Modeling in Moderation

Process Innovations

June 2004

Tom Davenport

President's Distinguished Professor of Information Technology & Management
Babson College

tdavenport@babson.edu

I thought enterprise modeling was dead. I thought I'd seen the last of huge, detailed, comprehensive—and unwieldy, incomprehensible, and obsolete—models a few years ago. Enterprise data modeling, I hoped, was a stone-cold corpse on the floor. But its leg just twitched. The idea of enterprise modeling seems to be semi-respectable again—if not for data modeling, at least for process modeling. I hear people talking about it with a straight face. I see vendors selling product to support it. Even enterprise data modeling seems to be discussable in polite company again.

Let's review a little bit of history. The real heyday of enterprise modeling was the late 80s and early 90s. Companies came to believe that the best way to identify systems requirements was to model both as-is and to-be process and data flows. The idea of doing this for individual systems evolved into doing it for an entire organization. Enterprise architects built enterprise data and process models that were supposed to yield common data, integrated systems, and efficient cross-functional processes across the enterprise.

Nice idea, but it didn't happen. A few things got in the way. First, these enterprise modeling efforts took so long that by the time they were finished (if they ever were), the organization would have changed. Secondly, they were seldom finished, because the executives who sponsored them seldom saw sufficient value to fund them to completion. Thirdly, the complexity of these models made it very difficult to even discuss them intelligently with non-technical managers—which in turn pretty much precluded the possibility of their affecting any real business activity within the organization. Finally, because analysts usually found the as-is processes to be offensive to a rational mind, they were seldom documented. Instead the focus was on how the processes and data would flow in a more rational to-be world. But these models were seldom realized in actual systems and processes, which meant that there was rarely any guide to where data could be found within a company or how work was really being done.

However, these problems didn't stop organizations from devoting billions of dollars and millions of person-hours to enterprise modeling. After all, it was easier to create detailed models than it was to create real change within organizations, so modeling became a popular pastime. The United States Defense Department, for example, spent almost a decade (and many millions of U.S. taxpayer dollars) on data and process modeling, and at the end had very little to show for it.

If the problems I described above didn't totally stop enterprise modeling, what did? Three letters: SAP (and, to a lesser degree, Oracle and PeopleSoft). It proved to be a lot easier to implement an integrated set of systems from a vendor than to harmonize legacy systems through enterprise-level data and process modeling. Of course, SAP came with a set of reference models, and one could buy more detailed industry-specific models from organizations such as IDS Scheer. This helped some organizations to figure out just what SAP was really doing, and how best to configure it to meet specific needs. Again, it was a lot easier than building your own models.

Now, don't get me wrong. I am not an opponent of modeling in general. In fact I'm a supporter. But like sex, drugs, and rock and roll, modeling is best done in

Process Innovations

by Tom Davenport

June 2004

moderation. Should you model a to-be process before custom-building a system to support it? Yes, absolutely. Can you profit from customizing a reference model of a process that's supplied with a package? Almost surely. Should you do a quick as-is model of a process before reengineering it? Indubitably—otherwise you won't understand how and why the process flows today, and you won't know by how much you've improved it. Is it useful to do a high-level model of a process before you even think about what system will support it? In most cases, yes—it will help in discussing systems support with process owners, and will probably assist in determining systems requirements.

Some valid applications for process modeling have little to do with systems. These other areas for which high-level process modeling is useful are in facilitating process governance, measuring process performance, and capturing process knowledge. It can be useful in discussing "who owns what" from a process ownership standpoint to have available high-level models of the processes involved. For organizations that want to measure process performance, some of the more sophisticated process modeling and management tools now allow the ongoing monitoring of process performance indicators in a graphical-display modeling format. And being a big fan of knowledge management, I also think it's a great idea to capture what we know about a process—benchmarks, ideas for improvement, workarounds, etc.—along with the graphical depiction of the model.

But there are other circumstances in which the virtues of modeling are more questionable. Should you create an as-is model of a process you have no intent of improving? That's probably not worth the trouble. How about a to-be model of a process to which you're going to apply a package? There is probably little benefit in doing this if you're not going to customize or heavily configure the package.

Doing an enterprise-wide model, whether it's of processes, data, or both, is almost certainly a bad idea. You will probably lose your job before you are finished, much less before the benefits of such an activity would be realized. You won't change anyone's behavior, and you won't know how your processes actually flow or where your data actually resides.

Aside from limiting the scope of modeling, what else can organizations do to improve its chances of success? The most important principle is to make the models sufficiently simple and straightforward so as to facilitate dialogue with business executives. Keep the models to one page whenever possible. Use the language of the business—avoid jargon and computer-oriented terms like "cust_no." Don't agonize over what tool, icons, or modeling language to employ—they are not the key issue. The key issue is business change and how best to accomplish it. Detail and technological complexity are the enemies of that change.