Value Delivery in Systems Engineering

in support of my Talk at Quality Days, Vienna, January 18, 2016

"Engineering Quality into Software: Quantifying All Qualities in Requirements, Architecture and Project Management"

Tom Gilb
Tom@Gilb.com

Abstract.

Sponsors who order and pay for systems engineering projects, must justify their money spent based on the expected consequential effects (hereafter called 'value') of the systems.

Most values are 'qualities' (defined as 'how well' a system functions. But we cannot focus on qualities alone. We must manage *all* values and costs. [Value Planning].

At one extreme if a system met all technical requirements, but was never deployed in practice – it might have no possibility of delivering the value expected. This paper will argue that the definition of the expected value should form an integral part of the high level requirements of the system. It will argue that we need specific design and implementation planning to improve the probability that the value will be delivered and will be maintained.

The Value Delivery Problem

Sponsors who order and pay for systems engineering projects, must justify their money spent based on the expected desirable consequential effects (hereafter called 'value') of the systems.

The value of the technical system is often expressed in presentation slides and requirements documents as a set of nice-sounding words, under various titles such as "System Objectives", and "Business Problem Definition". But the problem with these is that:

• their source or authority may be undocumented and unknown
• they are probably not at all clear about exactly what will happen, where or when, or under which conditions
• there is no contract, to pay only upon such results being delivered
• there is no specific design or architecture, to enable the technical product to achieve these goals

For example: (Real, engineering system, but doctored for anonymity)

1. Central to The Corporations 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 (see rewrite in example below)

8. Major improvements in data quality over current practices

The above example was the basis in 1999 for a project that had in 2006 spent over $100 million, for 8 years and had never delivered any value whatsoever to the corporation. There was never any quantified or testable definition of the above. There was never any direct link from the project activity, requirements, or architecture, to these primary top management (CEO and next level directors) objectives. The project was doomed from the start.

Another Real (Doctored) Example: Financial Corp.

1. Reduce the costs associated with managing redundant / regionally disparate systems.

2. Single global portfolio management system.

3. Reduce overall spending with a reduction in redundant initiatives.

4. Governance structures - system agnostic.

5. All projects in project portfolio system.

6. Reduce development project spend on low priority work with better alignment between Technology and business demand.

7. Project portfolio Framework, Business Value metrics for prioritization.

8. Reduction in cost over runs.

9. Definition criteria for project success.

10. Metrics and exception reporting for cost management.

11. Linkage of actual costs to forecast.

12. Increase revenue with a faster time to market.

13. Knowledge management, project ramp up templates.

This project spent about $50 million, in a single year. Responsible management, impatient for some results, discovered to their horror, through an audit, that the above primary objectives had never been clarified or taken seriously. The responsible ('former') project manager had chosen to ignore the opportunity, planned by a major component supplier, to clarify these objectives. The project manager spent a lot of effort obtaining requirements from users, but no further effort on *these* primary objectives above. Serious effort was, after the audit, then immediately spent quantifying and taking seriously these objectives. It took a single day to draft a quantified version. The quantified version made a clear distinction between technical objectives (system quality – examples 2 and 5 above) and stakeholder values (making the business better, examples 8 and 12 above).

The experienced reader will recognize this type of 'objectives', because they are universally written so badly. All large projects seem to have them. Maybe I just never get to see the projects that take this top level of 'requirements' seriously. I suspect we have a serious management culture problem. Management does not seem to have training or culture to quantify the top level critical variable 'values' of the project. They fail to even distinguish between their management ends and the technical means.

The purpose of this paper is to bring the problem to the surface, and discuss remedies for those who want to do better.

Some Assertions

Let me make a series of assertions about the problem and its solution.

Assertion 1. When top management allows large projects to proceed, with such badly formulated primary objectives, then they are responsible as managers for the outcome (failure). They cannot plead ignorance.

Assertion 2. The failure of technical staff (project management) to react to the lack of primary objective formulation by top management is also a total failure to do reasonable systems engineering. Management might have a poor requirements culture, but we should routinely save them from themselves.

Assertion 3. Both top managers and project personnel can be trained and motivated to clarify and quantify critical objectives routinely. But until the poor external culture of education and practice changes, it may take strong CEO action to make this happen in your corporation. My experience is that no one else will fight for this.

Assertion 4. All top level system performance improvements, are by definition, variables. So, we can expect to define them quantitatively. We can also expect to be able to measure or test the current level of performance. Words like 'enhanced', 'reduced', 'improved' are not serious sys-

tems engineering requirements terms.

For Example:

I rewrote the top level system requirement in the above example using Planguage [Gilb 2005]:

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

to be:

> **Rock Solid Robustness**:
> **Type**: *Complex* Product Quality Requirement.
> **Includes**: {Software Downtime, Restore Speed, Testability, Fault Prevention Capability, Fault Isolation Capability, Fault Analysis Capability, Hardware Debugging Capability}.

Example 1a: I decomposed the complex attribute Robustness into a reasonable set of 7 contributing attributes. I then proceeded to define 3 of them initially, as in the examples below. 'Rock Solid Robustness' a now formally defined system attribute, is defined as a set of 7 sub-attributes.

> **Software Downtime**:
> **Type**: Software Quality Requirement. **Version**: 25 October 2007.
> **Part of**: Rock Solid Robustness.
> **Ambition**: to have minimal downtime due to software failures <- HFA 6.1
> **Issue**: does this not imply that there is a system wide downtime requirement?
>
> **Scale**: <mean time between forced restarts for defined [Activity], for a defined [Intensity].>
>
> **Fail** [Any Release or Evo Step, Activity = Recompute, Intensity = Peak Level] 14 days <- HFA 6.1.1
>
> **Goal** [By 2008?, Activity = Data Acquisition, Intensity = Lowest level] : 300 days ??
> **Stretch**: 600 days.

Example 1b: the key parameter here is the 'Scale' of measure. All parameter concepts, such as 'Fail' are formally defined in Planguage [Gilb 2005, Glossary also at www.gilb.com]. In this initial draft, I have not defined all possible or useful parameters such as 'Supports', or 'Meter'.

> **Restore Speed:**
> **Type**: Software Quality Requirement. **Version**: 25 October 2007.
> **Part of**: Rock Solid Robustness
> **Ambition**: Should an error occur (or the user otherwise desire to do so), the system shall be able to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.
>
> **Scale**: Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.
>
> **Initiation**: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps] 1 minute?
Fail [Initial and all subsequent released and Evo steps] 10 minutes. <- 6.1.2 HFA
Catastrophe: 100 minutes.

Example 1c: The 'Ambition' level statement is a high level, not too quantitatively rigorous, summary of the requirement level we want. It is a healthy agreed prelude to more-rigorous definition. We often use the statements made poorly by management, and cite them as the source or authority. We then process with Scale and Goal etc. to define in an engineering manner.

Testability:
Type: Software Quality Requirement.
Part of: Rock Solid Robustness
Initial Version: 20 Oct 2006
Version: 25 October 2007.
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*

Example 1d: notice the parameterized Scale of Measure ('[Volume]', '[Type]'), and the corresponding definitions in the Goal statement ('Type = WireXXXX vs. DXX'). This allows us to define highly reusable generic scales of measure. But we can both in the scale and in the many types of benchmark, constraint and target levels on that Scale, we can get very precise about performance levels for a specific set of conditions (when, where, and 'if' (an event is true)).

Notice the 'Design Hypothesis'. The initial specifications, after a poorly defined requirement ('robustness') massively (about 24 pages for the 8 top level requirements) specified (as if it were a requirement) this sort of 'design'. My point here was to show how my client could initially deal with the outpouring of technical design ideas, at the requirement stage, by declaring them to be mere 'hypothesis'. Designs to be examined by estimates and testing later. But not to be considered 'required'; as seems to be the initial ($100 million spent on the design) case.

Note on the example: I did this example in front of, and with the help of, 3 client engineers, while having a beer at a Corporate Conference Hotel, while waiting for my wife to get dressed for a Dinner. It took about 45 minutes. The point was to show how easy it would have been to set clear top level requirements; eight years earlier. A director at the same conference revealed to me that he was the only dissenting voice amongst top management, he voted to not start the project, and specifically because the initial requirements were unclear. He was over-ruled by the

others, including the current CEO. Does that tell you something about the management problem we face in getting clear top-level project requirements for the essential values to be delivered?

Assertion 5. If the hardware/software systems supplier is not prepared to deal with the system level that delivers the value from their product, then someone, internally or an external contractor needs to undertake the project of delivering the value expected.

