CHAPTER 1

Software Development Explained: Creativity Meets Complexity

There are shelves of books and hundreds of thousands of articles dedicated to making software development better. Why has it been so hard for smart professionals to just make software projects run smoothly, on time and on budget? What's up here?

A Definition of Software Development

Software development is any activity that involves the creation or customization of software. As noted in the introduction, it can include

- Launching websites
- Installing a CRM (customer relationship management) tool
- Implementing a new accounting package
- Building a custom application for your business

All these activities qualify as software development. Most businesses will, at some point, be confronted by a software development project. Technology is now so intrinsically integrated into business that it's impossible to avoid.

Why Is Software Development So Difficult? Hint: It's *Not* Like Building a House

A lot of people use the metaphor of house building as a comparison for the activity of software development. I believe this metaphor does an enormous

disservice to the process. I reject this metaphor because it gives people a false sense of security and a false understanding of the nature of software development.

A house is concrete and well understood by all. We have all been inside houses. We all share comparable assumptions about what a house is. The same cannot be said for software. In many cases, I have sat in a room with people with completely divergent views about even the most basic aspects of their software project.

Frequently, in developing software, you are creating something from nothing. That means the end product could be practically anything. Here are some endeavors in which, as in software development, you are creating something and the outcome could be a wide range of possibilities:

- Writing a novel
- Growing a garden
- Composing a symphony

There is a "blank slate" quality to creative activities. When you begin a novel, for example, the end product could take an almost infinite variety of forms. The listed endeavors, you'll notice, are all open to a wide scope of interpretation. One person's assumptions regarding the nature of a garden (flowers) may not remotely match another person's (vegetables). So it is with software development: You might think that the parameters of your project ought to be "obvious"—but they may not be obvious to your colleagues.

The previous examples all capture a fundamental reality of software development. By its very nature, software development is creation. You're going from a state where something doesn't exist to one where it does. At the beginning, the outcome could be anything, which means that everyone in the room probably has a different understanding about what the project actually is.

The Simple, the Complicated, and the Complex

In *The Checklist Manifesto* (2009, Metropolitan Books), a book I highly recommend, author Atul Gawande talks about three types of endeavors—the simple, the complicated, and the complex. It's helpful to understand these distinctions because software development almost always involves all three:

- Simple project components are easy to conceptualize. You know what needs to get done, and you simply need to get out the elbow grease and do it.
- Complicated project components are hard to understand and involve a lot of steps, but they are not very risky. If you read the directions carefully enough and follow them, you will get the project done.

- Complex pieces of projects, on the other hand, have a lot of variables like the complicated, but they are also highly fluid and very risky.

As already noted, software projects almost always involve all three types of activities: the simple, the complicated, and the complex. The percent mixture of each depends on the project.

I use three of my own metaphors to describe the simple, complicated, and complex as they relate to software development. Mastering these three metaphors and learning to apply them in software will help you manage any software project more successfully.

Metaphor #1: Piles of Snow

"Piles of snow" is the phrase I use to describe the *simple* activities within a software project.

Imagine a massive snowstorm blew through overnight. You wake up and your 200-foot-long driveway is completely blanketed in white. Worse, the plow guy called to say he can't make it. The city plow comes through and there is now an even greater pile at the end of your driveway (Figure 1.1).

What you need to do is absolutely clear. Get a shovel and dig! How to do it is also no mystery.

Keep in mind *simple* doesn't mean *easy*. In fact, most of the time a lot of labor is involved in piles-of-snow software tasks. It's going to be a lot of work

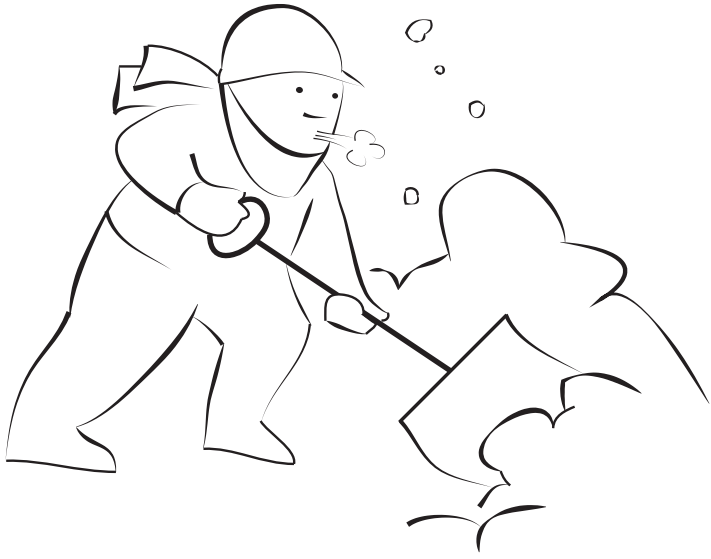

Figure 1.1

to shovel that driveway, especially if there's no one to help. But the "project" is simple in concept and in execution: Dig until you are done.

