Engineering Productivity:

Some ways to measure it and manage it.

Tom Gilb
Tom@Gilb.com

Abstract. There are often too few qualified engineers. I am mostly referring to product design engineers – software engineers and systems engineers. One reason we have too few is that we misuse their time so badly – we waste at least 50% of it. But when we can longer desire or afford to solve the problem by hiring or off-shoring to get more warm-bodies, we need to consider getting more productivity from the engineers we already have. There is one great advantage from that tactic – they already have plenty of *experience* in *our* company! There are several tactics to improve productivity. They can take many years to come to full effect, but a steady long term improvement, and dramatic short term improvement, should be possible. The key idea in this paper is that we can define our *own* productivity quantitatively – and manage the improvement of it quite systematically. Your *own* definition of productivity demands several simultaneous dimensions of productivity. The definition of productivity also requires substantial *tailoring* to your organization, and to its current environment. I am going to assert that the best short term measure of engineering productivity is *agreed value (requirements) delivered*; and the best long term measure of engineering productivity is *stakeholder benefits actually delivered*.

The Engineering Productivity Principles:

Here are some basic suggestions for a framework for getting control over engineering productivity:

1. **Subjective Productivity:** Productivity is someone's subjective opinion of what values we want to create for our critical stakeholders.

2. **Measurable Productivity:** Productivity can be defined as a set of quantified and measurable variables.

3. **Productivity Tools:** Productivity can be developed through the individual competence and motivation, the way we organize people, and the tools we give them.

4. **Avoid Rework:** The initial attack on productivity improvement should be reduction of wasted effort

5. **Productive Output**: The next level of attack on productivity should be to improve the agreed value delivered to stakeholders.

6. **Infinite Improvement:** Productivity improvement can always be done: there are no known limits.

7. **Perfection Costs Infinity**: Increasing system performance towards perfection costs far more than increasing volume of system function.

8. **Value Varies:** Product attributes are viewed and valued quite differently even by members of the same stakeholder group.

9. **Practice Proves Productivity:** You cannot be sure how well a productivity improvement strategy will work until you try it in practice

10. **Productivity Dwindles:** Yesterday's winning productivity tactic may not continue to work as well forever.

Defining Productivity

Let me tell you what I think productivity *is*, maybe even what 'engineering' is.

Productivity is **delivering promised value to stakeholders**.

'Deliver' means actually measurable handed over and available to stakeholders.

'Promised' means that clear written agreements, are made in contacts, requirements, documents and slides, or clear undeniable expectations are set.

'Value' means something of perceived use, to the stakeholder; they need it, they want it, they

are willing to sacrifice resources to get it, they will be unhappy if it is late or lower in power than their expectations.

'Benefits' are the results of the perceived value to stakeholders. Benefits are what *really* happens, though time, as a result of the engineering value delivered.

It is an open question whether systems engineering should attempt to take some planning responsibility for enhancing benefits realization, or whether this is the system recipient stakeholders that should be responsible for planning an environment to maximize benefits. Someone has to take this responsibility, and I fear that the system users with their 'day jobs', do not feel they are responsible or capable. In which case an opportunity for systems engineers, to enlarge their conventional scope of planning, exists.

So, we can simplify and say 'engineering productivity' is the **ability to deliver agreed requirements**.

Our formal **requirements**, should ideally be the 'meeting place' for stakeholder values and engineering commitments.

An engineer is **productive** to the degree they contribute to an engineering effort that is successful in delivering promised requirements, to real stakeholders, in a timely manner (at or before agreed deadlines).

An engineer is more **efficient** if they can reduce the resources needed to deliver requirements on time to stakeholders.

Stakeholders are any people, groups of people, types of people, or instances that have requirements (like laws, contracts).

Engineers are technical people who, as a team, master the arts of

- determining a necessary set of *requirements* for a system,
- determining a necessary set of *solutions*, and
- planning and carrying out the necessary processes to actually *delivering* the promised requirements (the value, the potential benefits) to the stakeholders.

