International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:9, 2017

Software Obsolescence Drivers in Aerospace: An Industry Analysis

Raúl González Muñoz, Essam Shehab, Martin Weinitzke, Chris Fowler, Paul Baguley

***Abstract*—**Software applications have become crucial for the aerospace industry, providing a wide range of functionalities and capabilities. However, due to the considerable time difference between aircraft and software life cycles, obsolescence has turned into a major challenge for industry in last decades. This paper aims to provide a view on the different causes of software obsolescence within aerospace industry, as well as a perception on the importance of each of them. The key research question addressed is what drives software obsolescence in the aerospace industry, managing large software application portfolios. This question has been addressed by conducting firstly an in depth review of current literature and secondly by arranging an industry workshop with professionals from aerospace and consulting companies. The result is a set of drivers of software obsolescence, distributed among three different environments and several domains. By incorporating monitoring methodologies to assess those software obsolescence drivers, benefits in maintenance efforts and operations disruption avoidance are expected.

***Keywords*—**Aerospace industry, obsolescence drivers, software lifecycle, software obsolescence.

I. Background

AEROSPACE product development has traditionally been on the frontline of technological advancement. During the last decade, aerospace companies started to adopt and benefit from the IT revolution in order to boost innovation in design, manufacturing, and support. As a result, software is increasing its weight in aerospace, providing unique capabilities which are now critical to the industry. This has resulted, however, in complex data structures, with diverse data formats and coming from a wide range of software applications.

In aerospace, the lifecycle of the products can comprise many decades, even being extended further on time if given the necessary conditions [1]. Due to the high costs and long life times associated with technology insertion and design refresh, aircrafts tend to fall behind the technology wave [2], [3]. As such, one critical issue aircrafts will face during its lifecycle is obsolescence [4]. This is especially noticeable in the case of software, as can be seen in Fig. 1.

Regarding software obsolescence, some people would argue that software applications cannot become obsolete as they are not affected by degradation (and hence do not require replacement) and can be easily replicated. Their misunderstanding is to try to employ the same reasoning to software obsolescence as to mechanical or electrical component obsolescence. It is required to comprehend the different nature of the software obsolescence issue. The significance of obsolescence is that it prevents from maintaining and supporting the system, hence creating a risk of disruption on the operations [5]. This issue is especially critical due to the number of processes, data, and people which depend entirely on software with a shorter lifecycle to continue business operations, as seen in Fig. 2. Furthermore, due to the long certification processes in aerospace, there is the need to maintain data usable for long periods of time, adding an additional challenge [6].

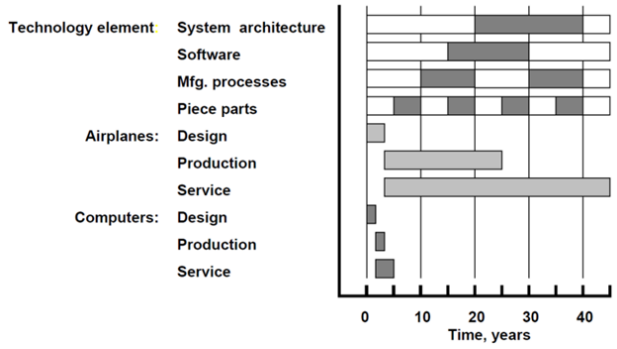

Fig. 1 Component technology, airplane, and computer lifetimes [4]

During the last years, several researchers have recognized the criticality of software obsolescence, especially regarding COTS software [7], [8], defense and aerospace [5], as well as other long-life assets [9]. British Standards Institute (BSI) and The Institute of Obsolescence Management (IIOM) have also played a key role, remarking the importance of this topic and publishing about it [10], [11]. Finally, it is especially remarkable to mention the last publications on the topic, considering new aspects such as employee skills and its impact on software obsolescence [12].

II. Research Methodology

The research methodology involved three main phases, as illustrated in Fig. 3.

Raúl González Muñoz and Paul Baguley are with the Department of Manufacturing, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedfordshire, MK43 0AL, United Kingdom.

Essam Shehab is with the Department of Manufacturing, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedfordshire, MK43 0AL, United Kingdom (corresponding author, phone: +44 79 50554 084, e-mail: e.shehab@cranfield.ac.uk).

Martin Weinitzke is with the Airbus Operations, Hamburg, 21129, Germany.

Chris Fowler is with the Airbus Operations, Filton-Bristol, BS34 7PA, United Kingdom.

