"VALUE MANAGEMENT"

SUPER MANAGEMENT TECHNIQUES

SUMMARY

In a Sound Bite

Value Delivery Depends on Good Management

A One Liner

Management can control value delivery by quantifying values as well as they do money and time

Main Ideas

1. Stakeholders determine critical values
2. All critical values can be expressed as quantitatively as you do time or money
3. All strategies for delivering values can be estimated and measured
4. Contracting can be based on real incremental delivery of useful value improvements
5. Motivation and responsibility can be value driven

Principles

1. If you analyze your stakeholders well enough, you will discover values critical to your success or failure
2. You can pin down all critical values clearly by *quantifying* them.

3. You can analyze, understand and make good decisions on strategies if you *estimate and measure* their values and costs.
4. You can deliver big successful improvements in values, in very early *small increments*, so that failure is impossible, and impressive results are inevitable.
5. Extreme *focus* on values will result in values: do not get distracted.

6. If your organization does not seriously care about delivering real values, then consider switching to one that will survive.

1. MANAGING VALUE REQUIREMENTS[1]

Not Good Ways of Managing Value Requirements

Value Requirements: are defined as the top level, maybe top 10, objectives for your current project. They define success, and failing to reach any one or more of them, defines failure[2].

The basic problem with value requirements is that our current culture hides them from sight. Does not consciously deal with them. Specifies them badly. Distracts you from them with other similar things, like the solutions, not the value problems.

So if you just accept something *called* requirements, or objectives, for your project, as given, and correct; you are doomed to fail to deliver the real critical values of your critical stakeholders.

The conventional thinking is highly misleading, so your job in managing values is to make sure you are really dealing with the values that actually count.

It does not help the situation that your higher ups in the organization, do not understand much about this situation at all. Now you *could* play along with this ineffective culture, and fail along

[1] See Value Requirements book, 2019 gilb.com for 240 pages of detail on the subject of Value Requirements

[2]On Failure, see reference B, The Happy Project Saboteur. https://tinyurl.com/HappyITSaboteur

with them. But you do have an option to diplomatically improve the project objectives, and then do something really useful about them.

This will make someone look good. Be sure to give all credit for supporting and initiating your efforts to your boss.

A Big Failure Case Study of CEO-Driven Fuzzy Values.

In one client case, the CEO demanded 'Rock Solid Robustness'[3] and about 7 other equally fuzzy top level values. Nobody dared challenge the CEO[4], not even the CIO I talked to. The project was in failure mode after 8 years, $100 million, and about 80 people on the project. They could never succeed, because nobody had a clear definition of success values! I showed them how to quantify their values (see footnote for quantified detail), and they finally succeeded.[5]

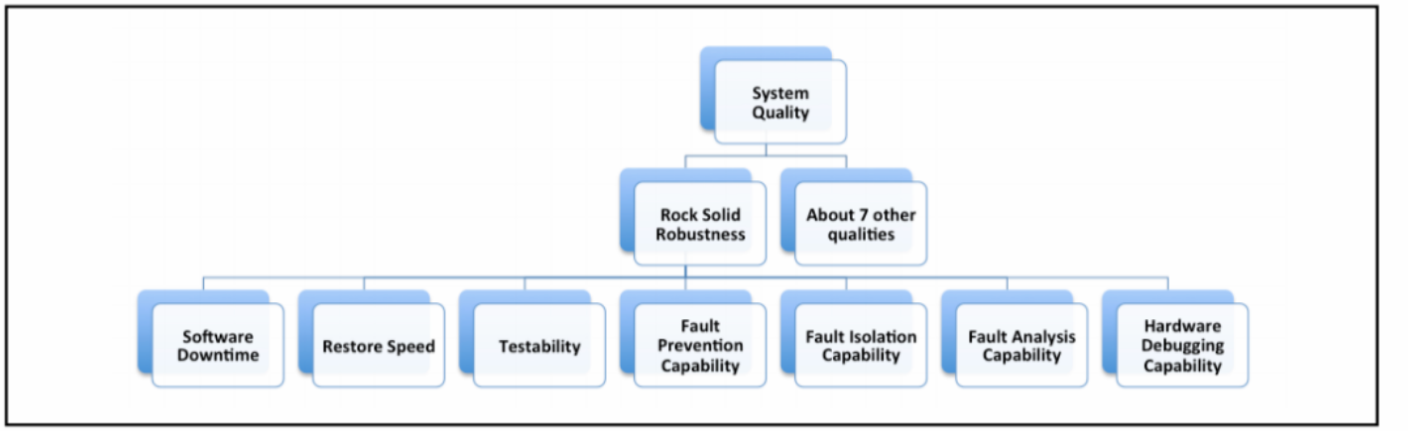

FIGURE 1 (Robustness) In the case above, the key to decoding what the CEO wanted for Robustness, was to decompose it into 7, fairly-conventional engineering understandings of it[6]. Then to quantify each one of them.

---

[3] https://accu.org/content/conf2013/Tom_Gilb_Quantifying_Robustness_Lightning_Talk_ACCU_2013.pdf, See also Value Planning case 1.7

[4] there is no question of opposing the CEO. He wanted the right values. But the CIO and PM needed to translate the CEOs ideas into more-detailed quantified actionable values. The CEO did not want this big failure! Of course the CEO *could* have had policies to make people quantify values and deliver them incrementally. We did that successfully in other parts of that same company. But it was NOT Corporate Wide, yet.

[5] Quantifying critical CEO values was half the trick to saving this project. The other half was to get out of Big Bang mode, and to deliver priority value increments: a subject we will talk about later in this book.

[6] For a similar breakdown of Maintainability see the Competitive Engineering book Chapter 5, page 156, Chapter 5: Scales of Measure: http://www.gilb.com/DL26. I used that pattern to solve this problem.

Focus on Users and Customers is too Narrow

It seems like a good idea, at first. 'Focus on your users and customers'. But while it is not a bad idea, it is too narrow. There are really critical sources of project values, that are outside this 'user and customer' scope. They are called *stakeholders*. You have to get the stakeholders, like shareholders, victims, and laws[7], that are *not* customers and users. So if you see a focus on *users* and *customers*, and you do not hear the word **stakeholder**, you know you are in *bad* company. The project is now on a *failure* path.

Requirements are not Values, but are Technology and Solutions

There are worldwide cultures for projects that do not *really seriously* deal with their core values. Just lip service: 'safety is our first priority'. They will specify as objectives, or more technically, as 'requirements', the following types of things:

1. Technical solutions: like 'migration', 'digitalization[8]', 'modernization', AI-system, Big Data.
2. Functions: what the system is supposed to **do**, but not ***'how well'*** it must do it (values).[9]
3. Unintelligible 'value' statements: like 'state of the art security', 'the most exciting and competitive products on the market', 'a

[7] See stakeholder map just below, Figure 1 Stakeholders for more examples.

[8] why? Those answers are the values, and if you do not manage them, you get failed 'digitization' as we have too many examples of as large (fire the Minister) public scandals, like in Norway and UK, everywhere.

[9] most systems already do their necessary functions. Replicating existing function is wasteful. Improving the values, qualities and cost levels of the old system, is the main point in all cases. Fastest way to do that is to incrementally improve the existing, bad old, system, not to rebuild functionality from scratch.

welfare system for the underdogs', 'radical reduction in pollution', 'education for the worthy and deserving'

Values are Nice-Sounding Words

The values are usually floating around somewhere, such as presentation slides to sell the project.

But they are called things like: 'Expected benefits of the new system'.

They are formulated in vague appealing terms, with no clear commitment to exactly what value levels will be delivered to whom, when, and in which special cases (handicapped, emergency, war, novices, etc).

These fuzzy phrases are dangerous and worthless to your project. You cannot let them stand, and do nothing. But you can convert them into far more precise commitments, of real value.

As an extremely simplified example of this:

'High Security', could be rewritten as

'95% chance of catching a hacker within 5 seconds'

With 'High Security' you have no idea what the value actually is, or what your project has to do, of it your project has done it. You can fail in the eyes of others, if they are expecting more than you are interpreting. With the 95% statement, you have the beginning of a rational managed process of value management.

