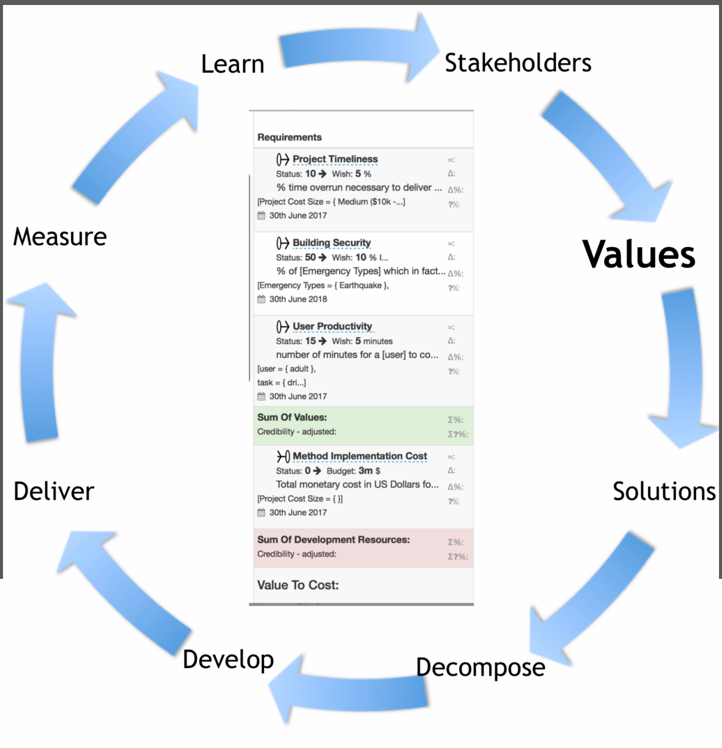
‘VALUE AGILE’

AGILE AS IT SHOULD BE



Requirements

Project Timeliness =:
Status: 10 → Wish: 5 % Δ:
% time overrun necessary to deliver ... Δ%:
[Project Cost Size = { Medium ($10k -...] ?%:
30th June 2017

Building Security =:
Status: 50 → Wish: 10 % l... Δ:
% of [Emergency Types] which in fact... Δ%:
[Emergency Types = { Earthquake }, ?%:
30th June 2018

User Productivity =:
Status: 15 → Wish: 5 minutes Δ:
number of minutes for a [user] to co... Δ%:
[user = { adult }, ?%:
task = { dri...]
30th June 2017

Sum Of Values: Σ%:
Credibility - adjusted: Σ?%:

Method Implementation Cost =:
Status: 0 → Budget: 3m $ Δ:
Total monetary cost in US Dollars fo... Δ%:
[Project Cost Size = { }] ?%:
30th June 2017

Sum Of Development Resources: Σ%:
Credibility - adjusted: Σ?%:

Value To Cost:

By

Tom Gilb

WHO IS THIS BOOK FOR?

This booklet is for for people who are ready for a critical analysis of the 'agile hype' that has been spread.

The problem is not agile itself, nor is it the people spreading agile. It is the people *buying into* agile, who seem to have lost their critical sense, and common sense. But that is not new in history[1].

This book will have no useful effect on the multitude of uncritical people who have bought in to agile, and practice it today. They are probably not capable of changing their religion, or understanding the ideas here.

This book is for highly intelligent, critical, idealistic and energetic people, who already feel the need for something which will work much better for them, and their organizations.

This is for open-minded people who are willing to learn more, and work harder, in order to get much better results for their projects and their organizations.

Experience historically is that the masses want *simple* solutions, even if they do not work. Even if they destroy their organization, or their nation.

So this book is for people who are willing to be idea leaders, and lead their colleagues and fellow citizens. Our real common purpose is to plan and communicate, and do projects so that we really do produce high values, at low costs, with low risks, and close to zero large failures.

[1] I am thinking of some political and religious ideas which have caused humanity great troubles for a long time, and still persist.

My experience is that less than 1% (maybe 0.1%) of professionals have the idealism and energy to take the lead in a culture change. But if you are that exceptional person[2], then I hope this book, and its supplementary references, will give you hope, courage and much better results for yourself, your projects, your organization and your nation.

Right now we have a crisis in management, and in communication with each other, and we do not even seem to realize how bad it is, and what its causes are.

I can tell you right away that the problem is 'management bullshit', and the solution is 'crystal clear communication'. I will also teach you exactly how to communicate better in practice. It is not very difficult, you mainly have to decide to get a lot clearer, like 100X clearer, 100X less BS. Is that clear?

We will start with an in-depth analysis of the Infamous 'Agile Manifesto'. We will continue with a Chapter on why projects get messed up. On our way we will give quite a few, usually freely-downloadable, references, for constructive 'how to' detail.

The result for some of you is that you can get started on a lifetime journey, and maybe really help save civilization, from destroying itself.[3]

[2] if you are not that exceptional person, but you know one such energetic idealist, please pass this book on to them, with my compliments.

[3] I can offer you a starter on exactly that subject. My booklet 'Sustainability Planning' (September 2019) which looks at United Nations Goals in depth. Short term https://www.dropbox.com/sh/gc65fds9h0gv3cm/AABJvW4fwAnqVn25bPtY9bmia?dl=0, and longer term see my website www.Gilb.com. Book Reference A.

INTRODUCTION

HOW WELL DOES THE AGILE MANIFESTO ALIGN WITH PRINCIPLES THAT LEAD TO SUCCESS IN PRODUCT DEVELOPMENT?

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 then had more-quantified documented success, than any of the The Agile Manifesto offspring. I was not alone, but rather, a quiet minority.

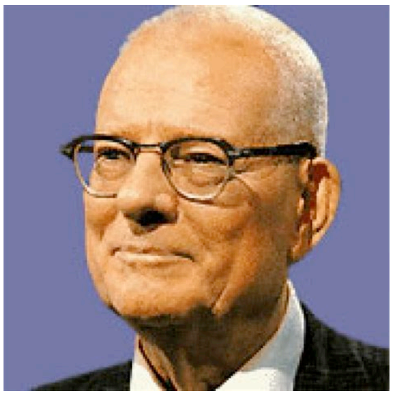
Unfortunately, the Agile Manifesto states embarrassing platitudes, with no visible foundation or purpose.

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 particular agile manifesto 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[4] said.

Figure 1.1a Deming. His Plan Do Study Act cycle, “Deming/Shewhart Cycle’ is an early method formalization of incremental result delivery (agile).[5] Long before ‘software’. He is saying that if you make bad choices in your development methods, you might totally fail. But that is not his problem.

If I were to put blame on a single factor, I would blame the management MBA culture.

[4] *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).

[5] Original Deming Lecture to Top Management 1950 Japan:
http://hclectures.blogspot.no/1970/08/demings-1950-lecture-to-japanese.html
https://blog.deming.org/2012/11/speech-by-dr-deming-to-japanese-business-leaders-in-1950/

Too much 'bean counting', and too little about 'managing values' and 'delivering qualities' that actually provide financial value. [34, 22].

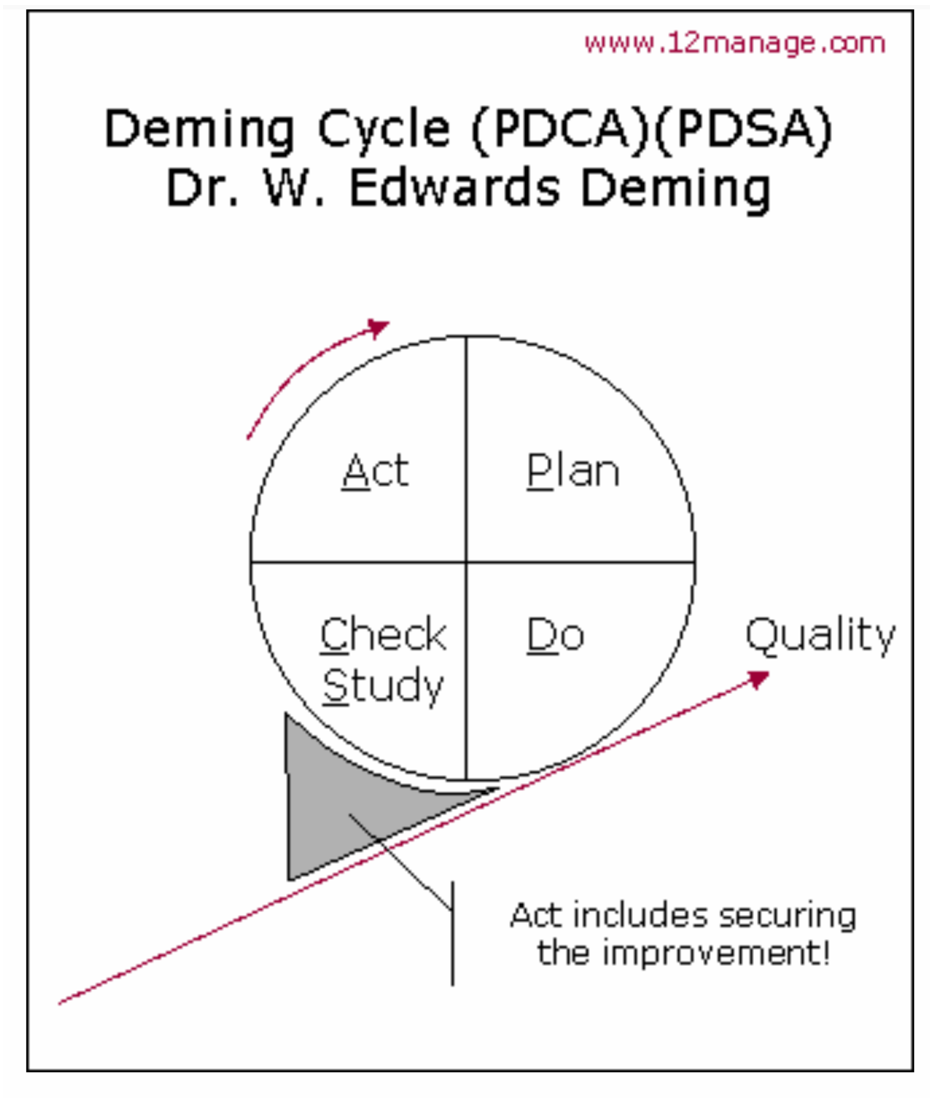

www.12manage.com
Deming Cycle (PDCA)(PDSA)
Dr. W. Edwards Deming
Act
Plan
Check
Study
Do
Quality
Act includes securing the improvement!

Figure 1.1b PDSA[6] This is agile. This is one basis for the Gilb Value cycle (Fig. 1.3)

[6] https://www.12manage.com/images/picture_deming_cycle_pdsa.gif

CHAPTER 1. 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.

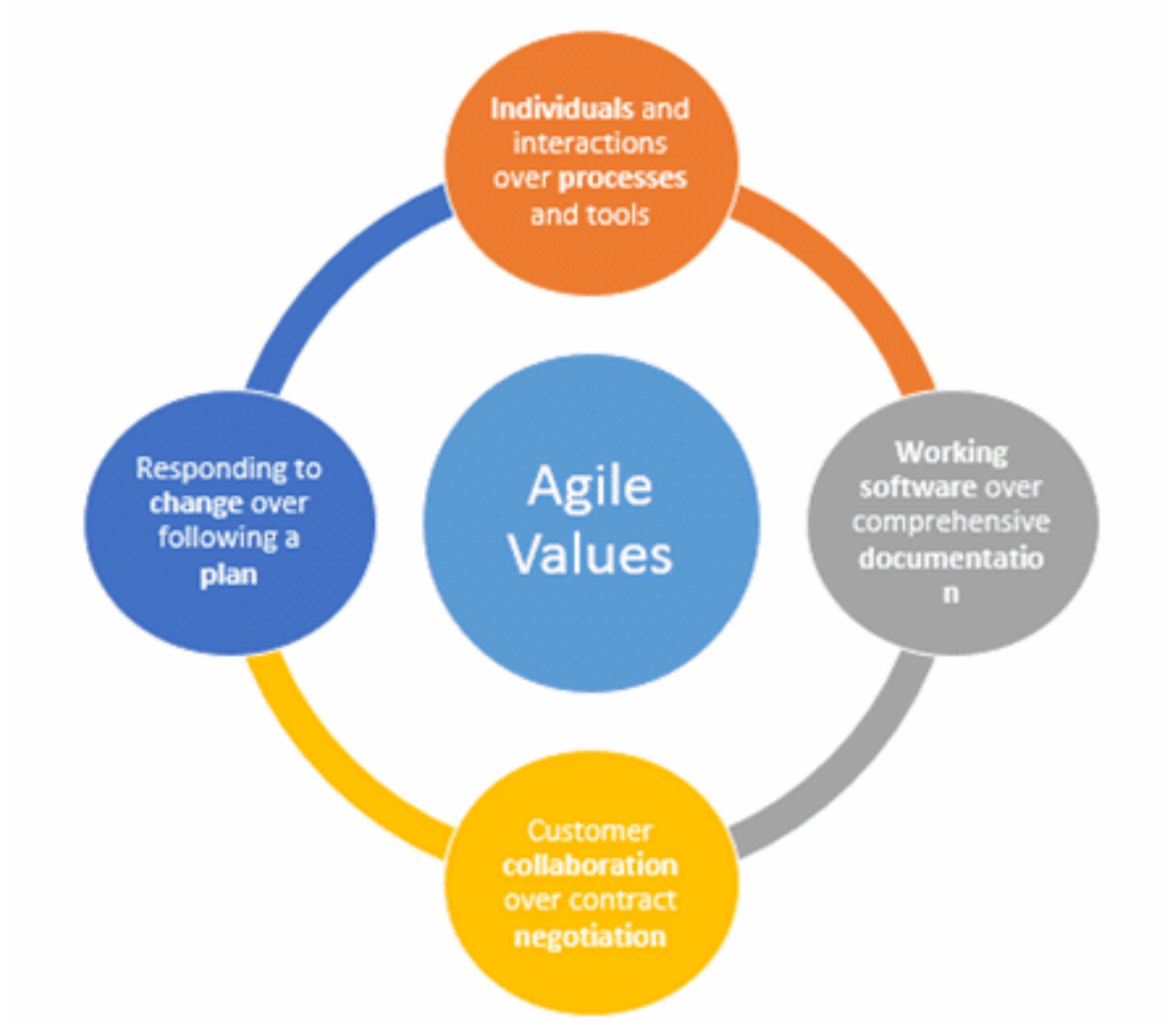

Individuals and interactions over processes and tools
Responding to change over following a plan
Agile Values
Working software over comprehensive documentatio n
Customer collaboration over contract negotiation

Figure 1.2 Four Agile Values.[7]

[7] https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/07/Agile-values.png

VALUE 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, we learn about 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].