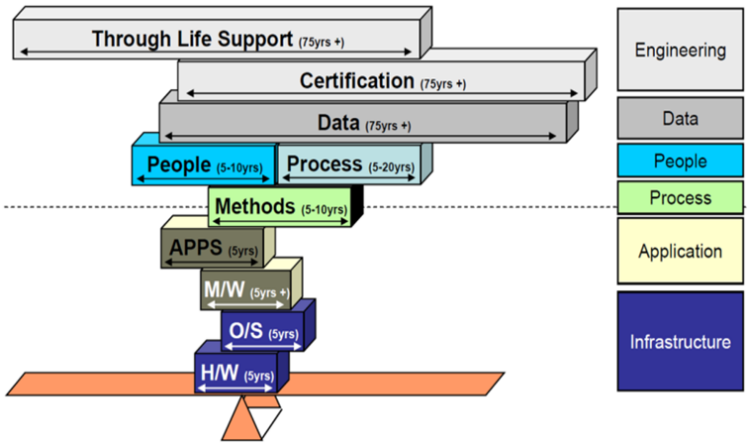

Fig. 2 Blocks certification stack [6]

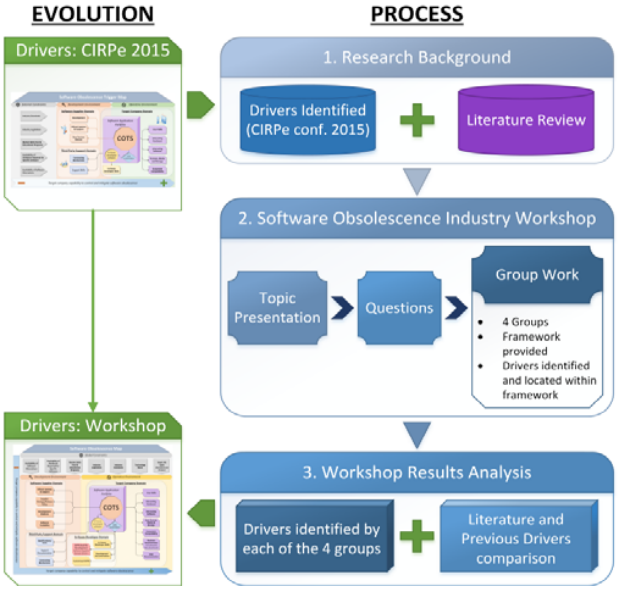

Fig. 3 Schematic depiction of the research methodology

The initial phase focused in literature review within the fields of obsolescence, software obsolescence, and data management. Thanks to this analysis of the state of-the-art, a starting point in the research was established. At the same time, a review of the previous research was conducted [13], with the aim of providing a baseline together with the literature.

The second phase involved an industry workshop with professionals from the aerospace industry, as seen in Table I. The workshop was arranged in two main sessions, a background presentation and the group work, with a total duration of seven hours.

The structure of the event was the following one:

- Topic background: A general introduction to the field of software obsolescence was presented to the participants
- The participants asked questions regarding the presentation
- The participants were divided into four groups and a framework was provided to them in order to brainstorm and place software obsolescence drivers within it. The groups were created in advance in order to ensure diversity of opinion and wide range of experience
- The discussions within each of the groups were captured for future reference
- The drivers were then presented by each of the groups and the results were shared with the rest of participants

TABLE I
WORKSHOP PARTICIPANTS

Participants	Experience	Role
Participant 1	29 years	Design Process Architect
Participant 2	6 years	DES Fellow/Senior IOH Estimator
Participant 3	7 years	Obsolescence Management Team
Participant 4	37 years	Senior Advisor Asset Management
Participant 5	25 years	Academic Reader
Participant 6	30 years	CTO
Participant 7	4 years	Project Leader
Participant 8	26 years	Engineering Computing Specialist
Participant 9	14 years	Chief of Design Technology
Participant 10	16 years	Obsolescence Manager
Participant 11	13 years	Senior Software Engineer
Participant 12	5 years	Application Services Portfolio Manager
Participant 13	30 years	Application Services Portfolio Manager
Participant 14	30 years	Consultant
Participant 15	37 years	Head of Application Management
Participant 16	37 years	Global IT Director
Participant 17	10 years	Director
Participant 18	10 years	Team Leader
Participant 19	22 years	Software Specialist

The third and final phase concerned the analysis of the workshop results, comparing the outcome of each of the groups with previous research and existing literature. The final deliverables were an improved software obsolescence map, with a set of drivers distributed in several environments and domains, and a ranking of each of the drivers based on the participant's perceptions.

