Value Juggling:

How to keep all the Grenades in the air, without dropping any.

By © tom@Gilb.com Version 25-26 April 2017

Introduction

Realistic plans require you to manage a number of critical stakeholder values at the same time.

My definition of 'critical' says that failure of even one *single* 'critical' value, is essentially a threat to the entire system, product, or plan result.

Think of a juggler with 10 objects in the air, and each one is a live grenade, which will explode if it falls to the ground.

My experience is that most planners have no systematic education or training in dealing with this problem. Sometimes they think they know how to deal with it. Because they have lists of objectives they need to manage. But *naming the problems* is not the same as *solving* the problems. I believe there are a number of fundamental disciplines, that you will need to master, to avoid the grenade exploding.

This paper will outline the value juggling disciplines: and far more practical detail is available in my books [1] and website [2].

The Bare Necessities

Here, in my opinion, are the most basic requirements for becoming a Master Value Juggler.

We will supply more detail below, but lets declare the basics.

1. **Extreme Clarity of Purposes**: *Quantification* of *all* critical value objectives: no management BS allowed.
2. **Laser Focus on High Value Stakeholders**: *Rich modelling* of the problem environment; stakeholders, activities, qualifications, tasks, and much more. No oversimplified generalizations. Get real. Get practical. Prioritize.
3. **Agile Change to Critical Objectives**: Ability to *modify the problem statements* continuously to reflect reality and learning. Don't get stuck in false objectives, by fixed contracts and remote delegation.
4. **Solution Agility**: Ability to *freely select any useful solutions*, strategies, architecture based on their ability to satisfy your critical value objectives and constraints. Don't get stuck with the bosses favorite idea, when much better ideas are available, or become available.
5. **Value Decomposition Capability**: the ability to *decompose any strategy* or solution into small units of real measurable practical stakeholder value delivery, so that we can test ideas safely, change to better ideas early, and deliver very high value as an early stream.
6. **Value Motivation:** all parties to the project need to be *highly motivated* to actually *deliver* real measurable critical stakeholder *value*. You need various motivation systems, but they include supplier payment for measured value delivered, and bonuses for employee value delivered.

Is this enough for you?

I assume, if you are reading this at all that you are highly intelligent, highly ambitious, open minded, well educated and experienced.

Welcome! But here is the bad news. My experience with people like you is that your education, and corporate training have not given you any real knowledge of how to manage the Bare Necessities above, at all. You have *not got a clue* as to the answers I am talking about. No theory. No practical experience. No useful models of practice.

You tell me if I am wrong, after you learn more. I am sure some of you know some of these things. But I think there is a 95% 'blackout' of knowledge. I have worked with the best organizations worldwide for 60 years. Lectured at the finest universities. That is my evidence for these potentially insulting assertions.

But I am not here to insult you. Just reading this makes *you*, in my opinion, superior to 99.9% of other professionals. And my aim is to dramatically empower you to succeed where others fail. To help you be be a Super Juggler. I am going to do it for free too! Why? Just like you, I want to (continue to) 'make a difference', and you are my potential disciple to spread the good news!

But don't you be arrogant yet, about what you *think* you know. leave *that* arrogance to me for the moment, a privilege of age. To be frank, the amount of stuff 'I do *not* know about' is infinite, and I love to learn more of the useful type of stuff. I am also sure you know a lot of great ideas and methods that I would love to know more about. But share with me later, when you know what i don't know. I have put most of my *methods knowledge* in ten 300-800 page textbooks [1]. Not to mention slides, papers and videos. [2]

Now if you *really already know* how to do the Bare Essentials, you are done. Thanks for bearing with me thus far. I assume all your projects are 100% successful. Be generous and share your case studies with me and others. Lets compare notes.

But if you want to learn more, read this paper, and perhaps some of the references, especially my new Value Planning book [1]. I will give you a free link to the Value Planning book, if you promise to read it all in a month, and give me feedback. It is not properly published yet. But I have spent 2 years writing it, and 60 years learning what to write.