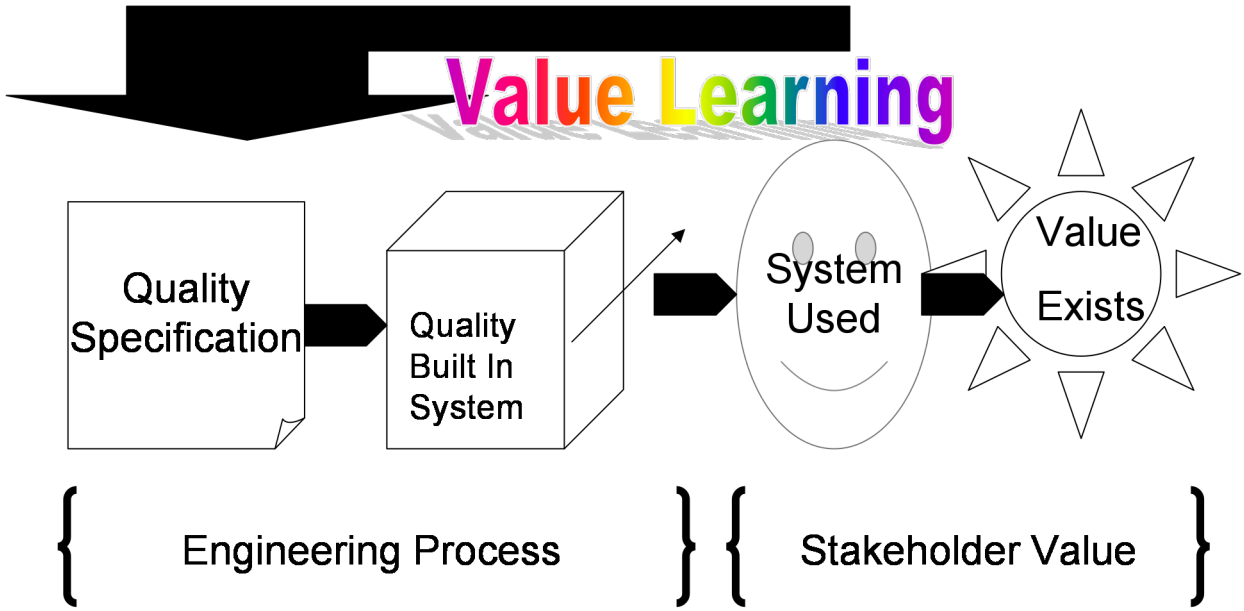

*Illustration: engineers can be productive by generating the conditions for stakeholders to get value from the system. The question is, does the **systems** engineering responsibility stop at the technical system? Or, should it extend into the stakeholder domain? Should systems engineers at least plan (engineer) everything necessary to get the intended value in practice? Is it 'good enough' that value perception exists, but the benefits are* not *finally brought in, in practice? The next diagram adds a stage, regarding bringing in the benefits.*

