SYSTEMS ENGINEERING NEWSLETTER

brought to you by

Project Performance International (PPI)

SyEN 62 – February 23, 2018

SyEN is an independent free newsletter containing informative reading for the technical project professional, with scores of news and other items summarizing developments in the field, including related industry, month by month. This newsletter and a newsletter archive are also available at www.ppi-int.com.

Systems engineering can be thought of as the problem-independent, solution technology-independent, life-cycle-oriented principles and methods, based on systems thinking, for defining, performing, and controlling the engineering effort within a technical project. The approach aims to maximize the benefit delivered to the enterprise, as influenced by the needs and values of the applicable stakeholders.

If you are presently receiving this newsletter from an associate, you may wish to receive the newsletter directly in future by signing up for this free service of PPI, using the form at www.ppi-int.com. If you do not wish to receive future Systems Engineering Newsletters, please unsubscribe by clicking on the link at the bottom of this email.

We hope that you find this newsletter to be informative and useful. Please tell us what you think. Email us at syen@ppi-int.info.

The views expressed in externally authored articles are those of the author(s), and not necessarily those of PPI or its professional staff.

IN THIS EDITION

Quotations to Open On

Read More…

Feature Article

- How Well Does the Agile Manifesto Align with Principles that Lead to Success in Product Development? *by Tom Gilb*

Read More…

Articles

- Why Agile Product Development Systematically Fails, and What to Do About It! *by Kai Gilb*
- Integrating Program Management and Systems Engineering, *by Dr. Ralph Young*

Read More…

Systems Engineering News

- Project Success Rates are Rising!
- Newly Released Science & Engineering Indicators Document Describes the State of American Technology
- Suggestions for the Upcoming INCOSE International Symposium
- Capers Jones Publishes “The Mess of Software Metrics”

Read More…

Featured Organizations

- Association of Business Process Management Professionals (ABPMP)
- The System Dynamics Society

Read More…

Conferences and Meetings

Read More…

Some Systems Engineering-Relevant Websites

Read More…

Systems Engineering Publications

- Value Planning: Practical Tools for Clearer Management Communication
- Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage
- Success Rates Rise: Transforming the High Cost of Low Performance
- Agile Project Management for Dummies
- Writings of Interest *by Tom Gilb*
- Writing of Interest *by Tom and Kai Gilb*
- Systems Science, Systems Thinking, Systems and Software Engineering: Addressing Transdisciplinary Frontiers, Practice and Education

Read More…

Systems Engineering Tools News

- Major Release for Capella Studio
- The Arcadia Method
- Automated Scrum Process with Integrated Agile Toolset

Read More…

Education and Academia

- Systems Engineering at Drexel University, Philadelphia, Pennsylvania, USA

Read More…

Standards and Guides

- Software Engineering Standards. A User's Road Map
- Communicating Project Management: The Integrated Vocabulary of Project Management and Systems Engineering
- Practical Merger and Acquisition Execution and Integration: A Step by Step Guide to Successful Strategy, Risk and Integration Management
- A Beginner's Guide to Understanding the Agile Method
- Agile for the Enterprise: Five Steps to Successful Adoption
- Agile Project Management for Dummies Cheat Sheet

Read More…

Definitions to Close On

Read More…

PPI and CTI News

Read More…

Upcoming PPI and CTI Participation in Professional Conferences

Read More…

PPI and CTI Events

Read More…

QUOTATIONS TO OPEN ON

“The principles of risk management are straight-forward. Implementation is a little tricky.”

Robert John Halligan

“Nature and books belong to the eyes that see them.”

Ralph Waldo Emerson

“Between stimulus and response there is a space. In that space is our power to choose our response. In our response lies our growth and our freedom.”

Viktor Frankl

“To the optimist, the glass is half full. To the pessimist, the glass is half empty. To the engineer, the glass is twice as big as it needs to be.”

Unknown

FEATURE ARTICLE

How Well Does the Agile Manifesto Align with Principles that Lead to Success in Product Development?

by

Tom Gilb
Value Delivery Expert
Independent Teacher, Consultant, and Author

Email: Tom@Gilb.com

Background

I carried out my first 20-value-delivery-step agile IT project in 1960 on an invoicing system in Oslo when I was 20 and I just used my common sense. It was a radical re-architecting of what IBM initially sold to my client. So I realized that 'smarter architecture' might be needed to deliver results stepwise, with learning at each step.

Then I began to realize not everyone in this business has common sense. But many smarter people shared the agile ideas, which we called "Evolutionary" at the time [see 28B].

With few exceptions [18, 19, 28B, 30, 31] I was for over 35 years a lone voice in the wilderness: the masses, including the U.S. Department of Defense (DoD), believed in Waterfall, and I was obviously a bit unconventional and ignored, as I often am today concerning the need for engineering methods in software and management [1, 2].

Fortunately for me, there were several exceptional organizations that requested me to help them with these 'evolutionary' ideas, for example, HP [29], Intel [15], Boeing, Ericsson, and Confirmit in Norway [20] - and others, all of whom had more quantified documented success than any of the Manifesto Agile offspring. I was not alone, rather, a quiet minority.

Unfortunately, the Agile Manifesto states embarrassing platitudes, with no visible foundation or purpose. In this article, I will discuss the Agile Manifesto point by point: its four values and ten principles. I will first attempt to answer the question of how I aligned with the value or principle. Then I will add my own ideas, and a reformulation of the principles.

In general, I was never impressed [5, 27] with the expositions concerning Agile because of what I considered their “fuzziness”. But the thirsty world out there did get seduced by that fuzziness. 'Survival is not mandatory' as W. Edwards Deming said[1].

If I were to put blame on a single factor, I would blame the management MBA culture. Too much 'bean counting' and too little about 'managing values' and 'delivering qualities' that actually provide financial value. [34, 22].

The Four Values of the Agile Manifesto

Reference: http://agilemanifesto.org

I have long since written my counter-proposal for Agile Values [36B]. I believe that the value statements provided in “Values for Value” are much better and clearer than the fuzzy stuff in the Manifesto.

1. Individuals and interactions over processes and tools

Well, of course. Live human reality beats theory and planning.

Planguage (I use this term for specification language for requirements, design, stakeholders, and results) and Evo (the term I use for iterative, incremental, learning project management process [1, 2], are 'tools and interactions' which deeply support stakeholders, learning, feedback, and change; in multiple dimensions (of values and costs) simultaneously.

Of course, 'stakeholders first' and their ‘interactions with requirements and systems' before bureaucracy. However, people obviously have to be taught suitable processes to support stakeholders, and the Manifesto hardly mentions 'stakeholders': only the narrow category 'users and customers' dominates (for example, in the practices, user stories, and use cases that might better be called 'stakeholder stories' and 'stakeholder cases').

My conclusion is that the Manifesto is dangerously ‘narrow-minded' concerning people and interactions. Figure 1 below, from my slides Advanced Agile Software Engineering (2018) [37] (http://concepts.gilb.com/dl915) provides many examples of stakeholder categories, and expresses the idea that stakeholder analysis interacts with values (requirements) in a continuous, iterative, learning way [51, 52].

[1] *Out of the Crisis*. Cambridge, Massachusetts, USA: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986. Dr. Deming is the father of quality in Japan and did much for the United States as he emphasized giving more attention to it. See also Mary Walton, *The Deming Management Method* (New York, N.Y. USA, The Berkley Publishing Group, 1986).

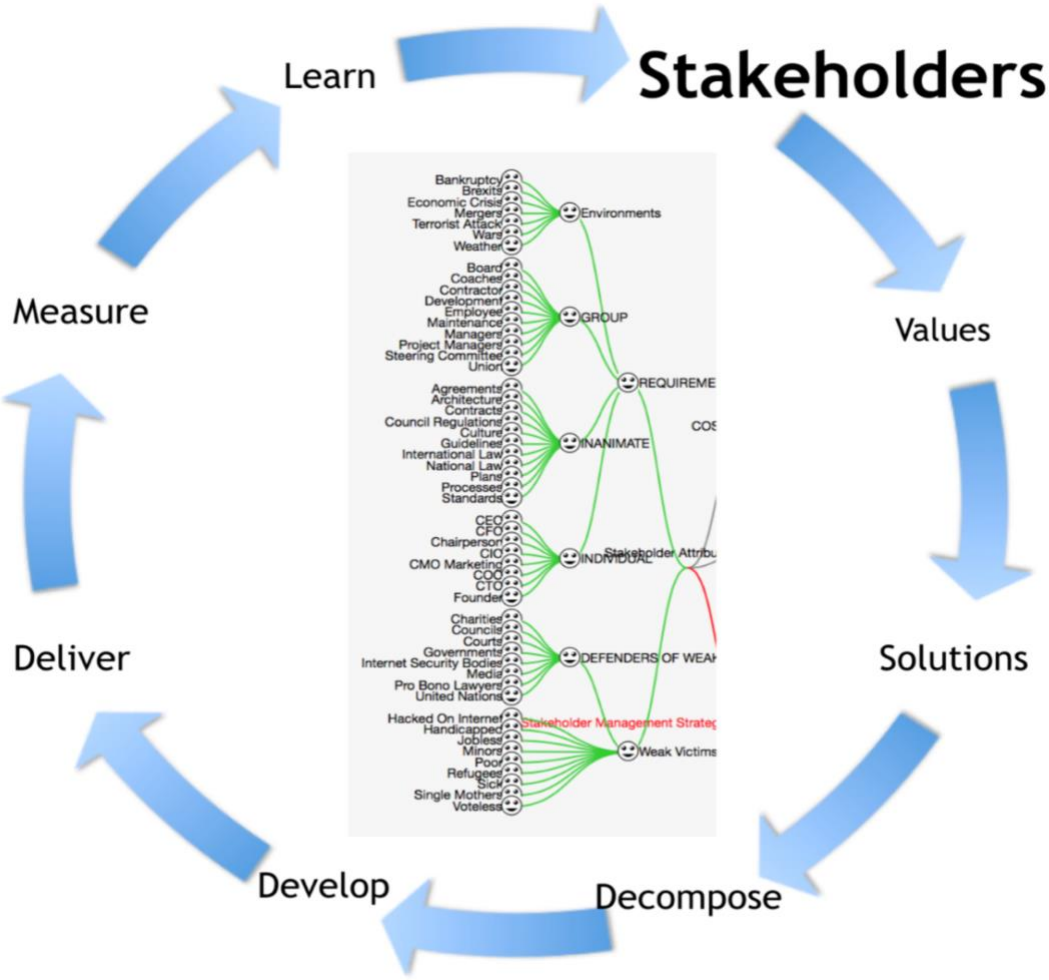

Figure 1: Continuous Iterative Interaction of Stakeholders with Requirements

2. Working software over comprehensive documentation

Absolutely agree.

That is why Evo suggests a maximum of 1-week front-end planning, before diving in and attempting to deliver real measurable stakeholder-value increments, on the 2nd and all following weeks [1, 5. 6, 8], until no stakeholder value deliveries *can* be prioritized [10].

Notice I said 'deliver real measurable stakeholder value' rather than "Working software is the primary measure of progress"; or even worse, "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software".

It is why Evo has a 'startup process' that is 'time-boxed' to a maximum of one week [6], and why we do the 'top-ten critical stakeholder values quantified', on a *single* page, in a *single* day [5A].

We then specify the 'top-ten critical architecture ideas' on the second day, on a *single* page [6] and continue on in the next 2 days [6] with estimation of 'architecture value impacts' and 'architecture costs', and then selection of 'next week's agile value delivery sprint' on day 4. Unfortunately, Manifesto Agile suggests none of this [32, 33].

Manifesto Agile practices (not in the Manifesto *itself*, but rather in XP) does user stories and epics. That is, it provides language and documentation. But this is less valuable documentation, premature overhead, and are 'amateur designs' pretending to be 'requirements', and they are overrated detail overhead, suitable for coders, but not project managers, and not result designers. [5B].

I understand this value as a reaction to 'excess quantities of documentation' in some earlier waterfall methods [30, 31]. But the reaction is a 'programmer's eye view of the world', and does not really consider the primary and critical purposes of all projects: to deliver value to stakeholders, NOT 'code to computers'.

