SATM
Innovation & Technology Management News
Alliance for Technology Management
A Business Resource at STEVENS Institute of Technology
Fall 1997
Volume 1, Issue 3

Software Myths, Metrics, and Other Thoughts
by Richard E. Machol

Management of software development projects has been a recent focus of Alliance attention. This topic was addressed at the June SATM Conference, and our September Roundtable continued the discussions. Learnings from these forums are highlighted in this newsletter, along with some additional perspectives.

Larry Gastwirt
Director

Introduction

"Software development is an art; therefore, standard engineering principles used to estimate cost or schedule cannot be applied." Have you heard this, or thought it? I must ashamedly admit that I've had those thoughts in the distant past, but now I say, "NONSENSE!"

In a June 1993 publication, SRI International published an extremely interesting paper on the results of its study of software projects in American industry. It found that less than 1% of commercial software projects shipped on the *originally* scheduled date. Perhaps, you might conjecture, that was then but now the industry is doing much better. I say, "Show me the data!"

At the risk of offending some, let me mention several of the more recent, highly publicized, software *slips*.

> In July 1994, the software controlled baggage handling system for the new Denver airport slipped again. A year late, and still not working, the city of Denver was incurring a million dollar a day expense for the new airport, which couldn't open because the baggage system didn't work.
>
> We all are quite familiar with Windows 95, released in September 1995. What was the originally scheduled date? I believe it was January, 1994.

(continued on page 5)

Software Myths (continued from page 1.)

I don't mean to fixate on Microsoft, but the anticipated Windows 97 will probably be released sometime in 1998.

In a Time Magazine cover story in February, 1996, it reported : "FAA pulls plug on new Air Traffic Control System after cost estimates ballooned from $8 Billion to $37 Billion, with no functioning system in sight."

In March 1997, the Wall Street Journal reported. "The IRS has spent over $3 Billion on improving their systems over the last three years, and has *scrapped* the project."

The San Francisco Chronicle in March 1997 reported: "The National Crime Information Center database of criminal records was originally estimated to cost $73 Million. Current spending exceeds $183 Million. It's four years behind schedule, and it is uncertain if it will ever perform as originally anticipated."

What causes software project delays? Numerous reasons, including:

- Poor project size estimating
- Attempts at schedule compression by adding staff
- Poor feedback on progress
- Passive user involvement
- Lack of software controls
- Poor system run time performance
- Introduction of too many new technologies at one time
- Lack of a solid software architecture

I would summarize the reasons as *lack of software project management.* Yes, it's the responsibility of a software project manager to ensure that issues like the ones listed are addressed up-front in a project. A project manager is not just a correspondent, reporting on project status. Rather, she or he must be intimately involved in the issues critical to project success. The fundamental teaching in *Project Management 101* is to develop a project plan and to manage it rigorously.

A complete project plan must address all of the issues identified as lacking (see above), and many more. Entire books are written on these issues, some dating back twenty years or more. The classic, "The Mythical Man Month," highlights the fallacy of adding yet more staff to projects already behind schedule. And, I'm embarrassed to say, we continue to make this mistake. Seems that we in the software development community need a refresher on what is known. This might prevent us from repeating mistakes of the past.

Sofware Myths

As in most disciplines, myths abound in software and contribute to an industry-wide poor software delivery track record. I've selected four myths to discuss: (i) anyone can develop software, (ii) anyone can "play any position", (iii) prototypes will be thrown away and, (iv) software vendors advertise product limitations.

Myth #1: *Anyone Can Develop Software*

If only it were true, but it isn't. Software development is an engineering discipline requiring strong analytical skills. Many colleges and universities have outstanding engineering and computer science programs which prepare students to *enter* the software profession. Yet, far too many software practitioners lack this preparation. This, in my opinion, is a major problem in the software industry today. I readily admit that these skills also can be learned through years of practical on-the-job experience. Whatever the vehicle for learning, possession of these skills is critical to project success (and can differentiate a good developer from a good developer/manager).

Myth #2: *Anyone can "Play Any Position"*

This myth is an extension of the first myth. It's the fallacious assumption that any software developer can be effective in any role on a project. Perhaps the metaphor below will illustrate the point.

Suppose members of a major symphony walked on stage, and each was randomly handed an instrument to play for the performance. One received a violin, another a trumpet, a third an oboe, etc.

What would be your expectation of the ensuing performance? Not very high, I suspect. So too, software projects require experts in many

Coming this Spring '98
SATM Conference
on
Radical Innovation

Software Myths (continued)

disciplines and they are not necessarily interchangeable. One needs a set of skills to ascertain requirements from the customer. Another needs an in-depth understanding of software architecture. Still another needs skills to develop a comprehensive test plan. The point is that many unique skills are required on a successful software project, and individuals possessing those unique skills are critical to project success.

Myth #3: *Prototypes Will Be Thrown Away*

This myth just won't die. Prototypes are quickly constructed software piece-parts, developed as throw-away, and with the intent of either demonstrating project feasibility or ascertaining customer needs. However, invariably it seems, once a prototype is built, pressures mount to quickly turn the prototype into a product. The customer expectation frequently is that this will be able to be done quickly. Often this leads to misunderstandings, and sometimes project disaster.

