Application Frameworks Technology in Theory and Practice

Wusheng Zhang

Mik Kim

Application Frameworks Technology in Theory and Practice

Wusheng Zhang[1] and Mik Kim[2]
Centre for International Corporate Governance Research[1]
Centre for Hospitality and Tourism Research[2]
Victoria University, Melbourne, Australia
Email: wusheng.zhang@vu.edu.au mik.kim@vu.edu.au

Abstract: Application frameworks is a technology concerning with building and implementing reusable software artefacts. Most current application frameworks are object-oriented and often domain specific. Advocates of application frameworks claim that the technology is one of the most promising technologies supporting large-scale reuse, increasing the productivity and quality, and reducing the cost of software development. Many of them project that the next decade would be the major challenge for the development and deployment of the technology. The objective of this study is to investigate the theory and practice of application frameworks and to determine if it has made a difference in systems development. The study indicates that the technology is still immature and not yet to be another silver bullet but potential is imminent.

I. Introduction

Software development markets expect the developers or development companies to deliver quality products at an affordable price within a required time frame. The developers and management alike are looking for technologies that can be used to increase the productivity and quality of the software products. Matured engineering disciplines, such as automobile design, have proven that reuse is the best way to increase the quality and productivity of products. However, despite the effort of last decades-long research the result of software reuse is still limited in code or class reuse (also known as small-scale), and developers are still 'reinventing the wheels'. Application frameworks are a technology anchored in this situation to promote reuse in terms of not only the code or class but also the module and architecture (also known as large-scale) of the reusable software artefacts to increase software productivity and quality.

II. Application Framework Technology

The Microsoft Encarta English Dictionary 2001 defines framework as " a structure of connected horizontal and vertical bars with spaces between them, especially one that forms the skeleton of another structure; a set of ideas, principles, agreements, or rules that provides the basis or the

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 769 - 776.

outline for something that is more fully developed at a later stage." The common sense of the word of framework appears to be "the skeleton of another structure", which has been well adopted into the context of modern information systems development. Booch, Rumbaugh and Jacobson [4] define a framework as "an architectural pattern that provides an extensible template for an application within a domain". In this context a framework is essentially a design skeleton that allows systems developers to create part of a system in the first place, and add design details when necessary. Johnson [23] states that the definitions of frameworks vary, but the one used most is that "a framework is a reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact". Another common definition is "a framework is the skeleton of an application that can be customized by an application developer". The former concerns the structure of a framework while the latter describes the purpose of the framework. Lewis et al. [24] argues that a framework is more than a class hierarchy. Fayad [14] claims that a framework is a reusable, 'semi-complete' application that can be specialised to produce custom applications. Zamir [34] defines "an object-oriented framework is the reusable design of a system or subsystem implemented through a collection of concrete and abstract class and their collaborations. The concrete classes provide the reusable components, while the design provides the context in which they are used. " The concepts of frameworks and application frameworks are often used interchangeably in the context of systems development. The definitions by different researchers or authors vary, some of them are more abstract and concerning more on the analysis and design phase, and others are more interested in design and development phase. The different emphases does not conflict each other but rather enrich and enlighten the further research issues related to the field of application frameworks technology.

Application frameworks can be domain independent such as graphical user interface (GUI) framework or domain dependent/specific such as CIM framework. They can also be classified according to the scope, reuse perspective, the control aspect and the development process of application frameworks. According to the scope of application frameworks, Fayad [17] proposes to classify it into three categories namely as system infrastructure frameworks i.e.