More Effective Ways of Managing Value Requirements

Stakeholder Management

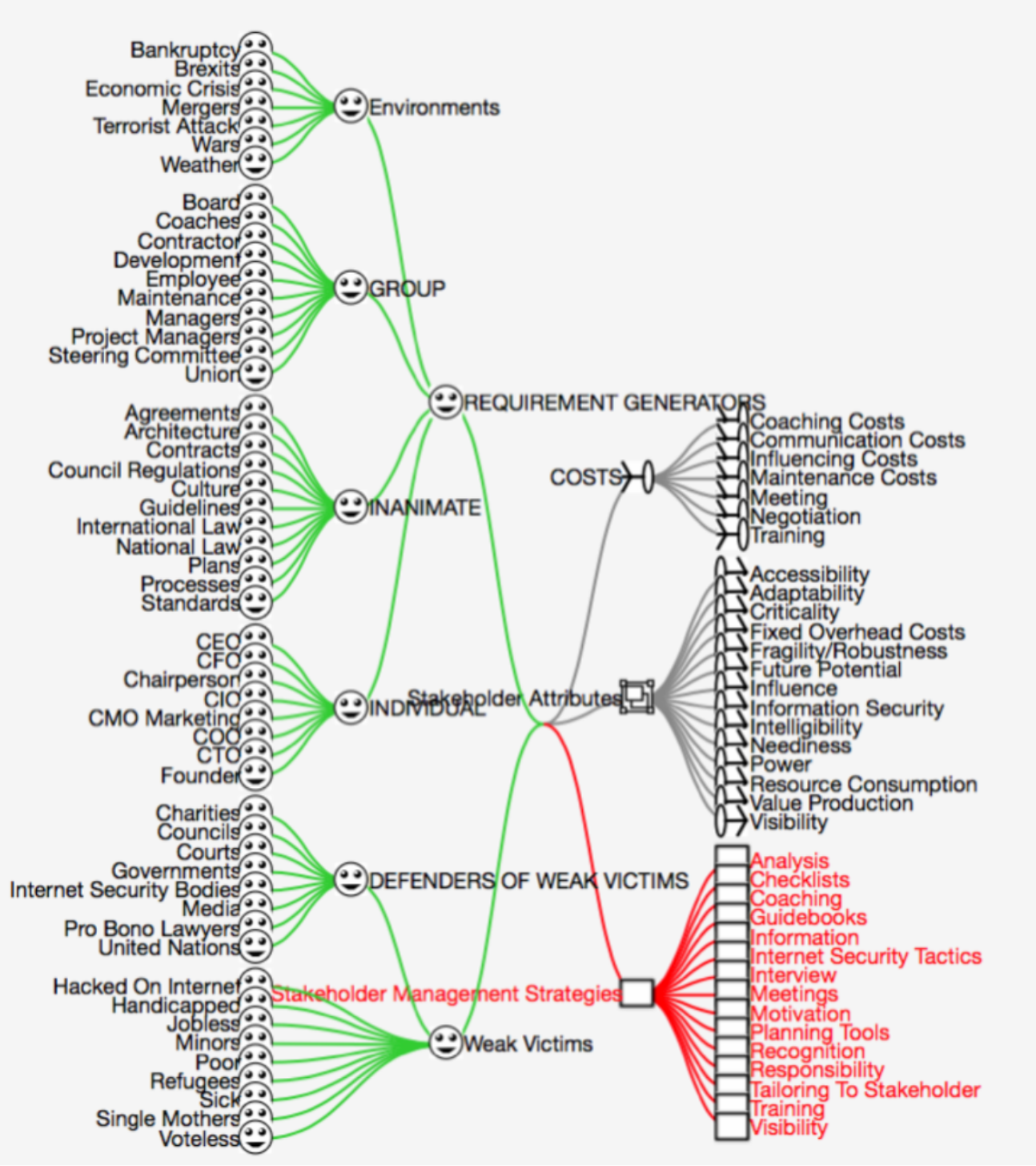

Figure 1 (Stakeholders): on the left sides of this generic stakeholder map are classes of stakeholders, and to their left, examples of types of them. Top right are examples of costs of dealing with stakeholders. Top

middle are examples of attributes of stakeholders, that tell us about their priority. And right bottom are some potential strategies for eliciting stakeholder values from stakeholders.(Source: Value Requirements book, 2019, from Tom Gilb personal models of stakeholders.[10]).

Your business analysts, or whatever you call the people who analyze and specify requirements, need training, guidance, and standards for analyzing stakeholders.

[10] See Reference (A), Stakeholders for more detail.

Value Quantification

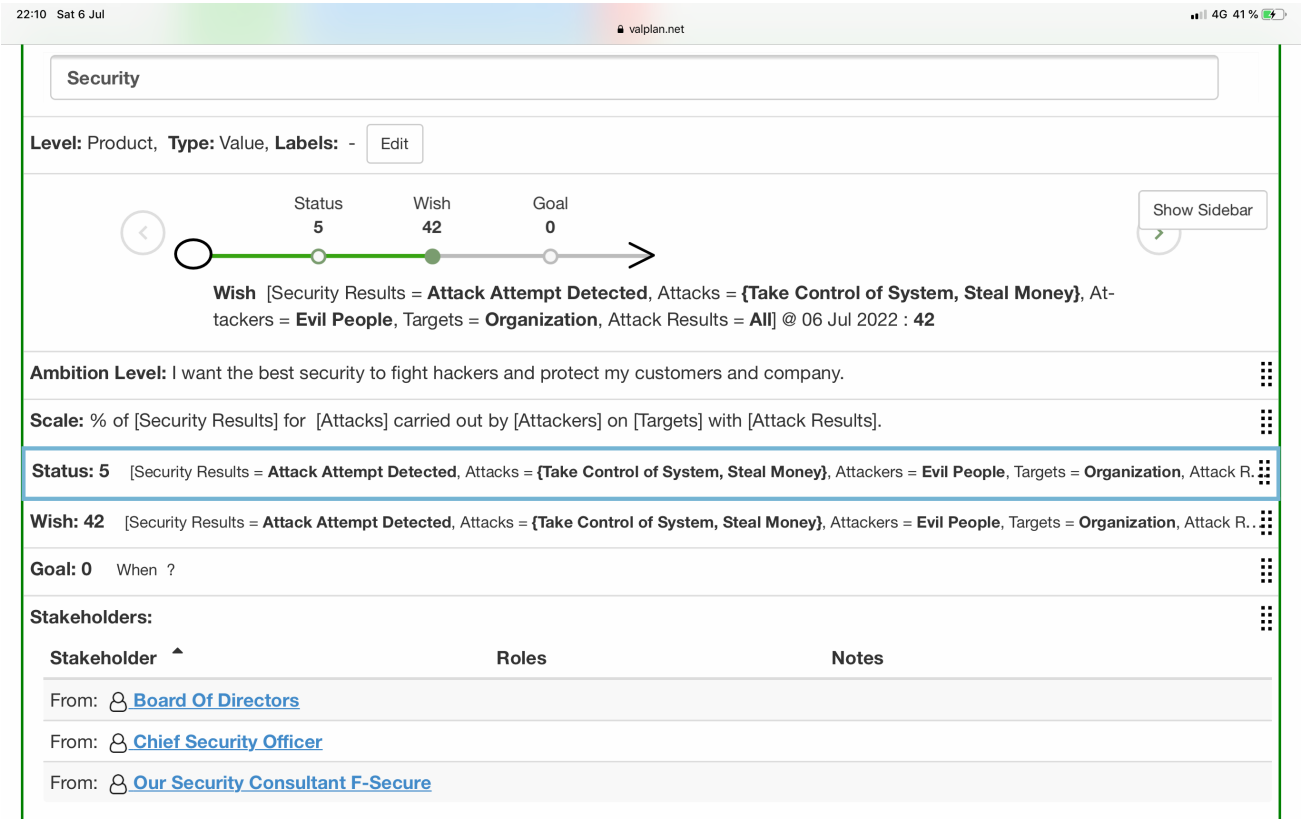

Figure 1(Quantification): Here is an example of quantification, and clarification (Source: Value Requirements book, 2019).

In the value quantification example above, the fuzzy 'Ambition Level' is 'clarified' by a defined 'Scale' of measure. The Wish level specifies an improved value of security (42 on the defined Scale), to be delivered at a define deadline.

The Wish requirement level also specified certain classes of Security Results, Attacks, Attackers, and Targets. These are special dimensions of our problem. They are 'conditions' for the requirement to be applicable.

Notice that there are three different stakeholders explicitly associated with this one critical objective.

Principles (Warning Signals)

1. If you focus on 'users and technology', instead of stakeholders, values, and what is critical: you will probably fail.
2. If you quantify values, and make all related conditions clear, you have the basics for value-delivery success; otherwise failure is pretty well guaranteed.

Policies (Leadership Declarations)

1. We will discover and track all critical stakeholders, in order to discover the critical values and constraints of our project.
2. We will quantify and clarify all critical values, so they cannot be misunderstood, and will be delivered as specified.

Checklists (Analytical Warnings)

1. Are your people only talking 'users' and 'customers', or do they have an 'all critical stakeholders' culture?
2. Are the requirements, objectives, and 'expected benefits' of the project specified in fuzzy words? Can they be rewritten clearly and quantitatively instead?

3. When you seem to have difficulty quantifying a value (like 'Security', try searching the internet with a keyword like 'Security metrics'. It is amazing how many people have solved this quantification problem for you already. For free for you.

2. MANAGING VALUE DESIGN[11]

What Is Design, and Then 'Value Design'?

A 'design process' is needed to move us from *abstract ideas* of

- *how good* a future system will become (values),
- and how few *resources* (money, time, people) we will need to achieve those value levels.

The output of a design process is *designs*, the *how*, we intend to achieve our values, within our limited resources.

The ratio of 'Values/Resources' is an expression of the *efficiency* of the design itself.

Design ideas are usually very concrete ideas of what we need to do or build, in order to achieve our value levels.

A design process involves abstract thinking, and some people have difficulty with it. They prefer to think about concrete things. The world needs both types.

If there is no design process, people will just keep what they have, or take a chance on a new fad, which might be worse. But there will be no *conscious* improvement in value levels and costs. Just stagnation, and accidental change for better or worse.

---

[11] See the booklet Value Design, 2019, Value Design
BOOK: https://tinyurl.com/ValueDesignBook
VIDEO https://www.youtube.com/playlist?list=PLKBhokJ0qd3_wlvr0j85YhmNfNj8ZJ8M-
SLIDES http://concepts.gilb.com/dl972 for more technical detail on value design methods.

This is where management comes in. Management has these leadership and organizational roles:

- Identify necessary value and cost ambition levels (requirements)
- Get these ambitions translated into specific actions (designs)
- Make sure the designs are brought to life, giving the values (results)
- Make sure that the big picture (all critical values, all critical stakeholders, all critical resources, all available design options) is dealt with. Avoid sub-optimization. (Systems thinking)

Without these management roles being done well, stuff will happen, or stay as it is: but it will not be competitive, improved, and sooner or later, there will be a need for better management.

