Investigating the feasibility and worth of migrating legacy systems

Candidate: Patrick Naish - pn3g10@zepler.net
Supervisor: Prof. Michael Butler - mjb@ecs.soton.ac.uk

UNIVERSITY OF Southampton

Introduction

Mainframe computing has long been the backbone of technological industry. Despite having fallen out of favour in deference to the modern client-server style of interaction, mainframe software still underpins a vast amount of modern systems. Despite the fact that there were an estimated 200 billion lines of COBOL in use in 2008 [2], COBOL is widely considered to be a dead language, and the fact that young computer scientists are primarily taught languages such as Java [5] and Python means that many are unwilling to learn older languages. Those developers who do know the languages required to maintain legacy mainframe systems are rapidly reaching the age of retirement and leaving the industry. Thus, it is important to find a means by which to modernise legacy systems to preserve their functionality, while allowing for maintenance.

Approaches to modernisation

The four main approaches suggested by Almonaies et al. [1] are:

Replacement The original system is rewritten in a new language or replaced with an off-the-shelf solution.

Reengineering Modern functionality is reverse-engineered into the legacy system.

Wrapping An interface is created to access legacy components as services without changing them.

Migration The legacy system is transitioned into a modern environment while retaining functionality.

The most common end-goal for modernisation is to be able to access the system as a **Web Service**[a].

[a] See `http://www.w3.org/TR/wsdl`

Techniques

References

[1] A Almonaies, James Cordy, and Thomas Dean. Legacy system evolution towards service-oriented architecture. In *International Workshop on SOA Migration and Evolution (SOAME 2010)*, pages 53–62, 2010.

[2] Datamonitor. COBOL - continuing to drive value in the 21st Century. Technical report, 2008.

[3] L. O'Brien, D. Smith, and G. Lewis. Supporting Migration to Services using Software Architecture Reconstruction. *13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05)*, pages 81–91, 2005.

[4] Harry M. Sneed. A pilot project for migrating COBOL code to web services, 2009.

[5] Harry M. Sneed and Katalin Erdoes. Migrating AS400-COBOL to Java: A Report from the Field. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 231–240, 2013.

[6] H.M. Sneed. COB2WEB a toolset for migrating to web services. *2008 10th International Symposium on Web Site Evolution*, 2008.

Tools

There are various tools available for aiding the modernisation effort. The examined tools fell into two categories: those for **analysis** of systems, and those for the actual **migration**.

An example of a tool for analysis is **COBAudit**, which collects metrics about COBOL programs [4]. These metrics can then be used to assess the complexity of a program, and said program's suitability for modernisation. **ARMIN** is a somewhat more specialised tool, which reconstructs a program's architecture from a static analysis of the code [3]. This can be used to identify dependencies within programs, which can then inform the creation of a migration strategy.

One set of tools for migration is **COB2WEB**, which comprises COBAudit along with COBStrip, COBWrap and COBLink [6]. **COBStrip** separates out a program into 'slices', which each represent a path through the program performing a distinct function. These are used to create cut-down versions of the original program performing only those functions, allowing for each piece of functionality to be represented by a separate Web Service. To this end, **COBWrap** adds an I/O layer to sit between the legacy code and the web, and **COBLink** generates Web Service Description Language (WSDL) interfaces for the same.

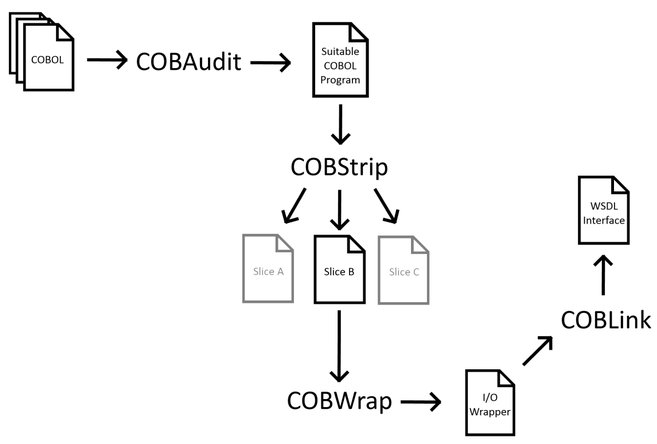

Figure 1: An illustration of the COB2WEB process

Conclusion