3rd CIRP Global Web Conference

Impact of software obsolescence in defence manufacturing sectors

S. Rajagopalan*, J. A. Erkoyuncub, R. Royb

*School of Applied Science, Cranfield University, Cranfield, MK43 0AL, UK
bSchool of Applied Science, Cranfield University, Cranfield, MK43 0AL, UK

* Corresponding author. Tel.: +44-753-921-6931; E-mail address: s.rajagopal@cranfield.ac.uk.

Abstract

Software plays an important role in Defence industry. Almost every project in defence manufacturing sectors has got software with various degrees of complexity and dependencies. Whilst various research and studies has been conducted on system obsolescence, and tools developed to cost systems and components obsolescence, no major research has been undertaken to develop a framework to estimate cost of software obsolescence. Software becomes obsolete even when it is completely functional but not useful. The aim of this paper is to understand the current practices in identifying the software obsolescence, types of software obsolescence and costing of software obsolescence.

1. Introduction

Software is defined as programs, procedures, rules, data and documentation associated with programmable aspects of system hardware and infrastructure [IEC 62402]. In simple terms, software are those set of programs, which will make the functionality of system or hardware more simple and easy to use, and it comes with a manual.

According to NASA (Cost Estimating Handbook):

- 55% of software projects exceed budgets by at least 90%.
- Software projects at large companies are not completed by 91%.
- Of completed projects only 42% of them have original proposed features.
- Historical Cost estimates for NASA projects are under estimated by a factor of at least 2.

2. Software in Defence

Software plays an important role in defence. Almost every project in defence has got software with various degrees of complexity and dependencies. The range of software depends on the different platforms, from major complex weapon system to command and communication system, from mission planning to fighter aircraft. Software is an integral part of every project in various dependencies and complexities. [2]

3. Software Obsolescence

“The only big companies that succeed will be those that obsolete their own products before someone else does” (Bill Gates, 2006). This is true with the defence projects as MoD depends heavily on COTS and vendor’s stops support to this software as soon as they deliver this software and moves on to produce different software. As soon as this happens the software becomes obsolete.

Today, the military acquires, test, and fields system of systems mixing COTS and non-COTS products [1]. These projects span for decades and the support and maintenance of these products exceed the vendor’s capacity. According to a study conducted by BMT and MoD it was found that for every pound spent in developing the software £2 is spent on the maintenance of software and if it is bespoke software then you are looking at £20.
Software becomes obsolete when there is a new technological advancement; the software functionality is not required or other market factors. Vendors stop supporting when they release the new version of the same programme. Software becomes obsolete even when it is completely functional but not useful [1].

Software gets obsolete generally due to one of the three main causes [3]. 1. **Functional obsolescence**: If there are changes to the hardware, system or other software in the same system. This will behave like a ripple effect as changes in one software or hardware will affect the next one and so on until it affects the entire system or capabilities. 2. **Technological Obsolescence**: This happens when vendor stops supporting the products or unavailability of the software in market etc. 3. **Logistical Obsolescence**: This happens when the media or the hard drive for example does not support the software. For example some of the new software will not function in old processors.

It is good to have some classification of the causes of software obsolescence but P Sandborn did not mention the human obsolescence [3] and the skills obsolescence issues. Human obsolescence plays a major role because its human nature to move from one place to another or in this case from one project to another or from one company to another. It will be impossible for a vendor to accommodate these migrations. When the employees move they take their knowledge of the product with them.

Skills obsolescence is another major factor. This is a key issue as in defence most of the safety critical aviation software is written in Ada 95 and there is not enough people that can maintain these software and vendors cannot afford to keep them either.

A recent study on software obsolescence came up with a different classification to P Sandborn’s classification. [3] This classification looked at skills and human obsolescence as shown in Fig. 1 [4]

![Fig. 1. Types of software obsolescence (Romero Rojo et al. 2009)](image)

The applicable definition of software obsolescence varies depending on the system that use the software and where and how that system is being used. COTS software has got both end of sale dates and end of support dates that can be separated by a long period of time [5].

It is been argued that software will not become obsolete as it is not affected by the degradation and can be easily replicated. They are trying to apply the same reasoning to software obsolescence as mechanical or electrical component obsolescence. When an electronic or mechanical component becomes obsolete and there is no more stock available, the system cannot be maintained according to the original planning. Analogously, the software obsolescence prevents the software from being maintained accordingly [4]. Software obsolescence happens when the original developer and authorised third party cease to provide support with regular updates, upgrades or fixes or due to changes in the target environment, systems, and hardware, which makes software unusable. This study will be based on this definition of software obsolescence.