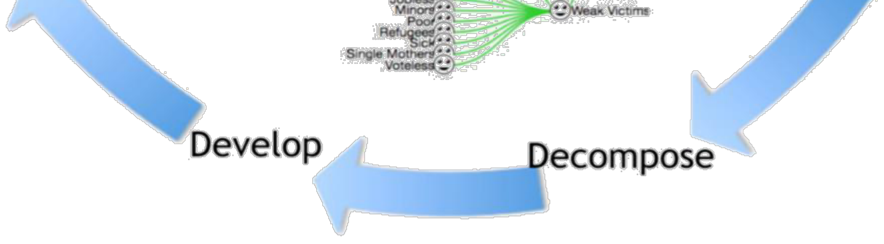


Figure 1.3 : Continuous Iterative Interaction of Stakeholders with Requirements[8]. The Gilb Value Delivery Cycle.

[8] 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)
http://concepts.gilb.com/dl918

VALUE 2. WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

I 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].

Figure 1.4 Source https://connexxo.com/wp-content/uploads/2015/01/Agile-Manifesto-value-2.png

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

That 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, The Agile Manifesto suggests none of this [32, 33].

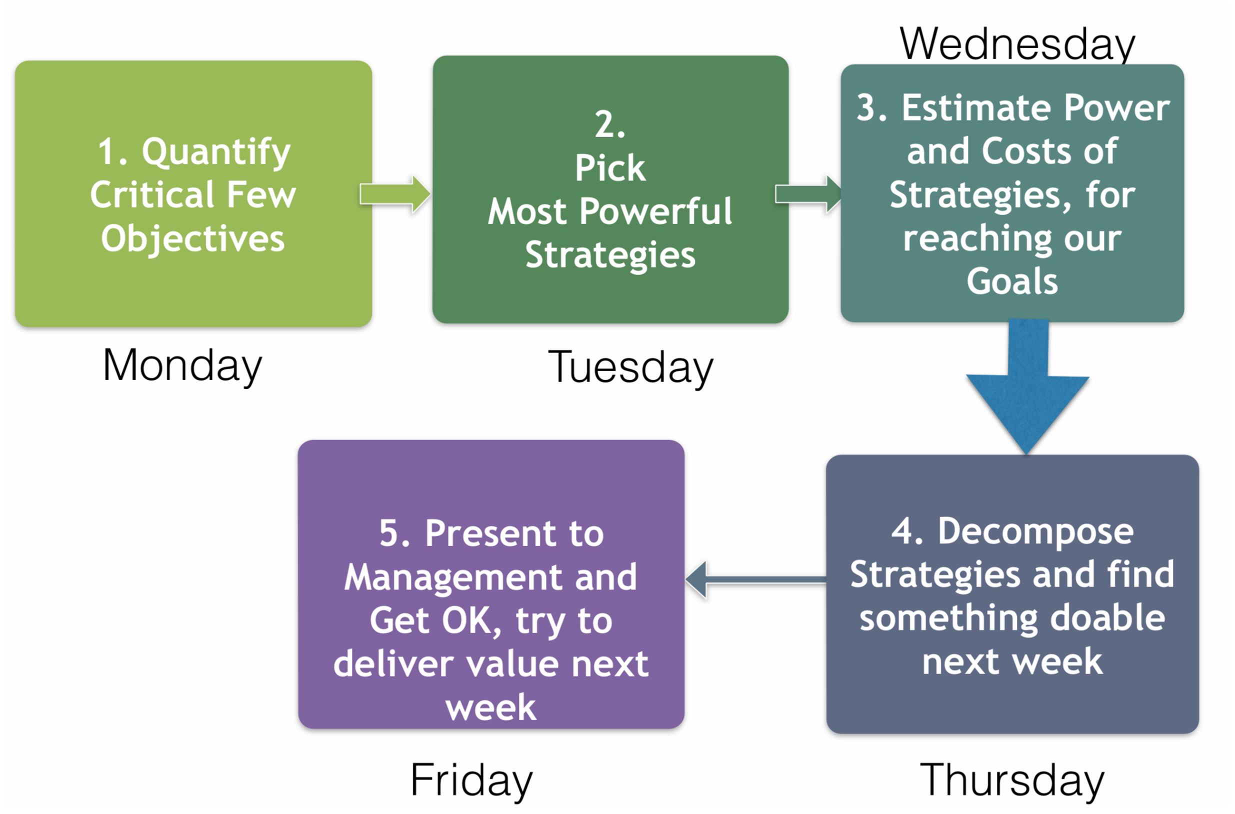

Wednesday
1. Quantify Critical Few Objectives
2. Pick Most Powerful Strategies
3. Estimate Power and Costs of Strategies, for reaching our Goals
Monday
Tuesday
5. Present to Management and Get OK, try to deliver value next week
4. Decompose Strategies and find something doable next week
Friday
Thursday

Figure 1.5 The Evo startup week process.[9]

[9] source http://concepts.gilb.com/dl851

The Agile Manifesto 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 is often 'amateur designs', pretending to be 'requirements', and they are overrated detail, suitable for coders, but not project managers, and not result designers. [5B].

I understand this Manifesto ‘value 2’ 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 understanding. They have not been agile and upgraded their methods

Good managers could have prevented narrow-minded excesses.

VALUE 3. CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION

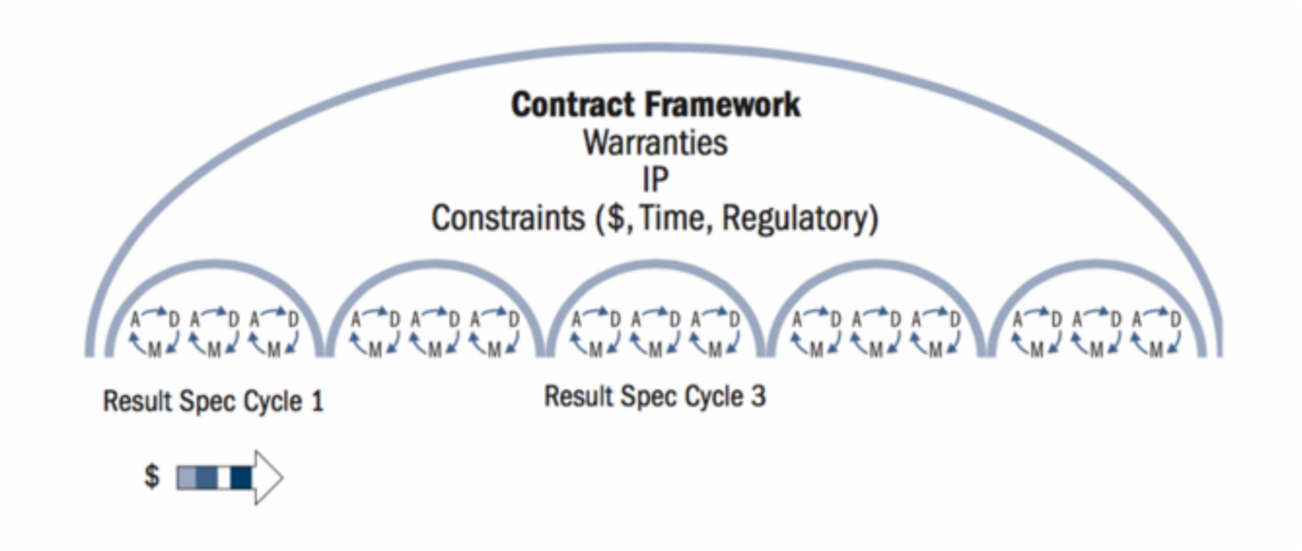
I believe this ‘Manifesto Value 3’ notion, was prompted by inadequate USA/DoD contracting practices, compounded by even worse development processes: waterfall, fixed price, and fixed dates, with contract technical design specifications, instead of contract **results**, and specifications.

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. 10

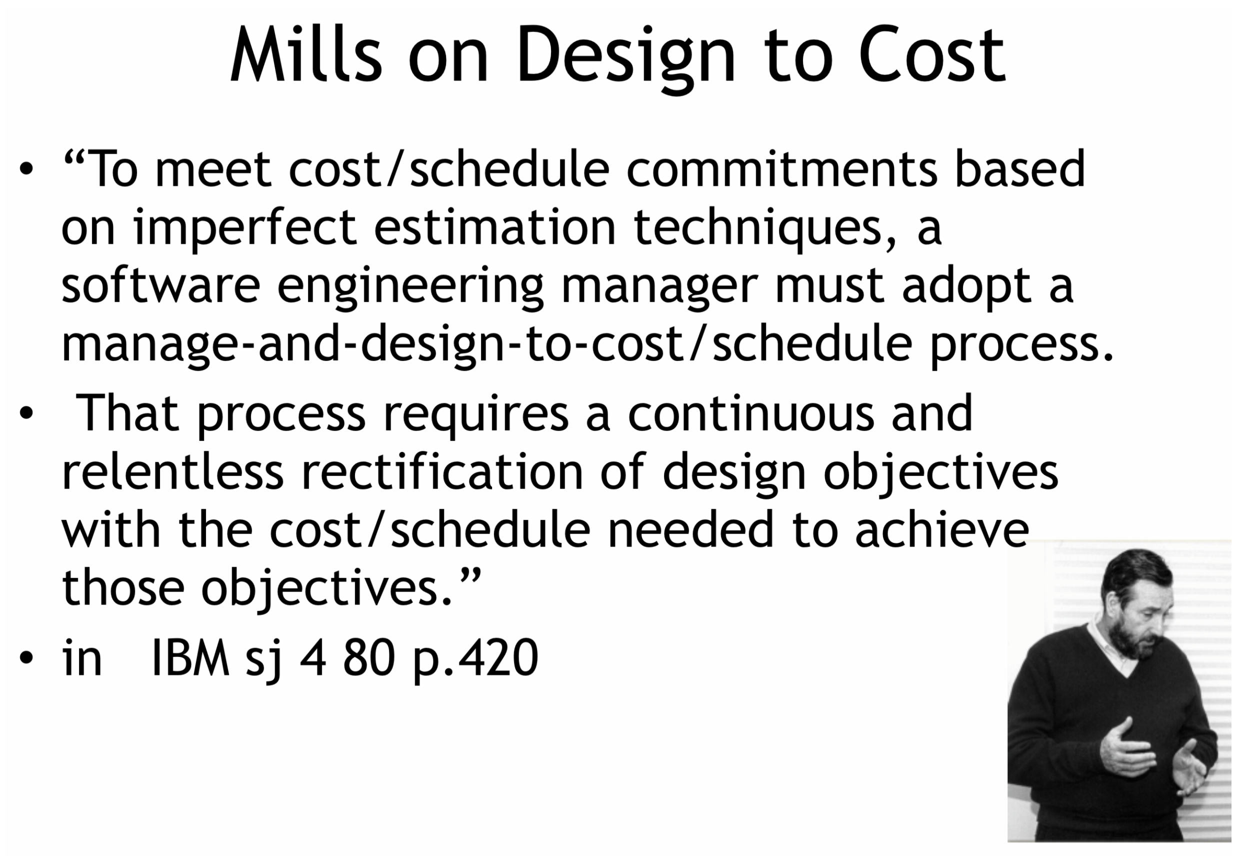
It is all worked out in 'collaboration with the customer' step by step. If step results are measurably delivered, payment is due. [39]. 'Negotiation' is done step by step, as we learn, get results, and build confidence.

Figure 1.10 An agile results-based contracting framework. Incremental value delivery, payment based on results.

[10] http://concepts.gilb.com/dl864 source, Contracting for Value slides



In earlier times, in a professional culture 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 actually got control over qualities, costs, and time, by quantification, measurement and learning, coupled with consequent step-by-step ‘re-architecture’ [Quinnan, 18].



Mills on Design to Cost

- “To meet cost/schedule commitments based on imperfect estimation techniques, a software engineering manager must adopt a manage-and-design-to-cost/schedule process.
- That process requires a continuous and relentless rectification of design objectives with the cost/schedule needed to achieve those objectives.”
- in IBM sj 4 80 p.420

Figure 1.7[11] Harlan Mills IBM Federal Systems Division, explaining the dynamic agile adjustment process to deliver projects on time and under budget to state of the art quality levels for space

[11]http://concepts.gilb.com/dl896 Slides on Mills and Quinnan dynamic design to cost process.

and military. This is identical to our Evo agile process in these principles. Robert Quinnan was the architect who actually did the design adjustments in each delivery step.

Since the Agile Manifesto has no *architecture* concept, it is incapable of doing agile architecture the way Quinnan and Mills did it at IBM (30 years earlier, and, for example, in deliveries of 43 measurable result improvement increments).

Their published results [19] were not like The Agile Manifesto (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. You read that correctly. No problems!

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' is totally absent from The Agile Manifesto.

