



XML Broadly Connecting Canadian IT managers through Career, Industry, and Technology insight

This Blog

About

Email

Feb	March 2006			Apr		
S	M	T	W	T	F	S
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Search
 Go
Archives

- March 2006 (12)
- February 2006 (41)
- January 2006 (26)
- December 2005 (26)
- November 2005 (8)
- October 2005 (6)
- September 2005 (2)
- August 2005 (7)

News

These postings are provided "AS IS" with no warranties, and confers no rights. You assume all risk for your use.

» Blogs that link here



Canada's Association of Information Technology (IT) Professionals

Resident Bloggers

Stephen Ibaraki
Technology Journalist I.S.P.,
DF/NPA, CNP

John Oxley
Director IT Pro Evangelism
Microsoft Canada

Barnaby Jeans
IT Pro Advisor
Microsoft Canada

Guest Bloggers

Val Matison
CIO
Info3

GOT A QUESTION?**Curing Software Engineering's Chronic Concerns: Part 1/3**

How can we make your work as IT Managers and IT Pros easier? One answer: by collaboration and sharing the expertise of top-ranking professionals. I was talking with the noted computer science authority, Paul Bassett recently about dramatically increasing productivity, and he agreed to provide this three part blog series on "Curing Software Engineering's Chronic Concerns."

However, before I share his first post, **here is some background on Paul:**

Paul G. Bassett has given keynote addresses around the world, and was a member of the IEEE's Distinguished Visitor Program from 1998 - 2001. He was General Chair for the Symposium on Software Reuse held May 18-20, 2001, held in conjunction with the International Conference on Software Engineering.

Ed Yourdon called Paul's book, Framing Software Reuse: Lessons from the Real World (Prentice Hall, 1997) "the best book about reuse I've seen in my career." DeMarco and Lister republished his 1987 IEEE paper, Frame-based Software Engineering, in their compilation of the 27 most significant papers of that decade. Paul also co-authored the IEEE's Standard P1517: Software Reuse Lifecycle Processes.

Paul has over 35 years of academic and industrial software engineering experience. He taught computer science at York University for seven years, co-founded Sigmatics Computer Corporation and Netron Inc., two on-going software engineering companies, and for over twenty years he helped governments and businesses (a partial list: the US and Canadian federal governments, The Hudson's Bay Co., IBM, Fiserv, TD Bank, Ameritech, Union Gas, Teleglobe Insurance, Noma Industries) to improve their software development tools and techniques. He has a M.Sc. (U. of Toronto Computer Science), and is a CIPS information systems professional (retired). He is currently a senior consultant with the Cutter Consortium <http://www.cutter.com/index.shtml>.

Paul received the Information Technology Innovation Award from the Canadian Information Processing Society (CIPS) for his invention of frame technology. He later co-chaired their Certification Council; was a member of their Accreditation Council, and now helps to accredit honours programs in computer science and software engineering, as well as chairing the CIPS Committee on Software Engineering Issues.

Curing Software Engineering's Chronic Concerns – Part 1/3

Since our industry's inception over 50 years ago, software systems have been notoriously late, over budget, and of mediocre quality. Already low, programmer productivity has actually worsened since "object oriented programming" went mainstream[[1]]. In what looks like a desperation move, organizations have been in a race to replace their expensive local software talent with cheaper offshore labour. Yes, this tactic can save money in the short run. But offshoring does nothing for our industry's chronic inability to deliver quality results in a timely fashion. Indeed, it could well make matters worse.

Clearly, something is wrong with this picture. We don't import food from countries that till their fields with water buffaloes. Why? Because modern agribusinesses have made the transition from being inefficient craft industries to producing cornucopias of low cost, high quality food. The software industry can and should do likewise. Why? Because, as we shall see, a key enabler – so-called frame technology (FT) with its associated processes and infrastructure – has been fulfilling this dream in diverse commercial settings for more than twenty years! What is wrong with this picture is our penchant for rejecting ideas that challenge deeply held beliefs, even when those ideas can cure chronic concerns.

I'll illustrate by challenging a cherished myth of our craft: only human programmers can modify software intelligently. The fact is, people are poorly suited for making technical code changes, especially when the changes are both numerous and interrelated. This is precisely the situation that pertains to the cascades of program changes that are logically entailed by altered requirements. It should come as no surprise that machines can excel at fussy, detailed symbol manipulation. Several frame technologies[[2]] now exist that are designed to formalize software adaptability. FTs modify software much faster than human programmers, and they do so tirelessly and without error, not to mention much more cheaply than the off-shore variety. What sounds too good to be true is that software's entire life cycle benefits, not just construction. But more on this in a later blog.

An analogy helps explain how a typical FT works: Imagine a generic fender, one that can be automatically fitted to any make or model of car while it is being assembled. Such adaptability, applied to each type of part, would eliminate combinatorial explosions of part variants – conventional manufacturing would be revolutionized. Alas, such parts cannot exist in the physical world. But such adaptive components, aka frames, are not constrained by the physical world, and thus possess an amorphous and adaptive quality that makes perfect sense in the abstract world of software. Each frame can arbitrarily adapt any number of component frames, and conversely, be adapted by other frames. FT assembles each software product from its frame-parts in a manner analogous to conventional manufacturing, but because FT is not constrained by the physical world, it may exploit adaptability far beyond what is possible with physical parts. It is a technology with demonstrated revolutionary potential. Stay tuned; my next blog will reveal how you use a FT to start that revolution.

Navigation

[Home](#)
[Photos](#)

Post Categories

[Adam Cole \(2\)](#)
[CC Blogged Down \(5\)](#)
[Events \(6\)](#)
[Guest Bloggers \(17\)](#)
[Industry Perspectives \(13\)](#)
[Interviews \(7\)](#)
[Mitch Tulloch \(2\)](#)
[MS News \(10\)](#)
[Partners \(1\)](#)
[Stephen Ibaraki \(31\)](#)
[Training \(3\)](#)
[Val Matison \(3\)](#)

[1] According to project auditor Michael Mah of QSM Associates in a private communication to me.

[2] For detailed comparisons of several FTs, see http://www.informatik.fh-kl.de/~eisenecker/polite/polite_emtech.pdf and <http://www.cs.york.ac.uk/ftpdir/reports/YCS-2003-369.pdf> Two FTs I know well are a free-ware FT called XVCL (XML-based Variant Configuration Language) <http://sourceforge.net/projects/fwvcl> and Netron-Fusion (www.netron.com), a commercial toolset that includes a FT.

Paul, we look forward to Part 2 of this series...

Thank you,
Stephen Ibaraki

Published Wednesday, March 08, 2006 10:22 AM by [cdnimgor](#)
Filed Under: [Stephen Ibaraki](#), [Guest Bloggers](#)

Comment Notification

If you would like to receive an email when updates are made to this post, please register [here](#)

You can also stay up to date using your favorite aggregator by subscribing to the [CommentRss Feed](#)

Comments

Interesting Finds

Thursday, March 09, 2006 5:48 AM by [Jason Haley](#)

Curing Software Engineering's Chronic Concerns: Part 2/3

Friday, March 10, 2006 11:19 AM by [Canadian IT Managers](#)

See part 1 on this valuable series posted here March 8th. Here's the next post from Paul Bassett...

What do you think?

Title *(required)*

Name *required*

Your URL

Comments *(required)*

Remember Me?