CAI *Computer Aid, Inc.®*

**Presents
An IT Metrics and Productivity Journal Special Edition**

**Focus on Tom Gilb
Expert Evolutionary Development Practitioner and Author
A CAI State of the Practice Interview
December, 2005**

Biography of Tom Gilb:

Tom Gilb was born in Pasadena in 1940. He emigrated to London in 1956 and then to Norway in 1958, where he joined IBM for 5 years and where he currently resides and works.

Tom is recognized as a pioneer in software metrics and Evolutionary project management, as well as the inventor of the planning language Planguage. He is directly recognized as the idea source for parts of the Agile and Extreme programming methods (primarily the incremental cycles). He has published nine books, including *Principles of Software Engineering Management* (1988, 20th printing), *Software Inspection* (1993, 13th printing), and *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage*, which was published in July of 2005.

Tom has mainly worked within the software engineering community, but since 1983 with corporate top management problems, and since 1988 with large-scale systems engineering (Aircraft, Telecoms and Electronics). His methods have been widely and officially adopted by organizations such as IBM, Nokia, Ericsson, HP, Intel, Citigroup, Symbian, and Philips Medical.

Tom enjoys giving time to anyone, especially students, writers, consultants and teachers, who are interested in his ideas - or who have some good ideas of their own. He happily contributes teaching and consulting pro bono to developing countries (India, China, Russia for example), to Defense Organizations (UK, USA, Norway, NATO) and to charities (Norwegian Christian Aid and others).

Our interview between Tom Gilb and Michael Milutis, the IT Metrics and Productivity Institute's Executive Director, was conducted in December of 2005.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

CAI: What are evolutionary methods and how would you differentiate them from the classic waterfall paradigm?

TOM GILB: Perhaps the first thing I should explain is that about 10 years ago, in 1994,

the US Department of Defense officially changed their software engineering standards. They introduced Standard 498, which was the evolutionary standard. They officially turned their back on everything waterfall. That was after decades of learning that evolutionary methods succeed where waterfall methods fail.

Evolutionary methods simply involve the chopping up of any given project into a lot of smaller projects, a lot of smaller cycles. It is not unusual in evolutionary practice to chop things up into what we call two percent cycles, cycles of fifty increments. The idea here is that each increment will be an attempt to deliver some value to your stakeholders. The incremental approach will also help you figure out: 1) whether the technologies you're using really do deliver the value; 2) whether your staff really does understand what people want, and 3) whether your stakeholders acknowledge that you really did deliver the value you were talking about. From the perspective of a CIO, evolutionary methods give you a constant confirmation that everything is working in your organization. Moreover, you get this confirmation early and frequently. And when you have a choice of many things to do, you can actually choose the high-value things to do early, so they kick in early, with the obvious cash-flow ROI benefits that this has. That's one simple explanation - lots of small cycles of delivery and feedback that force you to get things right and if anything goes wrong, you find out about it early, whatever it is, whether it's your people, your organization, or a sub-supplier in India. Whatever is going on, if it is good or bad, you will know it immediately.

CAI: How widespread would you say evolutionary methods are these days?

TOM GILB: I've seen surveys done in which people claim that a very large percentage of organizations are doing it, something in the area of 30%. I'm quite surprised by those numbers, though.

CAI: Do you think they are too high or too low?

TOM GILB: I think they are too high. I would think a very small percentage - something between one and five percent - are doing it. A lot of people think that because they build and deliver code every week they are doing this. However, if they are not actually delivering final results, final requirements to real users and stakeholders, then they are not actually using evolutionary methods. A lot of people also think that simply working in weekly cycles and turning code out is equivalent to delivering results with value.

CAI: Are there any situations in which evolutionary methods would not be appropriate, or would not be the best approach to take?

TOM GILB: To my knowledge, no. And I've thought a lot about it.