A. Industry Participants

For this paper, an industry workshop with 19 participants from the aerospace industry and services related has been conducted. The profile of the different participants that were involved can be seen in Table I.

The sample of participants, even taking into account its variability in terms of experience, can be divided mainly into two main types:

- IT experts
- Obsolescence experts

The main reason for selecting these two profiles is that, due to its novelty, there are not many experts in software obsolescence specifically, hence the combination of these two profiles guarantees a good mix of expertise to fit the case. Furthermore, it was ensured that all participants had links with the aerospace field, to guarantee a good understanding of the topic by the participants when applied to this industry.

III. Software Obsolescence Drivers

Making use of the information gathered, from existing literature, previous research, and the workshop outcomes, a software obsolescence map was developed, with a total of 25 drivers. The aim was to validate and improve previously identified obsolescence drivers with the perspective of industry professionals from a range of companies, providing a clear view of the software obsolescence environment and its causes. This map is illustrated in Fig. 4 and comprises three main areas: global constraints, development environment and operative environment.

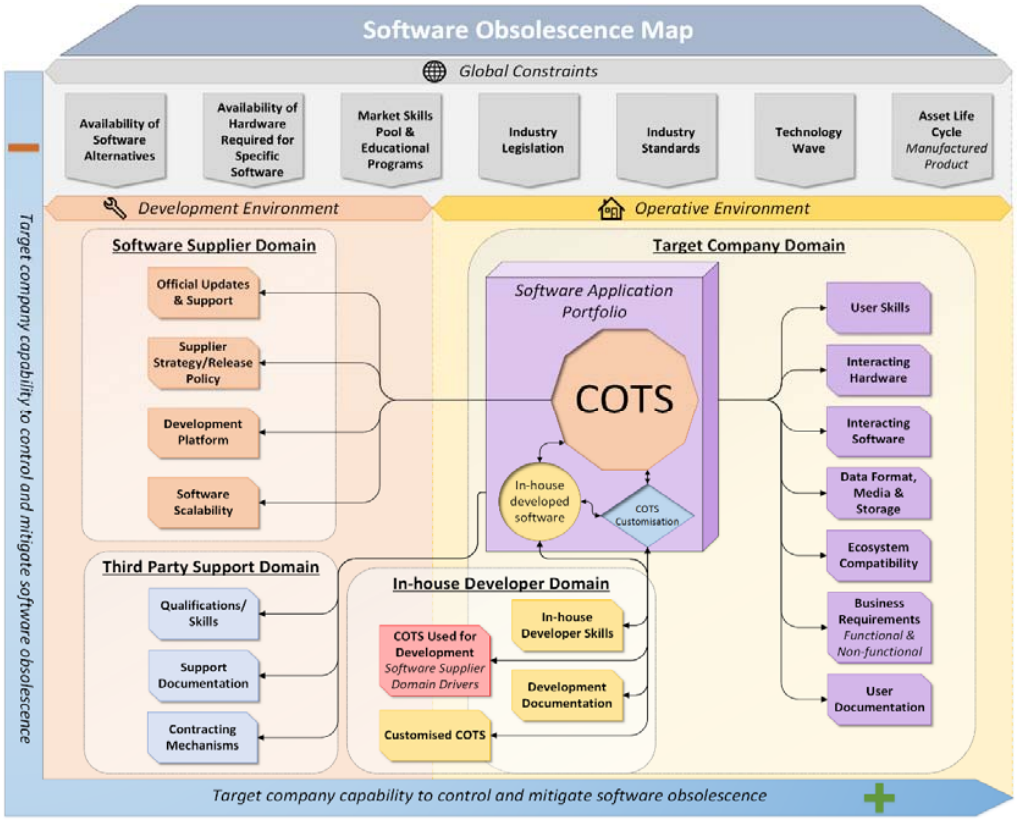

Fig. 4 Software obsolescence map

The operative environment is where the software applications are being used, providing service. The applications are grouped in a portfolio by the target company; hence the company is managing and making use of the applications. Within that portfolio, three main types of software can be found, namely COTS software, in-house developed software, and customised COTS software. In this environment, the main domain is the target company. Target company is the organisation using the applications to develop, support, and/or provide a range of services to the business. These services may vary substantially in nature and criticality, but the trend for the last decades has been to increasingly relay in software applications to perform activities previously manual. Within this domain, there can be found seven drivers of obsolescence, common to all the application portfolio.