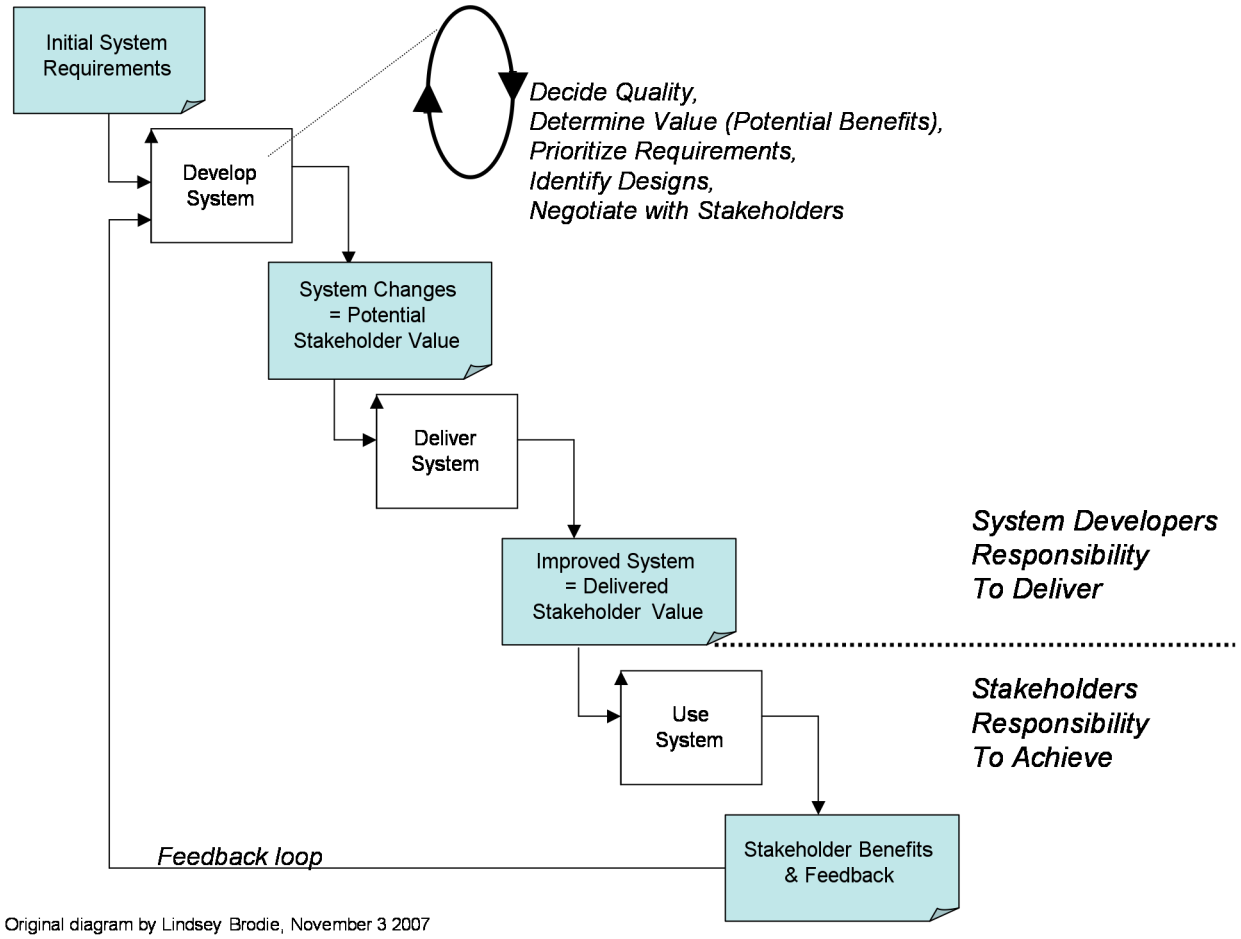

Illustration: this diagram makes the subtle distinction between handing over 'potential value' systems to stakeholders, (perhaps this is the limit of engineering responsibility?) and, then actually achieving the full long-term benefits that system deployment enables the stakeholder to do. The rectangle with a left arrow up, is a PDSA process, a Planguage symbol for a process in general.

What Engineering Productivity is *not*.

1. **Not Zero Results:** any failure to actually deliver the *value* agreed, no matter what the reason, or source of cause, means that the engineers have failed to be productive (even if it is not their 'fault').

2. **Not Specs**: productivity is not the ability to *generate* specifications of any kind. Specs are perhaps a necessary 'means', but the 'value' delivered is the key notion of real engineering productivity.

3. **Not Exceeding Value**: productivity is not *exceeding* agreed requirements, if there is no value, and no agreement with stakeholders.

4. **No Golden Hammer:** there is no one tool, method, principle or policy that will give you full-potential productivity: there are masses of details, and persistent improvement, and maintenance forever, that are necessary ingredients.

Some ways to measure engineering productivity

Direct Measures

Value Delivered:
% Lifetime Value Actually Delivered.

This is a summary of all measured or estimated real value delivered to real stakeholders for a defined time period, usually to date. This is % of plans made, of requirement targets that were set.

Potential Value Extrapolation:
% Lifetime Benefits Estimated achievable, under given conditions, based on real measurement and deployment to date.

This is our best estimate of the capability of the system to deliver planned benefits in the longer term, based on real experience of some real stakeholder deployment thus far. The set of future conditions for reaching these estimates, such as budgets, and access to skilled engineers and managers, willingness of stakeholders to continue use, market conditions; need to be spelled out clearly. If prudent, then steps need to be taken. to ensure those conditions are true, as far as we can exercise control over them.

Indirect Measure and Indicator

Technical Capability:
% of Target-Level Improvement of Performance Requirements that is Measurably Delivered

This indicates that the technical engineering work is succeeding. It does not measure that the technical capability has been converted into stakeholder *value* (deployed at the stakeholder). It could be that the technical system is not yet deployed to stakeholders, except in pilot versions.

