let's see the numbers

Adding Stakeholder Metrics to Agile Projects

by Tom Gilb

A 2001 paper in the *British Computer Society Review* found that only 13% of 1,027 surveyed IT projects were "successful" [7]. Another report indicated that although there has been some recent improvement, 23% of its surveyed projects were considered total failures, and only 28% were totally successful (on time, within budget, and with all functionality) [6]. A few years ago, the US Department of Defense (DoD) estimated that about half its software projects failed.[1] This represents an improvement on the 75% failed projects the DoD reported when the waterfall method dominated [5], but it is of extreme concern. By traditional measures, we must be doing something very wrong. What can senior management and IT project management do about this situation in practice?

Some people recommend complex development process standards such as CMM, CMMI, SPICE, and their like to remedy the problem. However, I am not convinced that these are "good medicine" for even very large systems engineering projects, and certainly they are overly complex for most IT projects. Other people recommend agile programming methods.[2] These are closer to my heart, but maybe, for nontrivial projects, they are currently *too simple*?

I believe agile methods would benefit if they included "stakeholder metrics." All projects, even agile projects, need to identify and focus on the "top few" critical stakeholder requirements. These top requirements need to be quantified and measurable in practice. Quantified management is a necessary minimum to control all but the smallest upgrade efforts.

In addition to the benefits of focusing on the critical requirements and measurement of progress, feedback is a third feature of using stakeholder metrics. This is one element that agile projects, especially, can utilize.

In this article, I shall present a simple, updated "agile," evolutionary project management process called Evo and explain the benefits of a more focused, quantified approach. The Evo method is shown in the sidebar on the next page.

The process and policy outlined in the sidebar capture all the key features of the Evo method: you need read no more! However, in case any reader would like more detail, here it is. I will comment on the process and policy definition, statement by statement.

THE EVO METHOD, STEP BY STEP

Project Process Description

1. Gather from all the key stakeholders the top 5-20 most critical goals that the project needs to deliver.

Projects need to learn to focus on *all stakeholders* that arguably can affect the success or failure of the project. The needs of all these stakeholders must be determined — by any useful methods — and converted into project requirements. By contrast, agile models like XP focus on a user/customer "in the next room." This is good enough if that customer *were* the only stakeholder, but it is disastrous for most real projects, in which the critical stakeholders are more varied in type and number. Agile processes that exhibit this dangerously narrow requirements focus risk outright failure, even if "the customer" gets all of his or her needs fulfilled.

[1] Norm Brown, Software Program Managers Network/US Navy, personal communication.

[2] See [1] for a review of agile methods.

EVO: A SIMPLE EVOLUTIONARY PROJECT MANAGEMENT METHOD

Tag: Quantified Simple Evo Project. Version: 8 July 2003 (3). Owner: Tom@Gilb.com. Status: Draft.

Project Process Description

1. Gather from all the key stakeholders the top 5-20 most critical performance goals (including qualities and savings) that the project needs to deliver. Give each goal a reference name (a tag).
2. For each goal, define a scale of measure and a "final" goal level (e.g., *Reliability: Scale: Mean Time Between Failure, Goal: >1 month*).
3. Define approximately four budgets for your most limited resources (e.g., time, people, money, and equipment).
4. Write up these plans for the goals and budgets (ensure this is kept to only one page).
5. Negotiate with the key stakeholders to formally agree upon the goals and budgets.
6. Plan to deliver some benefit (i.e., progress toward the goals) in *weekly* or shorter increments (i.e., Evo steps).
7. Implement the project in Evo steps. Report to project sponsors after each Evo step with your best available estimates or measures for each performance goal and each resource budget.
   - *On a single page*, summarize the progress to date toward achieving the goals and the costs incurred.
   - Based on numeric feedback and stakeholder feedback, *change whatever needs to be changed to reach goals*.
8. When all goals are reached, "Claim success and move on" [2]. Free the remaining resources for more profitable ventures.

Project Policy

1. The project manager, and the project, will be judged exclusively on the relationship of progress toward achieving the goals versus the amounts of the budgets used. The project team will do anything legal and ethical to deliver the goal levels within the budgets.
2. The team will be paid and rewarded for benefits delivered in relation to cost.
3. The team will find their own work process and their own design.
4. As experience dictates, the team will be free to suggest to the project sponsors (stakeholders) adjustments to "more realistic levels" of the goals and budgets.

2. For each goal, define a scale of measure and a "final" goal level (e.g., *Reliability: Scale: Mean Time Before Failure, Goal: >1 month*).

Using Evo, a project is initially defined in terms of clearly stated, quantified, critical objectives. Agile methods do not have any such quantification concept. The problem is that vague targets with no quantification and lacking in deadlines do not count as true goals: they are not measurable, and not testable, ideas.

