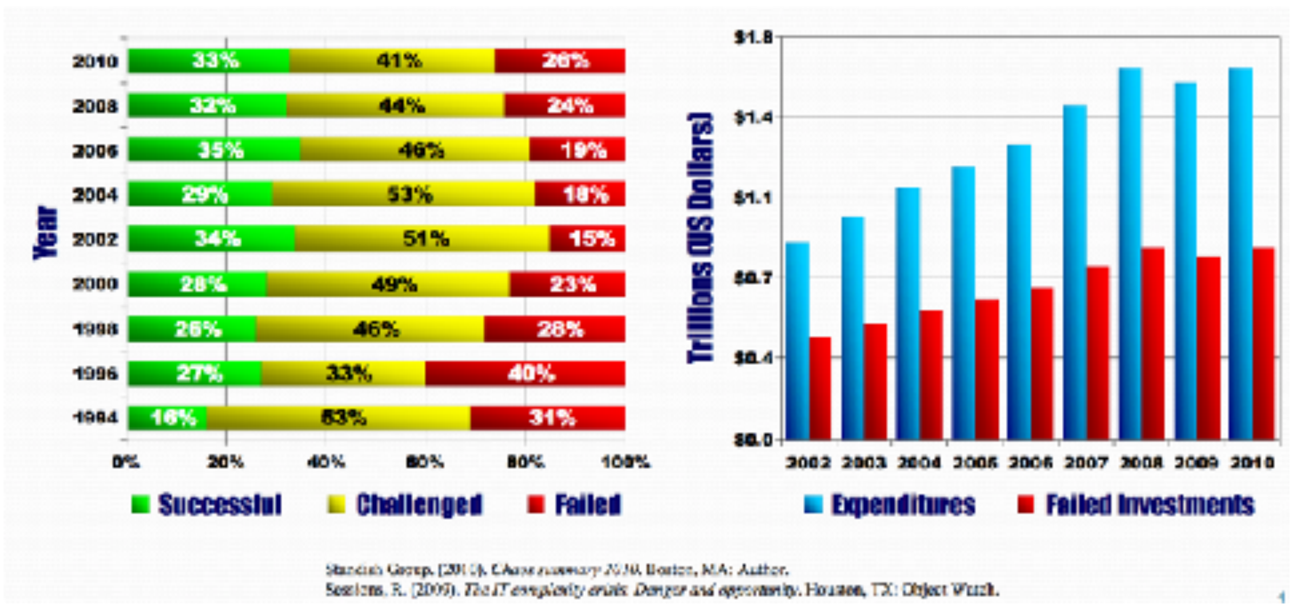
Too Simple:

Methods should be a simple as possible for delivering value, but no simpler.

By Tom Gilb, www.Gilb.com, Tom*gilb.com
Version 0.2 240317

Summary

IT and Software development *methods* have many problems. These currently popular methods are *trying* and *intending* to help us succeed better But up to now, after decades of trying, we still have a miserable failure rate. Very roughly 30% total failure and then additional 50% 'challenged' (partial failure). Alternatively since bigger projects fail more often [8C], the total waste, of all IT project spend, is about half. This is *professionally ridiculous*, and it is completely *unnecessary.*



Standish Group. (2010). *Chaos summary 2010.* Boston, MA: Author.
Sessions, R. (2009). *The IT complexity crisis: Danger and opportunity.* Houston, TX: Object Watch.

I would like to suggest something clearly and provably better than the unimpressive ideas that are currently popular (agile, lean, lan-ban, and the like).

Part of the problem is that the most popular methods are popular, because they are simple to learn (Scrum Master in 2 days), and simple conceptually (3 of this and 3 of that). But the currently popular methods, are not 'popular' *because* they are effective at delivering real value to stakeholders. They are popular because they are *simple*, and because they are *popular*. "Of course we are 'agile/lean' "!

There is nothing wrong with being simple to use, and simple to learn. I'm all for that. But the **main point of any method** is not simplicity or ease: it is **effectiveness** in delivering results.

The *effectiveness* of a method must be good enough to allow you to succeed in delivering the required value to the critical stakeholders. That is the main point of all development projects.