Value Planning is not simple, but it is the *simplest method that will consistently succeed*, and VP will consistently avoid embarrassing failure [3].

Failure is seductively simple, and most people seem to be in *value delivery failure* mode, though it is unpleasant to admit this failure, and recognize it. It is easier to blame 'some other reason' for failure than *you* and *your methods*.

The only excuse we recognize for failure is that you did not apply the Value Planning methods. We think **Zero Project Failures** should be the *norm*. But for decades IT projects have been at about 40% total failure plus 40% partial failure. And some methods like 'Scrum', brag publicly about only 19% failure! If my heart surgeon bragged that she only killed 1of 5 patients on the operating table, and had just done 4 straight non-lethal transplants, I would take my chances *without* her methods.

More Detail

1. **Extreme Clarity of Purposes**: *Quantification* of *all* critical value objectives: no management BS allowed.

Virtually no projects or organizations I have seen, and I have seen much of the best, except my trained clients, have quantified all their critical values [4]. Almost all of them have listed their top few critical values. But only 1 and at most 2 of those 'requirements' has any sort of a number attached to it, if at all. Most of them are slide bullet-points, about the great things that this project will deliver. Pure management BS [5].

They look like this:

*1. Central to The Corporation's business strategy is to be the world's **premier** integrated <domain> service **provider**.*

*2. Will provide a much more efficient **user** experience*

*3. Dramatically scale back the **time** frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to **generate** the desired **products***

*4. Make the system much **easier** to **understand** and **use** than has been the case for previous system.*

*5. A primary goal is to provide a much more **productive** system **development** environment than was previously the case.*

*6. Will provide a richer set of functionality for **supporting** next-generation logging **tools** and applications.*

*7. **Robustness** is an essential system requirement*

*8. Major improvements in **data quality** over current practices*

Figure 1: Real example of top level, C.E.O. approved, objectives, for a $160 million, 8 year, 90 people failed project. Source: a major engineering multinational. Using our advice it was turned around towards success, but it took 2 years to turn that Titanic, away from the iceberg. They had no clarity of purpose. So they built a system that delivered random and insufficient values. Nobody know what the required values actually were!

In 40 minutes, over a beer, I demonstrated what they should have done on the first day of the project [6]. Quantify and clarify all critical objectives. I tackled **Robustness** first, and decomposed it (Cartesian analysis, a 'good trick' to enable quantification of complex objectives).

- **{Software Downtime,**
- **Restore Speed,**
- **Testability,**
- **Fault Prevention Capability,**
- **Fault Isolation Capability,**
- **Fault Analysis Capability,**
- **Hardware Debugging Capability}.**

Figure 2: The decomposition of 'Robustness' into a potentially quantifiable set of critical sub-objectives. This is a partial definition of what we mean by Robustness.

I then used about 30 minutes to define in more detail, the first 3 of these, to give a structure and template for defining in more detail.

Testability:
Type: Software Quality Requirement.
Version: 20 Oct 2006-10-20
Status: Demo draft,
Stakeholder: {Operator, Tester}.
Ambition: Rapid-duration automatic testing of <critical complex tests>, with extreme operator setup and initiation.

Scale: the duration of a defined [Volume] of testing, or a defined [Type], by a defined [Skill Level] of system operator, under defined [Operating Conditions].

Goal [All Customer Use, Volume = 1,000,000 data items, Type = WireXXXX Vs DXX, Skill = First Time Novice, Operating Conditions = Field, {Sea Or Desert}. <10 mins.