graphical user interface (GUI) and Microsoft Foundation Class (MFC); Middleware integration frameworks i.e. BEST and JAWS; Enterprise application frameworks i.e. SEMATECH CIM, OSEEFA and PRM. He claims [15] that application frameworks are generally domain specific applications such as computer-integrated manufacturing frameworks, distributed systems, networking and telecommunications, or multimedia collaborative work environment. From the perspective of the reuse application frameworks can be classified into whitebox and blackbox frameworks [33] [17]. Fayad [17] also proposes a greybox approach, which is a mix between the whitebox and blackbox frameworks. A whitebox application framework is a framework customised by subclassing existing framework classes and providing concrete implementations. To implement whitebox frameworks application developers use more inheritance and polymorphism. Application specific functionality is expressed by inheritance and new implementations. Implementation inheritance tends to require knowledge of the superclasses' implementations.In the last few years application frameworks researchers are more interested to develop blackbox application frameworks, which rely more on composition rather than inheritance. In blackbox frameworks approach, the extendability of the framework is achieved by defining interface for components that can be plugged into the framework using composition. Object composition is based on forwarding rather than delegation, merely relies on the interfaces of the involved objects. In a blackbox framework [17], an application developer selects from the set of subclasses provided by the framework as blackbox components and binds it to the hot spot (plug in point). Thus, the developer may create an application without programming, merely by selecting, configuring, and parameterizing framework components. From the perspective of taking control application frameworks can also be classified as callable framework and calling framework [17]. A callable framework allows the application to retain the thread of control and provides services when the application calls the frameworks. A calling framework provides a control loop that calls application-provided code at appropriate times. From a development process perspective application frameworks can be divided into analysis frameworks, design frameworks, and implementation frameworks. Analysis frameworks typically focus on analysis level constructs, without making any commitment. They are typically the product of domain analysis. Most of the current application frameworks are either a design framework or an implementation framework.

Most of the application frameworks are domain specific such as a financial application framework or a manufacturing framework. An application framework domain is a set of rules and roles and their semantic models codified in the framework itself. It provides a generic incomplete solution to a set of similar problems within an application domain. Fayad [17] states that an application framework embodies generalised expertise in the domain based on analysis and synthesis of a wide range of specific solutions. He argues that analysis and synthesis of a wide range of specific solutions will help to understand a design of the proposed application framework. It is proven that the research community has more understanding in some domains such as financial, manufacturing, communication and networks and social welfare than others [13][17].

The development of application frameworks research is related to the development of objects technology although there is no evidence of that the technology is exclusive to objects technology. However, the majority of the researchers in that area of application frameworks and most of the current application frameworks are object-oriented. Iterative and incremental development approaches have been the main development methodology supporting the development of application frameworks [15]. Application frameworks design can be bottom-up and pattern driven or top-down and target driven [17][31][33]. The bottom-up design works well where an application framework domain is already well understood. Starting from proven patterns and working one's way up has the advantage of avoiding idiosyncratic solutions in the small, problematic solutions that should be replaced by application of an established pattern. Top-down and target driven approach is preferable where an application framework domain has not yet been sufficiently explored but where the target domain to be served by the framework is well understood.

Research [15][17][24][29] has indicated that building application frameworks is hard and implementing application frameworks is as hard as building application frameworks. According to a survey [17] the minimum time spent in developing an application framework was half of person month and the maximum time to develop an application framework was 1000 person months. The average time to develop an application framework was about 21 person months. An application framework conventionally consists of the core classes of an application, and one has to understand the basic architecture of a particular application type to be able to specialise the framework [29]. Using an application framework may simplify application developers' life since a framework provides generic solutions for a particular application domain. However, the average learning time is a big factor in establishing the cost of final application. The application developers have to understand what solutions the framework provides, and to comply with the rules imposed by the framework.

Chen [8] promotes that reusable components and frameworks must be accumulated in a standardised format. Most of the researchers agree that the classification structure of an application framework must be appropriate and manageable. Application developers will have difficulties with understanding the framework if the structure of the framework is not clear and standardised. Currently literatures indicate that application frameworks still lack of standards. Fayad [15] [16] [17] states that building and implementing application frameworks still need more methodological support.

