Chapter 7

Business Process Modeling with Services: Reverse Engineering Databases

Yousef Baghdadi
Sultan Qaboos University, Oman

Naoufel Kraiem
Sultan Qaboos University, Oman

ABSTRACT

Reverse engineering techniques have become very important within the maintenance process providing several benefits. They retrieve abstract representations that not only facilitate the comprehension of legacy systems but also refactor these representations. Business process archaeology has emerged as a set of techniques and tools to recover business processes from source code and to preserve the existing business functions and rules buried in legacy source code. This chapter presents a reverse engineering process and a tool to retrieve services from running databases. These services are further reused in composing business processes with respect to Service-Oriented Architecture, a new architectural style that promotes agility.

INTRODUCTION

Reverse engineering techniques have become very important within the maintenance process providing several benefits. Firstly, reverse engineering allows maintainers to retrieve abstract representations to facilitate the comprehension of different legacy systems. For example, it focuses on relational databases (Baghdadi, 2006; Cleve and Hainaut, 2008), aspect oriented systems (Bernardi, 2008), or quality of the system design (Foutse, 2009). Secondly, abstract representations obtained
by reverse engineering from legacy systems can be refactored to improve their maintainability or add new functionalities to evolve legacy systems.

To address the mentioned maintenance activities, reverse engineering techniques are nowadays well-supported by tools which often obtain artifacts at system design abstraction level (e.g., service, class or sequence diagrams from source code) (Baghdadi and Al-Bulushi, 2013; Canfora and Di Penta, 2007).

However, Software Engineering industry is demanding additional reverse engineering techniques and tools to retrieve business-aware artifacts at higher abstraction level (Khusidman and Ulrich, 2007). The Software Engineering Institute (SEI) argues that business rules recovery is the cornerstone to evolutionary maintenance towards modern paradigms like Service Oriented Architecture (SOA) (Lewis et al., 2010). In fact, enterprises willing to move to SOA with web services to challenge changes in business requirements need to modernize their legacy applications (Al-Rawahi and Baghdadi, 2005). Indeed,

1. SOA is mainly about reuse of assets (Cummins, 2009); in this regard, legacy applications are running smoothly and performing critical tasks,
2. Most of the business functions are locked within them (Galinium and Shabaz, 2012),
3. Legacy applications were built at high cost; and we need to preserve these investments (Linthicum, 2004), and
4. Migration to SOA can give new life to legacy applications (Bhallahmudi and Telly, 2011).

A solution consists in extending the critical business logic of the legacy applications while preserving the investments, through their migration to web services and SOA environments (Baghdadi and Al-Bulushi, 2013). This leads IT departments to select an appropriate modernization technique, which requires a guidance process to avoid any failure risk (Baghdadi and Al-Blushi, 2013).

The process would include analysis, selection of business functions, and wrapping. In addition, Business Process (BP) reengineering, innovation and improvement have always been given much attention in any organization (Davenport, 1993; Hammer and Champy, 993, Wekse, 2007; van der Aalst, 2012).

To meet these demands, Business Process Archeology (BPA) has emerged as a set of techniques and tools to recover Business Processes (BPs) from source code (Pérez-Castillo et al., 2011).

BPA studies the BPs in an organization by analyzing the existing software artifacts. The objective is to discover the business forces that motivated the construction of the enterprise information systems. Maintenance benefits of BPA are that they preserve business behavior buried in legacy source code and it retrieves BPs providing more opportunities for refactoring (Pérez-Castillo et al., 2011).

Some techniques in literature support BP recovery. For instances, Zou et al. (2006) recover workflows by statically analyzing source code and applying some heuristic rules to discover business knowledge. Paradauskas et al. (2006) retrieve business knowledge through the inspection of the data stored in databases. DiFrancescomarino et al. (2009) consider graphical user interfaces of web applications to discover BPs. Cai et al. (2009) combine requirement reacquisition based on use cases with dynamic and static analysis techniques. Finally, van der Aalst et al. (2009) focus on mining BPs from event logs registered during system execution.

Recently service-orientation starts making its way in information systems by promoting BPs as compositions of loosely coupled services having separated concerns (Baghdadi, 2012). For instances, in (Baghdadi, 2004; Baghdadi 2005), the author showed how to integrate BPs, namely B2B with web services that are derived from a business model. Jaeger (2006) established the relations between workflow modeling and BP for service composition. He explains the differences
Related Content

BROOD: Business Rules-Driven Object Oriented Design
[www.igi-global.com/chapter/brood-business-rules-driven-object/29434?camid=4v1a](www.igi-global.com/chapter/brood-business-rules-driven-object/29434?camid=4v1a)

Creating Applications for Real-Time Collaboration with XMPP and Android on Mobile Devices
[www.igi-global.com/chapter/creating-applications-real-time-collaboration/66501?camid=4v1a](www.igi-global.com/chapter/creating-applications-real-time-collaboration/66501?camid=4v1a)

The Exokernel Operating System and Active Networks
[www.igi-global.com/chapter/exokernel-operating-system-active-networks/37948?camid=4v1a](www.igi-global.com/chapter/exokernel-operating-system-active-networks/37948?camid=4v1a)

Using DRAM as Cache for Non-Volatile Main Memory Swapping
[www.igi-global.com/article/using-dram-as-cache-for-non-volatile-main-memory-swapping/144142?camid=4v1a](www.igi-global.com/article/using-dram-as-cache-for-non-volatile-main-memory-swapping/144142?camid=4v1a)