- User skills: It refers to the skills and knowledge to properly use and support the applications. Without the proper set of skills, especially when no user documentation is available, it may be impossible for a company to make use of the software.
- Interacting software: It concerns the other software. The application interacts with during its usage, encompassing

applications, middleware and operative systems. Obsolescence in those interacting pieces of software can cause the application to become obsolete as well, creating the risk of cascade effects within the software application portfolio.

- Interacting hardware: It represents the hardware that supports certain software. The incapacity of maintaining it may make the supported software obsolete as well. The detailed analysis of this driver can grow in complexity quickly, as hardware obsolescence it is a field of study by its own, with specific drivers and metrics
- Format, media, and storage: It regards the data formats, way of storage and documentation associated. Aircrafts have a long lifecycle, and during all this time span, all the original data need to be accessible, in order to be able to support the aircraft during its operational life and to be responsive to external requirements.
- Ecosystem compatibility: It concerns the compatibility of each of the applications with the whole environment of the target company domain. This encompasses a wide range of business assets and processes.
- Business requirements: It refers to both, functional and non-functional:
  - Functions of a system or its components. A function is described as a set of inputs, the behaviour, and outputs. It may be calculations, technical details, data manipulation and processing and other specific functionality that define what a software application is supposed to accomplish. These required functions are dictated by the target company and an eventual mismatch between business required functions and functions provided by the application can render the software obsolete.
  - Requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours. While functional requirements define what a system is supposed to do, non-functional requirements define how a system is supposed to be, defining an overall property of the system as a whole or of a particular aspect and not a specific function. These requirements are dictated by the target company, and an eventual mismatch between them and the ones provided by the application can render the software obsolete.
- User documentation: It involves the documentation that describes each feature of the software application, and assists the user in realising these features. It is very important for user documents to not be confusing, and for them to be up to date, as well as to have a thorough index. Consistency and simplicity are also very valuable. The lack of proper user documentation can turn a piece of software obsolete, as it may be not possible for the business to support or make use of the application without it in the long run.

The other domain within the operative environment is the in-house developer domain. The in-house developer is the domain within the target company in charge of developing new software from scratch or to customise existing cots in order to satisfy the functional requirements of the business. It lays between the operative environment and the development environment, as it is actively involved in both development and usage of the software within the target company. There are four drivers in this domain.

- In-house developer skills: It refers to the skills of the target company developer to develop and maintain in-house developed applications, as well as customised COTS.
- COTS used for development: This driver is completely dependent on the software supplier domain. Its existence serves the purpose of reminding the dependency of a target company from COTS software regarding also its own developed software.
- Development documentation: It relates to both technical and architecture/design documentation.
  - Architecture/Design documentation: General view of the software. It shows the structure of software, comprising its components and the relationships among them
  - Technical documentation: Documentation regarding interfaces, application programming interfaces (APIs), algorithms and the code
- Customised COTS: It concerns those COTS that have a customisation implemented, either by the original developer, a third party or the target company itself. These customisations often become a trigger for obsolescence, as the support of them is not guaranteed by the original COTS developer, being up to the target company to implement updates and/or modifications into the software (incurring in much higher maintenance/ upgrade costs)

The development environment is where the software applications are developed, maintained, and improved. it encompasses mainly two different domains, the software supplier domain and the third party support domain.

The software supplier refers to the original developer and seller of the COTS software, as well as its customisations in some cases. Within its domain, four different obsolescence drivers can be found.

- Official updates and support: The lack of updates and support from the original developer may be as well a cause of obsolescence, as even with a third party support, the access to a certain tools or the code may not be possible without the original supplier
- Supplier strategy/release policy: It concerns the planning of the software supplier in terms of expected time of support for each software product and release time of new versions of that software. The release of new versions of software can produce obsolescence, depending on the support policy, the new formats and the existing infrastructure.
- Development platform: it refers to the technologies used to create a software application and to implement modifications, fixes and support. It encompasses mainly languages, but all aspects of integrated development environment are included. The disruption of these tools and technology by the community/industry may render obsolete the applications developed with them.

- Software scalability: It regards the capabilities of a software application to be deployed in diverse locations, supporting several languages, an increasing number of users and bearing additional features if required. The inability to meet the scalability requirements may render an application obsolete from the point of view of the business.