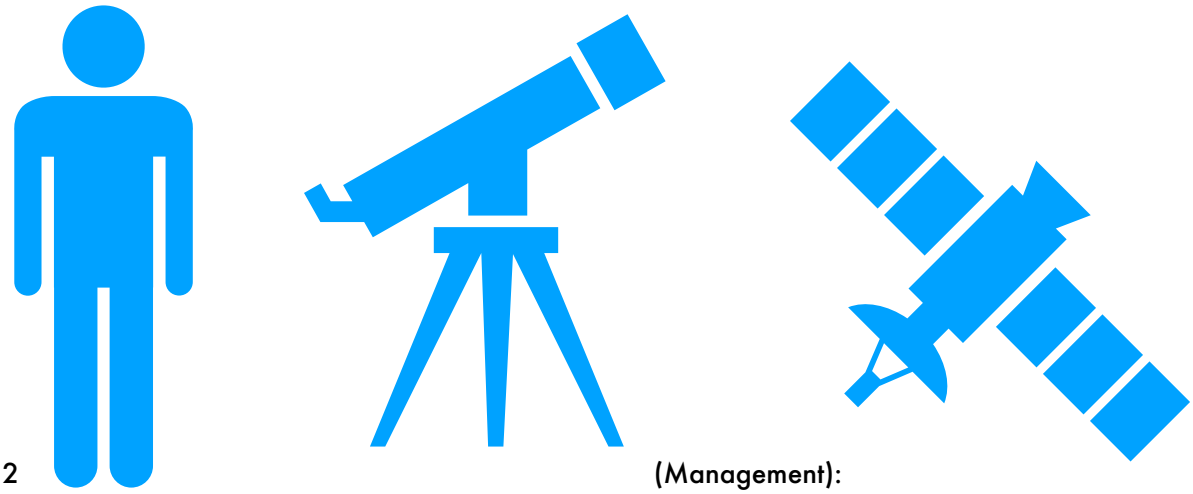

Figure 2 (Management): Values ? Design?

How Not To Manage the Design Process

No design

At one extreme, there is no official design process at all. No designers, no engineers, no architects, no strategic planners. Just people involved in constructing the solution. Not in delivering the critical stakeholder values, or managing resources.

There is a class of simple system-building project, which might succeed with no design process. It would reuse previous designs which worked well, be small, low cost, and there might be no clear requirement, from any stakeholder, to improve the value levels, or to reduce the costs.

It might be as simple as 'to buy a product or service', use it, live with it. I do that all the time, personally.

So what are the conditions that really need, perhaps *demand*, a *design* process?

- Large size (many people involved)
- Long duration (months, years)
- High capital costs
- Potentially high, operational and maintenance costs
- Many stakeholders
- International scope
- Political pressure
- Responsibility, legal, financial, political

- Complexity (many related systems together)
- New technology (you have not used it, few or none have)
- Very high 'quality requirements' (safety, security, availability, usability)
- And many other disturbing factors

A conscious design effort is necessary to predictably deliver the value levels to the stakeholders, at acceptable resource levels.

Another simple method, without design, is 'just have a go'. Fail. Have another go. Fail. Repeat until people are happy. The problem is, this might go on forever, and in any case probably cost far more time and money, than designing a good solution to begin with.

Design is about using our brains to find smart solutions, to well-defined problems. The effort to design successfully, should be far less than 'muddling through', trying and failing.

The Agile Non-Design Culture.

There is one popular culture today, agile[12] (Scrum, for example) which does not have a clear *design* culture.

---

[12] 'How Well Does the Agile Manifesto Align with Principles that Lead to Success in Product Development?' by Tom Gilb
and 'Why Agile Product Development Systematically Fails, and What to Do About It!' by Kai Gilb and quite a few links to our other books and papers. 26 Feb 2018 in SyEN
https://www.ppi-int.com/wp-content/uploads/2018/02/SyEN_62.pdf

Agile was, historically, an overreaction, from experiencing too much bureaucratic overhead, in US Government contracting, in the previous century. Possibly, they would argue, agile is a 'framework' into which you can, and should, insert the level of 'design process' you need.

But looked at from my point of view, agile is mainly a process for managing production of software, and is sponsored by people who have no declared interest in design, values, costs, and stakeholders. Some, like 'SAFe' talk warmly about value delivery, and do make *some* attempt to quantify values. But they do not meet my standards for clarity, and conscious design.

So these craftspeople must be excused. There is no point in criticizing a carpenter, because they are not an architect.

The management problem here, is when 'management itself' sponsors an organizational cultural change 'to be agile', and does not even realize that *that* change, does *not* contain design. But they will, with lack of good design, tend to fail in their projects, and blame it on something else.

Most other things labeled 'agile' today are, in my view, no use at all in helping management design good value/cost solutions, in large complex systems projects. Agility itself, of course is a good attribute! And it should imply quick redesign when incremental feedback measures bad design (as in IBM Cleanroom, Ref. C).

Pseudo Design

At another extreme of bad practice, there *do exist* official designers (UX)[13], and 'architects' (enterprise architecture)[14], or strategic planners[15]. But they are not trained and managed to do the job *properly*. What does that mean? 'Properly'?

They do get training. I suppose they have managers. They do spout designs, architecture, and strategies, but they lack:

- Any serious stakeholder analysis (they do 'users')
- Any serious clear and quantified specification of value requirements (including quality requirements), and constraints (including time, money, operational costs, and legality).
- Any serious numeric analysis, of any design suggested; in terms of its impacts, on any of the many values or costs.
- They often have no sense of the larger system, and live in a narrow domain.

I find it mind-boggling that I see large expensive projects, with these enterprise architects, who have no concern for critical values and costs. They do not even feel ashamed or embarrassed.

Of course the root problem is that *managers* permit these dangerous beasts to exist, and to determine values-and-costs results, blindly.

We have a very high project failure rate, and this is one reason.

---

[13] 10 Suggested Principles for Human Factors in Systems Engineering, http://concepts.gilb.com/dl911 Keynote at WUD (Worldwide Usability Day) 2017. https://youtu.be/TlDCwmVgDJQ , Video 42 min.

[14] Real Architecture: Engineering or Pompous Bullshit? www.gilb.com/dl741,

[15] See Value Planning book on strategy planning. Refs. (3), and (4) http://concepts.gilb.com/dl926

Do not allow these practices, this ‘witchcraft’, to persist on your management watch. Take responsibility for critical values and costs.

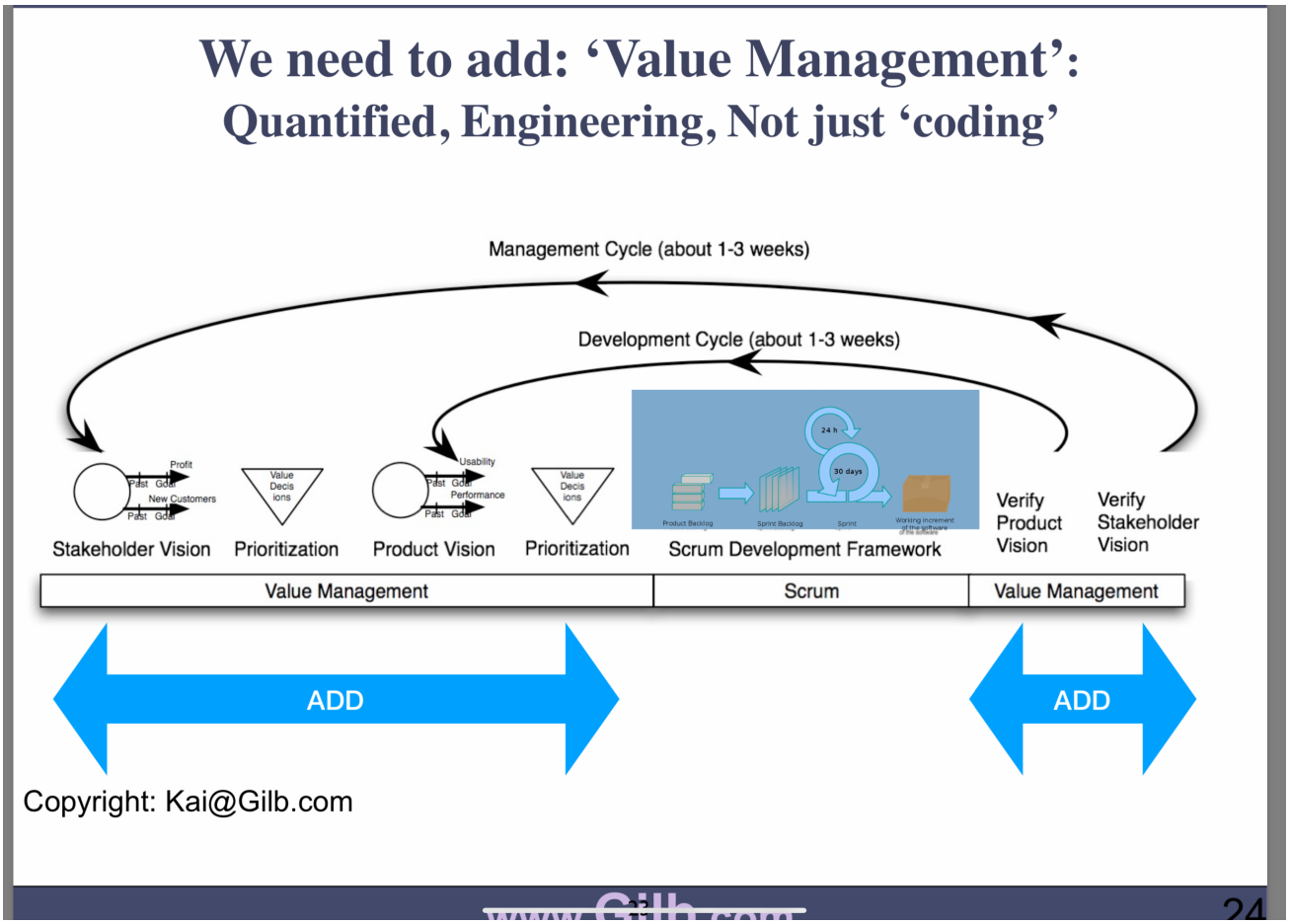

Figure 2 (Scrum). Agile/Scrum is missing ‘Value Management’. (Adding Value to a building process)

See the Systems Enterprise/Engineering Architecture ‘SEA’ book for more detail on my version of better architecture [15].

Value Design Management. The Right Stuff.

So, what does a manager need to do, to manage the design process?

In simple terms:

These pre-design steps were discussed above.

