Old Software IS Viable

One of the drivers of new systems in business today is replacement of “ageing” systems. Problem is that software does NOT grow old, only the programmers do and new programmers can be trained.

Legitimate software obsolescence is driven by genuinely obsolete hardware coupled to software that is associated with the operating environment for that hardware for which there is NO effective port to new systems, or badly maintained software – which frequently comes about because of a CIO who decides to kill the system by NOT investing in people and maintenance. The latter case can be remediated

There ARE situations in which legacy software can have its life extended considerably. This article discusses this. Following are some key points:

1. **Instructions for the bricklayer**
   Source code is “instructions for the bricklayer” – the English like language is translated by the compiler into binary code and processor and device addresses – in this respect the compiler is like a bricklayer simply following instructions to generate the executable code.

2. **Programming language ONLY for the programmer**
   Programming languages are simply there for the convenience of the programmers so that they can write more complex and elaborate software applications.

   To say that a language is obsolete is to say that the people who knew how to program in that language are either all dead or have moved on to other languages. Insofar that in even the oldest languages there are still programmers alive the question then becomes one of providing the right financial incentive to bring old programmers back from other languages.

   Alternatively, NOTHING prevents you for training young programmers on old languages. The world is full of factories with old machines that are maintained by technicians and mechanics who have been trained up on equipment that was manufactured before they were born. There is NO reason NOT to do this with software.

3. **Development languages are a fashion industry**

Application development languages have historically progressed to a point and then been superseded by new languages that have been written to perform different functions, but to a point it is a fashion business. It is my view that this trend is coming to an end. The current mainstream languages will probably be around for a very long time.

In some cases of really leading edge technology applications there MAY be real benefit moving to a new language, BUT in the case of core mundane operational business information systems there is generally little or NO benefit. Remember always that you can develop new components with a new language and leave the existing components in the old language. Once they are compiled the computer does NOT know the difference.

4. **Software NEVER wears out**
   Software NEVER wears out, it has NO moving parts. The older languages died out because the computer processors died -- that does NOT happen anymore.

5. **Once software works it always works**
   Once software works it ALWAYS works unless some human being does something that stops it working – either as an act of incompetence, negligence or sabotage.

6. **Mainstream legacy languages remain valid**
   Mainstream older languages, such as Cobol in particular, remain valid because compilers have been ported to run on modern processors so barring minor conversion idiosyncrasies old code will work perfectly well on modern computers. See "COBOL Still Used More Than Google".

7. **Once compiled the processor is ignorant of the language**
   It is vital to be aware that once compiled the processor does NOT know whether it is running 30 year old Cobol or the latest "dot net" or whatever language. The language is PURELY for the human beings who write the program.

8. **"Obsolete" is therefore most frequently a fashion statement**
   "Obsolete" is therefore most frequently entirely a fashion statement, the developers are still around and many will happily work on old code for a fair wage. Many of the old guard prefer the older languages because they are leaner and more efficient.

9. **It is NOT necessary to replace the entire application**
   Note that there is NO need for all parts of a software application to be written in the same language. If you have an elderly Cobol suite of software and you want a pretty web front end interface in the latest language, NOTHING stops you writing the new front end in that language and writing an interface to the database of the old application or writing an interface to feed data files from the old application to the new database.

   There are tools specifically for that purpose such as "Advantage Data Transformer".

10. **The 80:20 Phenomenon**

Note also that as with most things in life, of the order of 80% of the software that is used in a business does NOT change for years. This can remain in your legacy language. Just replace the 20% of the software where there REALLY ARE benefits from upgrading. The minute you move away from a mind-set that says that you MUST replace the ENTIRE solution interesting possibilities open up.

11. **Mainstream databases are still in existence**

    Some of the older databases HAVE died out and this MAY legitimately drive replacement but in MANY cases there is a solution and some mainstream databases that have faded from the limelight but are still out there and supported, such as DB2 and Informix -- you just have to look.

    Again, in many cases the problem is fashion NOT technology obsolescence.

12. **The ONLY limitation is the ingenuity of the developers (and their willingness)**

    In many (most?) cases the limitations with regard to legacy software relate to the ingenuity of the developers and managers and their willingness to find solutions.

    It is TOO easy to tell management that the old systems are obsolete because you have a career need to learn a new language so that you are more marketable and can earn a higher salary, or just because it is an adrenalin rush to learn something new. Those are NOT valid reasons.

Bottom line is that probably 90% of the legacy application out there CAN be maintained and used indefinitely and can be extended by bringing in selected modern components.

It is also so that if you look at the REAL costs of implementing and operating the major ERP and related products and apply a percentage of that cost to maintaining your existing systems you will be very well off. There may WELL be a case for a project to remediate and strengthen your existing systems and extend them to better support your current and future strategic position.

Note also that, given the massive failures that are occurring, the business risks associated with progressively and incrementally remediating and enhancing your existing systems are MASSIVELY lower.

Dr James A Robertson PrEng
11 September 2014