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

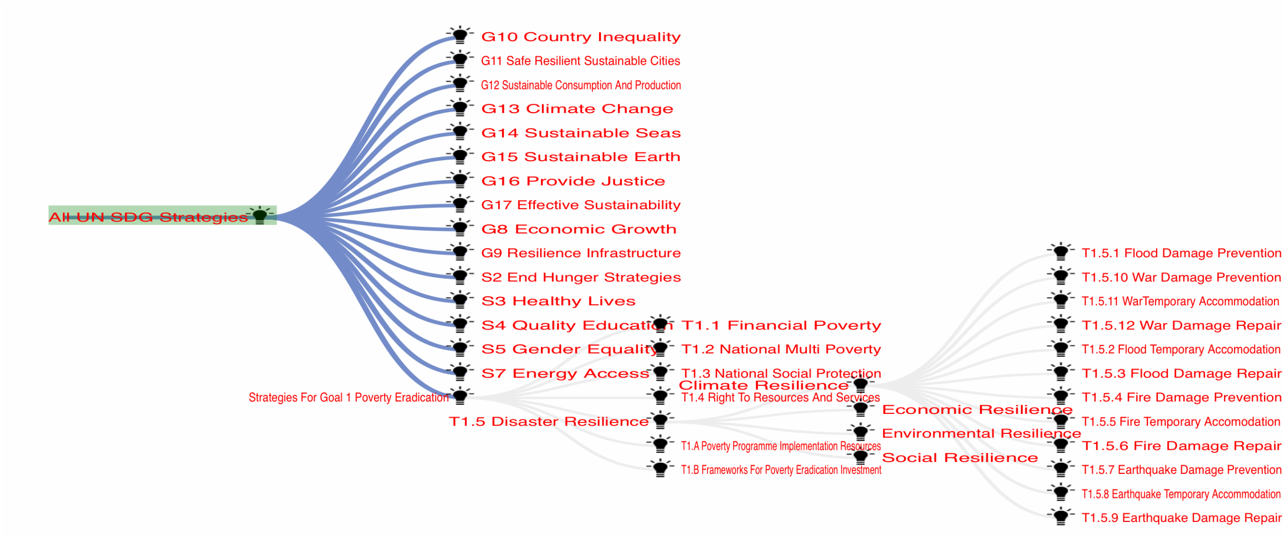
VALUE 4. 'RESPONDING TO CHANGE OVER FOLLOWING A PLAN'

Of course, I agree with the above 'platitude', as noted previously. This is the essence of 'agile' ; *responsiveness*.

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.

My preference is 'plans that focus on a few critical, quantified, top-level, long-term **value improvements**'.



~~All UN SDG Strategies~~
G10 Country Inequality
G11 Safe Resilient Sustainable Cities
G12 Sustainable Consumption And Production
G13 Climate Change
G14 Sustainable Seas
G15 Sustainable Earth
G16 Provide Justice
G17 Effective Sustainability
G8 Economic Growth
G9 Resilience Infrastructure
S2 End Hunger Strategies
S3 Healthy Lives
S4 Quality Education
S5 Gender Equality
S7 Energy Access
Strategies For Goal 1 Poverty Eradication
T1.1 Financial Poverty
T1.2 National Multi Poverty
T1.3 National Social Protection
T1.4 Right To Resources And Services
T1.5 Disaster Resilience
T1.A Poverty Programme Implementation Resources
T1.B Frameworks For Poverty Eradication Investment
Climate Resilience
Economic Resilience
Environmental Resilience
Social Resilience
T1.5.1 Flood Damage Prevention
T1.5.10 War Damage Prevention
T1.5.11 WarTemporary Accommodation
T1.5.12 War Damage Repair
T1.5.2 Flood Temporary Accomodation
T1.5.3 Flood Damage Repair
T1.5.4 Fire Damage Prevention
T1.5.5 Fire Temporary Accomodation
T1.5.6 Fire Damage Repair
T1.5.7 Earthquake Damage Prevention
T1.5.8 Earthquake Temporary Accommodation
T1.5.9 Earthquake Damage Repair

Figure 1.8 A few critical top level long term goals. In this case the 18 United Nations Sustainability Goals, with some decomposition. From my book Sustainability Planning, 2019.

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).

The Agile Manifesto 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.[12]

[12] *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

CHAPTER 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.

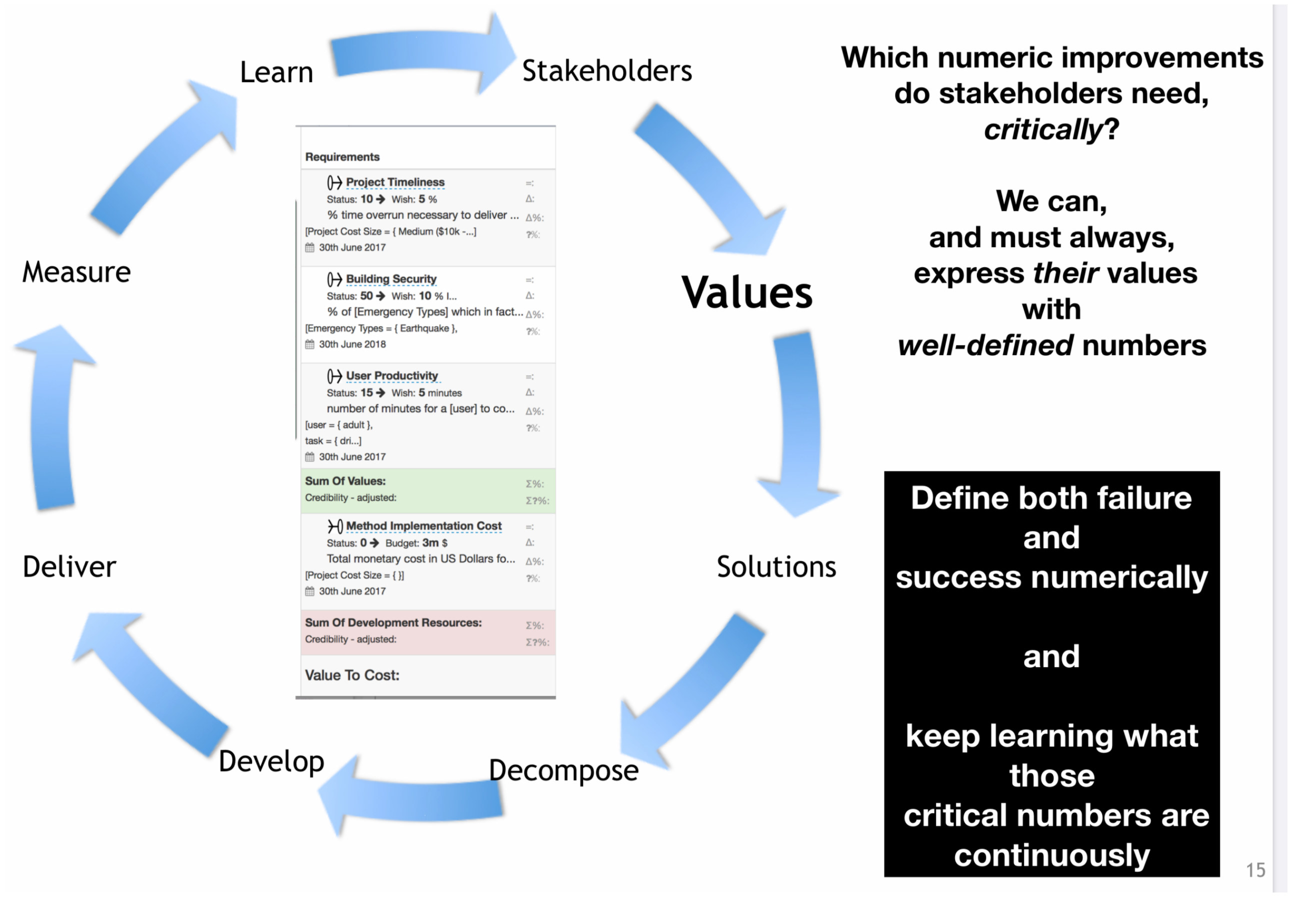
I provide here my direct comments on the principles as published.

PRINCIPLE 1: 'OUR HIGHEST PRIORITY IS TO SATISFY THE CUSTOMER THROUGH EARLY AND CONTINUOUS DELIVERY OF VALUABLE SOFTWARE'[13].

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

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

[13] this is the principle cited in the old manifesto at agilemanifesto.org



Requirements

Project Timeliness
Status: **10** → Wish: **5** %
% time overrun necessary to deliver ...
[Project Cost Size = { Medium ($10k -...]
30th June 2017
=: Δ: Δ%: ?%:

Building Security
Status: **50** → Wish: **10** % I...
% of [Emergency Types] which in fact...
[Emergency Types = { Earthquake },
30th June 2018
=: Δ: Δ%: ?%:

User Productivity
Status: **15** → Wish: **5** minutes
number of minutes for a [user] to co...
[user = { adult },
task = { dri...]
30th June 2017
=: Δ: Δ%: ?%:

Sum Of Values: Σ%:
Credibility - adjusted: Σ?%:

Method Implementation Cost
Status: **0** → Budget: **3m** $
Total monetary cost in US Dollars fo...
[Project Cost Size = { }]
30th June 2017
=: Δ: Δ%: ?%:

Sum Of Development Resources: Σ%:
Credibility - adjusted: Σ?%:

Value To Cost:

Which numeric improvements do stakeholders need, *critically*?

We can, and must always, express *their* values with *well-defined* numbers

Define both failure and success numerically

and

keep learning what those critical numbers are continuously

Figure 2.1 The highest priority is delivery of critical stakeholder values, and these values need quantification to understand, and to manage them. Conventional Agile has totally missed this essential idea. It even does not seem to recognize that there is more to the world of projects than software.[14]

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.

[14] 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)
http://concepts.gilb.com/dl918

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[15].

[15] 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

PRINCIPLE 2. ‘WELCOME CHANGING REQUIREMENTS, EVEN LATE IN DEVELOPMENT. AGILE PROCESSES HARNESS CHANGE FOR THE CUSTOMER'S COMPETITIVE ADVANTAGE’.

Our Evo and Planguage 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).[16]

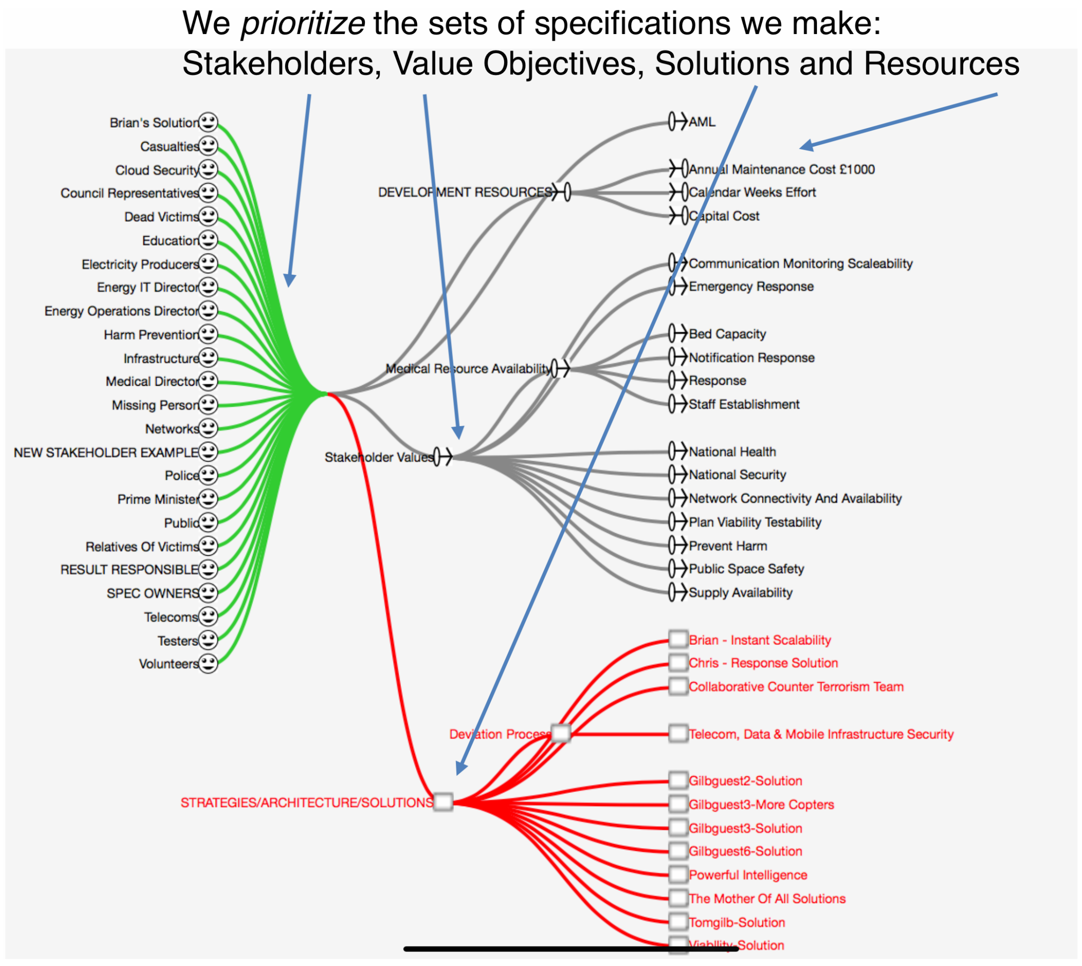


Figure 2.2. [17]. There are many planning components (stakeholders, requirements, designs) each of which has a partial influence on the priority, the chosen sequence of incremental value delivery.

[16] See my book ‘Value Requirements’, 2019, reference in Book References B ‘VR’ about 250 pages.

[17] http://concepts.gilb.com/dl908. Architecture Prioritization, BCS Talk London 17 July 2017

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 'ValPlan' Planning Tool (see www.ValPlan.net). This is far superior to common agile practices such as using a product owner.

