Application Framework Issues when Evolving Business Applications for Electronic Commerce

Garry Froehlich
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2H1
garry@cs.ualberta.ca

H. James Hoover
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2H1
hoover@cs.ualberta.ca

Wendy Liew
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2H1
wendy@cs.ualberta.ca

Paul G. Sorenson
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2H1
sorenson@cs.ualberta.ca

Abstract

When an organization embarks on e-commerce it rarely has a chance to re-engineer its existing business applications. However, if these business applications were built using an application framework, then one might hope to reuse many of the existing legacy applications in the new e-commerce context. This paper examines the general issues created by migrating applications to e-commerce, and proposes an architecture for application frameworks that must support e-commerce.

1. Introduction

1.1. E-Commerce

The current general definition of e-commerce is *the ability to do business on-line via the internet.*

Traditional e-commerce focused on business-to-business transactions over proprietary networks in the form of EDI (Electronic Data Interchange). The outcome of this limited business-to-business form of e-commerce were, among others,

- Increased efficiency of order processing,
- Reduced costs due to just-in-time inventory management,
- Locking-in trading partners by requiring all suppliers to participate in the EDI network.

The advent of the internet and the explosion of the web have opened up e-commerce to small businesses and consumers in general.

The situation facing organizations as they move from a locally based enterprise towards a web-based e-commerce enterprise is depicted in Figure 1. Their business processes have to expand into a new domain, and if they use an application framework to build their business applications, then that framework also has to expand into a new domain.

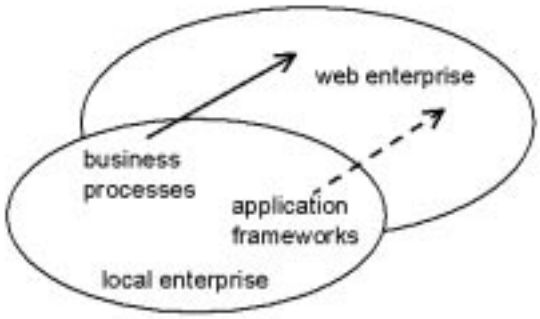

Figure 1: Expanding into E-commerce means moving both business processes and the technology to implement them into a new domain.

With Internet-based e-commerce come new types of transactions, which are loosely categorized along

two axes: the parties in the transactions, and the things involved:

- Parties: business, consumer, government
- Things: tangible goods, intangible goods, services

A transaction type is then some combination of parties and things. For example, one can have consumer-business transactions for tangible goods, such as buying books; or consumer-government transactions for services, such as filing income tax.

This taxonomy is useful for detailed workflow analysis when building an e-commerce application. For example, if only intangible goods are involved, then one can ignore the issues associated with shipping, receiving, and its associated tracking. But from a framework standpoint, there is nothing essentially different between these various transactions. A good framework for electronic commerce should enable one to build applications for any of these transaction styles.

1.2. Is E-Commerce Different From Normal Commerce?

How different is e-commerce from normal commerce? One could claim that since almost all contemporary business transactions are mediated by computers over networks, all commerce is electronic. But e-commerce is not just the presence of computers or absence of paper. It implies more, such as:

- Using a non-proprietary open network, the Internet, with its associated issues of security and reliability.
- Not requiring proprietary client software, that is, any browser should do.
- Service that is 24 hours a day, 7 days a week, and its associated system reliability requirements.
- A greatly expanded, non-exclusive club of possible parties. Customers and suppliers can be geographically distributed worldwide.
- A change in the relationships between trading parties. Since it becomes simpler to locate and compare products, the role of 3rd party brokers and middlemen changes. The must provide some significant value added service if they do not want to be bypassed.
- The need to establish the identities of parties without requiring physical contact, and the resulting authentication problems.
- The need to establish off-line contact between parties through email, voice, fax or other means.
- The ability to collect data and profile parties. Vendors can know more about their customers habits and needs and so provide more precise marketing. Customers can search and compare among vendors, and customer groups can keep data on suppliers.

