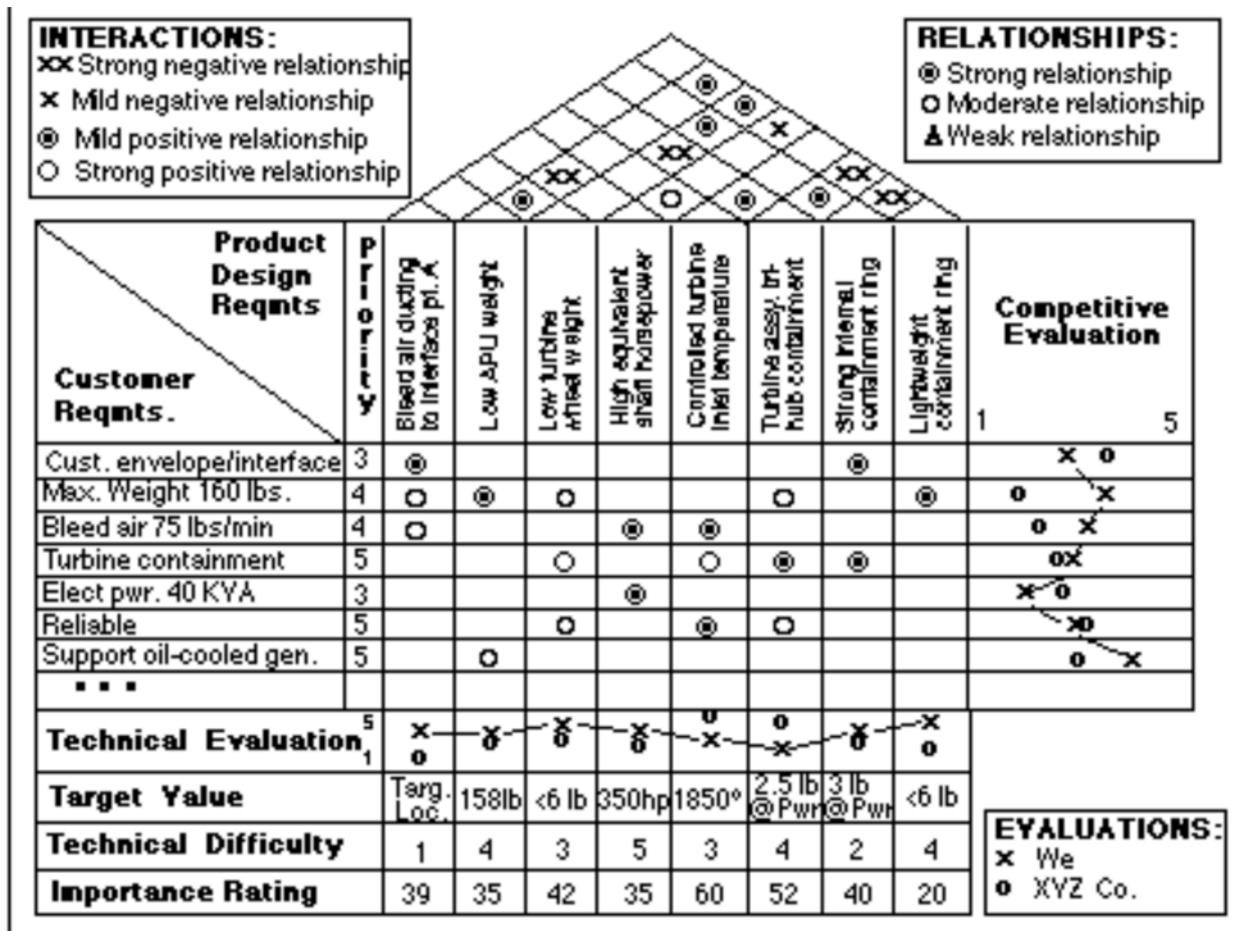


Figure 15.6.1 : typical QFD and House of Quality example from www.

Just about everything is wrong with this method. I used to joke in my classes and lectures that QFD is so bad that it must be Japanese Fake News to destroy western industry. The fact that Toyota really did that intentionally, fake news to fool Western Industry, was revealed to me in 2018 (ref. f).

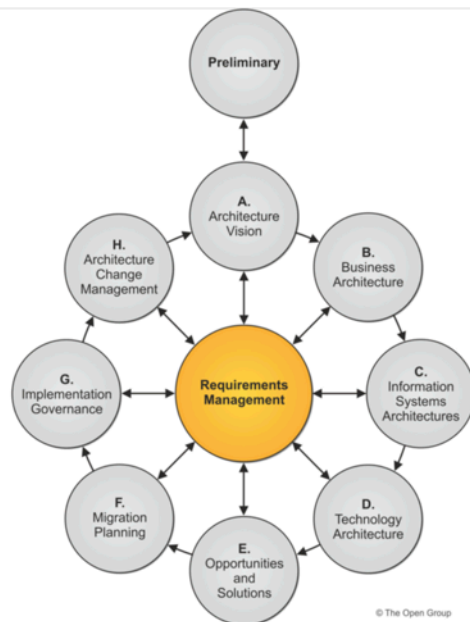
The above visual example is filled with badly-defined values, and subjective judgements. See the references (I, f) for details, and see the checklist above. But it looks so systematic!

The shocking thing from my point of view is that this is taught at universities, without any critical points against it being made.

The mentality, is, 'well they used it at Toyota, and Toyota make good cars, so it must be a good method'.

I'm told Toyota workers eat once a day at least, so that must be a good method for success too.

15.7 Togaf



© The Open Group

pubs.opengroup.org

The TOGAF Standard, Version 9.2 - ADM Architecture Requirements

Figure 15.7.1: Requirements are, formally, central in the Togaf Architecture Method.

There is much talk in Togaf of quantification, stakeholders, assumptions, constraints, KPIs, and Success definitions. But it is difficult to find any detail, of what this means in practice.

The screenshot shows the ArchiSurance - Visual Paradigm Enterprise interface. The main content area is titled 'Architecture Requirements' and contains three sections:

- Business Domain Requirements:** A table with two rows. The first row has 'Requirement' 'Digital customer management capability' and 'Description' 'B * ≡ * ≡ * F * [UI] * [G] * [F] * + * [H] *'. The description text below reads: 'The Digital Customer Intimacy strategy of ArchiSurance requires the digital channel management capability, to be realized by social media competency and social media apps.' The second row has 'Requirement' 'Data-driven insurance capability' and 'Description' 'B * ≡ * ≡ * F * [UI] * [G] * [F] * + * [H] *'. The description text below reads: 'The Digital Customer Intimacy strategy of ArchiSurance requires the data acquisition and data analysis capability.'
- Constraints:** A table with one row. The row has 'ID' 'C1', 'Title' 'Consistent behavior in different devices.', 'Description' 'B * ≡ * ≡ * F * [UI] * [G] * [F] *'. The description text below reads: 'The interaction patterns between devices (Android, iOS) must be consistent.' 'Priority' is 'High' and 'Consequences' is 'A unified UX design has to be made for different devices.'
- Assumptions:** A table with one row. The row has 'ID' 'Enter ...', 'Title' 'Enter input here.', 'Description' 'B * ≡ * ≡ * F * [UI] * [G] * [F] * ». The description text below reads: 'Enter input here.' 'Date' is 'Input date here (y...', 'Source' is 'Enter input here.', and 'Owner' is 'Enter input here.'

Fig-

ure 15.7.2 : Here is the best concrete example I could find, on the internet, of Togaf requirements practice. Just above this example, is text about 'quantified requirements' being mandatory. Do you see any quantification?

I see some kind of an 'Ambition Level' ("Consistent Behavior", "...Capability") buried in the name tags.

I also see the usual combination of a sort of requirement ("Digital Customer Management Capability") together with a suggested technical architecture ("the data acquisition and data analysis capability"). Bad combination: vague ends and suggested vague means!

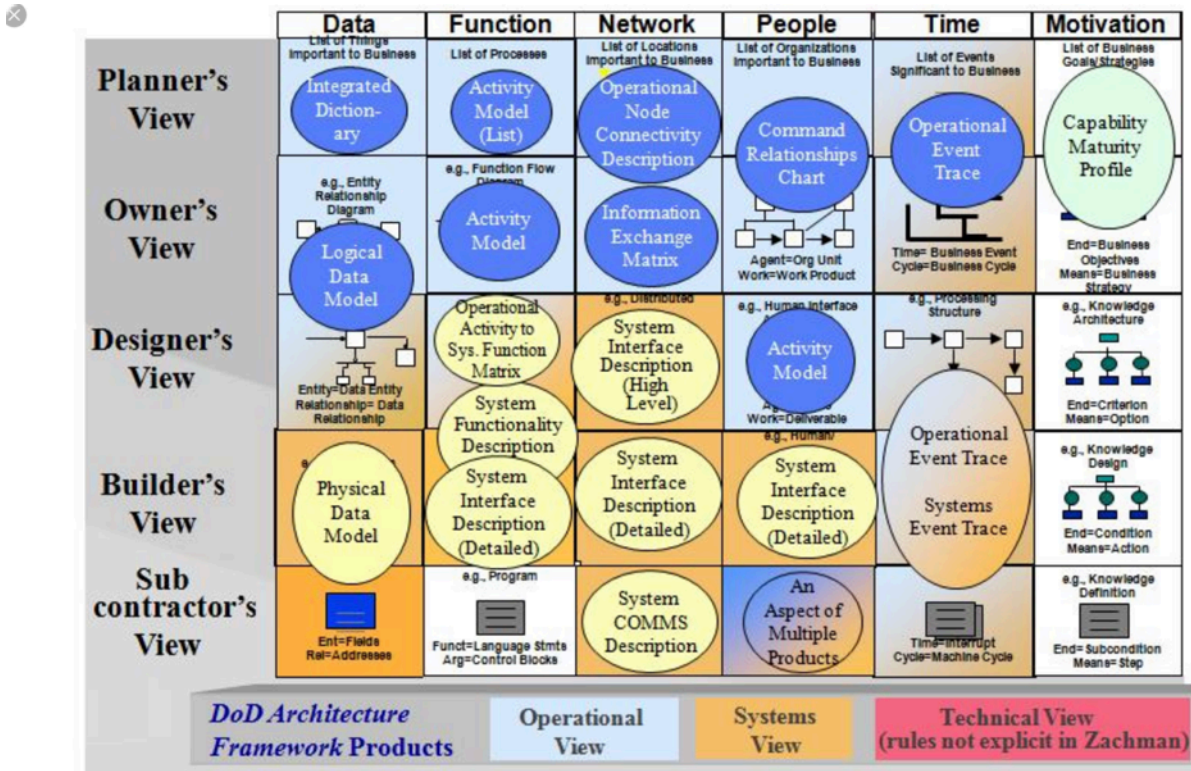
I have also seen Togaf practice amongst my clients, and my Architecture Engineering course students. I was never impressed. Just disappointed in the practice.

My conclusion regarding requirements for Togaf is that the quantified practice they recommend, and I agree with, is simply not taught or practiced.

When architecture rests on such a bad 'requirements foundation', the result must be disappointing.

Togaf people are of course welcome to adopt the ideas in this book. These would conform with many of their stated ideals. No extra charge, just credit your sources.

15.8 Zachmann Framework



en.wikipedia.org

File:DoD Products Map to the Zachman Framework Cells.jpg - Wikipedia

Figure 15.8 : Zachmann framework mapped to US DoD Products.

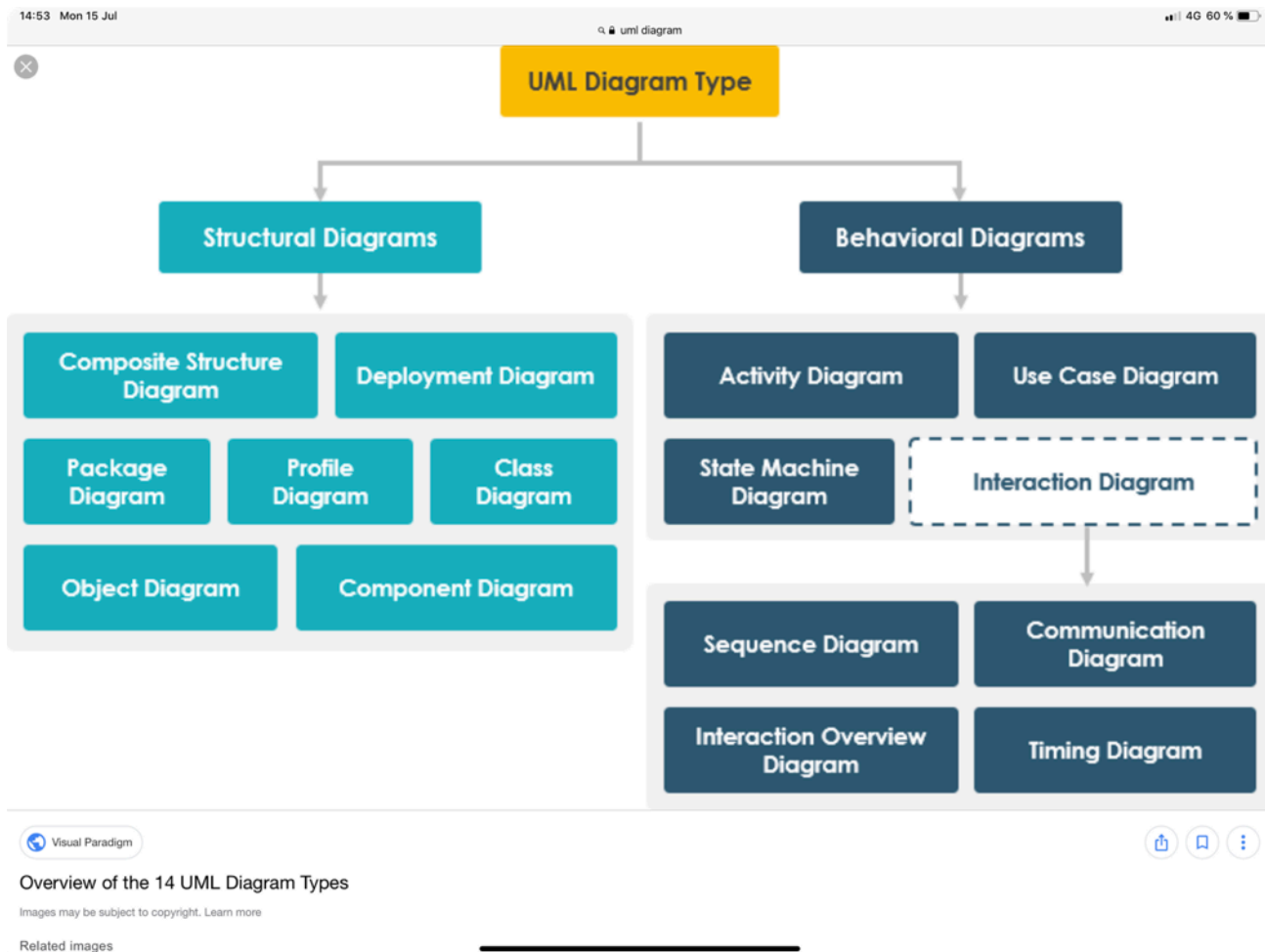
I confess to a weakness for Zachmann. His framework covers a lot of bases, a lot of the system.

As this figure reminds us, we cannot expect much detail of a framework. It is up to the user of the framework to fill the intersections with specific methodology.

The Capability Maturity Model, level 4, was explicitly (Ron Radice IBM) based on my Software Metrics (1976) book, quantification of qualities and values. So that is an example of these Value Requirements ideas in this book, put in any framework (CMM, Ch. 15.14), you like.

15.9 UML: Unified Modeling Language

Fig-



ure

15.9: a set of UML modeling diagrams.

UML models a lot of things, but values, costs and qualities are not amongst them.

From my point of view this makes UML, and many others like it, quite inadequate for modeling the real world, and some of its most important aspects (Values, qualities, and costs).

I believe this is due to the built-in narrow-mindedness of a computer-programming culture, where the program can be constructed without reference to costs and qualities.