There were far too many 'coders at heart' who negotiated the Manifesto. Apparently, they had no understanding of the notion of delivering measurable and useful stakeholder value. This can be done without coding at all! Some of them (Sutherland and Cohn, for example) do appreciate the 'value and quality' notion better today, but their methods do not instantiate the consequences of that.

Good managers could have prevented narrow-minded excess.

3. Customer collaboration over contract negotiation

I believe this Manifesto value notion has its roots in inadequate contracting practices, compounded by even worse development processes: waterfall, fixed price, and fixed dates, with contract technical design specifications instead of contract results, and specifications.

Some professional friends of mine have built a simple legal framework for doing agile. There is no fixed long term cost, or specs, or deadline.

It is all worked out in 'collaboration with the customer' step by step. If step results are measurably delivered, payment is due. [39]. 'Negotiation' is done step by step, as we learn, get results, and build confidence.

In earlier times, in a situation where fixed price, fixed date, and fixed high quality levels were simply handed to the developers, a smart team at IBM Federal Systems Division, led by Harlan Mills [18, 19], developed a process called 'Cleanroom' which was completely agile, but more like Evo since it got control over qualities, costs, and time by quantification, measurement and learning, coupled with in step re-architecture [Quinnan, 18].

Since Manifesto Agile has no architecture concept, it is incapable of doing agile architecture the way Quinnan did it at IBM (30 years earlier in deliveries of 43 increments).

Their published results [19] were not like Manifesto Agile (20-60% failure) [33]. Their result was what we experience and expect with the cousin process 'Evo': 'all projects on time, under budget' year after year, without exception.

The success reason is simple, 'lean': early continuous feedback and learning, based on quantification and measurement of critical values and qualities (software 'engineering') [2, 19, 25, and 28]. The 'systems engineering' and 'software engineering' which is totally absent from Manifesto Agile.

If one does not state value improvements and costs quantitatively up front, and then iterate towards meeting targets, within resource constraints (engineering, Evo, Cleanroom), one cannot see deviation from plans early enough. One will not be successful [33].

4. Responding to change over following a plan

Of course, I agree, as noted previously.

But, there are several kinds of 'plans', for example: immature fixed ones that are based on lack of deep understanding of complex stakeholder values; 'plans which specify badly designed architecture', rather than end results for stakeholders.

Our preference is 'plans that focus on a few critical, quantified, top level, long-term value improvements'. Of course, these quantified plans are subject to incremental change, for example, change directed by high level guidance, from top management, on behalf of their stakeholders, providing good directions of change and improvement.

I believe [1] that we need much better, and much higher level 'plans' [1, 5A], and that our responses need to be caused by 'numeric deviation from plans', or numeric need to change these numeric plans *to reflect the real world*.

This is both because we get to understand that real world, by trying to deliver change, and because the real world itself needs to change top-level requirements (business, market, and society changes, for example). And thirdly because of the necessity of change to improved top-level architectures (technology change).

Manifesto Agile is light-years distant from (really and practically) dealing with these realities. It is likely to fail, except in the simplest of small programming projects.

In summary, the 'four values' are poorly stated by the Manifesto committee. Planguage and Evo methods are far better suited to the mature intent of the values.[2]

The Twelve Agile Manifesto Principles

Reference: http://agilemanifesto.org/principles.html.

I provided my personal counter-proposal for Agile Principles in 2010 [see 36A].

I believe that the 'principles' statements provided there are much better and clearer than those in the Manifesto.

[2] *See Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, available at https://www.gilb.com/p/competitive-engineering. See also Planguage: A Software and Systems Engineering Language, for Evaluating Methods, Managing Projects for Zero Failure, and Maximum 'Value Efficiency'. Keynote: International Conference on Software Process and Product Measurement (Mensura). Available at http://concepts.gilb.com/dl918.

I provide here my direct comments on the principles as published.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Gilb methods (Planguage, Evo, [1, 2]) are devoted to 'stakeholder satisfaction' but in consideration of constraints such as legality, money, time, and 'balance with all other multiple values of a set of stakeholders'. I like the sentiment of Principle 1, but dislike the formulation.[3]

I believe that true customer satisfaction needs to be defined unambiguously and quantitatively in terms of stakeholder values.

The Manifesto has no such serious ‘stakeholder value' understanding, and seems to suggest that 'code delivery' is the same as 'customer satisfaction'. Or, at the common-agile-practices level, that 'user stories delivery' is 'satisfaction'. I disagree.

Here is my constructive reformulation:

1. Development efforts should attempt to deliver, measurably and cost-effectively, a well-defined set of prioritized stakeholder value-levels, as early as possible.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Gilb methods are completely tuned to 'rapidly', and to some degree 'automatically', accommodate changing requirements, of all kinds, including all critical stakeholder values; not just functional requirements and designs (i.e. not just user stories, functional requirements, or designs).

We not only can easily adjust any requirements, but we can compute the changed priority [see 10] for implementation sequence, using such tools as Value Decision Tables and the Needs and Means Planning Tool (see www.needsandmeans.com). This is far superior to common agile practices such as using a product owner.

Here is my constructive reformulation:

2. Development processes must be able to discover and incorporate changes in stakeholder requirements, as soon as possible, and to understand their priority, their consequences to other stakeholders, to system architecture plans, to project plans, and contracts.

[3] See Tom Gilb’s article, “The 10 Most Powerful Principles for Quality in Software and Software Organizations” [56] for an excellent tutorial concerning how to provide quality software.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

I do not believe that this is a useful principle. I believe that it is 'delivery of defined and approved stakeholder values' which is useful.

Including the idea of delivering 'values for resources consumed'. Meaning 'profitability' and 'efficiency'.

Evo and Planguage [1, 2] would be quite happy, even in the realm of IT systems, if we never wrote code, and never delivered it. Code is not the point, except for coders.

The objective is to achieve ‘business and organizational improvements', and if we can find better, more cost-effective ways, to deliver those values, we should use those methods.

We need, I believe, to approach most of our projects from a 'systems' point of view, that is, a view that considers the interactive nature and interdependence of external and internal factors. Not a dangerously narrow 'program code' point of view. The Manifesto has failed us here.

Here is my constructive reformulation:

3. Plan to deliver some measurable degree of improvement, to planned and prioritized stakeholder value requirements, as soon, and as frequently, as resources permit.

Not, 'working software', just real stakeholder results. Personally, I prefer weekly or 2% of budget steps. Keep the measurable improvements 'continuously' flowing, however you choose to do it. I recommend not waiting a couple of months, if you can do better than that.

4. Business people and developers must work together daily throughout the project.

We support the *spirit* of this principle (except the unnecessary limitation of the adjective 'business'). But it is clumsily formulated, and unnecessarily proscriptive.

There are available a large number of practical tools to assist collaboration: not least the basic idea that all required value improvements can and will be expressed quantitatively. All parties can work together towards that common set of objectives.

The Planguage 'stakeholder value quantification' [1] is a great tool for improving collaboration. This is because all stakeholders and all developers will be able to understand the same thing, and track progress in actual value delivery.

'Stakeholders', including *critical* stakeholders, is a much broader category of critical requirements sources than 'business stakeholders'. See the example stakeholder map above (Figure 1).

The terms 'together', 'daily', and 'work' are ambiguous. When does the project begin and end? Who are the business people?

Here is my constructive reformulation:

4. All parties to a development effort (stakeholders), need to have a relevant voice for their interests (requirements), and an insight into the parts of the effort that they will potentially impact, or which can impact them, on a continuous basis, including into operations and decommissioning of a system.

Note: this does not happen by 'working together daily'. That becomes impractical and unworkable in large scale distributed systems. I believe that by having controlled access to a common project database in Planguage and using a tool such as needsandmeans.com, we can provide a 'relevant voice' to all stakeholders, and we can provide insight into consequences of plans and decisions for all.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Well of course.

Projects need to be built around a balanced, logically-prioritized set of stakeholder needs [10], and with consideration of available resources (people, time, and money).

Projects and project methods can be designed to motivate various types of individuals and stakeholder types. But this concept of hiring or employing individuals who are motivated sounds optimistic to me. Motivated people can get 'turned off' for such a large number of reasons.

And, of course, we all prefer competent experts over motivated untrained novices.

Trust, but verify. [1, see Quality Control, especially part 2 and Part 4, and Chapter 10, Quality Management].

Here is my constructive reformulation:

5. Motivate stakeholders and developers, by agreeing on their high-level priority objectives, and give them freedom to find the most cost-effective solutions. [42]

6. Enable face-to-face interactions

Sometimes, yes; but not always. Face to face interactions are not always possible, not always cost-effective, and not always desirable.

I understand the frustration of not being able to discuss things with stakeholders, and the dangers of not being able to motivate, get motivated, clarify, and get clarity by interactions.

So it sounds great: face-to-face, electronically or in the flesh, as long as it the most cost-effective way to deliver real measurable value.

But the really important idea is not face-to-face, itself.

The important idea, independent of the 'how' (the chosen means to communicate), is *communication of ideas*, like requirements, designs, risks, progress, problems. Often for a very large number of stakeholders and individuals, over great time and distances, in a fast-changing world. We need to be able to tackle very large and complex systems, and there becomes a point where face-to-face is not the most important communication technique.

I believe that Planguage and Evo methods, including the app needsandmeans.com which works well in 'distance online meetings and discussions', offer a more effective solution to this communication problem. And a Planguage database is one in which we can feed in or out from; and use for better face to face interactions, than if we do not use such tools.

Here is my constructive reformulation:

6. Enable clear communication, in writing, in a common project database. Enable collection and prioritization, and continuous updates, of all considerations about requirements, designs, economics, constraints, risks, issues, dependencies, and prioritization.

Note that I did *not* mention face-to-face. I suggest that people should be free and agile enough to figure out their best current available mode of communication.

The critical idea is not face-to- face, rather the quality of relevant information from and for stakeholders, including developers. My experience is that oral communication is not a good way to formulate complex things clearly. But oral and visual communication can assist in improving the clearer formulation, in the project database. This is a general truth in all advanced disciplines. Face-to-face is more appropriate in simpler and more primitive situations, and as a useful *supplement* to communicating with and about the project database.

7. Working software is the primary measure of progress.

This principle is ridiculous, unless one believes that writing code is the primary objective.

The primary measures (plural!) of progress are the measurements of the delivery of an official approved current set of requirements, for a given stakeholder type, or individual. These must necessarily be quantified, and well defined. They must be relevant measures, not just something 'easy to measure'. This is the core discipline, and it is not mentioned in the manifesto or in most agile practices (except Cleanroom and Evo) [1, 2, 7, 18, 19, 20]

Here is my constructive reformulation:

7. The primary measure of development progress is the 'degree of actual stakeholder-delivered planned value levels' with respect to planned resources, such as budgets and deadlines.

Note that this can be expressed as a simplified overall measure, for any set of value requirements as 'average % of goal levels of required values delivered on time'. This applies even when no working

software is delivered at all. So, databases are not software, right? I do like to call them dataware, but they are soft, and they can change value delivery, for example.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

There are a large number of methods, which need to be applied simultaneously to ensure sustainable development. For example, some possible sources of more effective approaches include Capers Jones at www.namcook.com, David Rico at http://www.davidfrico.com/, and Jack Caine's slides at https://www.linkedin.com/in/jack-caine-58690a13/ [32].

I believe that value should be delivered from the project's second week; continuously; and then 'intelligent dynamic prioritization', coupled with quantified values and qualities, are the key ideas that need to be assimilated into projects [1, 2, 6, 10, 15].

Here is my constructive reformulation:

8. We believe that a wide variety of strategies, adapted to current local cultures, can be used to maintain a reasonable workload for developers, and other stakeholders; so that stress and pressures, which result in failed systems, need not occur.

Note: again, we raise our sights to a higher level (avoid pressure leading to problems), and leave the discovery and creativity to the practitioners [42].

In the case of the IBM Defect Prevention Process at the IBM Minnesota Labs [1, 2, 14, 42], there were an estimated 2,167 process changes made to the software working processes, over 18 months, to improve quality and productivity. This was pre-manifesto.

9. Continuous attention to technical excellence and good design enhances agility.

Is *agility* in itself, a good thing, or is it a *possible means* to some *specific ends*?

Planguage and Evo [1, 2] specify very specific methods and tools, to plan and measure 'technical excellence' [7]; and to plan and measure 'good design' (the Value Decision Table [1, 2]).