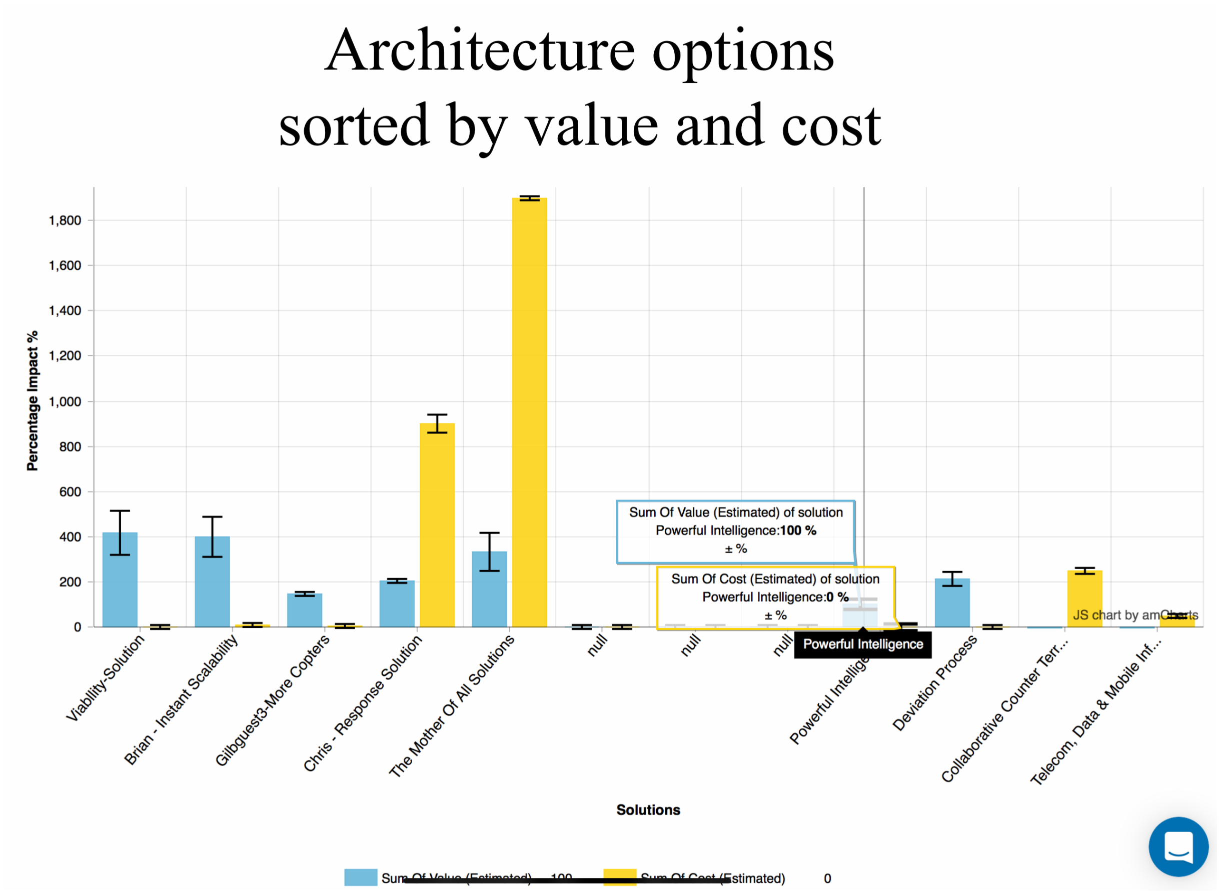


Figure 2.3[18]. The total value estimate for all chosen values, and the sum of all their costs, are used to compute a value to cost ratio. This ratio, of efficiency of a strategy, is used here to suggest a possible implementation sequence of agile sprints. This should help you get good results faster, and to avoid running out of time and money before you achieve high value results. This can be recomputed at each step, in order to consider learnings and new data.

[18] http://concepts.gilb.com/dl908. Architecture Prioritization, BCS Talk London 17 July 2017

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, AND TO PROJECT PLANS, AND CONTRACTS.

PRINCIPLE 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 limited realm of IT systems, if we never wrote code, and never delivered it. Code is not the point, except for coders.

Every organization's **real** 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 efficient 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**.

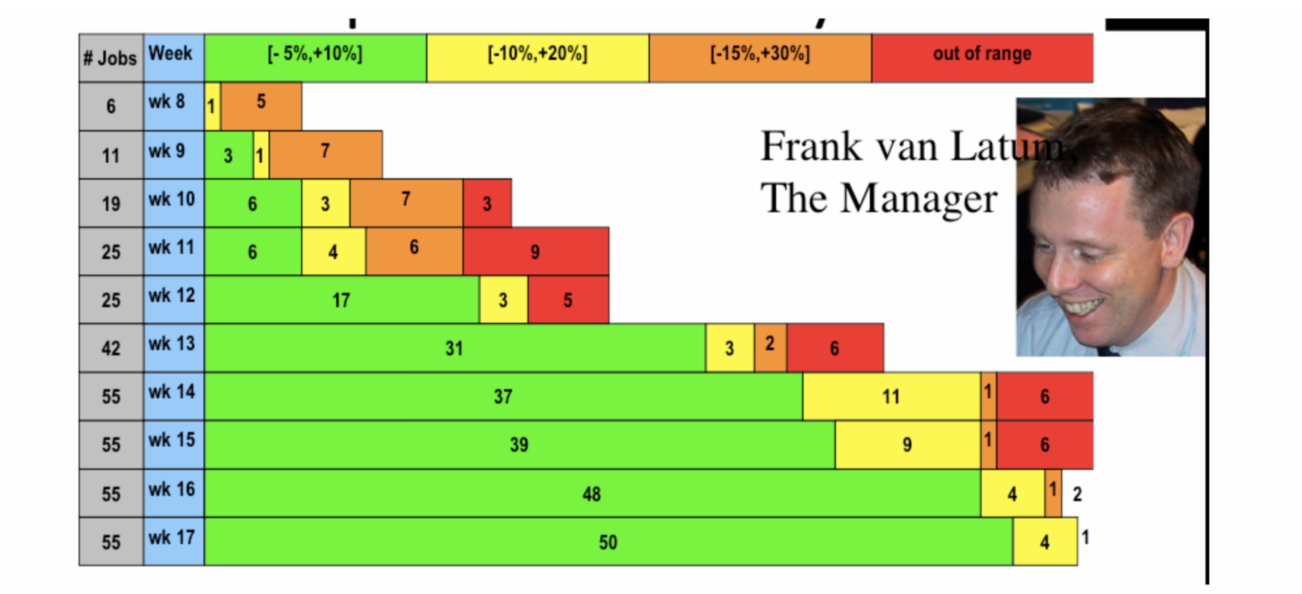


Figure 5.6 Philips Value Delivery Cycles Results. The % is the accuracy of predicting a production run of electronic circuits, before that actual run. Green is good, red is bad.

Personally, I prefer weekly, or 2%, of budget steps. [19]

Figure 2.4[20]. One of my clients, Philips, was able to break out of a 'no results' situation, by using my methods of decomposition, to deliver value early and weekly. To cumulate the long term values. Frank was the hero, the project manager who decided to go with my advice when his director did not believe it could work at all. He later won applause from the director and his team for the results he could deliver to Philips.

Keep the measurable improvements 'continuously' flowing, however you choose to do it.

I recommend *not* waiting for a couple of months, if you can do better than that.

[19] https://tinyurl.com/VPDecomposition A book chapter on decomposition

[20] https://tinyurl.com/VPDecomposition Fig 5.6 is the source of this illustration.

PRINCIPLE 4. 'BUSINESS PEOPLE AND DEVELOPERS MUST WORK TOGETHER DAILY THROUGHOUT THE PROJECT'.

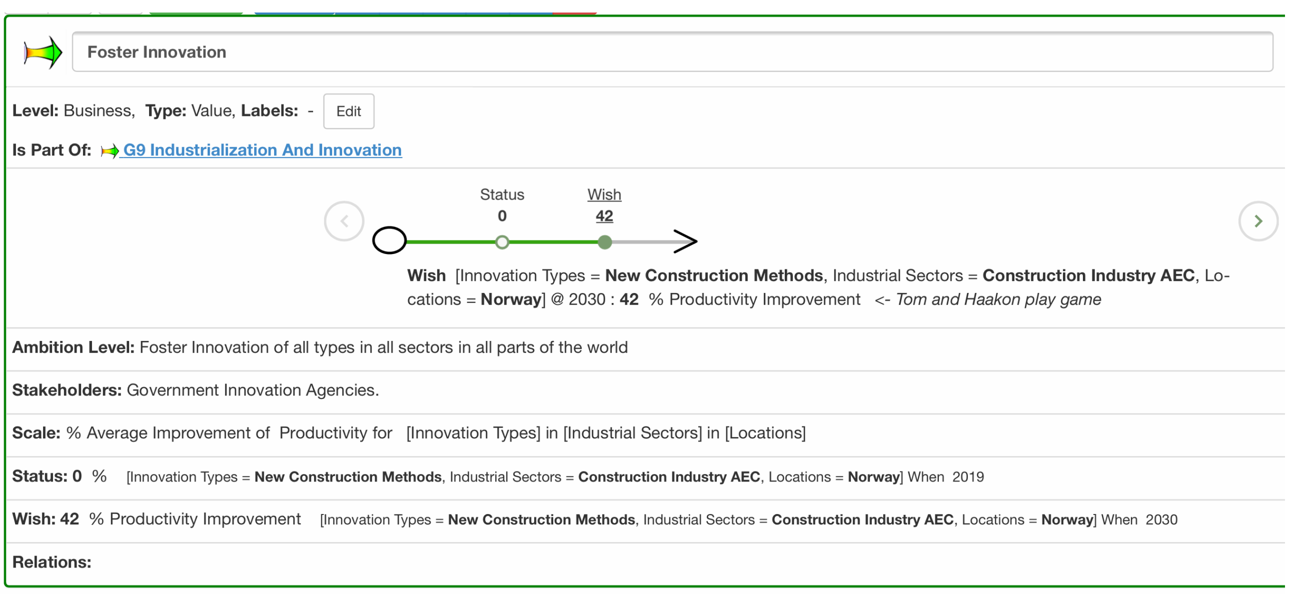
We support the *spirit* of this principle (except the unnecessary limitation of the adjective 'business').

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[21] 'stakeholder value quantification' [1] is a great tool for improving collaboration. This is because all project participants need to focus on the same value delivery. So it helps to get measurable value communication and feedback, as the main development process.

[21] 'Planguage' is the Planning Language invented by Gilb's, and defined in Competitive Engineering, and other books.


Foster Innovation

Level: Business, **Type:** Value, **Labels:** - Edit

Is Part Of: G9 Industrialization And Innovation

Wish [Innovation Types = **New Construction Methods**, Industrial Sectors = **Construction Industry AEC**, Locations = **Norway**] @ 2030 : **42** % Productivity Improvement <- *Tom and Haakon play game*

Ambition Level: Foster Innovation of all types in all sectors in all parts of the world

Stakeholders: Government Innovation Agencies.

Scale: % Average Improvement of Productivity for [Innovation Types] in [Industrial Sectors] in [Locations]

Status: 0 % [Innovation Types = **New Construction Methods**, Industrial Sectors = **Construction Industry AEC**, Locations = **Norway**] When 2019

Wish: 42 % Productivity Improvement [Innovation Types = **New Construction Methods**, Industrial Sectors = **Construction Industry AEC**, Locations = **Norway**] When 2030

Relations:

Figure 2.5[22]. An example of quantifying a value, to 'Foster Innovation'. The fuzzy source, before quantification and structuring (see the Scale) is in the 'Ambition Level' statement.

I am suggesting that this language (Planguage) for communicating, in this case for a 'value requirement', is superior to a 'face to face' explanation of the requirement.

We can communicate more exact and rich information using this Planguage format. We can update this info from anywhere at anytime. We can link and exploit this information digitally as part of the larger total picture of all requirements, designs, stakeholders. Daily developer-to-business cannot do this at all.

This daily collaboration principle is useful for small scale and temporary discussions. This particular goal in the example above is a United Nations year 2030 worldwide goal. So who is going to do the daily collaboration thing?

The written Planguage format allows any large number of people, anywhere in the world, to check out the exact requirement, today and all days in the future. They will even see the latest changes in the future. The person who last updated the specification does so once, and does not personally have to

[22] Source, Gilb, Sustainability Planning, Appendix Goal 9 (Oct 2019). This is in 'Planguage' using 'ValPlan.net' app.

‘collaborate’ daily with anybody who needs to know the requirement. That would be ridiculous.

The written Planguage format can serve as a reliable and comprehensive reference point for any presentations and discussion people want to have, both directly and asynchronously.

Oral can be useful, but it is important that oral communication does not *replace* clear, official, quality controlled, precise specifications. It only *refers* to them. It is based on good specifications.

'Stakeholders', including *critical* stakeholders, is a much broader category of critical requirements sources than '*business* people' (Principle 4). See the example stakeholder map above (Figure 1.3).

The terms 'together', 'daily', and 'work' (Principle 4) 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 ValPlan.net, we can provide a 'relevant voice' to all stakeholders, and we can provide insight into consequences of plans and decisions for all.

PRINCIPLE 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. Nice platitude[23].

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].

[23] keep in mind the Manifesto was made by a committee, seeking agreement and compromise.

[9] **How problems with Quality Function Deployment's**
(QFD's) House of Quality (HoQ) can be addressed by
applying some concepts of Impact Estimation (IE)
http://www.gilb.com/DL119

[10] **"Stakeholder Power:The Key to Project Failure or Success**
including 10 Stakeholder Principles"
http://concepts.gilb.com/dl880
2016 Paper

[11] **"Principles of Clear Communication"**, 2018 Digital book.
The Anti-Project-Failure Handbook.
€14, https://www.gilb.com/store/oJCCxtsM
If you can't afford this price for saving your expensive project, I'll send you a free copy.
But then you send me 50% of the value improvements of your next project, OK?
Project Failure Agents are trying to suppress this disinformation, as they call it, Fake News.
One clear signal is the paper publishers show no interest, and earn big on Project Tourist Books, in sectors such as Agile, Lean, Digital Transformation

[12] **CEO** Dennis Muilenburg says **Boeing** will maintain its "relentless **commitment** to make **safe** airplanes even safer." Mar 18, 2019

'Me-thinks he doth protest too much'. (Explaining the phrase I used)
https://en.wikipedia.org/wiki/The_lady_doth_protest_too_much,_methinks

"Boeing's CEO Dennis Muilenburg has been fired as the company continues to battle fallout from its 737 Max crisis"
Will Martin and Graham Rapier Dec 23, 2019, 3:41 PM
https://www.businessinsider.com/boeing-737-max-ceo-dennis-muilenburg-out-amid-plane-crisis-2019-12?r=US&IR=T

[13] **User Stories**

A. http://vimeo.com/53159408

2012 Talk Video Recording, in English

8 Minutes

Published on Dec 15, 2012

Tom Gilb discusses the dangers of User Stories at ‘Smidig 2012’ (annual conference of Agile software development in Oslo.)