Some strategies to increase engineering productivity

Primary Strategies for Value-Delivery Productivity

1. Measuring Value as a strategy

It is all too common, in the many international industries I am personally witness to, that many of the acknowledged critical factors that determine value are not expressed in quantified terms. This seems to be a problem for both management and engineering cultures. We are taught a selection of metrics, for accounting and engineering, but we are not taught that all critical factors must be dealt with quantitatively, even if we have to invent suitable metrics. Senior managers and engineers are not taught, and they do not know *how to quantify* the very factors they have just acknowledged are critical to the project at hand. They use words, but not numbers.

Examples of **real**, fuzzy, critical, top level, project objectives

Technical Goals: "rock-solid robustness", "to 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 by semi-automating and/or performing these activities as the data comes in", "to make the software much easier to understand and use than has been the case for previous software", "to provide a much more productive software development environment than was previously the case.", "software development difficulty should scale", "will provide a richer equipment model that better fits modern hardware configurations", "Minimal down-time", "major improvements in data quality over current practices wherein the job planning process is much more haphazard."

■ *Business Systems: "Business Result Alignment: maximize delivery speed and client satisfaction level across the Change the Firm Book of Work to achieve key business goals.", "Eliminate IT efforts that duplicate other IT efforts.", "Make use of existing tools and avoid reinventing the wheel", "Deliver high-significance real-time metrics on critical aspects of project results and resources.", "to be the world's premier integrated service provider" (in our sector).", "a much more efficient user experience"*

Engineering Organization Objectives:

A special effort is underway to improve the timeliness of Engineering Drawings. An additional special effort is needed to significantly improve drawing quality. This Board establishes an Engineering Quality Work Group (EQWG) to lead Engineering to a breakthrough level of quality for the future. To be competitive, our company must greatly improve productivity. Engineering should make major contributions to the improvement. The simplest is to reduce drawing errors, which result in the AIR (After Initial Release) change traffic that slows down the efficiency of the manufacturing and procurement process. Bigger challenges are to help make CAD/CAM a universal way of doing business within the company, effective use of group classification technology, and teamwork with Manufacturing and suppliers to develop and implement truly innovative design concepts that lead to quality products at lower cost. The EQWG is expected to develop 'end state' concepts and implementation plans for changes of organization, operation, procedures, standards and design concepts to guide our future growth. The target of the EQWG is breakthrough in performance, not just 'work harder'. The group will phase their conceptualizing and recommendations to be effective in the long term and to influence the large number of drawings now being produced by Group 1 and Group 2 design teams.

Example: Real example from a 5,000-engineer corporation (1989). Source: CE, page 71, Case 2.8 where a detailed analysis of this text is given. In this case the Director for Productivity and Quality for Engineering was denied about $60 million from the Board, to fund this project (which was to buy more automation of engineering work processes). He was quite surprised, because in the past, this level of proposal had worked! Can you work out the proposed value of the investment from this?

The quoted examples are real (1989-1998-2006-2007 vintage), and reflect real projects where the $50 million in one case, and $100 million (in another case) *actually spent* was *totally* wasted, no value delivered at all. In the last example, the Board was smart enough to NOT waste the money!

The major initial culprit, in my opinion, was *lack of quantification* of these management-

acknowledged, top-level, large project, objectives. At least one top manager in each case totally agreed with my conclusion. The root cause of this bad practice, in my opinion, was lack of corporate policy, regarding quantification of top-level objectives for big projects. There was no common-sense culture (to make up for the lack of formal culture), amongst the managers approving the 'investment', to acknowledge that the objectives were on very shaky ground.

2. Estimating Long Term Value – strategy

We are all familiar with the 'business case'. A typical business case will probably insist that we feed it with some monetary figure regarding long-term savings, or additional earnings as a result of the investment in the project (monetary value) – the 'benefits'.

The problem with this, is it is not ever based on a *detailed* analysis of the *many* stakeholders, and their value set. It might even typically ignore all stakeholders except 'us' ourselves. It will probably focus entirely on monetary advantages, and seriously ignore all other advantages, even though the other advantages may well be *listed* as 'Critical Business Objectives' (see above examples, strategy 1).