I've been doing this since 1960. I've practiced. And I've seen no exceptions. Nevertheless, time and time again I meet people who say "my project can't be done evolutionarily." And then we sit down for about an hour and I show them how to do it. In my new book *Competitive Engineering*, in the chapter on evolutionary deliveries, I have a section on how to decompose projects into such small increments. That's usually the area in which people put up initial resistance. They say, "I would love to start, I would love to deliver results early, but I just can not chop up my project into such small increments." That's what they say. It inevitably turns out, however, that the real problem is something else. The real problem is that they don't know how to do it. They've simply not been trained in how to do it.

CAI: For organizations that are seriously interested in going down this path, is there anything that they should be careful to avoid? Could you offer any advice to organizations that would like to get started with an evolutionary program?

TOM GILB: The beauty of the evolutionary program is that, in the short term, if you're doing it right, you will be able to find out right away whether it works or not. You won't have to wait around years to find out. Just until next week.

Consequently, my first piece of advice is to do it. Just do it. Take any project, think of it as a sort of experiment, and see for yourself if it really works. Better yet, get a team to volunteer to try it out. I would also suggest reading up on the literature, if you are serious about this.

Regarding caveats, the primary thing that people aren't doing well here is their quantification of what the stakeholders and the users really want. For example, if a company really wants to save money on staff, and that's the whole point of the project, then I think it's necessary to continually emphasize that this is the primary objective of the project. Each evolutionary step should demonstrate that saving of staff money. In other words, it doesn't matter how many function points or stories you've written. If your handover of software does not actually result in savings, then you are not delivering the value to the customer that the customer really wants.

CAI: How does one do this at each step of the way?

TOM GILB: That's a practical matter that would need to be worked out in detail. However, to continue with the previous example, if you manage to save the time of an employee, but you cannot fire them, then you are not necessarily going to save any money. So you are going to have to decide whether you are trying to save time, time that will later be converted into savings, or if you are trying to save money. In this case, you would have to have a very clear understanding that the whole point of the IT project is to generate staff savings, not just time savings. And every evolutionary step would then have to prove that it is aligned with that objective. Every step must attempt to do this.

Now if nobody can figure out how to do this at all, even in a simple way, then I would suggest that the whole project might as well stop. If nobody knows what they are doing, they won't know how to create savings. You will find that people are always able to make a lot of excuses, though. "We have to spend 10 million dollars." "We have to spend 10 years on this to create the desired savings." When you hear these lines, keep in mind that history has been, and continues to be, pretty clear about the flop rate on projects.

Evolutionary methods, in this respect, are simply a way of making sure that if an IT project is going to deliver something, the organization will have to figure out how to deliver in the short term. It's just easier doing things on a smaller scale than on a large scale. More importantly, if you can't even do things on a small scale, there's certainly no reason to believe that you could be successful on a large scale.

CAI: Could you give me any real world examples, from your own experience, of how evolutionary methods have been successful?

TOM GILB: There is a small Norwegian company called FIRM which stands for Future Information Research Management. They have a piece of software called Confirmit, and it basically allows you to conduct research surveys on the web. The product is five or six years old. It is kind of a legacy product and so they're trying to make it better. At one point they discovered that, in order to set up a market research survey, the average user needed two hours. Obviously, they realized that there would be value in reducing the set-up time for a market survey. So they focused on that, and a half a year later, after twenty iterations of evolution, they had reduced the set-up time to 15 seconds.

My point is that instead of saying, "We'll build a faster better IT system with some new technology and then everything will go faster," they focused on the simple idea that it takes two hours to set up a survey. They simply focused on those two hours and asked, "What is it that's causing this to take two hours and how can we speed it up," and then they delivered that. They continually reduced their targets in one week cycles - 120 minutes to 30 minutes to 15 minutes and on and on - just by staying focused on it.

They had never focused on these things. They had always focused on the traditional list of functions and features that the salesmen maintained Boeing or Microsoft were asking for. So they kept on building these functions and features into the product, but as it turned out – and as the US Department of Defense also found out - half of all these functions and features were never being used by any of the customers. There was a complete miscommunication. They were spending all of their energy building stuff that nobody wanted.

CAI: So from an evolutionary perspective, you're simply saying that at each step of the way you have to be able to demonstrate that you are working towards the objective?

