Chapter 6 Conclusion

To develop an application framework for a group of related application systems requires analysis of the domain. Therefore, any library related application system shares common domain analysis. An application framework portrays this common domain of libraries. The best current practice in modelling the library domain is the use case technique. Domain experts who are not software developers can understand the use case model easily due to its simplicity. The act of getting them involved is important to capture the core requirements of a reusable library system without error. In fact, it is encouraged to also get them involved in the system analysis stage and be educated on the static class diagram to express their clear opinions on the requirements.

A distributed environment of library system domain where the application is deployed over the Internet or a campus linked up by a network requires a layered system architecture. Every selected architecture layer of framework development is cause for concern to ensure compatibility and reusability over related systems. The selection of Java as the development language is influenced by its portability, platform-independence feature and a common language for remote object invocation by networking communication without requiring the object request broker such as CORBA for inter-communication of disparate systems.

The design of an application framework for library systems will improve over time, as more insight is revealed about the patterns of activities that typically happens in
the domain. Over time, newly discovered library features that are useful can be added into the framework for system upgrades. A framework development is an iterative process that can be refined. By separating operations of a system from the user interface that invokes it will produce a more maintainable application with higher cohesion and less coupling. A group of library classes in components is a neatly arranged set of related classes that has some internal dependency and it declares an interface that is recognised with a unique facade to deal by other components. The application framework components are used as the architecture foundation of development of a library system application.

With an application framework, application developers can focus more time on solving the business domain problems instead of programming problems concerning the library architecture foundation. A framework incorporates common abstraction of business requirements, good design practices that are recognisable, and capture of domain expertise. As such, it provides application developers the assistance and expertise to integrate the framework into an application and quickly package and market their product. An application framework has less maintenance concerns because of its reliable design that is consistent with the framework architecture across related applications. Code reuse with an application framework would be significantly achieved for the library systems domain by adopting the application framework as an architecture of application system development. Hence, this code reuse approach reduces development effort and time in complex applications and problem domain of library systems. As compare to small-scale class libraries which are without a domain architecture, software reuse is significantly lesser than would be for an application framework.
The library systems domain application framework consists of the business rules that can be dynamically configured to meet the current requirements which may vary in future. The end result of library applications built from a framework is a better integrated system that is especially helpful to application component engineers and system integrators who can gain increased productivity through software component reuse, more manageable development and an improved ability to deal with requirement changes in the business area. Subsystems can be integrated together to build the enterprise-wide systems because application frameworks can also be integrated. The reusable library system is designed to be scalable and flexible.

Software components packaged with published interfaces can be integrated to other software components to achieve the aim of plug and play. Software integration has come close to the ways of producing products that resemble electrical products built from components.

6.1 Strength of Research

The main strength of the library systems domain application framework lies in its business analysis, design and process commonality which can be reused in the form of software components in other related library application systems development. The salient point is to recognise software reusability of the particular business domain architecture.

The development of the application framework with a default working application realises the benefits and potentials of the framework software components and subsystems by offering domain expertise in an easily reusable way.
The sample working application encourages application developers to quickly understand the features that the framework offers and reduces the time to learn it.

The development of a wizard tool system and its inclusion into the final framework system offers step-by-step instructions to framework users to make selections of library system variable requirements that are specific to the local needs. The first exposure of the application framework is the wizard as part of the initial system set-up installation of framework components and/or a default application.

Another strength of the application framework is the inclusion of library business rules as system parameters in an initialisation file that can be edited for configuration to end-user requirements which are variable features of specific application and localised systems.

The use of Java has many benefits, such as its portability across different operating systems and this lessens the need for an interoperable middleware brokering system. Java also offers a collection framework for ease of development where the data structures are deployed as utility classes, methods and interfaces. A library domain entity type, such as, Loan is implemented as a class of compound attributes and collection objects that can be easily traversed, inserted, deleted or searched with the assistance of the Java Collection framework.

The use of a Java-based object-oriented database, such as POET, is a good point because it is seamlessly integrated into the development language and is able to capture all the objects and its relationships.

99
6.2 Limitations of Research

This research project studies the aspect of developing an application framework and its implications in terms of software reusability especially of framework components. In order to achieve a more significant degree of success in software reusability, the sample application developed on top of the framework components should, in future, incorporate networking features and reusable across enterprise-wide library domain. The framework analysis and design, particularly the layered system architecture, will be more convincingly reliable and accurate for further reuse without major complications in its network architecture and design.

Most software development is iterative in nature to improve on the requirement features and reliability. The lack of time to fully test the interface subsystems and components integration is cause for concern. A framework which consists of reusable components must be cohesive and well-defined to make it very usable.

The balance of generic, commonality, and variability in a framework design is important because we should capture only the common requirements into the framework components and yet remain reasonably useful without being too simplistic to avoid too much intervention to the application developer and modifications to the source code. This optimum framework design is not yet attainable at this early framework development iteration life cycle. The initial expected or unexpected variations in a reusable library system can sometimes become a common requirement later that warrants inclusion into the framework component system design.
6.3 Future Work

The library prototype application system developed from the library systems domain application framework is an independent system. A future development effort can be planned for a two or three-tier distributed system so that the layered approach to analysis and design of application framework as recommended by Application Family Engineering (AFE) (Jacobson, 1997) can be fully implemented and tested. The subsystems can reuse common components from the middleware layer, business-specific layer or application systems layer. By doing so, we can truly benefit from framework component software reusability significantly. One of the future plans is to create a window's interface for system configuration so that application developers can flexibly set dynamic initialisation values without changing the compile-timed source code.

The future work on the framework is to improve on the variation points that exist or may exist in a use case depending on the library activity scenario. Some variation points are more significant than others in library systems domain requirements, hence, they can be converted into an abstraction model which analyse the domain entity types or use case activity in greater detail. An application system can be built with component systems that are robust, quickly added and meet the application developers expectations with extensions and specialisation on the variable requirements.

6.4 Concluding Remarks

The benefit of high-level software reusability is significant in terms of quantity and quality, if specially intended for a library domain. This is because a domain-
specific application framework modelled and designed in a layered system architecture offers a higher-level reuse compared to black-box and class library code reuse. With the superordinate model of analysis types and design classes (and types), the library systems domain's high-level abstraction requirements can closely correspond use case models to implementation models. By developing a stable system layered architecture that captures the framework design abstractions into package components with interfaces and facades will simplify understanding of the business model and reuse process. Hence, the Application Family Engineering process assists framework developers in producing component systems that application system developers can reuse from a framework. Library systems domain application framework components can be reused in library-related application systems as long as components have compatible interfaces and facades which allow adaptable configurations. The configurable library systems requirements are designed to accommodate variation points, customisations and assembly of components for an application system and is assisted by a wizard tool as part of the framework utility. Therefore, we can conclude from this research project that in a specific domain, an application framework can result in rapid application system development from software reuse and flexibility in customised requirements.