To cope with the mounting demands for large, complex information-system applications, most organizations turn to a systems development methodology. SDMs are logically appealing, offering a flexible framework for the sequence of tasks needed to develop an application, as well as tools and techniques for accomplishing these tasks.

By creating an engineering-like development discipline, SDMs provide explicit deliverables and consistency as information systems are built. In its intent, an SDM should reduce the risk associated with shortcuts and mistakes and ensure that quality infuses the software process. On the surface, at least, it would seem that every organization should have an SDM in place.

Unfortunately, most IS organizations fail to successfully implement and utilize an SDM. The question is, why?

**WHO ARE SDM IMPLEMENTATION STAKEHOLDERS?**

SDM stakeholders are people with a vested interest in a successfully implemented methodology. They include functional and IS managers, systems personnel, external consultants, and end users. Each stakeholder views SDMs from a different perspective. To successfully adopt an SDM, all stakeholders must be convinced that the SDM

- has a relative advantage over the existing norms offered by no methodology;
- is compatible with the organization and its structure, culture, and skills sets; and
- is not overly complex (thereby generating more costs than benefits).

Because stakeholders are often unaware of an SDM’s benefits and pitfalls, training at all levels is necessary. Without training, any attempt to change work and management styles, job descriptions, procedures, roles, and responsibilities—all required when implementing an SDM—is doomed to failure. In all cases, people must change the way they do their work. But changing the existing norms and work habits that have been institutionalized through repetition and a shared common interpretation is extremely difficult.

**WHAT FACTORS ARE IMPORTANT TO IMPLEMENTING AN SDM?**

In a survey of potential SDM adopters that my colleagues and I conducted (“Factors That Influence System Development Methodology,” IEEE Trans. Software Eng., Vol. 24, No. 8, Aug. 1998, pp. 640–649), we asked what factors they perceived as important to implementing an SDM. The following summarizes the results:

- Understanding methodology specifics and benefits. Stakeholders needed information to understand the new SDM life cycle, activities, techniques, application of techniques, and supporting tools and their benefits. To reduce resistance to change, SDM expertise can help company management sell the need for the SDM to system and functional personnel.
- System personnel manager involvement with and responsibility for organizational SDM transition. Clearly, the transition from the old SDM to the new one represents cultural change. Stakeholders must understand what is new, what is different, and how these innovations and differences will lead to direct personal and organizational benefit.
- Functional manager involvement and support. Functional managers should be involved in an SDM implementation’s initial planning and provide resources for completing the SDM implementation project. Success demands their commitment to the new SDM’s implementation and use. These stakeholders should also develop ways to measure the implementation’s progress.
- External support. Because the personnel in
most companies typically do not have the knowledge or experience to implement an SDM, external consultants can provide significant assistance. External support is typically needed in planning and completing the SDM implementation, developing training courses and materials, training personnel on various aspects of the new SDM, and assessing the implementation’s success.

Use of models. Most SDMs use modeling methods. An important part of using an SDM is knowing how and when to use modeling methods, how different models relate to each other, and how end users participate in creating the models and keeping the models integrated.

That our early adopters identified these success factors for a new SDM’s implementation is really not a surprise. Unfortunately, when probed for specifics, the survey’s participants indicated that their SDM implementation efforts fell short with regard to all of these factors. Stakeholders revealed a wide gap in what they believed should occur and what was happening in practice. The real question becomes, “If you know what you need to do to implement an SDM, why are you not doing it?”

WHY ARE IMPLEMENTATIONS SO DIFFICULT TO ACCOMPLISH?

As I’ve said, SDMs represent cultural change—and cultural change is difficult. Historically, organizations combat resistance to change with education about the new SDM’s benefits and capabilities. My survey found, however, that stakeholders already understood the importance of implementing a new SDM and its benefits. Still, the implementation process was not going as they expected. There was a huge gap in what they believed they should be doing and what they were doing to implement the new SDM. With this gap in mind, I’ll suggest a few causes.

Technological dynamics

Hardware, communication, and software capabilities, as well as software development tool resources, are changing so fast that organizations do not know which process to adopt. Implementing an SDM usually takes at least two years—an eternity in today’s technological time frame. Most stakeholders won’t commit to a process that might be obsolete before implementation is complete.

Companies suffer from something akin to attention deficit disorder when implementing SDMs. As technologies change, companies jump to the next SDM before completely and successfully implementing the original. Because systems stakeholders know about new technologies and processes, they hesitate to fully commit to the SDM, preferring rather to look over the horizon to what’s coming next month or next year.

Backlog

Given the two years to implement a typical SDM, the organization wants to help itself in the long run with process-improvement projects such as SDM implementation, but is too busy meeting short-term requirements to tend to these types of projects. Organizational defense mechanisms add to this strained situation, making it extremely difficult for organizations to do what they know is best for their long-term survival and success. Short-term project demands dictate a different path.

Espoused theory

According to propositions derived from the theory, individuals have and use two theories of action: the ones they espouse and those they actually use. Espoused theories are the values, beliefs, and attitudes individuals express and give allegiance to when questioned. However, the one that actually governs their actions is the theory-in-use, which might or might not be compatible with their espoused theory. So, stakeholders are really “talking a good game” with regard to implementing an SDM. While everyone says they want to implement the new SDM, the commitment to the new process is missing. Stakeholders espouse the politically correct answers, but follow up with little or no action to implement the SDM.

Freedom of movement

When implementing an SDM, the fear is not the change itself, but the addition of controls to the development process. An SDM imposes checkpoints and controls visible throughout the development effort, which ensure that commitments are met. Developers often complain that methodologies are confining, stifle creative thought, and do not fit the way systems are actually developed. These sentiments—conveyed to the other constituents with comments about how their positions would be diminished through the SDM—could be informally communicated and undermine the implementa-

Continued on page 75
stantly be updated on milestones in achieving that goal. Identified measures of success are critical for any successful improvement program.

A software executive must also be willing to make the tough calls, whether that means being late on a deliverable to meet quality criteria or making changes in the team. I also believe that it is important to walk the halls, talk to team members, listen to what is going on within the organization, and actively participate in quality activities.

What would you have done differently?

Ahuja: In hindsight, we know that there were a number of detours we took along the way. If we had focused on predictability earlier, we could have shortened the road to Level 5. We had some very long debates over the need for predictability in the software development process. We decided to take some other steps first. We should have attacked this area earlier than we did. It would have been hard, but the rewards would have been worth the effort.

What things would you like to predict right now?

Ahuja: Of course, I'd like to be able to predict during development if a project is going to be on time and if it is going to be within cost. But most importantly, can we predict the quality that a product is going to have when it is delivered? Predictable processes take much of the mystery out of development. What shocks me now is that in our industry, good predictive practices are the exception. They should be the norm.

What advice would you give to people who want to become software executives?

Ahuja: My advice is very basic. Be true to your commitment—your commitment to your employees and to your customers. When you empathize with your customers, you will not ship poor quality, and you will deliver on time. If you care for your employees, you will create and manage a work environment that is conducive to success. Have a thorough understanding of the business you are in and provide leadership. Businesses fail not because employees don't work hard or don't want to excel. They fail due to poor leadership.