In addition, there may be no obligation, culture, will-power, or ability to actually follow-up and derive the projected benefits in practice. Last month I was told frankly at one place I visited, that although projects said in project justifications, for example, they would "save 20 employees", they were routinely *never* actually saved, and everyone knew there were *no penalties for failing* to make the saving real, when new systems were delivered.

A respectable strategy would be to make estimations of long-term *benefits expected* for all aspects of value, for all stakeholders of significance. We should of course include information on the conditions and assumptions necessary for these benefits to be realized in practice.

3. Focus on Delivery of Value to Stakeholders – strategy

We have a tendency to focus on value to *our* corporation; the one investing in the project. Or we focus on value to our *main customer*, paying for the project.

We have to shift culture, to a time-honored systems engineering notion, that of the *many project stakeholders* [SEH, references in 80 sections to stakeholder]. Each one of say 40 stakeholders will have one, or probably more, value delivery potentials from the project. We need to map all significant stakeholder values, even though they are not 'ours'.

These values are *not the same* at requirements! Stakeholder values represent *potential* requirements ***if*** they are *technically* possible, *economically* possible and *prioritized!* They are, for the moment, just stakeholder *needs* and *values*, **not** *committed* system requirements.

The engineers doing this will increase their real 'productivity' by helping to plan the actual delivery of those values. And *perhaps* even contribute to planning the total systems problem of delivering real benefits on the back of the values deployed technically.

We need to plan to help stakeholders and inform stakeholders, and get co-operation of many of those stakeholders, so that they understand and commit to *their* role in deriving those final benefits for themselves, and for *other* stakeholders.

Example of a Design Specification

Tag: OPP Integration.
Type: Design Idea [Architectural].
========== Basic Information ==========
Version:
Status:
Quality Level:
Owner:
Expert:
Authority:
Source: System Specification Volume 1 Version 1.1, SIG, February 4 – Precise reference <to be supplied by Andy>.

Gist: The X-999 would integrate both 'Push Server' and 'Push Client' roles of the Object Push Profile (OPP).
Description: Defined X-999 software acts in accordance with the <specification> defined for both the Push Server and Push Client roles of the Object Push Profile (OPP).
Only when official certification is actually and correctly granted; has the {developer or supplier or any real integrator, whoever it really is doing the integration} completed their task correctly.
This includes correct proven interface to any other related modules specified in the specification.

Stakeholders: Phonebook, Scheduler, Testers, <Product Architect>, Product Planner, Software Engineers, User Interface Designer, Project Team Leader, Company engineers, Developers from other Company product departments which we interface with, the supplier of the TTT, CC. "Other than Owner and Expert. The people we are writing this particular requirement for."

========== Design Relationships ==========
Reuse of Other Design:
Reuse of This Design:
Design Constraints:
Sub-Designs:

========== Impacts Relationships ==========
Impacts [Functions]:
Impacts [Intended]: Interoperability.
Impacts [Side Effects]:
Impacts [Costs]:
Impacts [Other Designs]:
Interoperability: Defined As: Certified that this device can exchange information with any other device produced by this project.

========== Impact Estimation/Feedback ==========
Tag: Interoperability.
Scale:
Percentage Impact [Interoperability, Estimate]: <100% of Interoperability objective with other devices that support OPP on time is estimated to be the result>.
========== Priority and Risk Management ==========
Rationale:
Value:
Assumptions: There are some performance requirements within our certification process regarding probability of connection and transmission etc. that we do not remember <-TG.
Dependencies:
Risks:
We do not 'understand' fully (because we don't have information to hand here) our certification requirements, so we risk that our design will fail certification <-TG.
Priority:
Issues:
========== Implementation Control ==========
Not yet filled in.
========== Location of Specification ==========
Location of Master Specification: <Give the intranet web location of this master specification>.

Example: a design template, partly filled out in Planguage (Real, telecoms, about 2000). It has collected information on defined stakeholders that are impacted by this design. It has identified a critical technical requirement (Interoperability) impacted by this design. It has yet-unfilled parameters about impact relationships, that challenge us to enrich our understanding of this engineering artifact. The engineer can increase their productivity by analyzing deeper, and acting on the analytical insights. It is not about producing more, but about producing more potentially-fruitful insights for engineering and managing value to stakeholders. Source [CE], page 199.

Secondary Strategies: that will improve our ability to deliver value.

Quantifying Performance, particularly qualities.

