Chapter 84
Analyses of Evolving Legacy Software into Secure Service-Oriented Software using Scrum and a Visual Model

Sam Chung
Institute of Technology, University of Washington, USA

Barbara Endicott-Popovsky
University of Washington, USA

Conrado Crompton
Institute of Technology, University of Washington, USA

Seung-Ho Baeg
Korea Institute of Industrial Technology, Korea

Yan Bai
Institute of Technology, University of Washington, USA

Sangdeok Park
Korea Institute of Industrial Technology, Korea

ABSTRACT

This chapter explores using service-oriented computing to reengineer non-secure legacy software applications to create new secure target applications. Two objectives of this chapter are: (1) to analyze the architectural changes required in order to adopt new web technologies and cope with resultant vulnerabilities in source code; and (2) to measure the level of effort required to modernize software by adopting new web technologies and adding security countermeasures. To meet these objectives, a model-driven Scrum for Service-Oriented Software Reengineering (mScrum4SOSR) methodology was chosen and applied to a reengineering project. Scrum is employed to manage the reengineering project, as well as to measure implementation effort related to the modernization process. Further, a re-documentation technique called 5W1H Re-Doc is used to re-document the non-secure software application at a high level of abstraction in order to help project participants comprehend what is needed to identify candidate services for service-oriented reengineering. Case studies with and without security features are created for different types of applications - a desktop graphical user interface, a web application, a web services application, a restful web services application, and an enterprise service bus application.

DOI: 10.4018/978-1-4666-4301-7.ch084

INTRODUCTION

The collected empirical results in this chapter show that the combination of Scrum and the visual model can help software reengineers to conduct, not only project management and candidate service identification, but also comparisons of modernization efforts in service-oriented reengineering: New architectures that incorporate web technologies do not, necessarily, bring serious complexity to a well-architected legacy system. Modernization from a web application to a Simple Object Access Protocol web services application requires less effort, compared to migrating to a Representational State Transfer web services application. When we incorporated security countermeasures to mitigate known vulnerabilities in a given legacy system, the requirements did not result in major architectural changes.

This chapter discusses how new web technologies and security requirements affect the process of reengineering non-secure and non-service-oriented legacy applications into secure and service-oriented target applications. Software reengineers are interested in understanding how new web technologies and application security requirements affect various software architectures and how much effort is needed to modernize legacy systems into target systems that use Service-Oriented Computing (SOC) (Erl, 2005, Papazoglou & Heuvel, 2007). The process for reengineering a non-service-oriented legacy system into a service-oriented target system is called Service-Oriented Software Reengineering (SOSR) (Chung, Davalos, An, & Iwahara, 2008), a special purpose software reengineering process (Chikofsky & Cross, 1990). It is different from other reengineering problems. SOSR requires identifying the components of a legacy system to be converted into services and how they should be converted. The method used for converting services depends on how the services are implemented. Further, converted services also should be rendered secure during this process.

SOSR can be done at an application, organizational, or cross-organizational level. In this chapter, we are interested in SOSR at the application level for service producers: software reengineering practitioners, who are interested in modernizing legacy applications, want to know what architectural changes new web services technologies bring to an existing desktop or web application. They also want to understand the level of effort required to migrate a legacy application to a target application that embraces web technologies and mitigates vulnerabilities in source code. Meeting these objectives requires methods for collecting information about the software architecture changes and the effort expended during reengineering.

To manage a software engineering project, several agile methods, such as Scrum (Schwaber & Beedle, 2002, Sander, 2007), XP (Hayes, 2003), etc. (Highsmith, 2002), have been broadly used by the software development community. Documentation methods such as 5W1H Re-Doc (Chung, Won, Baeg, & Park, 2009), ICONIX (Rosenberg & Scott, 2001, Rosenberg, Stephens, & Collins-Cope, 2005), etc., have been employed to help development participants comprehend a given application. Several other methods have been proposed to identify candidate services from a given legacy application by Gomaa and Shin (2009), Shin and Gomaa (2006), Peterson (2006), and Chung et al. (2009). Among these approaches, we chose a software reengineering methodology called Model-Driven Scrum for Service-Oriented Software Reengineering (mScrum4SOSR), which was proposed by Chung et al. (2009). The mScrum-4SOSR guides the reengineering process using burn down charts and re-documents each system at a high level of abstraction through a visual model. It also allows us to conduct analysis of the impact of new web technologies on the development process and identify vulnerabilities in legacy system code.

In mScrum4SOSR, both Scrum agile software development and 5W1H Re-Doc documentation methods are used. Scrum's unique agile method visualizes the reengineering process through burn