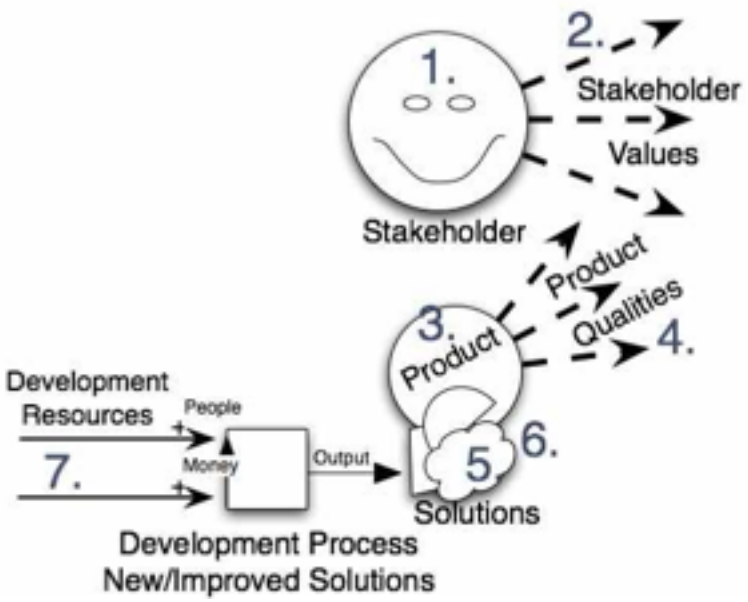
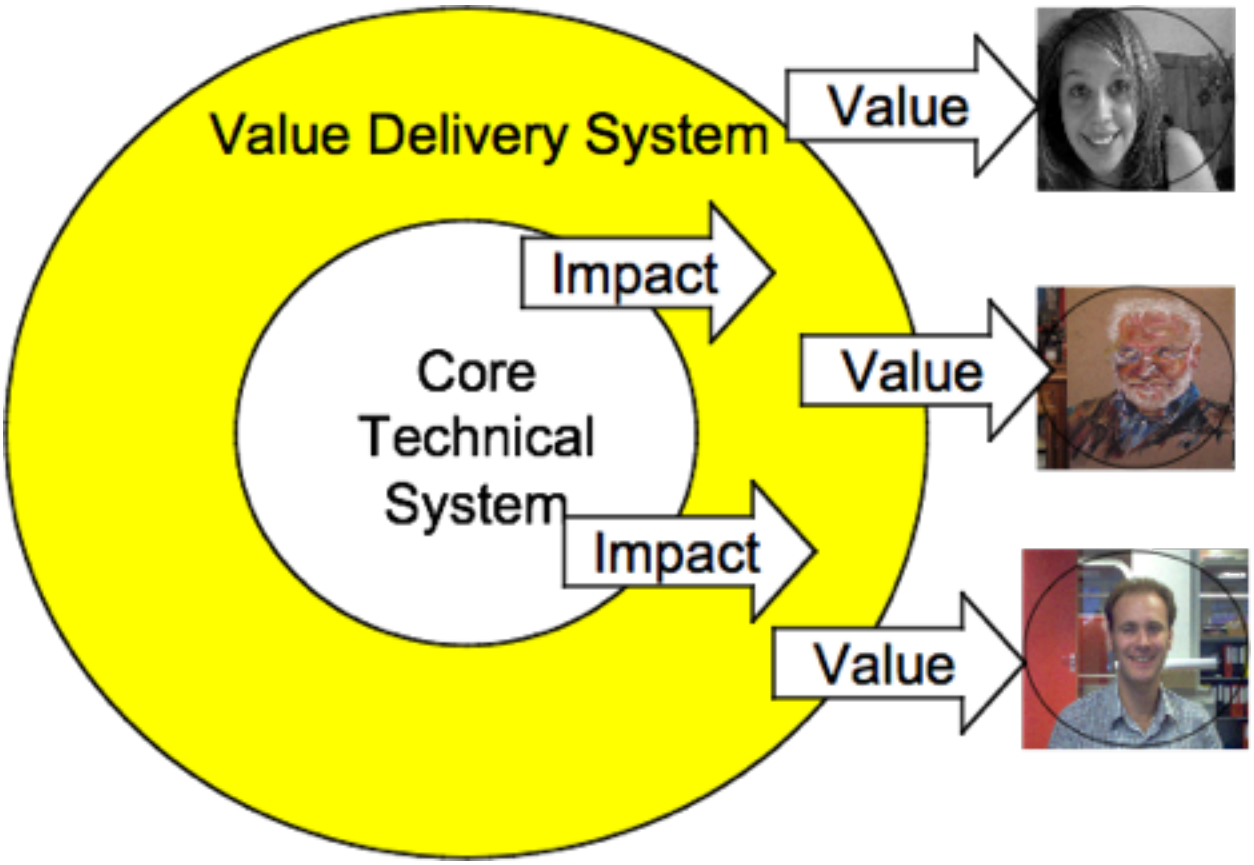


Figure 0: (7) Development Resources are used to run the development process. The Development Process develops new improved (5) Solutions with enhanced (4) Product Qualities. When the Stakeholder uses the new product with the enhanced (4) Product Qualities it improves on their (2) Stakeholder Values. Source: Kai Gilb, www.gilb.com [Evo book and Glossary].