Here is my constructive reformulation:

9. Technical excellence in products, services, systems and organizations, can and should be quantified, for any serious discussion or application. The suggested strategies or architectures, for reaching these 'quantified excellence requirements', should be estimated, using Value Decision Tables [45, 1, 2], and then measured in early small incremental delivery steps.

To add specificity:

9. Quality must be quantified, and supporting designs for quality must be estimated and measured. [4]

10. Simplicity - the art of maximizing the amount of work not done - is essential.

Following are my detailed and practical complexity-simplification ideas and methods [46, 47, 48, 49]:

Small iterative value-delivery steps (Evo, and some other 'agile' methods like Cleanroom, are major simplification techniques [18]). There are very many more simplification techniques [1, 2, 46, 47, 48, 49].

My favorite single practical method for dealing with complexity is the Value Decision Table [1, 2, 5]. It immediately helps us divide up the complexity.

And there are many other tools for decomposing complex problems into simpler problems [9].

Here is my constructive reformulation:

10. We need to learn and apply methods, of which there are many available, to help us understand complex systems and complex relations. [1, 2, 46, 47, 48, 49] and succeed in meeting our goals in spite of them.

Note: I did not mention the 'work not done' phrase. That is merely one possible outcome of many - avoiding 'muda' or wasted work. Other outcomes of simplification might primarily lead to delivering the right quality and values on time, and might well involve more work than failed projects would use. That seemingly 'extra work' is essential to the purpose of successful value delivery. Things should be as simple as possible.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

Where is the proof or evidence or research, or even a credible case study to back this up?

Scrum and other agile teams are intended to be self-organizing, but the level of requirements, architecture, and design they produce is from 'nothing at all' to 'abysmally and embarrassingly bad', in my opinion [50].

Most so-called enterprise architects (99% according to my informal 300 architect survey [50]) cannot quantify the qualities or costs of their architecture.

I believe in self-organizing teams, as a smart way of designing organizations and products [42], and add Dave Snowdon's Sensemaker to the mix of exciting approaches [http://cognitive-edge.com/sensemaker/], but the above principle is too vague, unsubstantiated, and impractical.

Here is my constructive reformulation:

11. A. The most useful value and quality requirements will be quantified, and will use other mechanisms, including careful corresponding stakeholder analysis [1, 51, and 52], to facilitate understanding.

11 B. The most cost-effective designs/architecture, with respect to our quantified value and resource requirements, will be estimated and progress tracked, utilizing a Value Decision Table with its evidence, sources, and uncertainty. They will be prioritized by values/resources with respect to risks [45].

Simplified:

11. We will use engineering quantification for all variable requirements, and for all architecture.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

This is a nice sentiment for small teams with no intent to help the larger organization learn anything from them, or without allowing that larger organization to teach them anything, or to validate their ideas on a larger scale. It is called silo thinking [53].

For a typical organization, the best proven method I know of, with facts and numbers is the Defect Prevention Process [1, 14, 42] invented by IBM. This allows any number of employees and developers, anywhere, to 'reflect on how to become more effective' in small teams, and then escalate deployment to the larger organization for proven ideas.

Here is my constructive reformulation:

1*2. A process like the Defect Prevention Process (DPP), or another more-suitable for current culture, which delegates power to analyze and cure organizational weaknesses, will be applied: using participation from small self-organized teams to define and prove more cost-effective work environments, tools, methods, and processes.*

DPP is based on grass roots sampling analysis of defects found in inspections and in tests (this is also called Specification Quality Control and static tests). The grass roots estimate defects with common root causes, estimate the root causes, and estimate potential cures. Cures are explored and scaled up if they work.

Acknowledgement

A sincere thank you to my good friend of more than 30 years, Ralph Young, SyEN Editor, for his extensive and very helpful collaboration with me in providing this article.

References

[1] Value Planning (2017) Link to book: https://www.gilb.com/store/2W2zCX6z.

[2] Tom Gilb, *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage* (2005). Obtain a free e-copy of the '*Competitive Engineering'* book. See https://www.gilb.com/p/competitive-engineering. Also available at https://www.amazon.com/Competitive-Engineering-Handbook-Requirements-Planguage/dp/0750665076/ref=sr_1_1?s=books&ie=UTF8&qid=1515499392&sr=1-1&keywords=tom+gilb.

[3] www.gilb.com

[4] TedX Talk: Tom Gilb, 'Quantify the Unquantifiable' https://www.youtube.com/watch?v=kOfK6rSLVTA.

[5] Gilb's Agile Methodology:

[5A] "The Top 10 Critical Requirements are the Most Agile Way to Run Agile Projects". See http://www.gilb.com/dl797.

[5B] "User Stories: A Skeptical View". Available at http://www.gilb.com/DL461.

[6] Gilb: An Agile Project Startup Week. See www.gilb.com/dl568.

[7] Gilb, *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage* (2005), Chapter 5, Scales of Measure.

[8] In CE book [2] and in the Persinscom Case: US DOD Army Personnel System. "111111 Unity Method of Decomposition into weekly increments of value delivery" (10 min. talk slides). See http://www.gilb.com/DL451.

[9] See the Chapter 5 Decomposition chapter in Value Planning [1] or visit https://www.dropbox.com/sh/dc7v636m7w7vvgx/AABfMAW_FnJny23XZKQZQkF4a?dl=0.

[10] Ch. 6 Prioritization, in Tom Gilb, *Value Planning* (2017) ("VP book") Reference [1] above, or visit https://www.dropbox.com/sh/34llx1a7ckyagxl/AAA0pDzSxN5WmoP9lOKR0Mpca?dl=0.

[11] Ch. 7 in Tom Gilb, *Value Planning* (2017) [1], Risk Management, or visit https://www.dropbox.com/sh/fxvtya6gyvgwkfa/AAA5-vrLUt_z0h9EYt1ql3Uma?dl=0.

[12] Ch. 9 in Tom Gilb, *Value Planning* (2017) [1], Communication.

[13] Ch. 10 in Tom Gilb, *Value Planning* (2017) [1], Quality Management, or visit https://www.dropbox.com/sh/vjwybhqfxrvctk7/AAAdabECBSo5x-tSOI85R-1da?dl=0.

[14] Tom Gilb and Dorothy Graham, *Software Inspection*, 1994. Available at https://www.amazon.com/Software-Inspection-Tom-Gilb/dp/0201631814/ref=sr_1_3?s=books&ie=UTF8&qid=1515499275&sr=1-3&keywords=tom+gilb.

[15] J. Terzakis, (Intel) "The impact of requirements on software quality across three product generations," 2013 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, 2013, pp. 284-289. https://www.thinkmind.org/download.php?articleid=iccgi_2013_3_10_10012.

[16] Ch. 2 in Tom Gilb, *Value Planning* (2017) [1], Strategies, or visit https://www.dropbox.com/sh/xab857l9ksfs7w0/AACKonxV1x_LI5TW62FICMMPa?dl=0.

[17] Tom Gilb, "The Logic of Design: Design Process Principles". 2015 paper. Available at http://www.gilb.com/dl857.

[18] Cleanroom, Quinnan, in Tom Gilb, *Value Planning* (2017) [1], Case 2.5 [8], QUINNAN AND MILLS CLEANROOM. Available at http://concepts.gilb.com/dl896.

[19A] Mills, H. 1980. “The Management of Software Engineering, Part 1: Principles of Software Engineering.” IBM Systems Journal 19, issue 4 (Dec.):414-420. Available at

http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan.

[19B] Mills, Harlan D.; Dyer, M.; and Linger, R. C., "Cleanroom Software Engineering" (1987). The Harlan D. Mills Collection. Available at http://trace.tennessee.edu/utk_harlan/18.

[20] Value Driven Project Management 17.5MB slides 2008 '152'. Includes Confirmit Case (slides 70-93). http://www.gilb.com/dl152, http://www.gilb.com/dl50.

[21] T. Gilb and L. Brodie, "How Problems with Quality Function Deployment's (QFD’s) House of Quality (HoQ) can be addressed by applying some concepts of Impact Estimation (IE)". Available at http://www.gilb.com/DL119.

[22] Elon Musk, the bio by Ashlee Vance, 2015. The only fault in my Tesla S from Oct 2016 to December 2017 was that the remote tire pressure sensors did not work properly with non-standard winter tires. This was fixed for free by Tesla. That quality is not an accident, it is a result of iterative intelligent prioritization, weekly, forever. 20 upgrades a week to production cars, and 10 a week to delivered cars (software over air).

[23] Tom Gilb, *Value Planning*, Chapter 3, Levels of Interest and Levels of Control (Concerns stakeholders). https://www.dropbox.com/sh/xbzn5s8imf9vla0/AAB8h-OFvQmJ_w3wNhrDxa9_a?dl=0.

[24] Needs and Means Planning Tool. www.needsandmeans.com – requires signing up for a free planning tool.

[25] Tom Gilb, “Everyday Superpowers”. Paper that provides information concerning the methods taught by Tom Gilb, Version 020118. Available at http://concepts.gilb.com/dl914.

[26] References and reviews for Tom Gilb concerning Agile. Available at https://www.dropbox.com/s/4gd9214tttfiwym/Agile%20References%20Gilb.pdf?dl=0.

[27] Some of Tom Gilb’s Agile-related papers slides and videos. Available at:

https://www.dropbox.com/s/p6iinwnaak0339o/gilb%20agile%20papers%20slides%20per%20231017.pdf?dl=0.

[28] Tom Gilb, *Principles of Software Engineering Management*, 1988. This is the foundational agile book referred to by many Manifesto signers [26].

[28A] See PDF, Chapter 14, The Management of Software Productivity, Available at http://concepts.gilb.com/dl560.

[28B] Chapter 15, some deeper and broader perspectives on evolutionary delivery and related technology. Available at http://concepts.gilb.com/dl561.

[28C] Go to https://www.amazon.com/Principles-Software-Engineering-Management-Gilb/dp/0201192462 to obtain a free digital copy. First sign up at www.gilb.com.

[29] Elaine L. May and Barbara A. Zimmer, “The Evolutionary Development Model for Software”. August 1996 Hewlett-Packard Journal. Available at http://www.gilb.com/DL67.

[29A] Todd Cotton, “Evolutionary Fusion: A Customer-Oriented Incremental Life Cycle for Fusion” August 1996 Hewlett-Packard Journal. Available at http://www.gilb.com/DL35.

[29B] RAPID AND FLEXIBLE PRODUCT DEVELOPMENT: AN ANALYSIS OF SOFTWARE PROJECTS AT HEWLETT PACKARD AND AGILENT (2001), by Sharma Upadhyayula [M.S., Computer Engineering University of South Carolina, 1991, and Massachusetts Institute of Technology, January 2001]. http://www.gilb.com/DL65.

[29C] Darren Bronson, “Best Practices for Evolutionary Software Development”. 1999. http://www.gilb.com/dl825.

[30] Mil Standard 498. Software Development and Documentation. A copy may be requested by filling in a form at https://kkovacs.eu/free-project-management-template-mil-std-498.

This standard supports multiple program strategies. It clearly says that there are three ways of doing projects: "Grand design" (also known as, waterfall), "Incremental" (which most Agile projects do), and "Evolutionary" (which includes exploratory projects like prototypes).

[31] Lewis Gray. Standards Comparison slides 498 and later civil standards. 1999. A Comparison of IEEE/EIA 12207, ISO/IEC 12207, J-STD-016, and MIL-STD-498 for Acquirers and Developers. Presentation slides are available at http://abelia.com/docs/122_016.pdf.

[32] Jack Caine’s 'Agile Lean slide collection'. Access at www.linkedin.com/in/jack-caine-58690a13. Well over 1,000 very good slides, which one can reuse (Jack told me) for agile presentations. Please credit the original author of the slide.

[33] AGILE FAILURE RATES: Google: 'Agile IT failure rates'.

[33A] Derwyn Harris, “Why we should rethink the Agile Manifesto: Projects Still Fail”. September 10, 2014. See links to several articles concerning this subject at: https://www.google.no/search?client=safari&rls=en&q=agile+it+failure+rate&ie=UTF-8&oe=UTF-8&gfe_rd=cr&dcr=0&ei=LR9NWobkI8-q8wfMq67YBg.