It does not matter how 'simple' a method is, if you fail in the project using it. It is **worthless**.

If *only one* known set-of-methods, leads to your project's success, then whatever complexity or cost they have, that is your 'minimum price of success'. *More* simplicity is not a successful option. That would be 'too simple'.

If several sets of methods lead to success, than we can be picky, and choose the set with the least burdensome resource demands. The cheapest set. We can optimize for method 'efficiency' (effectiveness over costs).

I am afraid that we, IT, have a software life-time decades-long failure culture. I am ashamed to be part of such childish and failure-prone behaviour.

I think I know what we need to do. And some people have done it and succeeded. But the Immature majority (99%) is not doing it, yet. And this failure future does not look like ending anytime soon. We might get World Peace, before we get Successful Software Projects.

But, in the vain hope that things might get better faster, I will offer my opinion, and maybe, at least, some of you will act on the ideas. And maybe at least *you* will become more successful; even if the world does not change much in your lifetime. I know from experience that there are thousands of really smart people out there, just dying to do great successful work, only impeded by their *education*, their *culture* and their *management .*

"**Bad management causes more failures than bad software engineering.**" (C. Jones [8C])

Maybe my ideas can give some help to break away from those negative influences.

"Things should be as simple as possible, but no simpler" [1, 4, 5, 6, 7]

Under this banner (simplicity for purpose) I am going to argue that:

For any given project, and environment *the simplest set of methods*, that leads to successful delivery of planned or needed value, is a good enough set. And I do not believe that such a set is likely to be a 'universal standard' of any kind. It just works in practice now. And there will be a large number of practical details that make up the successful set.

This implies that we need methods for *selecting* and *validating* the set of methods, partly *within* a changing **environment or culture**, and finally for a final tuning or selection of methods **within the project** itself.
This implies that we are able *measure* the values produced and costs incurred *during a project*, to tell us if the current set of methods are working or not.

We need effective processes of continuously learning what works and does not, in delivering the values to stakeholders.

The Problem

The central problem, as I see it, in the IT and Software culture, is the extremely high rate of development project failures and part failures.

Google 'IT Project Failures' (71 million hits today, juicy stuff on first screen): a good approximation of the status is 30% total failure, plus the next 50% Challenged (partial failure).

Of course things vary, and definitions of 'success' varies, as well as *degrees* of success. But the IT-project failure rate has been with us for decades, and is almost constant. No matter what methods are used.

I am very much in doubt as to whether I would classify the so-called 'successful projects' as really and truly successful; in the sense of really delivering value as planned to stakeholders. Most projects do not even define that stakeholder value in their requirements. They do not even know how to defined it: like quantifying 'security' or 'efficiency'.

In the Norwegian 'Successful Project' study [11] you can see that the notions of success are wildly varying, and very informal. Nobody in the survey mentions actual *measurable* delivery of planned stakeholder value levels. They *talk* about value and customer happiness, but there is nothing serious visible behind those platitudes.

In general I think we are, at best, looking at a definition of success, in that survey, as 'delivering function without too many bugs, within time and budget'. Very close to Agile Manifesto.

My experience is that almost nobody, and no culture is serious about measuring the critical stakeholder value delivery of a project. But to my mind, that is the *only* valid idea of 'success'.

Is there any hope?

Not really. No more than wars ending, or greedy people and companies suddenly becoming ethical and generous. Bad world-wide cultures change very slowly, and no Silver Bullet software method is going to change the root causes of our failure. IT Failure seems largely independent of particular popular methods through the decades.

The **root** causes of long term IT Project failure, are more constant.

