Software Project Failure Lessons Learned

Date: Nov. 1999
From: Communications of the ACM(Vol. 42, Issue 11)
Publisher: Association for Computing Machinery, Inc.
Document Type: Letter to the editor
Length: 2,602 words

Full Text:

THIS LETTER, TRIGGERED BY Robert Glass's "Practical Programmer" column "Buzzwordism and the Epic $150 Million Software Debacle" (Aug. 1999, p. 17), aims at a more general scope than just replying to Glass's claims.

Let me start, however, with a claim I share 100% with Glass: There is a big and unfortunate chasm between academe and industry. But am I going to help him bridge the gap? Probably not. The problem is that Glass himself is not helping reach such a goal, and perhaps may be headed in the opposite direction. He claims, "I would like to believe that [this column], which may appear to be an attack on academe, is actually the beginning of the end of such attacks ..."

So who are the bad guys? Is it those academicians with no practical experience, who cheat management and even their colleagues by publishing false stories of success, who oversell their buzzwords, and who regularly fail when they are faced with the real problems?

And the good guys? Industry people who really know how to manage difficulties but don't have enough political clout to convince their managers of the risks they're up against?

To oppose Glass's story, I could offer several tales about arrogant industrial managers and engineers who blamed useless academics who wasted a lot of money or those about academic projects that achieved their intended results. But this would not help bridge the chasm. I could also describe a few cases of successful cooperation. Perhaps they might help.

Another point Glass makes, about which I am in full agreement, is his complaint against buzzwordism. I too regret that academe abuses buzzwords in an attempt to claim old ideas as new and to capture the attention of the audience in any way, no matter what the technical value of the product being promoted. I used to blame academe for borrowing such a bad attitude from commercial advertisements. (How revolutionary was Windows 95?)

What is unusual in this long-standing and somewhat tedious controversy is not the blaming of the two communities but that the criticism against academe originates within the academic world itself. Glass is certainly not alone in believing this. In general, each community defends itself at the expense of the global interest of the larger community. The academic community is an exception. At some point, this rule should be considered beneficial and a symptom of positive self-criticism. However, the real attitude of self-critical academicians is that all academicians do useless things and waste taxpayers' money. By saying this, I realize I'm blaming academicians who blame academicians, thus blaming myself.

To conclude, I believe the real message is simply that the good guys are in both communities, and the bad guys are there as well. The good guys are those ready to accept ideas and cooperation from the other side; the bad guys are those who blame the other side just to hold onto the power. The good guys should join their efforts to bridge the gap. My apologies for ending with such a trivial "non-nonconformist" claim. Sometimes the truth is not a novelty.

DINO MANDRIOLI Politecnico di Milano Italy

I'D LIKE TO POINT OUT THREE standard hazards of software projects: overestimation of competence, internal warfare, and project management that does not fit the business environment. All three spring from the immaturity of large-scale software development as a human activity; each is potent by itself, and together they are deadly.

Overestimation of competence. It is easy for a recognized expert in one field to be given undeserved credit for competence in another field. For someone who has done advanced work in software design to be given a big industrial project is like assigning the designer of the X-15 rocket plane to head up development of the Boeing 777. He or she would immediately confront unfamiliar requirements--economy issues, smoothness of flight, noise level, manufacturability, and maintainability in remote locations. The competence to meet these requirements exists, but probably not in this particular leader. Similarly, for large software systems, requirements for performance, reliability, testability, deployability, maintainability, and ability for integration with existing systems loom large. High levels of competence needed in meeting all these requirements (and the software's functional requirements as well) are not so easy to find in either academia or in industry.

Internal warfare. Sometimes internal factions are set against each other deliberately, as a way to keep both parties in line. More often it happens accidentally, and is left in place by management who sees no great harm in it. In software development, it is almost always deleterious to have warring factions. The basic reason is that each faction has an enemy--the complexity of the project--that is more formidable than any other faction, and if the various factions do not unite against this enemy, it will crush them all. In particular, specialists in meeting subsets of different requirements need to work cooperatively in order to meet all the requirements. Note that factions at war have reason to deceive each other. I leave it to your imagination to consider the effects of deliberate misinformation on a large, complex, and already fragile endeavor.