- Make sure reasonably thorough *stakeholder analysis* is done.
- Make sure Critical 'design' *requirements*, for values and resources are specified: as a minimum, that they are *all quantified*.

Now we are ready for the 'design' steps

- Make sure that all *design options are estimated,* before choice and prioritization
  - The main-asserted critical design-value *impact* is estimated
  - All other critical values (the other 9 in top 10) are estimated for the impact of the design (*side-effects*)
  - All budgeted *resources* impacts (time, money, operational costs) are estimated
  - All designs evaluated for all stated legal/cultural/contractural/ stakeholder *constraints*.
  - All estimates of value and cost, are made with respect to ± *uncertainty* range
  - All estimates have *evidence stated,* and rated (credibility level)
  - All designs and estimates specs are *quality controlled* (Spec QC, chapter 3 below)

In case this sounds like a lot of work, we normally get a small team to do a pretty good draft of this, in a single day. See figure below.

If you make sure these design evaluation tasks are done, then you as a manager, are making sure that *designs are seriously evaluated*, *before* they are selected for implementation.

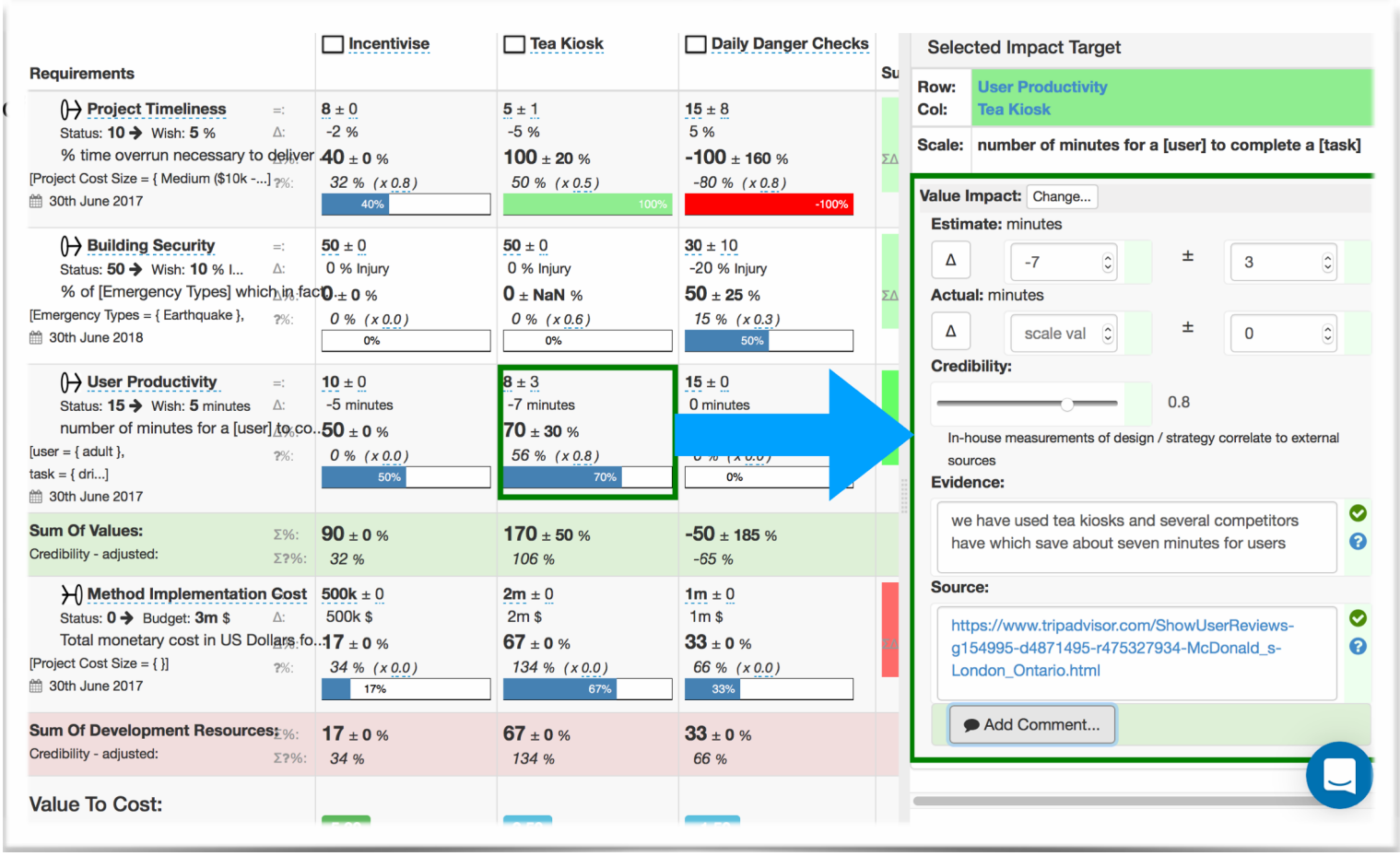

Figure 2 (Value Table). This table is a systematic way to evaluate the values and costs of designs.

This is simply rational, logical thinking. It is evidence-based decision-making. It is clear-headed management. It is not fuzzy intuitive yellow-sticky, seat-of-the-pants, failure-prone, culture.

One approach to this, is to keep on asking the Tough Questions.[16]

[16] see Ref. [16] '12?: Twelve Tough Questions for Better Management', https://tinyurl.com/12TOUGH, booklet 2020

"So you tell me this strategy is more secure? Can you give me a number for experience, you can cite, of how secure it is?"

"To what degree would your design or architecture be sure to meet our numeric Goal levels, within our short deadlines and miserly budgets?"

Principles for Design

1. Your designs must contribute substantially to your value objectives at low costs: estimate and measure the levels
2. If you try out your design ideas in small increments, you can adjust designs, and never fail, on a large scale.
3. All designs have at least 9 side-effects on your critical values, and at least 6 cost aspects, some very negative cost-effects; so you need to try to discover these, as soon as you can, estimate, then measure the design effect-delivery, incrementally.
4. Your designs need to be tried out in practice, in small increments, so if they disappoint, you can dump them fast.

Policies for Design

1. Designs will be estimated *before* their selection.
2. Designs will be tested in practice, *before* keeping them in place.

3. The preferred designs will be those with ‘high values’ and ‘low costs’.
4. Our designers will justify their design suggestions with numeric facts, and evidence, and practical demonstrations.
5. The name of the sponsoring designer/architect/strategy sponsor will be annotated, and public.

Checklists for Design

1. Is each design idea specified, in enough detail, to enable us to understand, its value-and-cost ranges ?
2. Are there any estimates, with evidence, for the design values and costs.
3. Are large designs decomposed, into small implementable increments-to-existing systems?
4. Who exactly is name-responsible, for the *failure* of any specified design/architecture/strategy?

3. MANAGING QUALITY ASSURANCE (QA).

Some Basic QA Definitions.

Quality: *How Well* a function 'functions'. Often ending in '-ility'

Quality Assurance (QA): any process that contributes to meeting and maintaining quality levels, including all specified stakeholder value level objectives. The primary emphasis should be on *prevention* of lower qualities than planned, not merely emphasis on *detection* and *correction*. Typically prevention tactics are organizational and technical improvement processes, so that subsequent work processes, lead to better quality. For example QA is engineering, architecture, design, process improvement. Avoid misuse of this term to simply mean 'testing' .

Quality Control (QC): *checking* any type of quality level, and related factors, to make sure they are OK. Removal and correction of bad-quality artifacts. Typically QC is *reviews* and *testing* processes.

The QA processes related to *design* and *implementation* are covered in the other parts of this book. So in *this* chapter I am going to focus on *process improvement*, and *quality control reviews*.

The Process Improvement QA Processes.

It is management's job to make sure that all work processes are both *sufficient to reach organizational aspirations*, and are also *cost effective*. That is what you hope, I assume, to get ideas about, from this book.

There are many specific methods for process improvement, so I am going to simplify, and focus on those I personally have best experience and faith in. Anything that is more cost-effective for your purposes than these, I would encourage you to make use of.

Designing Your Own QA Process

Rather than pointing you to any one of the many QA processes, or even to the many 'Process Improvement' processes; I will simply make use of the basic ideas in this book, to 'design' a QA process. This is the method I have used with my clients for decades.

The advantage of this *organizational design process* is that it is so general, that it can be used for many other purposes: designing any other organizational process, and designing any product, service or system.

The Organizational Improvement Process.

1. Define your organizational improvement *objectives* quantitatively. Improve your definition periodically
2. Find candidate organizational QA *strategies*, and estimate their effectiveness, and costs.

3. *Decompose* big strategies, into smaller implementable strategies, and try them out in practice. Keep if good. Modify if necessary
4. Continue this process until *all your current objectives* are reached.

Here is an example from one of our Clients.

ERICSSON CASE[17]

Ericsson of Sweden, Mobile Base Stations needed organizational improvement, so that they could produce more product faster, for an eager international marketplace.

This involved many simultaneous, critical, value-improvements, including quality and productivity improvements. Both highly related.

Support
the **Fundamental** Objectives (Profit, survival)
Software Productivity:
Lines of Code Generation Ability
Lead-Time:
Predictability.
TTMP: Predictability of Time To Market:
Product Attributes:
Customer Satisfaction:
Profitability:

Figure 3.1 (Ericsson Objectives). The Chief Technical Officer level objectives for improving about 3,000 engineering staff. *Detailed* specification is in the footnote reference (gilb.com/dl559).

These objectives (Software Productivity ... Profitability) were the CTO's supporting

[17] Productivity Slides incl Ericsson
http://www.gilb.com/dl559

objectives ('Strategic Objectives') , for their the higher-level corporate objectives (profit and survival). The CTO objectives were themselves supported by a set of 'Means Objectives' (see ref gilb.com/dl559 for detail)

Figure 3.2 (Ericsson Strategies) These were our candidates for improving the organization. The top rated ones are in red, upper left. We rated known international experience in similar corporations (IBM, Raytheon, HP) as our method of getting hard evidence of what was most cost-effective, and most-sure to work. Some of the ideas, bottom right, are empty nonsense, and only included for 'political' reasons. But they never 'got off the list', because *no evidence* existed for them. (Source gilb.com/dl559.)