TOM GILB: You have to be able to measure that you are getting the numeric savings or improvements that you intend to get, even if you are only getting them partially. For example, FIRM works toward a release in about 12 weeks and they use 12 one week cycles to do this. Every one of those 12 weeks has to be heading towards the goal they set for that release. And at the end of every week, they will literally measure this at the customer site. In so doing, the customer gets to see that these improvements are really in their product, that it is an everyday, progressive, incremental thing.

CAI: Earlier in your interview, you described the division of projects into 50 two-percent steps. What exactly do you mean by two percent steps?

TOM GILB: Evo policy says that things are to be done evolutionarily in steps and that no step is allowed to be planned which uses more than two percent of the total budget. Additionally, no step is allowed to be planned which uses more than two percent of the time to deadline.

Most untrained people will squirm and scream and shout that this is not possible. And this is never true. You can always divide anything, and I've even done this on very large Department of Defense projects such as the US Army personnel system. I've found that within one hour, I can find a way to divide things up, always. At Hewlett Packard, for example, they've been doing this for 17 years, as a matter of course. There's no question there that they should do things in weekly cycles. And Hewlett Packard is a big company. This little company FIRM has spent two years now doing things in weekly cycles and they have never had a problem finding weekly increments. You can always find something to do within a week.

CAI: Would you say that one of the primary advantages of the two percent method is that it allows an organization to be more nimble, more flexible?

TOM GILB: Absolutely. That's why the agile people love it. They perform short cycles, they deliver something, and they keep the customer happy. By the way, Microsoft is a big user of this method, too. This was originally documented in "Microsoft Secrets," a book by Dr. Michael Cusumano of MIT. At that point, Microsoft was a client of mine too. And I was working with the testing director, who said to me, "Tom, Evolutionary delivery is a way of life here at Microsoft." I didn't know that at the time. That's because they didn't call it evolutionary. They called it "synchronize and stabilize." They called it a whole lot of things.

In practice, however, you will see that Microsoft has three different levels of evolution. They've got their daily builds, they've got their 'milestones' that take about 6 weeks, and they've got their releases. It's nothing new. Microsoft, in fact, is one of the largest practitioners of evolutionary methods.

CAI: You mentioned earlier that the Department of Defense has been using evolutionary methods now for 10 years? Do you see this anywhere else at the Federal level?

TOM GILB: First of all, let's be very clear about something. The Department of Defense has their own practice. What they really do is, for example, take a four year project and use monthly increments. They've been doing that since the 1980s. That's still a two percent step, though.

Harlan Mills' IBM Federal Systems Division did a project over a four year period and every month – that's two percent of four years – they measured and tested and delivered. And they succeeded in getting all of their projects completed on time and under budget. This included the Space Shuttle ground software which consisted of 10,000 man years of development work. Harlan Mills demonstrated that this worked at NASA, for the US Department of Defense, and for the Navy. And this was in the 1980's. In fact, we recently discovered that the flight software on NASA was done in this manner, too. We also found something that we didn't expect. We found that space scientists and rocket scientists have been doing this since the end of the Second World War. Nevertheless, most of the rest of humanity has been stuck using the waterfall model, to this day.

The rocket scientists actually asked themselves, "How does a rocket get to its target?" They were talking about intelligent missiles, not SCUD missiles. An intelligent missile knows where its target is. It even keeps track of its target as it moves. This is the same thing that happens with shifting requirements.

"How does a rocket get to its target?" What a rocket does, what it keeps on saying is, "Am I actually heading towards where the target is? Evolutionary methods try to track changing requirements in this manner. Evolutionary methods make it possible to adapt to changing requirements because, within a two percent maximum, you will always be able to change course. IT people frequently give excuses like, "You (management) can't change requirements in the middle of the project." But Evo states that you have to be able to adapt to changing requirements. This is a fundamental reality that must be acknowledged.

Questions? Suggestions? Comments? Please contact the IT Metrics and Productivity Journal Editor at **michael_milutis@compaid.com**