[33B] UK wasting £37 billion a year on failed Agile IT projects. Link provided to this article at https://www.google.no/search?client=safari&rls=en&q=agile+it+failure+rate&ie=UTF-8&oe=UTF-8&gfe_rd=cr&dcr=0&ei=LR9NWobkI8-q8wfMq67YBg.

"The survey found that 34% of failed agile projects failed because of a lack of upfront and ongoing planning. Planning is a casualty of today’s interpretation of the Agile Manifesto, which can cause Agile

teams to lose their way and their customers to lose visibility of what they are getting for their money, now and in the future."

"68% of CIOs agree that agile teams require more Architects. From defining strategy, to championing technical requirements (such as performance and security) to ensuring development teams stick to the rules of the game, the role of the Architect is sorely missed in the agile space. It must be reintroduced. But the report did uncover some relatively good news for the UK tech industry. While the rate of complete failure of agile projects was 12% in the UK – in the US, CIO's report 21% of agile projects result in complete failure."

[34] Ken and Will Hopper: The Puritan Gift. See https://www.amazon.com/Puritan-Gift-Reclaiming-American-Financial/dp/184511986X.

An analysis of how American management went bad, starting with Harvard Business School. Managers still have almost no idea how to 'balance their scorecard', by quantifying business and customer values, as well as they do the 'financials'. Management verbiage still reigns. Musk [22] and Jobs did not get an MBA. But they are value-driven, not finance-narrow.

[35] 'What's Wrong with Agile Methods? Some Principles and Values To Encourage Quantification' with Confirmit Case'. See http://www.methodsandtools.com/archive/archive.php?id=58

[36] Some Alternative Ideas on Agile Values for Delivering Stakeholder Value Principles and Values – Agility is the Tool, Not the Master. Two papers - see links below.

[36A] Gilb's Ten Key Agile Principles to deliver stakeholder value, avoid bureaucracy and give creative freedom" Part 1 of 2. http://www.gilb.com/DL431 (The Paper from Agile Record) (Note: the gilb.com references will work if you delete the "tiki-download_file.php?fileid=" text, and insert DL in front of the number (www.gilb.com/DLnn) where nn is the number at the end of the old URL).

[36B] Part 2 "Values for Value" http://www.gilb.com/DL448 Agile Record 2010, www.agilerecord.com, October 2010, Issue 4. (Note that the gilb.com references will work if you delete the "tiki-download_file.php?fileid=" text, and insert DL in front of the number (www.gilb.com/DLnn) where nn is the number at the end of the old URL).

[37] Advanced Agile Software Engineering (Agile Turkey April 2018 slides talk) Version 2 Jan 2018 1st draft. Might be much changed by presentation date. http://concepts.gilb.com/dl915

[38] Alan Cooper. *The Inmates are Running the Asylum*. https://www.goodreads.com/book/show/44098

[39] Contracting for Value

[39A] Slides http://www.gilb.com/dl86439B. Paper, Agile Contracting for Results: The Next Level of Agile Project Management: Gilb's Methodology Column Agilerecord August 2013. See http://www.gilb.com//dl581.

[40] ON SCALING

[40A] SLIDES. Practical Scaling Methods for Industrial Systems Unicom Conference, London, 31 Oct 2017, slides. http://concepts.gilb.com/dl916.

[40B] Tom Gilb, “Beyond Scaling: Scale-free Principles for Agile Value Delivery - Agile Engineering”. Paper available at http://www.gilb.com/dl865. Concerns Intel experiences with Gilb methods. Jan 8, 2016.

[40C] “SCALE-FREE: Practical Scaling Methods for Industrial Systems Engineering” lecture slides. With reference to Intel experiences with Gilb methods. Available at http://concepts.gilb.com/dl892

[41] Planguage: A Software and Systems Engineering Language, for Evaluating Methods, and Managing Projects for Zero Failure, and Maximum ‘Value Efficiency’. Keynote: International Conference on Software Process and Product Measurement (Mensura). Available at http://concepts.gilb.com/dl918.

[42] Power to the Programmers (about delegation and motivation in practice), as held Krakow ACE Conference, June 2014. Video: http://vimeo.com/98733453 Link to Oct 2015 Prague Slides, “Power to the Programmers”. Available at http://concepts.gilb.com/dl841.

[43] See https://technology.amis.nl/2008/10/03/agile-software-development-the-principles-principle-8-agile-processes-promote-sustainable-development-the-sponsors-developers-and-users-should-be-able-to-maintain-a-constant-pace-indefinitely/.

[44] Gilb, Quantifying Management Bullshit: forcing IT Stakeholders to reveal the value they really want from your IT Project. Available at http://www.gilb.com/dl465.

[45] Tom Gilb, 'Estimating Efficiency of Means for Ends: A Dummies Guide to Impact Estimation Tables'. Available at http://concepts.gilb.com/dl906.

[46] Simplicity Talk. ACCU Conference. Einstein and Simplicity Principles. Available at http://www.gilb.com/DL464.

[47] Gilb, Confronting Wicked Problems: and some Planguage Tools to deal with them. A detailed paper on how Planguage, in practice, can simplify 'wicked' problems (which then are no longer so wicked). Jan 10 2016. Available at http://www.gilb.com/dl866.

[48] Software Engineering Complexity and Challenges of Software Engineering slides. Quality Days, Vienna, Talk, 18 January 2017. Available at http://concepts.gilb.com/dl889.

[49] Practical Tools for Simplification of Software Quality Engineering. Half-Day Afternoon Workshop at Quality Days, Vienna 17 January 2017. Available at http://concepts.gilb.com/dl888.

[50] What is Wrong with Software Architecture? Gilb Keynote in London Oct 2013. Available at https://www.youtube.com/watch?v=HNasoyrxzy8. Or use https://vimeo.com/28763240 (Javazone, Oslo).

What is Wrong with Current Software Architecture Methods: 10 Principles for Improvement, 30 Sept 2013 and Oct London week Available at http://concepts.gilb.com/dl587.

[51] Stakeholder Power: The Key to Project Failure or Success, including 10 Stakeholder Principles. Available at http://concepts.gilb.com/dl880.

[52] Stakeholders and their values, GilbFest 2017, slides. Available at http://concepts.gilb.com/dl920.

[53] Gillian Tett, “The Silo Effect: The Peril of Expertise and the Promise of Breaking Down Barriers”. 2017.

[54] Value Manifesto, Tom Gilb. March 2017. This is the core of my Agile Ideas. http://concepts.gilb.com/dl898 See updated version January 2018 at www.Gilb.com (Blog).

[55] Douglas, Bruce Powell. *Agile Systems Engineering*. Burlington, Massachusetts USA: Morgan Kaufman Publishers (October 28, 2016). ISBN-13: 978-0128021200.

[56] Gilb, Tom, “The 10 Most Powerful Principles for Quality in Software and Software Organizations”.

CrossTalk, November 2002. Available at http://www.crosstalkonline.org/back-issues/.

Note: If any links are broken, please report them to tom@gilb.com.

Author Biography

Tom Gilb was born in Pasadena, California USA in 1940, emigrated to London in 1956, and to Norway in 1958. There he joined IBM for five years. Tom resides and works in Norway when not traveling extensively.

He has mainly worked within the software engineering community, but since 1983 with corporate top management, and since 1988 with large-scale systems engineering (aircraft, telecoms, and electronics).

He is an independent teacher, consultant, and writer. He has published nine books, including the early coining of the term "Software Metrics" (1976) which is the recognized foundation ideas for IBM CMM/SEI CMM/CMMI Level 4.

He wrote, *Principles of Software Engineering Management* (1988, in 2006 in 20th printing), and *Software Inspection* (1993, 14th printing). Both titles are systems engineering books in software disguise. His latest book is *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Management Using Planguage*, published by Elsevier, first published in 2005.

In 2016, Tom released his new management planning book, ‘Value Planning,’ in digital format only.

He is a frequent keynote speaker, invited speaker, panelist, and tutorial speaker at international conferences.

He has published hundreds of papers. One paper (Laws of Unreliability, *Datamation*, March 1975) provided his Laws of Unreliability that received more than 22,000 Google hits.

He has guest lectured at many universities (including U. C. Berkeley, Stanford, Seattle University, London School of Economics, University of Oslo, Technical University of Trondheim, TU Munich, Tampere and Helsinki Technical Universities, University of San Luis Obispo, and The International Institute of Information Technology IIIT Bangalore.

He is recognized as the founder or major driver of several technical disciplines such as 'software metrics' and 'evolutionary project management,' as well as being an innovative pioneer in inspections, and the inventor of the planning language "Planguage". He is recognized as the idea source for parts of the Agile and Extreme Programming methods (primarily the incremental cycles). Tom and his son Kai have developed their Agile Inspections and Agile Evolutionary Project Management processes that are being successfully used by clients.

He consults and teaches in partnership with Kai Gilb, worldwide. He happily contributes teaching and consulting pro bono to developing countries (including India, China, and Russia), to Defense Organizations (UK, USA, Norway, NATO) and charities (Norwegian Christian Aid and others).

He enjoys giving time to anyone, especially students, writers, consultants, and teachers, who are interested in his ideas - or who have some good ideas of their own.

His methods are widely and officially adopted by many organizations including IBM, Nokia, Ericsson, HP, Intel, Citigroup, and many other large and small organizations.

ARTICLE

Why Agile Product Development Systematically Fails, And What to Do About It!

by

Kai Gilb

Value Delivery Expert

Email: Kai@Gilb.com

LinkedIn: www.linkedin.com/in/kaigilb/

Abstract

Agile has been proffered by its proponents as a superior development approach. However, the way it is taught and practiced today, Agile is not succeeding in facilitating product development. The issue is not why it might fail here and there; the issue is why it repeatedly fails. It has been clear to us for a long time that the decades-long IT project failure rate has not been 'cured' by the availability of agile methods. The

2015 CHAOS Report published by the Standish Group reports a 61% failure rate of agile projects. In this article, I will discuss why it fails and what can be done to make it successful.

Introduction

Agile is a methodology created to allow small groups to create high quality products in a relatively short amount of time. Agile has tried to evolve to apply to larger-scale IT projects and has always intended to be responsive to change and emerging needs. Agile has been advocated by its proponents as a superior development approach [1, 4, and 5]. However, the *2015 CHAOS Report* published by the Standish Group [7] reports a 61% failure rate of agile projects. The purpose of this article is to share the experience of the author in serving as value delivery expert and project coach over the past 25 years; specifically, to address why agile development has systematically failed and what he has done for organizations to make it successful. Some Agile methods are not new, for example, Evolutionary Project Management (Evo)[4] [3, Chapter 10] and Cleanroom Software Engineering [13 and 14].

What Does It Mean to Succeed In Product Development?

People are willing to invest a specific amount of resources in a team, expecting to get something specific back. If something can be delivered to them, within the resources made available, then a level of success can be achieved.

The important question is, what is that something?

According to the 2015 CHAOS Report, experience is that those advocating Agile have not understood this question well nor the answer, and as a result, Agile is failing.

Reviewing the literature concerning Agile, it appears that it is obsessed with shippable code, minimum viable product, user-stories, burndown charts, Kanban, standup meetings, and so forth.

However, the people investing in the project are expecting that their jobs, tasks, products, and services get better, easier, faster, cheaper, and more efficient. In our experience, serving as project coach for many organizations, Agile does not prioritize these improvements. Although Agile does mention that value is important, typically “value” is not defined, specified, or quantified. Consequently, the logical next phase of creating success in product development has not happened. The important step is to prioritize all actions, decisions, and solutions towards creating the defined value.

Example Case

[4] Surprisingly, many project cultures have little formal knowledge of Evolutionary Project Management (Evo) (also known as Iterative and Incremental Development [6]) as a modern practice, even though it has been in use since the 1960’s. Prominent software-engineering thought leaders from each succeeding decade supported Evo practices, and many large projects used them successfully. These practices may have differed in their details, but all had a common theme— to avoid a single-pass, sequential, document-driven, gated-step approach. See Larman and Basili, 2003 [6] for a brief history of Evo. See also [3], Chapter 10, Evolutionary Project Management, for an excellent explanation of how to implement it.

