Chapter 3  OO Application Framework on Library Systems Domain: An Architectural View

The advantages of software architecture designed in multi-tier or multi-layer is a known fact that divides the complexity of different types of software components into manageable and relevant layers. Interoperability between library systems components in the application framework warrants us to look into the software architectural layers involved. Each layer must be looked into for the right choice of systems or components to form a scalable and flexible architecture to library systems domain application framework development.

3.1 Layered Architecture

A layered software system organises software into layers where each layer is built on top of another more general layer. A layer can be defined as a set of systems with the same degree of generality. A business IT architecture typically consists of a distributed system that is made of a layered software system.
Figure 3.1: A Typical Layered Software System.

The library systems domain application framework will be built based on the layered software system as shown in Figure 3.1, which is adapted from (Jacobson, 1997). A typical layered software system consists of four layers as follows:

- application systems
- business-specific
- middleware
- system software

3.1.1 Application Systems Layer

Different types of applications exist in the application systems layer. They are high-level components and classes that are placed on this topmost layer of a software architecture. The application systems developed with variable functions for the library domain belong to this layer. The application systems layer can have several distinct application systems whereby each application system could offer a coherent set of use cases to some end users. These application systems may interoperate directly with each other through their interfaces and may also interoperate indirectly through some services or objects imported from component
systems in the lower layers, such as the business-specific layer and middleware layer.

3.1.2 Business-Specific Layer

The business-specific layer consists of a number of component systems specific to the type of business. This layer reflects the application domain, and hence, is business specific. A domain framework will have component systems located in this layer that are imported by the variant and distinct application systems of the topmost layer.

The business-specific component systems offer services of a domain framework to the application engineers to build application systems from these reusable components which are specific to the domain. This layer offers high-level component systems of business abstractions that will fulfill the major business requirements.

3.1.3 Middleware Layer for Library Systems Domain

In many businesses, a networking environment is a vital presence to carry out enterprise-wide transactions with multiple connections and the different operating platforms depend on the middleware layer to communicate. The business-specific components are built on top of the middleware layer to achieve the distributed presence. The middleware layer actually offers component systems which provide utility classes and platform-independent services such as distributed object computing in a heterogenous environment like CORBA and Java. Architectural framework developers would use the middleware components for interoperability to
interact with systems on different platforms, or simply reuse technical frameworks such as utility packages to build business-specific components and application systems.

3.1.4 System Software Layer

The bottom layer is the system software layer, better known as the operating system which is the infrastructure to build component systems and application systems. Well-known operating systems such as Windows, Macintosh, UNIX and LINUX are situated in this bottom layer.

Application and component systems that are developed on a particular system software layer are dependent on the type of operating system because of the different hardware architecture. Java programming language has overcome this limitation for distributed system and application portability.

3.2 A Family of Library Applications

3.2.1 Library Application Systems Layer

There can be a family of library applications that make up the library application systems such as a primary school library system, public library system, commercial rental system, college rental system and secondary school library system. An application system family can reap the benefits of framework component reusability because of dependency and importation of common base classes and functions. As shown in Figure 3.2, several library application systems are developed by importing reusable components from the library component systems in the layered
system architecture. The <<import>> dependency between the library application systems layer and the library business-specific layer subsystems ensures that library components such as Borrower, Title, Loan, Reservation and Item are reused for a family of related applications that exist in different environments such as a primary school, secondary school, college, university or commercial rental centres.

![Layered System Architecture](image)

Figure 3.2: Family of Applications on top of a Layered System Architecture.

3.2.2 Library Systems Domain Layer

The library systems domain layer consists of library model packages and components that form the infrastructure of the application framework. The business objects are adopted as entity types of reusable components across related library systems. The commonality of all library processes is captured in this framework components for reuse and the variability is an extension left to the application

46
systems layer for customisation. The application systems layer will have different application systems developed on top of this common library systems domain components which form the data model in this type of business domain layer.

3.2.3 Middleware Layer for Library Systems Domain

Java middleware system offers utility packages such as collection algorithm, networking, relational database connectivity and servlets support that are imported by component systems across the domain. Since Java is a platform independent language implementation system, placing it in the middleware is a good choice for interoperability of distributed and multi-tier systems and offers substantial reuse to Java applications system family. CORBA, as an object request broker, is also a popular choice for a library distributed system because it offers systems interoperability, such as C++ language implementation which is platform dependent and Java.

By having the right framework architecture, a library system can perform in an efficient and widespread manner with good reusability of framework components across the business enterprise.

3.2.4 System Software Layer for Library Systems Domain

The choice of operating system for the system software layer is very much the choice of framework developers, but bear in mind that its support for Java has to be a major consideration if the language is the choice of framework implementation. This
is due to the fact that platform independence for Java offers a single and coherent solution even with the mix and match of different operating systems.

Any robust network operating system such as Windows NT, UNIX or Novell is also a good choice for a library system because of its centralised application and database server connected to multiple distributed users of the library system.

### 3.3 Summary

The primary aim of the layered system architecture is to identify the underlying system platform to support the development of an application framework, such as the system software layer which has a specific operating system supporting feature and "look-and-feel", while the middleware layer provides the same technical library of language development such as Java Collections Framework. The library business-specific layer is comprised of the library systems domain components which function based on the underlying system layers. Application developers will find such architecture easy to learn, deploy and compatible with additional add-on components for developing the library application systems.