The third party support refers to the contractor the target company hires to provide support in activities related to the management of the application portfolio. It may be the original developer of the software, or it may not. Under its domain, three different drivers can be found.

- Contracting mechanisms: Depending on the characteristics of the agreement between the target company and the third party support, issues may rise in the future and be a cause of obsolescence. This refers mainly to the lack of support during a period of time which is not considered initially, or unexpected situations which are not considered in the original agreement terms.
- Support documentation: It relates mainly to the documentation focused in the system administrator tasks and responsibilities, as well as support staff information in a lesser extent. If the third party also performs modifications and fixes, technical and architecture/design documentation would be encompassed as well. The omission of these documents may translate into great difficulties to manage and support properly the software application.
- Qualifications/Skills: It refers to the work skills and industry qualifications required to support and maintain the software applications. If the third party support has problems locating and keeping professionals with the required set of competencies, this situation may evolve into obsolescence, due to the inability of providing proper support to the applications.

Global constraints refer to the different drivers that act over the industry environment and that are completely external to the organizations. They have a wide scope and involve all the companies within an industry field.

- Industry standards: Changing standards within a specific industry field may cause obsolescence in software, as some of the applications may not be aligned with the new requirements.
- Industry legislation: Changes in regulatory laws and policies can turn software applications obsolete, usually because of a mismatch with the new requirements.
- Market skills pool and educational programs: Obsolescence can come from the degree of availability of professionals with certain skills. This availability comes from different factors, mainly the current content of educational programs. For example, certain code languages are not being taught anymore, but there are still a lot of applications, built in those languages, operating nowadays
- Availability of hardware required for specific software: Certain software requires specific hardware to work. Hence, the availability of this hardware in the market is important, due to the dependency relation. The stop of official production can be mitigated with alternative and second-hand markets, but it is still just a mitigation, not a long-term solution.
- Availability of software alternatives: The degree of availability of software alternatives for certain functions can be a key driver for obsolescence. The lack of availability of software alternatives may cause obsolescence and/or make it worse, if the original software becomes obsolete for any reason.
- Technology wave: Progress in technology, also called Technology Life Cycle (TLC) or S-curve Innovation, can be a reason for a specific software or group of applications sharing common features or tools to become obsolete
- Asset life cycle (manufactured product): The difference in time span between the asset produced and the software used to create and support production/operation of that asset can render applications obsolete, as usually software life cycle is rather short in comparison to some complex assets (aircrafts, ships, trains, nuclear facilities, etc.)

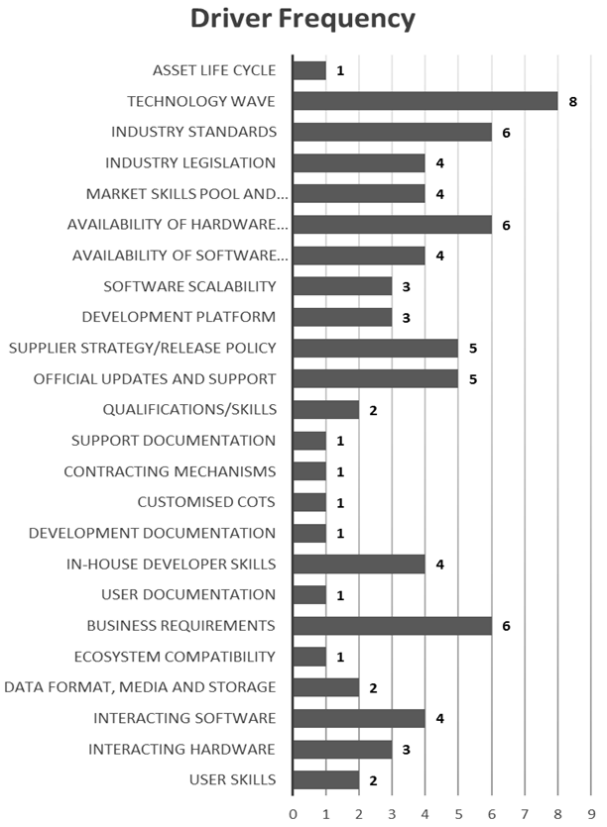

Fig. 5 Driver frequency during the workshop

The capability of the target company to act over obsolescence events will depend on the environment and domain. Hence, the target company will have control of its own domain, monitoring and acting over the obsolescence drivers. This includes the in-house developer domain, as it is

within the target company. However, this capability will be reduced in the domains of the software supplier and the third party support, as it will depend on the agreements in place between all the parts. Finally, in the case of global constraints, the target company is able just to adapt to the changes, being unable to act proactively.

