

Chapter 1 Introduction

In software engineering, like other conventional engineering fields, every project begins with careful analysis of expected requirements and variable changes to a software system. The software engineers would first analyse and design a software system according to proven safety standards which incur most minimal costs, avoiding frequent maintenance and planning to develop a stable system. At most times, certain software artifacts in the analysis and design can be used again in future projects. Reusable analysis and design artifacts are more reliable as they have been tried and tested.

In fact, the designs of certain software artifacts are commonly reused over and over again when the designers are comfortable with them as long as the requirements are similar, or they can be expanded to meet new additional requirements. Methods that are similarly designed from different projects can be identified into a design pattern. A combination of design patterns on a business domain will provide a powerful architecture to build a business domain framework. The interactions between various business objects are based on the design patterns incorporated into a framework and the interactions work in a generalised and consistent way. These design patterns can help application developers to quickly learn how to use and extend a framework (Carlson, 2000).

The framework architecture is planned with analysis and design that lays on an adaptable foundation. The framework should be highly scalable so that it can be used for related applications in the business domain and yet remains stable from changes. For example, a retail chain which rents items, a variety of books in small to big libraries and a business that loans items to its members, are examples of scenarios that can be incorporated a common application framework and scale it to accommodate changes. The design of a framework has the biggest influence on its scalability. An object-oriented (OO) application framework on library systems domain can assume some basic processes such as items are loaned, reserved or returned.

By modelling software in logical components, we include business policies at a higher level of system development, such as analysing the business domain before we eventually progress to lower level physical components such as building software components with the right interface and design for implementation (Andersen, 1998). Software designers take a logical perspective when they model the business domain in a framework architecture that is comprised of software components as the building blocks. The logical abstraction of key business policies in a framework is defined in details when software designers define the objects and functionality to be implemented. These individual software components are generally useful across a range of applications in the business domain.² These useful attributes appear as generic design features.³ Software components are packaged as black boxes with capabilities that are displayed as well-defined interfaces.

A framework can encourage its users, such as application developers, to think of how the mechanisms can be applied to solve a business problem instead of worrying about the details of implementation. The mechanisms are displayed by its design

patterns. A good framework can provide documentation that explains about the available patterns and the patterns are quickly recognised. Frameworks offer high-level reuse which is more flexible to changes and more reusable because of a layered architecture development and patterns that are generic designs adaptable to business rules as opposed to low-level reuse.

1.1 Motivations

This research project is motivated by several problems and needs in the library application systems development and the advancement in framework development. There are several kinds of library systems at different levels of education such as elementary school level, high school level, college level, university level and general public. All these libraries have many common and somewhat different activities.

Without a library systems domain application framework, most existing library systems are developed from scratch. Library application systems do not have to be repeatedly built from scratch because the design infrastructure or framework is similar except for some specific activity requirements. A faster development time and better cost saving can be achieved from using a library systems domain application framework as it offers this reusable infrastructure. In addition, application developers can avoid developing a library system from scratch for a library resource centre.

A library systems domain application framework can speed up the development, reduce cost, increase efficiency and reliability of a library system because reusable software components are plugged into every library application developed from a common framework. The framework architecture serves as a blueprint that is reused

as a design pattern and is reliably used over multiple applications in the same domain.

Research in reusable software architecture specifically on application framework is still in its infancy. The purpose of this research project is to experiment, explore and contribute some ideas in this research field.

1.2 Problem Statement

A project is undertaken to develop an object-oriented application framework for library systems domain. With this application framework, we can develop different library systems applications which may be similar in the common working features but vary in some other specialised needs that can be developed by simply reusing appropriate reusable components of the application framework. The size of a library system may be different such as providing multiple types of resources, for example, books, magazine, CD, audio visual apparatus, video tapes and others, which may be on loaned or rented. For instance, the following library systems would be in different environments and context.

- A school library system may be small which is to serve only school children and no registration fee is involved.
- A public library system which is big in size, and which provides loans of resources to the general public who are registered members.
- A university library system which serves university students and staff. Fines are imposed for late return. Reservations can be placed for a resource.

- A commercial resource centre that rents out things, such as a video store that imposes a rental fee for a period of time and fine for late return.

Although the above-mentioned library systems are in different scenarios, they share the same underlying features and system requirements. The domain component systems of an application framework can fulfil these common needs.

1.3 Objectives of Research

The main objective of this research is to develop an object-oriented library systems domain application framework and build a library application by reusing software components of this framework. This objective will then show the usefulness of the library systems domain application framework to application developers, so they do not have to write code from scratch which is time-consuming and error-prone. This is accomplished from the fact that the software engineering field can emulate the electrical engineering field by developing a working application from interfacing software components that are pre-built. Once the right framework architecture with important key business process abstractions are analysed and designed, we will develop a library application on top of the framework components system architecture to convince the benefits of high degree software reusability that a framework can provide. With that, we would have developed a generic and reusable framework.

The next objective is to provide a good software architecture for the framework by designing a layered system architecture that is suitable for most of the library systems domain application framework infrastructure. In this case, the identifications

of supporting software development efforts on the different levels are a prerequisite to a good design architecture.

The last objective is to demonstrate a faster software application system development approach through reusing a framework architecture by interfacing with framework components. The end process of an application framework development is to provide a convenient means to application developers to use it and avoid the steep learning curve to adopt it. A wizard software tool will be developed for the application framework to provide step-by-step assistance in framework component selection and configuration to a default application for quick deployment of the framework.

1.4 Outcome of Research

The expected outcome of this research is an application framework prototype that is reusable across several library-related application systems. It is to show that software components are meant to be generic if they are worthy of reusability. The key abstractions of the main features should be in a framework design so that it will be used as a foundation layer by related application systems.

1.5 Thesis Organisation

The thesis is organised as follows:

- Chapter 2 consists of a literature review on definition of a framework, identifying domain framework, descriptions of software components, survey of library systems domain and identifying abstractions of the library domain. This chapter will conclude with a proposal to the current research project.

- Chapter 3 presents an architectural view of the library systems domain application framework which consists of a layered software system. The superordinate design model and its object model of the framework will be included and described in this chapter.
- Chapter 4 presents the analysis and design of the application framework. Domain analysis is conducted on different library environments. The use case model of Unified Modeling Language (UML) is used to capture the requirements of the framework, as well as its analysis and design.
- Chapter 5 describes the framework implementation including its evaluation and testing.

Finally, chapter 6 which is the concluding chapter will describe the strength, limitations and future work of the library systems domain application framework.