Design Hypothesis: *Tool Simulators, Reverse Cracking Tool, Generation of simulated telemetry frames entirely in software, Application specific sophistication, for drilling – recorded mode simulation by playing back the dump file, Application test harness console <-6.2.1 HFA*

Figure 3. My draft of the 'Testability' objective. We are now using my planning Language 'Planguage' [1]. The Scale is the basic quantified definition. The 'Goal' quantifies the 'level' of successful value delivery. Notice the parameters (Volume, Type, Skill Level, Operating Conditions) which can be used to model many combinations of the real and very complex system. The 'Design Hypothesis' is part of their initial management Fluff specification. They were unable to clarify their real Goal, so instead they made a sexy list for 'strategies to play with', at great expense, and 'no value forthcoming'. They specified the 'means' and failed to clarify their 'ends'. Bad expensive mistake.

2. **Laser Focus on High Value Stakeholders**: *Rich modelling* of the problem environment; stakeholders, activities, qualifications, tasks, and much more. No oversimplified generalizations. Get real. Get practical. Prioritize.

Most projects have fuzzy objectives (Fig 1) at one-single level of concern only. They overgeneralize. The results are badly focussed value delivery to 'general public', or 'whoever'. Meaning, no interesting value to critical stakeholders. Mature planning cultures like Systems Engineering (INCOSE.com) analyze their stakeholders and stakeholder values in far more detail than typical business or IT planning- They may identify 50 to 500 stakeholders, each with one or more fairly critical, to them, values.

It is simpler to skip this 'detail' of the real critical stakeholders [7]. But this stakeholder oversight, by definition, (*critical* stakeholders and *critical* objectives) guarantees project failure; for those critical objectives and those stakeholders. Nobody knows what to deliver, and they will not do it by accident!

Analyze any project failure; and you will see that at least one cause is an unhappy stakeholder about a poorly-delivered value, or quality, or cost. Google 'project failure causes'. Requirements is usually top of the list.

In Planguage [1, 2] we start with serious stakeholder analysis. We tie critical stakeholders to every critical objective. We have special levels of objectives of the same type (like Security, Efficiency) for different stakeholders, at different deadlines, for different environments (tasks, product or service type, etc.)

We have laser focus on the real values that stakeholders have. And we are responsive to any critical changes in this values, for example with respect to levels, timings, areas. You can do that with *quantified* objectives. You cannot with the fuzzy objectives.

If you want more depth on stakeholders, get the digital Value Planning book [1] and search on 'stakeholder'. It permeates the entire book, and method, in depth.

The summary of the entire book is '**deliver real measurable stakeholder value now**'.

3. **Agile Change to Critical Objectives**: Ability to *modify the problem statements* continuously to reflect reality and learning. Don't get stuck in false objectives, by fixed contracts and remote delegation.

Many people assume, that because we like to have '*written* specifications', and like 'to *quantify* things', that our specification are **static**, carved in stone. Completely wrong deduction. We like 'written' and 'quantified' so that we can manage *subtle and rapid change* of specifications. [8]

What is the difference between very secure, highly secure, extremely secure, state of the art secure? What is the difference between

95% chance of detecting a Russian Hacked within 1 second by next year.
and
99.9 % change of detecting any Hacker within 1 picosecond by next week.

?

If you are not quantifying, and you are not written, you cannot expect to track critical and continuous critical changes!

The best practical example, of how this works, is when we use Planguage and the 'Needs And Means' tool [9]. The slightest change, for example 99.98 to 99.99 will immediately be reflected, just like in good spreadsheet technology, in the final summaries and recommendations. The plans can be updated by multiple parallel teams run in real time, with complete safety and transparency. (Concurrent Engineering). The exact sources of changes are well documented.

Figure 4. A Planguage table of Values and strategies with numeric evaluation of how efficient the strategies are, with respect to risks ('Worst Case'). From a student exercise for a real startup in London. 2017.

4. **Solution Agility**: Ability to *freely select any useful solutions*, strategies, architecture based on their ability to satisfy your critical value objectives and constraints. Don't get stuck with the boss' favorite idea, when much better ideas are available now, or become available.