15.10 Design Sprints (ref. g)

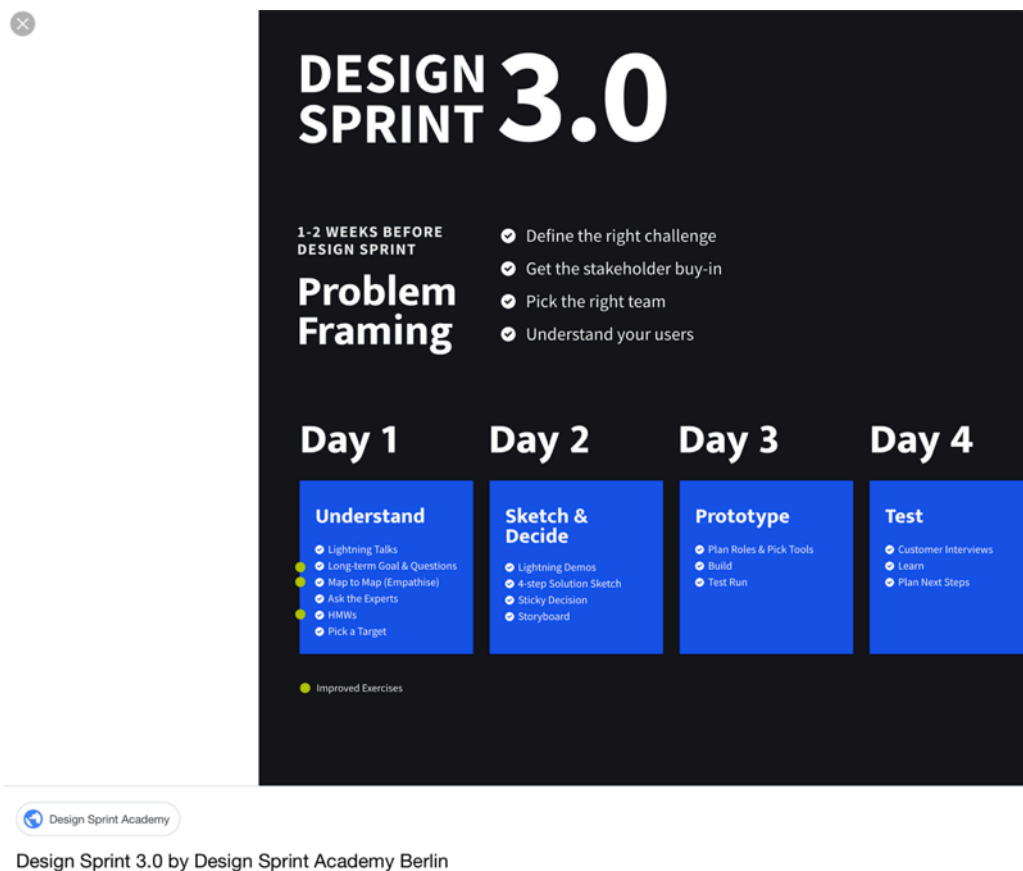


Figure 15.10.1 : Design Sprint 3.0

Design Sprints (1.0, 2.0, 3.0) are getting better, but they do not have any concept of quantification of Values as requirements. It is still a yellow-sticky culture, where the emphasis is on finding an app or web design, rather than departing from a clear set of multiple Value and constraint requirements. Maybe good for simpler problems: but I have looked and not found any studies comparing Design Sprints to anything else, for example in terms of project success, productivity, value for time spent.

Planguage offers a similar better startup week idea: The Project Startup Week (ref. K). It has been applied to large banking, aerospace, and defense projects successfully for decades.

15.11 The 'Evo' Project Startup Week (ref. K) : Values Driven Start

The Project Startup Week is fully compatible with the Value Requirements ideas in this book.

Day 1: Top 10 Critical Project Requirements Quantified

Day 2: top 10 architecture (design) options on the table

Day 3: Estimation of all designs impacts, on all Values and costs

Day 4: Decomposition of big designs into smaller ones, and selection of a high value-to-cost design-increment to implement in a 'sprint' next week.

Repeat every week, Step 4, to increment towards the Value Goals.

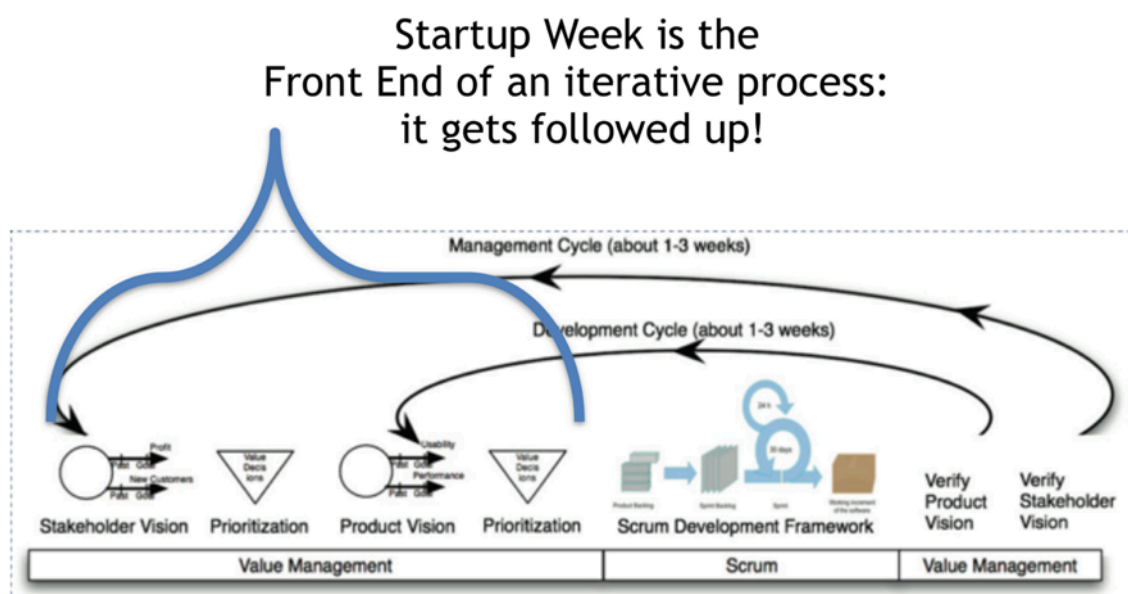


Fig-

ure 15.11.1 : The Evo Startup Week focuses on quick stakeholder value production, measurably.

This startup is primarily driven by a set of quantified critical stakeholder values.

It does not try to get a mock-up, or prototype, working in the first week. It tries to get real measurable results, a value stream, with currently existing products, services and systems.

It tries to learn by stakeholder feedback, and incremental measured results, what works, and what does not.

Evo's impact on Conconfirm product qualities 1st Qtr

- Only 5 highlights of the 25 impacts are listed here

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Conconfirm Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000



Release 8.5

Figure 15.11.2 : The best 5, of 25, measurable Value improvements in 1st release, 12 weeks of increments, from. Start of using the Evo Incremental Value process, after a startup week to quantify the 25 Values most critical. Source, Conconfirm.

15.12 OKR (K) Objectives and Key Results.

Jennifer				
	Description	Results	9-2-2018	16-2-2018
Objective	Create an engaging newsletter that creates brand advocates	0%	20%	94%
Key Results	Increase CTR by 3% (is currently 5.4%, KR = 8.4%)	0%	0%	0%
	Find 5 new ways to drive traffic to newsletter signup	0%	40%	60%
	Include 1 podcast section per month in newsletters (= 3 M30 related podcasts featured)	0%	33%	33%
	Give something away each month (= 3 things) - first is AgiNext	0%	33%	33%
Objective	Create an engaged social media that shares M30 with a broader audience & shows we care about our customers			
Key Results	Post five days a week on M30 Facebook = 60 posts https://www.facebook.com/pg/Management30/posts/	0%	0%	6%
	Send 5 announcements to the LinkedIn Group for Q1 https://www.linkedin.com/groups/4074448	0%	0%	20%
	Welcome new taters on social media - 1 a week (= 13 posts)	0%	0%	0%
	Bring back tater of the week but on Facebook/Twitter (=13 posts)	0%	0%	8
Objective	Hire a Paid Online Marketing person			
Key Results	Interview 3 people for role		33%	33%
Objective	Finding ways for HM to pivot to M30			
Key results	100% of HM social media profiles go to M30 (FB, LI & YT done, Twitter not)		50%	75%
	Half of all blogs go to M30 (will go up and down each week)		50%	66%
	Facilitate 1 Challenge Month (April regarding gratitude/Kudos)		0%	0%

Management 3.0



Do key results lead to objectives? My not-so love affair with OKRs ...

Figure 15.12.1 : an example of OKR planning. Notice the objectives are not quantified and clear ("Create an engaging newsletter"). The 'Key Results' are not business results, they are individual tasks (Interview 3 people"), which the individual assumes (hopes?) will produce the vague objective. Good luck!

I have no problem with OKR as a way to make individuals and small groups plan their weekly work tasks. Maybe it is a good thing?

But I have had problems finding any studies of OKR, and even good case studies, which *illuminate the values we get for the costs*. What was the result at Intel? Do they still use OKR?¹⁷

OKR is in no way a replacement for Value Requirements, which operate on larger systems (products, organizations, services) and

¹⁷ From Erik Simmons, July 2019: "During my tenure, OKRs were still used, but many teams had shifted to (or added) Landing Zones, which had elements of Planguage (and our training recommended using full Planguage behind each LZ row for clarity). Some teams still used OKRs to drive time-based behavior at the quarterly or yearly level, perhaps out of cultural inertia (though that was relatively low at Intel overall)."

guides us to find ways to create measurable value in the short and long term.

For clarity, I do not think that our Value Requirements methods are appropriate for this level of individual task planning.¹⁸

I would like to think that these same individuals, are all part of some larger projects, and that they are interested in, and committed to, improving Value requirement levels.

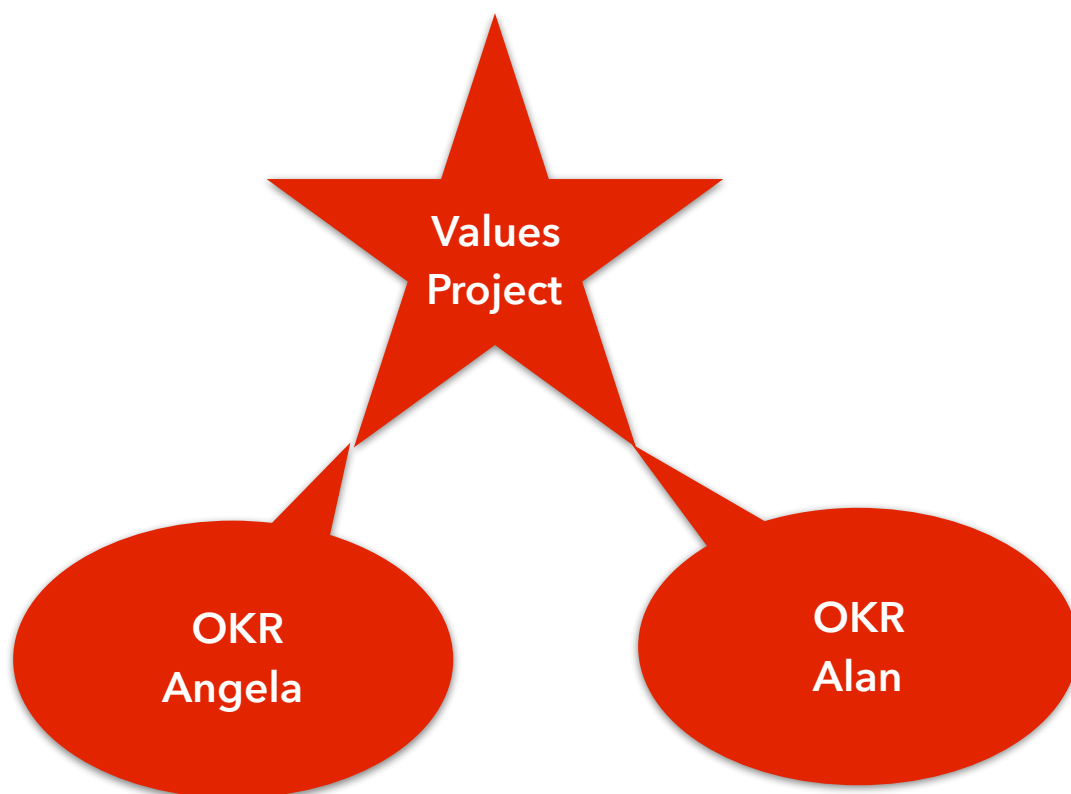
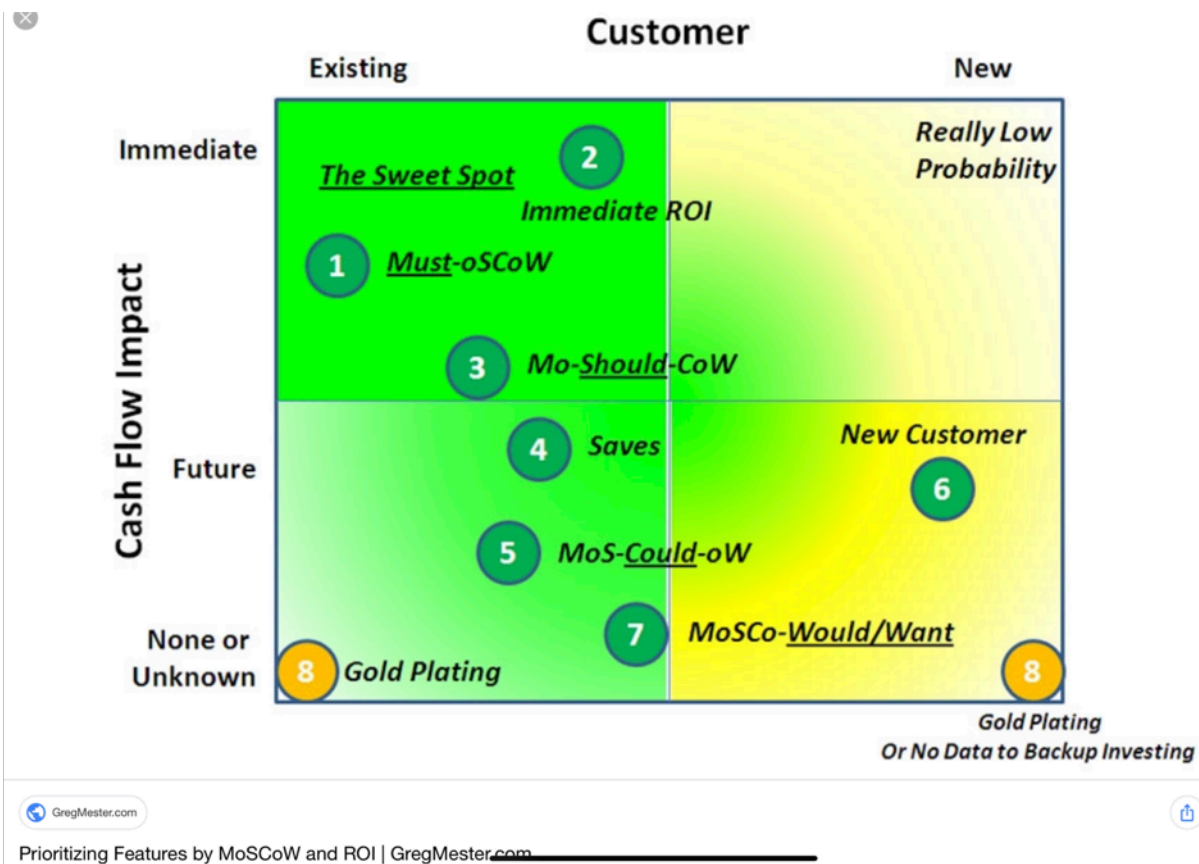


Figure 15.12.2: Individuals and small teams need to be well aligned to a higher 'project' level set of quantified Value requirements. OKR might help individuals with tasks, but it has not been designed and practiced to directly align with a higher purpose. This could well be because the higher purpose (Value Requirements) was never well-defined at all: as most of the methods in this part of the book fail to do.

¹⁸ our tool ValPlan.net does support task planning.

15.13 MoSCoW: Prioritization Method.



Figure

15.13: a presentation of the MoSCoW prioritization method, which tries to bring in financial and market factors in the decision-making.

Prioritization of actions is necessary when resources are limited, as they always are.

I am not impressed with most well-known prioritization methods, this one included, and especially fixed-weighting methods, as are found in for example Balanced Scorecard, and Quality Function Deployment (see above BSC, QFD).

But I'll admit that bad prioritization methods are better than none.

Here are some points about prioritization (ref. E)

Methods for simple short-term prioritization might need improvement for large, complex, critical projects

There are a variety of different **resources** that might be considered in priority decisions (time, staff, capital cost, reputation, hardware capacity, and many more). Even combinations of resources might be considered.

We need to be clear about what is being prioritized. Most methods seem to assume it is features, functions or User Stories: all of which are a bad idea.

I think if *stakeholder critical values* are the main point of all projects, then we need to prioritize getting to the Value requirements, and stop when we get to one. Then re-allocate resources to reach other value target levels which are not yet satisfied.

