

Project Management: Methodologies versus Madness in the 21st Century

George Kelly Flanagin & Susan R. Jacobs

Capital One Financial Services, Richmond, Virginia, USA

gf@eworksfs.com, srj@eworksfs.com

Abstract

This paper is a record of the invited speaker talk being given at Software Methods and Tools 2000. It is a catalogue of seven interlocked propositions about the future of software project management as a tool for getting software products out the door. The authors claim that while software project management is here to stay, the day-to-day activities of project managers are likely to dramatically change over the next ten years, and that the current methods of training project managers may not be giving the students the most immediately usable skills.

1. Introduction

These predictions are based on a review of current data and good bit of intuition. The invited speaker talk is not a research paper: the current data are expiring fast, almost too fast to always be reliable, and that short shelf life certainly one of our points. The reader may be left wondering exactly what intuition is, and we offer a useful working description provided by Philip Davis & Ruben Hersch:

“Intuition is the opposite of rigor. Intuitive means visual. Intuitive means plausible in the absence of proof. Intuitive means incomplete. Intuitive means relying on examples. Intuitive means holistic or integrative as opposed to detailed or analytic.”

The authors’ hope then, is that the reader finds the predictions to be provocative, entertaining, and most of all, debatable. We should look forward to rereading the paper in five years to see how many of the predictions hold up.

2. Our Predictions

2.1 Certification of software professionals is a growing trend that will continue unabated.

In a sense, it is about time that there be a certification for software professionals. In the United States, not only must attorneys attend law school full time for about three years, but they must pass a bar exam in the States in which they plan to practice. While a doctor is simply a graduate of an accredited medical school, the medical community recognizes the importance of hands on experience (medical residency) and uniform testing (board certification).

As well as PMP [Project Management Professional] certification offered by the PMI [Project Management Institute], we are seeing a vendor supplied credentials system for the understanding of certain popular commercial tools: “Microsoft Certified Java Developers,” and “A+” certification for PC maintenance come to mind. Lately, we have been seeing job applicant CVs with “MCSE” mentioned as a part of the alphabet soup just

like “MBA.” Vendor supplied credentials are *ipso facto* suspect: the vendors make money by offering a certification for their own product. This is a particularly risky situation for products such as tools from Rational Inc., where studies have shown that only a few customers use the product in its entirety.

At the current time, the system for credentials in the software business is about as far advanced as the system for therapists and counselors in California. In California, anyone who hasn’t been convicted of a felony can hang up a sign, get a business license, and begin charging people for advice. Certainly there are people who are licensed social workers and board certified clinical psychologists, but it is not a legal requirement.

There is big money in the system of software development credentials. ESI International offers a series of eight one-week classes at about \$US1700 each, and then offers additional classes to study for the PMP exam. An MSCE certification will set you back at least \$3000 in books and other materials alone! In the current economy, employers are paying for the classes and they are often offering small increases in pay to those who pass the exam.

It should be cautioned that none of the credentials offered at the current time ensures that the holder *can* or *ever has* delivered a production software system. In fact, the credential system may be promoting a further decline in the number of “soup to nuts” software developers through the increasing use of “drive-by” development strategies. A “drive-by” development strategy is one in which (as an example) an engineer who is expert on requirements gathering does this for one project at a time, moving on to a new project while the first projects are still in development ... never staying around to analyze the impact of the work done nor participate in the delivery. The most notorious example of this strategy is drive-by architecture, in which someone who has little or no responsibility to the users or the developers applies a bit of architecture at some point during the development process.

A further prediction: Until we see a board certification of software generalists, the proliferation of software development credentials will get worse and more confusing long before it gets better. At this time, we are unlikely to see a board certification of software developers because as an industry we suffer from the delusion that knowledge of the tools *is* equivalent to knowledge of

how to successfully complete the work to be done.

2.2 As software becomes a commodity, software professionals will go to software companies in increasing numbers.

The authors work at a large bank on the east coast of the United States. It is widely accepted at our place of business that the bottom rungs of the power ladder are in marketing and finance not in IT, even though there are around 1700 IT employees as of mid-2000. The median tenure of employment in IT is very close to one year, a figure that is a result of recently added jobs and high attrition.

Our exit interviews show that the IT professionals who leave our eCommerce group frequently leave to start a business of their own, or go to a business where the IT piece of the pie is quite a bit larger, and the product itself is a software product produced by a software company.

If these anecdotal data are to be believed, then the current recruiting difficulties will devolve into a depletion of the IT capable workforce. One is faced with the difficult prospect of needing to attract at least a small number of IT savvy project managers and designers to understand the needs of the organization. If your firm is a bank *per se*, as is ours, and is not an IT shop of its own, the more difficult attracting and retaining software professionals will become. Based on recent data in *The San Jose Mercury News*, the most difficult task of all may be finding software project managers.