There are a few domain specific application frameworks have been used in the industry, for example, San Francisco framework, SEFA framework and SEMATECH CIM framework. The San Francisco framework is an application framework based on Java technology to develop business applications. It provides both a software solution for implementing business applications and a collection of concepts or strategies to develop business applications. According to the white paper from IBM [22] that the San Francisco framework is motivated from over 130 software vendors. The project was started when several software vendors asked IBM to help modernising their application products but there were several barriers preventing them from being able to update their applications. IBM claims that the San Francisco framework can solve 70% of the business related problems and leave 30% to the applications developers. IBM also claims that the framework makes it easier to move to object-oriented technology because developers use well-tested services instead of building their own. The project helps to solve these problems by offering developers Business Process Components. It is designed as a framework that provides an object oriented infrastructure, and a consistent application programming model and some default business logic. The San Francisco Framework is designed to make many types of extensions easy for application developers, allows overriding the default business logic in supplied methods, and adds additional attributes to existing classes. The San Francisco Framework offers three layers of functionality such as business processes, business objects, and foundation classes, each of which may be used and extended by developers to build their applications. OSEFA (German: Objektorientierter SoftwarebaukastEn für FertigungsAnlagen) is a blackbox application framework. It contains models of technical components as well as models of conceptual task-related components, and of application logic components. The prototype of the project was built in 1993 and the full version of the framework became operational in early 1996.The architecture of OSEFA has five layers, a machine and communication specific layer, a standardised machine layer, a task specific layer, a part processing layer, and a machine order layer [31]. He claims that through the project of OSEFA they have demonstrated how to use design patterns to create a sophisticated class structure from a simple result of object-oriented analysis. OSEFA [32] discloses the cost issues and states that developing a blackbox application framework takes around two to three times as much effort as developing a fixed application from the domain. This number may vary with the domain and includes the costs of analysing the domain and sufficiently generalising the class structure, but not the cost to acquire knowledge of framework structure and design. OSEFA claims that developing an application framework is worth the effort. The investment will pay off after the creation of about the third application from the framework. OSEFA experience indicates that another advantage of using application frameworks approach is the short time span required for the creation of a customer-specific or product-specific application from a framework. SEMATECH is the semiconductor manufacturing technology consortium, whose member companies are AMD, Digital, Hewlett-Packard, IBM, Intel, Lucent, Motorola, National Semiconductor, Rockwell, and Texas Instruments. CIM is an instance of callable framework and component is the smallest unit of functionality that can be added, deleted, enabled or disabled in a CIM framework- compliant application. The goals of CIM include flexibility, interoperability, substitutability, integration and reuse [17]. Standardisation of CIM framework was started by the Semiconductor Materials and Equipment International (SEMI) organisation in early 1997. In 1998 the SEEMI Global CIM Framework Task Force initiated two successful letter ballots, resulting in adoption of the first two parts of the CIM framework standard. The first part, the Provisional Specification for CIM framework Domain Architecture establishes the architecture foundation for the component structure and partitioning, and identifies the responsibilities of each major component of the framework. The second part is a document- Guide for CIM Framework Technical Architecture, defines required infrastructure technologies needed to support framework and their customers must be prepared to make [17]. The experience from CIM [17] indicates that implementation experience is essential; frameworks increases initial cost; infrastructure coupling is hard to avoid; frameworks overlap; technology is immature and standardisation is an important issue for industry wide application frameworks such as ICM.

III. Foundations of The Application Frameworks

Though application frameworks is not exclusive to object-oriented community the majority application frameworks are developed and implemented using object oriented technology. Object-oriented technology is one of the fastest growing technologies of the last two decades promising better quality, productivity and interoperability through software reuse. Coad and Yourdon [9] define “an object is an abstraction of something in a problem domain, reflecting the capabilities of the system to keep information about it, interact with it or both”. In that sense objects are used to model an understanding of the application domain, which concerns the system and abstraction. Deitel [11] defines “Object technology is a packaging scheme that facilitates the creation of meaningful software units”. He explains that these units are usually large and focused on particular application areas and most of them can be reused. For example, there are data objects, time objects, audio objects, video objects, file objects, record objects and so on.

The central idea of the object-oriented technology subsumes abstraction, modularity, encapsulation, inheritance and polymorphism - concepts that, on the face of it, lend themselves to reuse. The notable development of the technology consists of a comprehensive set of object-oriented modelling methods for analysis, design, and implementation, designed to realise the concepts mentioned above. Consequently, object-oriented technology has led to the development of patterns, components and application frameworks. The object-oriented concepts have been applied in the process of developing and implementing application frameworks. Fayad [15][16][17] stresses that frameworks build upon object-oriented concepts, which provides a conceptual base for more complex programming constructs and reusable implementation structures for large systems application. Eliens [13] states that object oriented approach will pay off when we have arrived at stable abstractions from which we have good implementations, that may be reused for a variety of other applications. Application frameworks is a technology aimed to achieve large-scale reuse by applying object-oriented concepts. In the following sections some of the object-oriented concepts and principles will be explored in relation to application frameworks and systems development.