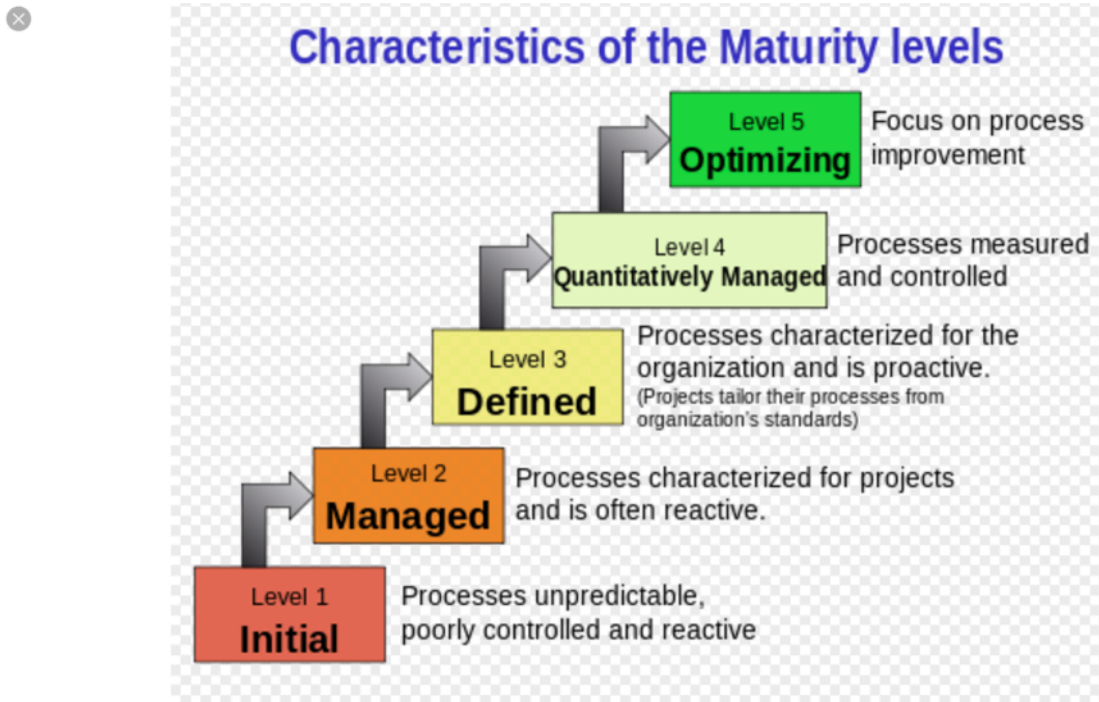
I think asking a single stakeholder, or a Product Owner to choose a priority is a bad idea (but better than none) because they do not have an overview of the rest of the stakeholders needs, and the resource situation in the future for the project.

In short I think priority needs to be computed logically, with delivery-step by delivery-step, based on an overview of the critical decision factors.

MoSCoW is for projects that do not understand or plan critical values quantitatively. It is not for large or complex projects.

It is path to failure I would guess.

15.14 CMM: CMMI, Capability Maturity Model



en.wikipedia.org

upload.wikimedia.org/wikipedia/commons/thumb/e/ec/...

Figure 15.14: CMM is a tool for assessing the maturity of an organization and its working processes.

19

PS Good suppliers should get well paid for **value** delivery, incrementally.

Forget 'Maturity Levels', for organizations.

Focus on Value, performance now, let organizations figure out what to do themselves (their cost-effective *processes*) to get paid: The Free Market of Competence, I'd call it.

19 The CMM level 4, 'quantitatively managed' is mainly based on my 1976 Software Metrics book ideas, according to Ron Radice, who invented it at IBM. But I think the focus on development processes, rather than stakeholder value, is a mistake. We need stakeholder value first, and process cost-effectiveness second.

Chapter 16. A Briefing on Use of Value Specifications for Design and Architecture

Value requirements are obviously just the initial step in a process of *value delivery* to stakeholders.

The next step is to exploit the Value Requirements in order to find out exactly *how* we are going to deliver the values at acceptable *costs*.

This process is known by a variety of names: *design, engineering, architecture, strategic planning, problem solving*, and many more.

But the design process attempts to answer the question: what can we do in practice, to deliver the Values required?

An approved set of critical clear complete Value requirements, sets the stage, for a dramatically better way of doing the design process.

Value requirements allow us to take a very systematic quantified approach to the design process: something which is logically impossible without the Value (and costs) quantification as the base, the foundation stone.

Put another way, we can now ask and answer the question:

“exactly how good is a design, for meeting all Values, within all costs?”

Designers are not accustomed to this situation, because they are not accustomed to quantifying all critical values, and costs, up front.

If you can get such multi-dimensional estimates of 'how good a design is', then you open the door to the following:

- You can look at the *entire problem at once*, all values, costs and designs.
- You can prioritize project actions, based on a view of all *stakeholder priorities*, and all consequent requirements.
- We can include, consideration of *uncertainties*, risks, and credibility of estimates, in our overall considerations and choices.
- You get well-documented *traceability* of the decisions made.
- You have the set of data needed to *automate decisions* such as 'what is the most cost-effective thing to do next?' in the project.

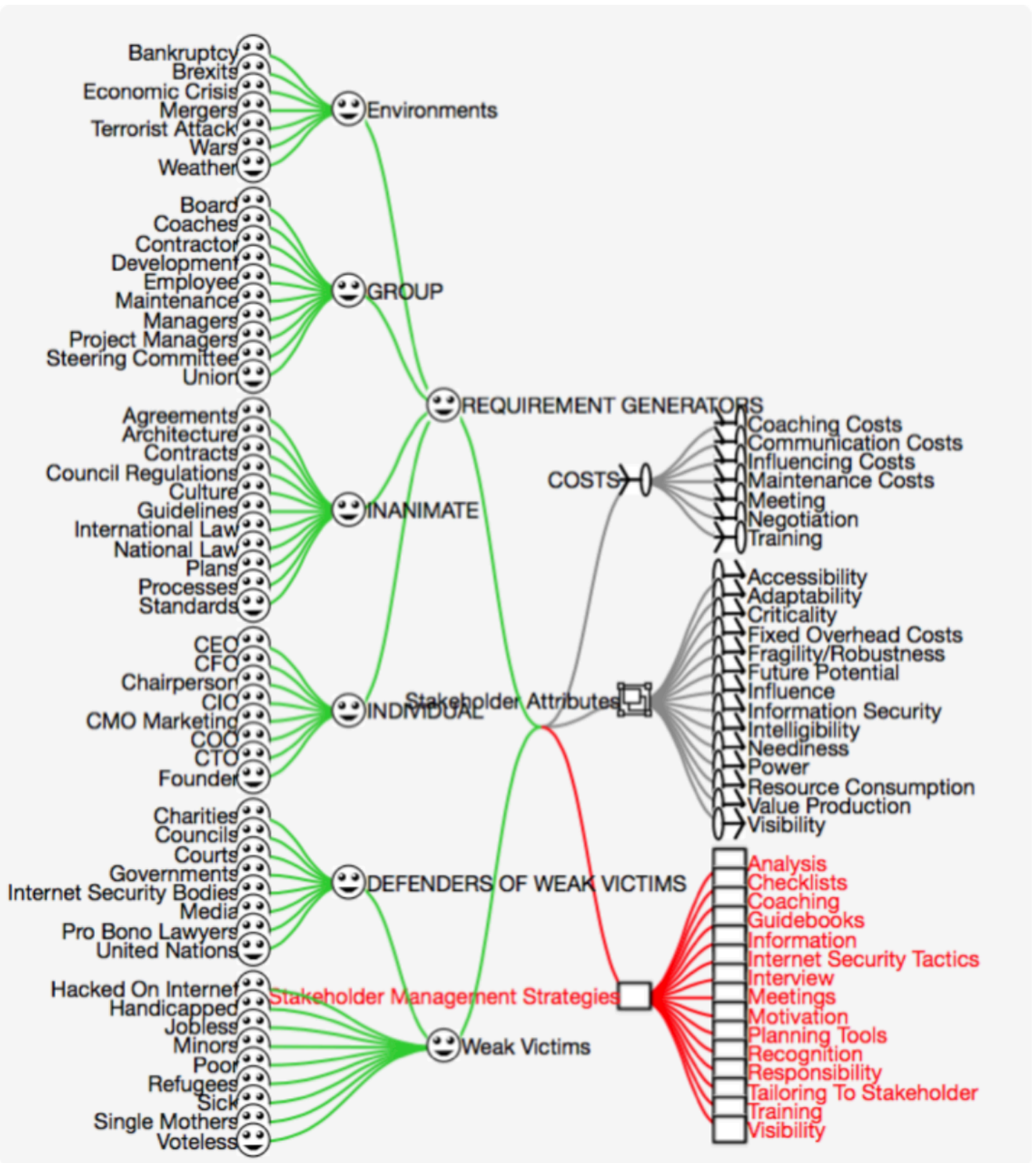


Figure 16.1 : overview of all top level factors in the system architecture. Stakeholders, Costs, Values, and Strategies (designs).

16.1 Architecture Engineering: disciplined and logical architecture.

Architecture, systems architecture, the architecture of any type of 'system' can be done in a variety of ways.

From the way we do our summer cabin, project by project, as we can afford it (the 8 new fjordview windows exchanged for the 1931 versions) , or when the water-well pump breaks down. To the way our neighbor Sven did recently by tearing down the 1931 cabin and commissioning a proper architect to design a rather modern building.

So incrementally, or all at once. You are still making value and cost judgements to satisfy a range of stakeholders.

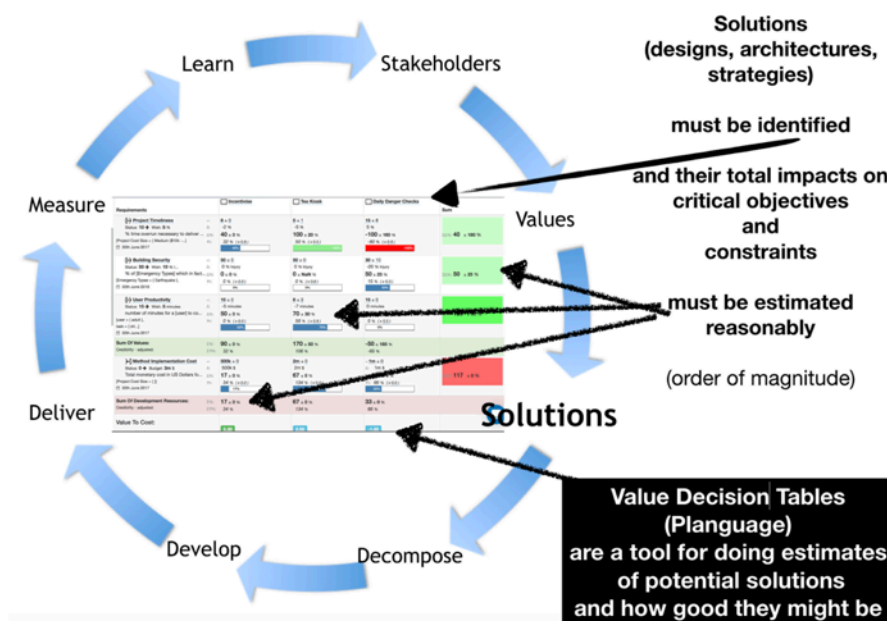
As some of the methods comments above indicate (Togaf, Zachmann, QFD and more) I have been dissatisfied with the unserious way some systems architecture has been done, particularly in the software and IT systems field.

And then I find similar problems in management planning, everywhere there, but the Balanced Scorecard and Business School teachings give evidence as to how bad management education is.

My core dissatisfaction in 'enterprise architecture' was complete lack of requirements discipline. *Talking* about quantified requirements, not actually *doing* it.

But my second related area of dissatisfaction is the almost complete lack of *systematic* (quantitative, all Values, all costs, all risks) evaluation of the subject matter itself, evaluation of the 'Architecture' (by whatever synonym for the *means to the ends*).

So for quite a few years, in Oslo and London I have offered a free course of 2 (practical exercises) and 3 days (theory lectures, case studies): called '**Architecture Engineering**'. Aimed directly at the IT Enterprise Architect group, many of whom have learned Togaf, Zachmann and similar methods. Maybe a book I will write soon, will have that title²⁰. And this book is Volume I, the foundation, for the 'how do we evaluate the actual architecture ideas?' methods.



25. THE PROCESS OF DESIGNING AND DELIVERING MULTIPLE VALUE-REQUIREMENTS: ENGINEERING VALUE. USING VALUE DECISION TABLES (VDT)

Source: ADVANCED AGILE SOFTWARE ENGINEERING Turkey April :

Figure 16.2 : Architecture Engineering is a very systematic quantified discipline, which relies on quantification of Value Requirements (topic of this book) as an input.

²⁰ | did write 'Value Design' in 2019 after Value Requirements.

Architecture processes give feedback to the Values specifications ('no architecture available for this..').

Then *architecture specification processes* get feedback from their implementation project ('the strategy/design doesn't work').

16.2 Design Engineering: specialist areas with clear consideration and balance for all other values and constraints.

In large complex systems, in systems engineering, there is naturally a set of specialist disciplines, contributing to the whole system.

This is good and necessary from the point of view of getting expertise into the system development.

But it always raises some problems:

- The specialist is not fully aware of all other concurrent specialists, nor of all stakeholders: not to mention *future* changes in both those areas
- the specialist can inadvertently make decisions which conflict with other equally-prioritized things, other values, other costs, other stakeholder feelings.

So, somehow we need to connect the many design-specialist decisions, and tentative decisions, with the *rest* of the systems effort.

This *synchronization* of specialist engineers, is in principle, the territory of architecture. But the architecture I see in practice, is totally incompetent to deal with these conflicts. They have not got the methods technology to deal with it. They do not deal with it.

So, something new has to be made applied.

Here are the conditions that a co-ordinated Design Engineering discipline needs to deal with, or fulfill.

- every budgeted *resource* aspect of a design specification, needs to be transmitted to the Project Engineering Model (PEM ?)
- This includes capital expenditure, operational expenditure, calendar time, Human Resources, technical debt, maintenance costs, and all such limited budgeted resources.
- if the resource consequences of one specialist-designer's design, potentially **threatens the available resources** of all the others, then feedback needs to be sent to that causing-designer immediately. ("You have eaten more than your fair share of our budget. Can you reduce maintenance costs by at least 2X by re-design?")
- If any design has a **side-effect** (for example 'if Security Design hurts Usability design') on any other's *Values territory*, especially if the effect is *negative*, it needs to immediately²¹ be reported to the responsible design specialist affected, and the Chief Systems Architect; so that it can be resolved.
- The specialist designers agree and understand their responsibility to estimate, and measure the side effects of their designs, with regard to **uncertainty and risk**.

²¹ after being estimated, or later when actually measured.

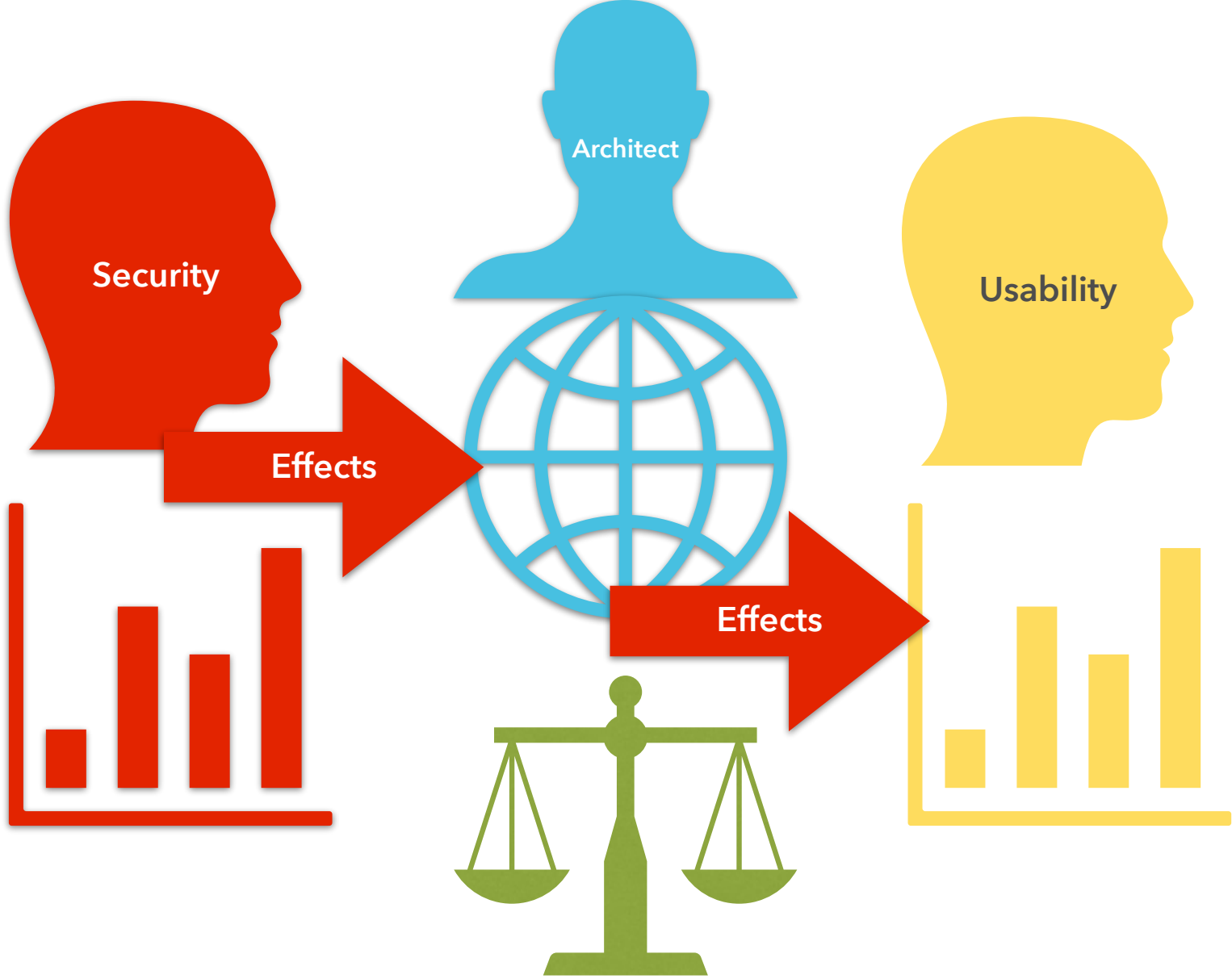


Figure 16.3 : Concurrent Value Engineering. Specialist design engineers must estimate and measure their *side effects*, and report and resolve them with affected parties.