Are e-commerce business processes any different from normal? Most business objects and analysis patterns [3] for e-commerce are the same as those for usual business systems. For example, whether done electronically or not, the typical business-business supply chain procurement process has to support tasks like: Account Opening, Order Entry, Order Change, Order Status Inquiry, Product Configuration, Acknowledgement and Feedback, Delivery Advice, Invoicing Protocol, Payment Protocol, and Customer Support. It will require business objects like Customers, Accounts, Orders, Shipments, and so on. E-commerce does not really change this basic analysis, but it does provide the opportunity to enhance the business processes, for example by building customer profiles. It also comes with its own problems, such as authentication, which influences the details of the business processes.

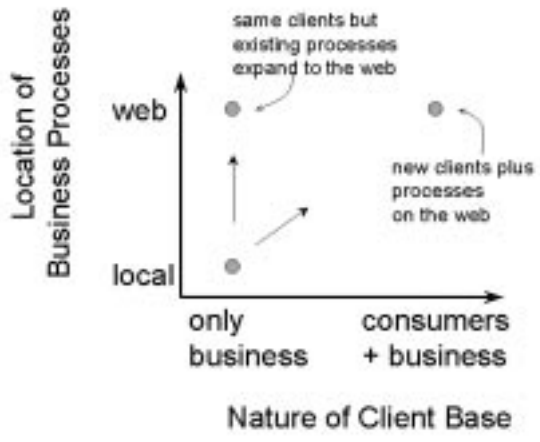

Figure 2: OfficeCo's decision to embark on E-commerce permits extending its existing processes, and provides opportunities to expand the client base from only other businesses to also including consumers.

E-commerce provides a business with an opportunity to change the way it delivers services or to expand its client base. For example, suppose that *OfficeCo*, an office products company, currently deals only with a small number of business clients. *OfficeCo* may want to allow its existing client base to use the web for procurement in addition to the normal phone and fax route. In this case they are taking existing processes and extending them over the web. Or perhaps *OfficeCo* wants to expand into the consumer market. To do that using existing methods might require expanding their sales staff, or fragmenting sales staff who now must deal with a mix of traditional business customers and with consumers with different expectations. Thus expanding to the consumer base might only be possible if it occurs in conjunction with extending the business processes over the web. This kind of situation is illustrated in Figure 2.

An organization desiring to support e-commerce is not usually in a position to completely re-engineer its business processes. So any e-commerce effort will have to deal with legacy business processes. In particular, it will have to address the issue of how to migrate existing applications. If existing applications are built with an application framework, then the question becomes one of how the application framework can be extended to support e-commerce. This extension could involve attempts to use additional frameworks, such as one already extant for web applications, and integrating them within the new application.

1.3. Focus on Application Frameworks

An *application framework* is a software architecture, along with its implementation, that provides all of the generic capabilities required by applications in a particular domain. A new application in the domain is developed by adding custom application-specific code to points in the framework called *hooks*. These hooks vary considerably in nature, from simple parameterizations of existing classes, to registering callback methods, to inheriting from abstract base classes for implementing new architectural features.

The promise of application frameworks is that they reuse of the portion that is common to all applications in the domain. The price one pays for this reuse is that one must accept the overall architectural solution provided by the framework. You have to adapt your problem to the framework.

The typical business application framework has to provide the following:

- workflow support — being able to express the business processes of the organization,
- user interface foundation — the basic look and feel of the user interface, and mechanisms for it to monitor and control the workflow,
- service provision — providing the services necessary to implement the workflow, such as authentication.

Most application frameworks are built with a particular style of user interface and workflow in mind. It is very difficult to use the framework if your application does not fit into that workflow model. So a crucial question for migrating legacy applications is whether their user interface and workflow model makes sense in an e-commerce context.

