



---

## WHY SCHEDULES SLIP

by Lynne Nix, Senior Consultant, Cutter Consortium  
e-Project Management Advisory Service

It was my pleasure to be the guest editor of the Cutter Information Technology Journal, December 2003 and March 2004 issues. The topic of both issues was, “When to kill an IT project.” As a project management consultant, this is a topic that has always been of interest to me. It is amazing how many organizations continue to invest in poorly conceived projects. As a consultant, I am often asked to do a project review. When things have not gone well, the stakeholders are looking for recommendations to get their project back on track. When projects get into trouble, it is often for one (or more) of the following reasons:

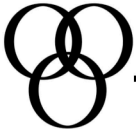
- Lack of a clear vision or business case
- We don’t know what we don’t know
- Scheduling without specs and late changes to specs
- Schedules based on desire, not reality
- Inadequate resources
- Cross-product dependencies
- Vendor dependencies
- Inadequate tools

### **Lack of a Clear Vision**

Bright ideas alone are not enough for project success. The project’s business case should reflect a compelling, understood, and agreed upon business need. Think about the number of projects you’ve seen that have begun as nothing more than a verbal request with little or no analysis behind it. The project vision (or charter) should articulate why the project is being undertaken and explicitly note what the organization expects to achieve, the rationale for any deadline, who the target customers are, who the competitors are within that market, and what is and is not going to be included in the scope of the project. If this foundational material cannot be articulated in one to two pages, the project will be under a cloud of uncertainty from the beginning. However, the project team will go forward with enthusiasm and will compensate for the uncertainty by making assumptions that often go untested until it’s too late.

### **We Don’t Know What We Don’t Know**

This is especially true in new product development. The problem occurs when we don’t admit that there may be key areas of knowledge, business climate factors, and other important items, about which we don’t know. And, to preserve our egos we lead stakeholders to believe we know what we’re talking about. One way to



---

accommodate uncertainty that is inherently unknowable is to adopt a “feature driven development” approach. By producing a very preliminary functional spec at the beginning with the most important features being detailed and developed in the earlier iterations of development, feature-driven development allows the project team to adapt as better information becomes available. Feature-driven development couples with risk management and schedule buffer to allow teams to make mid-course corrections while managing stakeholder expectations.

### **Scheduling without Specs and Late Changes to Specs**

These are really one in the same problem. Schedules without specifications are pure fantasy. A project schedule reflects the work to be done as outlined in the specification. Specifications always drive the schedule, not the other way around. When I see a lot of late changes to a spec, it usually indicates not much time was spent on the spec and what looks like “changes” is really the spec being created. Specs are often compromised in order to get developers working on the product. The spec serves as an implementation guide for development and test. At a minimum, the development iteration at hand should have a clear spec to guide the work of the developers and testers.

### **Schedules Based on Desire**

Too often when a project is begun, the only thing known is the deadline. The definition of the rest of what is to be done is pretty much content free. While there is nothing inherently wrong with an aggressive release date, it must also be realistic. The time to talk with stakeholders (usually marketing) about unrealistic deadlines is after the team has had time to assess what can and cannot be accomplished. This means after some preliminary spec and schedule development has been done. It is much more helpful to tell stakeholders what they can have by when than to commit to a fantasy project doomed to failure.

### **Inadequate Resources**

Organizations, like individuals, do not function well under extreme schedule pressure. One of the ways this manifests itself in organizations is in assigning the same individuals to multiple priority one projects. Using the shared resource approach, instead of getting the most important project completed and out the door first, that project is doomed to be completed at the same time (or later) than the least important project. Remember, there can only be one “priority one” project.

### **Cross-product Dependencies**

At a strategic level it makes economic sense to utilize products, or their components, that are being developed by different teams within the organization. Why should each group re-invent (re-invest) the wheel? Unfortunately, this is easily said



---

and hard to implement. Product groups usually have different release schedules, different target customers, different competitive demands, and when asked to deliver a product internally for use, guess what the lowest priority is for that group? That's right: delivering the product for internal use, especially if that need is seen as financially less important to the organization than the external demands. Many things are required to allow cross-product work to operate more smoothly, but fundamentally management must set cross-group priorities and ensure that rewards are consistent across efforts to complete internal and external products.

### **Vendor Dependencies**

Have you ever experienced vendors not delivering on time, not delivering what was specified, bringing their technical leads to the initial meeting then staffing the project with less experienced staff. Why would responsible vendors do this? Often it's because there is no compelling legal reason to do otherwise. Take a look at your contract. Have you specifically stated your review and acceptance authority regarding the initial and subsequent vendor staff? The same due diligence should be spent finding and qualifying vendors as the organization spends in recruiting and hiring full time staff. And that includes the contract. Too often, in a rush to bring a vendor on board, little time is spent putting the specs into the contract and defining the terms of that business relationship. To avoid later difficulties, spend time specifying the deliverables, defining unambiguous acceptance criteria, determining how and when payment is to be made, stating your review and acceptance authority over interim and the final product, and vendor requirements for correcting errors to the product after release. While a meeting of the minds is key to a productive vendor relationship, you need to get it in writing.

### **Inadequate Tools**

Don't get me wrong, it's rarely all a "tool problem," but trying to develop a product with tools that are also under development is very difficult. If new technologies and non-existence tools are a requirement for the release, your back-up plan should include a release with the old tool set or make sure the tool developers are part of your project team.

It's a "pay me now, or pay me *a lot more* later" issue. If you make the investment of time and effort to create a well-defined project at the outset, you won't be spending time killing the project later, and most importantly, you won't be writing off the money you invested in a "doomed to fail" project.