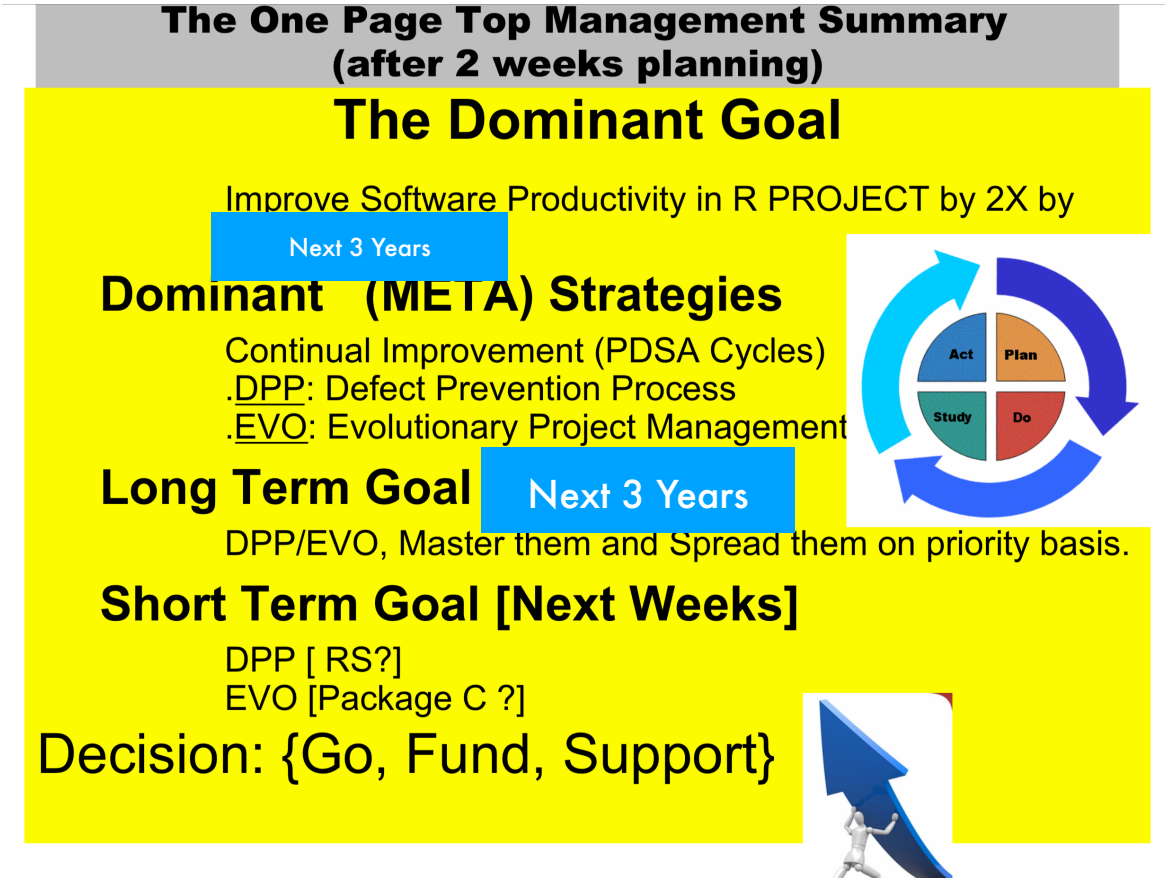

Figure 3.3 (Decision) This is the decision, after Kai Gilb and I worked with the Ericsson CTO (Thomas Ericsson) Team for 2 weeks. Three specific processes were selected, based on credible cost-effectiveness, and we got, in the CTO presentation meeting, acceptance to start using 2 of them by the next week.

The 'DPP' mentioned above, is the 'Defect Prevention Process' (Reference E), and the DPP *decomposition (to a next-week value-delivery)* is to apply DPP only on 'RS' (Requirement Spec). We had spent the previous year, working directly with the CTO (Thomas Ericsson) using DPP, so he was already convinced, that it was a good process, for improving the quality of their 'product development processes'.

The 'Evo' is our Gilb 'Evolutionary Value Optimization' project management process[18]. This was no easy decision because it was in

[18] Chapter 10: Evolutionary Project Management:
http://www.gilb.com/DL77

conflict with the prevailing (U Model, Waterfall) corporate standard. But we won out because there was good evidence from others (IBM Cleanroom (Ref. C), HP use of Evo (Ref. D)), and because this Mobile Base Station group ('ERA') had some good experiences, with similar incremental methods, in getting products to Japan quickly.

The DPP, Defect Prevention Process Raytheon Case (Ref. E)

My favorite specific process for improving organizational qualities is the little known 'Defect Prevention Process', which I have used at Boeing and Ericsson. IBM never 'marketed' it, just used it internally. But at least it is free, compared to some other processes like CMMI which can cost an *arm and a leg*.

<u>DPP is simple:</u>

- Grass roots professionals analyze their *own* everyday faults and problems (it is *not* done by managers or consultants)
- Grass Roots suggest process and tool changes, to prevent the problems re-occurring
- They can try out the suggested changes, before scaling up
- The new process is measured against current critical improvement objectives (qualities of product and services)

The key DPP successful idea was *delegation of power* to analyze and be creative, delegation to the troops, not to top management and

their consultant corporations (who always have a big idea that fails). (Ref. F)

'Management's job' is to make sure their organizations are improving their *value objectives*, and here is an example of a proven way to do it.

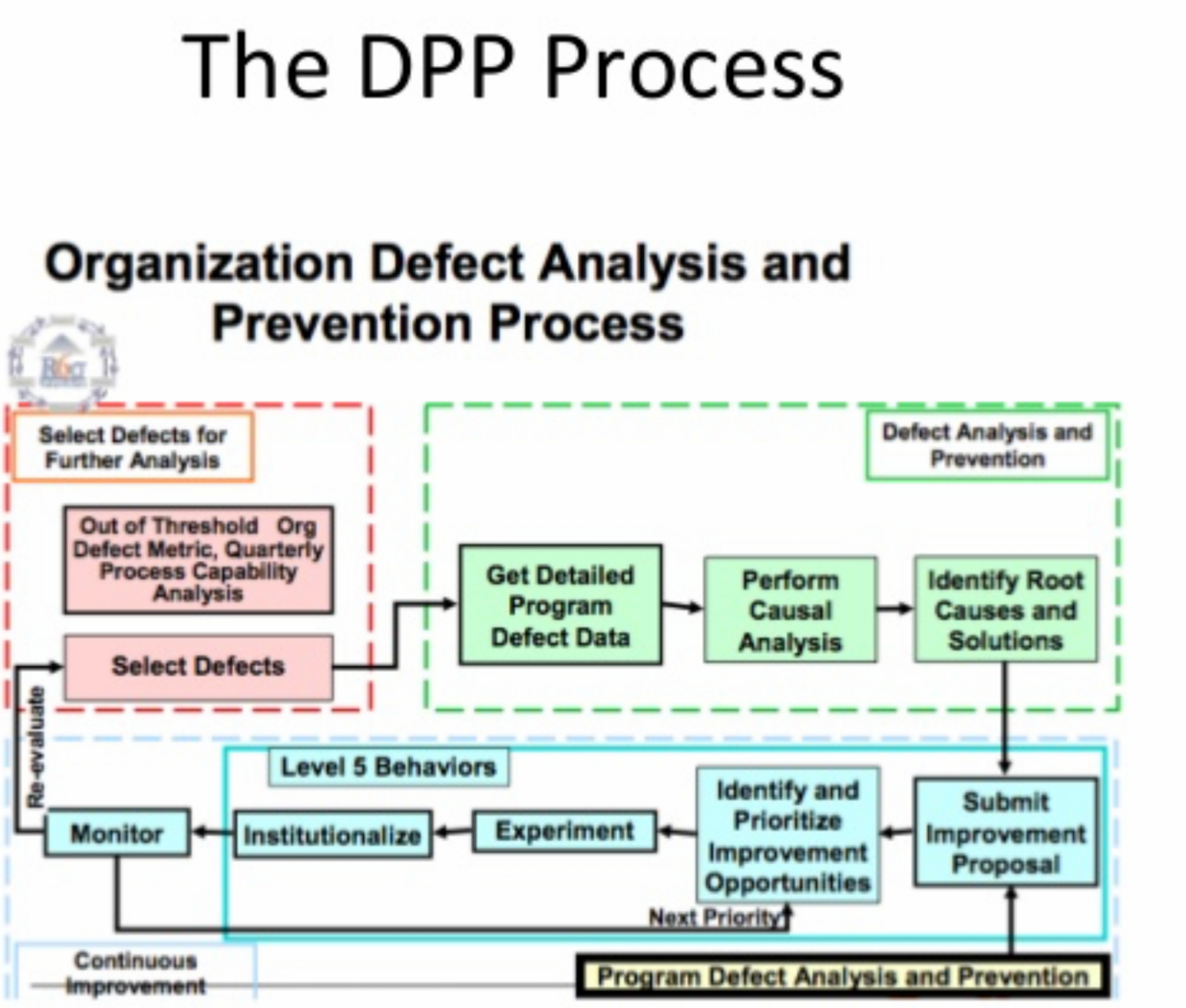

Figure 3.4 DPP Elements of the Defect Prevention Process

The Raytheon Values Achieved Using DPP[19]