Assertion 6. This 'value delivery process' is likely to entail considerable human and organizational aspects, and little hardware and software technology. So it may be inappropriate work for systems engineers who are not expert in, and committed to, the social, political, and organizational aspects of systems engineering. But of course this 'social' ability is a necessary and valid component of full systems engineering – or we cannot call it 'systems' engineering and exclude the social, political system aspects.

Figure 1: The Value Delivery System: some level of systems engineering has to take responsibility for final delivery of expected value to stakeholders.



Do we need a Chief Value Officer (CVO) ?

We seem to have a Chief Technical Officer, a Chief Financial Officer, a Chief Information Officer. But we seem to be missing someone with primary responsibility for delivering value to the organization and its stakeholders. Maybe we need a Chief Value Officer (CVO) to help the CEO in this responsibility. It seems strange that we build technology that all too often does not deliver the value that responsible management has imagined it would.

The CVO would:

• be responsible for the 'value accounting system' in the same way the CFO is responsible for the cost accounting system, and budgets.

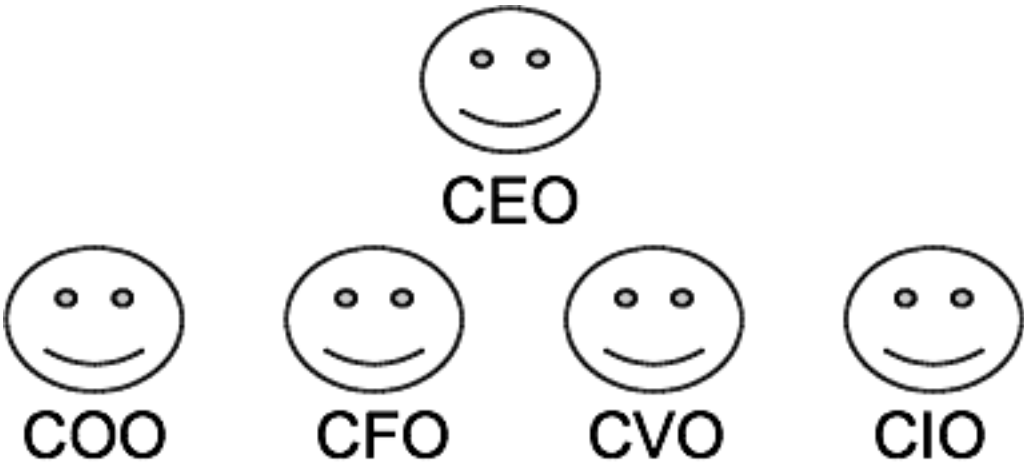
• make sure that project investments had clear and valid value arguments (Value Budgets)

• make sure that all levels of management and technology knew how to specify values, design for value, measure value, contract for value, and deliver value in practice.

Figure 2: do we need a CVO to make sure value is taken seriously? Up to now only the CEO has been effective in making this happen, in my experience. The other CxO's do not consider it part of their job.

A Value Policy: Without a CVO (yet)

The Value Manifesto:



- Really useful value, for real stakeholders will be defined measurably. No nice-sounding emotive words please.
- Value will be seen in light of total long term costs – as a decent return on investment.
- Powerful management devices, like motivation and follow-up, will make sure that the value for money is really delivered – or that the failure is punished, and the success is rewarded.
- The value will be delivered evolutionarily – not all at the end.
- That is, we will create a stream of prioritized value delivery to stakeholders, at the beginning of our value delivery projects; and continue as long as the real return on investment is suitably large.
- The CEO is primarily responsible for making all this happen effectively. The CFO will be charged with tracking all value to cost progress. The CTO and CIO will be charged with formulating all their efforts in terms of measurable value for resources.

The Value Principles:

1. Value can always be articulated quantitatively, so that we can understand it, agree to it, track it, contract for it and understand it in relation to costs.

2. Value is a result, delivered to a real set of stakeholders.

3. Value must be seen in light of lifetime total cost aspects, and must be as profitable as alternative investments.

4. Value occurs through time, as a stakeholder experience: it is not delivered when a system to enable it is delivered – only when that system is successfully used to extract the value.