It is important to note that application frameworks are not simply the domain of sophisticated software developers. Even small organizations use application frameworks. Off-the-shelf application builders like Visual Basic, databases like Access and Paradox, and even spreadsheets are all a form of application framework. Small organizations especially can have a significant investment in these home grown applications, and it is next to impossible to migrate these applications to an e-commerce context.

Our focus for the rest of this paper is on issues associated with applications built using a framework. Legacy applications built in an ad hoc way are much more problematic (although sometimes wrapping technology can be used to hide the original system).

2. E-Commerce Demands on Workflow

What makes an application framework for e-commerce any different from one for normal business processes? Does an e-commerce framework have to handle any unusual demands that are not already raised in normal business processes?

Many of these issues, like dispersed customer locations, will already be addressed in the business processes of large firms, while small firms with walk-in clients may have never experienced them before. It is also important to note that, because the network is open, a task like authentication now becomes a concern at every interaction in a process.

The requirements analysis for the e-commerce business process has to re-examine every existing process and business object in the e-commerce context. The kinds of issues it will expose are illustrated in the following example of the tasks in a typical procurement workflow. It shows how the the process and interface

of the tasks are affected by introducing e-commerce and the Internet.

1. **Account Opening:** How does identification get verified? For example, a clerk used to be able to visually examine a driver's license. How does the identification information get transferred securely over the network? Parties can now be in vastly different geographical and jurisdictional locations (tax and legal), this must be anticipated and captured.

2. **Order Entry:** The user interface needs to be friendly, since now the user and not a sales clerk is constructing the order. Access to the catalog for parts numbers, details, and so on is needed, as well as a mechanism for price quotations. Should the order entry system adapt to the profile of the customer, for example by rearranging the catalog to the users style (for example if the customer is an NT shop, only list NT compatible software).

3. **Order Change:** Users must only retrieve orders they are authorized to see, which will require a modified search facility from that provided to a sales representative. How does the impact of a change request get communicated to a customer, for example changing a component to something incompatible with the other parts being ordered. A change to an order is a modification of the original contract, so how is this recorded and communicated for audit purposes?

4. **Order Status Inquiry:** A status inquiry might involve a third party, for example forwarding the request to the courier company's system to get a location status for the shipment. So the supplier may need to use authentication information provided by the customer.

5. **Product Configuration:** Customers may not be experienced in specifying the configuration of the product, for example when buying a computer system. What kind of support will be provided?

6. **Acknowledgement and Feedback:** Customers may want to select among alternative forms of feedback, for example some may be happy with email acknowledgement of orders, others my want a fax. What happens in exceptional circumstances that require contacting the customer, for example if a shipment is delayed, product cancelled, or payment fails?

7. **Delivery Advice:** How should choice of delivery method be handled. This may require scheduling with the customer in the case of delivery or installation. How does the customer or supplier handle revised delivery requirements and rescheduling?

8. **Invoicing Protocol:** How are customers to be billed? How are receipts for delivery and payment going to be generated?

9. **Payment Protocol:** How are payments to be made? What about pricing and payments in foreign currencies? What about taxes and other charges that are a function of the customer and supplier's locations? One can be certain that this particular issue will evolve rapidly in the near term.

10. **Customer Support:** Many operations do not have any existing support structure. How is this to be integrated into their business systems? Should there be product support newsgroups, and should they be open to all or just customers?

As one can see from just this cursory examination, migrating an organization's business processes out to the customer is non-trivial. The basic problem is that the user community for applications undergoes a radical change in domain (as in Figure 1) from being experienced in-house users to inexperienced, potentially hostile, outsiders. It may be that this creates such a radical alteration in the workflow that an entire set of new applications will have to be created for the e-commerce side of the business. Of course the hope is that because existing applications were constructed with a framework, the effort in producing new ones will be dramatically less.

3. An E-Commerce Application Framework

Most off-the-shelf business application frameworks were not built with e-commerce in mind. They have a very specific purpose: enable the rapid construction of two-tier business applications. A typical two-tier business application framework is illustrated in Figure 3.

