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-16

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 causes 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]. For example, 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].

Strategies for obsolescence mitigation/management have generally focused on hardware obsolescence problems. This is despite software obsolescence costs often equalling or exceeding that of hardware. Proactive measures to mitigate obsolescence (e.g. improving code portability or using open-source software) often require either a large amount of resources or resources that are unavailable. Reactive strategies for tackling obsolescence include software license downgrades (i.e. customers can adapt legacy versions of the software by purchasing a licence for the latest version and applying that to the older product); purchasing the source code and third party support (i.e. a third party will take over maintenance of the legacy software). If these strategies are insufficient, the legacy system may have to be redeveloped or rehosted [4]. However, many such software modernisation projects are abandoned or not completed within the planned timescale/budget [8].

Model-driven software modernisation (MDSM) is a set of strategies that use model-driven engineering (MDE) to address the above issues by (partially) automating the process of modernisation. These approaches have already proved feasible by Kowalczyk and Kwiecińska who demonstrated the modernisation of a project written in Java 1.4 and the Hibernate 2.x framework, which uses XML mappings, into Java 1.5 and Hibernate 3.x which uses Java Persistence API (JPA) annotations [8]. Additionally, a model-driven approach is employed in industry by the Sodifrance company who have migrated a large-scale banking system (at a size of around one million lines of code) to J2EE [9].

The Defence Science and Technology Laboratory (DSTL) at the Ministry of Defence (MoD) have identified several instances of software obsolescence they would like to tackle. A problem of particular interest is "the migration of an entire software system from a legacy hardware platform to a modern more powerful platform" [10]. This report will explore how MDSM processes can be used to partially automate the migration of code between microprocessors. A case study implementing the migration of code between a Parallax Propeller Activity Board and an Arduino Uno will be used to demonstrate the approach developed.

1.2 Project Goals

TODO

1.3 Project Scope

TODO

1.4 Report Structure

TODO

2 Literature Review

TODO: write introduction to the literature review chapter

2.1 Model Driven Engineering

Model Driven Engineering (MDE) was developed in order to address complexity within the problem space of computing. In the past, approaches for enabling programmers to develop code more easily have focused on simplifying the solution space. This was achieved by providing abstractions such as higher-level programming languages or providing operating systems to manage the difficulty of programming hardware directly. However, these solutions are unable to express domain concepts effectively, unlike MDE techniques [11]. Furthermore, it has been found that MDE can also be utilised in automating the software development process [12]. The rest of this section introduces the terminology and main components of MDE.

2.1.1 Domain-Specific Modelling Languages (DSMLs)

Meta-meta-modelling mechanisms such as the Object Management Group's (OMG) Meta Object Facility (MOF) can be used to create modelling languages for a given problem domain [13]. The key components of the OMG's approach to DSMLs are [14]:

Models: A model is a simplification of a system built with an intended goal in mind. Models should be easier to use than the original system. This is achieved by abstracting out details of the system that are unnecessary for the target model's intended purpose.

Meta-models: A meta-model is the explicit specification of an abstraction (i.e. model).

Meta-meta-models: A meta-meta-model is used to define meta-models. In particular, the OMG define the MOF (Meta Object Facility) which is a self-defined meta-meta-model.

2.1.2 Model Transformations

There are three main categories of model transformation:

Model-to-model (M2M): M2M transformations are used to translate source models to target models. The source and target models can be instances of the same or different meta-models [15].

Model-to-Text (M2T): These can be considered a special case of M2M transformations. It is often the case that M2T transformations are used for code generation. The most common approach to M2T transformation is a template-based approach. A template contains mixtures of static text and dynamic sections that can be used to access information from the source model [15].

Text-to-Model (T2M): T2M transformations can be used to transform code to a model (conforming to a language meta-model). Model discoverer tools such as MoDisco [16] are the easiest way to perform T2M transformations on code [8].

2.1.3 The Epsilon Framework

2.2 Microprocessor Background

2.3 Existing Work

2.3.1 MDE Approaches

2.3.2 Other Approaches

2.4 Tools

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] K. Kowalczyk and A. Kwiecinska, *Model-driven software modernization*, 2009.

[9] F. Fleurey, E. Breton, B. Baudry, A. Nicolas and J.-M. Jézéquel, 'Model-driven engineering for software migration in a large industrial context', in *International conference on model driven engineering languages and systems*, Springer, 2007, pp. 482–497.

[10] 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.

[11] D. C. Schmidt, 'Model-driven engineering', *Computer-ieee computer society-*, vol. 39, no. 2, p. 25, 2006.

[12] J. Bézivin, 'In search of a basic principle for model driven engineering', *Novatica journal, special issue*, vol. 5, no. 2, pp. 21–24, 2004.

[13] G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. Cheng, P. Collet, B. Combemale, R. B. France, R. Heldal, J. Hill *et al.*, 'The relevance of model-driven engineering thirty years from now', in *International conference on model driven engineering languages and systems*, Springer, 2014, pp. 183–200.

[14] J. Bézivin and O. Gerbé, 'Towards a precise definition of the omg/mda framework', in *Automated software engineering, 2001.(ase 2001). proceedings. 16th annual international conference on*, IEEE, 2001, pp. 273–280.

[15] K. Czarnecki and S. Helsen, 'Classification of model transformation approaches', in *Proceedings of the 2nd oopsla workshop on generative techniques in the context of the model driven architecture*, USA, vol. 45, 2003, pp. 1–17.

[16] H. Bruneliere, J. Cabot, G. Dupé and F. Madiot, 'Modisco: A model driven reverse engineering framework', *Information and software technology*, vol. 56, no. 8, pp. 1012–1032, 2014.