5. Value can be delivered early, and for part of one stakeholder's domain. This proves the value potential, and actually improves the real organization.

6. There is never a really sufficient reason to put off value delivery until large-scale long-term investments are made. This is just a common excuse from the many weak, ignorant, cowards who would like to spend a lot of money before being held to account.

7. People who cannot deliver a little value early, in practice, cannot be entrusted to deliver a lot of value for a larger investment.

8. The top management must be primarily responsible for making value delivery happen in their organization. The specialist managers will never in practice take the responsibility, unless they are aiming to take over the top job.

9. Value is a multiplicity of improvements, and certainly not all related to money or savings – but we still need to quantify the value proposition in order to understand it, and manage it.

10. If we prioritize highest value for money first, then we should normally experience an immediate and continuous flow of dramatic results, that the entire organization can value and relate to. Be deeply suspicious of long-term visions with no short-term proof.

The Value Principles with some technical background

1. Value can always be articulated quantitatively, so that we can understand it, agree to it, track it, contract for it and understand it in relation to costs.

We constantly see value ideas, like the examples at the top of this paper, articulated with a series of words. "Enhanced Agility".

My experience is all such 'improvement ideas' can always be expressed quantitatively (Gilb CE esp Ch 5). Most all managers can themselves, if asked, come up with suggested scales of measure. And they can negotiate agreement on suitable scales of measure, for almost anything. Common sense and experience are largely enough to get quantification ideas. But if imagination runs out Google "<the name of your objective> metric", and that usually gives you what you need.