Note that in Evo, requirements can be changed and tuned during a project, based on practical experience, insights gained, external pressures, and feedback from each Evo step. They are not cast in concrete, even though they are extremely specific.

3. Define approximately four budgets for your most limited resources (e.g., time, people, money, and equipment).

Conventional methods do set financial and staffing budgets, but usually at too macro a level. They do not seem to manage directly, and in detail, the array of limited resources we have. Admittedly there are some such mechanisms in place in agile methods, such as the incremental weekly (or so) development cycles, which handle time. However, the Evo method sets an explicit numeric budget for any useful set of limited resources.

But it does not stop there. Evo cycles estimate, and then record, actual resource use and analyze the deviation (between estimate and actual use) on *every* Evo cycle, in order to understand and control the economics of the project — concurrently with measuring and monitoring the performance characteristics (see Figure 1). This is an essential distinction between incremental and evolutionary development methods. Evolutionary methods measure and react to this feedback.

4. Write up these plans for the goals and budgets (ensure this is kept to only one page).

All the key quantified performance targets and resource budgets are presented simultaneously on a single overview page. Additional detail about them can, of course, be captured off of this one "focus" page.

5. Negotiate with the key stakeholders to formally agree upon the goals and budgets.

Once the requirements — the version derived from the developers' understanding of stakeholder needs — are clearly articulated, we need to go back to the real stakeholders and check that they agree with our "clear" (but potentially incorrect or outdated) interpretation.

It is also certainly a wise precaution to check back later, as the project evolves, with the specific stakeholders who will be impacted with a particular Evo step. We will want to know how they feel about a particular choice of step content (design) and how it impacts the performance and cost aspect estimates. We will need to determine whether our estimates are realistic in the current implementation environment and to check for any new insights regarding the long-term requirements.

In the Evo method, it is not simply "What shall we do next?" It is "What is most effective to do next?

6. Plan to deliver some benefit (i.e., progress towards the goals) in *weekly* or shorter increments, (i.e., Evo steps).

Many agile methods adopt a weekly delivery cycle; this is good. However, the notion of *measurement* each cycle, on multiple performance and resource requirements, is absent.

In the Evo method, the project team chooses the next Evo step based on highest stakeholder value–to–cost ratios. It is not simply "What shall we do next?" It is "What is most effective to do next? What is of highest value to the stakeholders when we consider our resources?"

			Estimate		Actual		Estimate		Actual	
			Step 12 Buttons.Rubber				Step 13 Buttons.Shape & Layout			
	Goals		Impacts				Impacts			
1	User-Friendliness.Learn		-10	33%	-5	17%	-5	20%	5	-20%
	30	5								
	by	*one year*								
2	Reliability		-3	-3%	-1	-1%	20	20%	2	2%
	99	200								
	by	*one year*								
	Resources		Actual				Actual			
	Project Budget		2,000	2%	2,500	3%	1,000	1%	1,000	1%
	2,500	100,000								
	by	*one year*								

Figure 1 — The use of an impact estimation table [3, 4] to plan and track critical performance and cost characteristics of a system. (Illustration courtesy of Kai Gilb). The pairs of numbers in the three lefthand columns (30, 5; etc.) are defined benchmarks (30, 99, 2500) and goal levels (5, 200, 100,000). The "%" figures are the real-scale impacts (e.g., 20) converted to a percentage of the way from benchmark to the goal levels (e.g., 20% of the distance from benchmark to goal).

The agile methods' notion of agreeing with a user about the functionality to be built during a weekly cycle is healthy, but the Evo method is focused on systematic, weekly, measured delivery toward long-range, higher-level objectives within numeric, multiple resource constraints. This means that the Evo method is more clearly focused on the wider stakeholder set of values and on total resource cost management.

The Evo method is focused on delivering useful results to an organically whole system. We reuse, buy, or exploit existing code just as happily as we write our own code. We build databases, train and motivate users, improve hardware, install telecommunications, create Web sites, improve the working environment, and/or improve motivation. So we become more like systems engineers ("Any technology to deliver the results!") than programmers ("What can we code for you today?").

7. Implement the project in Evo steps. Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures for each performance goal and each resource budget.

All agile methods agree that development needs to be done in short, frequent delivery cycles. The Evo method specifically insists that the closed-loop control of each cycle be done through numeric precycle estimates, end-cycle measurements, analysis of deviation from estimates, and appropriate changes to immediate planned cycles, estimates, and stakeholder expectations ("This is going to be late if we don't do X"). It is the use of stakeholder metrics that allows Evo to have such control.

Projects need to be judged primarily on their ability to meet critical performance characteristics in a timely and profitable way.

A major feature of Evo is the clear intention to react to the feedback from the metrics and to any changes in stakeholder requirements. This helps keep the project on track and ensures relevance.

