My first exposure to the term configuration management—well, the first exposure I was required to notice—was early in my career when I was managing software tools, porting, and acquisitions for several divisions of a major test instrumentation company. In this case, a group of software development managers came to me and said, “We must do configuration management. Find us a configuration management tool.” I thought for a moment and replied with the first question that came to mind. In retrospect, this question has turned out to be one of the most profound of my career: “What is configuration management?” Their reply was equally profound: “Er . . . uh . . . we don’t know, but we think it has something to do with RCS.”

In the years since that seminal exchange, I have observed that when people say they “want to do configuration management” or even say “we are doing configuration management,” they often have no idea what configuration management is, or what gain they should be deriving from its use. Why, then, would they want to do configuration management if they don’t know what it is or why they should use it? The answer lies, I believe, in human nature. When walking in uncharted territory over complex terrain, our survival tends to depend on our actions being under control. In that respect, the desire to do
configuration management can almost be categorized as instinctual. This notion conforms to my experience and tends to explain the behavior of those original managers. They knew they were dealing with new product development that included the often-overwhelming complexity of software development projects. They instinctually felt that they needed to control what was going on, even if they couldn't articulate that need.

What follows, then, is the definition of software configuration management (SCM) that I like the most: “Software configuration management is what you do to control the evolution of software projects.” It is a simple definition, and others are certainly more detailed, but the one I like is accurate, easy to remember, and easy to articulate.

Some years later, I found myself in a job with Atria Software that allowed me to deploy the ClearCase product in many different organizations. Each organization I visited suffered to some degree with the problem illustrated by that question-and-answer exchange I first experienced those many years earlier: “What is configuration management? We don't know, but we think we need to do it.” In every case, these organizations were already doing some form of software configuration management, usually using a simple version control tool such as RCS or CVS, but they really couldn't say why they were doing it or what benefit they were getting from their practice. In every case, though, they had identified that their current way of doing software configuration management was not providing the level of control required to deal with the complexity of the development they were doing. They knew instinctually that they needed more control but were largely unable to articulate exactly what kind of control or exactly what they expected to gain from that control.

I found that part and parcel of each implementation was a need to describe the kinds of things that a good software configuration management scheme would allow you to do. I found it also useful to note that just because you can do something doesn't mean you have to. A good SCM system should allow you to tailor your control rather than dictate it to you. This ability to tailor your SCM to suit your organization, however, brings with it the need to know not only how the SCM
system works, but also what suits your organization. The second part of that equation, defining how your organization wants to work, is actually the harder part.

In working through these deployments, I found that the following principles were reliable guidelines:

- **If you can't articulate your vision, no one will want to follow you.** This is important if you want to gain advocates for the changes you desire in your organization. These advocates can then win others over, saving you much time, effort, and frustration. It all starts with your knowing what you want to do and being able to explain it clearly (assuming that what you want to do is a good idea in the first place!).

- **There's no substitute for knowing what you're talking about.** This means not only knowing what tools you want to use and the technical details of how they work, but also the more difficult task of understanding how your organization currently works: what works well, what is broken, and how your plan offers a net gain for the organization.

The other thing I learned in these deployments was that beyond the haze surrounding the somewhat theoretical discussion of what SCM is and what you should get from it is the part where you have to make it work. So, to the two preceding principles, I add this one:

- **At the end of the day, your solution must work.** Your solution must function technically, but it also must be useable. I’ve seen many organizations spend tens or hundreds of thousands of dollars implementing an SCM system only to discover, after it is rolled out and developers start using it, that it just doesn't work. Either the system itself doesn't work because of technical flaws, or it is so cumbersome and painful to use that developers rebel and won’t use it—or worse, sabotage its use.

  The deadly temptation that leads to this disaster is usually trying to do too much. Remember, just because you can do
something doesn't mean you have to. I've found it best to keep SCM simple at first and then iterate into complexity rather than trying to engineer a complex and arcane SCM scheme before anyone gets to try it.

This book is grounded in these fundamental principles. First, the authors are explicit about what software configuration management is and what you should get from it. Second, they are clear about the need to articulate that vision, the techniques you'll need to use, and to whom you should be prepared to articulate your vision. Finally, this book isn't just a theoretical treatise on software configuration management principles; it also has the “make it work” piece. *The Art of ClearCase® Deployment* is appropriate for the individual or the organization new to deploying ClearCase, as well as for those who have done it many times before. The extensive hands-on, how-to information on managing a ClearCase deployment project is material I wish I had when I first started deploying ClearCase, and I still find it valuable today.

*Tom Milligan*
*IBM Rational Software*