Let's say a banker wants information faster. Then that is what should be delivered in order that invested effort is considered to be successful. We need to determine what type of information, under what circumstances, and how fast. To determine how fast, we need first to understand how fast the banker gets that information today. Let's say that for the type of information desired, it takes 3 seconds today. Now we know what would be faster: faster than 3 seconds. Next, we need to set a target, most desirably together with the banker and the development team. Let's say we set the target to 0.5-seconds. I will name it the “Banker-Fast.Goal”. Now we have determined the improvement the banker expects. Delivering the Banker-Fast.Goal, within the resources agreed upon, would be considered a successful product delivery.

To make the Banker-Fast.Goal happen, everyone needs to exercise extreme focus and prioritization towards reaching the 0.5-second goal. Every decision, every action of all people involved, needs to be directed towards shaving off the 2.5 seconds. For all decisions, we must ask: what will best improve the situation for the banker moving from 3 seconds to 0.5-seconds. For every prioritization, we must ask which choice will best improve the situation for the banker moving from 3 seconds towards 0.5 seconds. For every task we do, we must challenge ourselves concerning the degree to which this task improves the situation for the banker from 3 seconds towards 0.5 seconds. A manager should be inquiring of all members of the team, "Concerning the task you are doing right now, how will it improve the situation for the banker from 3 seconds towards the goal of 0.5 seconds?"

Measure

Then we must measure reality, and adjust accordingly. And repeat until we get to 0.5 seconds. That the task is 'done' is entirely irrelevant. *The only relevant question is; how much faster does the banker get the information?*

Agile has not prioritized actions, decisions, and solutions towards creating defined value. If one is concerned with user-stories and trying to organize and distribute tasks, creating burndown charts, and asking about the definition of done, this is not a success-oriented approach. Until we realize this fundamental idea of succeeding in projects, we will keep on delivering code and be experiencing unhappy customers. You might call your project a success, but ask the users and the other stakeholders if they are positively thrilled with the delivery they are getting.

Video Tutorials

The author has made available at no charge for all readers of SyEN video tutorials that explain how one in practice can capture, specify, quantify, measure, prioritize, develop, and deliver value to stakeholders. The three tutorials are provided under the title, What Every Successful Project Manager should know!

Tutorial 1 of 3 - Your customer is "always" wrong! Tutorial 2 of 3 - How not to pay your contractors, unless you succeed! Tutorial 3 of 3 - How you can deliver value early, rapidly and continuously!

The tutorials are available at www.gilb.com/p/SyEN.

Conclusions

The success of agile development and implementation can be further strengthened and improved by clarifying what value is for customers. By defining, specifying, and quantifying what value is for customers, and through prioritizing improvement opportunities, the effectiveness of agile development efforts can result in better products that are provided easier, faster, less expensively, and more efficiently. Recent reports concerning project success rates are encouraging [2, 8]. It appears that we finally are applying the results of the lessons of the past 25 years.

References

[1] Douglas, Bruce Powell. *Agile Systems Engineering*. Burlington, Massachusetts USA: Morgan Kaufman Publishers (October 28, 2016). ISBN-13: 978-0128021200.

[2] Florentine, Sharon, “IT Project Success Rates are Finally Improving”. Framingham, Massachusetts USA: CIO, February 27, 2017. See https://www.cio.com/article/3174516/project-management/it-project-success-rates-finally-improving.html.

[3] Gilb, Tom. *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage* (2005). Available from https://www.gilb.com/p/competitive-engineering

[4] Grau, Rainer, “The Impact of Effective Agile Teams on Organizations”. Victoria, Australia: *The Systems Engineering Newsletter*, published by Project Performance International, Volume 57, September 18, 2017. See http://www.ppi-int.com/wp-content/uploads/2017/11/SyEN-57.pdf.

[5] Larman, Craig and Bas Vodde. *Scaling Lean and Agile Development*. Boston, Massachusetts USA: Addison Wesley, December 8, 2008. ISBN13: 978-0321480965.

[6] Larman, Craig and Victor R. Basili, "Iterative and Incremental Development: A Brief History", *Computer*, vol. 36, no., pp. 47-56, June 2003, doi:10.1109/MC.2003.1204375. Available at http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf

[7] The Standish Group International. The CHAOS Report. Dennis, Massachusetts USA: The Standish Group International, 2015.

[8] Project Management Institute (PMI). *Success Rates Rise: Transforming the High Cost of Low Performance.* Pulse of the Profession 2017. Newtown Square, Pennsylvania USA: The Project Management Institute, February 2017. Available at https://www.pmi.org/learning/thought-leadership/pulse/pulse-of-the-profession-2017.

[9] Young, Ralph R. *Effective Requirements Practices*. Boston, Massachusetts USA: Addison-Wesley, 2001.

[10] Young, Ralph R. *Project Requirements: A Guide to Best Practices*. Vienna, Virginia USA: Management Concepts, 2006.

[11] Zalavadia, Sanjay, “Why Agile Fails in Large Enterprises”. *InfoQ Weekly*, December 1, 2015. See https://www.infoq.com/articles/agile-fails-enterprise.

[12] Gilb, Kai. *Evo Book Manuscript*. Project Management and Product Development: How Successful People Manage Projects Delivering Stakeholder Values and Product Qualities Competitively, Early, and Predictably. Available at https://www.gilb.com/store/kNgRApLA.

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

[14] Mills, H. 1980. The Management of Software Engineering - Part 1: Principles of Software Engineering. *IBM Systems Journal* 19, issue 4 (Dec.):414-420. Available at http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan

Author Biography

Kai Gilb works passionately in his quest to help project teams all around the world learn to succeed in product development. Kai has been the apprentice of his father Tom Gilb since 1992. He considers himself fortunate to have learned from the many brilliant people he has worked with all around the globe.

Kai coaches managers, product owners, and development teams; lectures; develops and facilitates workshops; consults worldwide, and writes articles. Currently, he is authoring a book [9] - ‘Evo – Evolutionary Project Management & Product Development.’

Kai was teaching Agile project management long before the term “Agile” existed, for example, in 1993, teaching at large corporations such as Ericsson and Nokia, and working on large projects including mobile phone base stations and networks.

Among Kai's clients are Alcatel, BAA, Boeing, Bosch, Citibank, Confirmit, Credit Suisse, HP, Microsoft, NTNU, Philips, If Insurance, Qualcomm, Schlumberger, TomTom, UECC, and Statoil.

Kai resides in Oslo, Norway, is married, and the father of two children.

Kai welcomes feedback concerning this article – reach Kai at Kai@Gilb.com.

ARTICLE

Integrating Program Management and Systems Engineering

by

Dr. Ralph R. Young

Editor, SyEN

This month we provide a summary of Chapter 7, Integration in Practice in the F/A-18E/F Super Hornet Program, in the book titled *Integrating Program Management and Systems Engineering* (IPMSE). The book is a collaboration of the International Council on Systems Engineering (INCOSE), the Project Management Institute (PMI), and the Consortium for Engineering Program Excellence (CEPE) at the Massachusetts (USA) Institute of Technology (MIT).

This Chapter provides one of several case studies in 'The Book' which were researched during the development of the book that describe in detail "what it looks like when program managers and systems engineers work together in an integrated fashion". It provides an example of a program that was successful, but, more importantly, that success was decisively enabled by employing the principles described in the **Integration Framework** which was provided and discussed in this column in last month's issue of *SyEN*. This success was achieved in a setting where there were numerous competing demands for attention and resources; where the product requirements were complex and failure could be both punishing and an ever-present possibility; and where multiple specialized interests presented challenges to successful integration. The development of this aircraft provides an example of a program that embodies effective integration. It holds a rarified position as one of the few complex military programs to complete its development phase ahead of schedule (adherence to timeline) and within budgeted cost (adherence to budget).

In addition, this program provided several additional benefits. The aircraft completed its deployment phase weighing 1,000 pounds under its specification (client requirements fulfillment) with one-third fewer parts than predecessor aircraft, a heavier payload, longer range, increased reliability, and margin for additional growth in capabilities. Not only is it meeting its program requirements regarding quantity of aircraft delivered, but it is also expanding into new applications that boost the overall production. Building the full program requirements of aircraft benefits the workforce and supply base by having a stable production program over many years. The overall program benefits to the Navy are even more impressive from the standpoint of significant reduction in the system's total cost of ownership (factoring in operations and maintenance, logistics, and sustainment costs compared with operating the diverse array of aging legacy aircraft that this type replaces).

When viewed through the lens of the key dimensions characterizing effective integration, the Super Hornet Program represented a shift away from fundamental stovepipes to embrace rapid and effective decision-

making, collaborative work, and information sharing. **Program management and systems engineering became more integrated in this program.** By virtue of the relationship between the government customer and its contractor base, it also presents a more complex integration case because both the government and the contractors each had program management and systems engineering disciplines to be integrated into a large, unified program. Effective integration at the program level included integration across the contractual boundary, often a more challenging integration task than between disciplines within a single organization. Considering all of the stakeholders that needed to be integrated to make the program a success, this presented a significant challenge.

The transition away from traditional stovepipe mindsets toward a more integrated approach was initiated by the tops levels of leadership within the U. S. Navy's Naval Air Systems Command (NAVAIR) and McDonnell Douglas Corporation. This case study traces the history of the program as it made this transition. **It illustrates the principles of integration in practice and conveys a sense of what it is like to be in a program where integration is practiced and prioritized.**

Hallmarks of the Super Hornet Program included:

- The objective of the F/A-18 E/F program was to develop, test, produce, and deploy an upgraded F/A-18 with increased mission range, increased aircraft carrier recovery payload, additional growth potential, and enhanced survivability.
- The F/A-18 E/F was immediately critical to naval air planning and the Navy overall. Modernization was essential for both combat effectiveness and operational affordability because the existing fleet of F/A-18 C/Ds consisted of aging aircraft with limited options for capability improvement. The Navy's reputation as an agency capable of effective aircraft procurement was at stake.
- Leadership.
- Use of a 12-day organizational effectiveness meeting that created motivation for success.
- Use of a Navy/contractor integrated test team during Development, Test, and Evaluation (DT&E).
- Reducing program complexity.
- Both contractors provided experienced teams and drew heavily from existing supplies and the industrial base.
- Integration of disciplines required to develop the aircraft.
- Development of a new concept of operations for acquisition management within NAVAIR through which all functional areas worked together as a team.
- Transition to Integrated Product Teams (IPTs).
- Improved decision-making.

- The Work Breakdown Structure aligned with the product.
- Promotion of effective collaboration.
- Empowerment of teams.
- Proactive identification and management of risk.
- Open communication and information sharing.
- Ability to absorb significant funding cuts during development.

The F/A-18E/F Super Hornet case study illustrates in some detail a program where integration was a core value, driven by an understanding that the cost of *not* being integrated was unacceptably high. There was an awareness that the traditional ways that teams had worked together in the past would not meet a critical schedule-driven need to provide key capabilities to the Navy fleet. What emerged was a program that subordinated the individual functional and organizational identities to the needs of the overall program. The mantra of the program leaders was “the product is the boss” and was used to ensure that individuals and teams worked to make the right choices and acted as needed to obtain overall success.

This case study not only provides evidence to support the Integration Framework, but also illustrates how the various elements are expressed in an actual program and how they inter-relate. The numerous organizations involved in the program were deliberate about their intent to have a more integrated, high-functioning relationship between program management and systems engineering. A number of practices and approaches were employed to encourage increased integration. Senior leadership played a defining role in establishing the vision for greater integration and were willing to expend time, resources, and leadership capital over an extended period of time to ensure that the entire program participated in enacting that vision.

The value to you of thoroughly digesting this case study may be discovery of insights that prove invaluable. What are the similarities and differences between your program and the case study? What practice or set of practices would you urge your organization to adopt? What factors from the Integration Framework do you think would be of benefit to your organization if adopted? What principle barriers might you anticipate encountering as you try to replicate proven practices? What technique do you think is most effective for ensuring that stakeholders’ concerns are addressed early in the life cycle of the program?

I commend study of this chapter to every systems engineer. I believe that you will glean insights that will make a difference in the ways in which you practice systems engineering. The book is available here at a discount for members of INCOSE.

SYSTEMS ENGINEERING NEWS

Project Success Rates are Rising!

