Phillip G. Armour

The Spiritual Life of Projects

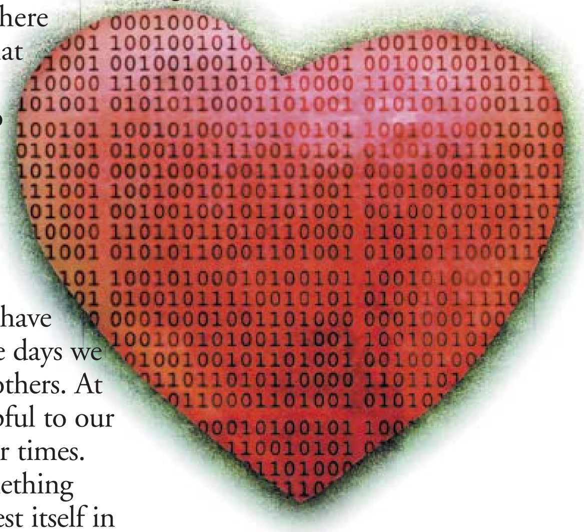
The human factor in software development is the ingredient that ultimately gives a project team its soul.

Software development is a wholly human and a largely intellectual activity. There are few aspects of creating computer systems that do not have their basis in thought, reason, and logic. We software folk are exceedingly cerebral and so tend to live north of the shoulders. But there are other aspects to building systems and working together that are not quite so literal and mechanical. There are more dimensions to being human than just the intellect, and we need to take care of them to be truly effective as a business.

A Jesuit priest, roughly 4-ft. 9-in. tall, explained these dimensions to me many years ago. Father Jim asked me: "Where does courage come from? What is compassion?" He explained that people grow in four dimensions that are somewhat independent of each other. Those four dimensions are physical, intellectual, emotional and what Father Jim chose to call "spiritual." "Before you get all bent out of shape," Father Jim said in his best (but genuine) *Bells of St. Mary's* Irish brogue, "let me explain what 'spiritual' means."

He went on to explain using some examples of the first three: running a marathon is a highly physical activity requiring a great deal of training and more than a little innate corporeal capability. Balancing your checkbook is an intellectual activity requiring abstract reasoning skills. Being angry is an example of exercising an emotion. But there are many attributes that humans have, Father Jim explained, that do not fall into one of these categories.

Courage, tolerance, compassion, and honesty, among others, are characteristics we may have or may not have. Some days we are more honest than others. At times we are more helpful to our colleagues than at other times. Helpfulness is not something physical; it may manifest itself in performing some physical act, but it is not a fine motor skill, and we can have more of this capacity sometimes than others. Compassion is not an emotion, though it is often accompanied by the emotion of sadness. Spiritual and emotional attributes are often closely associated, one of the more obvious being the relationship between courage and fear. Being courageous is not being fearless; it is the capacity to work through fear. Running a marathon might require the spiritual quality of fortitude to push through the physical experience of significant pain. Balancing your checkbook may generate real emotion.


TERRY MIURA

Father Jim's point was simple: There are capabilities we have as humans that are neither physical, intellectual, or emotional. Father Jim summarized "Let's take every other capability humans have, that doesn't fall under one of these three labels, and call them '*spiritual.*'"

We software folk are exceedingly cerebral and so tend to live north of the shoulders.

Spirit on Software Projects

So what does this have to do with software projects? As an industry, we have spent much time working on the physical and intellectual dimensions of building systems. What about the others? Specifically, what about the spirit of projects? I have never seen any research on the spirit of projects. Anecdotally, however, my personal experience has been that spirit—present or not present on a project—has been the deciding factor. I have worked on projects with an abundance of intellect and a dearth of spirit, which were relatively unsuccessful. On the other hand, I've worked on projects that did not have the strongest intellects but we had *something* that made us successful. Certainly if you asked the people who worked on those projects, they would all agree the projects were valuable and rewarding learning experiences. That in itself is worth a great deal.

We can intellectually construct methodologies and processes, we can physically locate ourselves in close proximity, or install powerful workstations, we can even to some extent, intentionally manage the emotional components of projects. But, what about the spirit? Can we manage and optimize the non-physical, non-intellectual, and non-emotional components of projects? I think so. In fact, I think this might be one of our greatest challenges as a business, and one that truly gives meaning to what we do. Personally, just knocking off another system doesn't do much for me anymore. What does matter to me is whether we made a difference. Did the system add value to lives and industries? Did we learn? Is it important? Do others think it is important? Did we grow? Are we better?

How would we do this? Here are a few ideas:

A code of conduct. I put this first, even before purpose. I have found, particularly over the last several years, that how a team behaves can be even more important than its professed goals. That is, the actual code of conduct adhered to on the project is more important, and has more impact than its roles, responsibilities, tasks, methodology, language, and process. The reason for this is simple: If the de facto code of conduct allows that people sign up to do things they have no intention of doing, it really doesn't matter what roles are assigned, they won't be done. And if the behavioral norms of the project allow people to use whatever process they care to, it doesn't matter what process the project chooses to use. The behaviors that people exhibit are more significant. It's important to note that projects *always* have a set of acceptable behaviors, whether they intentionally sign on for them or not. The code of conduct may be explicit or implicit, but it is always there. The question is does the code of conduct help or hinder the team?