2.3 More companies who are in need of software will turn to outsourcing.

Outsourcing is the logical next step in recovering from the flight of IT professionals to IT product companies. In its most basic form, outsourcing is simply an admission that one’s company is in some other business than the software business. For a few companies, this can be a spectacularly difficult admission, one best made in the office of the corporate psychiatrist. Whatever the difficulty, at this point one then turns to the software vendor in the same way that one turns to janitorial companies, grounds keeping companies, or general contractors to build new facilities.

Project managers who remain with the customer organization must learn to deal with the reality that actual construction of the software products will be done by third parties. Learning to trust vendors is a non-trivial process. One bad experience with work diverted to a vendor can result in years of keeping one's corporate head in the sand, as the authors have found.

2.4 The growth of software education in “third world” countries will gradually change “outsourced” to “offshore.”

If we accept the idea that there is an IT labor shortage in the United States, Canada, Australia, and certain areas of Western Europe, then one must look elsewhere to fill the jobs. The system of work visas in the United States puts certain logical limits on the number of workers who can be directly added to the local rolls through importation of the labor force, and there is still marked resentment and fear that foreign workers will take domestic jobs despite evidence that there are more jobs than there are people to fill them. India, in particular, has a fine and well developed educational selection process and a large number of applicants to it. The graduates of India's university system offer the prospective employer and customer access to a population in the upper tail of the bell curve.

Given the logistical limitations, there are three types of offshore development facilities to consider, we will call these Types 1, 2, and 3.

Type 1 offshore development is the business model used by several Indian software companies including Infosys, Tata Infotech, Cognizant, Satyam, and many others. In this arrangement specs are sent to a development team located in India in a facility owned and operated by the vendor. From a project management standpoint, the interesting problem is where to draw the line between the work done in house and the work that is sent out. Does one send out functional specs, requirements specs, detailed design specs, or what combination of the above? For that matter, what types of projects make good candidates for offshore development?

Type 2 offshore development refers to owning a wholly owned subsidiary in the foreign country. For some larger companies, this is the “end game,” the place where they hope their business strategy will eventually take them.

It has the advantage of directly competing for the labor force in the foreign country, and has the disadvantage of needing to supply one's own management, recruiting, facilities, etc., in a country where one may not know all the rules of engagement.

Type 3 is the system by which the overseas vendor brings a significant number of employees to the customer's site. The vendor provides the infrastructure (project management, work visas, and general support) and the personnel while the customer provides the site. In most cases, this approach is used only by customers who don't feel entirely ready to jump to Type 1 or Type 2 immediately.

2.5 More products will be produced through integration of existing services than through writing software from scratch.

Most of what one reads about project management and the product lifecycle is based on the idea that the product will be produced in house and *ex nihilo*. A tremendous amount of the literature of project management has gone toward finding adequate definitions and common terminology for various phases of the product lifecycle. We offer the personal opinion that one of the appeals of “iterative” or “spiral” development models is that they blur the lines between phases quite effectively, and conceal the truth that *much of what goes on in software construction is simply structured hacking!* It ought not be the case that construction is a quasi-chaotic activity sandwiched between requirements definition that precedes it, and the testing and deployment that follow it, but that is often what goes on despite the best efforts of project managers.

To complicate matters, one may want to consider that a typical internet e-business endeavor may have separate vendors for:

- hosting
- new technology
- web development
- interfaces to legacy systems
- application server development for the new product

as well as one's own in house employees. The project manager could easily be looking at several different methodologies and terminologies. A project manager has a job that is much closer to managing a manufactur-

ing assembly area than the older analogy of overseeing the manufacture of custom furniture.

2.6 In this sense the Internet is radically changing software development *and* project management.

In any Internet product here will also be dynamic content, an effort to incorporate some new technology, and production support changes that will take place *in situ*. In a large enough project, one runs the risk that some one or more of the vendors may go out of business before the project is even completed. A project manager rarely has the political clout to get the customer to pay for a backup plan in addition to the mainline of product development.

As a result, the make-or-break project management skill set lies, as some would claim it always has, in adapting quickly to changing circumstances and having adequate backup plans rather than a deep theoretical understanding of methodology. The most essential project management skill becomes thinking on one's feet. Thinking on one's feet is a valuable skill.

Consider the following example: In years gone by there was a fairly well understood definition of "alpha" as having a meaning very close to "functionality complete." While things might not work perfectly or even well, the features of the software product were all present and could be compared directly with the list of *required* features from the previously written specs. A project manager could schedule an audit of "alpha readiness" and be assured that one had reached one of Fred Brooks' "sharp milestones."