In this kind of framework, the basic model is that business objects are stored in the database, and forms are used to create new business objects and to change their state. Typically each form is populated by a

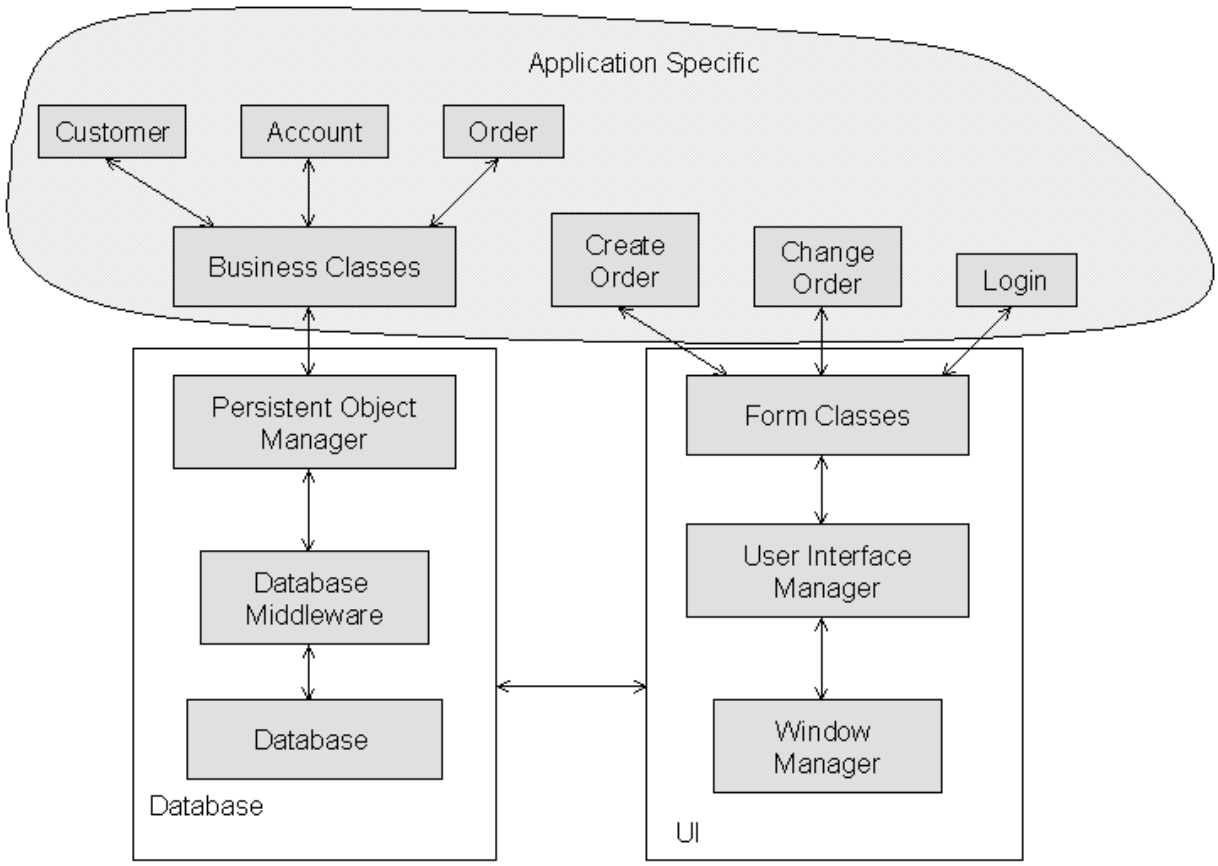

Figure 3: A standard two-tier business framework.

database query associated with the form, and a form changes object states through database post operations.

A developer designs the database schema and the forms. The framework provides the skeleton, and resulting look and feel for the application, along with services like transaction locking and rollback.