Project management that does not fit the business environment. All business efforts, software projects included, operate within a framework for allocating the business's resources. Believe it or not, at the outset, there are lots of business efforts for which it is even less obvious how things will turn out than for software projects. Consequently, the resource allocation framework is designed to funnel resources toward efforts that seem to be delivering on their potential and to kill as early as possible things that are not working. Therefore, any project has to submit to periodic measurement (often monthly) of progress, measured in nontechnical, externally understandable quantities toward fulfilling its potential.

Most engineers and almost all academicians hate this process. In fact, the only people who typically enjoy it are the executives who need it in order to do their jobs of wisely spending their companies' resources. The process is often subverted in various ways. (I once heard about some Soviet scientists who foiled the bureaucracy by submitting proposals for work already completed, thus ensuring satisfactory performance, then used the money for their next job, repeating the process.) I believe a project manager's only ethical course of action is to somehow make the allocation process work as originally intended. It does only harm to keep a failing project alive beyond its time, or to let a successful one die because its progress was underreported. This is one area where technicians and business people must work effectively across their cultural boundaries to design and execute a measurement process that actually supports wise resource allocation.

I'll leave it as an exercise--I hope an easy one--for the reader to explain CS90 in terms of these three phenomena.

PAUL ZEIGER Hurleyville, NY

I STRONGLY SYMPATHIZE WITH Glass in his excellent "Practical Programmer" column. Yet all of the project failures I have witnessed were perpetuated by people who emphatically were not academics.

Always, the real, underlying reason for project failure is a breakdown in logic. Incredibly, in a profession that uses logic as its basic material (much like the material of a blacksmith is metal), few programmers understand logic. For example, they don't know that problem domains or businesses require logical structures appropriate for a particular problem domain. The concept of inventory in a retail business is not relevant in, say, a banking application.

The answer to project failure is a better understanding of the intellectual structures required by specific problem domains. In fact, some professional programmers have begun to study these structures. The work of the pattern community is an example. David C. Hay has written an excellent book on database model patterns for specific user domains. More could be done. Most effective would be an addition to the academic curriculum that teaches explicitly intellectual models using real-world examples, tracing from problem domain to program code.

Identifying the structures of important problem domains should be the primary focus of our profession, because invalid structures, along with inappropriate or poor logic, is the main reason for project failure. Type of methodology, brand of case tool, and choice of computer language just don't matter in comparison.

DAN CHISHAM Columbus, OH

ON THE SUBJECT OF THE $15 million failure of the Westpac project, Glass fails to prove his point--that computer science concepts are incompatible with mainstream IT (Glass's "communication chasm").

However, as an industry practitioner for the past 28 years, I see nothing wrong with using object-orientation, supertypes, inheritance, and state machines in a banking system.

Glass's "recipe for disaster" can be found elsewhere. There are two clues in the column. Glass says "... there was an illusion of careful project management where none in fact existed" and "... the development methodology was unworkable." These clues would have been more than sufficient to sink a project of that size, even if the design technology centered on Cobol68. In fact, statistics from metrics experts, such as T. Capers Jones, suggest that any project of that size (huge!) is in deep trouble from the start, no matter which computer science techniques are or are not used.

I believe Glass is probably committing the logical fallacy of "post hoc, ergo propter hoc" (after this, therefore because of this). Just because a project uses advanced computer science concepts and then fails doesn't prove these concepts are the reason for the failure. It also doesn't prove they are not the reason. The burden of proof is on Glass--the person making the claim. My position is that he failed to support this burden in his column.

MARK WALLACE Los Angeles, CA

THE STORY ABOUT CS90 Is interesting, but I don't know what lesson Glass wants us to draw from it. Should academics never develop commercial software? Dave Thomas of Carleton University started OTI, an object-oriented software development company, and had a dozen large projects, all on time and on budget. Academics need to develop more commercial software, not less. One of the best ways to narrow the gap is for people on each side to experience life on the other side.

One obvious mistake is developing lots of software without any domain knowledge. Bell Labs did this in its 5ESS project, which ended up a lot bigger and more expensive than it should have been (according to people I know at Bell Labs, who might very well be biased). So, academics aren't the only ones who can make this mistake. They are probably more likely to do so than people in industry, however, since they believe they can overcome any problem.

Perhaps the biggest mistake academics make is using new technology on large projects without first using it on smaller ones. A $100 million project probably shouldn't be started with any technology unless a $10 million project has used it first, and a $10 million project shouldn't be conducted until a $1 million project has been completed. A new technology will have all sorts of rough edges. And rough edges should be rounded out on a small project, where problems are easier to solve.