In the internet product assembly model, the unlucky project manager will have more than one vendor, each of which may be using an iterative development model. Any significant state of "functionality complete" is likely to occur only near the end of the construction phase, and this fact considerably complicates reliable testing strategies that are concerned, as they should be, with feature interoperability! Most dangerous of all is a hidden danger that no project manager can easily defend against: iterative development assumes that the software possesses an essential design that supports feature by feature additions. While this is possible to achieve, it is not easy!

2.7 The net result of all these changes will be an increasing amount of mass customization through self customization.

Capital One is considered by some to have pioneered the concept of "mass customization" of credit products. At the current time, Capital One offers somewhere around 4,000 different sets of "terms and conditions" which translate to some combination of credit limit, interest rates, fees, deposits, etc. Mass customization refers to the situation in which the vendors supply the terms and conditions.

We are seeing an amalgam of available features in web based software. MyYahoo. MyAOL. In these cases and many more, it is the user who chooses the terms and conditions. If one considers this phenomenon from a high enough altitude, it is clear that fifth generation programming languages have already arrived. What is offered to the end user is not so much a "product" in the conventional sense, but a meta-programming language through which the user can customize his view of the data. This system of choice is self customization.

The default product offerings can be thought of more correctly as sample programs that the user can run. From a management and methodology standpoint we have to consider that the focus of product testing and deployment must radically change. Since it is not possible to test all the programs that might be created, the thrust of any quality effort must go toward testing the sample programs and the hidden programming language. Anyone who has written a compiler validation suite knows that the validation suite is considerably harder to write than the language compiler itself.

Considering the complications, one need not be an expert on Gödel's Proof of Incompleteness to realize that an incredible number of unusable combinations will result from uncontrollable, unwise, and unpredictable user behavior. We are not quite at the place where self-customization has created this catastrophe, but the time is not far away. The weak security model and high degree of kludge of web-ware is the most common place where we see trouble today. The need for flexibility of access and the security of one's own and other users' financial data is a case in point.

3. Conclusion

With Y2K behind us, it is probably a good time to discuss the future developments in software project management that will allow us to cope better with the system we already have partially in place. Just because we have a self-perpetuating methodology of questionable value already does not mean that we should avoid a search for the truth!

If we look back to our Genesis text on software project management, *The Mythical Man Month* by Fred Brooks, we can note that his description of the “surgical team” points out many of the lessons that have been absorbed into the mainstream thought: the need for key documentation to be easily accessible and widely distributed, the need to have the best engineers working on the construction, and many others. However, we have made two errors of omission:

The first is the neglect of the need for the Renaissance software generalist. Project management skills should be in the general sphere of knowledge of the people now known as architects. A knowledge of what is inherently doable within the bounds of current technology belongs in the body of knowledge of the project manager. While we may have successfully factored the workload across Brooks’ surgical team, we have left the choice of a surgeon up to each organization.

The other error of omission is that we have fallen for Peter DeGrace’s “wampum theory” of software development expressed in his book *Wicked Problems, Righteous Solutions*. DeGrace’s statement is that there are many who fall for the idea that mountains of documentation can somehow be exchanged for the final product, and that the larger the pile, the better the chance of finding an acceptable exchange rate. This is not to say that the customers who wish to buy software and the businesses that wish to sell software do not benefit from some predictability in the way work is done, managed, and reported. In a volatile job market where the average tenure of employment is around fifteen months and the average duration of a project is less than six months, there is a good chance that a project manager will not last the duration of the work being done. But we cannot rely on documents alone as an insurance policy for getting the job done.

How do we change this sad state of affairs? We offer that one must learn project management a bit like one

learns to drive an automobile. Classroom instruction is valuable, but there comes a time when one must have the behind the wheel instruction, under very dangerous circumstances, but with an instructor in the other front seat. One must learn to weigh strict observance of the law against the need for one’s own safety.

Is there anything in the canon of project management to support this point of view? Yes, and once again we return to Fred Brooks, this time in “No Silver Bullet.” Brooks offers us the still controversial suggestion that raising the performance a group comes not from democratizing the training and average level of performance, but from apprenticing the best newcomers to the best experienced professionals. It is only through this system of one-on-one learning and the return of the mentor that we are likely to see the needed return of the Renaissance software generalist who understands what can be done and how to do it.

References

- Brooks, Frederick P., *The Mythical Man Month*, Addison Wesley, USA, 1995.
- Davis, Philip J., and Reuben Hersh, *The Mathematical Experience*, Houghton Mifflin, USA, 1999.
- DeGrace, Peter, and Leslie Hulet Stahl, *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms*, Yourdon, USA, 1990.