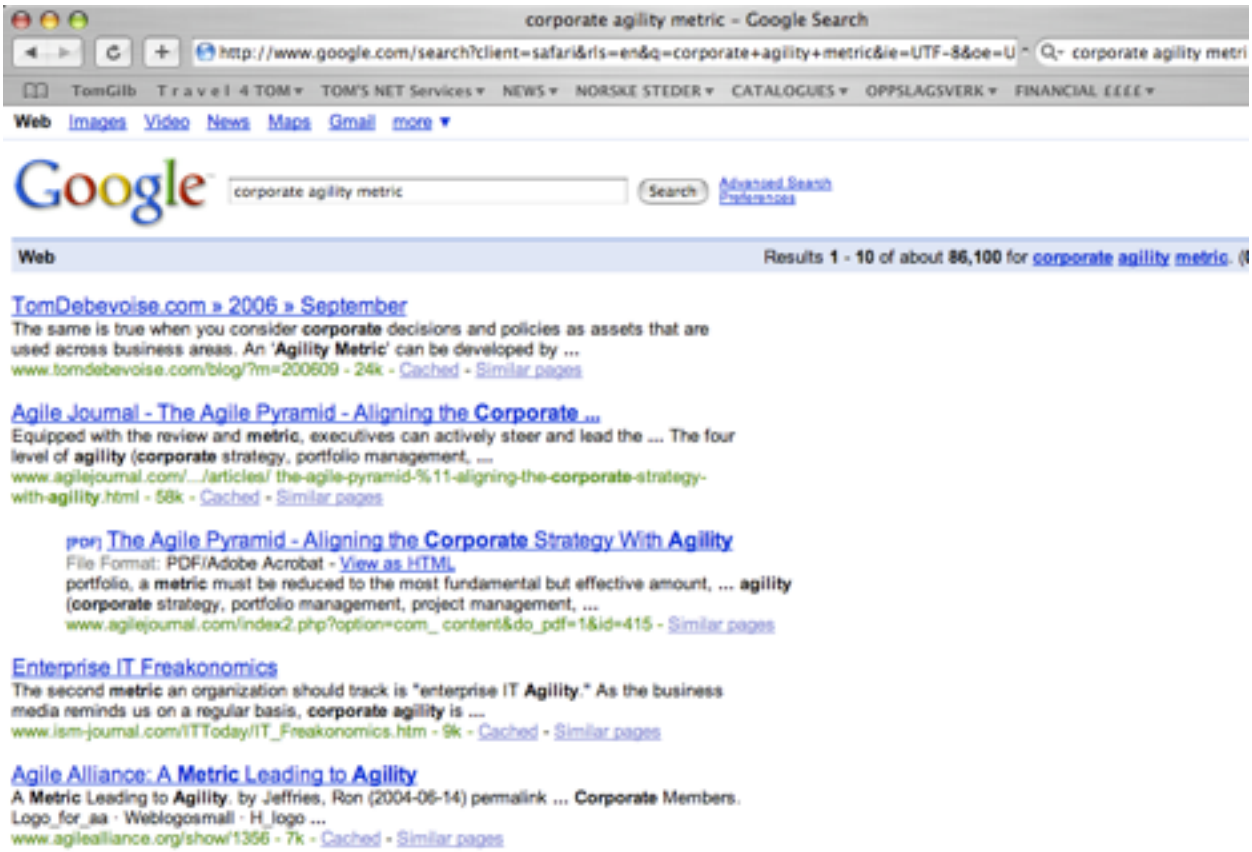


Figure 3: letting the internet advise you on quantification of anything

2. Value is a result delivered to a real set of stakeholders.

Value is not 'activated' by a technical performance characteristic alone, like Usability, security or Robustness. It is only created when it meets real people in their everyday stakeholder situation of work: Call Center, Battlefield Analyst, Corporate Trader. It has to save them time, or make their work better. The value created by the interaction with a stakeholder type may be cumulated every time the system is used for some new activity, customer, transaction, or decision. It may be cumulated by a very large number of that type of stakeholder (10,000 sales people). And through a very long time (years).

It is obvious from this common sense observation that value is *not* created by the technical system performance characteristics (speedy response, user friendly), but by making those technical system characteristics available in practice to as many real people, and as many transactions, and for as long a time as possible.

3. Value must be seen in light of lifetime total cost aspects, and must be as profitable as alternative investments.

We cannot allow ourselves to be blinded narrowly by quantified value. We must constantly estimate, and manage the value for money: the return on investment. And if the costs of deliv-

ering the value get out of hand, and exceed the value – it is time to either reengineer the system or decommission it. Who will do this if not some constant CVO vigilance?

4. Value occurs through time, as a stakeholder experience: it is not delivered when 'a system to enable it' is delivered – only when that system is successfully *used* to extract the value.

A conscious strategy, and conscious formal plan, must be made to deploy a technical system so that the value is delivered. We have to deal with political problems – like power centers (trade unions, management fiefdoms) and economic waste centers. We have to motivate people to give up their comfortable older systems and deploy scary new ones. We have to support the correct use by training, call centers, local consultancy, measurement and

- feedback on the technical system, is it actually delivering what we need, in order to get people to use it at all, to use it well?
- feedback on the stakeholder environments it is deployed in: are they happy with it? Do they have improvement suggestions? Are there undesired variations in costs and benefits?
- feedback on deployment to the entire scope of stakeholders, in relation to time plans: is it being deployed successfully rapidly enough?

Obviously this should be the natural concern and use of true systems engineering. But in fact, there is little in the training, the conferences, the handbooks [INCOSE SE Handbook], to verify that systems engineering as a discipline has matured to the point where these concerns are safely included. We are still too much 'engineers' (techies); and know and care too little about value management, and the organizational and management culture part of our domain.

5. Value can be delivered early, and for *part* of one stakeholder's domain. This proves the value potential, and actually improves the real organization.

Our systems development culture is still very much a 'waterfall' culture. Finish the big system, and then deploy it [INCOSE SE Handbook 2-3, and 3-2 for example]. There was no visible mention, in the Handbook, of a true evolutionary life cycle (even though the US DoD adopted one for software at least long ago, DoD Mil Std 498). There is no notion of early, frequent and gradual delivery of results to stakeholders, even though that has been practiced successfully in many large military, space and software systems for decades [Larman]. Big Bang is still our mentality.

I helped Douglas/Boeing to do value delivery Evolutionary projects for 25 aircraft projects in 1990. It was an unknown concept for them, but it was easily doable by every team we did it on; in real projects. We use 'next week' as our measure of when we would produce some useful value.

I know that this sounds incredible and impossible to conventional ears. But it is simple enough in practice, and very close indeed to weaponry progress during the Second World War [Discovery Channel!].

A Navy helicopter ship system, called LAMPS, provides a recent example.

LAMPS software was a four-year project of over 200 person-years of effort,

developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and a ship,

in 45 incremental deliveries.

Every one of those deliveries was on time and under budget.

A more extended example can be found in the NASA space program,

where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million bytes of program and data for ground and space processors in over a dozen projects.

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

Source: Harlan Mills [IBM Systems Journal No. 4, 1980, p. 415], Reprinted IBM SJ Vol. 38 1999, 289-295. Internet available both.