16.3 The Value Table for overview and synchronization

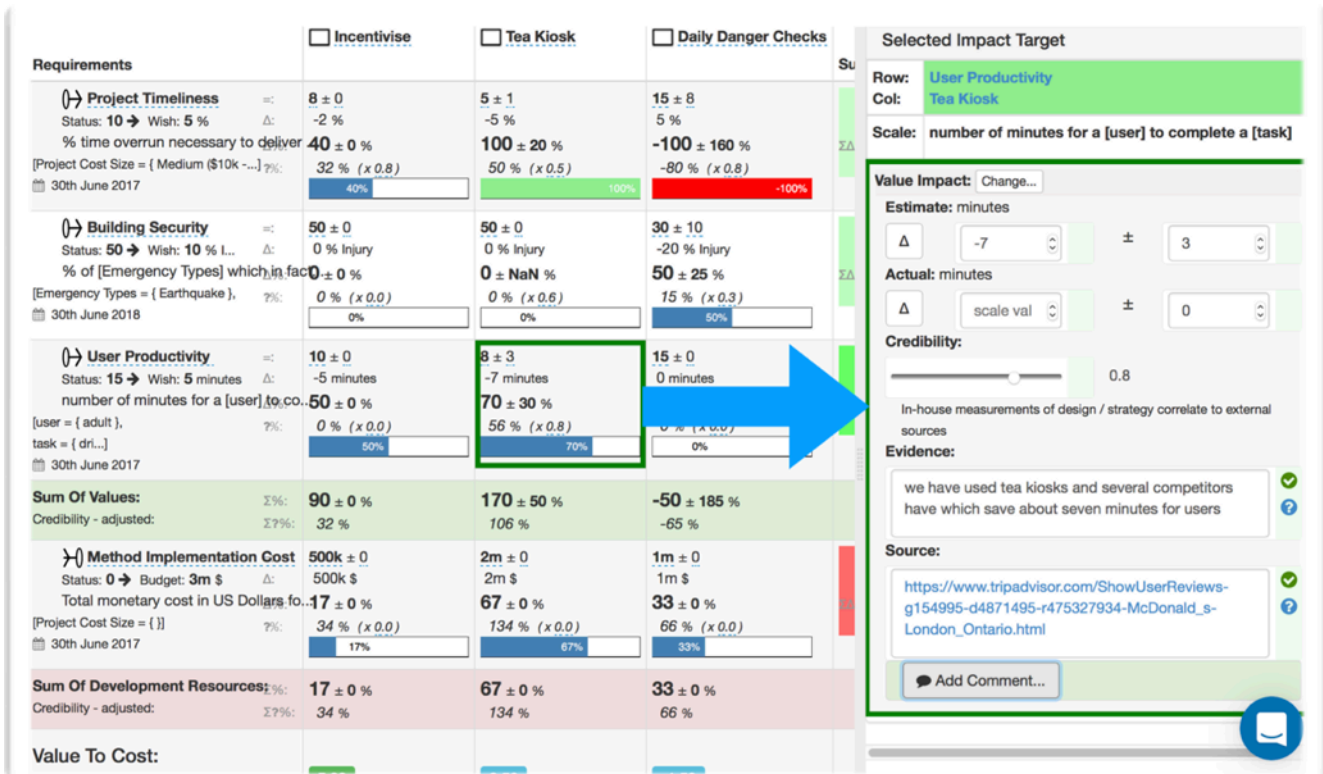


Figure 16.4 : We have developed a Value Table²² to enable us to both estimate, and track delivery values and costs. It can also collect risk data such as evidence, credibility, and uncertainty - for each individual estimate.

The Table can use any digital table or spreadsheet, and is here shown in the version contained in our tool ValPlan.net.

The Table can model any set of ends and means, including multiple levels of Values (Fundamental, Strategic, Means). It can model from a top-level overview and be decomposed, level by level, to any interesting level of decomposition and detail.

The Value Table enables genuine *concurrent engineering*

²² Also called Impact Estimation Table, IET, and Value Decision Table, VDT

16.4 Design-attribute feedback incrementally, and adjustment of value levels, timings, and resource budgets, in mid-project.

The IBM Cleanroom method (ref. d) and our own Evo method (ref. Books CE, VP) have been mentioned in passing earlier.

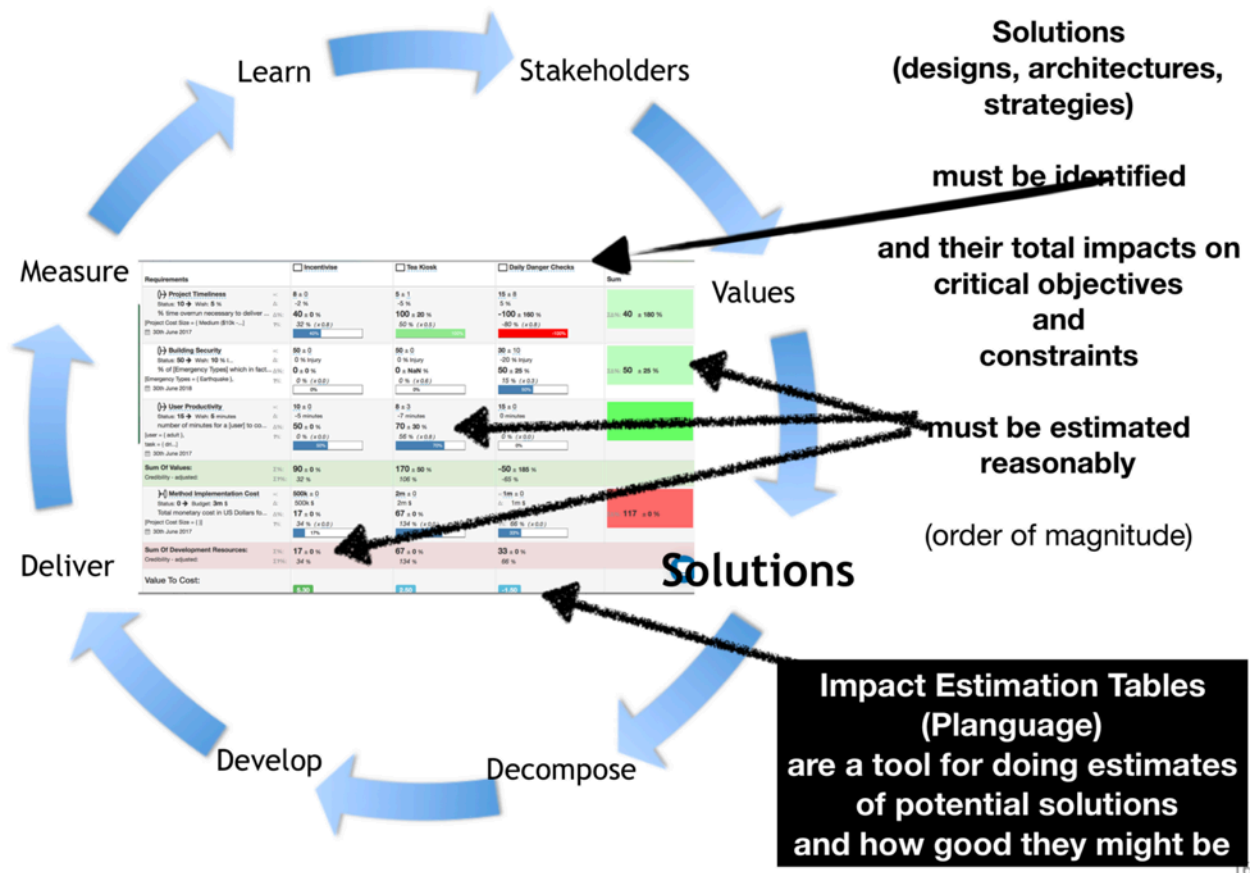


Figure 16.5 : the Gilb Evo Value Cycle. Measure, Compare, Learn, and if necessary re-design.

The Architect, or a specialist Design Engineer, needs to keep measurably tracking their design-ideas effects, when designs are incrementally implemented.

When necessary to reach planned Value levels, engineers or planners need to act and improve their designs.

16.5 The Evo Value Cycle

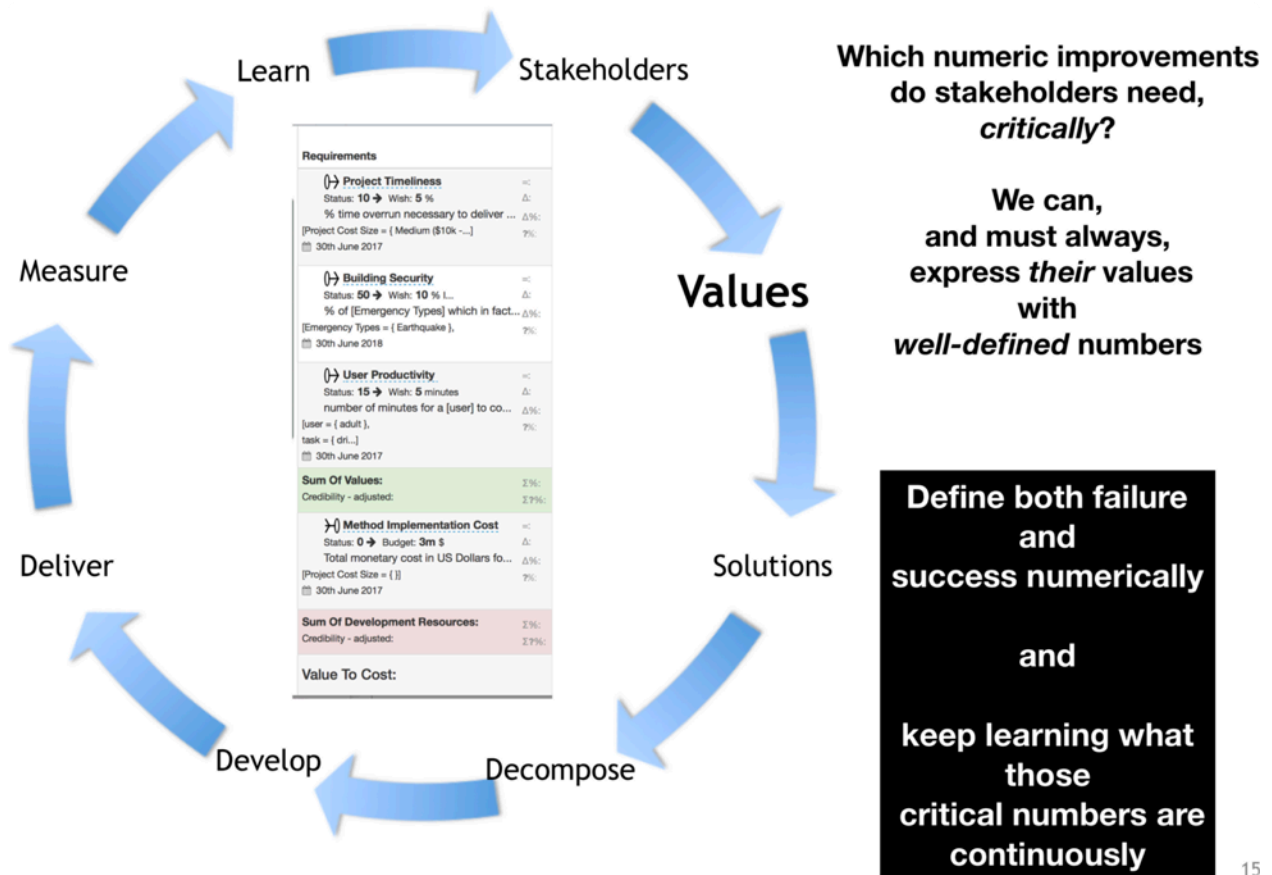


Figure 16.6 : The Evolutionary Value Optimization cycle. Evo, or EVO.

Our Evo Cycle is driven by the Value Requirements. It is essentially a Shewhart-Deming Plan Do Study Act cycle, with a few additional details, like Stakeholder needs analysis, Decompose Solutions, and Measure effects of solution delivery.

The core idea is that every part of the cycle is dealing with the Value Requirements.

Deming told me once that the PDSA cycle is eternal, as long as you have competition. Good insight. Projects have become processes.

You can also begin *anywhere* if the cycle is *short*.

Chapter 17. Formal Standards for Value Specifications

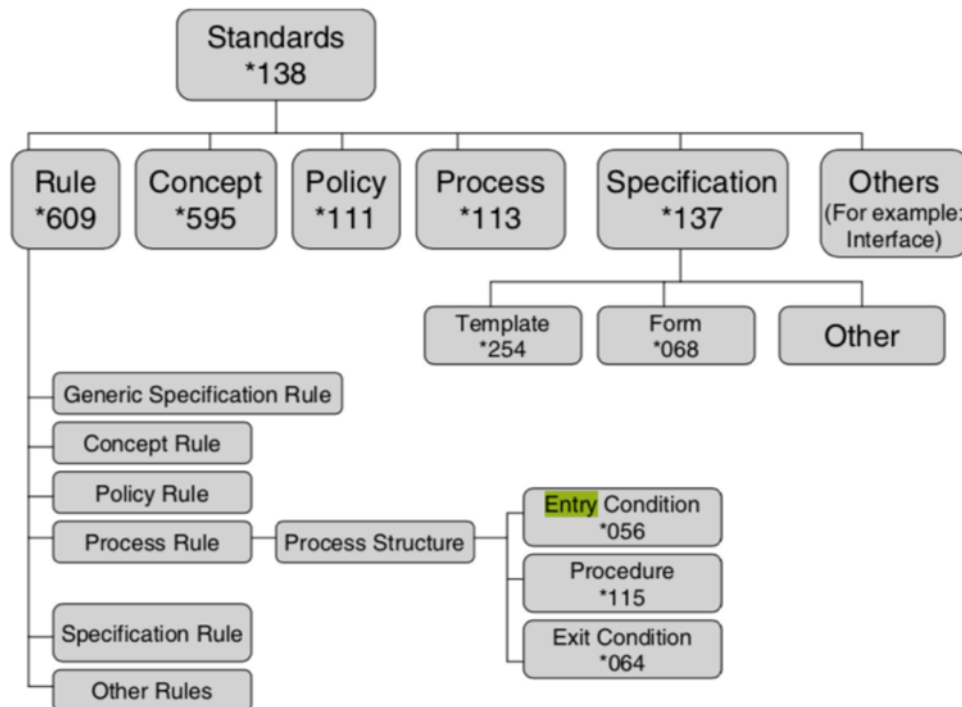


Figure 17.1 : Language Standards categories. Provide us with specified best practices.

Many of these have been discussed earlier in this book. The rest are detailed in Competitive Engineering (CE).

17.1 Competitive Engineering as Planguage Standard.



<https://www.gilb.com/competitive-engineering>

Use the above link for a free digital copy and signup on our website.

Paper copies available from:

<https://www.amazon.com/Competitive-Engineering-Handbook-Requirements-Planguage/dp/0750665076>

Figure 17.2 CE Cover.

Competitive Engineering, the book, is the formal set of standards defining Planguage, as well as associated processes.

Anybody can freely adopt these standards, or can modify them to suit their purposes. They are freeware.

17.2 ValPlan as a tool containing standards

www.valplan.net

ValPlan is an app based on Planguage and on the Competitive Engineering book.