Technical system qualities, are not the same is the stakeholder value we discussed above. The technical qualities are the pre-requisites, or 'drivers', of value. But qualities are not the value derived finally by stakeholders.

For example if a system is designed to have a security quality of identifying 99% of attempted system intrusions within 1.o seconds, a 'quality level [Security Quantification]. There is no value if the system is not yet deployed, and if it has no effect on the hacker activity

(because no hackers are aware of the capability, and choose to avoid the system), or if no hackers are caught in the act.

For another example, if a system is designed for high usability, in order to make it unnecessary to train people for a week on the use of the system, but an organization persists in delivering the useless training in spite of this, then no value is actually delivered to the stakeholder. The potential is *there*, but not exploited.

Now, just as the above (1. Measuring Value as a strategy), argues that we cannot expect to engineer the value achievement, **if** the value aspects are not defined *quantitatively*, the same argument applies, for the same reasons, at the level *below* stakeholder values, the *system quality levels.*

System quality levels must be quantified by engineers, and must be engineered into existence. That is a minimum prerequisite for enabling the system to deliver value to stakeholders. [QQ].

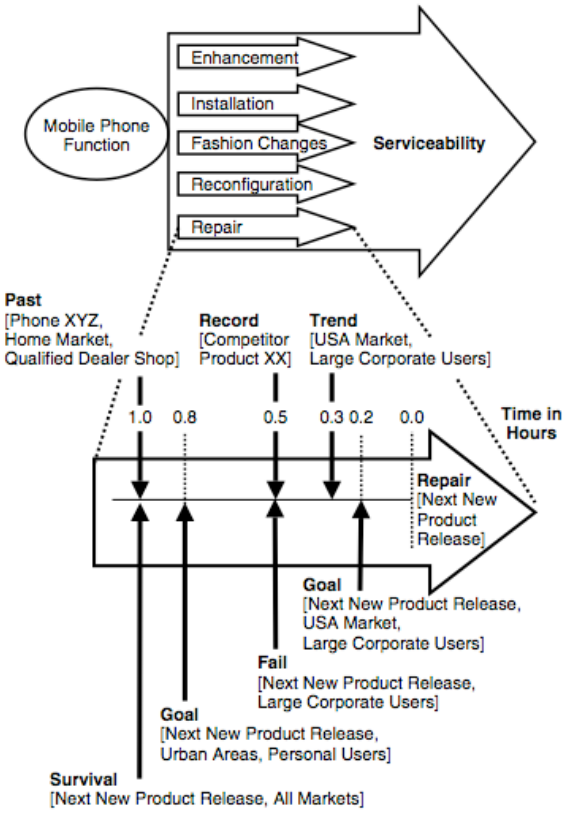

Illustration: the engineering-specification structure of a single quality-aspect (Repair) of a system. This quality aspect would have no value to any stakeholder if the system was never deployed or released, or never had a fault needing repair, or if repair activity were never attempted, or if it were not attempted using the technology designed in the system to give this repair speed. Technical qualities are the basis for deriving value, but they are not to be confused with the ***value ('perceived potential benefit')*** *itself, or even with the long-term* ***benefits ('value delivered to stakeholders')*** *derived from the quality of the system. Source: [CE, SoM] Figure 4.3, page 115.*

Evolutionary Project Management, feedback and correction.

In complex, state of the art, multi-stakeholder, large-scale systems it is acknowledged [US DoD Mil Std 498, for example] that it is impossible to know all the right requirements at the beginning. We have to learn more about, and adjust, initial assumptions, as realities emerge.

From my perspective a major tool to help the systems engineer dialogue with the reality of both the technical, political, economic and other stakeholder environments, is that we create an engineering process that *learns*. The engineering process learns about stakeholder *values*, about necessary and possible *requirements*, about emerging *technology*, about the real ability to make *benefits* happen, and many other uncertain variables. The engineering process learns *early*, *frequently*, and is narrowly focused – not distracted by overwhelming size and complexity.

The class of project management methods that do this are broadly known as 'evolutionary' methods. These are *iterative*, they are *incremental*; but they have one more attribute that makes them '*evolutionary'*: feedback on each cycle, *learning*, and *corrective action* to benefit from the feedback and analysis. In short they are also 'learning' processes.