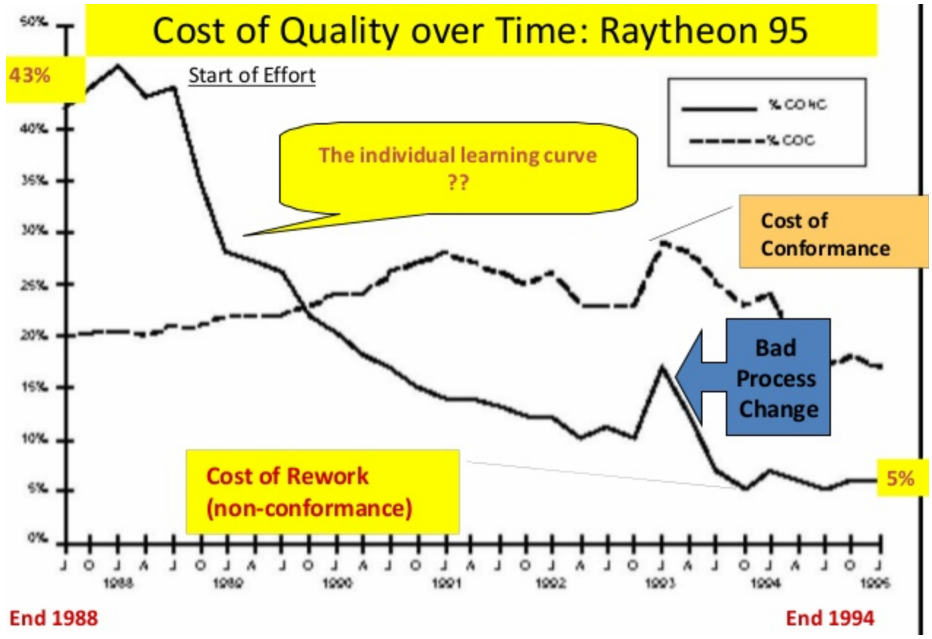

Figure 3.5 Cost of Q. As a result of DPP, the wasted effort, 'Cost of Rework', went down 10X

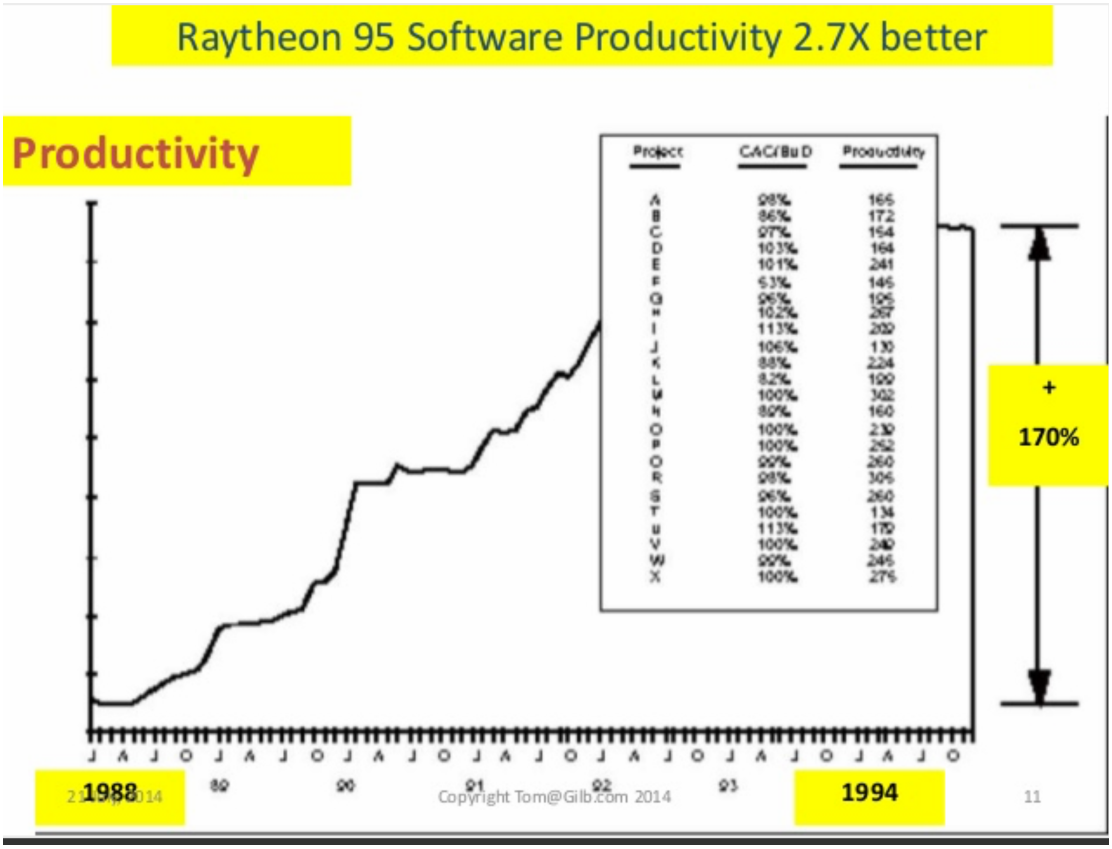

Figure 3.6 Productivity. As a result of DPP, productivity went up by 2.7X

[19] it is worth noting that the rate of return on investment for these improvements was 7.70 to 1

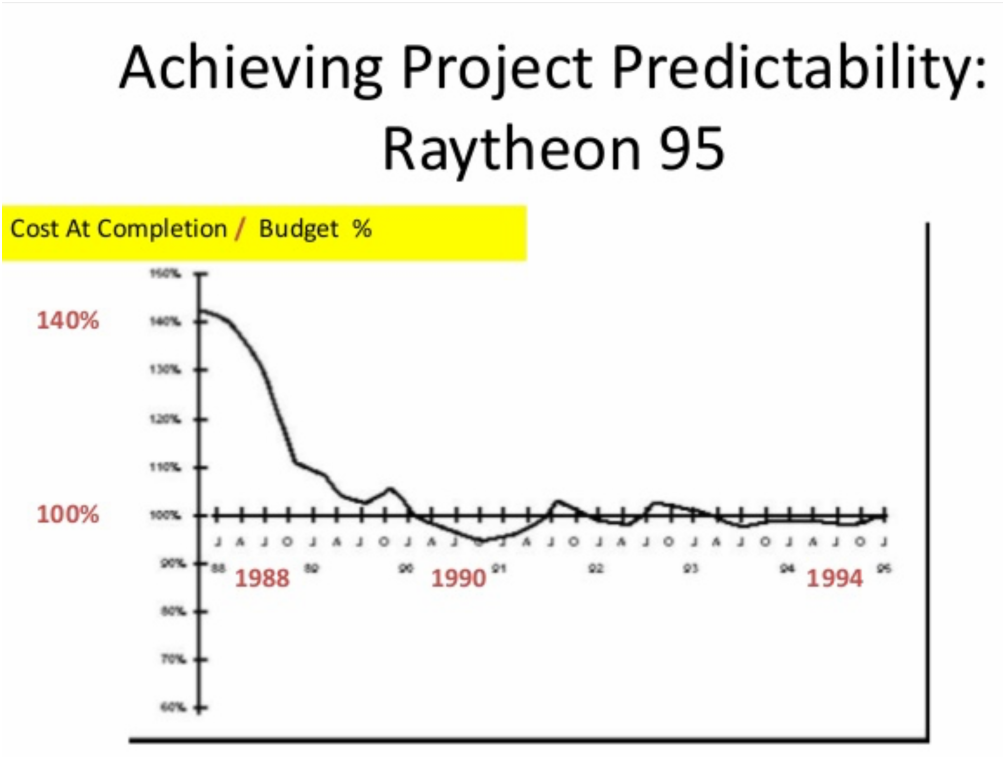

Figure 3.7 Predictability. As a result of DPP the ability to deliver on budget got much better.

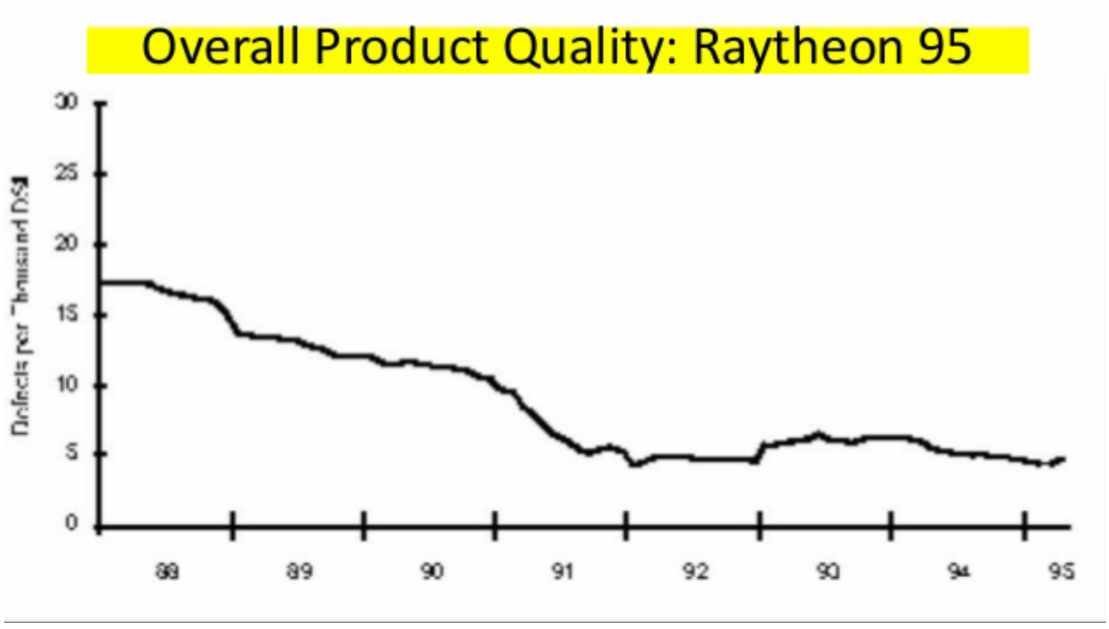

Figure 3.8 Product Quality. As a result of DPP the software bug released rate went down substantially.

QA Principles

1. 'A stitch in time, saves nine'. It pays off to tackle problems upstream, easily. And to *prevent* bad stuff, rather than to detect it *late*.
2. QA improvement is very-many small organizational tweaks, which need to be incremented gradually. The 'big idea' is to manage the QA-improvement process (like with DPP) as a stream of incremental improvements.
3. QA is not about 'testing', it is about *doing things right the first time*; not just as an *individual*, but because the *organization makes that possible*, enables good work, for all people.

QA Policies

1. We will guide Quality Assurance tactics by means of our top-ten critical, numeric, *organizational* value-objectives. These are our *definition* of our *organizational* quality values.
2. We will make good use of our grass roots professionals, to find practical process improvements for QA, and to try them out initially.
3. We will base our QA improvement efforts on a long-term process, of systematically getting better, and measuring better, and making sure the new methods are *really* embedded in the organization, *widely* and for the *long term*.

QA Checklists

1. Can you trace a long-term improvement curve, towards your critical, multiple, organizational, quality value-objectives?
2. Is CTO-Level management onboard, with quantification of critical objectives, for quality values? Is there long-term sustained support, and real action, to reach the goals?
3. Do you have a high rate-of-embedding the many small organizational improvements, like *at least one a week?*[20]

[20] IBM Minnesota had 2,167 changes in about 2 years. Steve Kan IBM Systems Journal 1994

The Reviewing QC Process.

Most review processes I know about, are weak to worthless. But a few are very effective, and *measurably* so. I am going to focus on our best known single process, but if you find better ones, let me know.

We[21] call it Specification Quality Control (Spec QC).

It works well, because it is simple, it is quantitative, and it can be as powerful as you want to make it.

It can be used to measure any written plans, and is especially useful on critical things like requirements. (and contracts, designs, test plans, software code, everything)

It has two basic modes:

- Checking that something is *clear*
- Checking that something is 'right'

The first test, 'clear', is a *prerequisite* for checking that the clear stuff is also *correct*; consistent with the facts and truth.

[21] Kai and Tom Gilb invented it for Citigroup, about 2003. Derived from decades of Software Inspection. We practiced versions of it informally before that, but formally defined it, and let it replace older Inspections at that point. Citigroup reported Defects per page down (82 to 10) within 6 months.

The basic SQC process is simple, we sample a document, and count the 'specification defects' ('Rule' (your standards) violations) which we find. This gives us a measure of pollution (of *not* following our organizations own Rules). For example, if the one Rule is 'must be clear', then all 'unclear words' are counted as 'defects'.

If there are 'too many' defects; we have to do something.

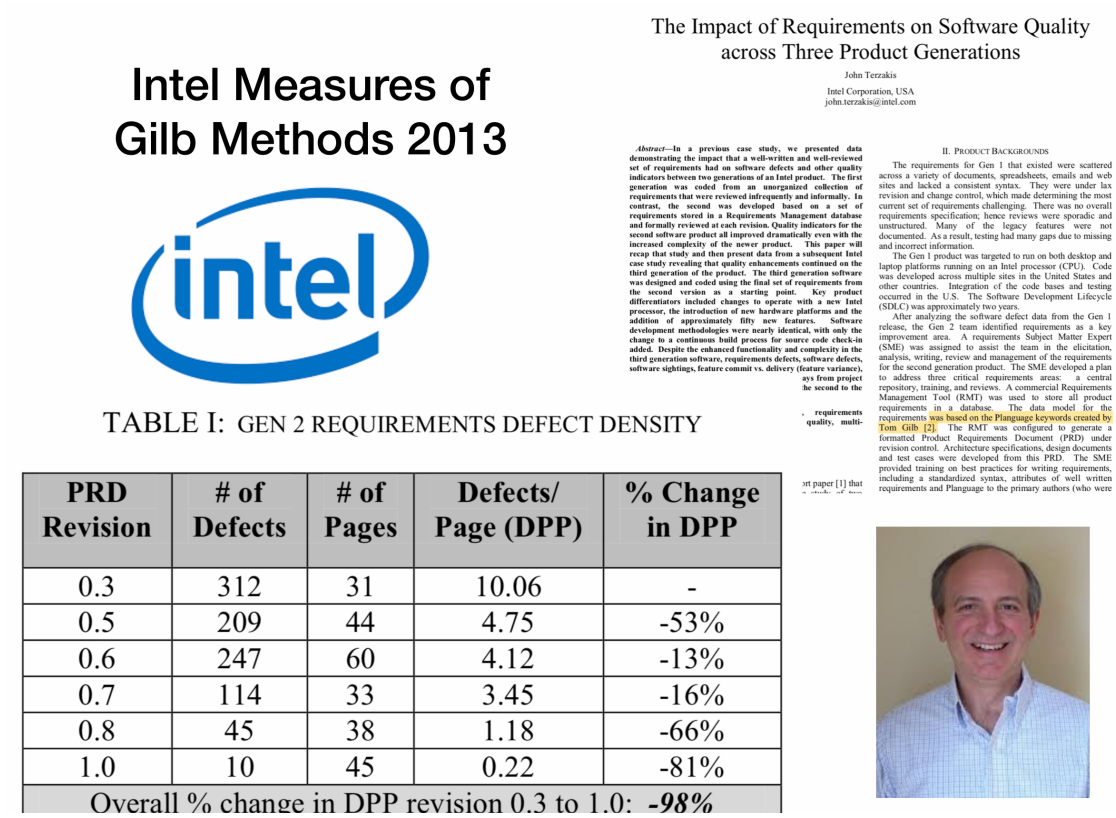

Figure 3 (Intel). Case study of using our methods for over 21,000 engineers at Intel. They use our requirements language 'Planguage' for clear specifications. And SQC to measure how well they write clearly. Source (ref. G)

In this example they initially measure, using SQC, 10 Defects per page (600 words here). This is initially 50X worse (10 DPP) than acceptable levels (0.2 DPP), so the requirements team has to rewrite and to learn to *follow the rules*, which they finally do. At that point, the requirements document can 'Exit' to the next working processes.

This measurement achieves the following:

- The team is motivated to learn the corporate standards in practice
- Bad quality specification does not ‘get approved, and then pollute the rest of the working process’.
- Intel reported 233% productivity improvement using these methods (they are not 'fighting pollution’ at late, costly, stages of work).
- The document can be released with a 98% reduction in defects compared with their first SQC submission. This is a *learning* degree, and it persists. People learn and remember how to do it right.

The Management Responsibility for QC

- make sure all serious specification work is quality controlled
- Make sure standards are defined for that type of work, which work effectively in practice (corporate learning)
- Make sure bad quality-level work does not escape to the next process, even under pressure of time (you will lose more time if you do allow premature escape)

None of these things will happen just because you employ educated professionals. You the *management* will have to lead and make sure it is in place.

Your guiding star can be

"Quantified

Value for Money"

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. Planning Principles.** https://www.gilb.com/offers/Shju4Zqn/checkout

(9) Technoscopes 2018. **TECHNOSCOPES**. 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) VR, 2019. Tom Gilb **Value Requirements**. Book manuscript written 3 July to 22nd July 2019. This book is a precursor text to the current book (Value Design) as it gets the Design Requirements ready for the design process. Value Requirements Slides . leanpub.com/ValueRequirements

3 Hour BCS VR course, 22 April 2020, **Slides** = http://concepts.gilb.com/dl970, Paper publications and translations are interesting to get done, so let me hear from you.

(13) VD 2019. Tom Gilb, **Value Design**. Manuscript drafted between 26July 2019 and 7th August 2019. Should be available digitally from Value Design BOOK: leanpub.com/ValueDesign VIDEO https://www.youtube.com/playlist?list=PLKBhokJ0qd3_wlvr0j85YhmNfNj8ZJ8M-, SLIDES http://concepts.gilb.com/dl972 VM 2019,

(14) Tom Gilb, **Value Management.** Manuscript start 7 Aug 2019, Good draft completed 16August2019. Major Edit 150321

Value Management book and course slides and video 2020

Value Management

This **book** Drafted August 2019, edit 150321, 250721 Leanpub.com/ValueManagement

Major edit done on this in 150321

Video 'Value Management', 2.5 hours, 13 May 2020, BCS, https://www.youtube.com/watch?v=mr9gUFWj4Jg

Slides, 'Value Management'= https://tinyurl.com/ValueMgtBCS, http://concepts.gilb.com/dl975

(15) **Systems Enterprise Architecture** (SEA) BOOK, Leanpub.com/SysEntArchBook, (2020)

My 2021 Meetup Series, Based on Systems Enterprise Architecture SEA Book (2020), Held in January Feb 2021 2x a week. 1 hours lecture + 30 minutes Q&A. The videos are now published on the OSWA youtube channel. All are available on this link:https://www.youtube.com/channel/UCb508f8y_d0PqRZKPCVVVxQ

(16) '**12?**: '**Twelve Tough Questions for Better Management**', leanpub.com/12ToughQuestions booklet 2020-1.

(17) **Governeering: Government Systems Engineering Planning**, leanpub.com/Governeering 2020-1

See related **Proper Public Planning Principles:**'Engineering Society', Responsibly, SLIDES = http://concepts.gilb.com/dl980 (pdf, 230620 VERSION), Video (90 min.BCS Lecture, 23 June 2020) = https://youtu.be/mIaVLHvQOp0

PAPER AND SLIDE REFERENCES

(A) Stakeholders.

Stakeholders
and their values
GilbFest 2017,slides
http://concepts.gilb.com/dl920

Paper:
Quantifying Stakeholder Values
INCOSe 2006
http://www.gilb.com/dl36

2016 Paper
Stakeholder Power:The Key to Project Failure or Success
including 10 Stakeholder Principles
http://concepts.gilb.com/dl880

Stakeholder Engineering, 2021, Book not sure where it ends up. Try Leanpub.com/StakeholderEngineering.
https://tinyurl.com/StakeholderBook
July 5 to 18 2021, Version 1.2 Complete Draft
Free Download

(B). **"The Happy Project Saboteur**
Principles of Project Failure: How to sabotage a project, without anyone noticing you."
https://tinyurl.com/HappyITSaboteur (to Gilb Blog)
http://concepts.gilb.com/dl955 (to Gilb.com Library)
by Agent 20-7 Version 050619

(C) 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

QUINNAN AND MILLS CLEANROOM
MIlls and Quinnan Slides
http://concepts.gilb.com/dl896
Excellent example of Early and great Value Agile 1970's!