The **deeper** root causes, according to my observations are something like these:

- increasing complexity and size of the IT problem
- lack of *relevant* higher education for managers, decision-makers, politicians, and even engineers
- a culture of *short-termed* thinking, motivation and planning
- lack of personal *accountability* for project failure
- lack of *focus* on delivering stakeholder critical values
- *greed* on the part of sub-suppliers to do more billable work on an IT project, irrespective of results
- *greed* on the part of systems suppliers to deliver 'kit', hardware and software, independent of useful results
- other factors I am sure; but these will do

The *superficial* 'root causes' , the ones some methods are trying to deal with, look something like this:

- bad quality requirements
- stakeholders who keep on changing their and, and are unclear about what they want
- inability to estimate costs and timings reasonably
- lack of good enough project management

- weaknesses in various development processes (architecture, reviews, testing, maintenance, coding, etc.)
- and all the other things systematically quantified by Jones [8]

Capers Jones [8] publishes a lot of detailed data about various processes and their error proneness. (namcook.com)

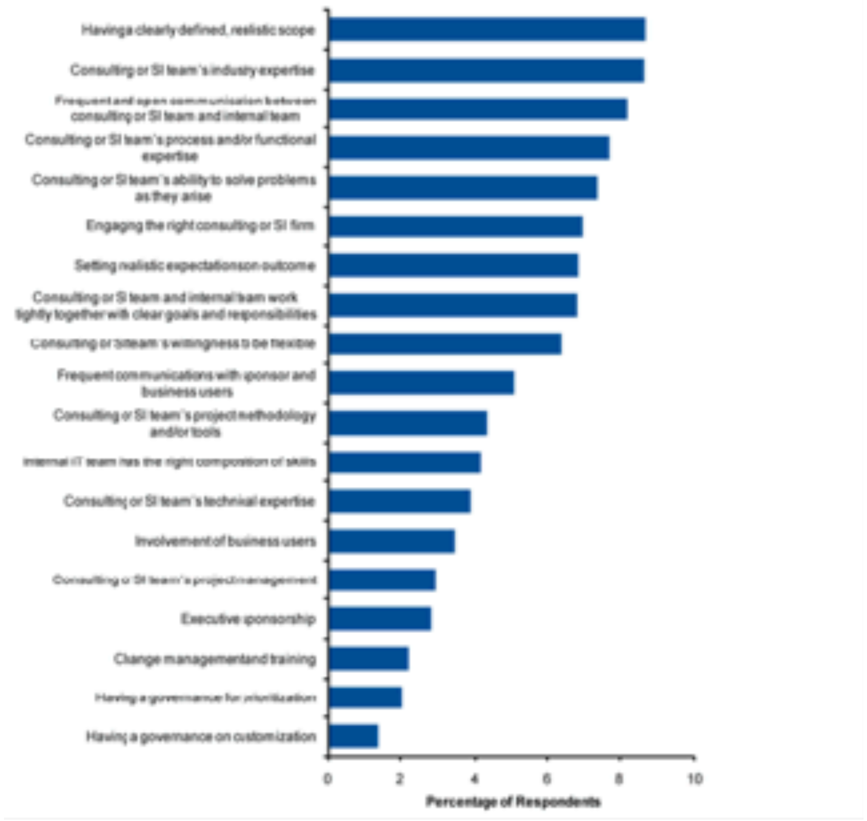


Figure: some opinions about causes of project failure. (C. Jones) [8]

But the detail of these superficial causes, these *direct* causes, is not my primary concern. I am initially interested in *what leads* to these development practices to be so poor. The 'deeper root causes' above.

I have an interest in speculating as to *why* these deeper root causes exist, and hurt us. And I have some opinions as to what we might do about them. But I do not hold out a lot of hope about changing world culture and human greed.

Only when we successfully improve the *organizational and environmental* root causes, will people be motivated and empowered to seek out far more effective development methods, learn them properly and apply them successfully in practice.

I do hold out hope for some individuals and some enlightened and motivated organizations to do much better than otherwise, in the short term. Their conditions allow them to learn and apply better development methods, than the oversimplified failed methods of today.

I know from my international work with enlightened companies like IBM, Intel, HP, Boeing, Ericsson that there really are environments that let their developers and engineers get on with the job of finding and applying superior methods for project success, corporate wide. I know it is the cultural environment, sometimes assisted with a sense of impending doom, to leads to successful adoption of more successful methods for delivering real value.

The becomes even more clear when I compare with some of my (nameless) clients where a corner of the company (a division of say 1,000 out of 100-300,000) has successfully implemented the advanced value delivery methods; only to be snuffed out by power hungry greedy elements higher up in the unhealthy organization. Or sometimes by market forces that make such improvements irrelevant; too little too late. The larger organization was not healthy enough to allow success to spread to the larger organization. There are even unpleasant degrees of this organizational unhealthiness, in even the best of the organizations.