8. When all goals are reached, "Claim success and move on" [2]. Free remaining resources for more profitable ventures.

A major advantage of numeric goal and budget levels as compared to more informal tracking methods is that it is quite clear when your objectives are reached within budgets. The set of numeric goal and budget levels formally defines "success" in advance.

Projects need to be evaluated on performance delivered in relation to resources used. This is a measure of project management efficiency. When targets are reached, we need to avoid misusing resources to deliver more than is required. No additional effort should be expended to improve upon an objective unless a new, improved target level is set.

Project Policy

1. The project manager, and the project, will be judged exclusively on the relationship of progress toward achieving the goals versus the amounts of the budgets used. The project team will do anything legal and ethical to deliver the goal levels within the budgets.

Projects need to be judged primarily on their ability to meet critical performance characteristics in a timely and profitable way. This cannot be expected if the project team is paid by "effort expended."

2. The team will be paid and rewarded for benefits delivered in relation to cost.

Teams need to be paid by results delivered in relationship to costs, or in other words, by their project efficiency. This means that super-efficient teams will get terribly rich, and failing teams will go "bankrupt."

When only 13% of 1,027 IT projects are "successful" [7], we clearly need to find better mechanisms for rewarding success and not rewarding failure. I suggest that sharp numeric definition of success levels and consequent rewards for reaching them is the most appropriate mechanism for any software project.

3. The team will find their own work process and their own design.

Agile methods advocate reducing unnecessarily cumbersome corporate-mandated processes.

I agree. They also believe in empowering the project team to find the processes, designs, and methods that really work for them locally. I heartily agree! But I believe that sharp numeric definition of objectives and budgets, coupled with frequent estimation and measurement of progress, is a clearly superior mechanism for enabling this empowerment. The price for this — a few estimates and measures weekly — seems a small price to pay for superior control over project efficiency.

4. As experience dictates, the team will be free to suggest to the project sponsors (stakeholders) adjustments to "more realistic levels" of the goals and budgets.

No project team should be stuck trying to satisfy unrealistic or conflicting stakeholder dreams within constrained resources. The project team can only be charged with delivering inside state-of-the-art performance levels at inside state-of-the-art costs. Exceeding state-of-the-art performance is likely to incur "exponential" costs.

SUMMARY

A number of agile" methods have appeared that try to simplify project management and systems implementation. They have all missed the central point of *quantified* feedback. Evolutionary project management (Evo) uses quantified feedback about critical goals and budgets. It also insists that early, frequent, small, high–stakeholder value deliveries (Evo steps) be made to real users. This allows better focus, more measurement of progress, and more flexibility to change. It is time agile methods adopted quantified, critical stakeholder metrics.

REFERENCES

1. Abrahamsson, Pekka, Puti Salo, Jussi Ronkainen, and Juhani Warsta. "Agile Software Development Methods." *Proceedings of ESPOO 2002* (VTT Publications 478). VTT Information Service, 2002.

2. Gerstner, Louis V., Jr. *Who Says Elephants Can't Dance? Inside IBM's Historic Turnaround*. HarperBusiness, 2002.

3. Gilb, Tom. *Principles of Software Engineering Management*. Addison-Wesley, 1988.

4. Gilb, Tom. *Competitive Engineering: A Handbook for Systems and Software Engineering Management Using Planguage*. Addison-Wesley, forthcoming 2004.

5. Jarzombek, Stanley J. *Proceedings of the 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium (JAWS S3)*. US Government Printing Office Press, 1999.

6. Johnson, Jim, Karen D. Boucher, Kyle Connors, and James Robinson, "Collaborating on Project Success." *Software Magazine* (February 2001), www.softwaremag.com/L.cfm?Doc=archive/2001feb/CollaborativeMgt.html.

7. Taylor, Andrew. "IT Projects Sink or Swim." *British Computer Society Review* (2001), www.bcs.org.uk/review/2001/html/p061.htm.

Tom Gilb is a consultant, author, and teacher. He has published nine books and numerous papers. Immediately forthcoming is his latest book, Competitive Engineering.

Mr. Gilb primarily works at changing systems engineering cultures in major multinational corporations. His major technical interests are in the areas of requirements engineering, design and architecture, evolutionary project management, and specification quality control (inspection). His clients include McDonnell Douglas/Boeing, BAE Systems, Hewlett Packard, Nokia, Sony/Ericsson, Philips, CitiGroup, Intel, Microsoft, Canon, and United Defense. He does pro bono *work for US DoD, UK MoD, various charitable organizations, and in developing countries, such as India, China, and Korea.*

Mr. Gilb started working for IBM in 1958. He stayed at IBM for five years and has been an independent consultant since then. He was born in California and lives in Norway.

Mr. Gilb can be reached at Tom@Gilb.com; Web site: www.gilb.com.