DMSMS Knowledge Sharing Portal Newsletter, Vol. 4, Issue 4, pp. 3 and 11, June 2006

The Other Half of the DMSMS Problem - Software Obsolescence

Peter Sandborn - CALCE, University of Maryland
Galen Plunkett – ASSETT, Inc.

Today there are a growing number of methodologies, databases and tools that address status, forecasting, risk, mitigation and management of technology obsolescence. The one common attribute of all the methodologies, databases and tools that are in use today, whether reactive, proactive or strategic, is that they focus on the hardware life cycle. In most complex systems, software life cycle costs (redesign, re-hosting and re-qualification) contribute as much or more to the total life cycle cost as the hardware, and the hardware and software must be sustained together.

Software obsolescence is generally due to one of three main causes:

1. Functional Obsolescence: Hardware, requirements, or other software changes to the system, obsolete the functionality of the software (includes hardware obsolescence precipitated software obsolescence; and software that obsoletes software).
2. Technological Obsolescence: The sales and/or support for COTS software terminates:
   - The original supplier no longer sells the software as new
   - The inability to expand or renew licensing agreements (legally unprocurable)
   - Software maintenance terminates - the original supplier and third parties no longer support the software
3. Logistical Obsolescence: Digital media obsolescence, formatting, or degradation limits or terminates access to software.

Analogously, hardware obsolescence can be categorized similarly to software obsolescence: functional obsolescence in hardware is driven by software upgrades that will not execute correctly on the hardware (e.g., Microsoft Office 2005 will not function on a 80486 processor based PC); technological obsolescence for hardware means that more technologically advanced hardware is available; and logistical obsolescence means that you can no longer procure a part.

Although some proactive measures can be taken to reduce the obsolescence mitigation footprint of software including: making code more portable, using open-source software, and third-party escrow where possible; these measures fall short of solving the problem and it is not practical to think that software obsolescence can somehow be avoided. Just like hardware, military and avionics systems have little or no control over the supply chain for COTS software or much of the software development infrastructure they may depend upon for developing and supporting organic software. Need proof? Consider the following quote:

> "The only big companies that succeed will be those that obsolete their own products before someone else does"
>
> Bill Gates
> Founder, Chairman, Microsoft Corp.
> "The Bill Gates Method," APT News, July 21, 2003

Obviously, Microsoft's business plan is driven by motivations that do not include minimizing the sustainment footprint of military and avionics systems.

In the COTS world, hardware and software have developed a symbiotic supply chain relationship where hardware improvements drive software manufactures to obsolete software, which in turn cause older hardware to become obsolete – from Dell and Microsoft's viewpoint, this is a win-win strategy. Besides COTS software (hardware specific and non-hardware specific), system sustainment depends on organic application software, software that provides infrastructure for hardware and software development and testing, and software that exists at the interfaces between system components (enabling interoperability. While hardware obsolescence precipitated software obsolescence is becoming primarily an exercise in finding new COTS software (and more often COTS software and new hardware are bundled together), the more challenging software obsolescence management problem is often found at the interfaces between applications, applications and the operating system, and drivers.

The Costs Associated with Software Obsolescence

There are significant costs associated with the management and mitigation of software obsolescence. In general the following areas of cost/resource/time have to be considered:

Mitigation – Few, if any, of the mainstream hardware mitigation approaches are applicable or even meaningful for the management of software obsolescence. The most common approaches to mitigating software obsolescence are: Software License Downgrade – A license downgrade is negotiated with the software vendor and enables the users to expand or extend the authorized use of an older product by purchasing additional licenses of the latest version and applying those licenses to the older product; Source Code Purchase – In some cases, the original vendor may allow the customer to purchase the source code for the product (this may be negotiated in the original contract for the software); and Third Party Support – In some cases a third-party can be contracted to continue support of an obsolete software application.

Re-development – Modifying software to operate correctly with new application hardware (or with other new software). This can range from retesting the software to re-architecting and may also include: re-integration, data porting (data migration), re-training and document revision.

Re-qualifying – Either modified or un-modified legacy software may have to be tested and re-qualified in an operational environment.

Rehosting – Rehosting means modifying existing software to operate correctly in a new development environment (also called technology porting). Rehosting is applicable to legacy software systems originally created with languages and systems that themselves have become obsolete.

Media Management – Storage and maintenance of the medium the software is archived on is a critical element of software obsolescence. There are several problems (and costs) involved that depend on the type of media, the method of storing the media and version control.

Case Resolution – Just like hardware DMSMS case resolution, software DMSMS cases must be managed and tracked as well.

So where do we go from here? The specific impacts on system software are largely ignored throughout the existing DMSMS tool/database set. In reality, DMSMS is a hardware/software co-sustainment problem, not just a hardware sustainment problem. In existing tools, systems are treated as a disembodied Bill of Materials with little or no coupling from part-to-part and little functional knowledge of the operation of the system. Software obsolescence (and its connection to hardware obsolescence) is not well defined and current DMSMS planning tools are not generally capable of capturing the connection between hardware and software.

For additional information on various aspects of software obsolescence, readers are encouraged to see the following sources:

The CALCE Electronic Systems Cost Modeling Laboratory at the University of Maryland, http://www.enme.umd.edu/ESCML/obsolescence.htm

L. Merola, "The COTS Software Obsolescence Threat," *Proceedings of the International Conference on Commercial-Off-The-Shelf (COTS) Based Software Systems*, 2006.

T. Rickman and G. Singh, "Strategies for Handling Obsolescence, End-of-Life and Long-Term Support of COTS Software," *COTS Journal*, pp. 17-21, January 2002.