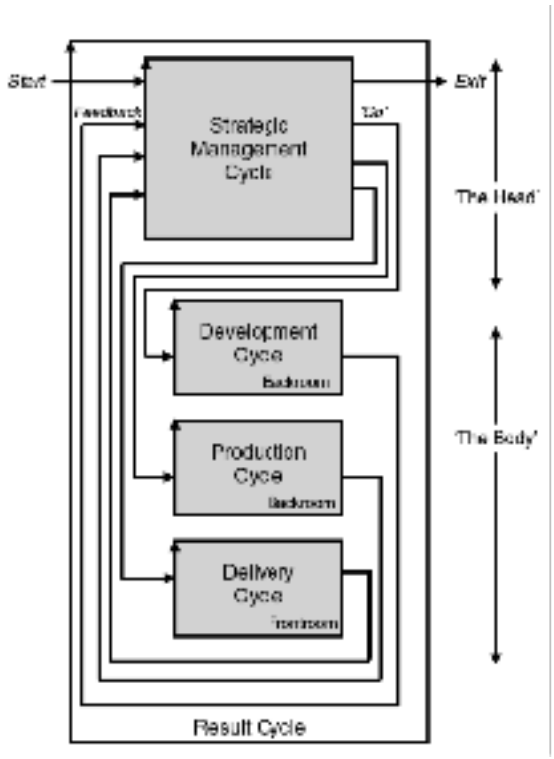
Quotation: we have extensive systems engineering experience in evolutionary delivery of systems, but it is not reflected in what is taught, written or known in most of our SE culture today.

Figure 4: The kind of systems engineering life cycle we should be using normally to deliver value early and frequently. And to avoid the scandalous failure rates of projects [Morris]. Source Gilb 2005, Evo Chapter 10.

6. There is never a really sufficient reason to put off value delivery until large-scale long-term investments are made. This is just a common excuse from the many weak, ignorant, cowards who would like to spend a lot of money before being held to account.

There are vested interests who will happily consume public and private corporate money forever and deliver failure or little or no real value. The consumer and their representatives seem happy to contract for *effort*, but not contract for *value*. I cannot believe there are so many foolish people with so much money as I have had occasion to observe in practice (example the $50 to $100 million wasted projects at the beginning of this paper, which are in fact small by comparision with some; like documented DoD waste in software engineering alone ($20 billion annually, many years ago).

This is not necessary! We could avoid it by contracting for value and results. [Gilb, No Cure No Pay]. This is hardly on the agenda, and not discussed at all in the INCOSE Handbook.



It would require two technical pieces of knowledge:

- The ability to quantify and measure value
- The ability to decompose large projects into much smaller increments of value delivery.

These exist, but the 'will to contract for value' does not. Some management leadership please!

7. People who cannot deliver a little value early in practice, cannot be entrusted to deliver a lot of value for a larger investment.

Ericsson of Sweden, who learned to deliver mobile telephone base stations in 1990 in monthly evolutionary steps observed this principle (Jack Järkvik). If you are going to spend $100,000,000 before anything happens, and nothing then does. It might have been a good idea to offer the project or supplier a mere $1 million (1%) and ask if they could create some of the long-term projected value for that 1% of budget. If they cannot, then there is no reason to believe they will use your $100 million wisely. If they can; do so, then feed them millions, one at a time until it is no longer profitable!

8. The top management must be primarily responsible for making value delivery happen in their organization. The specialist managers will never, in practice, take the responsibility, unless they are aiming to take over the top job.

Top management, the CEO, needs to decide they are primarily responsible for value for money, and dictate a policy of focus on 'value for money' (see earlier in this paper for policy ideas).

One excellent CEO client of mine who did so, Robb Wilmott of ICL UK (23,000 employees then), turned years of losses into 14 straight years of profit for his computer company – unlike competitors, like IBM, at the time. My observation was:

• it only happened because the CEO threatened all other top managers with loss of power and budget if they did *not* 'quantify the value' they were going to deliver

• they began to think clearly about their responsibilities, perhaps for the first time

• it helps if the CEO is an engineer, not an MBA ☺

Another UK CEO, pulled the same trick – about 2003. But had to fire the marketing director, and the sales director, for refusing to really play ball. Some directors have a real fear of being specific about what they are responsible for. Interestingly the current Chairman of *this* company was one of the above-mentioned ICL Directors (Marketing) who we trained to quantify, things like the primary new product line vision, 'Adaptability' of his product.

9. 'Value' is a multiplicity of improvements, and certainly not all related to money or savings – but we still need to quantify the value proposition in order to understand it, and manage it.

I strongly dislike value schemes that try to turn all values into money. Do they really think management understands no other concept?

