

---

## Core Value 5 — Fear of Flying: A Story of Failed Scope Control

---

I fly a lot. In the consulting business you need to visit your clients wherever they may be. As a matter of fact, I am writing this from seat 39E, somewhere between Chicago and Atlanta. As a frequent flyer, it is a bit nerve-wracking to think about the 1960, IBM 9020E computers supporting air traffic control for the airports in Chicago, my hometown. The basic problem is that these computers, supporting air traffic control for the last 30 years, frequently break down. As I think a bit more about it, it is more than nerve-racking, it is absolutely baffling that our government could fail to upgrade these mission critical computers in 30 years. How could this happen?

Simply put, the states of the air traffic control systems in this country are the same as the state of many new client-server applications: poorly estimated, out of control, and out of scope. More system implementations fail due to a lack of system estimation skills, project management skills, and scope control than any other factors. Over the past ten years a number of efforts to replace the aging air traffic control system have failed because the scope of the project has grown beyond the ability to build a new system. This has led to a failure to deliver even a replacement built on hardware from this generation that is easier to maintain, and more reliable than the current software. Let us analyze the source of these failures.

Our first reaction with many failed projects is to blame external factors. We may blame our failures on the users - they keep changing their minds. We blame our failures on the technology - too many bugs in the tools we selected. Or on management - they did not give us enough time. However, we eventually realize that all of these problems are caused by a failure to match the project deliverables with the resources available. Projects fail because scope has not been properly defined, estimated, and controlled.

Controlling scope is not easy. Often, a user, or manager, or our customer comes to us and asks "Can we have a system with these features, in this amount of time" and the easiest answer to give the user is, "Yes." After saying yes a number of times and failing to meet commitments, we often switch to another approach. We say no. Saying no is not any better than saying yes. The correct approach is to manage the definition, estimation, and scope control processes.

To manage these processes, one must begin by understanding the dynamics of building systems. It was relatively easy to understand how programs worked when computers were first introduced. Most programs contained hundreds, or at most, thousands of lines of instructions. Large systems were divided into a number of linear, top-down, smaller programs. A simple flow chart could be used to diagram the system and each program within it. In the late sixties and early seventies, programs became more complex making it much more difficult for any single programmer to understand all of the code in a system. Increased complexity led to difficulties in estimating how long it would take to complete the design, coding, and testing of a new program. For example, while building the IBM OS-360 operating system the project's leadership coined the phrase, "The Mythical Man Month," to describe the team's

frustration with the process of estimating software development. We are all aware of the many falsely publicized delivery dates in the personal computer industry that have led to the term, "vaporware." Vaporware is software that has never been released to the public or seen by a real user. Although it is not easy, we can avoid measuring our estimates in mythical man months and having our software classified as vaporware by adhering to the following steps.

The first step in delivering software on time is to accurately estimate the initial effort. This is not trivial. By controlling the initial software development estimates and the number of additions and changes that included in the effort, scope can be controlled. By controlling scope we increase our ability to deliver systems on time. Over the last twenty years a number of techniques have been developed to help with this process. All of these techniques can be distilled into three basic procedures: partitioning, abstracting, and project management.

The first step to estimating a new project is to understand it. The size of the project you or I conceptualize depends on individual experience. For a new software developer, a project consisting of more than a couple of modules may be intimidating and too big to estimate. Where an experienced project leader may be comfortable estimating a system with hundreds of modules. To accommodate the experience level of the estimator we need to *partition* the system into understandable sub-systems or modules. The size of each sub-system should be based on our own personal experience. Do not try to estimate a sub-system that is bigger than the largest system you have previously worked on. Any size system can be divided into sub-systems of a size that an estimator can conceptualize. If you find yourself paralyzed by the fear of failing the next time your manager asks you to estimate a software development effort, begin your task by breaking the system down into components you can understand. Components of a size you have built before.

After partitioning the system, do not attempt to design every detail of every sub-system before continuing with the estimate. Instead, *abstract* each module by viewing it from fifty thousand feet. For example, if a sub-system is going to be made up of 200 screens, do not attempt to understand the details of each screen. Rather, divide the screens into three classifications: simple, moderately difficult, and complex. Simple screens may use data from a single table. Moderately difficult screens may include two to five tables but limited logic. Complex screens may include numerous tables and hundreds of lines of business logic. Apply abstraction again. Abstract your past experience gained when you implemented screens for other systems. Estimate, or calculate, if you have a good project tracking system. Determine how long it has taken you in the past to implement a typical simple, moderately difficult, and complex screen. With these historical averages in hand, and the number of screens of each type, you can create an estimate with simple multiplication: multiply the number of screens in each classification by the average time it took to complete a screen in the past. With completed estimates, you can now begin project management and scope control.

The art of *project management*, of which scope control is a critical component, can be viewed as the art of list management. I do not mean "lists" as in "linked lists." This is not a technically complex task - I mean lists of things to do; lists of things to remember; lists of people to talk to. To control the scope of your project begin creating lists. One of the first lists that should be created is the project change control list. By

documenting and estimating each proposed change, the burden of change control will shift from you, the project manager, to the users.

For example, a user approaches you three weeks after the completion of the functional design phase of your project and asks, "Can we make a minor change to the system? I'm sure it will only take a few minutes." Your initial response should be, "Tell me a little more about what you want to do." First, reiterate what you heard to the user to ensure that you clearly understood the user's request, and then say, "Let me look into it." Never say "Yes" or "No" on the spot. The next step is to put the change request on your change control list. Write down the user's request and then apply the estimating techniques I just described. Break the request down into chunks you can abstract, compare to past efforts, and estimate. Next, add the task estimation information to the change control list. After these techniques are applied, you are ready to respond to the user's request. For each change request repeat your understanding of the request, how long it will take, and what it will cost. Do not forget to include in your estimate, project management time, testing time, documentation time, and risk. More often than not, when presented with complete information your user will control scope for you.

In conclusion, to properly complete any complex project problem, partitioning and abstraction need to be applied to the estimating process. To control scope, apply your new estimation skills in conjunction with list management skills, and your chances of a successful project will considerably increase.