

Re: J4 – presentation/discussion on "Future of the COBOL Standard"

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2008-03/msg00215.html>

- *From:* "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 12 Mar 2008 11:48:14 +1300
-

"Jeff Campbell" <n8wxs@xxxxxxx> wrote in message
news:1205259923_658@xxxxxxx

Pete Dashwood wrote:

"Rick Smith" <ricksmith@xxxxxxx> wrote in message
news:13tbq3hb852cjc9@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"Pete Dashwood"
<dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in
message
news:63jqf7F27lussU1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[snip]

Despite the claim of billions of lines of
existing code (dubious at
best;

it

has been eroded yearly for the last 5 years
(at least...) at a rate of
millions of lines every year, by replacement
with packaged solutions,
refactoring, and migration to Java and other
solutions...), even the
most
optimistic observer cannot see an expanding
future for COBOL as a

procedural

paradigm based language in a world that is
increasingly more visual and

Re: J4 – presentation/discussion on "Future of the COBOL Standard"

more

non-procedural.

Just a couple of notes from an "optimistic observer". <g>

Other viewpoints always welcomed by me... :-)

"COBOL (COmmon Business Oriented Language) is the programming language most widely used for commercial and administrative data processing." -- Micro Focus LRM, probably from the COBOL 85 standard.

Well, as it is the core of their business, they WOULD say that, wouldn't they? :-)

I don't believe it is even true today, but I can't prove it so won't argue it.

The most common paradigm for "commercial and administrative data processing" is "clerks performing procedures on or with data". COBOL came into existence as a domain-specific language for implementing that paradigm.

Not exactly as I recall... and I was there :-)

[snipped]

Regardless of the implementation paradigm (procedural, OO, functional, etc.) the result will necessarily reflect the underlying procedural basis for the required data processing. I suspect that a great deal of programming with OOPLs is procedural programming; but incorrectly claimed to be OO.

Maybe, but that is an academic argument. It doesn't matter what you call it, the fact is that the new languages can be written quicker, re-written quicker, require much less code, and have facilities that COBOL simply doesn't. Whether they are doing procedural processing under the guise of OO or not is immaterial; they can be generated to do it quicker than a good COBOL programmer can code the thousands of lines it takes.

I have been astounded at the encapsulated functionality in languages like

Re: J4 – presentation/discussion on "Future of the COBOL Standard"

C#, that simply isn't available in COBOL. It doesn't even matter what the paradigm is, it is quicker to point and click in Visual Studio, than it is to write hundreds of lines of COBOL in ISPF. End of story.

Let me second or third or whatever that! 8-)

Thanks for your support, Jeff :-)

I discovered yesterday that list boxes in .NET store objects rather than just strings (strings are objects). That means I can store an array of records encapsulated as object instances in the list box. When an item in the displayed list is selected, the code processing the click event has direct access to the objects contents. No searching or sorting!

A PowerCOBOL list box ,at least as of version 5, only stores text, meaning that code needs to be written to find the corresponding record in a table. Indices need to be managed in a synchronized fashion in the list box and the backing table. Insertion, deletion, modification routines, etc. More code to write, debug and support.

The really cool thing about storing objects in the list box is that the objects do not have to all be of the same type. I'm not sure how often this will be useful but I can see having a list box that presents the user with a hierarchy of choices the choices relating to different types of things.

This is a good example, but it is just one of many...

Facilities like code reflection, generalized procedures through delegation, access to event models, simple building and processing of collections, access to a host of components that run across platforms and save incalculable amounts of time, are just some of the facilities that COBOL simply doesn't even begin to address...

For batch processing, it doesn't matter; beyond that, it certainly does matter...

Re: J4 – presentation/discussion on "Future of the COBOL Standard"

Re: J4 – presentation/discussion on "Future of the COBOL Standard"

Pete.

—

"I used to write COBOL...now I can do anything."

.