People get frozen in to 'early ideas' about how to solve their Value Objectives problems. They focus on building the new system, rather than focussing delivering the actually desired values. They focus on the perceived means, rather than their true ends. These are bad habits, bad culture and bad management. That is what happened in the large project failure cited in Figure 1 above.

Solutions need to be tried out early, and on a small scale. If solutions work, then scale up, and try out new elements of the solutions. But if they do not work in delivering expected value, or in costs and timing, then it is time to admit defeat early. Change or modify strategies, and find some ideas that actually work.

The early masters of this were at IBM Federal Systems Division, using the Cleanroom Method developed by Harlan Mills [1] with Robert Quinnan as the realtime (each delivery step) 're-architecter'. They were measuring *value delivered* at each 2% delivery stage, and modifying architecture, or other possible parameters, so that high 'space and military' quality was 'always delivered on time, and under budget', for years in a row, without exception.

This (Cleanroom, IBM, [11]) is a really good public example of *zero project failure*. Yet it was in the most-difficult lowest-bidder fixed-deadline, state-of-the-art quality environments.

This is the best successful example 'Agile as it should be'. It is essentially the same set of methods we are talking about in this paper, and in Value Planning [1, we call it 'Evo' there].

We have to develop, or adopt, new culture where we *keep focussed on the values* and costs, and *quickly, frequently, early* - try out solutions. Keeping what works well, adjusting what needs adjustment, and dumping bad solutions very early. It seems to be logically impossible to get a large scale embarrassing failure this way. If we do not change culture, the decades of failures will continue. *"Those who do not learn from history are doomed to repeat it."* (Santayana 1903)

The *worst case we can experience, with this method,* is a 'very small failure' (2% of resources wasted) before we recognize the problem and fix it, or stop - if there is no way forward. But we avoid large-scale waste of time, money, and the lack of 'having the necessary values delivered for use'. The methods and culture people use today have 300% (of initial budgets), and worse, waste of time and money, before they wake up to big public scandals.

Harlan Mills, IBM, Cleanroom Method developer.

5. **Value Decomposition Capability**: the ability to *decompose any strategy* or solution into small units of real measurable practical stakeholder value delivery, so that we can test ideas safely, change to better ideas early, and deliver very high value as an early stream.

There will usually be a *top level* of your set of strategies, for meeting a given set, or given level of stakeholder values. These are things like: buy in, enhance, add to the product or system or organization. Sometimes called *architecture*.

I have found that IT architects in Europe specify architecture, for example for a large banking system. But they do *not* (99% of the time) estimate the value that will be delivered, nor do they specify the various cost aspects [10]. The result is that they are *flying blind*, and going to crash.

Part of the problem is that management almost never hands the IT Enterprise Architects, a clear quantified set of value objectives. So, GIGO (Garbage In, Garbage Out) results.

The architects do not even know exactly which values they need to satisfy.
They do not try to satisfy defined values, or to estimate how well their architecture will do anything.

The result is that the foolish Enterprises will implement the architecture ideas at once, without much chance to evaluate the results, or to retract, or to modify, the architecture or strategy. Failure is inevitable. Since the blind (architects and project managers) are leading the blind (manager, enterprises) - and they are consistently failing world wide, and 'nobody seems to notice it'.

The fact that people with management education (MBA etc.) pervasively allow this 'failure of the architecture' to happen, tells me that they do not understand management at all. Common sense might be useful in that case, but it does not seem to be used by the managers either.

The solution to this problem is to decompose the strategies (architecture, solutions) by a factor of about 50 (2% increments) so that we can incrementally deploy a smaller part of them, and see what happens. Are they effective in finding hackers 95% of the time ? Do they blow the entire budget alone?

If *small increments* deliver *measurable value* at *estimated costs*, then we are safe, for the moment. Continue the process. By the way, we generally prioritize high value-to-cost increments. So a good side effect of this *cautious trialing* of 'big ideas', is that we get an *early flow of value*: long before the *entire* big architecture idea is implemented.