Software obsolescence plays a major role within the defence and commercial sectors and there is a lot of ground that needs to be covered. Due to this reason and to understand the software obsolescence it was important to understand the software life cycle. For this study software life cycle such as waterfall, spiral iterative/incremental life cycle was mapped across to the CADMID life cycle.

<table>
<thead>
<tr>
<th>Table 1. Mapping of software life cycle with CADMID project life cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S/W Obsolescence</strong></td>
</tr>
<tr>
<td><strong>Skills</strong></td>
</tr>
<tr>
<td>Knowledge</td>
</tr>
<tr>
<td>Edit Source</td>
</tr>
<tr>
<td>Recording</td>
</tr>
<tr>
<td>Rebuild</td>
</tr>
<tr>
<td>Documentati on</td>
</tr>
<tr>
<td><strong>S/W COTS</strong></td>
</tr>
<tr>
<td>Operating</td>
</tr>
<tr>
<td>System</td>
</tr>
<tr>
<td>Tools</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Programs</td>
</tr>
<tr>
<td>Languages</td>
</tr>
<tr>
<td>Compilers</td>
</tr>
<tr>
<td><strong>Media</strong></td>
</tr>
<tr>
<td>Formats</td>
</tr>
<tr>
<td>Data Storage</td>
</tr>
<tr>
<td>Storage</td>
</tr>
<tr>
<td>Material</td>
</tr>
</tbody>
</table>

This approach provided clarity on the literature review as this allowed focuses on the individual activities in the different phases of CADMID project life cycle.

Most of the literature available on software obsolescence is in the in-service phase to mitigate the software obsolescence reactively rather than proactively. This literature looks at the mitigating action on what needs to happen when software obsolescence occurs. There is a serious knowledge gap in the scientific, commercial and academic world on the cost of software obsolescence.
management of software obsolescence. There is also a major shortage of knowledge on how to combat software obsolescence on other phases of projects such as Concept, assessment, demonstration and manufacture. This has a profound effect on the shortage of knowledge and research in finding the cost of software obsolescence at a very early stage. Table 2 below provides a rough order of magnitude on the availability of literature on software obsolescence at various stages of the project life cycle. * 

<table>
<thead>
<tr>
<th>CADMID</th>
<th>Availability of Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>4%</td>
</tr>
<tr>
<td>Assessment</td>
<td>0%</td>
</tr>
<tr>
<td>Demonstration</td>
<td>0%</td>
</tr>
<tr>
<td>Manufacture</td>
<td>10%</td>
</tr>
<tr>
<td>In-Service</td>
<td>85%</td>
</tr>
<tr>
<td>Dispose</td>
<td>1%</td>
</tr>
</tbody>
</table>

4. Impact of Software Obsolescence

The US Obsolescence management group Diminishing Manufacturing Sources and Material Shortages (DMSMS) underpins a risk based approach to determine its management priorities in particular software obsolescence. Such as a system heavily dependent on COTS should assign it a relatively high priority (Jay, 2013). According to DMSMS there are three ways to determine software obsolescence management. Identify where they formulate an OMP and OMT, Assess phase where software impact assessment is carried out and analysis phase where they provide resolution to the problem. This is one way of approaching the software obsolescence and managing obsolescence as this is a heavy resource bound and does not provide an early indication of how much this will cost. A further study is required as this risk-based approach is difficult to quantify and the probability and the level of this risk happening is difficult to determine at a very early stage.

In the world of emerging dependencies on software, software development companies have to cope with the increase in demands of requirement. Technological advancement, increasing in the size and scope of the users and a difficult market place for software business. This is even bigger in the defence industries as the defence projects span for a longer period of time. This rapidly changing business environment is challenging traditional requirement engineering [6]. This makes the original requirement obsolete and thus increasing the project time scale, cost. Having a requirement obsolete before the project is delivered is a factor in identifying the cost of software obsolescence.

Software quality is measured from the foundation laid out by software requirement [7]. As stated earlier, software requirement changes more often and fast that the requirement becomes obsolete. If this is the case then the software quality is affected as this is based on the requirement. A study conducted by Hall ands Wilson [8] shows that the experience and views of grassroots practitioner are not listened to and there is a significant bureaucracy in controlling the software quality. There have been a number of studies undertaken to improve software quality since then. There is a clear indication from above that there is a direct relation between the requirement and quality which will have an effect on the software’s life making it prone to obsolescence.

Software reliability is one of the key attributes to the software quality. There has been several studies and research done on different software reliability model [9; 10] but there is a knowledge gap and a lack of literature to map software reliability with the software obsolescence.

One way to tackle software obsolescence is to undertake a long-term digital preservation [11]. What software uses along side with the system is the data and the data is key for the survival of any business. The preservation of the data is important for the success of the software business if the software becomes obsolete. The preservation of data can become a very expensive project if not proactively managed.