IV. Application Frameworks and Other Reuse Techniques

Application frameworks is a reuse technology aimed at large-scale reuse and it has a close relationship with other reuse techniques used in software engineering. An application framework is a collection of components, a generic solution for a class of problems, a frame of mind for solving problems and a set of architectural constraints. An application framework integrates and concretises a number of patterns to a degree required to ensure proper interleaving and interaction of participants involved. An application framework is a kind of library, which provides reusable objects for applications but in contrast to ordinary software class libraries, frameworks may at times take over control when the application runs. From reuse perspective the application frameworks technology is closely related to other reuse techniques such as patterns, class libraries and components. Application frameworks use those reuse techniques mentioned above to achieve the goal of large-scale reuse.

IV. 1 Architecture

Software architecture is the foundation of system construction. Graham [21] defines "Software architecture deals with abstraction, with composition and decomposition, and also with style and aesthetics. Bass et al. [3] describe, "The software architecture of a program or computing system is the structure or structures of the systems, which comprise software components, the externally visible properties of those components and the relationships among them". While, Szyperski [33] depicts "System architecture is a means to capture an overall generic approach that makes it more likely that concrete systems following the architecture will be understandable, maintainable, evolvable, and economic. It is this integrating principle, covering technology and market that links software architecture to its great role model and justifies its name". Despite the different concentration of the definitions above a software architecture is about an over view of a system. Generally speaking software architecture can be seen as a set of rules, guidelines, interfaces, and conventions used to define how components and applications communicate and interoperate with each other. Recent software development experience has proven that sound software architecture for the software systems is necessary as software systems are more complex than before. Szyperski [33] stresses that architecture prescribes proper frameworks for all involved mechanisms, limiting the degree of freedom to curb variations and enable cooperation. Architecture needs to be based on the principal considerations of overall functionality, performance, reliability, and security. Software engineers have learnt from practice such that architecture is needed in any systems if they seek for guiding rules for design and implementation.

Architecture needs to create simultaneously the basis for independence and cooperation of systems. Independence of systems aspect is required to enable multiple sources of solution parts. Cooperation between these otherwise independent aspects is essential in any no-trivial architecture. System architecture is the structure of a software system, which provides a platform for application developers to build the system. It may be as concrete as providing detailed implementation requirements to as abstract as given a generic idea of how the system should be implemented. Application frameworks technology promises reuse of not only the frameworks source code, but also more importantly architecture [15]. A standardisation structure allows a signification reduction of the size and complexity of code that application developers have to write.

IV. 2 Class libraries

Class libraries is as set of reusable classes, often defined as part of the implementation or design environment [34]. Many programming languages have some ready usable classes embedded available to application developers especially visual development such as VB Studio.Net and J2EE. Class library in general offers static inheritance facilities but framework is more likely to support dynamic, run time binding facilities. Application frameworks defines 'semicomplete' applications that embody domain specific object structures and functionality. It can be viewed as extensions to object oriented class libraries. In contrast, class libraries provide a smaller granularity of reuse. For example, class library components like classes for strings, complex numbers and arrays are typically low-level and more domains independent. Fayad [18] states that class libraries are typically passive and frameworks are active and exhibit 'inversion of control' at runtime.

IV. 3 Patterns

Classes and interaction structure of object-oriented designs may become fairly complex, and consequently difficult to develop and understand, which have led the study and development of patterns. Design patterns are standard solutions to recurring problems, named to help people discuss them easily and think about design. Design patterns can be used as a micro-architecture that applies to a cross-domain design problem such as linked list and other classical data structure design. A design pattern describes a concrete solution to an architectural problem that might arise in a specific context. The solution proposed by the patterns is typically a way of structuring a cluster of objects and their interaction [7]. Schmid [31] states that the repetitive use of design patterns created an overall architecture though each design pattern represents a micro architecture. He argues that design patterns give a better, since more concrete guidance for how to realise a framework. Patterns are abstract, there fore, are not ready-made plugable solutions. They are most often represented in object-oriented development by commonly recurring arrangements of classes and the structural and dynamic connections between them. Graham [21] argues that patterns are most useful because they provide a language for designers to communicate in. In particular, design pattern have proven their value in structuring the variable parts, called hot spots (allow plug in software artefacts) of a framework [28]. Fayad [17] defines patterns as a conceptual solution to a recurring problem. Schmid [31] argues that design patterns are an excellent means to describe the details of object and class interactions but they are not suited to give an overall picture. Design patterns are reusable architecture, object template, or design rule that has been shown to address a particular issue in an application domain [34]. Most of design patterns come either as a static description of a recurring pattern of architectural elements or as a rule to apply dynamically for when and how to apply the pattern. The majority of software patterns produced to date have been design patterns at various levels of abstraction but Fowler and Graham [20][21] introduce the idea of analysis patterns as opposed to design patterns. Fowler's patterns are reusable fragments of object-oriented specification model generic enough to be applicable across a number of specific application domains.