Although it is not difficult to see this kind of gradual learning process, in many forms, in engineering (multiple prototypes, multiple product versions, the long term evolution of most technologies), current *systems* engineering culture does *not* take such processes for granted at all. If anything, we have got a systems engineering culture that largely assumes something closer to a 'waterfall' model of development [SEH]. It hardly mentions evolutionary processes at all.

I would argue that a systems engineer must normally use, and master an evolutionary feedback project mechanism [Evo]. The fact that corporations and institutions routinely impose a heavy bureaucratic 'big bang' model, with attendant project failures, is a sorry comment on our present culture.

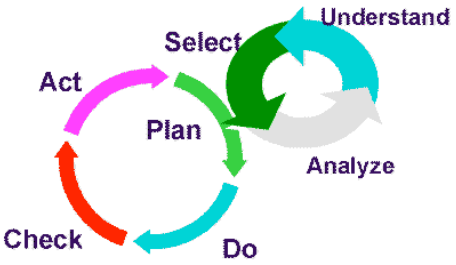

Illustration: A process-improvement cycle: "Understand-Select-Analyze-Plan-Do-Check-Act" which emphasizes that the plan must be based on the understanding of the system and the evaluation of the data on the system. ***We need to apply these cycles better to project management.*** *Source: http://www.triz-journal.com/archives/1998/12/g/Image99.gif*

One of the main conclusions Peter Morris made, in his great book on project management [Morris] was that there was "no good project management method". He was talking about projects like the Concorde, The Channel Tunnel, and the Atomic Bomb (Manhattan). He was talking about systems engineering. His main conclusion was that if we are to improve the project management model, it must include much more feedback – an *evolutionary* model. Systems engineering has not yet taken his advice to heart. Our SE culture is too slow to react to necessities.

One of my favorite tools

Impact Estimation Tables

I believe that the *productive engineer* needs another tool, which I have called the Impact Estimation table [CE], or a similar tool such as Quality Function Deployment (if it is carried out with the same quantified rigor in specification – rare to see in [QFD] practice – but I am told it exists). We need to be able to reason about complex systems, and about the value we are planning to deliver as a result of our technical engineering

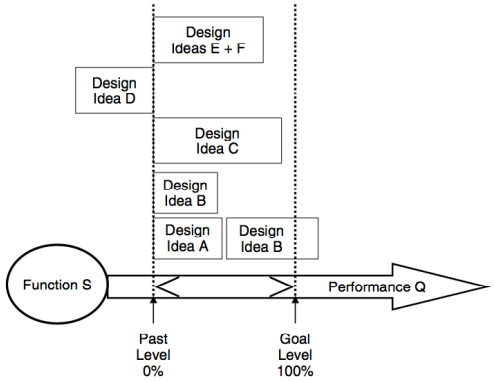

Illustration: the connection between design (for example, required technical system qualities) and Performance Goals (for example derived stakeholder value levels) can be both estimated, and later measured. The estimated or achieved value can be represented graphically, as above (in 'Planguage, [CE]) or on spreadsheet tables.

We need to avoid the common one-to-one reasoning ('we are going to use technology X to achieve Quality Y') and to understand more clearly that our *means* are likely to have multiple *effects* on many of our critical values. This is, of course, good conventional engineering (to worry about side effects) but I see too many real projects where this is not done systematically.

My opinion is that the use of a tool like the Impact Estimation table, would force the systems engineering team to consider their systems, as broadly as we must do in a real systems engineering environment.

