# 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).

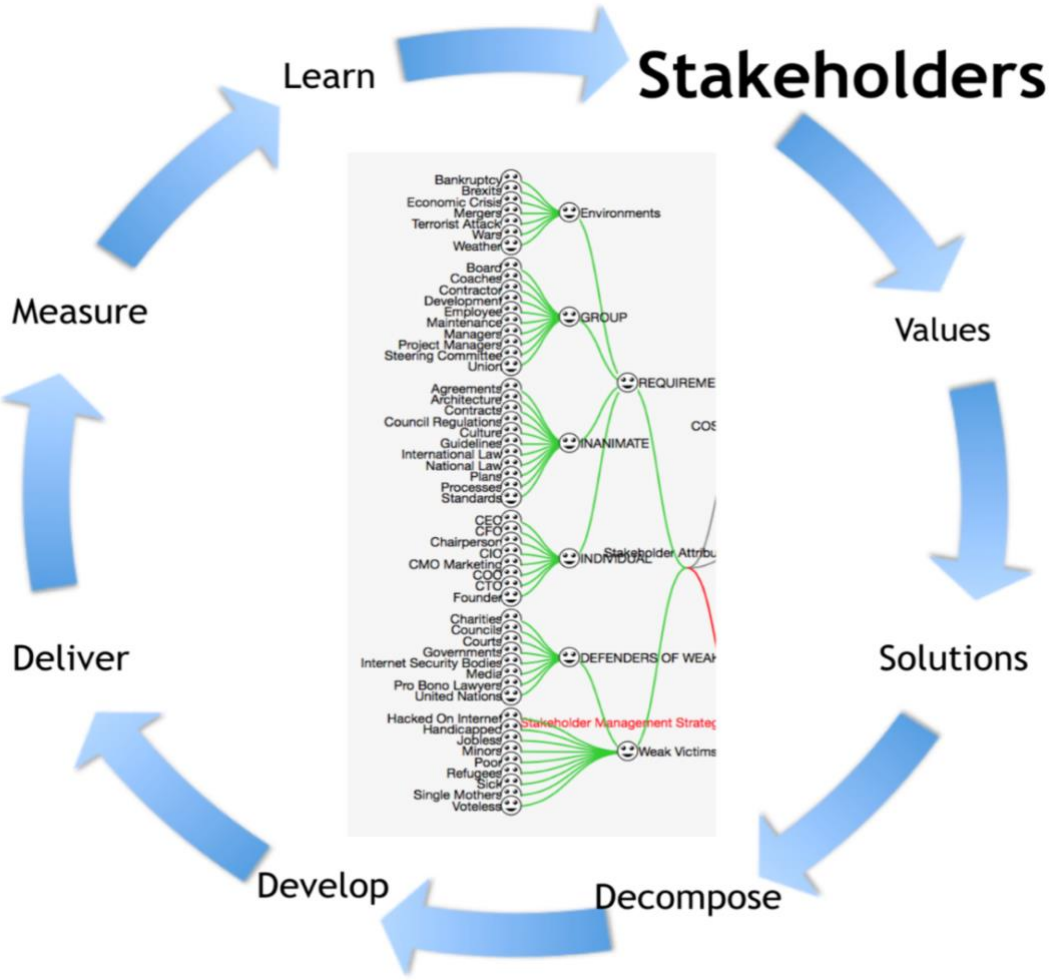


*Figure 1: Continuous Iterative Interaction of Stakeholders with Requirements*

***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:

12*. 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