A higher purpose. A definite purpose or goal is the sine qua non of a team. This goal can be mundane or inspiring, tedious or energizing. For projects to be really successful, the purpose has to be something that lifts peoples' spirits—something they can believe in. Ideally, the goal should be both shared and participative. Shared means most members of the team get a chance to help formulate it, and participative means it requires the combination of the efforts of most of the team to realize it. A few years ago, I worked with a collection of software research scientists who wanted to be a team. More correctly, their boss wanted them to be a team. As it turned out, their individual tasks had almost nothing in common. They were independent workers doing things that shared no common bond. The “team” was simply an administrative unit. We reasoned that, unless they developed a *superordinate* goal in which they could all believe, they would never be a team. Well, they didn’t, and they didn’t. Without a common, shared purpose, a team cannot be a team.

Contributing. Another element to the purpose that we have found necessary to the formation of a good team is the idea of contributing. The final result must add value to someone’s life, in some concrete identifiable way, for the goal to be inspirational. This invariably means the purpose is contributing to a higher cause, or at least a cause higher than personal aggrandizement. Incidentally, the cause of merely enhancing the profitability of a company usually fits in this category. Vague slogans such as “customer-focused” don’t work either.

Morale. It’s not just a phrase. In a number of software estimation tools, there is a separately adjustable productivity factor named “morale” or “motivation.” For example, in QSM’s SLIM-Estimate tool, increasing motivation for the development of a 10KLOC telecommunications system from its nominal lowest to its nominal highest value will increase the Productivity Index from 8.9 to 12.3. Such a project could realize a savings of $604,000 (a 65% improvement on $937K) and 2.7 months (a 27% improvement on 10.1 months) in development cycle [2]. No small change. In the spiritual life of effective projects, morale is discussed, assessed, and responded to on a daily basis.

Supporting each other. Teams are social organisms. We play a role in each other’s lives. In some cases, you might spend more waking time with your colleague than with your spouse. Project teams provide one of the major venues for interaction we experience in our workplace or even in our lives. The quid of this pro quo is that we have a responsibility to support each other. This means being able to give each other effective feedback, both positive and negative, to listen to each other, and to provide a, dare I say it, spiritual resource on which people can draw. At the lowest level, this might mean simply providing an ear when needed. At higher levels, it may mean validating the professional and personal worth of your colleagues and they of you.

Leadership. In general, leaders do not make teams. That implies a degree of control and manipulation that good leaders do not usually employ. Teams make teams. Good leaders tend to guide, mentor, assist, and challenge. One of the best definitions of leadership was given by Jerry Weinberg when he said “Leadership is the process of creating an environment in which people become empowered” [3]. What Weinberg is saying is leadership (a set of actions, not words) acts on the *environment* (not the people). It is, in essence, creating a situation where people will do what they want to do naturally, which is to work together, be cooperative, be productive, learn, and grow. Since we started by discussing the behavioral code of conduct we should note that how a leader behaves is much more important than what he or she says.

Taking care of weaker personnel. It is typical in most organizations to rate people, to assess their competence and ability. In times of economic hardship, the lower-assessed ones are those who are let go first. They are, after all, the lower performers. We usually use a bell-curve approach: a few people are really good, most people are in the middle, and a few are on the low end of the curve.

This model has some intrinsic appeal, but it is probably wrong.

In our cold, cost-cutting pursuit of efficiency and productivity, we shouldn't forget that software is only made by people. And people have a number of dimensions in which they must grow, and a variety of capabilities that make them effective and productive.

The fallacy is quite a simple one, and can be easily explained from statistics. The Gaussian or normal distribution is based on an assumption of randomly distributed values of independent variables. However, peoples' behaviors and skills are neither randomly distributed, nor independent.

At the very least people have self-selected their work—they do it because either they like it, they are good at it, or both. Any recruiter worth his or her salt should be filling vacancies with really good people. Now we could assume that the general improvement in the population due to self-selection is randomly distributed, which is simply moving the bell curve to the right. But this is probably not true, since it also likely skews the shape of the curve into a non-normal distribution. Even given this, though, the assumption of independence is manifestly wrong. The very reason we have teams and projects is because peoples' work is *not* independent. Once we lose independence of variables, the Gaussian curve goes out the window. In actual fact, low performers may serve a valuable role within teams and within organizations [1]. We could make a case for ridding ourselves of chronically low performers, of people whose motivation or work ethic persistently limits their effectiveness. But there are other reasons for low performance that are situational. Trashing someone because they do not have the skills or knowledge is not a good recipe for a healthy team or organization.

Removing someone who is not concentrating at work because of their concern about their spouse who is fighting cancer is not a good thing to do. I would contend that whatever improvement is made to performance due to the removal of that one person is more than offset by the damage done to the ethic of the team and organization. Indeed, I once worked on a project where we, as a team, took as a superordinate goal the redemption of one such chronically low performer. Not only did it help to cement the team, we did (finally) get good performance out of this person.

In our cold, cost-cutting pursuit of efficiency and productivity, we shouldn't forget that software is *only* made by people. And people have a number of dimensions in which they must grow, and a variety of capabilities that make them effective and productive.

Father Jim died a few years ago, but his spirit sure lives on in the people who knew him. For me it really showed that the business of software is also the business of people. We program machines, but we are not machines. It is the spirit that makes us different. C

References

1. Harvey, J.B. *How Come Every Time I Get Stabbed in the Back My Fingerprints Are on the Knife? : And Other Meditations on Management.* Jossey-Bass, San Francisco, CA 1999.
2. QSM Inc. Slim-Estimate (using default and typical values for project costs and other factors). McLean, VA.
3. Weinberg, G.M. *Becoming a Technical Leader: An Organic Problem-Solving Approach.* Dorset House, New York, NY (1986), 12.

Phillip G. Armour (armour@corvusintl.com) is a vice president and senior consultant at Corvus International Inc., Deer Park, IL.

© 2002 ACM 0002-0782/02/0100 $5.00