The Project Management Institute (PMI) reports in its 2017 *Pulse of the Profession* report that Organizations are experiencing more success implementing strategic initiatives and, for the first time in five years, more projects are meeting original goals and business intent and being completed within budget (see the article concerning this report in the SE Publications section, below).

The 2017 *Pulse* findings continue to show what we've learned in the past: that when proven project, program, systems engineering, and portfolio management practices are implemented, projects are more successful. PMI also concluded that the definition of success is evolving. The traditional measures of scope, time, and cost are no longer sufficient. Projects must deliver what they set out to do — the expected benefits. So, for the first time, when determining project success, PMI looked at levels of benefits realization maturity as well as the traditional measures.

Through this wider lens, PMI identified champions and underperformers as two new, measurable performance levels. PMI also identified the organizational practices that distinguish more successful project performance from average or poor performance.

More information

Newly Released Science & Engineering Indicators Document Describes the State of American Technology

The National Science Foundation and the National Science Board, both in the USA, have just released their biennial Science & Engineering Indicators document that provides an extensive compilation of facts and figures on research and development, innovation, and engineers. The document suggests opportunities to further strengthen and improve the technological base, such as overhaul immigration to favor high-skilled newcomers; raise defense spending on new technologies; increase other federal spending on basic research. Basic research describes the situation in which the government provides most of the money for the research, which is the quest of knowledge for its own sake that has cut spending in recent years.

More Information

Suggestions for the Upcoming INCOSE International Symposium

The upcoming INCOSE International Symposium in Washington DC spans the 6th to the 13th of July 2018 and many are hard at work to provide another outstanding event. In recent years, some strategic invited content has been provided as a complement to the peer-reviewed papers, panels, and tutorials that form the backbone of the event.

This year, INCOSE is focusing invited content on "Systems Engineering of the Future" which will build on the insights in INCOSE's Systems Engineering Vision 2025 and explore the global challenges that must be addressed, how systems engineering should evolve to enable our vision and mission, and the research agenda to be pursued.

Some of the areas you might consider in offering suggestions include:

- Relevance to "SE of the future"
- Fit with the INCOSE Vision and Mission
- Novelty and/or innovation
- A topic or voice that is new and relevant to INCOSE
- Something that can be accommodated into the program at low/no cost

Please email your suggestions to info@incose.org.

Capers Jones Publishes "The Mess of Software Metrics"

Capers Jones, industry expert and enormous contributor to systems and software engineering, has published another insightful and telling article, "The Mess of Software Metrics".

The software industry is one of the largest, wealthiest, and most important industries in the modern world. The software industry is also troubled by very poor quality and very high cost structures due to the expense of software development, maintenance, and endemic problems with poor quality control.

Accurate measurements of software development and maintenance costs and accurate measurement of quality would be extremely valuable. But as of 2016 the software industry labors under a variety of non-standard and highly inaccurate measures compounded by very sloppy measurement practices. For that matter, there is little empirical data about the efficacy of software standards themselves.

The industry also lacks effective basic definitions for "*software productivity*" and "*software quality*" and uses a variety of ambiguous definitions that are difficult to predict before software is released and difficult to measure after the software is released. This paper suggests definitions for both economic software productivity and software quality that are both predictable and measureable.

Summary and Conclusions

The current state of software metrics and measurement practices in 2017 is a professional embarrassment. The software industry continues to use metrics proven mathematically to be invalid and which violate standard economic assumptions.

Most universities do not carry out research studies on metrics validity but merely teach common metrics whether they work or not.

Until the software industry has a workable set of productivity and quality metrics that are standardized and widely used, progress will resemble a drunkard's walk. There are dozens of important topics that the software industry should know, but does not have effective data on circa 2017. Following are 21 samples where solid data would be valuable to the software industry:

Table 11: Twenty-One Problems that Lack Effective Metrics and Data Circa 2017

1. How does agile quality and productivity compare to other methods?
2. Does agile work well for projects > 10,000 function points?
3. How effective is pair programming compared to inspections and static analysis?
4. Do ISO/IEC quality standards have any tangible results in lowering defect levels?
5. How effective is the new Software Engineering Method and Theory (SEMAT) method of software engineering? For more information, visit http://semat.org/.
6. What are best productivity rates for 100, 1000, 10,000, and 100,000 function points?
7. What are best quality results for 100, 1000, 10,000, and 100,000 function points?
8. What are the best quality results for Capability Maturity Model Integration (CMMI) levels 1, 2, 3, 4, and 5 for large systems? For more information, visit http://cmmiinstitute.com/
9. What industries have the best software quality results?
10. What countries have the best software quality results?
11. How expensive are requirements and design compared to programming?
12. Do paper documents cost more than source code for defense software?
13. What is the optimal team size and composition for different kinds of software?
14. How does data quality compare to software quality?
15. How many delivered high-severity defects might indicate professional malpractice?
16. How often should software size be renormalized because of continuous growth?
17. How expensive is software governance?
18. What are the measured impacts of software reuse on productivity and quality?
19. What are the measured impacts of unpaid overtime on productivity and schedules?

20. What are the measured impacts of adding people to late software projects?

21. How does SNAP work for Cost of Quality (COQ), TCO, and activity-based costs?

These 21 issues are only the tip of the iceberg and dozens of other important topics are in urgent need of accurate predictions and accurate measurements. The software industry needs an effective suite of accurate and reliable metrics that can be used to predict and measure economic productivity and application quality. Until we have such a suite of effective metrics, software engineering should not be considered to be a true profession.

FEATURED ORGANIZATIONS

Association of Business Process Management Professionals (ABPMP)

ABPMP® is the non-profit professional association dedicated to the field of Business Process Management. Through a global network, ABPMP connects over 15,000+ individuals representing more than 750 corporations and 56 chapters worldwide. As the voice of the Business Process Management community, ABPMP supports the recognition of the BPM profession and discipline, and is dedicated to maintaining the global standard for BPM practices and certification.

ABPMP International is dedicated to the development of Business Process Management (BPM) educational programs and services in support of its goals for Business Process Management professional education. According to the ABPMP International's Guide to The Business Process Management Common Body Of Knowledge (BPM CBOK) ®, BPM is a disciplined approach to identify, design, execute, document, measure, monitor, and control both automated and non-automated business processes to achieve consistent, targeted results aligned with an organization's strategic goals.

In support of the Vision, Mission, and goals of the ABPMP International, the Education Development Committee is responsible to:

- Define the discipline and practice of BPM
- Maintain a BPM Common Body of Knowledge
- Develop and evolve a professional certification program for BPM professionals
    - CBPA® (Certified Business Process Associate) Exam
    - CBPP® (Certified Business Process Professional) Exam
    - CBPL™ (Certified Business Process Leader)
- Develop educational and training programs that support its products and services

- Develop and promote programs that contribute to the advancement of BPM skills and competencies of the BPM profession

Additional objectives of the committee are to:

- Create model curricula for academic programs (community colleges, undergraduate, graduate, and professional continuing education programs)
- Identify and endorse academic and research programs that align to its Guide to the Business Process Management Body of Knowledge (BPM CBOK®)
- Work with academic research organizations to promote an active research agenda in BPM
- Identify and endorse corporate training programs that align to the BPM CBOK® and its Certification programs (see above links)

More Information

The System Dynamics Society

The System Dynamics Society is an international, nonprofit organization devoted to encouraging the development and use of system dynamics and systems thinking around the world. The society provides a forum in which researchers, educators, students, consultants and practitioners in the academic, corporate and public sectors interact to keep abreast of current developments, build on each other's work and introduce newcomers to the field.

A short slide-show, "Introduction to the System Dynamics Society", is presented annually at the international conference. This presentation includes an overview of a few items: system dynamics evolution, the system dynamics society, society activities, products & publications, and how you can participate. This presentation is available here.

The 36th International Conference of the System Dynamics Society will take place in Reykjavík, Iceland from 6 to 10 August, 2018. The call for papers is available here.

CONFERENCES AND MEETINGS

For more information on systems engineering related conferences and meetings, please proceed to our website.

SOME SYSTEMS ENGINEERING-RELEVANT WEBSITES

Relationships between systems engineering and project management

A link to a SEBoK Wiki page containing a discussion and references for this popular topic. There is a lot of commonality between Systems Engineering and Project Management but also some differences. This page explores these similarities and distinctions.

http://sebokwiki.org/wiki/Relationships_between_Systems_Engineering_and_Project_Management

Systems engineering courses

A free 3-hour course on Systems Engineering giving an overview of the important SE topics. The course looks at systems and how they are important. The course explores modern engineering, systems methodologies, systems concepts, ways of thinking and systems modelling and diagramming.

https://alison.com/course/systems-engineering

Tech career profile: systems engineer

A website outlining the technical and non-technical skills a sound systems engineer should exhibit. The website goes further to outline the responsibilities, career opportunities and objectives for a systems engineer in practice. A useful checklist to assess one's skillset at a high level.

https://www.thebalance.com/tech-career-profile-systems-engineer-2071298

On project management

An online blogging community dedicated to assisting in project management by providing a place for project managers to improve their skills for free. The site contains tutorials and discussions project-management related that will be useful for any project manager or team leader.

http://www.projectengineer.net/

Value Planning

Practical Tools for Clearer Management Communication

by

Tom Gilb

Book description (from Tom Gilb's website):

This book is for project planners of any domain and level, who are focused on delivering measurable results fast. They must be willing to adopt a new culture or mindset that is based on the quantification of critical improvement values for any product, service, or organization. This book is intended for deep training, or self-training, in the Planning Language (Planguage).

Here is *how* this book's methods will help you:

- Shows you how to *quickly focus* on the critical results, the 'core', not the detail
- Shows you how to *quantify* all critical results, to facilitate clarity concerning ideas
- Shows you how to *estimate the impacts* of all your solutions and strategies, so that you can *evaluate* them and *prioritize* them, for *measurable early 'value* streams' (increments (Δ) of priority values)
- Shows you how to complete your 'feasibility study', in less than a week, even for very large projects
- Shows you how to dive in quickly (in days), and begin to see, real measurable change
- Shows you how to learn early, what works, and what it costs, and quickly find better options when results are disappointing
- Shows you how to connect many levels of concern: your stakeholders, management, team, and your downstream clients, customers, users and prospects
- Shows you how to understand when sales pitches, meeting presentations, and arguments, are wasted and meaningless: and what to do about it

Order the book here. Watch the five-minute video about the book here.

A PDF copy of the manuscript is also available here.

Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage

Image source

by

Tom Gilb

Book description (from the Amazon website):

Competitive Engineering documents Tom Gilb's unique, ground-breaking approach to communicating management objectives and systems engineering requirements, clearly and unambiguously.

Competitive Engineering is a revelation for anyone involved in management and risk control. Already used by thousands of project managers and systems engineers around the world, **this is a handbook for initiating, controlling, and delivering complex projects on time and within budget.** The Competitive Engineering methodology provides a practical set of tools and techniques that enable readers to effectively design, manage, and deliver results in any complex organization - in engineering, industry, systems engineering, software, IT, the service sector, and beyond. The methods described by Gilb in this book, often referred to in the systems and software literature as “Gilb’s Methods”, include scale of measure, design engineering, risk management, specification quality control, impact estimation, and evolutionary project management (Evo). Gilb has evolved a specification language called “**Planguage**”; his objective in providing Planguage is to support concepts, ideas, and methods. Planguage includes numerous practical strategies for projects to adopt, for example, quantifying requirements, maximize profit (not minimize risk), design out unacceptable risk, design in redundancy, monitor reality (get early, frequent, and measurable feedback from customers), reduce risk exposure, communicate about risk (so that there are no unexpected risks), reuse what you learn about risk, delegate personal responsibility for risk, and contract out risk.

Elegant, comprehensive and accessible, the Competitive Engineering methodology provides a practical set of tools and techniques that enable readers to effectively design, manage and deliver results in any complex organization - in engineering, industry, systems engineering, software, IT, the service sector and beyond.