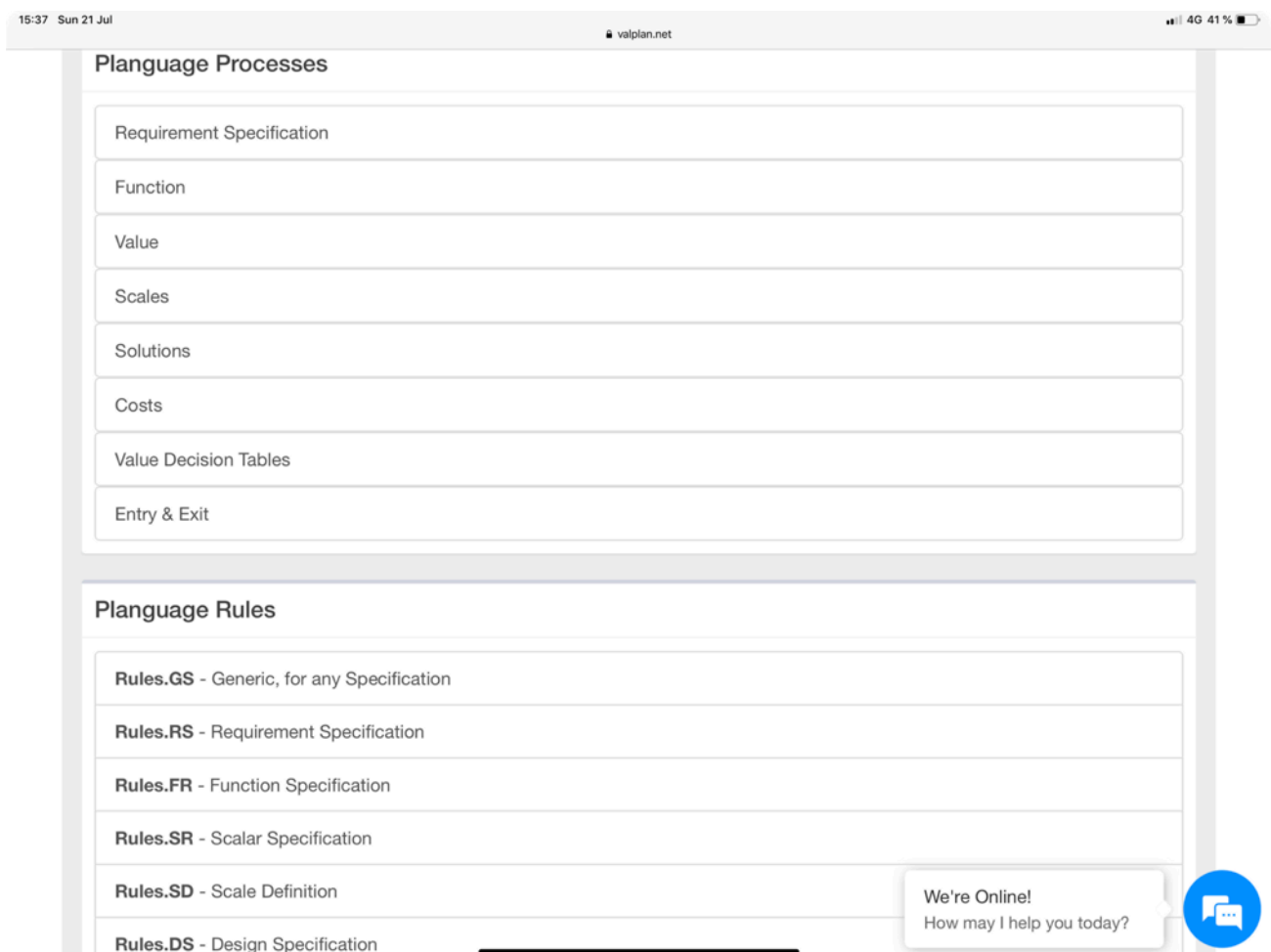


Figure 17.3 : Some of the processes and rules from CE book which are made available in the ValPlan tool.

A wide range of the Planguage/CE standards are built in to ValPlan.

In addition to rendering standards, such as the Glossary, Scales, Processes, and Rules, ValPlan is programmed to understand all the essential rules of Planguage, for example:

- all scalar level specs (Past, Wish, Tolerable etc.) belong to the one Scale defined in their local specification object (a requirement)
- [Scale Parameters] in a Scale demand definition, and these definitions can be used in scalar level specs (like Wish)
- Tags are essentially Unique, and allow reuse of, and reference to, the tagged spec.
- A specified benchmark and requirement level pair (like Past and Tolerable) are equal to the 0% impact level of a design, and the 100% level of a design, within the requirement level deadline. The App knows this and computes it.

17.3 Rules

Language **Rules for Writing** define good practice, and by definition, if not followed, Rules define 'bad practice' ie 'specification defects' in Spec Quality Control.

The Rules are divided into two main categories:

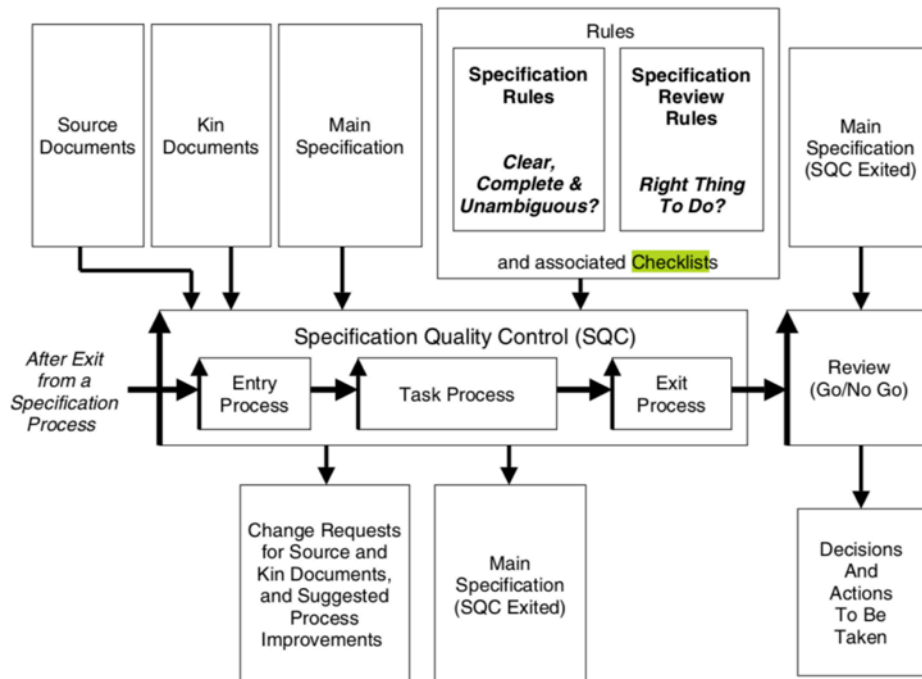
Generic Rules which apply to a broad range of different types of specs (like all requirements, or all designs): and **Specific Rules** which apply to a narrower range of specs such as Function Requirements, or Value Requirements.

The Generic Rules can be reused and learned better.

The Specific Rules can home in on details specific to a narrow class of specification, such as a Scale of Measure.

17.4 Processes

Process descriptions advise how to do something, and a recommended sequence of actions to do it.



**63. AN OVERVIEW OF PROCESS COMPONENTS IN THE SPEC QC PROCESS
'SOURCE DOCUMENTS', UPPER RIGHT CORNER**

Figure 17.4 : A process map. The Planguage drawn icon for a process is a rectangle, with an upwards (clockwise) arrow on the left side.

The Planguage convention is that inputs (documentation, standards) enter from the top-side. And process outputs exit from the bottom-side.

Upstream (previous) processes enter from the left-side, and exit to the *downstream*, or *next* process is from the right side.

17.5 Exit and Entry Processes

Entry processes are designed to protect the next, and larger, main process from being wasted, or from giving false results (avoid Garbage In Garbage Out: GIGO).

Entry processes check a list of one or more Entry Process **Conditions** (in themselves standards, for process entry).

These Entry Condition questions are trying to make sure that:

- the input documentation is itself of sufficiently high, trustworthy, defect-free quality (it has met the Exit level defect density standard from the previous process)
- That all related documentation is available in practice, updated and (approved, exited defect-level)
- That all related standards are made available, and will be used, including checklists and less-formal standards, templates, guidelines and tools.
- That people, especially process leaders, are trained and qualified for the main task (procedure) itself.

Here is the Defect Estimation Process, including the SQC sub-process in detail:

Entry Conditions:

1. A group of two, or more, suitable people, peers, are needed to do it.
2. You need to schedule 30 to 60 minutes. Irrespective of plan size.
3. You should have a trained (1/2 day) SQC team leader at the meeting to manage the process.

(or if that fails, follow this Process 10.5 and the Template 10.5 instructions, below.

Defect Estimation Procedure:

P1: Identify Checkers: Two people, maybe more, should be identified to carry out the checking. One should be the Owner or Author who drafted or updated the plans to be checked.

P2: Select Rules: The team identifies (or acknowledges a standard) at least three Rules to use for checking the specification. (My favorite Rules are clarity ('clear enough to test'), unambiguous ('to the intended readership'), and completeness ('compared to sources'). For 'requirements; I also use the Rule of 'no design'. Any appropriate set of rules can be used.

P3: Choose Sample(s): The team then selects sample(s) of about one 'page' in length (300 non-commentary words). Choosing a page at random can add credibility – so long as it is reasonably representative of the overall content. The team should decide whether all the checkers should use the same sample, or whether different samples for some people are more appropriate or useful. If 300-600 words or less do it all.

P4: Instruct Checkers: The team leader briefly instructs the checker(s) about how to use or interpret the Rules, the time to be used checking (10 to 30 minutes), and how to document, list, count - any perceived defects, and how to determine if they are major defects (majors).

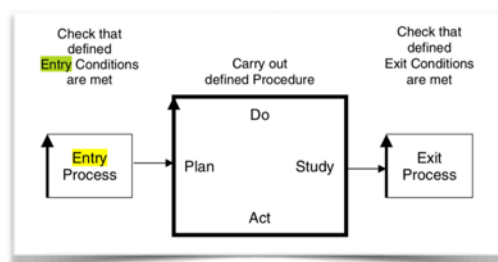


Figure 17.5 : Entry Process and 3 Entry Process Conditions.

If any single Entry Condition fails, then the standard is that we are not allowed to enter the following process. If we do, under pressure, we should be personally held responsible for any consequent delay or damage.

The problem there, is that the damage might only show up far downstream, and the Short-Cut Cheaters will not really be blamed.

So process management (designers of the process and Entry conditions) needs to make sure these Entry conditions are really worthwhile. And local project management needs to be accountable for people following the Entry process conditions, on faith!

Exit Process conditions are trying to make sure we did a good enough job on the process: complete, correct, following process standards, and specification rules. The most critical Exit process condition is usually that the Defect Density, as measured by a quick sample and Spec QC process, is acceptably low; meaning it pays off to exit, and to proceed downstream.

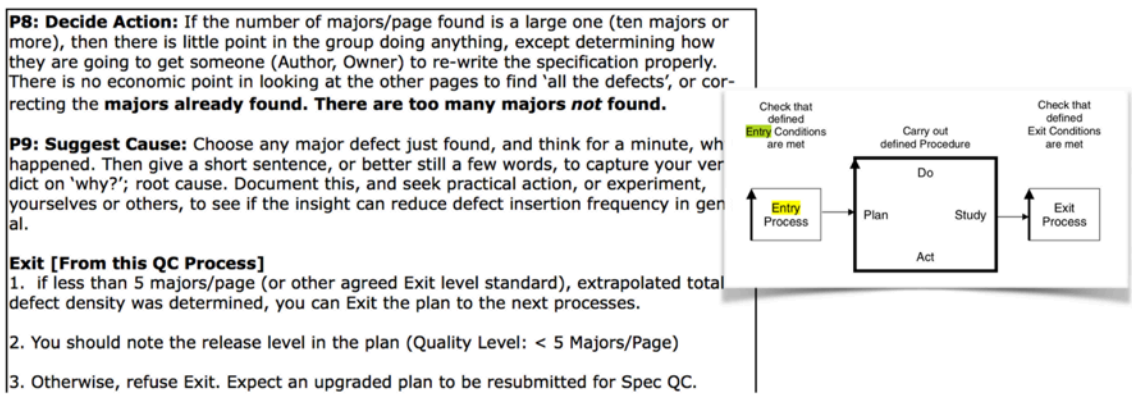
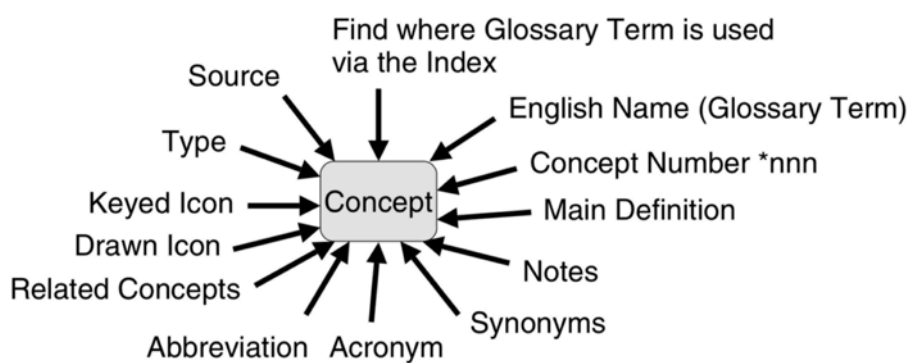


Figure 17.6 : 3 exit conditions from Spec QC process.

17.6 'Concept' Standards: several levels

Planguage has a basic Glossary of several hundred concepts, like *Scale, Goal, Design Idea, Function*. See *short Glossary at end of this book*.

An organization is free to modify this in any useful way, for their needs. As all of my clients do in practice.



26.
CONCEPTS CAN BE REFERENCED BY A WIDE VARIETY OF TERMS AND SYMBOLS. THIS SOLVES THE 'DICTIONARY PROBLEM' OF PEOPLE DIFFERING ON 'WHAT THEY MEAN BY A WORD'. IS ALSO OPENS UP FOR OTHER LANGUAGES TERMS, 'POINTING TO A CONCEPT DEFINITION'. AND IT ALLOWS GRAPHICAL SYMBOLS TO POINT DIRECTLY TO A CONCEPT DEFINITION, RATHER THAN TO A TERM, WHICH WILL HAVE 'VARIOUS' INTERPRETATIONS.

Figure 17.7 : in Planguage the 'Concept Itself' is central. Quite different from a dictionary. Figure source CE Glossary, Text from part 26 of Technoscopes book (2018)

We can *refer* to the concept, by any useful pointers, including 'logical implication in context'. For example if I use a Goal specification, I am then unconditionally pointing to the defined Scale in the same Requirement Specification Object, for the correct interpretation of the level number in the Goal statement.

There is a need for allowing *synonyms* to point to concept definitions. So the concept itself is stable, and well-defined. But useful synonyms are specified. We need to make sure they are not duplicated (i.e. same term does NOT point to 2 or more concepts).

There are other levels of concept definition, which are used locally in a specification. For example:

The screenshot shows a software interface for an 'Air Quality Index'. At the top, there is a title bar 'Air Quality Index'. Below it, the 'Level' is 'Stakeholder', 'Type' is 'Value', and 'Labels' is '-'. There is an 'Edit' button. Below that, it says 'Is Part Of: > Top Values' with a 'Show' button. A horizontal timeline shows a sequence of values: Past (135), Wish (67), Status (157), Wish (88), Status (71), Wish (33), and Status (38). The current value is 'Wish [Pollutant = Ozone, Area = Greater London] @ 2021 : 33 µg/m³'. Below the timeline, the 'Ambition Level' is 'A vast improvement in Air Quality for London within 3 years'. The 'Tag.Scale' section shows a definition: 'Maximum µg/m³ for defined [Pollutant] within defined [Area]'. It includes a 'Templates' dropdown, 'Area' defined as 'Greater London. Within M25', and 'Pollutant' defined as 'Nitrogen Dioxide, Sulphur Dioxide, Ozone, Particles'. There is a 'Target Time Units' section with an 'Advanced...' link.

Figure 17.8: Four Levels of definition types:

'Scale' is a normal Planguage-defined term. 'Pollutant' is *defined right here* locally in the 'Air Quality Index' specification object. 'Area' was defined in a Project Glossary, and reused here. The Project Glossary *could* have been raised to the level of an Organization-Wide Glossary; available to all projects. That would be a 5th level of definition spaces.

17.7 Teaching and enforcing standards using Spec QC and Exit levels

My experience is clear. It is not enough to define good practices. We need to continuously measure that they are carried out. We need to set high exit-level standards, with respect to clear effective Rules of practice (standards for writing, planning). Management must clearly enforce these standards: because they understand that the economics for doing so are overwhelming.