Sounds like a common sense idea, right? What is the problem? Why doesn't everybody do things that way?

The reason is that they for the most part have no theoretical knowledge, no culture, no experience in decomposing big ideas by value-delivery increments. Some people will succeed in decomposing by value-delivery increments, if you simply ask them to. But most people are never asked by their management or organization to 'deliver real value, early; as a stream of value'.

I have collected 20 principles of decomposition, and written 100-page chapter on Decomposition in the Value Planning book [1]. For those who want systematic methods for value decomposition.

But I have found a fairly simple method that works when training people or working with clients. Ask them to divide their big architecture ideas into about 10 sub-ideas. But they must ensure that the sub-ideas are both 'independently of each other implementable', and that when implemented; they can expect to deliver 'some measurable value' towards our defined quantified objectives. This works. Most people can do this. Sometimes with a bit of coaching, on the meaning of 'measurable value', and 'independently implementable'.

6. **Value Motivation:** all parties to the project need to be *highly motivated* to actually *deliver* real measurable critical stakeholder *value*. You need various motivation systems, but they include *supplier payment for measured value delivered*, and *bonuses for employee value delivered.*

Everybody claims to want value from plans and projects. But we constantly pay suppliers and employees for doing work, or supplying products and services, that do NOT deliver value.

This a crazy. But it is the norm.

In fact, on failed projects, with delays and overruns, and no value delivered yet whatsoever, we actually *reward* suppliers and employees with *more money to continue failing* to deliver value.

This is irresponsible use of shareholder and taxpayer money. Managers who persist in such waste deserve criminal penalties; as for any other but perhaps smaller, thieves, embezzlers and corrupt people.

There are many attempts to contract for value, but in large-scale US Department of Defence contracting, 'Earned Value Management' is not a great success. It is misused as 'spent money = Earned value', and *real value management* is not really managed. But some people are trying to find a way.

The proposals by two of my professional friends **www.flexiblecontracts.com** seem to me to be a sound practical framework. But we do not yet have experience with them to verify that. [12].

The idea is simple enough. You have a legal overall framework. But you make a new deal about measurable value, not just *financial* value!, and the price for each small increment of delivery. For example every week or month. That should keep people focussed on value. But it almost seems like there is no reward for people managing value successfully; so people stick to the conventional failure methods.

And of course most people do not master the bare essentials: the ability to quantify their value objectives, and the ability to decompose into these small 2% increments.

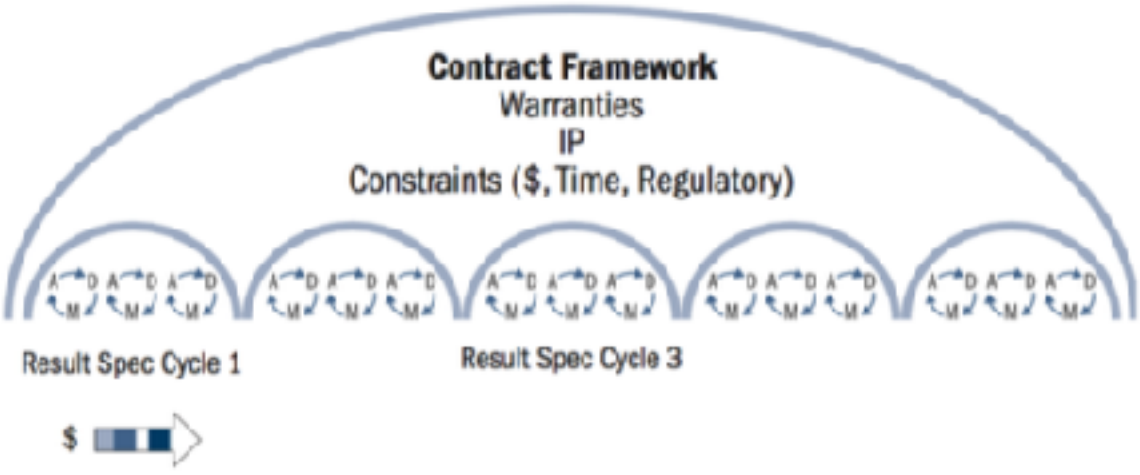

Figure 5. The *flexible contracts* framework. [12]

Summary:

Your real project must simultaneously juggle at least a dozen critical values objectives and costs simultaneously.