- Provides detailed, practical and innovative coverage of key subjects including requirements specification, design evaluation, specification quality control and evolutionary project management
- Offers a complete, proven and meaningful 'end-to-end' process for specifying, evaluating, managing and delivering high quality solutions
- Tom Gilb's clients include HP, Intel, CitiGroup, IBM, Nokia and the US Department of Defense

More Information

Success Rates Rise

Transforming the High Cost of Low Performance

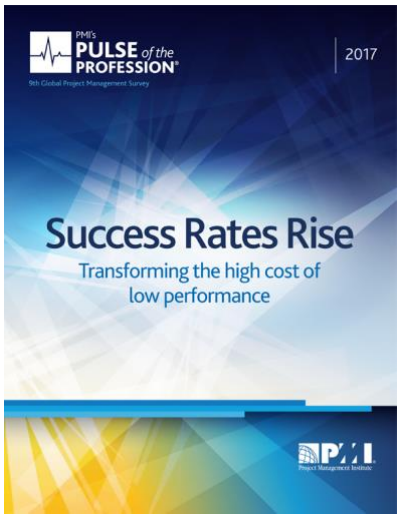

Image source

by

The Project Management Institute

Conducted since 2006, PMI's *Pulse of the Profession*® is the global survey of project management practitioners. The *Pulse* charts the major trends for project management now and in the future. It features original market research that reports feedback and insights from project, program, and portfolio managers, along with an analysis of third-party data.

The newest edition of the *Pulse* features feedback and insights from 3,234 project management professionals, 200 senior executives, and 510 PMO directors from a range of industries, and interviews with 10 corporate leaders and 7 PMO directors and directors of project management. Respondents span North America; Asia Pacific; Europe, the Middle East and Africa (EMEA); and Latin America and Caribbean regions.

The latest *Pulse of the Profession* research suggests a positive change in the way organizations are managing projects and programs. For the first time in five years, more projects are meeting original goals and business intent and being completed within budget. There has also been a significant decline in dollars lost: Organizations are wasting an average of $97 million for every $1 billion invested, due to poor project performance—that's a 20 percent decline from one year ago.

This report, PMI's annual survey of project management practitioners and leaders, strives to advance the conversation around the value of project management. The research represents feedback from 3,234 professionals globally who represent different levels within organizations from diverse industries. The findings continue to show what we have learned in the past: that when proven project, program, and portfolio management practices are implemented, projects are more successful.

At the same time, the definition of success is evolving. The traditional measures of scope, time, and cost are no longer sufficient in today's competitive environment. The ability of projects to deliver what they set out to do—the expected benefits—is just as important. So, for the first time, when determining project success, PMI looked at levels of benefits realization maturity as well as the traditional measures. Through this wider lens, two new performance levels were identified among responding organizations:

CHAMPIONS: Organizations with 80 percent or more of projects being completed on time and on budget, and meeting original goals and business intent—and having high benefits realization maturity.

UNDERPERFORMERS: Organizations with 60 percent or fewer projects being completed on time and on budget, and meeting original goals and business intent—and having low benefits realization maturity.

As expected, champions have higher project success rates (92% versus 33% of underperformers) and enjoy more successful business outcomes: They waste significantly less money due to poor project performance. These findings suggest that organizations are becoming more mature with their project management practices and are investing in the following factors that distinguish more successful project performance.

Download the report (Check to ensure that you are downloading in the preferred language) here.

Agile Project Management for Dummies

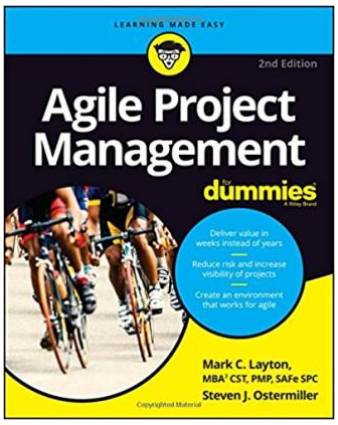

Image source

by

Mark C. Layton and Steven J. Ostermiller

Book description (from the Amazon website):

Agile project management is a fast and flexible approach to managing all projects, not just software development. By learning the principles and techniques in this book, you'll be able to create a product roadmap, schedule projects, and prepare for product launches with the ease of agile software developers. You'll discover how to manage scope, time, and cost, as well as team dynamics, quality, and risk of every project.

As mobile and web technologies continue to evolve rapidly, there is added pressure to develop and implement software projects in weeks instead of months—and *Agile Project Management For Dummies* can help you do just that. Providing a simple, step-by-step guide to Agile project management approaches, tools, and techniques, it shows product and project managers how to complete and implement projects more quickly than ever.

- Complete projects in weeks instead of months
- Reduce risk and leverage core benefits for projects
- Turn Agile theory into practice for all industries
- Effectively create an Agile environment

Get ready to grasp and apply agile principles for faster, more accurate development.

More Information

Writings of Interest

by Tom Gilb

Everyday Superpowers

This is a paper that provides information concerning the methods taught by Tom Gilb (Version 020118). These methods include: 1) Quantify value; 2) Decompose solutions; 3) Quantify and measure the value of technical specifications; 4) Estimate solution attributes; 5) Identify good solutions; 6) Learn numerically what works; 7) Prioritize fact-based solutions and requirements dynamically; 8) Quantify risks; 9) Understand stakeholders; and 10) Communicate clearly. The description of each method is limited to one page. The expected results from using these methods are: 1) Order of magnitude improvements in saving time and money; 2) the methods will work effectively at any scale of problem or product; 3) the powers help deal with problems upstream, early, and they prevent problems; 4) the methods work in agile value delivery cycles.

Available here.

Some Deeper and Broader Perspectives on Evolutionary Delivery (Evo) and Related Technology

In 1994, the U.S. Department of Defense issued a temporary military standard, MIL-STD-498, which explicitly supported the use of evolutionary project management (Evo). It also supported the related concept that projects initially do not have the 'final and correct user requirements' specified. This standard has now evolved into civil standards (such as IEEE standards) and is continuing to influence new standards. Such recognition for Evo is deserved since it has the best track record of any known project and management method (Larman and Basili, "Iterative and Incremental Development: A Brief History", *IEEE Computer*, June 2003, Pages 2-11). Evo has a track record of success; unfortunately, it is not widely known or practiced within systems engineering and other communities. This chapter provides the basic concepts of Evo: Learning from doing and acting on that learning; guided primarily by well-defined, quantified, but not necessarily static, multiple requirements for performance and cost; requirements that represent the value system of the stakeholders; controlling risks; demands early delivery of the high priority improvements. Evo provides better results, faster delivery to market, and schedule, risk, and cost control.

Chapter 10 in *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*

Available here.

Introduction to Impact Estimation Tables

Chapter 9 in *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*

Systems engineers and managers need a reliable way of analyzing how effective their design ideas and strategies are in meeting the requirements. Surprisingly, there are few methods being taught or used to do this. Impact Estimation (IE) is one of those methods. An advantage is that it uses quantified rigor. The intention of IE is that it helps answer the question of how design ideas impact all of a system's critical performance attributes (such as usability and reliability) and all of its resource budgets (such as the financial cost and staff headcount) for implementation and for running in operation. These topics are fundamental to providing effective systems engineering. IE can be used for a variety of project needs. Its most important uses include:

- Comparing alternative design ideas: "What's best?"
- Estimating the state of the overall design architecture: "Have we designed enough?"
- Analyzing risk: "Where are our biggest problems now?"
- Planning and controlling evolutionary delivery steps: "Is the project on track?"

In summary, IE is a practical method that can be used throughout the entire life cycle of a project to help identify and evaluate design ideas against system requirements.

Available here.

Stakeholder Power: The Key to Project Failure or Success

This article provides the definition of a stakeholder, a process for interaction with stakeholders, and "Ten Stakeholder Principles":

- Some stakeholders are more critical to your system than others.
- Some stakeholder needs are more critical to your system than others.
- Stakeholders are undisciplined: they may not know all their needs, or know them precisely, or know their value. But they can be analyzed, coached, and helped to get the best possible representation.
- Stakeholders may be inaccessible, unwilling, inanimate, oppositional, and worse; nevertheless, we need to deal with them intelligently.
- Stakeholders might well ask for the wrong thing, a 'means' rather than their real 'ends'. But they can be guided to understand that. Or their requests can be interpreted in their own real best interests.

- Stakeholders do not want to wait years, experience delays, invest money, and then receive little or no value. They want as much 'value improvement' of their current situation as they can get, as fast as they can get it, and for as little cost as possible.
- Stakeholders are not likely to have any realistic idea of what their real needs and demands are, nor what it will cost to satisfy them. So their evolved real requirements need to be based on value for costs, not on value alone. Delivering small increments, based on high value-to-cost, is one smart way to deal with this.
- If you think you have found 'all of the critical stakeholders', you should assume there is at least one more, and when you find that one, it's quite likely there is another. They will emerge, and they are not all there at the beginning.
- If you think you have found all critical needs of a stakeholder, there will always be at least one more need hiding, more likely several.
- If you do not understand and act on these principles, you will blame your failure on 'system complexity', and the unexpected and wicked problems. But in reality, it is your own fault and responsibility – a more positive and effective approach is to deal with it - up front, and constantly.

Available here.

Writing of Interest

by Tom and Kai Gilb

User Stories: A Skeptical View

This article, in Gilb's Mythodology Column, Tom and Kai Gilb argue that user stories need to be improved to support large projects. They describe six myths concerning user stories, and provide explanations concerning how each of these areas should be improved. They believe that user stories are useful, but they need more relevance for the more demanding project environments – they are too simple as a primary description for many environments.

Download the article here.

Practical Merger and Acquisition Execution and Integration: A Step by Step Guide to Successful Strategy, Risk and Integration Management

by

Michael McGrath

Book Description (from the Amazon website):

Few business activities can match Mergers & Acquisitions (M&A) in terms of the potential for reward and for danger. A successful merger or acquisition can allow a mid-tier company to leap into the top tier, bringing rich rewards to that company, and its employees and shareholders. The failure of a merger can, on the other hand, have a devastating impact, resulting a loss of credibility, destruction of value and in some cases bringing the parties to ruin. Depending on how you measure it, between 50% and 80% of M&A deals fail to attain their objectives, before or even after the deal is done. *Practical M&A Execution and Integration* is all about maximizing your chances of success.

Merging, de-merging, acquiring or acquired, if your organization is involved, or likely to be involved, you will need to manage the process, and following this Handbook will give you a clear, simple framework to get the job done and help your organization move on and attain the benefits and promise of the deal.

The book covers the following core topics:

- Fundamentals of M&A; the reasons for M&A, types of M&A deals and the challenges they present
- M&A Regulation
- Successful M&A, covering M&A power and providing a detailed look at the processes and people involved
- Delivering M&A
- The unique issues of Banking M&A, which differs significantly from other types of M&A deals.

The final section consists of document templates and suggested tables of contents which are designed to be used alongside the advice in the book, thus making *Practical M&A Execution and Integration* the complete guide to constructing a successful M&A deal.

More information

Communicating Project Management: The Integrated Vocabulary of Project Management and Systems Engineering

by

Hal Mooz and Kevin Forsberg

Book Description (from the Amazon website):

This integrated dictionary includes almost 2,000 terms in both project management and system engineering and software engineering by extension defined in a way that seamlessly integrates these overlapping and intertwined fields. Supported by illustrations and explanations that offer a practical context for the terminology, this one-of-a-kind resource bridges the gap between the separate vocabularies of these intersecting disciplines. Far more than a dictionary, this book includes reference sections that address the special problems of and techniques for communicating in the project environment.

More Information

Systems Science, Systems Thinking, Systems and Software Engineering

Addressing Transdisciplinary Frontiers, Practice and Education

The Systems series publishes books related to Systems Science, Systems Thinking, Systems Engineering and Software Engineering.

Systems Science having its contemporary roots in the first half of the 20th century is today made up of a diversity of approaches that have entered different fields of investigation. Systems Science explores how common features manifest in natural and social systems of varying complexity in order to provide scientific foundations for describing, understanding and designing systems.

Systems Thinking has grown during the latter part of the 20th century into highly useful discipline independent methods, languages and practices. Systems Thinking focuses upon applying concepts, principles, and paradigms in the analysis of the holistic structural and behavioral properties of complex systems - in particular the patterns of relationships that arise in the interactions of multiple systems.

Systems and Software Engineering Systems Engineering has gained momentum during the latter part of the 20th century and has led to engineering related practices and standards that can be used in the life cycle management of complex systems. Software Engineering has continued to grow in importance as the software content of most complex systems has steadily increased and in many cases have become the dominant elements. Both Systems and Software Engineering focus upon transforming the need for a system into products and services that meet the need in an effective, reliable and cost effective manner.