Patterns and frameworks both facilitate reuse by capturing successful software development strategies. The primary difference is that frameworks focus on reuse of concrete designs, algorithms, and implementations in a particular programming languages. In contrast, patterns focus on reuse of abstract designs and software architectures. Frameworks can be viewed as a reification of families of design patterns. Likewise, design patterns can be viewed as the micro architectural elements of frameworks that document and motivate the semantics of frameworks in an effective way [18]. Design patterns have been used extensively in developing application frameworks. Many researchers [15][17][31] have suggested using as many patterns as possible for developing application frameworks because the abstractness and design expertise are embedded in patterns.

IV. 4 Components

Szyperski [33] points out that component technology is standalone, which has gone beyond object orientation. He defines "software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system". In that definition a software component is best thought as a unit with well-defined interfaces and has explicit context dependencies. He explains that insisting on independence and binary form is essential to allow multiple independent vendors and robust integration. Components are not just a big object. Eliens [13] describes that components usually consist of a collection of objects that provide additional functionality that allows components to interact together. Szyperski [33] states that component is a unit of independent deployment, a unit of third party composition, and it has no persistent state. By contrast, an object is a unit of instantiation, which has a unique identity, it has state, which can be a persistent state, and an object encapsulates its state and behaviour. A component is likely to come to life through objects and therefore would normally consist of one or more classes or immutable prototype objects. Component oriented programming requires the support of polymorphism and modular encapsulation [33].

A component has well-specified functionality with standard interface and behaviours, and a concrete implementation of an area of the system. Components in a framework provide a generic architectural skeleton for a family of related applications and complete applications could be composed by inheriting from and /or instantiating framework components. Atkinson [1] states that there are two types of relationship between component instances that are important at runtime. The first is composition, which captures the idea that one component is a part of another. The key aspects of the composition relationship are: 1. Composite objects are responsible for the creation and destruction of their parts. 2. The parts of a composite object take their identity from their composite object. 3. Composition is transitive. The other one is the client/server relationship. A client/sever relationship between two components instances defines a contract between them. For components to be independently deployable, their granularity and mutual dependencies have to be carefully controlled from the outset.

Component and application frameworks have close relationship. Many application frameworks use Common Object Request Broker Architecture (CORBA) to increase the interoperability among each part of the framework. CORBA, a big component essentially has three parts: a set of invocation interfaces, the Object Request Broker (ORB), and a set of object adapters. For invocation interfaces and object adapters to work, two essential requirements need to be met. First, all object interfaces need to be described in a common language. Second, all languages used must have bindings to the common language [33]. Fayad [18] states that frameworks can be used to develop components. Equally, components can be used in blackbox frameworks.

V. The Analysis of Current Application Frameworks Practices

There are a number of application frameworks projects developed during the 90's including some notable domain specific application frameworks such as San Francisco, OSEFA, and SEMATECH CIM. The application frameworks research community has accumulated considerable experiences in some domains such as finance, manufactory and telecommunications about building and implementing application frameworks in the last decade. The followings are core experiences identified during the study:

V. 1 Object-oriented concepts applied

Object-oriented application frameworks approach takes advantage of object-oriented concepts such as abstraction, inheritance, encapsulation and polymorphism as well as the use of object-oriented programming language. Though application frameworks is not exclusive to object oriented technology, the object-oriented concepts have a great impact on the development of application frameworks as a foundation of building and implementing application frameworks.

V. 2 Large-scale reuse

Application frameworks is different from other reuse techniques such as code reuse, design pattern reuse, class library and component reuse. It is aimed at achieving large-scale reuse in which the application developers are not only reuse the code but also the module and architecture of the application frameworks.