Design Ideas ->	*Technology Investment*	*Business Practices*	*People*	*Empowerment*	*Principles of IMA Management*	*Business Process Re-engineering*	*Sum Requirements*
Customer Service ? <->0 Violation of agreement	50%	10%	5%	5%	5%	60%	185%
Availability 90% <-> 99.5% Up time	50%	5%	5–10%	0%	0%	200%	265%
Usability 200 <-> 60 Requests by Users	50%	5–10%	5–10%	50%	0%	10%	130%
Responsiveness 70% <-> ECP's on time	50%	10%	90%	25%	5%	50%	180%
Productivity 3:1 Return on Investment Morale 72 <-> 60 per month on Sick Leave	45% 50%	60% 5%	10% 75%	35% 45%	100% 15%	53% 61%	303% 251%
Data Integrity 88% <-> 97% Data Error %	42%	10%	25%	5%	70%	25%	177%
Technology Adaptability 75% Adapt Technology	5%	30%	5%	60%	0%	60%	160%
Requirement Adaptability ? <-> 2.6% Adapt to Change	80%	20%	60%	75%	20%	5%	260%
Resource Adaptability 2.1M <-> ? Resource Change	10%	80%	5%	50%	50%	75%	270%
Cost Reduction FADS <-> 30% Total Funding	50%	40%	10%	40%	50%	50%	240%
Sum of Performance	*482%*	*280%*	*305%*	*390%*	*315%*	*649%*	
Money % of total budget	15%	4%	3%	4%	6%	4%	36%
Time % total work months/year	15%	15%	20%	10%	20%	18%	98%
Sum of Costs	*30*	*19*	*23*	*14*	*26*	*22*	
Performance to Cost Ratio	*16:1*	*14:7*	*13:3*	*27:9*	*12:1*	*29:5*	

Illustration: a real US DoD Impact Estimation table, from the author's client, the Persinscom (US Army, Personnel System). Behind all tags (Customer Service, Technology Investment) are properly-defined requirements (quantified) and designs. This tool, enables us to get a better overview picture of how mutiple technological ideas, Source CE, page 284.

Some Management Policies for Engineering Productivity

1. **Productivity is Value Delivered:** SE Productivity is ultimately measured in terms of real *benefits* delivered to real stakeholders, as enabled by stakeholder *value* delivered, which is the short term measure of engineering productivity.

2. **Total Systems Engineering:** The engineering organization is responsible for all aspects of value delivery; if necessary including the design of the *organization* needed to continue to deliver the real benefits in the long term.

3. **Value Responsibility**: specified engineering organizational units will be held accountable for initial and long term planned value delivery.

4. **CVO**: A Chief Value Officer will oversee all technical and management efforts on value delivery; and report to the CEO on the situation, using Value Accounting.

Summary

We need to develop a culture in systems engineering, where the *delivered value* and *consequent benefits* are considered the primary purposes of systems engineering. *Value to stakeholders* can be a *primary* measure, short term, of the productivity of systems engineering. '*Delivered benefits'* is a better measure of the *real productivity* of the systems engineering function.

References

[CE] **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. as noted below:

[SoM] Chapter 5: Scales of Measure:

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

[Evo] Chapter 10: Evolutionary Project Management:

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

Gilb.com [www]: 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.

[QQ] Quantifying Quality theme:

[QSV] Quantifying Stakeholder Values (INCOSE 2006 paper)
http://www.gilb.com/community/tiki-download_file.php?fileId=36

[H2QQ] Main QQ paper 2007 Version "How to Quantify Quality: Finding Scales of Measure".

http://www.gilb.com/community/tiki-download_file.php?fileId=124
This was originally published as a paper at INCOSE.org conference Washington DC 2003.

[QQ Slides] http://www.gilb.com/community/tiki-download_file.php?fileId=131
This is a 2007 version of the slides used to lecture on quantifying quality, at universities and conferences, and for clients.

[QFD] What's Wrong with Quality Function Deployement?
http://www.gilb.com/community/tiki-download_file.php?fileId=119

[Security Quantification] Quantifying Security: How to specify security requirements in a quantified way. Unpublished paper. Tom Gilb. See also [SoM] above.

http://www.gilb.com/community/tiki-download_file.php?fileId=40

[Morris] The Management of Projects

Peter Morris, UMIST

ISBN: 9780727716934, 1994, Thomas Telford Ltd., 358 pp

http://www.thomastelford.com/books/bookshop_main.asp?ISBN=072771693X

[SEH] INCOSE Systems Engineering Handbook v. 3.

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

Biography

Tom Gilb is an international consultant, teacher and author.

His 9th book is '**Competitive Engineering**: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (August 2005 Publication, Elsevier) which is, among other things, a definition of the planning language 'Planguage'.

He works with major multinationals such as Credit Suisse, Schlumberger, Bosch, Qualcomm, HP, IBM, Nokia, Ericsson, Motorola, US DOD, UK MOD, Symbian, Philips, Intel, Citigroup, United Health, Boeing, Microsoft, and many smaller and lesser known others. See www.Gilb.com .

Version started Friday 2nov07 20:37-24, nov 3 from 1337- 01:45 Nov 4th