The single biggest problem with such two-tier frameworks is that the business logic is associated with the forms. Thus the user interface cannot be replaced without reimplementing much of the business logic. Since migrating the user interface out over the web is a key requirement of e-commerce, such frameworks are intrinsically unsuitable for e-commerce. Applications built using a two-tier model are very difficult to migrate. Even if an organization does not intend to pursue e-commerce immediately, it should still consider avoiding two-tier applications.

The logical extension to the two-tier framework factors out the business logic into a separate part of the framework. This is the so-called three-tier model. The basic philosophy of three-tier frameworks is that the business objects maintain the state of the business process, and workflow is controlled by the workflow manager. Users interact with the workflow manager through forms. Since their user interface component is, more or less, independent from the business logic, three-tier models are more amenable to distributing an application over the Internet. In particular, the three components need not be co-located. One can have a thin-client model, in which the UI is extended over the web, while the business logic and database are on a common server. Or one can have a thick-client model corresponding to the traditional two-tier application where the business logic and UI are on the same platform, and the database is on a server.

We suggest that even the three-tier model is inadequate for e-commerce. Because e-commerce can

affects the details of business processes, the framework must be able to not only replace subframeworks, but to integrate and simultaneously support different implementations of subframeworks, and to accommodate new services.

Figure 4 illustrates this extension of the three-tier model. The framework is composed of three main subframeworks, and a services pool:

- Object Management Framework — this is responsible for storing and retrieving all objects in the application, not just the business objects. It is also responsible for isolating the application from the underlying database, so that it is portable to different environments.

- Business Logic Framework — the purpose of the business logic framework is to encapsulate the business rules and processes independently of the user interface. This addresses a typical problem with forms-based applications where too much process is imbedded in the forms.

- User Interface Framework — the purpose of the user interface framework is to isolate the basic notion of a form (which the user interacts with) from the underlying window system and operating system. It also provides the glue the holds the interface together: the main form, menus, toolbars and so on that the user uses to initiate forms.

- Generic Services Pool — This can be viewed as a virtual machine, that provides services to all the other frameworks in a sufficiently abstracted form so that the services can be implemented in various ways. It contains a number of service frameworks, of various complexity such as error handlers, email agents, message passing services, authentication, and encryption.

The application specific code consists of defining the specifics of each business class, the forms that generate transactions, and the actual transactions that cause business classes to change state.

4. Migrating to an E-Commerce Framework

Marketing hype aside, there are enormous technical hurdles to overcome when evolving a framework to a new domain. Deciding on a migration strategy depends on thoroughly understanding the issues in this section for the existing and prospective frameworks.

Small to medium sized organizations often use commercial off the shelf components and frameworks or framework-like environments such as Borland Delphi. Typically, these frameworks and tools will not provide electronic commerce capabilities. When moving from an existing framework towards electronic commerce capabilities, developers must consider the amount of change required to add these capabilities. Three of the options they have are:

1. Extend the existing system. If only a small amount of change from the existing system is required, then the existing system can be extended or modified. However, as shown in Section 2., the shift to electronic commerce can touch several aspects of the business process and the system can require a fairly substantial amount of modification. Extending an existing framework may not be possible if the developers do not have access to the source code.

2. Integrate the existing system with a new electronic commerce framework. For example, a web-based user interface framework might be combined with an existing system if the current user interface for the existing system can be easily replaced.

3. Rebuild the system for electronic commerce. If the first two options are not feasible, then a new system may have to be developed. It may also be possible to purchase a new system to replace the old one.

Earlier parts of this paper touched on the amount and types of changes required to move to electronic commerce. The choice of whether to combine a new framework with the existing framework or to rebuild depends on the ease with which the frameworks can be integrated and the amount of resources already invested in the existing system. The rest of this section focuses on the issues involved in trying to integrate two or more distinct frameworks.

Like many software systems, object-oriented frameworks are usually designed as stand alone systems. The framework is intended to provide the basis for a complete application. However, a single framework will not always provide all of the functionality required by an application, particularly applications that are evolving into new areas such as electronic commerce. For this reason, an application may require multiple frameworks, for example one framework for the web-based user interface and another for the underlying business logic, and a means of integrating those frameworks into a coherent whole.