V. 3 Domain specific

Application frameworks is likely domain specific such as a financial framework or manufacturing framework. A domain specific framework extracts domain expertise and current solutions of the domain. Domain knowledge is crucial for developing application frameworks, which can be extracted from the domain experts and the current solutions of the domain. The development of application frameworks in some domains has accumulated more experience than others.

V. 4 Existence in any development stage

Concerning the scope of reuse and development process, an application framework can be as abstract as analysis framework or more concrete as implementation framework. Majority of the researchers tend to agree that application frameworks is a kind of semi-completed applications and the reuse can potentially exist in any development stage such as analysis, design and implementation.

V. 5 Development approaches

Though it is not proven that the object-oriented technology is the only way to develop application frameworks, most of the applications frameworks are developed by using the object-oriented technology. Iterative and incremental development approaches have been the main development methodology supporting the development of application frameworks. Framework design can be bottom-up and pattern driven or top-down and target driven.

V. 6 Issues identified to date

Application frameworks research has shown considerable achievement in the context of systems development. However, some obstacles have been identified during the study involving in the theory and practice of application frameworks using object-oriented technology. The issues are organised into the sections of 3.2.1 to 3.2.9.

V. 7 Developing and implementing effort

Developing application frameworks is hard. Due to the complexity, size of application frameworks, and the lack of understanding of framework design process, application frameworks is usually designed iteratively requiring substantial restructuring of numerous classes and long development cycles. Implementing application frameworks is as hard as developing application frameworks. A framework conventionally consists of the core classes of an application, and one has to understand the basic architecture of a particular application type to be able to specialise the framework.

V. 8 Infrastructure coupling

The application developers will have to rely on the architecture structure defined by the application framework while implementing the framework. Infrastructure coupling is very difficult to avoid for a whitebox framework since inheritance is the main mechanisms of implementing the framework that cause widespread coupling (the result of extensive use of inheritance), and consequently restrict the extendability of the application developed by the framework. In contrast, the traditional object-oriented development approach can eliminate the problem by reducing the unnecessary use of inheritance.

V. 9 Combining frameworks

Combining frameworks is not a straightforward task. One of the perceived advantages of using application frameworks is to increase extendability [17]. However, combining two or more frameworks without breaking their integrity is difficult because a framework assumes that it has the main control of an application [29]. Also, the difference of architectural style is another problem, which may potentially prevent two frameworks from combining together [4]. It is also difficult to combine applications developed from a framework with legacy systems because the new application generally contains behaviour for internal framework functionality in addition to the domain specific behaviour, but the legacy system in general only has the functionality of the domain.

V. 10 Overlap and potential gap

Application frameworks is often developed to capture specified domain knowledge. A domain specific application framework is a skeleton of architecture for a domain. It is possible in practice that the real world entity can be classified into different domains and consequently stored in different application frameworks. In application, it is desired to have only one object to represent the real world entity (single inheritance). The entity overlapping in different frameworks may confuse the applications building based on two or more frameworks. Also, potential gap may exist among two or more frameworks because it is possible that two frameworks cannot still be able to meet an application's requirement. In that situation substantial effort is needed to modify the design of the application or using mediating software [4]. In either case the implementation becomes very complex and difficult to handle.

V. 11 Testing

There is little research relating to frameworks testing. According to the nature of application frameworks, a framework is 'a skeleton of structure' with the notion of abstraction. It would, therefore, be very difficult to conduct testing. Currently, most of the researchers are testing the applications developed from the framework and iteratively evolve the framework. A well-designed framework component typically abstracts away from application specific details, however, it increases the complexity of module testing since the components cannot be validated in isolation from their specific instantiations [14]. It would be difficult to test the application developed using calling frameworks in which the frameworks calls the application when the application runs. It could be complicated to follow a thread of execution, which was mostly buried under framework code.

V. 12 Documentation

Documentation is used by both the framework developers and application developers who implement the framework. Fayad [17] classifies it into two categories such as for framework developers, which is used to modify and enhance the structure as well as the performance of the framework, and for application developers, which is employed to understand and use the framework. The current research indicates the documentation is still inadequate for application developers, which has potentially increased the difficulty of the learning curve [17]. The current documentation approach (text plus model diagram plus code example) is often difficult to acquire understanding of frameworks.

