

# Projects Failures: Ignoring the Warning Signs

By

**E.M. Bennatan**

**Advanced Project Solutions, Inc.**

What do we do when evidence conflicts with our beliefs? For example, if we believe that a project will be successful, how do we respond if the project begins to fail?

I was recently made aware of the work of behavioral psychologist Leon Festinger at Stanford during the 1950s<sup>1</sup>. Festinger explained that we tend to ignore the evidence that conflicts with our beliefs (especially if we hold them strongly) until it becomes overwhelming. Only then will we change our beliefs.

Festinger's theory (he called it the *theory of cognitive dissonance* [1]) explains one of the most confounding findings we discovered from research we conducted with the Cutter Consortium. It appears that the stage of software development, in which an organization most commonly realizes that a project is failing, is the end of the development schedule. Why would an organization wait that long? Surely the signs of failure must have been quite clear much earlier.

According to Festinger, if we strongly supported the project at the start, we will be inclined to ignore the negative signs while amplifying any positive scraps that we can find, ...until the project collapses in total failure. Clearly, not all project stakeholders ignore the warning signs, but the Cutter survey found that enough do and the result is that more than half of failed projects are diagnosed very late (often too late).

In this report we will look at some very interesting information from the Cutter research concerning the way organizations respond to failing projects. We will examine their success rate in getting projects back on track and we will discuss formal project reviews and early warning systems. Finally, we will look at ways for an organization to counter our inclination to ignore the warning signs.

## **HOW DO ORGANIZATIONS RESPOND TO FAILING PROJECTS?**

We know that no one likes admitting that their project is failing (as Festinger has shown), and we will initially examine the data related to this assertion. But we will also look at some additional data that indicates that putting off dealing with failing projects does not necessarily make failure inevitable.

Figure 1 shows the stage at which organizations identify failed software projects as being in serious trouble. We see that 55% of surveyed organizations identify failed projects very late in the development cycle; of them 37% do so towards the end of the original schedule, 15% after several schedule extensions, and 3% when the project ends (either due to completion or cancellation). Only 28% identified project failures during the first half of the original schedule (of them 7% during the first quarter). The rest stated that they do not know when failures are identified (9%), or have no project failures (8%).

---

<sup>1</sup>My thanks to Capers Jones.

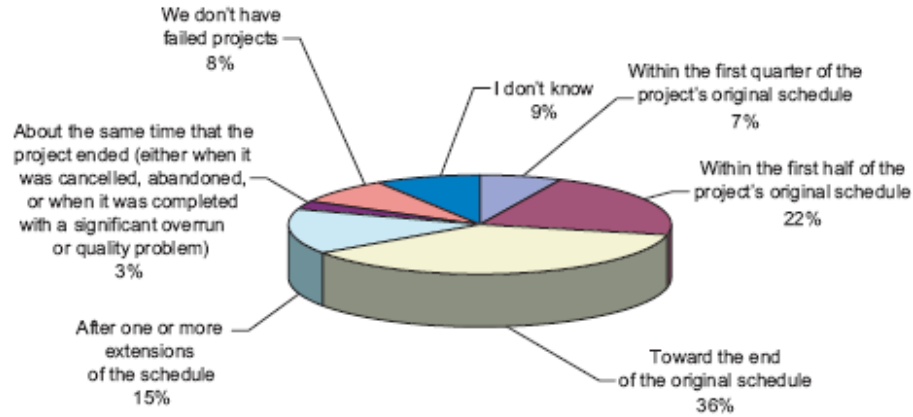


Figure 1 — In your organization, at what stage were failed projects identified as being in serious trouble?

Figure 1 copyright © 2005 The Cutter Consortium

If we remove the effect of the latent data (“no failures”, or “don’t know”), we find that about one third of software development organizations identify failed projects early (during the first half of the development schedule) and two thirds identify them late. This clearly shows that many failed projects are permitted to go on for too long. Apparently, software development organizations tend to hope that serious problems, if ignored, will eventually go away. But is this attitude really as destructive as it seems? Let us look at some more data.

Figure 2 shows that 45% of software development organizations almost always manage to get seriously troubled projects back on track. Another 30% say that they manage to do so sometimes (some are saved and some end up failing) and only 9% report that projects can rarely or never be saved. If we consider the 45% (“almost always”) and the 30% (“sometimes”) findings, we can speculate that about 50% or more of failing projects are saved. These numbers do not describe an ideal situation but they are certainly not bleak either.