The unsuccessful larger world population of IT will have to sort itself out without our help.

We need to focus on the limited opportunities to do good, where there is opportunity and receptiveness; and hope for the best, and hope it can last and spread. Hope it can set a good example to inspire others inside the organization and outside of it.

Do a great job on your own project, within your personal or team sphere. Inspire others, and feel good about your contribution. You will not always win. But you will learn, and you will be able to say ‘I really tried my best’. That’s what I do, anyway.

Many of us, my professional friends, feel the negative forces are disheartening. They are, but if you live as long as I have professionally (since 1958 at IBM) you can better see some real and interesting changes. But that’s the time frame it takes. The only thing that might change this year, is your own personal project, being better in some way.

The Solutions

I think I can suggest some methods which will increase the success of projects. Based on my experiences, and based on published experience and research [3, for detail and reference on the list of ideas below].

- continuous methods improvement, based on success measurement and delegation of power (IBM DPP)
- continuous stepwise delivery of stakeholder value, with measurement and feedback to guide the way to success (Quinnan, Cleanroom)
- no cure no pay contracting: value delivery payments for subcontractors (flexiblecontracts.com)
- personal and group responsibility for delivering value.
- transparency of plans and values: quantified and accounted, and audited.
- short-term ‘stopping’ mechanisms: when planned-value is not *really* being delivered

So, let us leave the technology aside: let’s talk about technology enablers

But these methods listed above, are all *software engineering* methods, methods which most professionals neither know about, learn properly or practice.

We still need to address the ‘deeper root causes’ that people do not take these advanced methods more seriously. Why the simpler, but less effective methods are more ‘popular’.

I should stress that what follows are opinions, speculations, and ideas for debate with friends. I know there is a lot written on this subject, and a lot of research on the internet. Yet I am tempted to offer my own observations and opinions.

Here is a list of my ‘root causes’ of why people do not employ much better development methods.

1. The ***higher education system***, universities, professors, have not caught up with the rapid change in technology and economics. They do not seem to have a clear opinion about what best to teach. They do not seem to know the specific ideas to teach. There is certainly no international consensus, which we are just ‘slow to implement’. There is no clear effort to discover these better ideas either. Hopeless really. The University might be obsolete, and being replaced by other faster and more effective ways of getting better. I have long believed that universities need to get back to their historical focus of teaching us to learn, rather than teaching us obsolete stuff [12, 13].

2. The **motivation system**; both individuals, and the organizations that employ them, have very little, in the way of clear operational motivation, for finding effective methods for really delivering stakeholder value [14]. There is little reward for increasing value delivery. There is little penalty for failing to deliver value. This is connected with our general inability as professionals to quantify and measure the many values we expect from our improvement efforts. If we were to be motivated we need this value quantification. But, the *quantification* methods, as little as many seem to know about them, are already there, massively, as the internet can demonstrate. We do not have the motivation to find out ‘what is known’ about ‘value quantification’ and ‘measurement’ [2,3], and apply it well. There *is* a big ‘greed’ motivation, to collect lots of money for bad IT system work. This is clearly much more powerful that any idealistic motivation to do good work for society.

3. **Lack of idealism**: the greater number of people I meet in my professional life are driven by their natural human need to earn a living, deliver to their family, have fun. But they do not strike me as driven to sacrifice any of those things if necessary, for the higher good of society [15]. Fortunately I have a circle of international professional friends who are ‘methods idealists’, and ‘society

idealists'. And we all work together and separately for improvement; for the greater good. But none of us is willing or able to push with the energy, urgency and power of a Ghandi, Martin Luther King, or even a Musk or Jobs for real change. There does not seem to be much point in 'dying for the cause'. If humanity is so lazy and foolish, they deserve what they get. Most do have some free will to do better, and they do not seem to care. Sometimes I compare the problems we have, with wasteful IT methods, to bigger problems like the Syria War, World War II, Refugee Migration, or the problems of education, equality, opportunity, health, nutrition and peace in much of Africa and other parts of the world. Clearly the 'IT waste' pales by comparison. Yet humanity is also terrible at solving these more-dramatic problems. Still, it is sad we do not seem to have far more selfless idealism. Elon Musk (Tesla, Space X) is the current example of a functioning idealist in the technical sector. Bill Gates was never the software idealist (Steve Jobs *was* the idealist); but the Gates are making up for it, in their current public work!