To avoid this problem, I urge the project manager to communicate to the customer exactly what a prototype is, and set the expectations at the outset that converting a prototype to an industrial strength product is an order of magnitude more work. Make the point that prototypes must be thrown away, and stick to it.

Myth #4: *Software Vendors Advertise Product Limitations.*

In the software industry today, most systems are built using many off-the-shelf software packages. This frequently results in higher quality, lower cost software products, but it also has put an additional responsibility on the project manager. Routinely, often very late in the development cycle, major product limitations are uncovered. The discovery of performance problems, in particular, seems to regularly reoccur.

It's the responsibility of the project manager to see that each vendor product used within a system is thoroughly evaluated. Evaluation must be done using a disciplined engineering approach, which includes an in-depth analysis of feature functionality and scalability. Scalability is particularly important, in my view. For example, what works in a system having ten users may completely fall apart with a hundred users or more. As the age old saying reminds us, *caveat emptor.* When it comes to using vendor products in a software project, I say to the project manager: evaluate, evaluate and finally, evaluate again!

Software Metrics

Metrics can be very useful in pursuit of higher software quality and productivity. Generally speaking, I suggest fewer metrics rather than more. Also, later in this paper I'll caution about the abuse of metrics. Six metrics which I've found helpful are:

1. User satisfaction
2. Interval Ratio
3. Cost Ratio
4. Customer found defects
5. Defect density
6. Function points per staff month

User Satisfaction is a measure of what the user thinks of the product. It can be gotten in many ways, such as through a round table discussion or by users filling out a formal customer satisfaction survey.

Interval Ratio is the ratio of the actual interval required to develop the software product divided by the original estimate interval. An interval ratio of 1 (one) would mean that the product was delivered on the originally scheduled date.

Cost Ratio, similarly, computes the actual cost to develop the product divided by the originally estimated cost. Again, a value of one would mean that the actual and estimated costs were the same.

Next Roundtable

Managing Customer/Contractor Teams for Product Development

December 17th
2:00- 5:00pm
AT&T Location (to be announced)

Contact Dr. Lem Tarshis :
ltarshis@aol.com

Software Myths (continued)

Customer Found Defects is a count of actual product defects found by the customer once the product is delivered. As you might expect, this number continues to increase as time passes, and more users access the system.

Defect Density is a metric computed during system test, prior to release of a software product. It's a normalized way of measuring the quality of software delivered to system test.

Function Points per staff month is a productivity measure for a software product. Function points are used industry-wide to help estimate the size of a project, as well as providing a measure of productivity.

Variations on these metrics are common. For example, dollars per function point might be a variation on function points per staff month. Whether these or other metrics are chosen, I strongly suggest not picking too many. I would offer that the one we're found most useful is the Customer Satisfaction survey. Even if productivity is high, and schedule and cost have been accurately estimated, in the end, what counts is whether the customer is satisfied.

Metrics Measure Process, Not People

I vividly recall one crucial lesson I learned while attending a Quality Management seminar conducted by W. Edwards Deming a number of years ago. During the lecture he conducted his world famous *Red Bead* experiment. Oh, how he relished acting out the production manager role in evaluating the willing workers. The lesson was driven home to me so well that I'll never forget it. That lesson was : **"Metrics should be used to measure process, not people!!!"** He argued convincingly that productivity is much more a function of the process and tools used, and much less a result of the people carrying out the process using the specified tools. His message is clear. Metrics should be used to measure and improve the quality process; never for punitive purposes. If you use them for other purposes, such as for merit review, you'll very likely get what you should expect, but it won't be improved productivity!

Creating a 'No-Good-News' Environment

I would like to conclude my thoughts with some comments about the environment in which we work. One reason, I believe, that software projects have such a dismal on-time track record has a great deal to do with the environment in which regular project status is reported. Frequently, such status meetings are structured to highlight positive aspects of the project status, and inhibit the flow of negative information. "Shoot the messenger," is a frequent sport among participants. Optimism often wins out over objectivity, and project status is often reported as 'somewhat behind, but we're going to catch up shortly.' Sure!!! People are discouraged from reporting 'bad news.' The result oftentimes is that by the time the project delay can no longer be disguised, it's too late to do anything about it. A technique I've tried with some success is to not allow good news at project status meetings. Require each speaker to only discuss what isn't going as well as hoped. And, don't shoot the messenger. In fact, a project manager or other attending managers should go out of their way to make comments like, "Thank you for sharing that bad news with us! It gives us an opportunity to do something about it while there is still time." This takes some discipline and self-restraint, but it works.

Summary

Software development is an engineering discipline and applying standard project management techniques can dramatically improve the 'track record' of software deliveries, industry-wide. Software myths must be challenged if the software industry is to improve its ability to accurately predict and reduce software project intervals and costs. The collection and analysis of relatively simple software metrics, when followed by an action plan for improvement, offer significant quality benefits. Finally, the work environment, especially related to reporting project status, is crucial for identifying project problems in a timely manner.

Richard.E.Machol is the Network Vice President of the Operations Technology Center at AT&T in Middletown, New Jersey. He is responsible for systems engineering, planning and development of operation systems for managing AT&T's long distance network; by far, the largest telecommunications network in the world.

Dick would welcome dialogue on his remarks via e-mail. His address is em@orbit.hr.att.com

Please forward all comments and suggestions regarding this newsletter to Dr. Jack McGourty (Editor) 908-879-2769 or jacmcg@aol.com