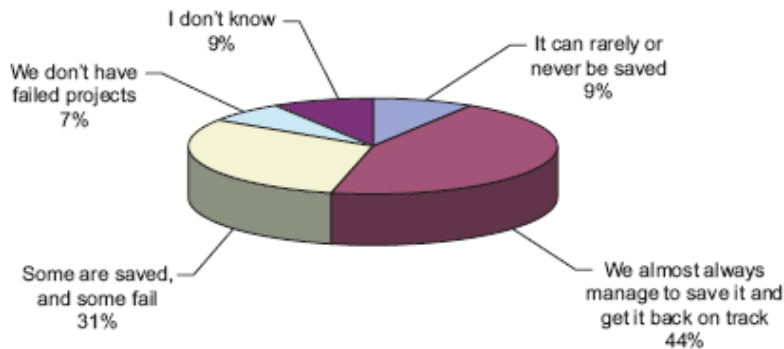


Figure 2 — In your organization, when a project is recognized as being in serious trouble, how likely is it that the project will get back on track (i.e., how often does your organization prevent the project from failing??)

Figure 2 copyright © 2005 The Cutter Consortium

So, it appears from the survey data that when a software project is in serious trouble it has a somewhat better than even chance of recovery. How does this finding fit in with the data in Figure 1 (failed projects go on for too long)?

The fact that a failing project was saved does not mean that it was rescued efficiently. While a failing project may plod along for quite a while before a rescue process is initiated, a more successful approach may have uncovered the problem six months earlier and saved the organization six months of development costs (not to mention frustration, poor moral, and missed market opportunities).

In the period between the appearance of severe project problems and the realization that the project is headed for failure, how do organizations respond?

Figure 3 provides us with much of the answer. When projects appear to be in danger of becoming failures, two of the most common responses are the demand for more hours from the team (57%), and the assignment of more people to the project (38%). But more than three decades ago the futility of these solutions was first documented (see, for example, Brooks [2]). Overtime and more people may sometimes be effective for peaks in the development cycle (such as during testing), but they rarely save a project that is in serious trouble. In such cases, the project needs a formal rescue process [3].

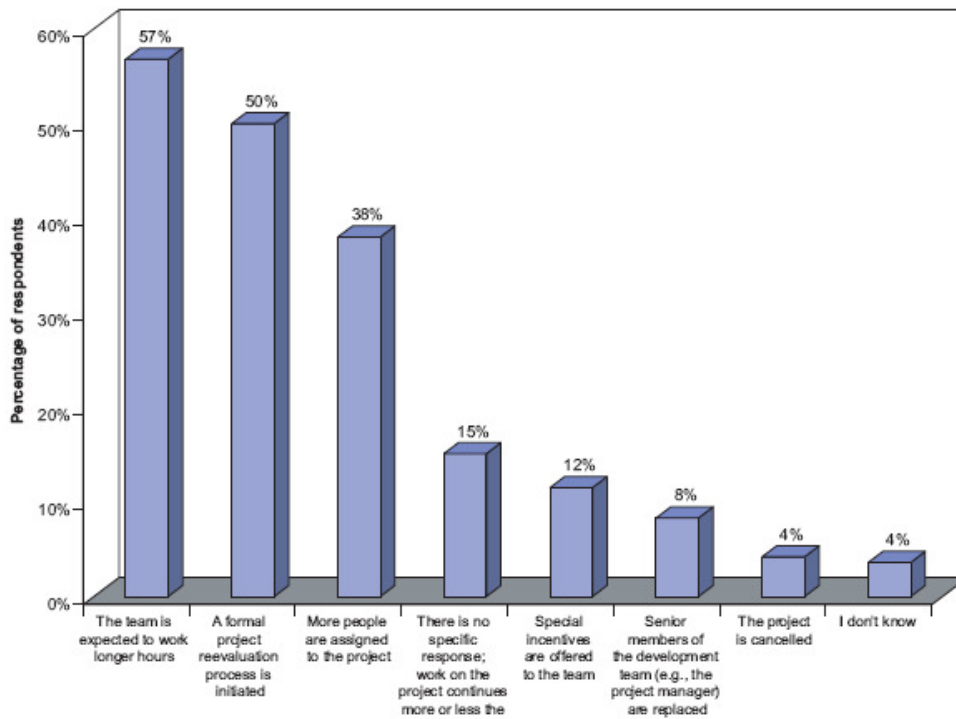


Figure 3 — When software projects appear to be in danger of becoming failures, how does your organization respond? (Please select all that apply.)

Figure 3 copyright © 2005 The Cutter Consortium

The good news is that when companies recognize that they have a troubled project on their hands, 50% of development organizations do launch a formal rescue process. This figure helps explain the high project rescue success rate.