I will return to the "piles of snow" metaphor again and again in this book.

Metaphor #2: The Ikea Desk

"Ikea desk" is a term I use to describe *complicated* aspects of software projects.

Furniture from the Swedish store Ikea comes boxed up and involves seemingly millions of little pieces (Figure 1.2). The directions are expressed solely in pictures, presumably because it's more efficient to do it this way when you serve an international customer base. Imagine the effort to translate all those directions into hundreds of languages!

To begin your Ikea desk project, you must lay out all the tiny pieces on the floor and match them to the illustrations in the directions. Then you must meticulously follow the directions. There is frequently trial and error. Wait. Was that the part? It doesn't seem to fit. No, I think this one is the right screw. It's the smaller-than-middle-sized one.

Anyone who's put together an Ikea desk remembers a weekend spent on the living room floor before the furniture piece is finally ready. It can be frustrating. It's time consuming. You may be missing a part and have to go back to the Ikea store and wait on the customer service line. Despite all this, success is virtually guaranteed. With enough time and Allen wrenches, you will get it done.

Figure 1.2

Ikea desks require more application of brainpower (e.g., reading and interpreting detailed directions), more concentration, and more backtracking and redoing than shoveling piles of snow. Further, the time it will take to assemble the Ikea desk may be harder to judge than the driveway shoveling. You may say something like, "I'll have this baby assembled by noon," only to realize you put the wrong screws in and assembled it backwards. It's more like midnight when you actually finish. But in both cases, a successful outcome is 99 percent guaranteed.

Metaphor #3: Heart Surgery

The Checklist Manifesto uses heart surgery as an example of a *complex* activity, and it works well to describe aspects of software development.

To perform heart surgery, you absolutely need extensive training and skill. Further, your patient might have an undiagnosed medical problem that causes the operation to unexpectedly fail. The human body is not 100 percent understood by anyone. It is squishy and organic and unpredictable. The fact is, no matter how much you plan, certain variables are out of your control (Figure 1.3).

Unlike Ikea desks or piles of snow, heart surgeries are unpredictable and risky. Training and experience helps, but even with all that, you have no guarantee that the surgery will succeed.

Hint: Finding a surgeon with experience in risky heart surgeries does help.

Figure 1.3

Using the Three Metaphors in Project Management

Ideally, your project would involve many piles of snow, a moderate number of Ikea desks, and as few heart surgeries as possible. You would strive to reduce complexity in your software to the degree possible. In many cases, this is the best approach and we'll continually return to it throughout this book.

But sometimes the choice isn't up to you. Your project may simply have complex elements you can't avoid. Many, if not most, software projects do.

Furthermore, to say something is "complex" or a "heart surgery" is just another way of saying it's risky, with many unpredictable and unknown elements. This brings us back to the concept of creativity, discussed at the beginning of this chapter. Creative endeavors involve, by definition, lots of unknowns. One of the main reasons people undertake software projects is, indeed, to be creative and to come up with something new. That's what we call innovation! And innovation is a key business goal for many organizations. Is there a business that does *not* wish to pursue innovation?

The problem comes in when people sign up for complexity in an unconscious way. They never look at the pieces of their project and evaluate which are snow piles, Ikea desks, and heart surgeries. Worse, believe it or not, some people seek out complexity. Why would someone do that? Signing up for complexity and risk sounds crazy, right?

One only needs to drive on the freeway for a reminder that human beings often and consistently seek out risk. Software development is usually exciting to business stakeholders, who may spend most of their lives in dusty old Excel spreadsheets. The chance to do something creative and techy brings out the risk taker in many people.

The reality is that most people, even programmers, do not analyze the projects they undertake in terms of the simple, the complicated, and the complex. They sign up for a project and simply accept, "It is what it is." However, especially in the first phases of your project, it is essential to identify what is a pile of snow, what is an Ikea desk, and what is likely to be a heart surgery. Then you can make informed decisions on project approach, planning, staffing, and budgeting. You can make considered choices about where you can and cannot reduce complexity and where you consciously wish to sign up for complexity to achieve your business goals.