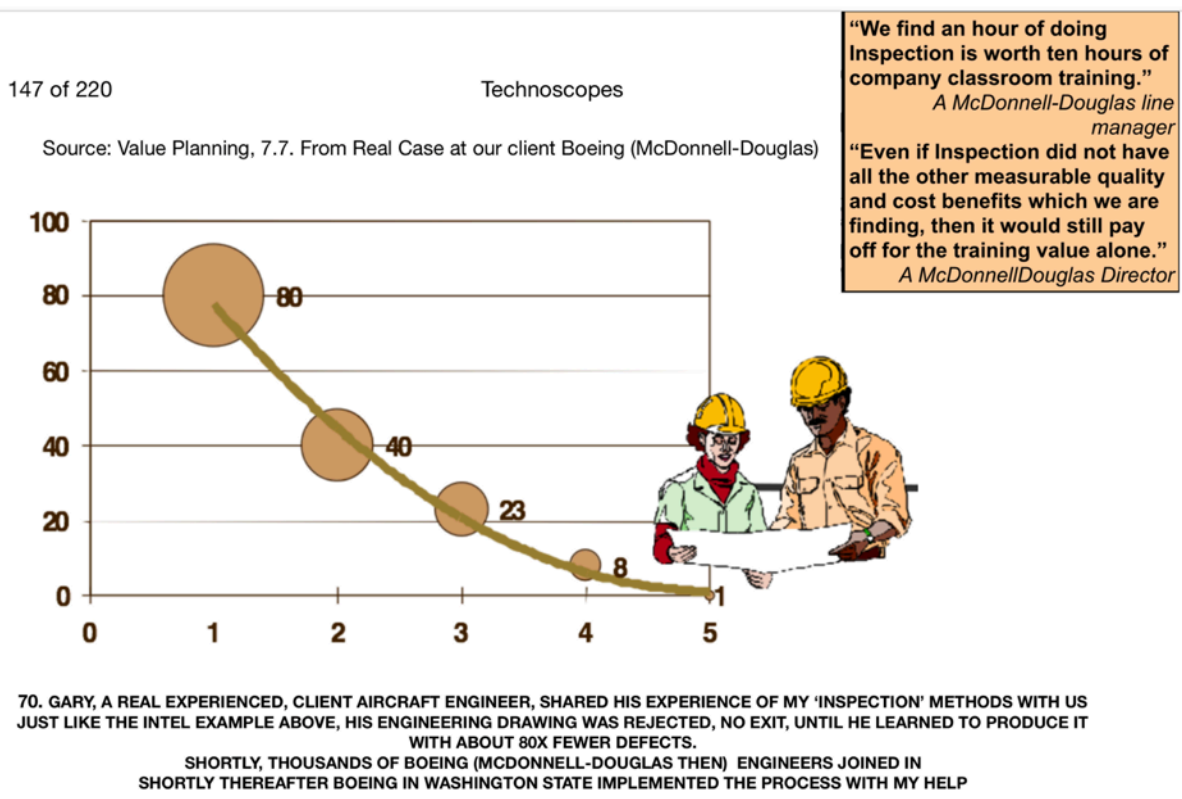


Figure 17.9 : Spec

QC (called 'Inspection' then) together with Exit levels is a dramatic way to really change large-scale organizational culture, to follow improved practices on an everyday basis.

We saw this effect earlier in this book at Intel (Terzakis, ref. A). My experience at two Aircraft Factories was a clear two-orders-of-magnitude (!) measurable improvement in weeks and months.

Chapter 18. ValPlan: and apps for Value Specifications

18.1 Brief history of Planguage tools.

Planguage was designed with a view towards automation.

Aspect Engine

In the late 70s Lech Krzanik, spent Summers in Norway, and built the Aspect Engine. Which he used for his PhD.

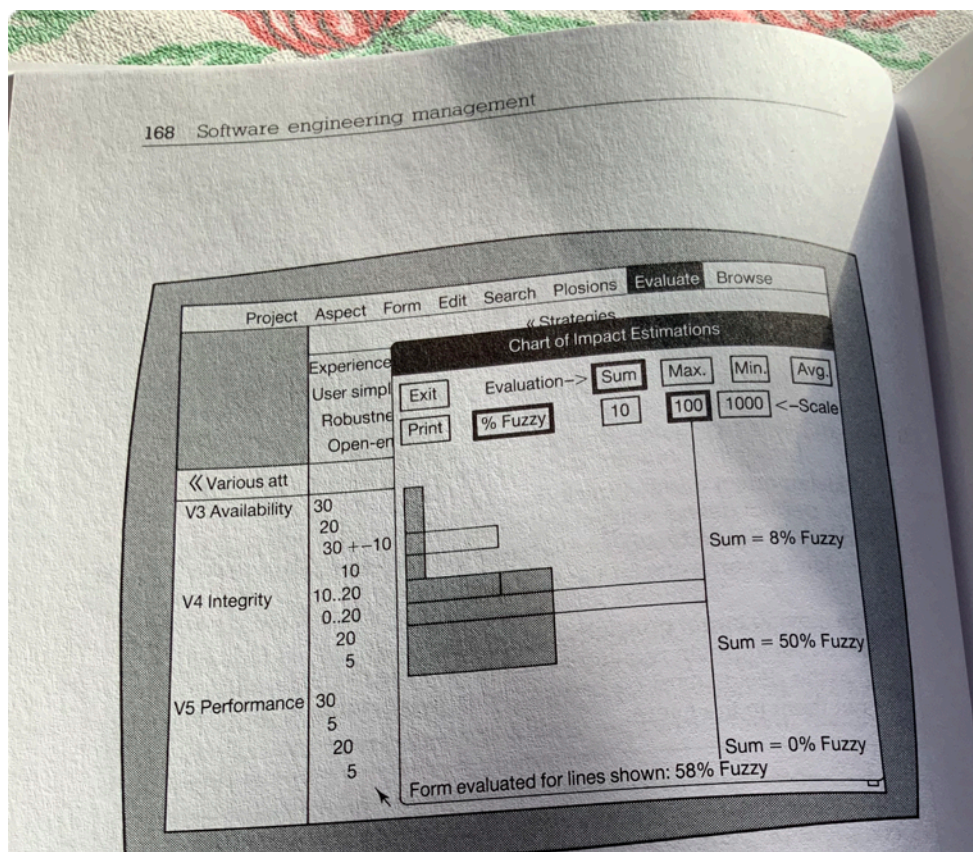


Figure 10.1 Automating the search for solutions. The figure shows a screen image from a prototype system for automated software engineering, called the ASPECT Engine, by Lech Krzanik, of Krakow, Poland. It can automatically search for solutions which match its attribute requirements and evaluate them multi-dimensionally.

V3, V4 and V5 represent various major attribute requirements and their sub-attributes defined in detail elsewhere. On the top, strategies such as 'experience' represent tags for suggested solutions to the required planned level of attributes. The 100% planned level of all attributes is represented by the vertical line down from the box with 100 in it. The bars represent the estimated impact of the selected and defined strategies on the required attributes. The hollow bars are the estimated impact (100% for integrity) and the shaded bars are the estimated degree of uncertainty (50% for integrity).

Figure 18.1 : The Aspect Engine. Written in 'Forth', on my Apple II. About 1979. Source 'Principles of Software Engineering Management' (1988)

The Aspect Engine had two Phases, the Ambition Level Value Requirements (like 'much higher Security') were automatically converted to a suitable Scale of measure (for Security), and a Goal level (the 'much higher' part).

These quantified Value Requirements were then fed into a database of known technologies (like passwords, encryption), with known attributes, and selected for best match to the numeric requirement.

Our database was small, symbolic, but we demonstrated that it worked. An early 'AI' system for automated requirements and design.

Kai Gilb's Spreadsheet Tool

From the mid 1990s until about 2015 Kai Gilb built a tool using Excel spreadsheets. It worked pretty well for real projects and for training courses. But there are limits to what you can program with Excel.

Confirmit.

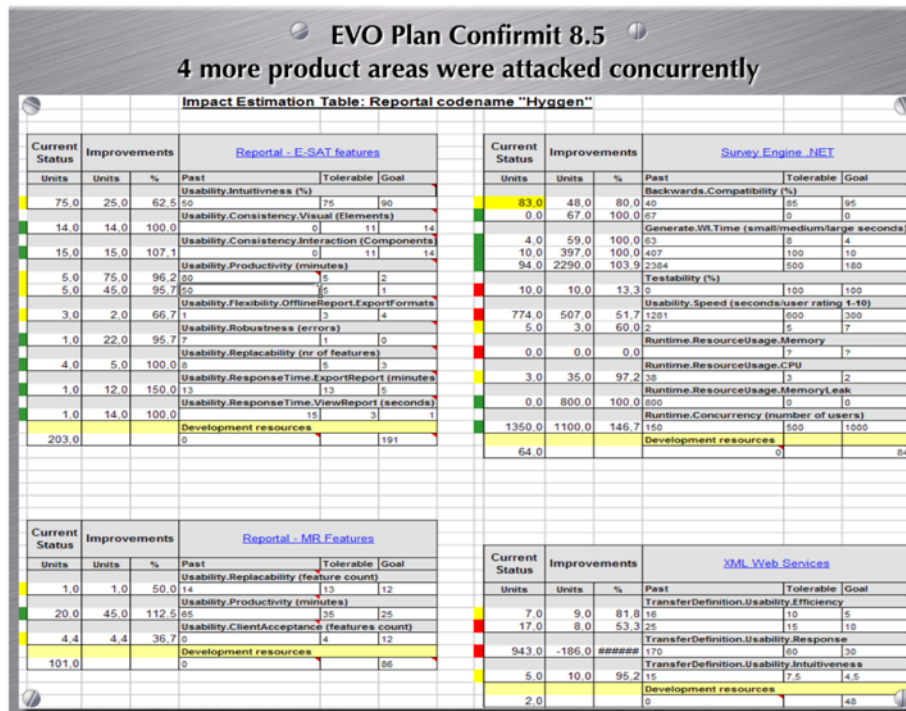


FIGURE 87. NOTICE THE 'IMPROVEMENT % COLUMN. 100% MEANS WE DELIVERED THE GOAL LEVEL, DUE 12 WEEKS FROM START OF THE QUARTER. THIS IS 9 WEEKS INTO THE 12 WEEKS. HOW 'DONE' IS THIS PROJECT? 3 WEEKS AHEAD OF THE DEADLINE. NOTICE THE TRAFFIC LIGHT SIGNALS: RED = NOT EVEN TOLERABLE, GREEN = DONE

Source: Trond Johansen, [Confirmit Public Presentation](#).

Figure 18.2 : simple spread-

sheet automation of Planguage, 2003. By 2005 they had invested in a more sophisticated design.

This simple tool gave Confirmit weekly numeric feedback on 25 Value Requirements, for 4 teams. The teams were free to use the feedback to do 'Dynamic Design to Cost and Value' (discussed in this book earlier).

They delivered amazing (1500% improvements, is that amazing?) multiple quality improvements to their market on a quarterly basis, and defeated their international competitors because of the quality differences. Smart engineers + good method.

18.2 ValPlan the app.

We have seen many examples of ValPlan throughout this book.

This is the brainchild of Richard Smith in UK (<http://rsbatechnology.co.uk/blog:8>) who built it on the side of his freelance consultant work in London.

Richard went on our training courses at Citigroup in 2003, and successfully practiced Planguage, and Evo after training at Citi.



Figure 18.3 : June 2019 at British Museum. Gilb International presents Richard Smith with a Maestro Award for extraordinary achievement using Planguage, with his app. P.S. yes that is USC Professor Dr. Barry Bohem, applauding in the white shirt.

Richard spent about 4 years of hard and brilliant work building the tool. We used the name 'NeedsandMeans.com'

It was in detail, based on the Competitive Engineering book's Planguage standards.

We Beta tested it with clients and classes.

Finally 2019 Gilb International took over the Marketing rights. And we re-christened it ValPlan.net

18.3 Open Endedness for Tool Builders.

We (Gilb International AS) are quite happy for any tool builder to make use of Planguage ideas, without license or permission, in part or whole. This is the purpose of publication, and training courses, to share the ideas for any good human purposes.

It is nice if we are credited, and we won't sue you if you 'forget'.

We would love for you to share your ideas and experiences on the internet with us, and to give us a *heads up* that you are doing it. We may be able to connect you with other people and ideas.

We believe in open competition to bring out the best ideas for us all.

Right now we are a very small group working to change a world-wide planning culture which does *not really* focus on 'Value First'.

We hope you are interested in joining forces with us: for both your own personal benefit, and we hope you are like us *Idealistic* and want to help others get better Value, '

just because it makes your life more 'Valuable'.

Make the world a more valuable place

18.4 GraphMetrix

In late 2015, Frederick C. Gibson, of the large building corporation Bechtel, contacted me. He is an architect, a proper one. He said he had read Competitive Engineering three times, and was using it as a framework for advanced building-planning tools at Bechtel.

What he had published and demonstrated to me was impressive.

Not too much later, he is running a startup, to further the ideas

[Http://graphmetrix.com/web/](http://graphmetrix.com/web/)

We are involved.



FIGURE 18.4 : FROM GRAPHMETRIX WEBSITE

The tool is at an advanced state of development. Funding is in place. And the advanced ambition levels (AI and all that) are in place.

I'd tell you more, but they are keeping it under wraps for competitive reasons. Keep an eye on the website, or ours, for developments. I'll update here when I can.

Chapter 19. Stakeholders and the planning environment



Figure 19.1 : Stakeholders, or 'Requirement Sources', which might be a more general description, are the Value Sources, the entities that need, demand, command, and desire.

Stakeholder Attributes

Stakeholders have a set of attributes, which determine their influence, power and priority of delivering their Values.

- Financial Power
- Legal Power
- Knowledge Power
- Network Influence

- Approval Power
- Political Power
- Persuasive Power
- Respectedness
- Veto Power
- Cultural Respect

Stakeholders are behind all our Value Requirements, whether we recognize the fact or not.

The better we get to know our stakeholders, the better we can determine our Value Requirements.

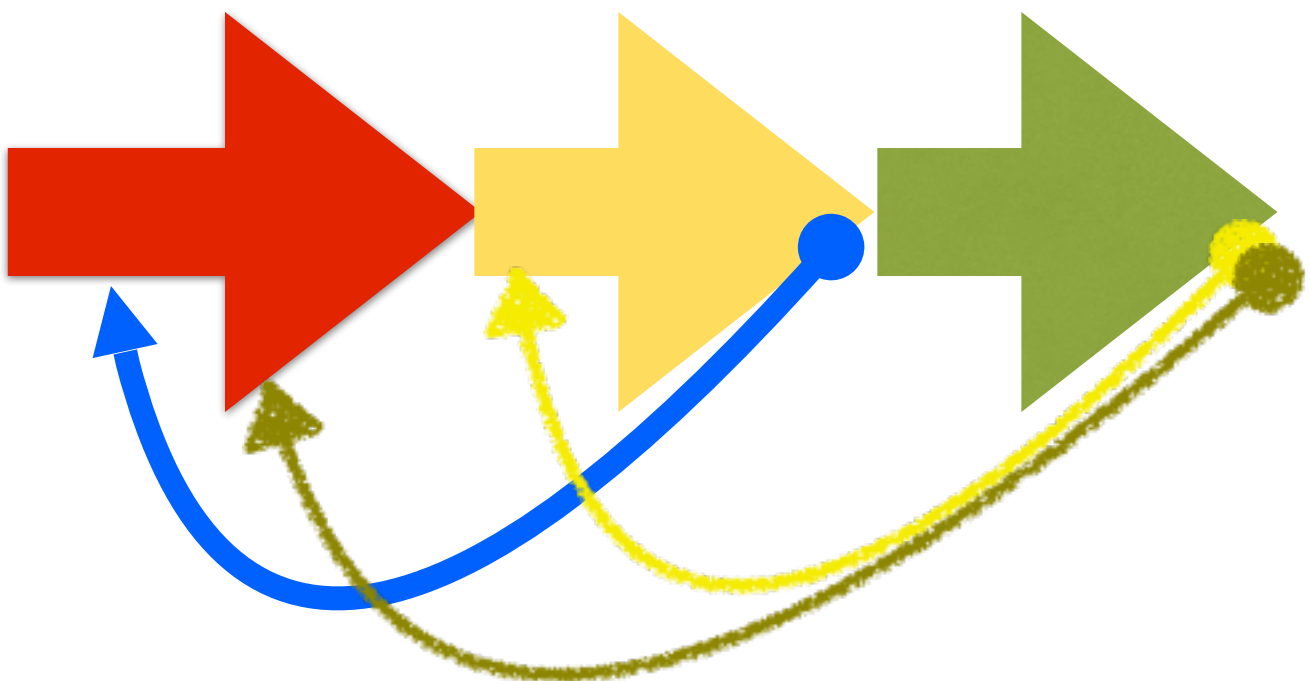
The more likely our projects are to succeed; and to avoid failure.

19.1 Determining your planning environment (ref. L)

Before we can determine which stakeholders are critical, and have got some 'critical requirements' for us, we need to define our *systems* environment.

As Pawel Nowak's ideas make clear (ref. L) there is no perfectly clear line we can draw around our environment. There will always be things outside of it, which just might be critical.

The best we can do, is to improve the *probability* that we identify most critical stakeholders, and capture most of their critical requirements, initially.



There will always be a learning process, where we realize that our environment is larger than we thought it was, when we realize there are critical stakeholders we did not see - or which recently

emerged, and where critical requirements get recognized unpleasantly late.

It will probably be more cost-effective to draft a good starting analysis to the environment, the stakeholders and their requirements: but to have a fairly-humble, but effective process of picking up additional candidates (stakeholders and values), as we incrementally delivery value, and get feedback from doing so.

Here is a guess at a principle:

Your *interesting* systems environment

contains all critical stakeholders.

Another useful tool is a good definition of 'stakeholders', which contains the **non-human** stakeholders (laws, plans, policies), and which also recognizes that **not** all stakeholders are **nice**, and certainly that not all stakeholders are *users* and *customers*.

