



| Related Features   |  |
|--|--|
| <a href="#">Assimilating BPM: Col. Sanders' Secret Recipe</a>  |  |
| <a href="#">BPI Carving Niche in Insurance Industry</a>        |  |
| <a href="#">Understanding Transactions At Different Levels</a> |  |

| Related Webinars   |  |
|--|--|
| <a href="#">Turning up Business Velocity: Competing on Time with BPM</a> |  |
| <a href="#">EAI Solutions for New Millennium Challenges</a>              |  |
| <a href="#">Empowering the Enterprise with Simply Smart BPM</a>          |  |

[All Related Content](#)

## MyBPML.org

04/05/2004

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

Page 1

The most embarrassing moment in my education was when I realized that I would never be able to understand the Hamiltonian. Or even what it was for. There it was, named after me (sort of) and it was on the other side of a wall I could never climb. Mind you, I did learn enough of the practical sort of mathematics to realize that you could do some useful stuff with it. I also learned that there was a lot more than the Hamiltonian on the other side of that wall. All that useful stuff taught me the value of symbolic representations, formal methods and so on: that they can be used to do real work in the real world.

The other day I encountered the calculi of business processes: the  $\lambda$  one and the  $\pi$  one. For me, just like most everyone else, they're on the other side of that wall. However, I picked up on the idea because I remembered the value of formal methods, particularly when dealing with computers and real world problems. As an aside here, I really like Java and, especially when you include the class library, it has to be one of the least formal languages. You may be able to do a lot, but even its best friends wouldn't call it concise. One of the most concise languages is APL: 136 primitive operators and no precedence, let alone input/output. But, returning to the point, the purpose of formality in language is the ability to produce a minimal complete system of representation.

Minimal because, like most of humanity, I'm thoroughly lazy and don't want to have to understand multiple ways of doing things (the Windows class libraries spring to mind here). Not only is there more to learn, but you then waste even more time trying to choose the best way of doing things. Whatever it is. Obviously, it also has to be complete. Why invest all that learning effort only to find that you can't use it as you want to. So my ideal system of representation has a formal basis and is minimal and complete. Incidentally, the world's most successful language, English, is neither formal not minimal, but is certainly complete.

There's another attribute I desire. Remember the success of SQL: a public domain standard. Remember the success of the PC: horrible design, but published and therefore copied, to the detriment of (so I'm told) a manifestly superior one. So if I'm going to invest my time and effort, I want to use a public domain standard. (I make exceptions with things like Java, which is almost public domain.) The lessons of the Open Source projects are obvious and relevant. They are the only real threat to the dominance of Microsoft. They

are successful because a vast number of people, who care, are involved. The sheer amount of intellectual horsepower available within the open source community can, and is, beating anything that any single supplier can deploy. So, in my opinion, if and when a public domain standard becomes supported by the open source community, it will be unbeatable.

Of course, you've guessed I'm talking about the present state and future prospects for BPML. I like it because it has a formal basis in calculus. I like it because it is complete and in the public domain. I'll like it even more when it is adopted by the open source community. All those in favour: please email me: [infavour@mybpml.org.uk](mailto:infavour@mybpml.org.uk), or [infavor@mybpml.org](mailto:infavor@mybpml.org) - I'm looking forward to hearing from you.

### **A Look Back At [Part One](#)**

#### **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](http://www.csc.com).

[Feature Archive](#)