B. **‘User Stories: A Skeptical View’**

www.gilb.com/DL t1

User Stories paper by Tom and Kai Gilb

In Gilbs' Mythodology Column, Agilerecord.com March 2011

[14] **Cleanroom**

Mills and Quinnan Slides

http://concepts.gilb.com/dl896

Mills, H. 1980. **The management of software engineering: part 1: principles of software engineering.** IBM Systems Journal 19, issue 4 (Dec.):414-420.

Direct Copy

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

Includes Mills, O’Niell, Linger, Dyer, Quinnan pg. 466 on

Library header

http://trace.tennessee.edu/utk_harlan/5/

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

http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1017&context=utk_harlan

(ACTUAL DOWNLOAD) IEEE

Mills Generally
http://trace.tennessee.edu/utk_harlan

[15] A. **'Agile Contracting for Results The Next Level of Agile Project Management':** Gilb's Mythodology Column Agilerecord August 2013
concepts.gilb.com/dl581

B. '**Agile Contracting for Results'**
Smidig 2014 Oslo Ten minute talk slides
http://www.gilb.com/dl832

CHAPTER 4 WHAT IS 'AGILE AS IT SHOULD BE'?

THE REAL AGILE

People who are interested in agile, are interested in managing projects and processes better.

I wonder if we can agree that the most important thing about projects and processes, is their cost-effectiveness? NOT exactly 'which methods' we choose to use, in order to get that cost-effectiveness.[52]

If any forms of agile increase our cost-effectiveness, they can be considered a *good* thing. If agile reduces cost-effectiveness, it is *not* a good thing.

There are many, good and bad, definitions of Agile. There are many different agile practices, some more cost-effective than others.

There is no standard definition of agile. There is no standard agile process. There is very little research which could tell us which variations of agile are most cost-effective. There is some research, but very little in comparison with wide-spread use

[52] www.agnosticagile.org is an open minded way of thinking about agile variations.

of various agile variations, including associated ideas like user stories. People ‘just do it’ and record and report very little about the cost-effectiveness.[53]

Ideas and methods of being agile have existed in many cultures, and for a long time. [54]

There is nothing new about the basic idea. Adapt or die, is not new. Indeed it may be baked into all creatures in nature.

LET’S DEFINE ‘COST-EFFECTIVENESS’

Effectiveness is the degree of delivery of a set of values, that are planned and desired.

Costs, are the short-term (acquisition, development) costs of delivering the planned values; as well as the longer-term lifetime costs, of continuing to deliver those values (maintenance, support). The costs are the consumption of any limited resources, such as money, time, talent, labor, credibility, image.

Cost-Effectiveness is a very **multi-dimensional** set of **quantifiable** variables. It is also *not* a standard set of values and costs. It completely depends on the stakeholders values, and *their* resources; and these are highly *varied*, and continuously *variable* during projects and processes.

One probable consequence of this, is that there is no one form of agile, which is best for all possible value-objectives, and cost-constraints.

---

[53] I do not consider measurements of ‘velocity of delivering user stories’ as very interesting measures. We need to know more about the business and market value-effectiveness, and the long-term and short-term costs.

[54] Chapter 15 in (1988) Principles of Software Engineering management
gilb.com/dl561 “Deeper Perspectives on Evolutionary Delivery”
plus a page extra of quotations from Agile Gurus crediting Evo as inspiration for them and being first.

Another consequence, of agile cost-effectiveness prioritization, is that anybody should be able to measure work in progress, on the fly, and see which types of agile actually improve numeric delivery of planned value objectives, and/or reduce resources needed to produce the values and maintain them.

You already know that all *cost* ideas are inherently measurable. But I can tell you that all *value* ideas are also quantifiable and measurable. Most people do not know that, and are not taught how to do it. But that is your problem of hopefully temporary ignorance.[55]

The fact that you can put adjectives in front of any value (very sustainable, highly secure) is a strong indicator of their variability, and their potential quantifiability.

The easiest way to see exactly how to quantify values is to Google them, with 'metrics' after their name ('Usability metrics', 'cooperation metrics').

[55] most references, particularly my books, show in detail how to quantify all values. See Competitive Engineering (2005), Value Planning (2015-2017), and Value Requirements (2019), Sustainability Planning (2019)

WHAT ARE THE CONSEQUENCES OF 'AGILE COST-EFFECTIVENESS' AS A TOOL?

This insight, that agile ideas can and should be measured, in terms of how agile practices impact your own real continuous cost-effectiveness, or 'efficiency' (Values/ Costs) means that you now have a practical tool for deciding if any given form of agile is good for you or not, no matter what popular opinion, fashion, or slick salespeople tell you.

If a given form of agile allows you to get the highest desired value-levels, at the lowest costs, then it is good for you. Fashion has nothing to do with it. It has to work well for *you,* and for your real current projects and processes.

Here is a list of some interesting consequences of this cost-effectiveness or values/ costs way of evaluating 'agile' methods:

1. You cannot evaluate any agile (or non-agile) methods unless you have quantified all ***your*** critical value objectives, in advance, to enable you to match methods with your values/costs profile. Most people do not.
2. You cannot evaluate agile methods unless you continuously measure the degree you have reached ***your*** stated value levels (goals, objectives). Most people don't.
3. The same applies to planning and measuring all types of critical resources. You have to quantify ***your*** budgets, and measure ***your*** consumption continuously, in order to decide if agile methods are doing your resource consumption any good.
4. Any instance (book, paper, video, talk, consultancy) which claims that their agile methods are good, needs to at least initially give some *evidence*, case studies, of comparably (before their agile medicine was taken) better values and costs. Most of these 'methods people' offer *none* of this. At most some claim of 'velocity' of

delivery of code (not value)[56]. You would do well to avoid these irrelevant salespeople.[57]

5. If a particular agile method, to my great surprise, can actually give evidence, for some instances, of its ability to deliver multiple critical values, at lower multiple costs, then the next question is: is this the right prescription for my own situation and culture? Will it work for me?
    1. Try it, but measure it.
    2. One beautiful thing about agile is that you can get useful measurable feedback early and frequently, even while continuously tuning agile methods, to suit your environment.

[56] One salesman of Scrum likes the velocity measure of his methods. But high velocity of delivering undefined critical values, is not really useful. He agrees in public to this point.

[57] to demonstrate that this evidence is not impossible, I refer the reader to the many cases in my references and books. One simple example in particular my client Confirmit. Green Week Confirmit Case
The Green Week: Reducing Technical Debt by Engineering, http://www.gilb.com/dl575
May 2013, and http://www.gilb.com/DL32, and http://www.gilb.com/dl152. At the other end of the scale see Intel reports, like Simmons and Terzakis www.dropbox.com/sh/cs9hke3uvgg4gp3/AACadHeI95lZpHzVqGKXSXDra?dl=0, http://concepts.gilb.com/dl892

WHAT ABOUT THE CORE AGILE IDEA: 'ADAPTABILITY' TO CHANGES, AND TO NEW INSIGHTS?

Agility is all about the short-term ability to change things, so that the long-term value objectives are still met, within constraints (resources, legality, etc.).

This means that we need real-time measurement (this week, today) of how things are going. We need to sense quickly when they are not going well, and - as quickly - change something, to try to make it go better.

If you do not have quantified values and costs, and do not measure them, you cannot manage cost-effectiveness, as discussed just above. You cannot measure and react quickly, in order to improve results.

You need to measure results, but you also need to measure *frequently*. If you discover something is wrong within a week, then you lose less time, than if you wait a year or more to discover it. It is not at all unlike cars, cycles, and rockets. It is not unlike wars, and business product competitiveness.

Most all agile methods seem to agree that we need to measure progress, or at least get feedback, in a week or two[58]. So I need not argue that case here.

The point I am making, above, is that our agile reaction cycle, has to be agile, analyze-react-improve, for ***relevant*** things.

Values and costs are those relevant things. Delivery of 'working software' is NOT. 'Sprints' of producing software, are NOT.

[58] at the extreme, in an internet product, over 50 experiments per day might be measured and corrected, *Lean Startup*.

CHAPTER 5

PARALLEL[59] AGILE VALUE DELIVERY

SOME DEFINITIONS FROM 'PLANGUAGE' [REF. S] AND 'COMPETITIVE ENGINEERING' [F] BOOK.

" Parallel Development **Concept *363 February 25, 2003**

Any development of more than one Evo step, that takes place at the same time.

Notes:

1. Within Evo, given the need to keep up the frequency of deliveries, and the length of time it can take to develop and produce certain steps, 'parallel development' is a potential solution.
2. While it means *potential* loss of the advantage of being able to use feedback experience-data, it does mean an advantage in utilization of elapsed time.
3. With intelligent step planning the loss of useful feedback can be minimized.

"'Partly simultaneous' development of closely related product variants.
The purpose is to reduce the TBSP (Time Between Successive Products)."
[JANDOUREK96]

Type [Parallel Development *363]: Process.

(Listed on page 456 of CE book, 2005 [F].)

Parallelity: **Concept *104. August 3, 2001**

Having a number of parallel Evolutionary cycles.
These can be any type of Evo cycles frontroom or backroom (concurrent engineering).

Related Concept:
• Parallel Development, *363

Domain: Project Management, Evo.Step Planning, Parallelity.

(end of quote from my Planguage Concept Glossary).

[59] The 2020 publication **'Parallel Agile'** [R] has reminded me to articulate and document my own published work from early 2000 in Planguage and before [N. O, P, Q] from 1976 and 1977, which the authors do not seem to know about.

Our method can be described as a form of **'Concurrent Engineering'.**

THE COMPARISON TO THE PARALLEL DEVELOPMENT AS WRITTEN IN THE 2020 'PARALLEL AGILE' BOOK.

As expected in the conventional agile culture, discussed extensively here, the focus in the 'Parallel Agile' book is not systems, or engineering, or value: it is coding. In the 'Parallel Agile' book, the Parallel Agile ideas seem to be only about writing code in parallel. As the reader is by now fully aware, we consider this dangerously narrow. Our 'parallel agile value delivery' is *not* focussed on 'code development parallelity'. But it is one of many possibilities.

In fact when solving the systems engineering problem of 'delivering values within constraints', there may be many situations where *no code* at all is called for. So, our Value Agile Parallelity is about two or more Evolutionary Value Delivery steps, always aimed at delivering stakeholder value, even if there is no code, and no IT system at all.

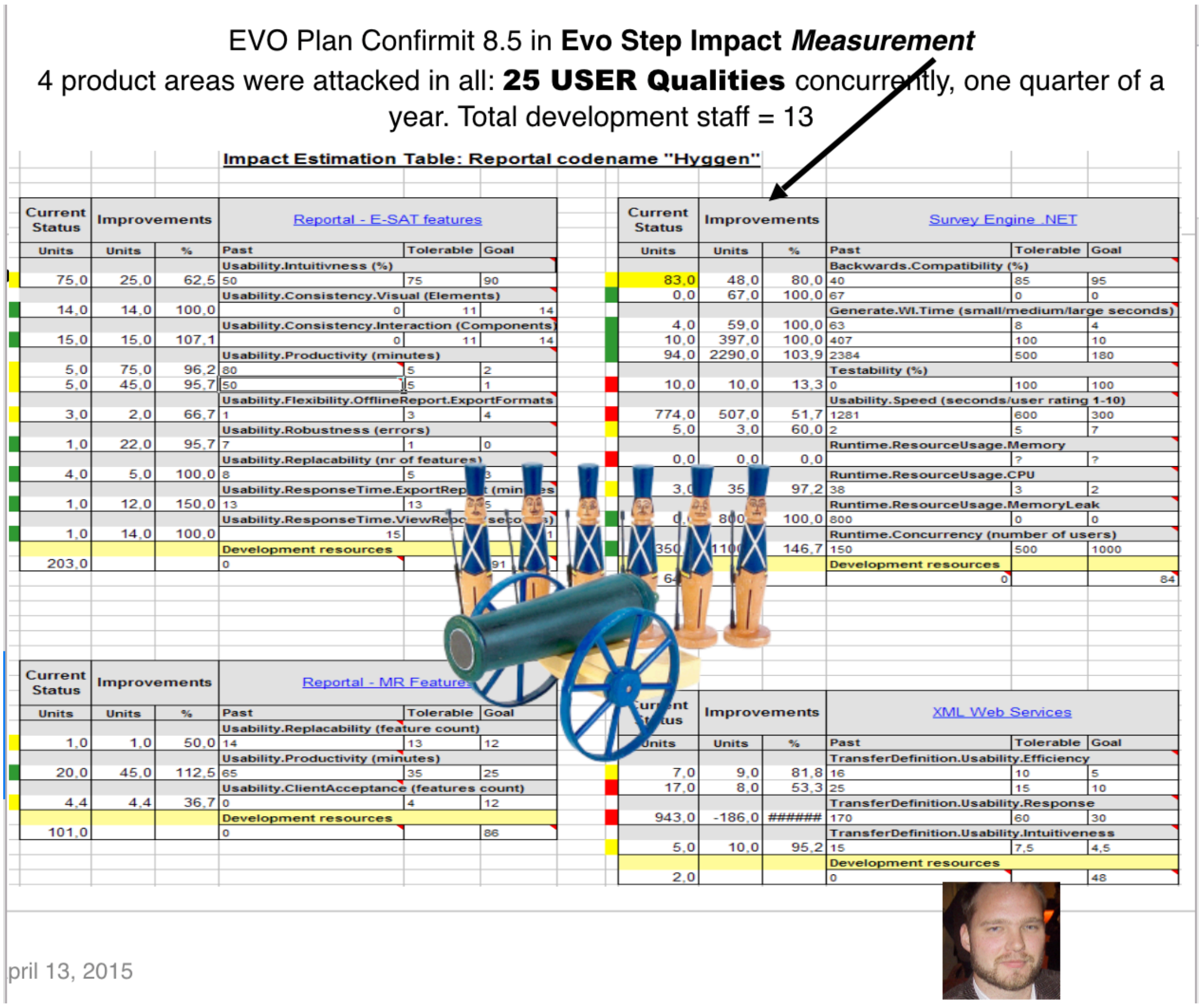
The book 'Parallel Agile' [R], does not seem to acknowledge the systems level concepts of *value* and *quality*. So we are talking about two different worlds.