Succeeding in delivering all of the values, within all budgeted resources, is very difficult, with ambitious competitive objectives, and limited resources.

There are some basic methods, discussed above, which when combined, will enable you to master value delivery. This will help you move towards 100% project success. But you will have to learn a number of disciplines, which are not yet common practice, or understanding.

That is your necessary sacrifice, if you want to get ‘super powers’ in delivering real stakeholder values efficiently.

If you stick with the currently popular, and pervasive, methods that have led to persistent frequent value-delivery failure for decades, you have wasted your time reading this paper. Sorry!

References

[1] **Value Planning**, the digital book
Value Planning book 5 minute video, and book details
https://www.gilb.com/store/2W2zCX6z
with a purchase offer.
OR
leanpub.com/ValuePlanning

[2] gilb.com. Gilb Website many free papers, slides, cases, videos, blogs.
Including free pdf of my book Competitive Engineering, which defines Planguage and Evo.
https://www.gilb.com/p/competitive-engineering

[3] 'Too Simple','Methods should be a simple as possible for delivering value, but no simpler.' http://concepts.gilb.com/dl903

[4] **Top 10 Critical Values**
Gilb's Mythodology Column
The Top 10 Critical Requirements are the Most Agile Way to Run Agile Projects
http://www.gilb.com/dl554
and
Project Success, every time! http://www.gilb.com/dl835
Novatec Workshop 051214

[5] **Quantifying Management Bullshit**: forcing IT Stakeholders to reveal the value they really want from your IT Project. http://www.gilb.com/DL465

[6] **Project Startup Week**.
An Agile Project Startup Week
www.gilb.com/dl568

[7] a. **Quantifying Stakeholder Values** http://www.gilb.com/dl36. This paper focusses on how to determine project stakeholder needs/requirements/values. It shows how to use Planguage to specify them quantitatively. This is the basis for deriving corresponding and supporting product qualities.
b. **Stakeholder Power**: The Key to Project Failure or Success. http://concepts.gilb.com/dl880 , 2016

[8] **Fear of Numbers**: the cure might be numbers themselves - used intelligently.
http://concepts.gilb.com/dl904, 25 April 2017

[9] www.NeedsandMeans.com
see this site for extensive examples and video tutorials etc. A great way of keeping track of changes in complex systems planning.

[10] What's Wrong with Software Architecture
Keynote Sw Architect Conf London 10 Oct 2013
http://www.gilb.com/dl603

Slides in pdf, https://www.youtube.com/watch?v=HNasoyrxzy8
the 1.5 hour talk. 3 of 300 said they specified costs of architecture, 1 of 300 participants said they specified effects on values like security.

[11] Mills and Quinnan, **IBM Cleanroom**. 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

Mills, Harlan D.; Dyer, M.; and Linger, R. C., "Cleanroom Software Engineering" (1987). The Harlan D. Mills Collection. http://trace.tennessee.edu/utk_harlan/18

[12] **Agile Contracting for Results;** The Next Level of Agile Project Management: Gilb's Mythodology Column Agilerecord August 2013
http://www.gilb.com//dl581

Agile Contracting for Results
Smidig 2014 Oslo Ten minute talk slides
http://www.gilb.com/dl832
The 10 minute Video at Agile 2014 Oslo
http://vimeo.com/album/3107510/video/110735250