Chapter 6

Achieving Effective Software Reuse for Business Systems

Daniel Brandon, Jr.
Christian Brothers University, USA

OVERVIEW

“Reuse (software) engineering is a process where a technology asset is designed and developed following architectural principles, and with the intent of being reused in the future” (Bean, 1999). “If programming has a Holy Grail, widespread code reuse is it with a bullet. While IT has made and continues to make laudable progress in our reuse, we never seem to make great strides in this area” (Grinzo, 1998). The quest for that Holy Grail has taken many developers over many years down unproductive paths” (Bowen, 1997).

This chapter reports on software reuse research (both literature research and design/coding research) and presents an approach for effective software reuse in the development of business systems. This approach is based on Object Oriented technology and provides for both the specification and enforcement of software reuse and corporate standards.

BUSINESS SYSTEMS

Business software systems are typically composed of three logical portions or layers as shown in Figure 1. The “presentation layer” involves the primary user interaction typically via a graphical user interface (GUI). The “business logic” layer provides database connectivity, validation, security, transaction control, and other sequencing or optimization control. This layer may be packaged by a vendor in an application or transaction server or written by a user. The “database layer” provides for the manipulation of persistent data, which for most business systems
today is stored in a relational database. The interface to this process is a well defined standard application programming interface (API) like ODBC or JDBC using SQL.

**NEED FOR REUSE**

Today’s software development is characterized by many disturbing but well documented facts, including:

- Most software development projects “fail” (60% [Williamson, 1997])
- The supply of qualified IT professionals is much less than the demand
- The complexity of software is constantly increasing
- IT needs “better,” “cheaper,” “faster” software development methods

“Object technology promises a way to deliver cost-effective, high quality and flexible systems on time to the customer” (McClure, 1996). “IS shops that institute component-based software development reduce failure, embrace efficiency and augment the bottom line” (Williamson, 1997). “The bottom line is this: while it takes time for reuse to settle into an organization – and for an organization to settle on reuse – you can add increasing value throughout the process” (Barrett, 1999). We say “object technology” not just adopting an object oriented language (such as C++ or Java), since one can still build poor, non object oriented, and non reusable software even using a fully object oriented language.

**TYPES AND APPLICATIONS OF REUSE**

Radding defines several different types of reusable components (Radding, 1998):

- GUI widgets – effective, but only provide modest payback”
- Server-Side components – provide significant payback but require extensive up-front design and an architectural foundation.
- Infrastructure components – generic services for transactions, messaging, and database … require extensive design and complex programming
- High-level patterns - identify components with high reuse potential
- Packaged applications – only guaranteed reuse, … may not offer the exact functionality required

![Diagram of three software layers: Presentation Layer, Business Logic Layer, Database Layer](image)
On Developing Hybrid Modeling Methods using Metamodeling Platforms: A Case of Physical Devices DSML Based on ADOxx
[www.igi-global.com/article/on-developing-hybrid-modeling-methods-using-metamodeling-platforms/123607?camid=4v1a](www.igi-global.com/article/on-developing-hybrid-modeling-methods-using-metamodeling-platforms/123607?camid=4v1a)