DISTINCT SOFTWARE

In my earlier work on 'Distinct Software' [N, O, P, Q], which started with my Datamation paper "**Parallel Programming**' [U, 1974]. I am exclusively looking at code. It is my other later work on Evo and Planguage which shifts the focus to value delivery by parallel development. But I think Parallel Value Delivery using Value Agile is far more interesting, so I will document some more about it below.

PRACTICAL CASE STUDY 2003 CONFIRMIT

Here is a real example of one of our best clients, who developed their main product (Confirmit, a product polling system) using 4 parallel teams of about 4 members. They did so on a weekly basis for 12 Evo delivery cycles, and using the result they quarterly delivered vastly increased value and quality to their customers. They actually destroyed competition, and took over their competitors (like Pulsetrain, UK) because their product got so good so fast. Part of the speed was in the parallel development. Partly also because they set high quality-and-value improvement numeric targets, and let the teams decide, based on numeric feedback, what *really* delivered the value levels needed.



EVO Plan Confirmit 8.5 in **Evo Step Impact *Measurement***

4 product areas were attacked in all: **25 USER Qualities** concurrently, one quarter of a year. Total development staff = 13

Impact Estimation Table: Reportal codename "Hyggen"

	Current Status	Improvements		Reportal - E-SAT features		
	Units	Units	%	Past	Tolerable	Goal
				Usability.Intuitivness (%)		
🟨	75,0	25,0	62,5	50	75	90
				Usability.Consistency.Visual (Elements)		
🟩	14,0	14,0	100,0	0	11	14
				Usability.Consistency.Interaction (Components)		
🟩	15,0	15,0	107,1	0	11	14
				Usability.Productivity (minutes)		
🟨	5,0	75,0	96,2	80	5	2
🟨	5,0	45,0	95,7	50	5	1
				Usability.Flexibility.OfflineReport.ExportFormats		
🟨	3,0	2,0	66,7	1	3	4
				Usability.Robustness (errors)		
🟩	1,0	22,0	95,7	7	1	0
				Usability.Replacability (nr of features)		
🟩	4,0	5,0	100,0	8	5	3
				Usability.ResponseTime.ExportRep[illegible]t (min[illegible]es		
🟩	1,0	12,0	150,0	13	13	5
				Usability.ResponseTime.ViewRep[illegible] (seco[illegible]s)		
🟩	1,0	14,0	100,0	15	[illegible]	1
				Development resources		
	203,0			0		91

	Current Status	Improvements		Survey Engine .NET		
	Units	Units	%	Past	Tolerable	Goal
				Backwards.Compatibility (%)		
🟨	83,0	48,0	80,0	40	85	95
🟩	0,0	67,0	100,0	67	0	0
				Generate.WI.Time (small/medium/large seconds)		
🟩	4,0	59,0	100,0	63	8	4
🟩	10,0	397,0	100,0	407	100	10
🟩	94,0	2290,0	103,9	2384	500	180
				Testability (%)		
🟥	10,0	10,0	13,3	0	100	100
				Usability.Speed (seconds/user rating 1-10)		
🟥	774,0	507,0	51,7	1281	600	300
🟨	5,0	3,0	60,0	2	5	7
				Runtime.ResourceUsage.Memory		
🟥	0,0	0,0	0,0		?	?
				Runtime.ResourceUsage.CPU		
🟨	3,0	35	97,2	38	3	2
				Runtime.ResourceUsage.MemoryLeak		
🟩	0,[illegible]	800	100,0	800	0	0
				Runtime.Concurrency (number of users)		
🟩	350	110[illegible]	146,7	150	500	1000
				Development resources		
	64			0		84

	Current Status	Improvements		Reportal - MR Features		
	Units	Units	%	Past	Tolerable	Goal
				Usability.Replacability (feature count)		
🟨	1,0	1,0	50,0	14	13	12
				Usability.Productivity (minutes)		
🟩	20,0	45,0	112,5	65	35	25
				Usability.ClientAcceptance (features count)		
🟨	4,4	4,4	36,7	0	4	12
				Development resources		
	101,0			0		86

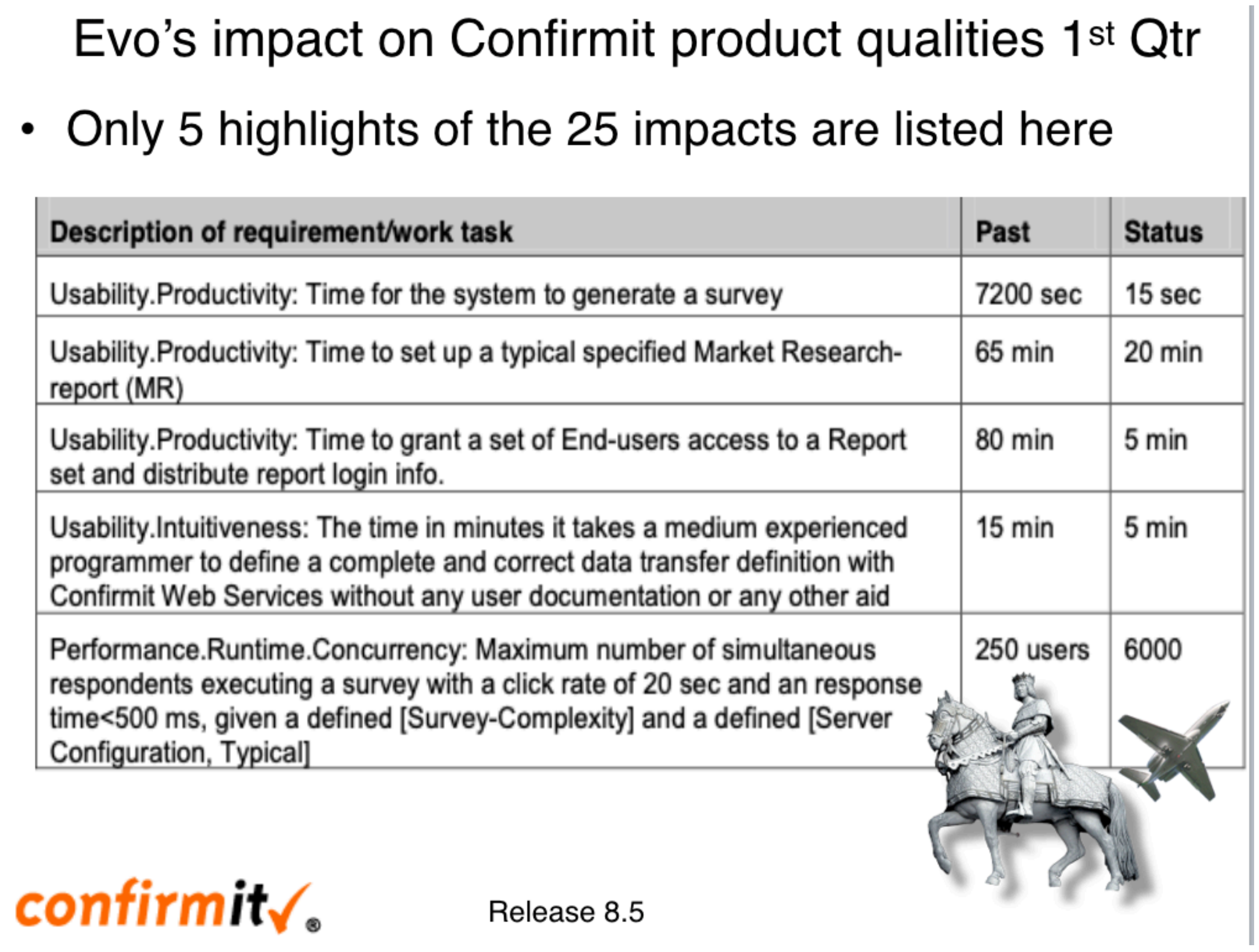
	Current Status	Improvements		XML Web Services		
	Units	Units	%	Past	Tolerable	Goal
				TransferDefinition.Usability.Efficiency		
🟨	7,0	9,0	81,8	16	10	5
🟥	17,0	8,0	53,3	25	15	10
				TransferDefinition.Usability.Response		
🟥	943,0	-186,0	######	170	60	30
				TransferDefinition.Usability.Intuitiveness		
🟨	5,0	10,0	95,2	15	7,5	4,5
				Development resources		
	2,0			0		48

pril 13, 2015

Figure 5.1 Confirmit, Oslo, Norway in the 9th of 12 weekly Evo value delivery cycles. Trond Johansen (photo) led the effort.

Comments on the figure.

1. Each parallel team has accepted, for deadline in 12 weeks delivery, to their world market of customers, a set of numeric value and quality Goals. (see 'Goal' column)
2. The % column is the cumulative value delivered to date. 100% means all of the Goal target. The are in 9 of 12 weeks when this snapshot was taken.
3. The teams (13 developers and 3 testers, in 4 parallel teams) are free (empowered) to choose, on each Evo value delivery step, to work on the value *they* want to work on.
4. The team is also empowered to find, estimate value of, and measure their own designs, for reaching the Goal levels.



Evo's impact on Confirmit product qualities 1st Qtr

- Only 5 highlights of the 25 impacts are listed here

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000

confirmit

Release 8.5

Figure 5.2 Best 5 results from Parallel Development. Far exceeding Goals.

In general, in all quarterly (12 weekly-sprint cycles of value delivery) parallel developments, all Goal levels were met or exceeded. I attribute this to the effectiveness of 'Dynamic Design to Value', as described just above. The teams were free to focus on weak levels, to stop working on achieved goal levels, and could quickly see, and measure, whether their designs were working or not.

These guys were largely writing code, but it was code to implement designs with clear primary focus, and responsibility, for delivering numeric stakeholder value.

THE BACKROOM AND FRONTROOM PARALLEL VALUE DEVELOPMENT

I have in detail elsewhere described a parallel development process, which I call Backroom/Frontroom.[E, F].

In short a Backroom development takes longer time than the usual Evo delivery cycle. This could be because it cannot be decomposed into smaller deliveries (see decomposition below), or because there are delays in a supply chain. When the backroom task is ready for stakeholder value delivery, it is made available as a Frontroom option. The Frontroom is where we encounter the stakeholders, and actually try to deliver measurable value, on a regular cumulative short cycle of delivery.

So, there is clearly a special type of parallel development going on in Backroom/ Frontroom. Not only can any number of short cycle Frontroom developments occur, as with Confirmit (above) but any number of parallel Backroom developments can be in play at the same time.

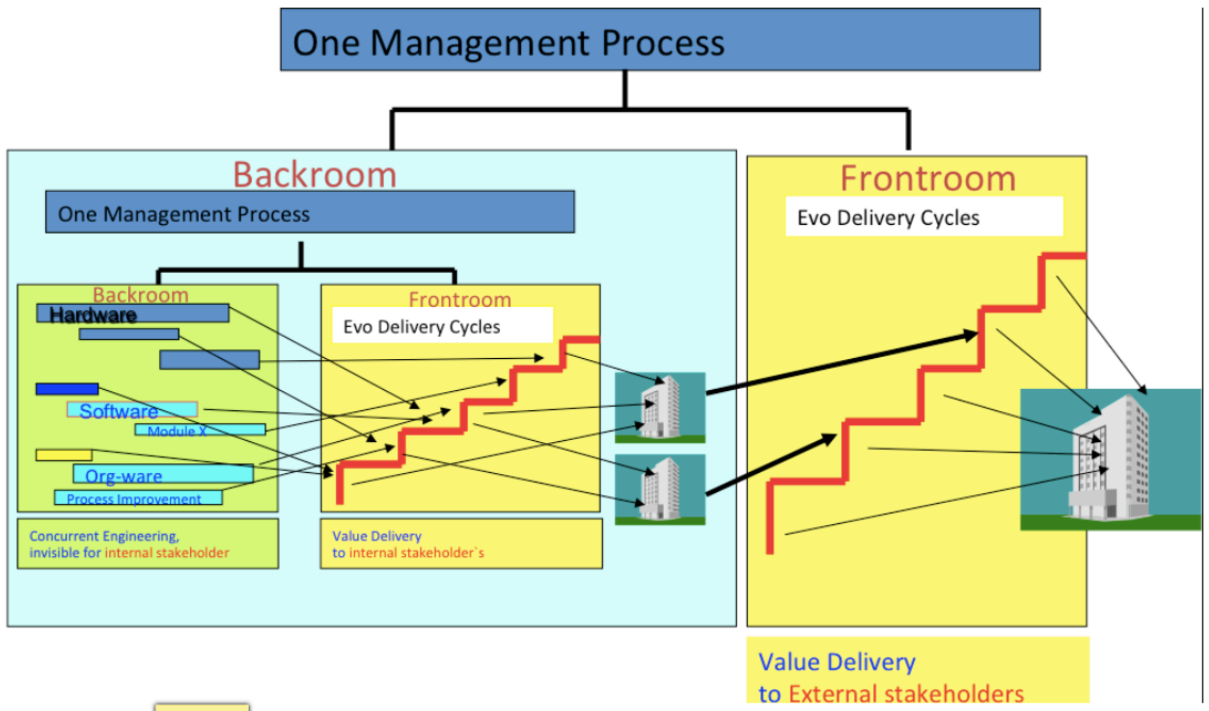


Figure 5.1 Philips Corporation in Holland use of our Backroom/Frontroom concepts, at 2 levels. See also Fig. 6.3 for Backroom visualization.

