The University of York Department of Computer Science

Submitted in part fulfilment for the degree of MSc in Software Engineering.

Model-driven software migration between microprocessors

Sophie Wood

Version 0.1, 2018-May-11

Supervisor: Simos Gerasimou

Number of words = 0, as counted by `wc -w`.
This includes the body of the report, and Appendices TODO, but not TODO.

Abstract

TODO

Acknowledgements

TODO

Contents

1 Introduction

1.1 Background and Motivation

Bartels et al. define obsolescence as "materials, parts, devices, software, services and processes that become non-procurable from their original manufacturer or supplier" [1]. Both software and hardware can be subject to obsolescence problems.

Issues with hardware obsolescence have been driven by the growth of the electronics industry. This has reduced the life cycle of electronic parts as competitors release products with better functionality and features. Existing products are no longer commercially viable and therefore go out of production [1]. This is a particular issue in the defence/aerospace sectors as the typical life cycle of a system is 20-30 years or longer [2]. As such, parts will become unavailable before the system is completed. Singh et al. found that these systems "often encounter obsolescence problems before they are fielded and always during their support life" [3].

Obsolescence is also a concern in the software industry. Businesses must continuously update their products in order to stay ahead. Bill Gates has said:

> "The only big companies that succeed will be those that obsolete their own products before someone else does." [1]

There are three main types of software obsolescence [4]:

(1) Logistical: digital media obsolescence, formatting, or degradation limits or terminates access to software

(2) Functional: hardware, requirements, or other software changes to the system obsolete the functionality of the software

(3) Technological: the sales and/or support for commercial off the shelf (COTS) software terminates

A particular issue is that hardware obsolescence can drive software obsolescence and vice versa. Software obsolescence can also be caused by lack of skills. For example, there may be a limited number of people that are competent in the language the system is written in. If these people leave the company, the system can no longer be maintained [5]. Again avionics/military and other "safety-critical" systems particularly struggle with software obsolescence issues as even small changes may have to go through extensive and costly qualification/certification processes [3].

Obsolescence problems can be costly, particularly in the case of systems with a long life cycle. The US Navy estimates that obsolescence problems can cost up to $750 million annually [6]. Sandborn and Myers also found that sustainment costs (which include costs related to obsolescence) dominate the system costs in the case of development of an F-16 military aircraft [7].

However, strategies for obsolescence management have generally focused on hardware obsolescence problems. This is despite software obsolescence costs often equalling or exceeding that of hardware [4].

The Defence Science and Technology Laboratory (DSTL) have identified as a problem of particular interest "the migration of an entire software system from a legacy hardware platform to a modern more powerful platform" [8].

This report will address how this problem can be tackled using a model-driven engineering (MDE) approach.

1.2 Project Goals

TODO

1.3 Project Scope

TODO

1.4 Report Structure

TODO

2 Literature Review

3 Methodology

4 Requirements

5 Design and Implementation

6 Evaluation

7 Conclusion

Bibliography

[1] B. Bartels, U. Ermel, P. Sandborn and M. G. Pecht, *Strategies to the prediction, mitigation and management of product obsolescence*. John Wiley & Sons, 2012, vol. 87.

[2] F. J. R. Rojo, R. Roy and E. Shehab, 'Obsolescence management for long-life contracts: State of the art and future trends', *The international journal of advanced manufacturing technology*, vol. 49, no. 9-12, pp. 1235–1250, 2010.

[3] P. Singh and P. Sandborn, 'Obsolescence driven design refresh planning for sustainment-dominated systems', *The engineering economist*, vol. 51, no. 2, pp. 115–139, 2006.

[4] P. Sandborn, 'Software obsolescence-complicating the part and technology obsolescence management problem', *Ieee transactions on components and packaging technologies*, vol. 30, no. 4, pp. 886–888, 2007.

[5] S. Rajagopal, J. Erkoyuncu and R. Roy, 'Software obsolescence in defence', *Procedia cirp*, vol. 22, pp. 76–80, 2014.

[6] C. Adams, 'Getting a handle on cots obsolescence', *Avionics magazine*, vol. 1, 2005.

[7] P. Sandborn and J. Myers, 'Designing engineering systems for sustainability', in *Handbook of performability engineering*, Springer, 2008, pp. 81–103.

[8] S. Gerasimou, D. Kolovos, R. Paige and M. Standish, 'Technical obsolescence management strategies for safety-related software for airborne systems', in *Federation of international conferences on software technologies: Applications and foundations*, Springer, 2017, pp. 385–393.