

Something has to be tested and maintained was Re: GoTo in Java

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2006-10/msg00240.html>

- *From:* Clark Morris <cfmtech@xxxxxxxx>
 - *Date:* Sat, 28 Oct 2006 02:18:44 GMT
-

My comments on the following from months back are top posted with nothing interspersed. As someone who is a modifier and reworker of existing systems, I won't really learn or understand Java or OO until I work with it which at my age may be never. However, I strongly believe that we in the field have failed to understand that scripts are code, control parameters be they sort control statements or table entries in a package are code and in fact anything used to control the manipulation of data is code. The systems that you describe in several of your postings scare me in many ways. I hope that whatever tool or tool set was used to create these applications with just drag and drop combining of components also produced the documentation to say what was done so that when the business changes some poor soul has a fighting chance of figuring out where to make the change(s) to what was done. I also hope that the development tools contain adequate means to handle the needed testing and not just unit testing.

The use of control parameters, control tables, etc. is powerful and allows the use of generalized packages such as operating system of choice or affliction, SAP or Oracle Financials, word processing packages, etc.. While some of these are bounded, others especially in house systems have no checking on them. Older portions of operating systems can be equally vulnerable to loose change mechanisms as I can testify from experience on IBM's MVS. I have nothing against languages and tools that operate at higher levels of abstraction. At least in theory they can be expressing things in ways that are closer to the description of the process to be implemented. The use of documented components is also a good thing and something that I am used to as a former systems programmer although not with some of the niceties of the OO paradigm. If these tools also are more self-documenting, especially to non-programmers, that too is something I applaud.

However the idea of something being unchanging and never needing upgrade is not in keeping with the world as I know it. Determining the pervasiveness of a change and whether it should affect all processing is something that probably has not been well thought in the past. I am used to a mainframe environment where a subroutine change is picked up by all users (COBOL programs compiled with the DYNAM

Something has to be tested and maintained was Re: GoTo in Java

NOT maintain source code. (In any language...)

Because of its COBOL and procedural programming roots, this forum is excited by language discussions and the problems attached to source languages. Yet I see people every day, with no knowledge of ANY programming language (apart from scripting, which is not maintained in the traditional sense of code maintenance) producing really impressive applications that run on the web, access databases, share information, and produce reports.

Component based development means you write code once and do NOT maintain it. (I know there are many places that don't adhere to this and they try and maintain the components. There is only marginal gain over a normal source code maintaining shop if you do this.) The idea is to decompose functionality to lower levels than has traditionally been the case, write flexible generalised solutions to these low level functions then leave them alone! All you have to do is build the components into applications. You should never change a component once it is written and its methods, events and properties are established. If you need to 'enhance' it write a wrapper that interfaces to it and extends it for that specific case, or write a new component.

The true benefits of code reuse are not gained if you keep rewriting the code. Encapsulate functionality and reuse it. Do it once; do it right.

Fortunately, modern environments are so component rich (even though many people do not realise the thousands of components that are provided free with your OS) that people are starting to simply plug things together and achieve impressive results. Scripts do not require thousands of lines of procedural code to achieve what WOULD require thousands of lines of procedural code, and maintenance is not a problem.

The rule of source code is over. The future belongs to object code. It will be developed (once :-)) in whatever language is most appropriate for it and things like ILs and CLR's will level the playing field. I have one application where components written in VB, C++, and PowerCOBOL all play together nicely on a single windows screen. The user is unaware of what the functionality is written in (and doesn't care), and I have NEVER changed these components in any way. (I couldn't if I wanted to, because I don't have the source for some of them...) This application has been live for several years, has had enhancements made to it, and has had some minor changes to its functionality. New screens have been added and some of the components mentioned appear on the new screens as well (there is only ever one copy of them in the system). The less source code you have to look after, the less risk to your investment.

Maintaining source code is yesterday's technology.

Having said that, I do realise that 'today' we have a huge investment in procedural source that HAS to be maintained so I don't think this discussion is pointless...:-)

Something has to be tested and maintained was Re: GoTo in Java

But I also think people need to start thinking 'beyond the square'. Today's kids are doing it. In 50 years the idea of sitting down and 'programming' a computer line by line, will be a quaint curiosity, like us using flint tools or speaking Latin in the Supermarket.

Pete.