V. 13 Maintenance

Current research has no indications of that the requirement for maintenance would be reduced by adopting application frameworks approach. As a long-term investment, frameworks evolve over time and need to be maintained. Most of the application framework developments adopt iterative and incremental development strategy. Thus, when a change made to a framework the applications developed using the framework will be affected as well because the application use the structure and partial code of the framework. In this case the application development must be delayed if the major new version of the frameworks is about to be available in the near future. The company may have even to maintain more than one version of the framework since there may exist applications based on the old version of the framework.

V. 14 Feasibility

There is little feasibility study conducted in the area of appli cation frameworks. However, OSEFA discloses the cost issu e and states that developing a blackbox framework takes aro und two to three times as much effort as developing a fixed a pplication from the domain. Reusable framework like other r eusable technologies is only as good as the people who build and use them. Developing robust, efficient, and reusable ap plication frameworks require the project team to have a wide range of skills. It needs expert analysts and designers who have mastered design patterns, and software architectures an d expert programmers who can implement these patterns and architectures in the application framework. Even though it may be feasible to develop a framework for a particular dom ain from a technological perspective it is not necessarily adv antageous from a business perspective. The return on invest ment from a framework developed may come from selling to other companies, but to a large extent, relies on future savin gs in development effort within the company itself. However , future earning is not obvious to be justified as the technolo gy itself and business environment is changing. It is difficult to estimate the amounts of work required for a specific appl ication. It is also difficult to foresee if a specified business re quirement is supported by the framework.

VI. Conclusions

The application frameworks technology supports large-scale reuse and increase the quality of the software products. An application framework can be described as a skeleton of software systems upon which application developers are able to build applications. Though application frameworks is not exclusive to object oriented technology, most of the current work is coming from object-oriented community. Application frameworks have a close relationship with object-oriented technology in which object-oriented principles are adopted. The results of the study indicate that application frameworks technology is still immature and not yet to be another silver bullet but potential is imminent. The experiences accumulated by the research community indicate that application frameworks apply object-oriented concepts, aimed at large-scale reuse, likely domain specific and can exist in any development stage. Also, some issues relating to application frameworks have been identified in terms of development and implementing effort, infrastructure coupling, combining frameworks, overlap and potential gap, testing, documentation, maintenance and feasibility. Applications developed by implementing application frameworks may increase quality in terms of correctness and reusability with some penalty factors. The extendability and interoperability may be reduced due to the high inheritance coupling nature of the application developed from application frameworks. The results of the study suggest that the use of application frameworks technology has made a difference in systems development in terms of: (1) a number of application frameworks such as GUI, San Francisco and CIM frameworks have been developed and used by industry; (2) application frameworks technology supports large-scale reuse by incorporating with other existing reuse techniques such as design patterns, class libraries and components. The results of the study discover that the methodological support concerning building and implementing application frameworks is inadequate. The results of the study also point out that one of the claims, made by the advocates of application frameworks technology regarding the technology can increase the extendability of the software systems developed by application frameworks, are debatable.

VII. Future Work

Application frameworks technology may become one of the promising technologies in systems development if the following two research areas achieve significant breakthrough in the future. (1) Methodological study especially, the process of building and implementing application frameworks. (2) Solving technical issues related to building and implementing application frameworks such as how to increase the interoperability and extendability in frameworks approach.