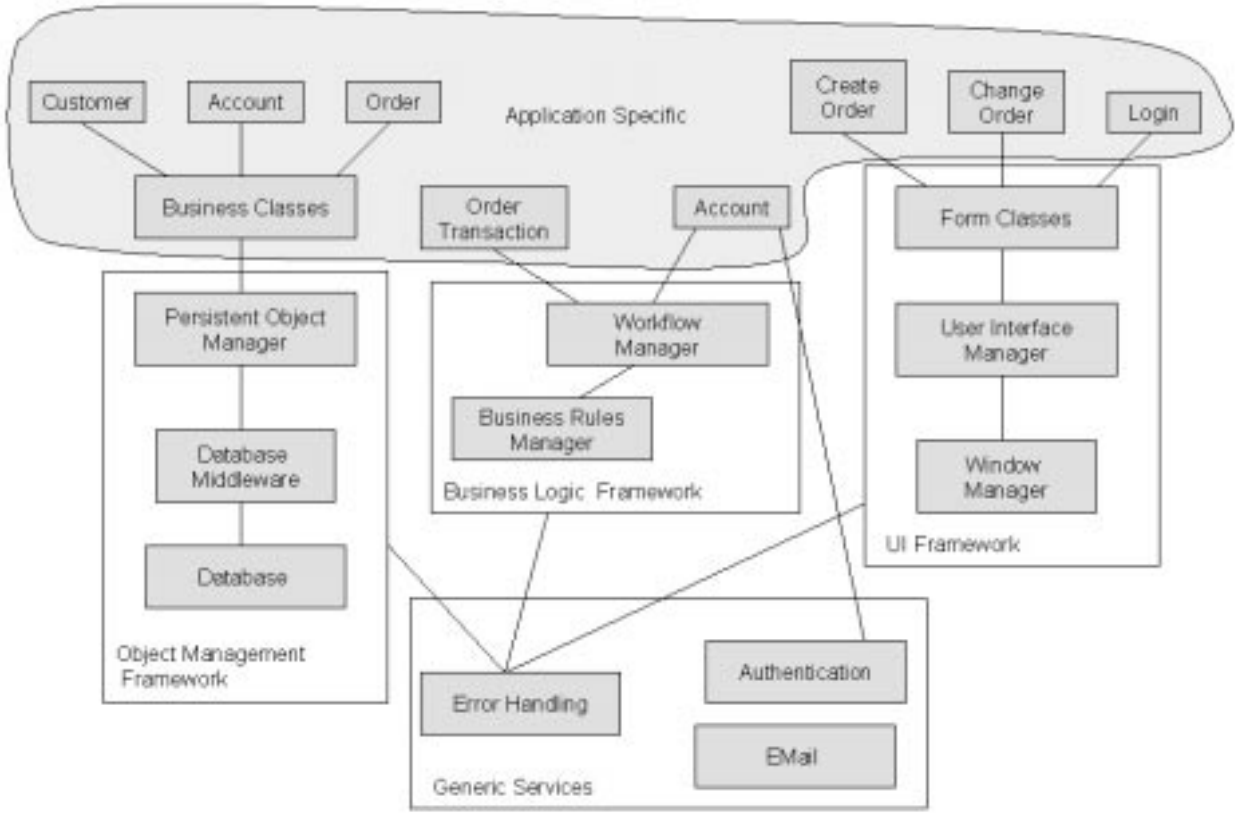

Figure 4: Proposed extension to the three-tiered business framework composed of sub-frameworks and a services pool of frameworks. Hooks are represented by lines between classes.