My own decision, regarding these underlying, root causes, is that there is no quick miracle going to happen. I can only do my very-ineffective-for-society personal work. I can develop and 'store (in books, video, papers, train younger people) 'effective knowledge'. I can hope that in time, people will be motivated to dig up this 'better know-how' (I hope!), and use it. But this is likely to be long after my lifetime.

History shows we take thousands, and hundreds, of years to learn to do the right things: but sometimes ancient ideas are employed like 'democracy', 'human rights', and perhaps one day 'high efficiency systems development'.

So, I am relaxed. Not given up exactly. Still in there trying to the bitter end. Maybe that is all most of us ordinary individuals can do; small ants that we are.

I am hoping that some of you reading this, will want to be more idealistic, and at least, as individuals do their best; and not lose hope at the slow pace of improvement.

Light a candle in the room you are standing in!

Principles of Success

1. Project success must be defined, and must be relevant, like real measurable values for stakeholders.
2. Responsibilities for success must be clear and personal
3. Payment should be for 'value delivered', not 'work done'
4. Successful value delivery must be early, stepwise, frequent, cumulative, and measurable.
5. When plans are not working to deliver value, then plans need to be improved immediately, so as to really deliver value.
6. Successful levels of value can be contracted for: that should be the main point of any supplier contract.
7. Don't expect anybody else to tell you what to do, do the right things to deliver value on your own initiative; hopefully you won't get fired too often for doing good.
8. Make sure you look at the big picture; the total system of people, motivation, culture, habits, hardware, databases, new emerging technology - and perhaps the 'software': code alone is worth nothing at all; don't pretend it is the 'main product'.
9. Reward creativity, and reward people and groups that improve successful practices in their own work, and who can inspire spreading practices elsewhere.
10. Simplify, and minimize, methods, but not to the point where success is less certain.

References

[1] Gilb, T. **Principles of Software Engineering Management**. (book) 1988.

Page 17: *"**Einstein's over-simplification principle**: Things should be as simple as possible, but no simpler!"* My attribution here was incorrect. So, you can quote *this (me, not AE !)* as the source!

[2] Gilb, T. **Competitive Engineering** (book) 2005
[3] Gilb, T. **Value Planning** (book manuscript 2016. leanpub.com/Valueplanning., and https://www.gilb.com/store/2W2zCX6z
[4] Alice Calaprice, **The Quotable Einstein**. ('not a confirmable quote' chapter: see "Things should be..")
[5] Minsky: Personal Communication. On Sun, Jun 6, 2010 at 5:13 PM, Tom Gilb <tomsgilb@gmail.com> wrote:
Prof Minsky
*"I just started re-reading 'Society of Mind'. I realized that your Einstein quotation is probably not correct and attributable to a checkable written source. I made the mistake myself.
See the Caprice book The Quotable Einstein for her conclusion."*

Minsky Replied to me by Email
*"What is this reference? Can you tell me her conclusion?
When my book came out, an Einstein archive in Jerusalem asked me for the source, because they could not find it.
I replied that I did not recall where I had heard it, but that I had known Einstein when I was at Princeton in 1950-1954, and perhaps had heard him say that -- or something like that. Also, I could have made a mistake, because I had trouble understanding his very foreign accent.
So perhaps he said **something simpler, but not much simpler**."*

[6] "**On the Method of Theoretical Physics**"
Albert Einstein
Philosophy of Science
Vol. 1, No. 2 (Apr., 1934), pp. 163-169

"It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience."

"It is essential for our point of view that we can arrive at these constructions and the laws relating them one with another by
adhering to the principle of searching for the mathematically simplest concepts and their connections."