Peter Drucker, I think it was (Management By Objectives, in 'The Practice of Management'), established long ago that no corporation is driven by money alone. Thus the Balanced Scorecard, to retain some non-financial balance, I suppose.

If the value you are aiming at is for example, 'increased potential customer willingness to short-list you', then there is an estimable money value for that, but I would be afraid of losing focus on the short-listing, by converting this idea to money. You would need to measure the quantity of real short-listing to manage that value, for example. I believe you need to state and measure things directly, especially of you want to track early lead indicators of value – and keep people focused on a dynamic and changing situation.

10. If we prioritize highest value for money first, then we should normally experience an immediate and continuous flow of dramatic results, that the entire organization can value and relate to. Be deeply suspicious of long-term visions with no short-term proof.

We should try to skim the cream off the top. With early realistic feedback, and changing technology and markets, we should be able to avoid a dramatic diminishing return on investment for some time.

Projects, at one extreme, should be practically self-funding; or at least not in need of huge initial budgets, then overspent by factor 3.14 (Pie instead of 'piece of cake') before management feels uncomfortable.

You have a lot of choice, in spite of some dependencies, to 'cherry pick' very high value for money, early deliveries. Not exactly a new marketing technique – but maybe alien to our Defence Supplier Systems Engineering mentality.

Again, if we contracted to pay them for value for money, they would be more focussed on making it happen. This is *our* problem, not theirs. We fail to motivate suppliers to do the right thing for us.

We fail to even discuss this in our systems engineering literature. We have progress payments, but not based on value delivery, early and frequently. 'Payment Schedules' (sounds nice and bureaucratic) are mentioned in the SE Handbook, but not 'Value Payments'. We need to extend the concept!

Summary

Top management needs to change their culture to manage the actual delivery of real value, and not leave it to systems engineers to drive this change. Systems Engineers can execute the value engineering and delivery – but only top management can make it happen.

References

Gilb: **Value Planning: Practical Tools for Clearer Management Communication**
LeanPub.com/ValuePlanning. Published 5 January 2016

Gilb, Tom, Competitive Engineering, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, ISBN 0750665076, **2005**, Publisher: Elsevier Butterworth-Heinemann. Sample chapters will be found at Gilb.com.

CE Chapter 5: Scales of Measure:

http://www.gilb.com/community/tiki-download_file.php?fileId=26

CE Chapter 10: Evolutionary Project Management:

http://www.gilb.com/community/tiki-download_file.php?fileId=77

Gilb.com: www.gilb.com. our website has a large number of free supporting papers , slides, book manuscripts, case studies and other artifacts which would help the reader go into more depth

Gilb, Tom, 'No Cure No Pay: How to Contract for Software Services', INCOSE Conference 2006.

http://www.gilb.com/community/tiki-download_file.php?fileId=38

INCOSE Systems Engineering Handbook v. 3

INCOSE-TP-2003-002-03, June 2006 , www.INCOSE.org

Note: I pursued all 68 uses of the word value. I would conclude that I could find no explicit reference to the stakeholder value derivation process I am speaking about here. 'Stakeholder value' gave no references (stakeholder alone 139 references. Section 4.2 Stakeholder Requirements Definition Process. Show laudable traditional SE understanding of stakeholders and their requirements. But does not make any remarks about deployment of technical systems over time to get the value that might come from the technical requirements. It is weak and general with too many of its brief statements ("• Avoid acceptance of unrealistic or competing objectives."). The main problem being that it does not clearly distinguish between the technical system requirements, and the consequent value delivery stream in stakeholder organizations.

Larman, Craig, and Vic Basili. "Iterative and Incremental Development: A Brief History". IEEE Computer, June **2003**
http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf

Morris, Peter, The Management of Projects, 1994 Telford, London, 1997 USA.

Peter Morris analyzed the degrees of failure of 5 decades of civil engineering projects. He concluded that 'there is no good project management method', and if there was one, it would be very iterative with feedback.

Biography

BIOGRAPHY

Tom Gilb is an international consultant, teacher and author. Living in Norway and UK.

His Tenth book is 'Value Planning', 2016. His 9th book is '**Competitive Engineering**: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (2005,) which is a definition of the planning language 'Planguage'.
He works with major multinationals such as Bosch, Qualcomm, HP, IBM, Nokia, Ericsson, Motorola, US DOD, UK MOD, Symbian, Philips, Intel, Citigroup* and many others. See www.Gilb.com .

* none of whom are the source of the examples in the paper.

Edit 7 January 2016