Chapter 7
An Agile Architecture for a Legacy Enterprise IT System

Chung-Yeung Pang
Seveco AG, Switzerland

ABSTRACT

Back in the 1970s, applications mainly comprised programs written in COBOL. Many of these applications are still in operation. To meet new business demands, new applications that have to collaborate with existing programs need to be developed. It is possible to have an agile software architecture that enables easy development, extension and maintenance in COBOL. Such an architecture, and the agile development process, are presented in this article. The architectural design is a combination of the layered, component-based and service-oriented architectural patterns. It also includes features such as the centralized control of the business process, plug-and-play autonomous COBOL modules and context container for storing state data. A model-driven approach is used in the agile development process. Application models include UML class diagrams, state charts and activity diagrams from which various software artefacts and COBOL codes are generated. The architecture and development approaches were first introduced in 2004 and have been successfully applied to 13 applications since then.

INTRODUCTION

Despite the advances in software technologies, software development remains a challenge for IT professionals. Statistics have shown that most software projects run late and over budget. The problem manifests to an even greater extent when a mixture of legacy and modern applications within a complex IT system comes into play. In fact, enterprise IT systems usually undergo a long period of evolution. Over the past decade, the software projects evolved into some of the most complex software systems. For large corporations, mainframe applications programmed in COBOL often form the backbone of the IT structure. Despite their obsolescence, legacy systems continue to provide a competitive advantage through supporting unique business processes and containing invaluable knowledge and historical data (Mitchell, 2012; Lauer, 2014).

DOI: 10.4018/978-1-5225-3422-8.ch007

Maintaining and upgrading legacy systems is one of the most difficult challenges many companies currently face. They struggle with the problem of modernizing these systems while keeping the day-to-day operation intact. Many large corporations have tried but failed to rebuild their legacy systems using an object-oriented language like Java or Smalltalk. At the same time, new applications developed in Java do not seem to provide significant advantages in terms of performance or enhancement in business agility. On the other hand, with a flexible but solid software architecture and a proper approach to development, one can build flexible, maintainable and agile applications in a legacy platform with a language like COBOL.

In recent years, the agile software-development process has become very popular in the software industry. This process takes the evolutionary and iterative approach to software development and focuses on adaptation to changes (Ambler, 2010; Larman, 2003). The approach eliminates a lot of problems in software projects (which will be discussed later in this article) and appears to be very promising. However, the process alone does not guarantee the success of a software project. There are many other factors involved. Attempts to just use the agile approach to software development can still fail (Harlow, 2014).

I have been taking an iterative incremental process and continuous integration approach to software development for many years. Over the last 15 years, many software projects have been led to completion by me. Some projects were under tremendous pressure, with time and budgeting constraints, and yet they were completed on time and within budget. In my experience, the key factors to success include an agile development process, software architecture and good teamwork among the members of the development team.

This article mainly contains my practical experience in such a software-development approach and architectural design. The aim of the article is to present an approach and techniques for software development that can contribute to the success of IT projects.

The article is organized as follows. The background section covers the historical background of software engineering. Following that section is a presentation of the agile development process, with the *Agile Manifesto* and its critics. (Agile Manifesto Group, 2001) next section is focused on software architecture, which, in my experience, is crucial to the agile process for enterprise application development. The design of our agile architecture for a legacy IT system follows. Infrastructures and tools are required to support application development using our software architecture. They are presented, together with our approach to software development, in a single section, which is followed by a section about our experiences and lessons learnt. The article then concludes and offers future research directions.

BACKGROUND

As background, the software engineering discipline, which emerged to counteract the well-known software crisis (Software Crisis, 2010) will be presented first. Following subsections address the software-development process and the challenges and successes of software projects.

Software Engineering

In the early days of software's history, programmers tended to develop their programs in an ad hoc style, with no documentation. A result of this was the software crisis of the 1960s, 1970s and 1980s (Software Crisis, 2010). Typical phenomena of the software crisis were: