TITLE: "IS SOFTWARE HARD?"

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@attglobal.net
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"Systems are logical, programming is physical."
- Bryce's Law

For something that is supposed to be "soft", software exhibits some pretty "hard" characteristics. The original premise behind the COBOL programming language was to devise a language that could be easily ported to several computers. As we all know, this never truly happened due to computer manufacturers who tweaked the language to suit their particular needs. What ran on an IBM machine, for example, didn't necessarily run the same on Honeywell, UNIVAC, or the rest of the BUNCH. Consequently, software developers had to maintain different versions of source code to suit the particular needs of the various computer compilers. This plagued all third generation languages until Sun introduced JAVA in the 1990's. The JAVA premise that a programmer should *"write once, run everywhere"* was the right idea and the language began to gain momentum, until it ran into Microsoft who didn't want to turn the operating system into an inconsequential afterthought. JAVA lives on, but not to the extent it should have, and developers are back to managing separate versions of source code.

The point is, software does in fact exhibit some very "hard" characteristics as it is married to the host computer configuration making it not very portable. As mentioned, this creates headaches for those of us, particularly commercial software vendors, in terms of maintaining consistency in the different versions of our products.

What to do?

Back in the 1970's and 1980's we were faced with the dilemma of managing a single product on over a dozen different platforms. We quickly came to the realization we would go stark raving mad managing multiple versions of source code and came to the conclusion we had better come up with a solution pretty quick. Because of our experience in converting software, we became well versed in the nuances of the various compilers and devised a Repository (we called it a "filter program" at the time) which maintained the rules of the various compilers. We were also very disciplined in writing code to specific standards and embedded certain switches in the base source code. When we were ready to produce a new release of our product, we would feed the base code into our "filter program" which would then create the different versions of the source code ready for compilation. This saved us an incredible amount of time and brought consistency to all of the versions of the product. In other words, our programming staff worked with only one set of code (not multiple variations), the "filter program" then analyzed it and created the necessary permeation for a targeted platform. As compilers changed, we would update the "filter program" accordingly.

We also learned to maintain print maps, screen panels, messages and help text separate from the source code, which greatly enhanced our ability to create a new version of the product to suit a foreign language and culture; see *"Creating Universal Systems"* at:

http://www.phmainstreet.com/mba/pride/isspus.htm

Let us take it a step further, for years we have touted there are logical and physical dimensions to Information Systems. Using the "PRIDE" Standard System Structure concept in "PRIDE"-ISEM, we look upon Systems and Sub-Systems (business processes) as logical constructs, and Procedures and Programs as physical constructs. Further, data components such as inputs, outputs, files, records and data elements can be specified logically and implemented physically many different ways. Let me give you an example; back in the 1980's one of our "PRIDE" users (a large Fortune 500 electronic conglomerate) bought into our logical/physical concept and decided to put it to the test. Working from their corporate offices, they designed a complete Payroll System which they wanted to implement as the corporate standard across all of their divisions and subsidiaries. They completed the system with a recommended programming solution they wrote themselves (no packages were used) which I believe was an IBM MVS solution using COBOL. However, they recognized early on this implementation wouldn't work across the board in the company. Consequently, they gave the system specifications to all of their divisions who would then program it themselves in-house. The project turned out to be a major success and the company ended up with multiple implementations of the same system under IBM MVS, VM, Honeywell GCOS, UNIVAC Exec, HP MPE, DEC VAX/VMS, and Prime; all working harmoniously together.

(continued on page 2)

IS SOFTWARE HARD?
(continued from page 1)

Other "PRIDE" users experienced similar successes, particularly in Japan.

All of this drives home the point that systems are logical in nature, and that programming is physical. If systems are designed properly, there is no reason they shouldn't behave identically on whatever computer platform you come up with. Better yet, it allows us to easily migrate our systems from one configuration to another. Uniformity and consistency in execution; and portability to boot. Imagine that.

END