While there are similarities between Systems and Software Engineering, the unique properties of software often require special expertise and approaches to life cycle management.

Systems Science, Systems Thinking, as well as Systems and Software Engineering can, and need to, be considered complementary in establishing the capability to individually and collectively "think" and "act" in terms of systems in order to face the complex challenges of modern systems.

SYSTEMS ENGINEERING TOOLS NEWS

Major Release for Capella Studio

Capella Studio Version 1.2, a MBSE solution for systems architectural design, was released in December 2017. Capella is an Open Source product hosted at polarsys.org. Capella provides a process and tooling for graphical modeling of systems, hardware or software architectures, in accordance with the principles and recommendations defined by the Arcadia method. Capella is an initiative of PolarSys, one of several Eclipse Foundation working groups. The changes in Release 1.2 are described at https://wiki.polarsys.org/Capella/Release_Notes#Capella_1.2.0

The Arcadia Method

The Arcadia method is based on the following principles:

- All engineering stakeholders share the same language, method set of engineering artifacts and information, description of the need and the product itself as a shared model;
- Each set of constraints (e.g. safety, performance, cost, mass, etc.) is formalized in a "viewpoint" against which each candidate architecture will be checked;
- Architecture verification rules are established and the model is challenged against them, so as to check that architecture definition meets expectations, as early as possible in the process;
- Co-engineering between the different levels of engineering is supported by the joint development of models. Models of various levels of the architecture and trade-offs are deduced, validated and/or connected with each other.

More information

Automated Scrum Process with Integrated Agile Toolset

Visual Paradigm is an enterprise management and software development suite, which provides features for enterprise architecture, project management, software development and team collaboration.

One of the many solutions provided by this tool is an automated Scrum process with an integrated Agile toolset.

The solution provides for:

- The breakdown of complicated software features into user activities, user tasks and epics and subsequently to user stories.
- The overseeing of user stories of the entire product in the user story screen.
- The creation, prioritization and relocation of user stories on-the-fly.
- The estimation of user stories based on their risks and the effort required to achieve them.
- The editing, prioritization and labelling of user stories with a drag-and-drop graphical user interface and managing and organizing them within multiple sprints and epics.
- The viewing and updating of the progress of user stories with drag-and-drop functionality.
- The automatic generation of user stories through a list of simple descriptions of features presented from the perspective of users.

A detailed flow is provided at https://www.visual-paradigm.com/solution/agiledev/automated-scrum-process/.

EDUCATION AND ACADEMIA

Systems Engineering at Drexel University

Philadelphia, Pennsylvania, USA

Drexel University offers an online Master's degree in systems engineering. The program is organized into four 10-week quarters per year (as opposed to the traditional two semester system) which means one can take more courses in a shorter time period. One semester credit is equivalent to 1.5 quarter credits. One can transfer up to 15 quarter credits from your institution, pending review from the Program Director.

As a top-ranked, regionally accredited institution, known for its interdisciplinary approach to applied education, Drexel consistently gets high marks from employers for its online degree and certificate programs. Their mission to provide practical knowledge for professional success.

Unlike most online education providers, Drexel offers study abroad opportunities for all virtual students.

More information

STANDARDS AND GUIDES

Software Engineering Standards. A User's Road Map

The new report from Telecoms and Computing Market Reports has been published. It provides an analysis of telecoms and computing industries updated in 2018.

The document describes the scope, roles, uses, and development trends of the most widely used software engineering standards. The book concentrates on important software engineering activities quality and project management, system engineering, dependability, and safety. The analysis and regrouping of the standard collections exposes you to key relationships between standards. **James Moore** provides valuable insights that allow you to select standards to fill your specific needs with precision.

Software Engineering Standards: A User's Road Map presents two types of diagrams that will guide you in designating and selecting the standards that meet your specific goals. The first is a layered view of standards that illustrates the internal relationships among standards within a collection. The road map, the second type, illustrates the external relationships among standards in same or different collections. By using the road map diagrams each chapter begins at a different starting point and leads you towards the selection of the software engineering standards that achieve your goals.

More information

A Beginner's Guide to Understanding the Agile Method

The agile method is a particular approach to project management that is utilized in software development. This method assists teams in responding to the unpredictability of constructing software. It uses incremental, iterative work sequences that are commonly known as sprints. This primer, provided by Linchpin Search Engine Optimization (SEO) in Raleigh, North Carolina USA, provides an overview of the agile process and method, including its general principles, history dating back to the 1970s, IBM's uses of agile, benefits of the agile method and criticisms of it, differences between agile and traditional (waterfall or spiral) development, and a glossary of terms associated with agile methods.

More information

Agile for the Enterprise: Five Steps to Successful Adoption

This article, by Dorian Simpson of Jama Software, notes that adopting Agile requires learning some new skills. The article is a thorough description of five recommended steps to achieve successful adoption of agile:

Align the vision of your agile initiative with planned iterations. Getting started too quickly can be wasteful and set your team up for quick frustration. Time and energy should be invested up front to gain a solid grounding in the product vision before distilling it into business requirements to provide direction.

Clarify the role of the Product Owners. The Product Owners are responsible for being the voice of the customer, the evangelists and decision-makers; they provide the appropriate blend of business acumen and technical savvy. Consider this role as a set of activities, interactions and desired outcomes rather than a job title. It is a hands-on role that requires time and commitment.

Build in customer feedback loops. A danger is that developers and Product Owners might take shortcuts, for example, provide personal opinions statements such as, “I’d want this feature,” “It should work like this,” or “The customer will need this,” to drive product decisions, rather than build in real customer feedback. The team must identify a set of target customers that can be relied upon to provide accurate, timely insight. Build skills to actively listen to them. Communicate learned insights into the development efforts by developing clear, prioritized use cases, explain the relative value of each feature and build test cases that reflect how your customers would want to experience your product.

Develop a “WaterScrumFall” Process. Management needs a roadmap, a schedule, a vision document, and a plan. This is one of the main reasons that many companies either overtly or unknowingly create a hybrid of Waterfall and Agile. They use Waterfall to clarify the front end to develop a plan, and then allow the development team to take over and use their Agile approach. Once the product is near market release, the team will attempt to go back to the plan developed under Waterfall thinking. This often includes applying the launch-readiness criteria developed in the front-end planning and gets a quick response from management. While this hybrid process can work, it creates great strain on the organization due to the management team following one process and the development team using different process philosophies, terms, and metrics. True Agile requires the plan to be consistently reprioritized and revised.

The marketing and sales teams must be clear on what customers deem most important and how market dynamics are impacting solution requirements to guide Agile efforts with every Agile sprint in order to guide the development team’s iterative approach.

In summary, Agile must provide an effective business process for delivering greater value to customers and competitive offerings to the marketplace.

More information

Agile Project Management for Dummies Cheat Sheet

This article, included in the *Agile Project Management for Dummies* book described above, is not really a “sheet” but rather a brief overview of the agile process and method. It identifies the Manifesto for Agile Software Development, commonly known as the Agile Manifesto, as an intentionally streamlined expression of the core values of agile project management, and provides the 212 Agile Principles. It

provides a “Roadmap to Value”, a high-level view of an Agile project. The stages of the Roadmap to Value are described in a diagram and a list. It describes the Agile project team members, including the product owner, the development team members, the Scrum master (project facilitator), stakeholders, and Agile mentor; and also Agile project artifacts, including the product vision statement, product roadmap, product backlog, release plan, Scrum backlog, and increment. It describes Agile project events, as well as Agile project management organizations, certifications, and resources.

More information

DEFINITIONS TO CLOSE ON

Agile software development (from Wikipedia)

Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams and their customer(s)/end user(s). It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.

More information

Graphical user interface (From Wikipedia)

The graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of Command-Line Interfaces (CLIs), which require commands to be typed on a computer keyboard.

More information

PPI AND CTI NEWS

PPI and CTI - A Year in Review - 2017 highlights

2017 was indeed a very busy year for PPI. We are delighted to announce that a total of sixty-four courses were held globally, with a strong participation rate from Europe and North America. Who would have thought that we would also leave our mark, for the very first time, in Sudan! Robert Halligan was invited by Zeta Automation Systems CO. LTD., to conduct our popular Systems Engineering 5-day course, which was a huge success. This is a company progressive in the industrial controls field. At the same time, twenty-six ASEP/CSEP preparation courses were delivered by our subsidiary company, CTI.

In addition, PPI and CTI exhibited at twelve systems engineering related conferences.

We are immensely proud to say that during that same year, three distinguished names were added to our ever-growing list of highly regarded presenters we warmly welcomed Suja Joseph-Malherbe, Paul Davies and Hanno Retief.

Suja's successes in 2017 were numerous. Not only is she the current INCOSE South Africa's president and an active contributor to our monthly newsletter SyEN, but she also attained the CSEP certification and went on to present a number of our CTI's courses with glowing results and all of this in just the one year! She is an experienced individual, quickly earning our respect and admiration. She is quite the achiever making a sound and firm impact! She holds a Master's degree in electrical and electronic engineering and a Bachelor's degree in electrical engineering.

Paul, who is based in the United Kingdom, is a systems engineer with many decades of experience across the defense, aerospace, nuclear and rail industries. He is a past president of the INCOSE UK Chapter and has presented a number of CTI courses since he joined our team in July 2017. The feedback from delegates who have been attending Paul's courses have been exceptionally good!

Another up-and-coming presenter is Hanno Retief, also based in South Africa, and whose knowledge and skills in the field of software development are well-known. Hanno is currently conducting PPI's Software Engineering courses which are quickly gaining popularity.

Just as 2017 brought about some very exciting and fulfilling achievements we just know that 2018 will bring about more memorable moments and accomplishments. Stay tuned!

UPCOMING PPI AND CTI PARTICIPATION IN PROFESSIONAL CONFERENCES

PPI will be participating in the following upcoming events.

The 12th Annual IEEE International Systems Conference

(Sponsoring)

23 - 26 April 2018

Vancouver, Canada

INCOSE IS2018

(Exhibiting)

7 – 12 July 2018

Washington, DC, USA

INCOSE SA 2018

(Exhibiting & Sponsoring)

3 - 5 October 2018

Pretoria, South Africa

PPI AND CTI EVENTS

Systems Engineering 5-Day Courses

Upcoming locations include:

- Amsterdam, the Netherlands
- Melbourne, Australia
- London, United Kingdom

Requirements Analysis and Specification Writing 5-Day Courses

Upcoming locations include:

- Pretoria, South Africa
- Adelaide, Australia
- Ankara, Turkey

Systems Engineering Management 5-Day Courses

Upcoming locations include:

- Stellenbosch, South Africa
- Amsterdam, the Netherlands
- Pretoria, South Africa

Requirements, OCD and CONOPS in Military Capability Development 5-Day Courses

Upcoming locations include:

- Melbourne, Australia
- Pretoria, South Africa

Architectural Design 5-Day Course

Upcoming locations include:

- Pretoria, South Africa
- Adelaide, Australia
- London, United Kingdom

Software Engineering 5-Day Courses

Upcoming locations include:

- London, United Kingdom
- Pretoria, South Africa
- Stellenbosch, South Africa

Human Systems Integration Public 5-Day Courses

Upcoming locations include:

- Melbourne, Australia

CSEP Preparation 5-Day Courses (Presented by Certification Training International, a PPI company)

Upcoming locations include:

- Denver, Colorado, United States of America
- Pretoria, South Africa
- New York, United States of America

Kind regards from the SyEN team:

Robert Halligan, Editor-in-Chief, email: rhalligan@ppi-int.com

Dr. Ralph Young, Editor, email: ryoung@ppi-int.com

Suja Joseph-Malherbe, Managing Editor, email: smalherbe@ppi-int.com

Project Performance International

2 Parkgate Drive, Ringwood North, Vic 3134 Australia Tel: +61 3 9876 7345 Fax: +61 3 9876 2664

Tel Brasil: +55 12 3937 6390

Tel UK: +44 20 3608 6754

Tel USA: +1 888 772 5174

Web: www.ppi-int.com

Email: contact@ppi-int.com

Tell us what you think of SyEN. Email us at syen@ppi-int.info.