# Chapter 91
# Reuse across Multiple Architectures

**Indika Kumara**
*WSO2 Inc, Sri Lanka*

**Chandana Gamage**
*University of Moratuwa, Sri Lanka*

## ABSTRACT

*The commonality across software systems can be exploited to develop multiple heterogeneous systems successfully without undue cost, time, and effort. The systematic reuse across different systems is of paramount importance. With a well-planned reuse approach, a vendor can offer individualized products, which are products tailored to meet the requirements of a particular user effectively, as well as the products constructed to deliver solutions for a greater variety of application domains such as enterprise application integration and business process management. This chapter describes the development of software systems having different architectures reusing most of the implementations of the required functionalities as-is. It presents a systematic process for crafting multi-architecture reusable components and for using those components in formulating software systems. Furthermore, the chapter highlights the significance of the strategic reuse across systems in three contemporary research spheres.*

## INTRODUCTION

Software reuse has been widely touted as means of amortizing the cost of the software development. Two notable engineering disciplines that have exploited reuse are software product line (SPL) and component-based development (CBD). SPL has been employing strategic reuse to formulate product families successfully (Bass, Clements, & Kazman, 2003). CBD has incorporated reusability into the software development process to make software reuse a driving force in developing software systems (Crnkovic & Larsson, 2002). Although these reuse paradigms can establish a process for reuse, without purposefully architecting systems, components and other artifacts for reuse, the desired reuse goals would not be attainable (Griss, 1999). For instance, as not every architecture style facilitates reuse, the employed architecture for a particular component-based

DOI: 10.4018/978-1-4666-4301-7.ch091

system may impede the opportunities for reuse. Moreover, a new product can use an existing component; however, the component cannot provide the software qualities that the product expects.

The capability to produce a variety of software systems offers both vendors and customers discernible advantages. A customer can use a tailor-made product that provides minimum yet optimum functionalities required for implementing only her or his own business solutions. In addition, a customer can gain the flexibility to adapt the systems and applications to support the changing business goals of the organization. From a vendor's perspective, the software systems developed to provide solutions for different application domains such as enterprise application integration (EAI), business process management (BPM), etc., can potentially enable a vendor to reach to wide-ranging customers. However, the heterogeneity of the systems poses grand challenges on how to develop multiple systems cost and time effectively. Typically, software architectures, features, and technologies make systems different. In order to achieve tremendous savings in costs and time in producing multiple different software systems, a systematic reuse across the systems is crucial (Altintas & Cetin, 2008) (Griss, 1999), and the components that are reusable across the systems can provide the required reusability.

A critical issue undermines a sustainability of a software system is *architectural erosion*, a phenomenon in which a system's architecture decays over time to the point that it does not exhibit its key attributes. An eroded architecture is unwieldy and cannot adequately respond to the changes in requirements. Redesigning such architecture from scratch is practical compared with maintaining it (Gurp, 2002). A key reason for *architectural erosion* is that a system has to undergo an evolution to accommodate ever changing customer requirements. Architectural erosion can be stemmed if the system can reform its architecture to fulfill the new emerging demands (Bernhard, 2010). Therefore, the capability for changing the architecture of a system with either practical or economical ease is a promising solution for *architectural erosion*. An approach to achieve a systematic reuse across systems having different architectures paves the way for such a solution.

There is no considerable research related to the reuse across multiple architectures, and we observe that previous research has focused merely on functionality reuse and have not taken into account that a reusable component should be capable of supporting quality requirements required by various systems. Furthermore, most of the existing approaches do not provide a suitable solution to prevent *architectural erosion* that weakens the reusability and sustainability of the components and the systems.

The objectives of this chapter are to introduce the reader *Multi-Architecture Reusable Components*, and to discuss the techniques for designing and implementing them. We also present an approach for using them in producing multiple software systems successfully. Our experiment with two middleware systems shows that our approach provides a systematic reuse across different systems effectively.

This chapter is structured as follows. Section *Background* explores the opportunities for exploiting reuse across systems, and highlights related work. Next sections unfold our approach for achieving large scale reuse across multiple architectures. Finally, we present our conclusions and future research directions.

## BACKGROUND

Simply stated, software reuse across multiple architectures is systematically creating software systems having diverse architectures from existing software assets. In this context, a strategic reuse precludes the mere functionality reuse and promotes the reuse of functionality with the quality attributes required for a particular software system. Such reuse provides a software vendor with