Source: http://www.jstor.org/stable/184387?origin=JSTOR-pdf&seq=1#page_scan_tab_contents

[7] The Quote Investigator take on this
http://quoteinvestigator.com/2011/05/13/einstein-simple/

[8] Capers Jones publishes a lot of detailed data about various processes and their error proneness. (namcook.com)
A. *MINIMIZING THE RISK OF LITIGATION:
PROBLEMS NOTED IN BREACH OF CONTRACT LITIGATION*

B. How to Increase Your Project Success Rate 'Gartner Article'. Link at homepage of at namcook.com

C. Personal communication from CJ: 12 March 2017:

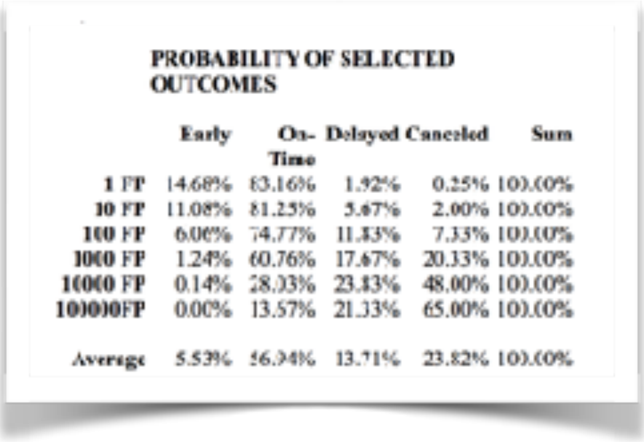
"Your points are good. Software failures are an endemic problem. I do a lot of expert witness work in the aftermath of failures so I know quite a few of the reasons. **Bad management causes more failures than bad software engineering.**

Here is my data on overall software failure rates and a paper I wrote that summarizes observations in various lawsuits for failed projects.

One of the reasons for failure is bad metrics and bad measurement practices. Most people don't know that agile is best for small projects and not good for big ones.

Here are samples of 13 methods in a side by side format. I have a book in production that has this kind of data for 60 methodologies.

Best Regards, Capers Jones"



PROBABILITY OF SELECTED OUTCOMES

	Early	On-Time	Delayed	Canceled	Sum
1 FP	14.68%	83.16%	1.92%	0.25%	100.00%
10 FP	11.08%	81.25%	5.67%	2.00%	100.00%
100 FP	6.06%	74.77%	11.83%	7.33%	100.00%
1000 FP	1.24%	60.76%	17.67%	20.33%	100.00%
10000 FP	0.14%	28.03%	23.83%	48.00%	100.00%
100000FP	0.00%	13.57%	21.33%	65.00%	100.00%
Average	5.53%	56.94%	13.71%	23.82%	100.00%

[9] David F Rico. www.DavidFRico.com
A rich collection of slides and studies of the relative efficiency of popular software engineering methods (like agile, Inspection and CMMI)

[10] "**Power to The Programmers**", as held Krakow ACE Conference June 2014
Video: http://vimeo.com/98733453. Slides http://concepts.gilb.com/dl841

Cases and experience of delegation of power to select methods to the 'shop floor' at IBM, Raytheon, Confirmit, Citigroup, Primark, ICL. Using Defect Prevention Process (organizational level), Evolutionary Value Delivery (Project level)

[11] Lubna Siddique and Bassam A. Hussein
A qualitative study of success criteria in Norwegian agile software projects from suppliers' perspective
International Journal of Information Systems and Project Management, Vol. 4, No. 2, 2016, 63-79
http://www.sciencesphere.org/ijispm/archive/ijispm-040204.pdf

[12] Gilb, Tom, "**Undergraduate Basics for Systems Engineering** (SE),
using The Principles, Measures, Concepts and Processes of Planguage".
(INCOSE Conference 2007 Presentation) http://www.gilb.com/DL98

[13] Gilb, "**Some Principles of Useful Knowledge**"
slides
http://www.gilb.com/dl844

[14] Gilb: “**Value Manifesto**”, March 2017, , http://concepts.gilb.com/dl898
[15] Gilb: '**Commercial Risk;**
Legal, Social, Ethical and Professional Issues in IT Project Management'
http://concepts.gilb.com/dl902

London Metropolitan University Lecture Slides
27 March 2017