Perhaps Glass had some other lessons he expected us to learn. The column's title implied the lesson was about buzzwords. But I fail to see a lesson about them. Classifying words as buzzwords is subjective. For example, articles from Microsoft seem full of meaningless buzzwords. But these words actually mean something to people who live in the Microsoft world. It takes a while to find out whether a buzzword has more than just "buzz." So, I hope Glass meant something more than "buzzwords are bad."

I bet if Glass interviewed others who worked on the CS90 project, they would have had an entirely different view of why it failed. A lot of complaints people make are merely symptoms. It is difficult to find root causes.

RALPH E. JOHNSON Champaign, IL

Robert Glass Responds:

I CERTAINLY AGREE WITH almost everything Johnson says--academics do need to develop commercial software; domain knowledge is important to project success; large projects are not the place to try out new technology; labeling things as buzzwords is subjective. I was trying to say (and apparently I didn't say it clearly enough) the key to the CS90 project is that trying to use nearly a dozen new buzzword technologies on a massive, bet-your-company-type project was clearly doomed to be the failure it did indeed become. And the project participants, especially including the academics who advocated their new technology, should have known better.

More Powerful Internet Connections

AS CHAIR AND MEMBER, respectively, of SIGGRAPH's Public Policy Committee, we read with interest Scott Tilley's "On Site" column ("The Need for Speed," July 1999, p. 23). Because the use of graphics and graphical user interfaces requires high-bandwidth Internet connections, our committee has decided to focus on this issue.

Tilley presents a reasonable introductory overview of the subject, although we probably would not include 56Kbps modems. Rather, we would include satellite and terrestrial wireless as other high-speed communications alternatives. Our version of such a survey appeared in a recent issue of SIGGRAPH's quarterly member publication Computer Graphics ("Last-Mile Bandwidth Recap and Committee Survey Activity," May 1999, pp. 49-53). The article is also available at www.siggraph.org/pub-policy/CGColumn-0599.html.

Our concern is that Tilley did not discuss a number of critical technical and public policy issues in his column. For example, between subscriber and central office, DSL has serious limitations. Most cable modems are nonstandard (newer standards have arrived late to market), and the service is a shared resource, making performance deteriorate drastically as more users sign up. Both of these technologies also have critical security problems due to their "always on" nature with a permanent IP address. And Windows users of either technology might be surprised to find their neighbors can easily access their files and even their printers unless certain parameters are appropriately set.

Additionally, numerous policy issues may influence prospective customers' decisions. Cable modem customers must use the ISP designated by their cable company. Also note that cable operators are not common carriers, like the phone companies, and customers may find the content of their transmissions monitored and perhaps censored by the cable operator and its captive ISP. DSL policy

issues include reasonable tariffing, openness of telcos' copper loops to independent DSL providers, and (for single-line homes) loop-sharing between a voice service provider and an unrelated data service provider, both using different frequencies on the same wire.

Wireless broadband users in apartments and condominiums, for example, despite recent legislation favoring them, remain limited in where they may mount outdoor antennas. Cellular and PCS providers are unable to offer broadband service due in part by the reluctance of regulators to impose uniform technical standards. (As for fiber carriers, non-conducting "wires" clash with such rules as powering users' telephones, even when commercial utility power isn't available.)

While use-oriented experience reports can be helpful, these examples hint at the complexity of a subject that deserves fuller treatment by a magazine with Communications' informed readership.

ROBERT A. ELLIS Fountain Hills, AZ

MYLES LOSCH Los Angeles, CA

Scott Tilley Responds:

I AGREE THE SUBJECT DESERVES a fuller treatment than I provided. However, I attempted to describe only my own experiences with three technologies. Since I have not personally used alternate connection methods, such as satellites and power lines, I did not discuss them. Space limitations also prevented me from covering some of the important legal and social issues Ellis and Losch mention. I welcome the opportunity to discuss them in the appropriate forum.

Please address all Forum correspondence to the Editor, Communications, 1515 Broadway, New York, NY 10036; email: crawfordd@acm.org.

Source Citation (MLA 8th Edition)
"Software Project Failure Lessons Learned." *Communications of the ACM*, vol. 42, no. 11, Nov. 1999, p. 21. *Gale Academic OneFile*, link.gale.com/apps/doc/A57746640/AONE?u=oran95108&sid=AONE&xid=fa48e95f. Accessed 1 Mar. 2021.

Gale Document Number: GALE|A57746640