Managed integration uses a services-based approach to enable inclusion of wide-ranging languages, drivers, interface, protocols, test executives and instrument types into hybrid systems that easily combine new and existing elements [12]. This is one way of dealing with system and software obsolescence in military test platforms. Weltzin & Schlonsky [13] talked about virtualisation of the of the Linux based test platform to counter obsolescence. Test equipment is used for a very long time and the costs associated with this ageing automated test equipment are exponential. $50B was spent on DoD related ATE between 1980 and 1992, over the lifetime of this equipment cost are expected exceed $250B [14].

Obsolescence and market growth rate drives the decision to issue an upgrade; the timing of upgrades is one of the most difficult decision [15].

Reengineering is a process of reverse engineering, programme restructuring, software understanding and translating legacy software into modern programming language. One of the dominant issues in recent years is the need for reengineering the legacy software into modern language to combat the effect of software obsolescence. Ahrens et al [16] came up with different scenarios of the reengineered software process. This scenario gives us an understanding of what process

---

* This is based on literature review of 120 literature searched using key word such as software obsolescence, software maintenance, software reliability, cost of software etc.
needs to be applied to combat obsolescence, but what this does not explain is the cost associated with this process. Even though this process as explained by the author is expensive but did not quantify the cost.

The British Standard dictates to have a good reactive and proactive strategy to mitigate the software obsolescence. But there are lots of other factors that need to be considered in mitigating software obsolescence, one of the key factors being cost. There are different ways that the software obsolescence can be mitigated such as software license downgrade, redevelopment of software which is then required to prequalify or rehosting of software will do the job but the main constraint to these are Cost, Resource and Time.

Component Obsolescence Group (COG) recommends using risk management based process (Fig.2).

![Fig. 2. Software Obsolescence Process Overview (COG) [20]](image)

5. Current Practice and Challenges

Most of the MoD projects are combination of integrating old platform with the new COTS or vice versa, hence the most challenging software obsolescence management problem is often found at these interfaces among applications or the integrating glue codes. In COTS world, hardware and software depends on each other heavily, software obsolescence is driven by the changes in hardware improvement.

Various people with various degrees of experience in software design and system integration from MoD project teams and from industry have been interviewed. It has been very difficult to understand the current practices as there are different dissimilar approaches taken by the project managers in DE&S. Participant 1 a Software Obsolescence Manager from the MoD project team advised me that they pass the risk to the industry. He also advised that there is a difference between the definition used by the government departments and industry. Whereas Participant 2 a system engineer from a different project team advised that they followed JSP 886 and have a proactive strategy in place, but were unable to quantify the cost of this strategies. Participant 3 and 4 from the industry spoke about planned obsolescence where they talked about releasing the newer version of the software when the current version does not hold ground in the market.

There was a similar response from different companies that attended COG conference in late July 2013. To a direct question on how to manage software obsolescence Plant Manager of Company A said they have not looked into software obsolescence but have developed a strategy for system obsolescence. Team leader Material and Obsolescence of Company B also commented that they understood that software obsolescence is a major issue.

Similar response was heard from different cost analysts who attended Price True Planning Annul Conference in August 2013.

Based on the interviews and literature review undertaken the main problem in identifying the key cost drivers of software obsolescence is:

- Unavailability of the data related to Software Obsolescence
- Inadequate design documentation and configuration

\[\text{\textsuperscript{b}}\] Name of the project team is not mentioned due to the security issues
\[\text{\textsuperscript{c}}\] Commercial companies not related to defence projects
\[\text{\textsuperscript{d}}\] One of the Largest public transport company in Europe
\[\text{\textsuperscript{e}}\] One of the largest Defence Contractor in UK
Unavailability of historic cost data associated with Software Obsolescence

Unavailability of Software obsolescence Management plan and strategies

In order to identify the key cost drivers we need to clearly differentiate between activities under software maintenance such as update, patches and upgrades with software obsolescence. There is a clear relationship between software obsolescence with software design and this will be further explored in my research to identify the key cost drivers.

6. Conclusions

Software Obsolescence is one of the key issues for any defence manufacturing industry, every system and every manufacturing techniques have got software embedded in them. It is becoming very important for the industry and defence sectors to address this issue and find an approachable way to counter the effect of the software obsolescence. There have been many research in mitigating the software obsolescence such as updates, upgrades, patches, emulating etc. It has become important to identify the software obsolescence at a very early stage to effectively manage this issue especially when the project life cycle is been extended to 25 to 30 years. Hence to find the cost of these issues at a very early stage to inform the funding decisions is very important. My research is focused on to identifying the key cost driver of software obsolescence and create a framework with which software developers and project managers can predict the cost of software obsolescence at a very early stage of the project to make an informed decision on the support contract.

References


