IEEE Trans on Components and Packaging Technologies, Vol. 30, No. 4, pp. 886-888, December 2007.

Software Obsolescence – Complicating the Part and Technology Obsolescence Management Problem

Peter Sandborn
CALCE, Department of Mechanical Engineering, University of Maryland

I. INTRODUCTION

As a result of the rapid growth of the electronics industry, many of the electronic parts in products have a procurement life cycle that is significantly shorter than the life cycle of the system they go into. A part becomes obsolete when it is no longer manufactured, either because demand has dropped to low enough levels that manufacturers choose not to continue to make it, or because the materials or technologies necessary to produce it are no longer available. The military refers to this situation as Diminishing Manufacturing Sources and Material Shortages (DMSMS). Avionics and military systems may encounter obsolescence before being fielded and always experience obsolescence problems during their field life, [1].

Today there are a growing number of methodologies, databases and tools that address status, forecasting, risk, mitigation and management of electronic parts obsolescence, [2]. However, 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 exclusively 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 concurrently sustained.

II. SOFTWARE OBSOLESCENCE

Software obsolescence (more specifically COTS – Commercial Off The Shelf 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 (end-of-sale)
   - The inability to expand or renew licensing agreements (legally unprocurable)
   - Software maintenance terminates - the original supplier and/or third parties no longer support the software (end-of-support)
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.

The applicable definition of software obsolescence varies depending on the system that uses the software, and where and how that system is being used. COTS software has both end-of-sale dates and end-of-support dates that can be separated by long periods of time. For many mainstream COTS software applications (e.g., PC operating systems), both the end-of-sale and end-of-support dates are published by the software vendors. For applications that have a connection to the public web (e.g., servers and communications systems), the relevant software obsolescence date for both the deployment of new systems and the continued use of fielded system is often the end-of-support date because that is the date on which security patches for the software terminate making use of the software a security risk . For other embedded or isolated applications, the relevant software obsolescence date is governed by either an inability to obtain the necessary licenses to continue using it or changes to the system that embeds it (functional obsolescence issues).

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[1] where possible; these measures fall short of solving the problem because they often require large or simply unavailable resources to take advantage of. It is also 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 the software development infrastructure they may depend upon for developing and supporting in-house software. Need proof? Consider the following quote [3]:

> "The only big companies that succeed will be those that obsolete their own products before someone else does"
>
> Bill Gates
> Founder, Microsoft Corp.

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 in-house (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.

III. Software Obsolescence Mitigation and Costs

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:

[1] A third-party receives source code if the original provider terminates support.

Mitigation – Few, if any, of the mainstream hardware mitigation approaches [4], are applicable or even meaningful for the management of software obsolescence. The most common approaches to mitigating software obsolescence are [5,6]: 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 – In re-development, software is modified 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 – Various DMSMS case resolution costs apply to software just like hardware DMSMS case resolution. These costs may include tracking various resolution metrics, version control, and database management.

IV. OUTLOOK

Little attention has been paid to software obsolescence. Besides digital preservation,[2] the termination of sales and support of software comprises the primary focus of the existing literature [5-10].

So where do we go from here? COTS software obsolescence is neither well understood nor managed today and 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.

[2] If, for example, one searches on "software obsolescence" in Google, the vast majority of what appears is related to "information or digital preservation."

Few, if any, system development and support organizations actually track and manage software obsolescence. Systems engineering approaches to concurrently manage software and hardware obsolescence are virtually non-existent and there are no formal organizations sharing obsolescence data and information on software. Various DMSMS working groups throughout industry and the defense community have only just started to address the management of software obsolescence impacts and welcome discussion on this topic.

REFERENCES

[1] P. Singh and P. Sandborn, "Obsolescence Driven Design Refresh Planning for Sustainment-Dominated Systems," *The Engineering Economist*, Vol. 51, No. 2, pp. 115-139, April-June 2006.

[2] P. Sandborn, R. Jung, R. Wong, and J. Becker, "A Taxonomy and Evaluation Criteria for DMSMS Tools, Databases and Services," *Proceedings of the 2007 Aging Aircraft Conference*, Palm Springs, CA, April 2007.

[3] The Bill Gates Method," APT News, July 21, 2003. (Available at: http://www.placementpartner.com/apt%20news/news7-21-03.html)

[4] R.C. Stogdill, "Dealing with Obsolete Parts," *IEEE Design & Test of Computers*, Vol. 16, No. 2, pp. 17-25, April-June 1999.

[5] 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.

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

[7] T. Mittelstaedt, "Network Community, Managing Software (and System) Obsolescence" *Computer Bits*, Vol. 9, No. 11, November 1999.

[8] R. Sheppard, "Software Obsolescence," *COG Workshop Presentation*, May 19, 2005.

[9] A. Gowland, "Managing Software Obsolescence in the COTS Environment," *Component Obsolescence Group (COG) Workshop Presentation*, November 27, 2002.

[10] "Does Software Go Obsolete?" *CIE Components in Electronics*, March 2006.