




BPM

Related Features

[Assimilating BPM: Col. Sanders' Secret Recipe](#)

[MyBPML.org](#)

[BPI Carving Niche in Insurance Industry](#) 

Related Webinars

[Turning up Business Velocity: Competing on Time with BPM](#)

[EAI Solutions for New Millennium Challenges](#)

[Empowering the Enterprise with Simply Smart BPM](#)

[All Related Content](#)

Why BPML?

03/29/2004

By John Hamilton, Senior Architect, Process Infrastructure, Computer Sciences Corporation

Page 1

The wisest guy I ever met in information technology was in charge of implementing one of the very first online database systems here in the UK. This was in the days before anyone used SQL which was, as we all knew, a wholly abstract concept, incapable of practical use. This guy had a theory about databases. He believed that data would rot. Really. He thought that if you stored data and didn't use it, it would rot away and contaminate other data.

The extraordinary thing about this is that he was right, and it applies to just about anything you store and don't use (we'll discuss the relevance of mothballs later).

If you don't believe me, just bring to mind all those data dictionary systems you've seen implemented, at fabulous cost, and never used. This is so common there is even a name for it: shelfware. Hands up, anyone who can honestly say they've never seen a pile of shelfware. I'm just as bad as you: I have a contact file a mile long and I know that I'll never use most of the contacts in it, but I haven't the nerve to get rid of them.

What is forgotten is just how pervasive this problem really is. A major manufacturer of silicon things spent many, many millions of dollars documenting its processes. It has 18,000 process diagrams. On the shelf. They've rotted away, because no one actually uses them.

If you want to store anything-- data, process diagrams, contacts, you've got to take it out and exercise it regularly-- "use it or lose it". Anything that is not in regular use pretty soon becomes unusable.

Now, I got interested a little while back in the upcoming fashion of business processes. It rapidly became clear that there were a great many people, and quite a few systems, that loved to do process documentation. (You've met these people. They all discuss theory at interminable length. They are the inheritors of the arguments about how many angels can dance on the head of a pin.) Most of them are safe in the knowledge that whatever mistakes they made could never be discovered because their documentation would rapidly become shelfware. Meanwhile, in the real world, real people were continuing to do what they had always done: largely ignoring the rulebook and adapting to changing

circumstances.

The “use it or lose it” rule applies here. Process documentation that can be ignored, *will* be ignored, by people wishing to get on with their work. What I wanted was some kind of process documentation that actually interacted with the real world. It appeared the other day: a formal language that was complete and sufficiently unambiguous as to be executable. This meets the “use it” criterion. If the documentation not only can be used, but has to be used, then it will become and will remain correct. A closed loop feedback mechanism applies here. If a tool or method is necessary for real work, then it will be maintained in good condition: Carpenters keep their chisels sharp. In real life, most people do not use documentation and do not care if it is wrong. They use tools and get mad as hell if they don’t work.

So if we now have technology available that will enable us to document, execute and, most important, maintain our business processes, what criteria should it meet? I use the words: formal, open and complete. Any language needs to have some formal basis, some calculus that underlies it. Otherwise there is too much potential for ambiguity. It needs to be open: I don’t want to invest my intellectual property in a proprietary system that locks me in to a supplier. I also remember the great success of Linux and the open source project. Finally, if it isn’t complete, it is obviously going to be useless anyway.

This is why I think BPML has a future. It enables us to create tools that not only document business processes, but also execute them. It meets my criteria for tool selection, and it enables us to bring process theory off the shelf and into the real world, where it belongs.

IN PART TWO: John gets personal about BPML

About the Author

IBM recruited John straight from college. He worked for them for years as an engineer, salesman and staffer. When IBM sold its Federal Systems Division, John was a lead software engineer on a major defense program. Later, he joined CSC, where his focus has been BPM. He attended Queens’ College, Cambridge, England, where he toured with the Footlights and with the Marlowe Society and was awarded a degree in Natural Sciences. He also has interests in programming languages and is familiar with a large set, ranging from assemblers, through high-level languages such as Basic and Java, to more esoteric ones such as APL and REXX.

[More by John Hamilton](#)

About Computer Sciences Corporation

Founded in 1959, Computer Sciences Corporation is a leading global IT services company. CSC’s mission is to provide customers in industry and government with solutions crafted to meet their specific challenges and enable them to profit from the advanced use of technology.

With approximately 90,000 employees, CSC provides innovative solutions for customers around the world by applying leading technologies and CSC’s own advanced capabilities. These include systems design and integration; IT and business process outsourcing; applications software development; Web and application hosting; and management consulting. Headquartered in El Segundo, Calif., CSC reported revenue of \$13.8 billion for the 12 months ended Jan. 2, 2004. For more information, visit the company’s Web site at www.csc.com.

[Feature Archive](#)