Other less common responses by development organizations to failing software projects include special incentives to the development team (12%), replacement of senior staff (such as the project manager) (8%), and cancellation of the project (4%). Among the rest, 15% say there was no special response, and 4% do not know how their organization responds.

## COUNTERING FESTINGER'S SYNDROME

There is rarely a single activity that determines software project failure or success; the outcome is usually the product of a combination of several factors. But one activity that strongly influences the outcome is the practice of conducting regular formal reviews. It is like taking frequent breaks during a long car trip to check your progress, review the map, and update your travel plans. It gives you an early warning if you are straying off course.

Almost all surveyed organizations reported that they conduct some type of project review (in other words they all look at the map occasionally). Slightly more than half (51%) conduct regular formal reviews; 38% at least once a month, and 13% at least every three months. Another 11% hold occasional formal reviews (not on a regular basis). Informal project reviews (no fixed structure or format) were reported by 30% of the respondents; 16% conduct them regularly and 14% occasionally. Just 8% state that they do not hold project reviews at all.

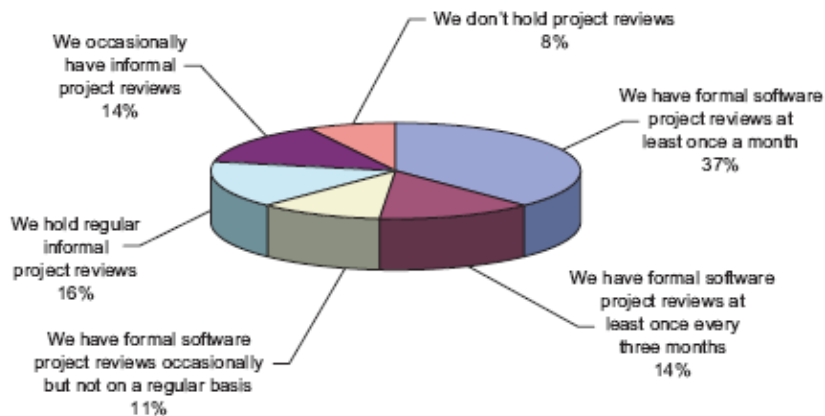


Figure 4 — Does your organization hold regular software project reviews with senior management?

Figure 4 copyright © 2005 The Cutter Consortium

These findings are interesting because they correlate well with the data in Figure 1 where we found that 34% of software development organizations identify failing projects within the first half of the development schedule (this is the figure that we derived by removing the latent data from Figure 1). We can safely speculate that there is significant overlap between these 34% and the 37% that conduct formal project reviews at least once a month (according to Figure 4) because frequent formal reviews increase the likelihood that serious project problems will be uncovered early. This can be an excellent remedy for Festinger's syndrome; it provides an early warning system based on an environment where data is evaluated and decisions are made in a more objective manner.

## IGNORING THE WARNING SIGNS

The survey data certainly seems to indicate that Festinger's theory is valid in software project development. The fact is that severely troubled projects are dealt with very late. Whether this is primarily because development organizations ignore the warning signs due to cognitive dissonance or because they have no early warning system in place is uncertain, but undoubtedly both phenomena contribute to the predicament.

While dealing with psychological issues in software development is an interesting challenge it is certainly not an easy one. At Advanced Project Solutions, we recommend dealing with failing projects with the installation of the early warning system, as discussed earlier, based on frequent formal project reviews, and by applying an organized process to get the troubled project back on track. This requires the collection of data, the installation of a warning trigger, and follow-up procedures. For software organizations that do not have a formal project review process, the practice can be started quite modestly with a relatively simple process that can be refined and expanded over time (see [4]).

Readers who would like to comment on this discussion or on their experience with software project failures and overruns are invited to e-mail me at: [info@AdvancedPS.com](mailto:info@AdvancedPS.com).

This article was originally published by Cutter Consortium as part of its Agile Product & Project Management online resource center. For more details see <http://www.cutter.com/project.html> at [www.cutter.com](http://www.cutter.com)

## References

1. Festinger, Leon, Theory of Cognitive Dissonance, Stanford University Press (June, 1957)
2. Brooks, Fredrick P., The Mythical Man Month, Addison Wesley, 1975
3. Bennatan, E.M., *Catastrophe Disentanglement: Getting Software Projects Back on Track*, CrossTalk, The Journal of Defense Software Engineering, Vol. 17, No. 10, October 2004. (You can download this article free at [www.AdvancedPS.com](http://www.AdvancedPS.com))
4. Bennatan, E.M., *Catastrophe Disentanglement: Getting Software Projects Back on Track*, (Chapter 12), Addison-Wesley, April 2006.