References

[1] Atkinson, C… et al. (2002). *Component based product line engineering with UML* London: Addison-Wesley.
[2] Ahmad, W.AI (2000). *Inheritance in object –oriented language-requirements and supporting mechanisms.* Journal of Object Oriented programming Jan 2000 New York.
[3] Bass, L., Clenments, P. & Kazman, R. (1998). *Software architecture in practice.* MA: Addison-Wesley.
[4] Booch, G., Rumbaugh, J. & Jacobson, I. (1999). *The unified modelling language user guide.* Addison-Wesley.
[5] Bosch, J., Molin, P., Mattson, M,. and Bengtson, P. (2000). *Object-oriented framework based software development: problems and experience.* ACM Computing Surveys, Volume 32, 2000.
[6] Bruel, J.M, & Bellahsene, Z.(eds. 2002) *Advanced object-oriented information Systems,* OOIS 2002 workshops Montpellier, France, September 2, 2002.
[7] Brugali, D… et al (2000). *Frameworks and pattern languages: an intriguing relationship,* ACM Computing Surveys, Volume 32, 2000.
[8] Chen, D. J., Koong, C. S., Chen, W. C., Huang, S. K., and Van Diepen. N.W.P (1999). *Integration of reusable software components and frameworks into visual software construction approach.* Journal of Information and Science and Engineering 2000.
[9] Coad, P. & Yourdon, E. (1990). *Object oriented analysis (2nd),* Englewood Cliffs, NJ: Yourdon Pres: Prentice-Hall.
[10] Cockburn A. (1997). *Surviving object-oriented projects: A Manager's Guide.* Addison-Wesley.
[11] Deitel, H.M (2003). *Visual Basic.net For Experienced Programmers Developer series.* Upper Saddle River, NJ: Pearson Education.
[12] Due, R.T. (2002). *Mentoring object technology projects* NJ07458: Prentice Hall PTR.
[13] Eliens, A. (2000). *Object-oriented software development, 2nd ed.* England: Pearson Education.
[14] Fayad, M.E., & Schmit, D.C. (1997). *Object-oriented applications frameworks.* Communication of the ACM, Oct 1997 v40 n10, 32.
[15] Fayad, M. E., Schmidt, D. & Johnson, R.E. (1999a). *Building application frameworks: object-oriented foundation of framework design.* New York: John Wiley & Sons.
[16] Fayad, M. E., Schmidt, D. & Johnson, R.E. (1999b). *Implementing application frameworks: object-oriented framework at work*. New York: John Wiley & Sons.
[17] Fayad, M. E., & Johnson, R.E. (2000). *Domain specific application frameworks: frameworks experience by industry.* New York: John Wiley & Sons.
[18] Fayad, M. (2000b). *Introduction to the computing surveys' electronic symposium on object oriented application frameworks.* ACM Computing Surveys, Volume 32, No.1, March 2000.
[19] Fayad, M. (2000d). *Enterprise Frameworks: Guidelines for selection.* ACM Computing Surveys, Volume 32, 2000.
[20] Fowler, M. (1997). *UML Distilled, 2nd ed,* Harlow, England: Addison-Wesley.
[21] Graham, I. (2001). *Object-oriented methods principles & practice (3rd).* London: Addison-Wesley.
[22] IBM San Francisco Frameworks white paper: http://www.javasoft.com/javareel/isv/ibm/SanFrancisco/white_paper.html.
[23] Johnson, R (1997). *Frameworks for object-oriented software development* Communication of the ACM, Oct 1997 v40 n10, 39.
[24] Lewis T., Rosenstein, L., Pree, W., Weinand, A., Gamma, E., Calder, P., Andert G., Vlissides J.& Schmucker, K. (1995). *Object-oriented application frameworks.* Greenwich: CT Manning.
[25] McConnell, S (1993). *Code complete : a practical handbook of software construction* Redmond, Washington: Microsoft Press.
[26] Meyer, B. (1988). *Object-oriented software construction. Englewood Cliffs* NJ: Prentice Hall.
[27] Paul, C et al (2002). *Evaluating software architectures methods and case studies.* Addison-Wesley.
[28] Pee.w (1994). *Design pattern for object-oriented software development.* Addison-Wesley.
[29] Pree, W. and Koskimies, K (2000). *Framelets- small and loosely coupled frameworks.* ACM Volume 32, Number 1es, March 2000.
[30] Schmidt, D. C. (1996). *Lessons learned building reusable OO telecommunication software frameworks.* Lucent Labs Multiuse Express magazine, Vol. 4, No. 6, December, 1996.
[31] Schmid, H, A. (1995). *Creating the architecture of a manufacturing framework by design patterns* OOPSLA 95 Austin.
[32] Schmid, H, A. (2000). *OSEFA:Framework for Manufacturing.* Fayad, M. E., & Johnson, R.E. (2000). *Domain specific application frameworks: frameworks experience by industry.* New York: John Wiley & Sons.
[33] Szyperski C. (1997). *Component software: Beyond object-oriented programming.* Addison-Wesley Longman.
[34] Zamir, S. (1999). *Handbook of object technology (ed.).* CRC Press LLC.