IV. Driver Ranking

During the industry workshop, the discussions within each of the groups were captured. As a result, it is worth providing an analysis. Such analysis is shown in Fig. 5. As it can be easily observed, the most mentioned driver was technology wave. This is likely due to the general understanding that technology evolves through time, affecting all industries.

Industry standards, availability of hardware and business requirements were also mentioned with high frequency, based on the professional experience of the participants.

Lastly, it seems to be an interesting fact that skills-related drivers were not mentioned much, possibly due to the novelty of this research area within software obsolescence [12].

V. Conclusions and Further Work

Software obsolescence is one of the key challenges currently in the aerospace industry, fueled by longer aircraft life cycles and shorter software time support. With the increasing trend towards digitalisation in the aerospace industry, it is expected that this matter will steadily increase in criticality. As such, it has become important to develop new methodologies and techniques to manage software life cycle through time.

In the present paper, a software obsolescence map has been developed, with the aim of identifying the scope and drivers of the software obsolescence landscape in aerospace. This will help both industry and academia to understand the nature of software obsolescence within the aerospace field.

Future research in the area should involve the development of metrics based in those obsolescence drivers. This will enable the creation of monitoring strategies to manage software life cycle, mitigating obsolescence and avoiding business disruption.

Acknowledgment

This research project is funded by Airbus and Cranfield University. The author would like to gratefully acknowledge the support and assistance of Airbus as well as the contribution of participants from several aerospace organisations during the research.

References

[1] Livingston, H. (2000). GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices. Proceedings of the 2000 DMSMS Conference, Jacksonville, Florida, 21-25 August.

[2] Sandborn, P., Mauro, F. and Knox, R., 2007, A data mining based approach to electronic part obsolescence forecasting, IEEE Transactions on Components and Packaging Technologies, 30; 3; 397-401.

[3] Madisetti, V., Jung, Y., Khan, M., Kim, J. and Finnessy, T., 2000. On upgrading legacy electronics systems: methodology, enabling technologies and tools, VHDL International Users Forum Fall Workshop, 2000. Proceedings, Orlando, FL, USA, 18-20 Oct.: 7-14.

[4] Condra L (1999) Combating electronic component obsolescence by using common processes for defense and commercial aerospace electronics. IECQ-CMC Avionics Working Group1, NDIA, September 1997.

[5] Romero Rojo, F. J., Roy, R., Shehab, E., Cheruvu, K., Blackman, I. and Rumney, G. A., Key Challenges in Managing Software Obsolescence for Industrial Product-Service Systems (IPS2). The 2nd CIRP Industrial Product-Service Systems Conference, Sweden, 14th–15th April 2010.

[6] AIRBUS Long Term Application and Data Management Whitepaper, IBM, 2009.

[7] Sandborn, P., 2007b. Software Obsolescence - Complicating the Part and Technology Obsolescence Management Problem. IEEE Transactions on Components and Packaging Technologies, 30; 4; 886-888.

[8] Merola, L., 2006, The COTS software obsolescence threat, Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, 13-16 Feb. 2006. San Diego, CA.

[9] Erkoyuncu, J. A., Ononiwu, S., Roy, R., Mitigating the Risk of Software Obsolescence in the Oil and Gas Sector, Procedia CIRP, 22: 81-86, 2014.

[10] IEC 62402 - Obsolescence Management – Application Guide, BSI, 2007.

[11] Rumney, G. A., 2007, The Software Obsolescence Minefield: A Guide for Supporting Software and Electronic Information, 1st ed., IIOM International Ltd., UK.

[12] Sandborn, P. and Prabhakar, V., 2015, The Forecasting and Impact of the Loss of Critical Human Skills Necessary for Supporting Legacy Systems, IEEE Transactions on Engineering Management, Vol. 62, N. 3, August 2015.

[13] González Muñoz, R.., Shehab, E., Weinitzke, M., Bence, R., Tothill, S., Fowler, C. and Baguley, P., Key Challenges in Software Application Complexity and Obsolescence Management within Aerospace Industry, CIRPe 2015 - Understanding the life cycle implications of manufacturing, UK, Procedia CIRP, Volume 37, 2015, Pages 24-29, 201.