(D) **HP**
Hewlett Packard Cases
HP Evo

AA. The Evolutionary Development Model for Software
by Elaine L. May and Barbara A. Zimmer
August 1996 Hewlett-Packard Journal
http://www.gilb.com/DL67

BB. Evolutionary Fusion: A Customer- Oriented Incremental Life Cycle for Fusion
by Todd Cotton
http://www.gilb.com/DL35

August 1996 Hewlett-Packard Journal

CC. RAPID AND FLEXIBLE PRODUCT DEVELOPMENT: AN ANALYSIS OF SOFTWARE PROJECTS AT HEWLETT PACKARD AND AGILENT (2001)
by
Sharma Upadhyayula
http://www.gilb.com/DL65

M.S., Computer Engineering University of South Carolina, 1991
And
Massachusetts Institute of Technology
January 2001

DD. Best Practices for Evolutionary Software Development
by
Darren Bronson
http://www.gilb.com/dl825

57 pages., 1999.

URI: http//hdl.handle.net/1721.1/80490

(E) DPP
Mays and Jones IBM SJ paper on Experiences
http://agileconsortium.pbworks.com/w/file/fetch/1527643/
Mays1990ExperiencesDefectPreventionIBMSysJ.pdf

(F) Raytheon Paper (2019 link)
https://figshare.com/articles/
Raytheon_Electronic_Systems_Experience_in_Software_Process_Improvement/6582863
DPP Experience.

(F) POWER TO THE PROGRAMMERS
“Power to The Programmers”, as held Krakow ACE Conference June 2014
Video: http://vimeo.com/98733453

Slides at Conference Prague
HTTP://CONCEPTS.gilb.com/dl841

(G) **Intel and Terzakis**
Intel Planguage Experiences
AA. Intel Report on SQC (Gilb methods used here <- E Simmons)
The Impact of a Requirements Specification on Software Defects and Other Quality Indicators by jterzakis@verizon.net
http://selab.fbk.eu/re11_download/industry/Terzakis.pdf
(SLIDES)

BB. Intel Experience with Planguage and SQC 2011
Erik Simmons, Intel, 2011, 21st -Century Requirements Engineering: A Pragmatic Guide to Best Practices, Erik Simmons PNSQC 2011 (Pacific Northwest Software Quality Conference)
http://www.uploads.pnsqc.org/2011/slides/Simmons_21st_Century_Requirements_slides.pdf

CC. This link gives Terzakis full Rio 2013 paper (Gilb annotated) and slides.
https://www.dropbox.com/sh/cs9hke3uvgg4gp3/AACadHeI95lZpHzVqGKXSXDra?dl=0

(H) **Proper Public Planning Principles:**'Engineering Society', Responsibly, SLIDES = http://concepts.gilb.com/dl980 (pdf, 230620 VERSION), Video (90 min.BCS Lecture, 23 June 2020) = https://youtu.be/mIaVLHvQOp0
See (17) Governeering book 2020

GLOSSARY

Concept Glossary[46].

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.

Architecture: a design process, producing a specification (The Architecture Spec) which is the top-level design process from a defined point of view, and which co-ordinates, or balances, all subsidary considerations of value, resources and constraints. (+ 290719)

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.

Backroom: the place where design ideas are readied for implementation.

---

[46] 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. It is built upon the Value Requirements book glossary of 220719. Adding concepts as necessary for the Design book.

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.

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 Component: any part of a larger design set, or architecture, which has some notion of independence. For example that it can be implemented incrementally. It can be removed or replaced.

Design Constraint: A requirement specification, that demands or forbids something regarding a design.

Design To Cost (D->C): a well-established engineering concept. You can find designs to meet a given cost requirement.

Design To Requirements (D->R): the combination, perhaps simultaneously in a single delivery cycle, of attempting to design to any set of both Value Requirement Levels, and Cost Requirement Levels.

Design To Value (D->V): the same concept as Design to Cost, except the design process is directed towards meeting a Value (including any quality) Requirement Level of Performance.

Downstream, Upstream: downstream refers to a process to be carried out at a later stage. Upstream, a previous process.

Dynamic Design to Requirements (DD->R): A Cyclical Design process, to meet any set of Value and or Cost requirements, but using measurement, after incremental design-implementation, comparing with requirements, predicting future cost and value levels, and re-designing, if necessary, to better reach the requirements. Note The Planguage Evo method (CE, PoSEM), and the IBM Cleanroom(2) Method both do DD>R. A term coined by Tom Gilb.

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.

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.

Frontroom: the place where design ideas are actually incrementally integrated into real systems.

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.

Implementation Responsible: a person (or group) which has taken responsibility for actual practical implementation of a design object. This can be for a requirement level (reach the requirement Goal), or for a design (deliver the design and try to get the maximum value from it).

Meter: a parameter which sketches major elements of a measurement process, for a particular Scalar Value or Cost.

Open-Ended Architecture: any architecture devices which make it easier to change the system through time.

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'

Quality Assurance: any process that contributes to meeting and maintaining quality levels, including all value levels. The primary emphasis should be on prevention of lower qualities than planned, not merely on detection and correction. Typically these are organizational and technical improvement processes, so that things lead to better quality. For example QA is engineering,

architecture, design, process improvement. Avoid misuse of this term to simply mean 'testing' .

Quality Control: checking any type of quality level, and related factors to make sure they are OK. Removal and correction of bad quality artifacts. Typically QC is reviews and testing processes.

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.

Risk: a risk is something that can go wrong. An opportunity is by contrast, something that can 'go right', get better. (+140819 tg)

Risk Dimensions: The risk *dimensions* are all critical *values*, and all critical *resource* limitations. The term 'Critical' by definition is a value or cost area, which is negative, to the potential extreme of total system failure, as a result.(+ 140819 tg)

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 Modelling of a system, and allows decomposition of problems so that ctitical 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.

Specification Owner: a person (or group) which has undertaken responsibility, by name, for the update and maintenance of a specification object, such as a requirement, a design, or a table.

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 for sale from May 2019 by Gilb International AS. It is based on Planguage and the Competitive Engineering book. Info at **gilb.com/valplan**

Value: value is perceived stakeholder

Value Analyst: analyzes stakeholder needs, and priorities, and selects critical, or possibly critical, needs and specified them as requirements, at least at the 'Wish' level (potential Goal requirement).

Value Architect: A person or team, who sits at the Apex of the system, and synchronizes all ongoing efforts in order to get maximum necessary value for available resources. Manages the top critical values, and the top level design architecture.

Value Contracting: this contracting can be done at both internal levels and external suppliers, and basically motivate suppliers to deliver value and get rewarded for it.[47]

Value Design Builders: people and organizations who build, physically, logically or organizationally, any design component or related activity.

Value Designer: a generic (all possible design areas) designer (or team) who undertakes to identify possible design components to reach a Value Requirement level, on time. To research them as to all side-effects and costs, documenting such facts in the design object and corresponding Value Tables. The Value Designer might hand over exploration of a design idea to a Specialist.

Value Director: the person, or group, responsible for focusing on the Value Delivery, and reporting to a steering committee or Board about the plans and accomplishments to date in Value Delivery.

Value Engineering Specialist: a designer with a narrow speciality (usability, security, performance, organizational improvement, AI) who is updated on the state of the art, and has a good international network of people and sources to find good specialist designs.

---

[47] Value Planning, Chapter 8 Delegation Outsourcing Contracting
https://www.dropbox.com/sh/eubo1zkvybl2q8k/AAD6cUUIOqco0aTPK2OZx6gua?dl=0

Value Policy: this is the written policy that gives clear guidance to the Value process, from organizational management. Perhaps Chief Technical Officer level.[48]

Value Process Manager: a person or team responsible for getting a best possible value stream flowing from the other people involved. Sort of like old project manager, except they are focussed on the Values/Costs numbers, not building stuff. They allocate resources (money, time), and assign people to specialist tasks.

Value Quality Control: these people carry out Specification Quality Control of specifications, to make sure the Defects Per Page is economically low enough before Exit to any other process. They are also responsible for measuring value levels and costs after incremental implementation. They will check that designs are in fact implemented as specified by suppliers for Exiting to integration delivery.[49]

Value Suppliers/Sub-contractors: internal and external to the organization people or organizations who undertake a specific responsibility, for construction, implementation, organizing, designing, or any other activity needed to deliver value.

---

[48] a considerable body of suggested Policies (over 100) is in every sub-chapter of Value Planning book. https://www.gilb.com/store/2W2zCX6z

[49]Value Planning Chapter 10 Quality Management
https://www.dropbox.com/sh/vjwybhqfxrvctk7/AAAdabECBSo5x-tSOI85R-1da?dl=0

BOOK MS EDIT NOTES

070819 STARTED AT WRITING DIGERUD CABIN (after finishing manuscripts for Value Requirements, and Value Design). This is sort of third in the Value Series 2019.

140819 RISK AND RISK DIMENSIONS GOT COMMA EDITED

160819: Good complete draft completed. Time to move on to next book, this Summer.

150321 Major all text edit and update newer books videos slides

added footnote better ref. As VD is not at gilb.com

Value Design

BOOK: https://tinyurl.com/ValueDesignBook

VIDEO https://www.youtube.com/playlist?list=PLKBhokJ0qd3_wlvr0j85YhmNfNj8ZJ8M-

SLIDES http://concepts.gilb.com/dl972

And in References added VM and VD BCS and Slides ,

And VReq book slides and video and new reference link added in references. Edited a lot of comma detail pages 20 on, added SEA book ref 15. Updated p25 12? Ref with new book.