One effective checklist, is this *generic stakeholder set* (figure below). It indirectly reminds us that all these **types** of stakeholders are part of our Systems Environment. By using those insights we might be able to identify even more of our real and critical stakeholders.

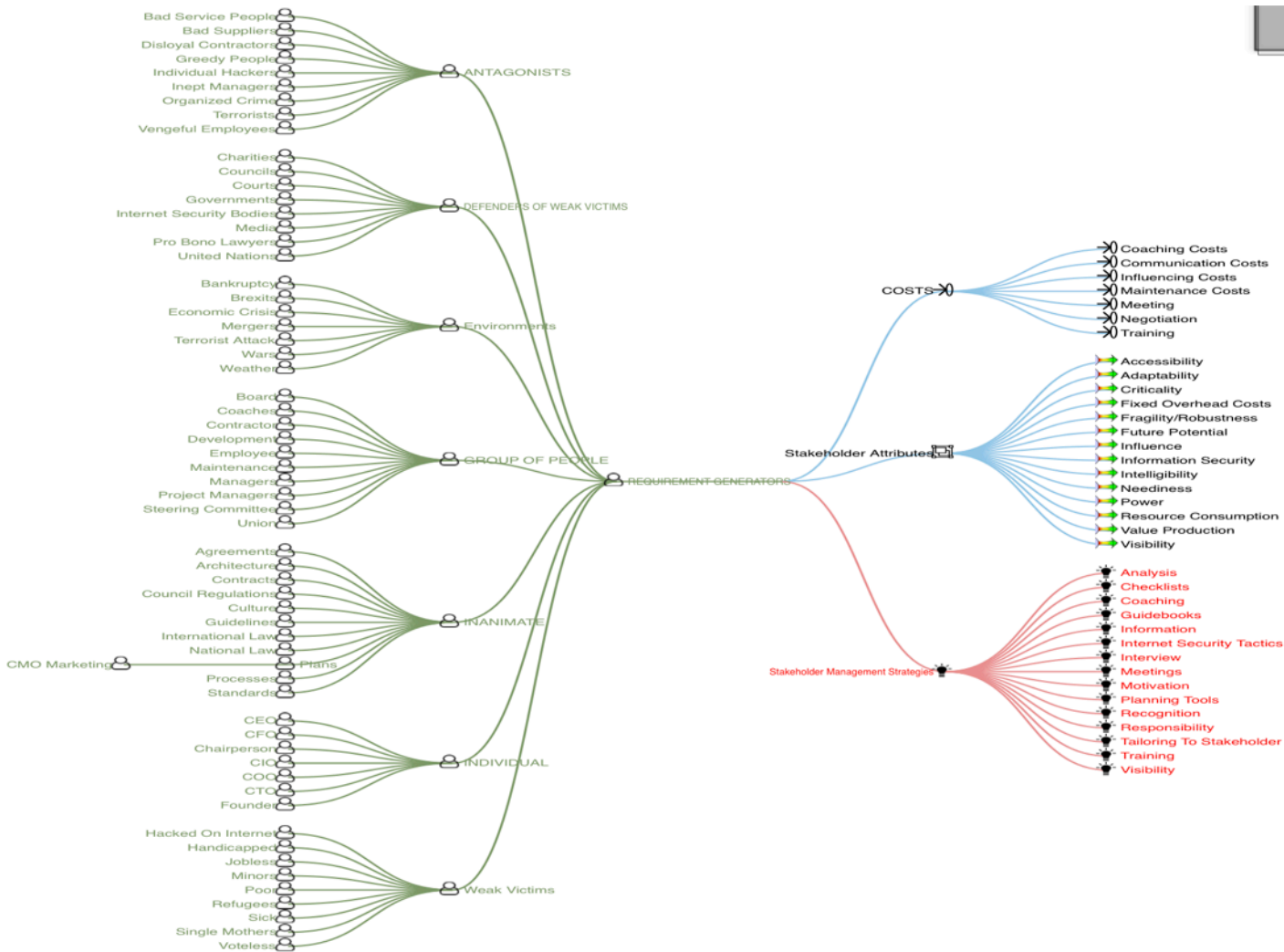


Figure 19.2: Note the seven major categories of stakeholder. Source: Tom Gilb's Stakeholder Analysis in ValPlan 2019.²³

Then note the handful of examples for each major category. These help you map your environment. This is not *your* systems list. This is just a *general* checklist to help you find your *own* systems list of stakeholders. Analyze and tailor to *your* particular environment, for best effect.

²³ 'Antagonist' category suggested by Petter Abrahamsen.

The costs, the value attributes, and the strategies at the right side of the above diagram, might also be additional ways of defining your particular environment in more depth.

For example a strategy like 'Responsibility' should trigger you to think of environments, and stakeholders associated with it, like *authority, culture, measurement*.

19.2 Determining your stakeholders

The most important reason we analyze *environments*, is to discover critical *stakeholders*. The reason we are interested in critical stakeholders is because they, by definition, have critical *requirements*. The reason we are interested in critical requirements is that they determine our *success* or failure.

Notice that concept **critical**?

Critical means, that it can probably *alone* decide whether your entire effort *fails* or *succeeds*.

Critical is a prioritization tool. You have to deal with all the critical stuff before you can spend any effort on the non-critical, or less critical stuff.

When you have identified a possibly, or probably, critical stakeholder, you can begin to analyze and identify, and perhaps *detail* their requirements.

Only when you have detailed a requirement can you actually decide that it is critical.

You cannot just say 'Security' is critical. You have to know the *level* of security (1%, 42%, 99.98% ?), the value of the security level (kills the business, can lose a small sale), the many Scale Parameter conditions (User Type, Task, External Conditions, Geographical Area) before you can say for sure "it is critical". That detailed Value requirement definition is the core of this book.

At some early point keep 2 lists: one for Stakeholders, one for Value Requirements, with 10 items (critical top 10) at the top of the list, and everything else below the line. Then in the bottom section: "PROBABLY NOT CRITICAL" Use yellow stickies if you must! (ref. B)

It does not hurt to make as complete lists as you can, note everything, forget them not. Some items will move up or down the lists as new items emerge, and new voices argue the case.

I would personally do this digitally, like in ValPlan. There is a facility there, exactly for keeping track of the critical top ten.

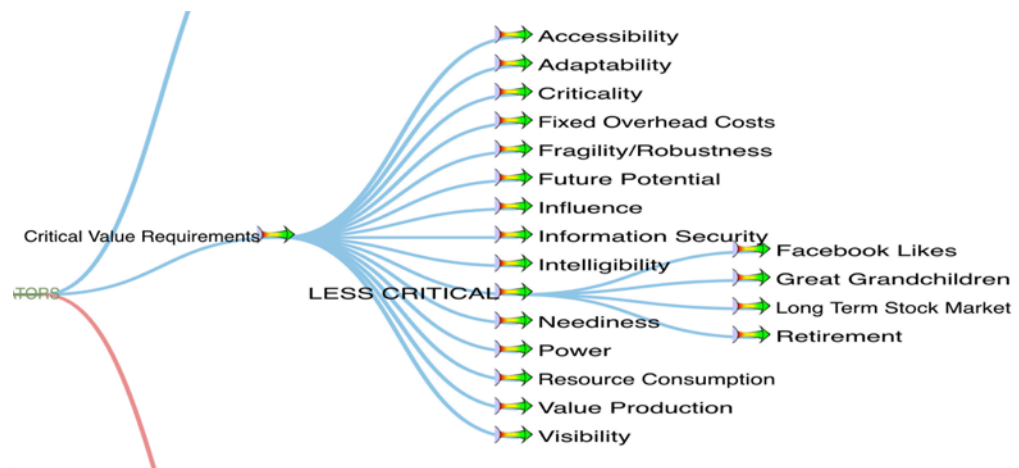


Figure 19.3 : one simple digital way to keep track of anything into Critical/Less-Critical Categories is to show the Less Critical as a separate hierarchy. It documents that they are considered at all.

But at this early stage of brainstorming, paper and writing instruments are fine.

Critical Project Objectives Copy Of Tom Gilb's STAKEHOLDER ANALYSIS

Tag: [Accessibility](#)

Ambition Level: we want to access the stakeholder insights, opinions and needs as soon as possible, same day would be great <- tag
Scale: Days from defined [Need] by a type of [Stakeholder] until we have a defined [Information] correct to a defined [Place]
Status [Need = <All>, Stakeholder = **Critical**, Information = **Changed Stakeholder Authority**, Place = {**Digital Planning System, Email to Spec Owner**}] @ 24 Jun 2017 : 7 Days to Get Info <- tsg
Wish [Need = <All>, Stakeholder = **Critical**, Information = **Changed Stakeholder Authority**, Place = {**Digital Planning System, Email to Spec Owner**}] @ 24 Jun 2017 : 1 Days to Get Info <- tag

Tag: [Criticality](#)

Ambition Level: Avoid failure caused by failing to serve critical stakeholders <- tsg just a draft
Scale: % probability of failing to satisfy a [Stakeholder Type] and [Stakeholder Class] with a particular [Value]
Status [Stakeholder Type = <All>, Stakeholder Class = <All>, Value = <All>] @ 28 Jun 2017 : 40 % Failure <- tsg
Wish [Stakeholder Type = <All>, Stakeholder Class = <All>, Value = <All>] @ 28 Jun 2019 : 1 % Failure
Stakeholders: Architecture, CTO, Project Managers,

Tag: [Fragility/Robustness](#)

Ambition Level:
Scale:
Status @ ? : 0
Wish @ ? : 0



ValPlan | Gilb International AS | [Terms Of Use](#) | [Privacy Policy](#) | [Cookie F](#)
Powered by Needs & Means © RSBA Technology Ltd 2015-2019

We're Online!
How may I help you today?



Figure 19.4 : Three of the requirements were Labeled 'Critical' and so we are able to keep track of them on this special Critical report.

19.3 Eliciting stakeholder needs and converting them into Requirements.

How do we identify potentially critical needs of a stakeholder?

This is traditional Business Analyst territory.

I often start by a meeting, and brainstorming, to get a basic list of 'critical improvements'.

When people suggest specific technology or strategies, as opposed to actual value improvement ideas, then we need to politely elicit the value itself (not some amateur's idea of a solution).

Encryption? So you *mean* we have to improve security? What type of security and how much better?

The logical reason for this is that no one can decide on a design, without clear understanding of the requirements. Requirements (clear) first. Then design.

However for many people the design is a very concrete artifact, and values *seem* abstract. So they will name the solutions, the design, the technology. Your job is to identify the Values which they are *actually* referring to, and put them on the table first.

You do not need to initially aim at the full requirement specification, as discussed earlier in this book. It is enough to get rough ideas. Polish them *outside* of the meeting, and go back later with a set of more-refined ideas (Scale etc).

19.4 Understanding stakeholder priority, power, competence and motivation.

See the **Stakeholder Attribute** list (Fig. 19.2) a few pages above.

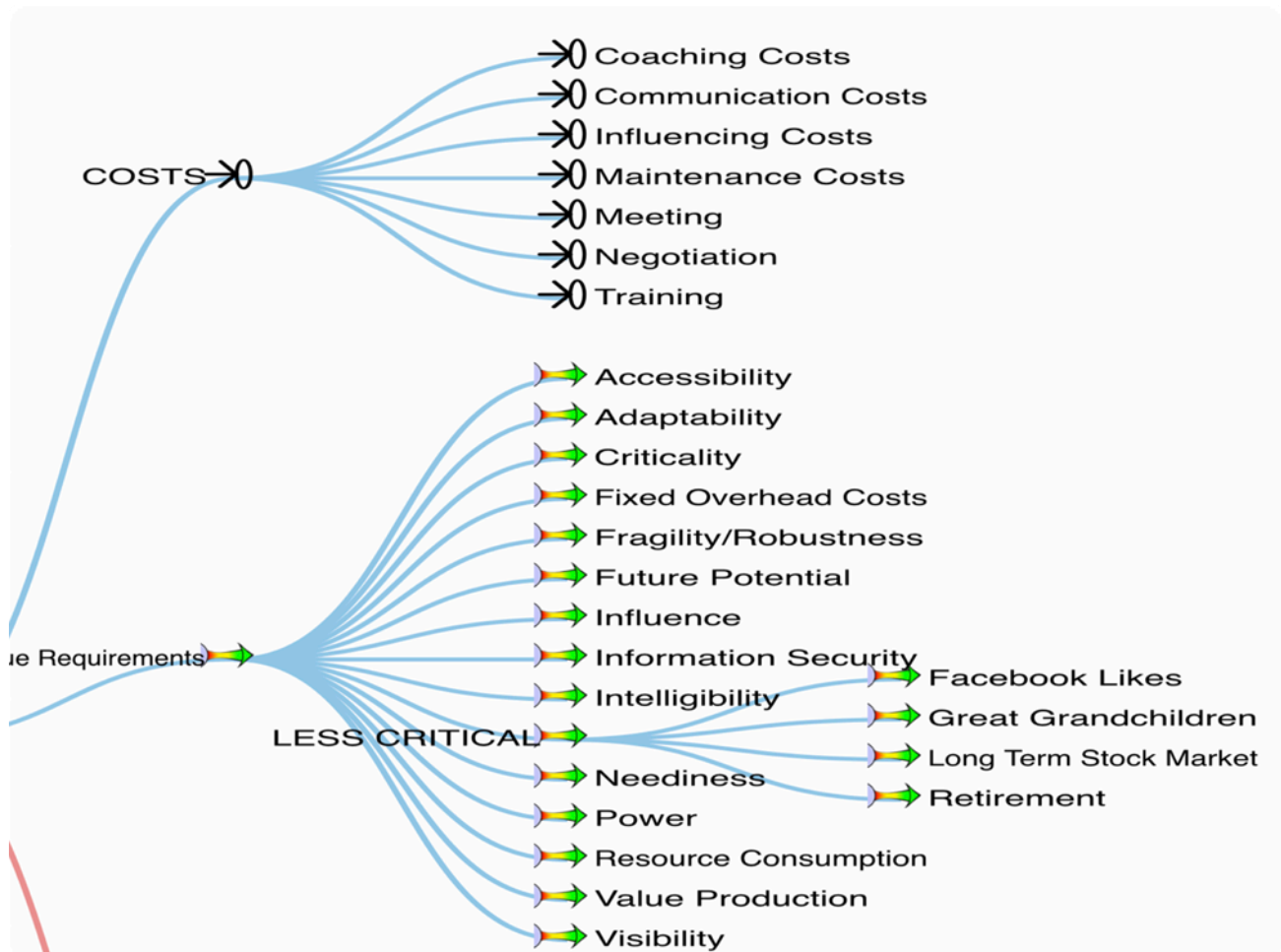


Figure 19.5 : Many of the Values and *many of the Costs* can be used to understand the relative cost-effectiveness of a stakeholder. But you need to quantify them all.

For example: what is their Influence, and Visibility, and what are the costs of dealing with them, like Training costs, Influencing costs.

Rough ideas here, but a starting point for understanding stakeholder value and costs of dealing with them.

Chapter 20. Summary of Value Requirements

Value Requirements are the main reasons for projects.

Other requirements are just useful and necessary details, to be taken into consideration.

We need to clarify and quantify the Value requirements just to have a fair chance of success in delivering them.

Then you have to consider many 'conflicting' Value Requirements at the same time. Finding reasonable balance. Extremes destroy the whole.

Then you have to consider all other types of requirements, like the functions, legal constraints and resource budgets.

This is getting kind of complex, isn't it?

But you have to do it professionally, or you will fail too early and too often in delivering the success factors: the improved Value Levels.



Figure: Values are many, and complex to co-ordinate. But Values must also consider all other types of simultaneous requirements.

Clarity of specification is the first line of attack on this problem.

Appendix 21 Book References

(1) CE: **Competitive Engineering**, 2005, Tom Gilb

Get a free e-copy of 'Competitive Engineering' book.

<https://www.gilb.com/p/competitive-engineering>

<https://www.amazon.com/Competitive-Engineering-Handbook-Requirements-Planguage/dp/0750665076>

(2) SM 1976. **Software Metrics**. ISBN-13 978-0862380342. Library or used copies only.

(3) VP 2016. **Value Planning**. Tom Gilb,

“Value Planning. Practical Tools for Clearer Management Communication”

Digital Only Book. 2016-2019, 893 pages, €10

<https://www.gilb.com/store/2W2zCX6z>

This book is aimed at management planning. It is based on the Planguage standards in 'Competitive Engineering' (2005). It contains detailed practical case studies and examples, as well as over 100 basic planning principles.

(4) VE 2017. **Vision Engineering**.

“Value Planning: Top Level Vision Engineering”

How to communicate critical visions and values quantitatively. Using The Planning Language.

<http://concepts.gilb.com/dl926>

A 64 Page pdf book. Aimed at demonstrating with examples how top management can communicate their 'visions' far more clearly.

This is the core front end of the Value Planning book (3).