Figure 5.3 (5.1 is ref. From Value Planning [E] book). Multiple parallel levels of Backroom and Frontroom were practiced by our client Philips.

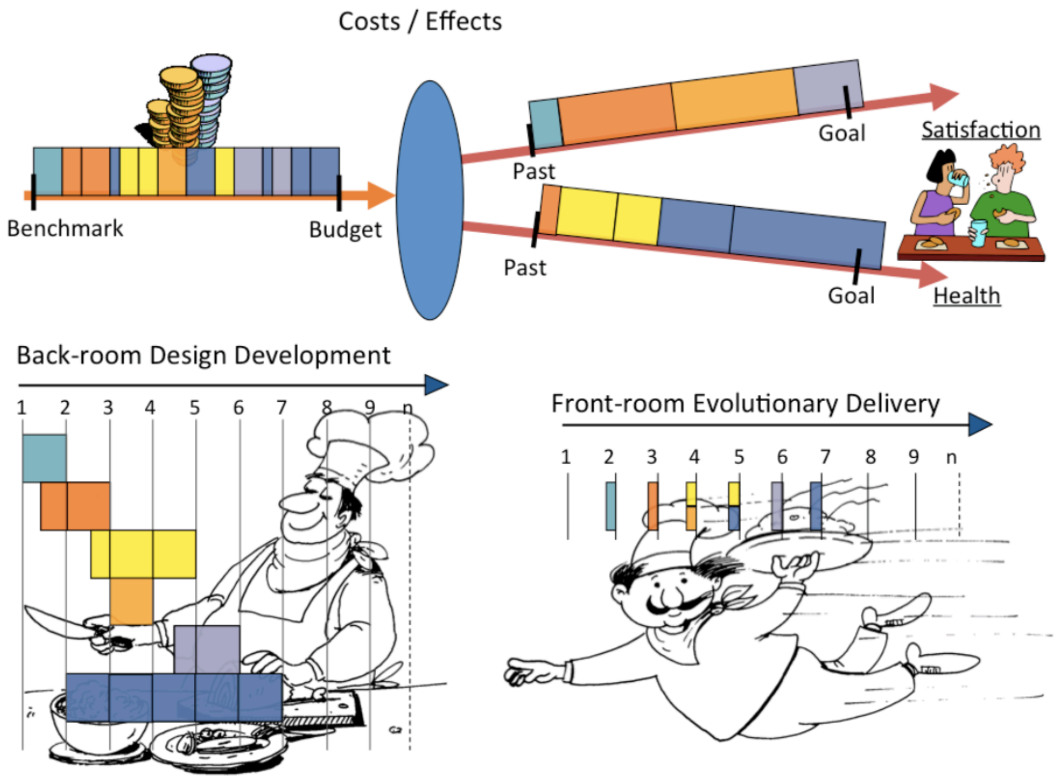


Figure 6.3 One visualization of the prioritization problem.
On the one hand, we are investing up front in the back room, consuming limited budget, and not immediately getting any value back. Is this a wise investment? A necessary evil? But we can track incremental value delivery from Past to Goal, and see the value build up. We need to figure out the lowest-cost set of sub-strategies to reach our Goal levels. Reality is of course at least ten times more complicated than this simple model.

Figure 5.4 (6.3 is from [E]) Here multiple Backroom parallel tasks build up, to the point they can be chosen, for a future Frontroom value delivery Evo step.

PARALLEL VALUE REQUIREMENT SPECIFICATION

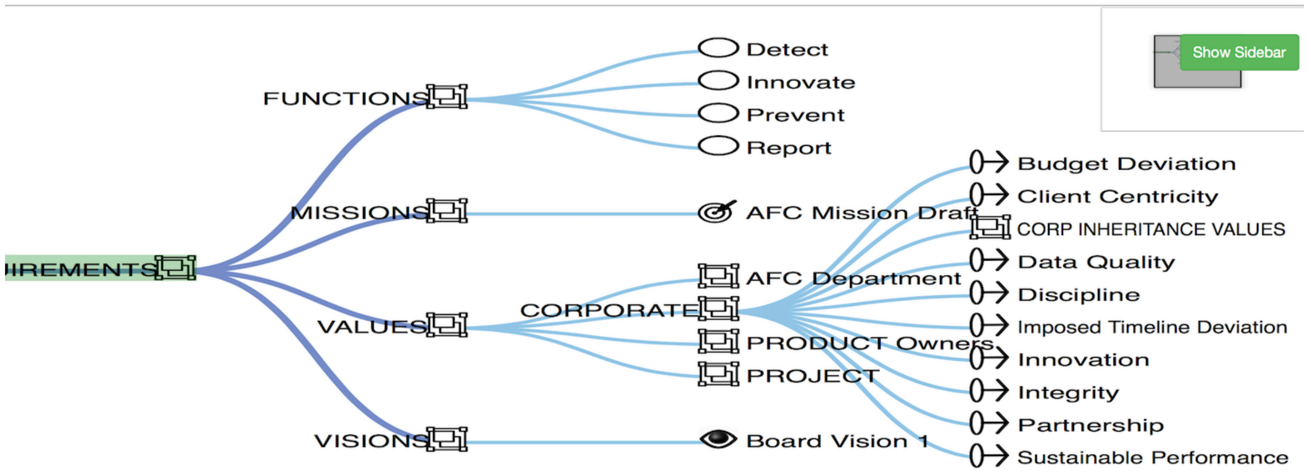


Figure 5.5. By defining a parallel set of quantified values, (Budget Deviation, ….Sustainable Performance (as detailed in Figure 2.10 above, Value Requirements [B] and my other books)

By defining our values quantitatively, and also with 'advanced (Scale Parameter, Specification Parameters) requirement specification structures', we have set up a checklist of potentially parallel targets to develop towards. Because the Goal levels are quantified, we can, as at Confirmit above, measure progress incrementally, and know when to stop (when Goal is reached) and when to put in effort, when the Goal is not yet reached. If you do not quantify Goals, and get incremental feedback, then your parallel efforts must be inefficient as you are flying blind.

Value *quantification* is arguably one more tool for encouraging and managing parallel development.

STRATEGY DECOMPOSITION, FOR INDEPENDENT IMPLEMENTATION

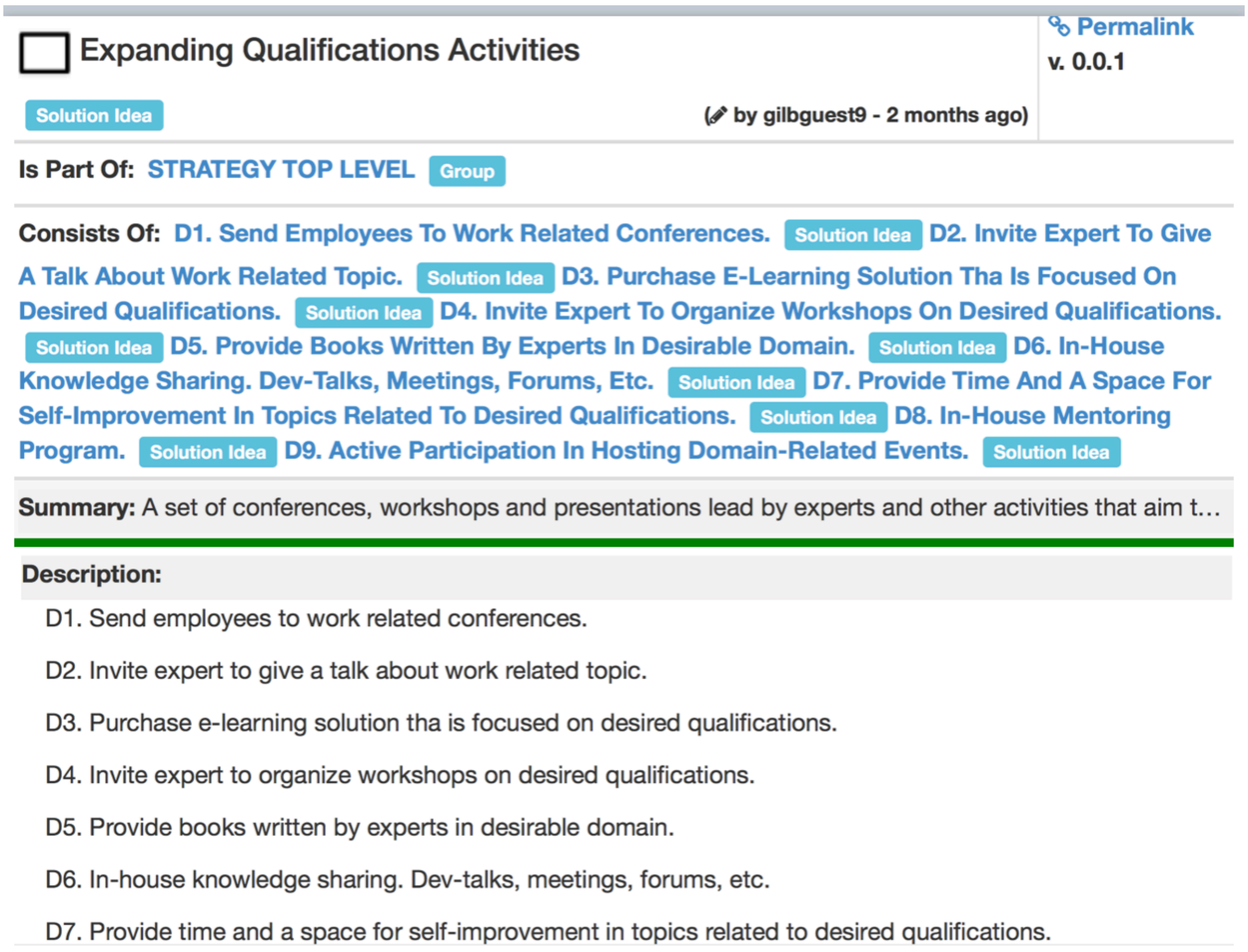
If you permit decomposition of epics into stories *without* specifically demanding that the stories (smaller deliverable units) can be implemented independently of the others you have decomposed too, then you will end up with *dependencies*. You cannot be as parallel as you otherwise would be.

When I teach my clients to decompose big strategies (architectures, designs, etc), I insist on the following rules.[60]

1. Each decomposed element must be implementable without any of the others being it's dependencies.
2. Each decomposed element must *alone* deliver measurable value.

[60] The Value Planning book, Decomposition Chapter
"Ch 5 Decomposition by Value" in my Dropbox:
https://tinyurl.com/VPDecomposition or see [C, and E]

People do not follow these rules intuitively, or from other practices. You have to spell it out, and check that they really understand. These (no dependencies, measurable value) are radically new paradigms for most professionals.


☐ Expanding Qualifications Activities

Solution Idea

(✎ by gilbguest9 - 2 months ago)

Permalink

v. 0.0.1

Is Part Of: STRATEGY TOP LEVEL Group

Consists Of: D1. Send Employees To Work Related Conferences. Solution Idea D2. Invite Expert To Give A Talk About Work Related Topic. Solution Idea D3. Purchase E-Learning Solution Tha Is Focused On Desired Qualifications. Solution Idea D4. Invite Expert To Organize Workshops On Desired Qualifications. Solution Idea D5. Provide Books Written By Experts In Desirable Domain. Solution Idea D6. In-House Knowledge Sharing. Dev-Talks, Meetings, Forums, Etc. Solution Idea D7. Provide Time And A Space For Self-Improvement In Topics Related To Desired Qualifications. Solution Idea D8. In-House Mentoring Program. Solution Idea D9. Active Participation In Hosting Domain-Related Events. Solution Idea

Summary: A set of conferences, workshops and presentations lead by experts and other activities that aim t…

Description:

D1. Send employees to work related conferences.

D2. Invite expert to give a talk about work related topic.

D3. Purchase e-learning solution tha is focused on desired qualifications.

D4. Invite expert to organize workshops on desired qualifications.

D5. Provide books written by experts in desirable domain.

D6. In-house knowledge sharing. Dev-talks, meetings, forums, etc.

D7. Provide time and a space for self-improvement in topics related to desired qualifications.

Figure 5.6 A digital decomposition of 'Expanding Qualifications Activities' big strategy, into 7 value-delivery independent sub-strategies. Warsaw Polish Export Planning case 2017.

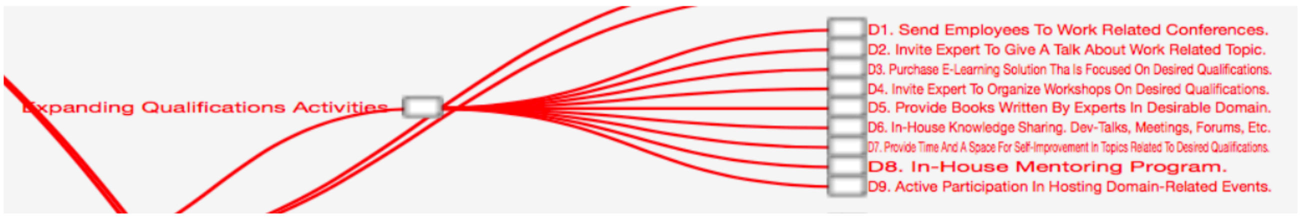
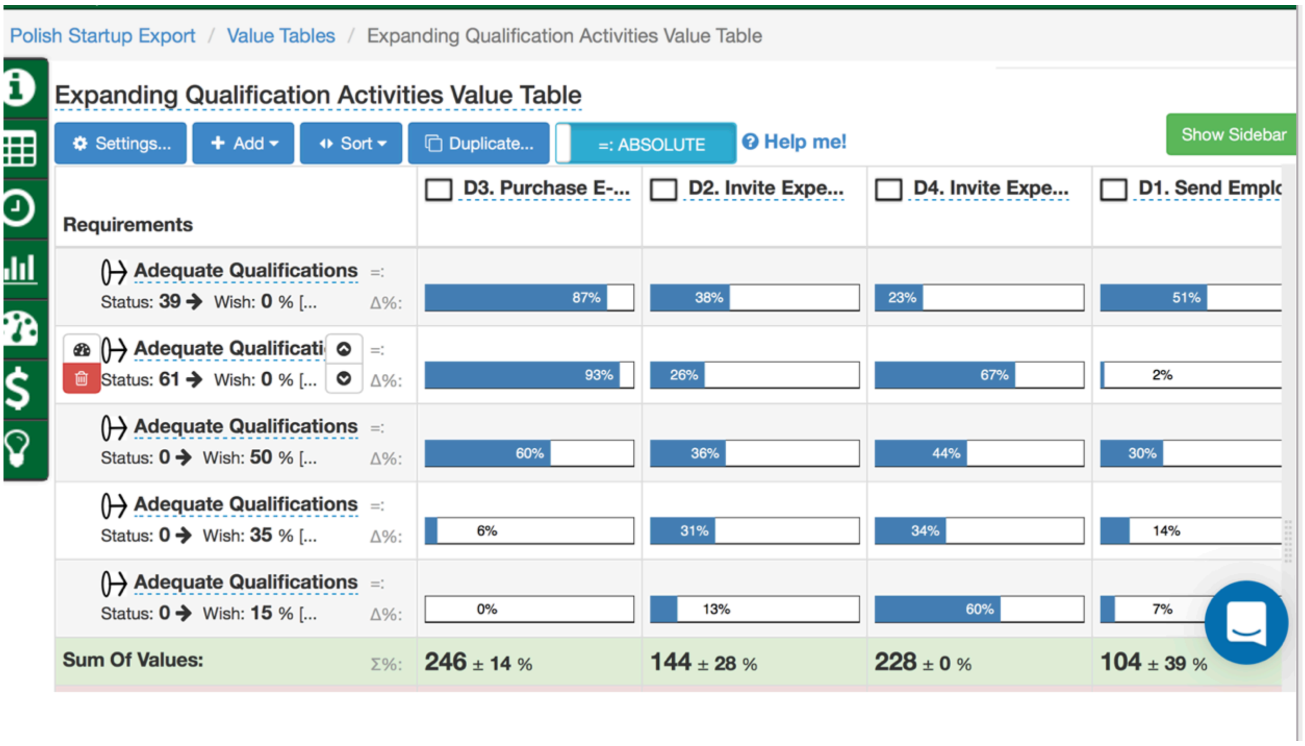

xpanding Qualifications Activities
D1. Send Employees To Work Related Conferences.
D2. Invite Expert To Give A Talk About Work Related Topic.
D3. Purchase E-Learning Solution Tha Is Focused On Desired Qualifications.
D4. Invite Expert To Organize Workshops On Desired Qualifications.
D5. Provide Books Written By Experts In Desirable Domain.
D6. In-House Knowledge Sharing. Dev-Talks, Meetings, Forums, Etc.
D7. Provide Time And A Space For Self-Improvement In Topics Related To Desired Qualifications.
D8. In-House Mentoring Program.
D9. Active Participation In Hosting Domain-Related Events.

Figure 5.7 A graphical representation of the strategy decomposition into potential parallel value delivery tasks, or Evo value delivery steps.

The above is a good example of delivering value with strategies that are not code!

USING VALUE TABLES TO SELECT PARALLEL DEVELOPMENTS



Polish Startup Export / Value Tables / Expanding Qualification Activities Value Table

Expanding Qualification Activities Value Table

Requirements	D3. Purchase E-...	D2. Invite Expe...	D4. Invite Expe...	D1. Send Emplc
Adequate Qualifications =: Status: 39 → Wish: 0 % [... Δ%:	87%	38%	23%	51%
Adequate Qualificati =: Status: 61 → Wish: 0 % [... Δ%:	93%	26%	67%	2%
Adequate Qualifications =: Status: 0 → Wish: 50 % [... Δ%:	60%	36%	44%	30%
Adequate Qualifications =: Status: 0 → Wish: 35 % [... Δ%:	6%	31%	34%	14%
Adequate Qualifications =: Status: 0 → Wish: 15 % [... Δ%:	0%	13%	60%	7%
Sum Of Values: Σ%:	246 ± 14 %	144 ± 28 %	228 ± 0 %	104 ± 39 %

Figure 5.8 From the Polish Export example above. The decompositions (D2, D3, etc) are estimated for their possible impact on a variety of target levels. The 'Sum of Values' indicates the potential overall for several requireemnts. So if I had development capacity for just two of these sub-strategies, I could go for D3 (246 Sum Value) and D4 (228 Sum Value) as promising choices for the next round of parallel development.

My point here is that selection, or prioritization, of parallel development tasks can be a rational logical transparent, automatically calculated decision (forget 'product owner').

A RESULT CYCLE [S]: HERE IS ANOTHER DETAILED CONCEPT DEFINITION FROM 2003 (PLANGUAGE CONCEPT GLOSSARY)

Result Cycle **Concept *122. January 3, 2003**

Within Evo, a result cycle is an entire Evo step cycle aimed at delivering a result that moves towards satisfying the overall requirements.

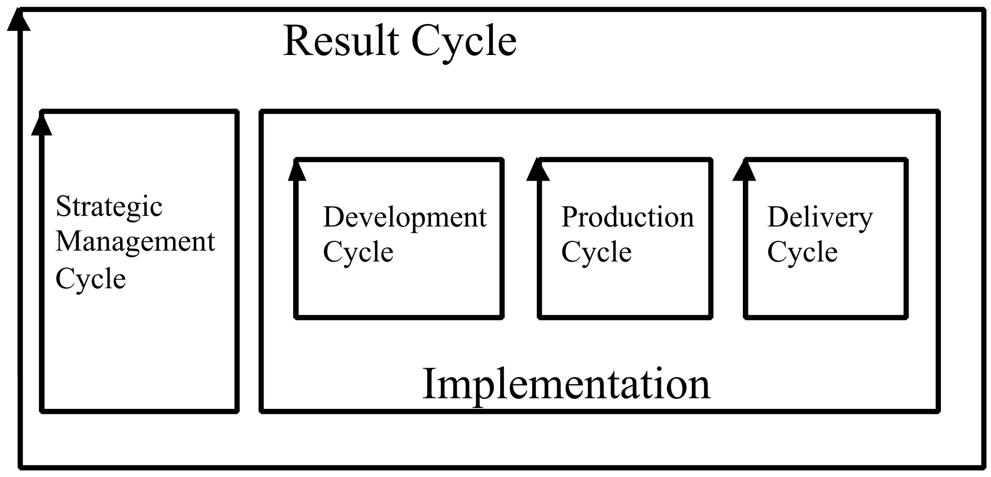
Notes:

1. A result cycle is a cycle consisting of 'Plan Do Study Act' activities.

2. It can involve any kind of system change, small or large: for example, factory production modification, software program alteration, organizational restructure, new software product development and design of new businesses.

3. A project using Evo will execute numerous result cycles. The emphasis is on 'contact with reality' and *using consequent feedback to adjust*.

Figure G.x: Diagram shows the component cycles of a Result Cycle.



4. A result cycle consists of:
• a strategic management cycle
• a development cycle (any development required or acquisition of the deliverables)
• a production cycle (any product integration or manufacture and distribution required)
• a delivery cycle (the actual delivery of the deliverable to the user)

5. Result cycles, for different steps, can be executed serially **and in parallel.** The reason for this is the variable times taken for implementation (specifically development and production cycles) and the Evo requirement to achieve a reasonably short delivery cycle frequency. For example, the average delivery cycle frequency could be stipulated to be weekly or monthly, but a specific result might take six months from initiation to actual result delivery, due to such factors as research cycles, order cycles, construction cycles and approval processes. These processes would normally be sought to be done *in parallel* with other Evo cycle activities, so that the Evo management team and their stakeholders would still experience some result delivery within the stipulated delivery cycle time.

6. The development and production cycles are termed ‘backroom’ activities and the delivery cycle is termed a ‘frontroom’ activity. One useful analogy is to think of the way in which a restaurant delivers to its customers. Ideally, delivery to the table is independent of food and drink preparation times!

Synonyms [Result Cycle *122]:
• Result Production Cycle
• Step Cycle

Type: Process.

VALUE AGILE 'REFERENCES'

A. My booklet **'Sustainability Planning'** (September 2019) which looks at United Nations Goals in depth. Short term https://www.dropbox.com/sh/gc65fds9h0gv3cm/AABJvW4fwAnqVn25bPtY9bmia?dl=0, and longer term see my website www.Gilb.com. Free forever.

B. 'VR'. **Value Requirements**, by Tom Gilb, 2019, See www.Gilb.com.[61]

https://www.dropbox.com/sh/0g4bfcjc3hi8uv7/AADGW6S6rVuFpDBTA8f_BR5Ta?dl=0

C. 'VD' **Value Design**, by Tom Gilb, 2019, See www.Gilb.com

https://www.dropbox.com/s/ldrofca89sfwzur/Value%20Design%20MASTER%20B2607%20V1408.pdf?dl=0

D. 'VM' Value Management, by Tom Gilb, 2019, See www.Gilb.com

https://www.dropbox.com/s/7utbgxzcmahfj0c/Value%20Management%20MASTER%20B070819%20V160819.2252.pdf?dl=0

E. **Value Planning** (2017) Link to book: https://www.gilb.com/store/2W2zCX6z.

F. 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.

G. Gilb, **Life Design**, 2018, https://www.gilb.com/store/kCBGcG6L.

H. Gilb, **100 Practical Planning Principles**, 2018, https://www.gilb.com/store/4vRbzX6X

I. Gilb, **Technoscopes**, 2018, https://www.gilb.com/store/Pd4tqL8s

J. Gilb, **Clear Communication**, 2018, https://www.gilb.com/store/oJCCxtsM

K. Gilb. **Innovative Creativity**, 2018, https://www.gilb.com/store/QMMQhn2g

L. Gilb, **Principles of Software Engineering Management**, 1988

[61] booklets B C D (VR, VD, VM) are written quite recently and are not as of 25 Dev. 2019 for sale or on my website. They are on my Dropbox, and can be shared on request, to 'Tom at Gilb com' until they are available later at Gilb.com.

M. Gilb, **Value Agile,** 2019-2020 (this book). See gilb.com (sales) and temporarily see https://www.dropbox.com/sh/o2g7ib3z2g2uzfw/AAAypXlN0yA2WS4obwlDzZR3a?dl=0

N. T. Gilb, **Data Engineering** was published by Studentlitteratur AB in 1976
ISBN 91-44-12621-2 , page 50-56 discussed "Parallel Programming and Dual Code" *this* book was simply not sold in US market

O. Dual code was discussed extensively in **Software Metrics** (Studentlitteratur 1976 ISBN 91-44-12631-X and the next year in the US edition. This was widespread in US and coined the term Software Metrics!

P. Infotech State of the Art Report , ISBN 8553-9380-7
1977, Software reliability has an article by T. Gilb
'**Distinct Software: A redundancy technology for reliable Software'** pp117-135, cited in full in [U]

Q. THERE ARE 4 REFERENCES TO DISTINCT SOFTWARE IN THE INDEX of T. Gilb, Principles of Software Engineering Management. 1993 Forword by B Boehm. In print.

R. **Parallel Agile – faster delivery, fewer defects, lower cost**

Hardcover – January 4, 2020

by Doug Rosenberg (Author), Barry Boehm (Author), & 4 more

S. **Planguage Full Glossary** (a subset is published in Competitive Engineering 2005 [F]) http://www.gilb.com/dl830. This is the website glossary referred to in the CE book page 456.

T. Lee, '**Fault Tolerance**' book cites many distinct software sources from me and others including this earliest one, [U]. Google: tom gilb Datamation dual programming.

U. [op cit T] T. Gilb, "**Parallel Programming**" , Datamation 20 (10), pp.160-161, October 1974. "This short note was one of the first to advocate the use of dual programming teams, albeit for testing and run-time error detection alone". (Lee, [T].

V. **Confirmit Case. (of parallel development)**
Product Development Using Evo and Planguage, with the Confirmit. Case Study

V1. https://youtu.be/vH4dSqsUv3I

Videoed 28 NOV.2018, Released 15 Dec. 2018

At Meetup in Warsaw, on Product Development

1 Hour presentation of Confirmit Case

Followed by 1 Hour of Q&A

Texted in English.

Highly varied quality of sound and texting.

But I still think it was a good presentation of my methods.

V2 http://concepts.gilb.com/dl33
CONFIRMIT SLIDES MADE BY KAI
————————————————————————
V3. http://www.gilb.com/DL32
PAPER 'FROM WATERFALL TO… BY TROND JOHANSEN AND TOM GILB

W.

VALUE AGILE EDITING LOG

22-210919: editing cleaning up the 2 basic papers Manifesto, Saboteur.

230919 added chapter 4, what is agile , cost effectiveness, values, costs.

181019 added footnote agnosticagile.org p 52. Reviewing book to see if i need to add things. +POSEM ch 15 ref footnote. Detailed edit of chapter 4. I total edited chapter on Manifesto values and principles.

191019 edited ch 3 and 4

Next step add illustrations.

20 oct 2019 started adding illustrations FIGURES

22oct. added illustrations and text to end of chapter 3

25 Dec 2019: (V251219) A complete read of the text and many (about 40) small edits for clarity and correcting typos). Added dropbox links to new 2019 books, in 'Value Agile' book references.

i am not sure if i can add meaningful illustration to the small chapter 4. must think, 251219 still thinking. Tempting to keep it simple text)

14 Feb 2020 added chapter 5 on concurrent delivery, and a lot of consequent references.

Very last page of this book.