(5) LD. 2018. **Life Design**. - eBook <https://www.gilb.com/offers/JHHzGSER/checkout>

(6) CC 2018. **Clear Communication**. <https://www.gilb.com/offers/Y36JRL6g/checkout>

- (7) IC 2018. **Innovative Creativity**. <https://www.gilb.com/offers/FnExtaw9/checkout>
- (8) PPP 2018. **100 Project Planning Principles**. <https://www.gilb.com/offers/Shju4Zqn/checkout>
- (9) **Technoscopes** 2018. <https://www.gilb.com/offers/YYAMFQBH/checkout>
- (10) PoSEM 1988. **Principles of Software Engineering Management**. <https://www.amazon.com/Principles-Software-Engineering-Management-Gilb/dp/0201192462>. \$46
- (11) SI. 1993. Gilb & Graham. **Software Inspection**. <https://www.amazon.com/Software-Inspection-Tom-Gilb/dp/0201631814>
- (12) **Value Design**, 2019. See www.gilb.com.
- (13) **Value Management**, 2019. See www.gilb.com.
- (14) **Sustainability Planning** , 2019. See www.gilb.com.
- (15) **Value Agile**, 2019. See www.gilb.com.

Appendix 22. Papers References, with Free URL Links

(A) **Intel** Cases. Simmons, Terzakis

J. Terzakis,

"The impact of requirements on software quality across three product generations," 2013 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, 2013, pp. 284-289. I think you have the full paper, but let me know if not and I can resend. There's a lot of good background there.

https://www.thinkmind.org/download.php?articleid=iccg_i_2013_3_10_10012

FREE LINK:

(with Gilb Annotations) <https://www.dropbox.com/sh/cs9hke3uvvgg4gp3/AACadHeI95lZpHz-VqGKXSXDra?dl=0>

PAID LINK 2013 RIO PAPER

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6636731&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D6636731

Paper link requires purchase and signin

(B) **The Top 10 Critical Requirements are the Most Agile Way to Run Agile Projects**

<http://www.gilb.com/dl554>

(C) Confirmit Test. Try to get John Watkins analysis of them, a free paper not his book

<https://www.cambridge.org/core/books/testing-it/confirmit/275AE17A5603289AB1F129A418572E1C>

(D)

Confirmit Paper Gilb and Johansson

From Waterfall to Evo

<http://concepts.gilb.com/dl32>

(E) **My Deeper Priority Writings**

1. Managing Priorities, A Key to Systematic Decision-making. With Mark Maier, 2005 (paper)

<http://www.gilb.com/DL60>

2. 'Choice and Priority Using Planguage: A wide variety of specification devices and analytical tools'. (paper)

<http://www.gilb.com/DL48>

3. VP Book, Chapter 6 Prioritization

<https://www.dropbox.com/sh/34llx1a7ckyagxl/AAA0pDzSxN5WmoP9IOKR0Mpca?dl=0>

(F) Gilb '**Estimation or Control**' paper
SQP Magazine, USA
<http://www.gilb.com/DL460>

Slides made for BCS SPA June 1 2011
'Estimation, a Waste of Time'
<http://www.gilb.com/DL70>

(G)
Defect Prevention Process
Mays and Jones IBM SJ paper on Experiences
<http://agileconsortium.pbworks.com/w/file/attach/1527643/Mays1990ExperiencesDefectPreventionIBMSysJ.pdf>

See also 2 DPP Chapters in Gilb & Graham
Software Inspection, 1993

(H) "**User Stories: A Skeptical View**"
<http://www.gilb.com/DL461>
User Stories paper by Tom and Kai Gilb
In Gilb's Mythodology Column, Agilerecord.com March 2011
www.agilerecord.com/agilerecord_06.pdf (whole issue)

(I) Gilb and Brodie, '**How problems with Quality Function Deployment's (QFD's) House of Quality (HoQ) can be addressed by applying some concepts of Impact Estimation (IE)**'
<http://www.gilb.com/DL119>

(J) **OKR Objectives and Key Results: what's wrong and how to fix it.**
<http://concepts.gilb.com/dl879>

Paper 2 Feb 2017

(K) **Project Startup Week**
Agile Project Startup Week Paper in
Gilb's Mythodologies series
gilb.com/dl568

And

‘An Agile Project Startup Week’

91 slides pdf

Talk slides pdf from ACCU Conference April 9 2014

90 minutes talk

Includes Startup Planning for Business Startups, Confirmit, US DoD case, 2 Bank cases, Detailed Startup week outlines and links to sources.

Bristol ACCU Conference

<http://www.gilb.com/dl812>

(L) Principles of Systems Environments

<http://concepts.gilb.com/dl961>

Gilb slides based on Pawel Nowak paper at GilbFest 260619

Based on Pawel Nowak, NOWY, "Context - between model and reality. My attempts to catch the elusive notion". Talk at GilbFest, London, June 26 2019

(M).

Appendix 23. Slides and Talk Video References, with Free URL Links

(a) US: **User Stories as value requirements**. Slides NEED TO MAKE ONE OR IN BOOK

User Stories with Value Metrics 20Feb17.pdf <http://concepts.gilb.com/dl883>

Slides Based on Kai Gilb's Experiments
Mike Cohn commented he liked this.
See also reference to paper (H).

(b)
tinyurl.com/GilbTedx
link tested Sept 2017
Quantify the un-quantifiable

(c)
XAI: Explaining AI
Lecture Slides
<http://concepts.gilb.com/dl958>

A Serious 'Multi-dimensional Metrics Attack' on Poor AI 'Academic and Standards' Thinking & Planning.
An analysis of published Principles for Managing and Standardizing AI, where about 10 AI Qualities like Safety and Transparency are shown to be quantifiable. This is prelude to rational thinking about the entire subject.
GilbFest Talk June 25 2019

(d) **IBM Cleanroom Method.**
Mills and Quinnan Slides
<http://concepts.gilb.com/dl896>

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

Includes Mills, O'Niell, Linger, Dyer, Quinnan p- 466

(e) **What is Wrong with Balanced Scorecard**, slides
<http://concepts.gilb.com/dl135>

(f) **"The Ohno Conspiracy" with**

Quality Function Deployment (QFD) detailed method analysis of method weaknesses.

<http://concepts.gilb.com/dl954>

**(g) 'Design Sprint
A Critical Analysis
and a Constructive Alternative'**

3 Analytical Slides on 'Design Sprint'. 2 Critical Analysis and 1 with my alternative Startup Planning Week.

<http://concepts.gilb.com/dl945>

Appendix 24. Concept Glossary²⁴

Ambition Level: an initial informal statement, from a stakeholder about the degree of a value improvement. Needs to be translated into clear and structured Value Requirement specifications.

Attribute: a characteristic of something. A quality, a cost, a function, anything which can describe and distinguish one artifact from another.

Background: planning specification which is not the core set of ideas, but is intended to give additional context for the ultimate purpose of prioritization, risk management, quality control, and presentation.

Benchmark: a class of reference level on a Scale of measure. It includes Past, Status, Ideal, Trend. It is used as Background specification to allow us to compare with Targets and Constraints.

Budget: a constraint level for a resource requirement.

Constraint: a requirement intended to restrict, to stop, to hinder us with regard to other requirements, possible designs, and any actions.

Defect: a Specification Defect is a violation of official specification Rules. It is poor practice and can lead to problems of using the specification correctly, and timely.

²⁴ This Glossary should be consistent with any other Planguage Glossary. But in the interests of simplicity and freshness I have simply defined things in a simple sentence or so.

Design Idea: (noun): any specification which is *intended* to help satisfy a higher level of Value, Cost and constraints.

Design (verb): the process of identifying and evaluating Design Ideas, for the purpose of satisfying stakeholder values within constraints imposed.

Design Constraint: A requirement specification, that demands or forbids something regarding a design.

Downstream, Upstream: downstream refers to a process to be carried out at a later stage. Upstream, a previous process.

Entry Process: a simple short QC process proceeding any main process, where Entry Conditions, of any useful kind, are checked as a prerequisite for proceeding to the main process. The intent is to make sure we do not waste time or encounter failure in the main process. The cost of the Entry Process should be very small compared to the average results if we did not use it. Above all we use to to motivate people to take the Entry Conditions seriously.

Environment: implicit, the critical design requirement stakeholder environment. An areas or scope where can can and must expect to find critical design requirements, if we study the stakeholders there and their needs. + (added 270719)

Exit Process: a Quality Control (QC) process after any Main Process to try to make sure that it is well done and the outputs are good enough for downstream use. A number of tailored-for-

process Exit Conditions are checked and if all are satisfied, Exit is permitted. If any one Condition fails, no exit is permitted.

Function: an action, do something, a description of what any system *does*. It contains no hint of information about the other attributes of that function, or its container system. Nor any hint of the designs used to create those attributes for the function, or the system.

Icon (Plicon): a graphic symbol which is assigned a Planguage concept. There are two topics, a drawn icon, and a keyed icon. The purpose of icons is to create a human-language independent symbol like music notation, or electrical notation.

Ideal: a perfect level on a Scale, such as 100% availability. Usually not attainable in practice, or without infinite costs.

Meter: a parameter which sketches major elements of a measurement process, for a particular Scalar Value or Cost.

Owner: a Specification Owner, parameter name shortened to Owner, has the exclusive right and responsibility for updating a given Specification Object, such as a requirement.

Parameter: a Planguage-defined Term, which announces the specification of its defined type of information, about a Specification Object, such as a Value Requirement.

Past: a Scale level which is historic. We can usually document in the Past statement, when, where, who etc. Any useful set of Scale Parameter attributes.

Performance: a systems engineering classification for the set of Value attributes. They include all qualities, speeds, work capacity, savings and any other positive attributes valued by stakeholders.

Planguage: a Planning Language invented, developed over decades, published in many books (from 1976 Software Metrics, Data Engineering, perhaps earlier books), and papers, by Tom Gilb, with feedback, maintenance, and creative improvements from Kai Gilb and many other professional collaborators. It is a systems engineering language, with focus on Values and Costs as primary drivers.

Prioritize: to decide sequence of activation.

Procedure: a specified sequence of activities for a defined purpose.

Process: a continuous, repetitive procedure with a possible ending when complete.

Quality: How Well a function functions. Often ending in '-ility'

Requirement: a stakeholder-desired future system state, which can be tested for presence, or measured for degree: but which might be impossible to deliver in practice.

Resource: any attribute which might be consumed, might be limited, and might be needed to build or maintain a system. Money, time, people, dominate but many other resource concepts are potentially useful such as image, qualities, functionality, space.

Rules: a standard in Planguage which specified the recommended way to do, or not do, a specification of any kind. Failure to follow a rules is classified as a specification defect.

Scale (of Measure): a Parameter which defines a Value or Cost scale of measure, for reuse and reference when specifying Benchmarks, Scalar Constraints, and Targets. It does NOT specify a measurement process, that is for the Meter or Test parameter

Scale Parameter: a dimension, announced in [Square Brackets] in the middle of a Scale specification. It is defined using a {set of Conditions}. This device permits quite detailed Modeling of a system, and allows decomposition of problems so that critical Conditions can be prioritized. Example: [Sex]

Scale Parameter Conditions: a set of named conditions which belong to a defined Scale Parameter. Example [Sex] = {Male, Female, Other, Unspecified, Unknown, Multiple}.

Source: the named origin: a person, group, stakeholder, document, or URL of some immediately-previous specifications in a Parameter Specification. The purpose is to enable QC, give credibility, lend authority.

Spec, Specification: a written planning item in Planguage: Requirements, Designs, Analysis, Project Plans, presentations.

Specification Object: a set of Planguage Parameter statements, comprising a meaningful unit of informations, typically a requirement, a design, or sets of these.

Stakeholder: an entity; human, organizational, or document, from which we can derive needs, demands, resource limits, constraints, and any form of information, which can be acknowledged as our potential project requirements, and specified formally and clearly as a requirement. A 'requirement source'.

Status: a numeric update of the incremental progress of a Scale Level as we incremental deliver a system design components and measure progress towards our requirement levels.

Standards: best accepted practices for developing and maintaining systems. These include, Rules, Procedures, Exit Levels, Concept Definitions, Templates, Scales of measures, and even App conventions.

Target: a level of Value that we are aiming to reach. It includes Wish, Goal, Stretch.

Trend: a Background Benchmark level, which estimates the future of that level. Useful for pointing our Value degradation, or potential competitor future levels of Performance.

Use Case: a written graphic description of how a system element might be used in practice. In Planguage it can be covered by using an appropriate Scale Parameter. Example: [Uses] : {Register, Delete, Update}.

User: a person who personally and physically interacts with a system.

User Story: a requirement statement in the format: Stakeholder + Requirement + Justification. This is roughly at the level of an Ambition Level, and can replace Ambition Level as a starting point for formulating a more detailed Planguage requirement.

ValPlan: ValPlan.net is the URL of an App released for sale May 2019 by Gilb International AS. It is based on Planguage and the Competitive Engineering book.

Value: value is perceived stakeholder benefit.

Appendix 99. Notes on editing the book and Versions.

Started about 3 July 2019.

22July2019. First complete version ready. At Cabin

■ Last page.

EDIT 5AUG 2019

FAULT TO RECTIFY PAGE 195

Note 5 aug2019 I do not know where this figure is or what it is.

I PUT IN A CYCLE W IET ON 29 SEPT 2019

Figure: Architecture Engineering is a very systematic quantified discipline, which relies on quantification of Value Requirements (topic of this book) as an input.

August 14 2019

Moved TOC to front and put ship picture on cover, and edited risk defy x 2

240919: added 4 Gilb Summer books 2019 to references

Noted need edit. page 188 missing 2 figures (THE 2 FIGURES PUT BACK IN SEPT 29 2019) 203 the word constraints under figure needs edit, also page 208 missing fig., and see over page 195

September 29 2018 edit of missing illustrations

Putting in Ill and example numbers

To do

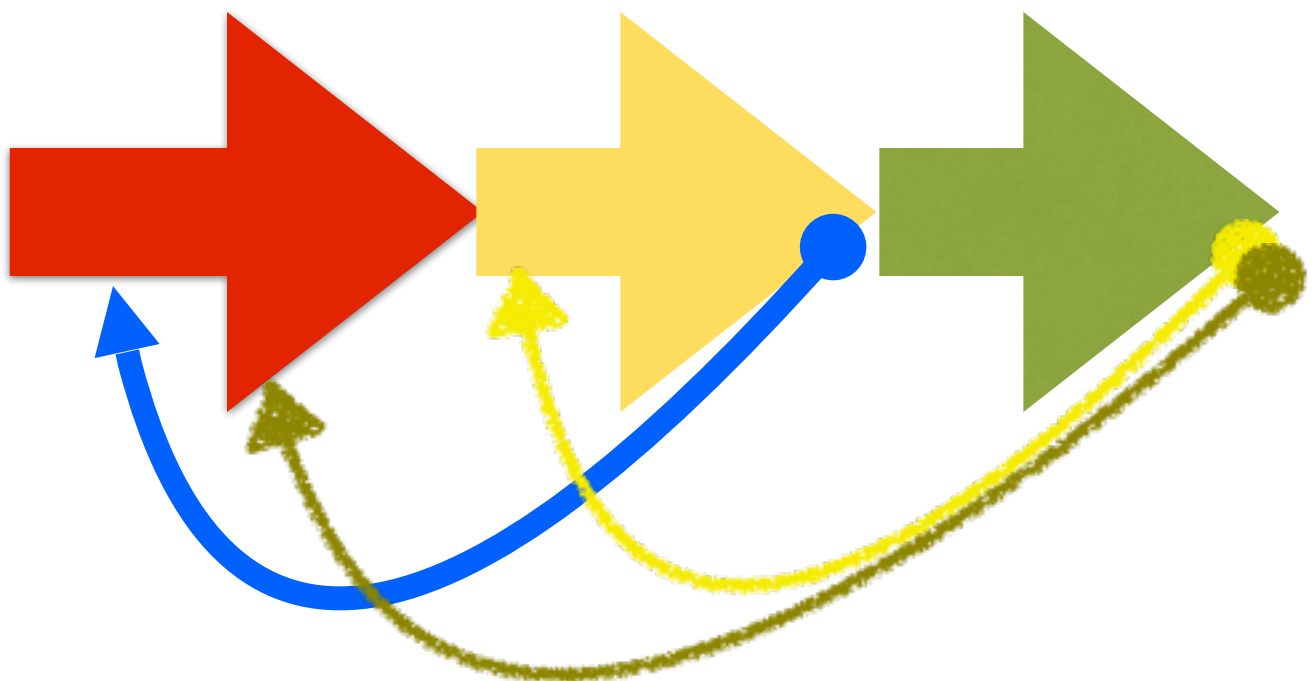
() Go to older copies and fix the 1.13? Tesla ill which is missing, get it from a backup copy

() Fig 1.14 ValPlan examples missing

Put in the word Chapter in every chapter, and numbered the sub chapters.

September 30 2019

1 Oct. 2019; full text edit of whole book. Clarifications and corrections .



I have not yet made this public either at twitter linkedin or gilb.com.

I have also received no feedback about the book. But I did send a version to selected GilbFest friends.

Full text edit whole book

1 oct lost diagram found around 15.2 use cases