Since single, monolithic frameworks are not designed to be used with other frameworks, problems arise when these frameworks are combined. Integration concerns occur in three main areas:

- the services provided by a framework to an application; what the framework does
- the thread of control model used by a framework; how the framework works
- the hooks of the framework that an application is meant to use; how the framework is used

The two main types of problems that can occur when integrating multiple frameworks are gap and overlap [13]. If the services expected by one framework do not match the services expected by another, then there is a gap. Overlap occurs when two or more frameworks provide the same services, or try to control the same resources. Each of the three areas are discussed below in terms of these problems.

4.1. Services

Each framework provides services that can be used by an application, and may also require services to be provided by the application. Many application frameworks provide all the services the framework requires. A user interface framework will provide standard user interface controls such as forms and buttons, but may also incorporate error handling and communication services. Integration problems occur when two or more frameworks provide the same service, or require an outside service.

To identify potential integration conflicts, a list of the services required and provided by the frameworks needs to be made. The initial list only needs to include high level or large grained services, such as error handling and authentication. Framework gap can occur when services required by one framework are not provided by another or are provided in a way that

does not meet the framework's requirements. The means of providing and using services in frameworks can be described by hooks and this issue is further expanded in the subsection on hooks.

Services that are required by the application and provided by two or more frameworks become sources of framework overlap problems and need to be further examined. Examples are error handling and communication services. Sometimes, the high level service can be broken down into more fine-grained services. If the conflicting frameworks provide different fine-grained aspects of the same large-grained service, then there may not be a conflict. Specialized communication to the object management framework or database may not need to be in the same service framework as communications with client programs or web browsers. Framework overlap occurs when two frameworks provide the same fine-grained service.

To deal with overlap problems in services, the two frameworks should be factored into smaller, service providing frameworks. A service providing framework is similar to a black box component or class library in that it does not have an independent thread of control. The service providing framework receives control from an outside source, performs a service, and returns control to the outside source. The frameworks in the Generic Services Pool, such as the Authentication subframework, are typical of service providing frameworks. Another vital characteristic of the service providing subframework is that it can be removed or factored out of the larger framework.

If the service cannot be factored out of either framework, then integrating them will require a great deal of effort involving re-engineering of one or both of the frameworks, and it may be simpler to build an entirely new system. If the service can be factored out of one of the two, then the adapter design pattern [5] can be used to allow the first framework to make use of the service provided by the second. Issues involving hooks will also arise when trying to integrate the two frameworks, and will need to be examined. Ideally, the common service can be factored out of both frameworks to become a third framework. Error handling is a good candidate for a separate service providing framework since many other frameworks will require the service and it is important to handle and present errors in a consistent way. The two frameworks will then rely on the third framework to provide the service.

Unfortunately, services are sometimes tied closely to others. Factoring them out may involve removing other parts of the system. A forms based user interface may incorporate some of the business logic within them and so when it is replaced with a web-based interface that logic has to be captured and added elsewhere in the system.

4.2. Control

Unlike pure object libraries, an object-oriented framework incorporates a control model which ties the parts of a framework together. The control model refers to the thread of execution used by a framework. For example, the typical control model for a window-based user interface framework involves an event loop inside the operating system itself which captures events that the framework can respond to.

For integrating frameworks together, the key issues in the control model are:

- what resources are being controlled
- does the framework use single or multiple threads of control
- what is the style of control
  - event-driven: external events are received and processed in the order they arrive, with the system idle between events. This style is used in most modern window based applications.
  - polling: the framework continuously monitors external controls and acts when the controls change
  - step-by-step: the framework has a directed graph of actions that the user must follow, and the user can only move to the immediate neighbors in the workflow. This is the screen based style used in many mainframe applications, and also modern 'wizards.'
  - batch: actions are performed in a predetermined sequence without outside intervention

Gap can occur when one framework, particularly a controlling framework, uses multiple threads of control, and a second service providing framework expects only a single thread of control. There is a gap or simply a mismatch between the type of control expected, and the one given. Under these conditions, the second framework is not guaranteed to perform correctly, so the first framework must be restricted to a single thread when interacting with the second.

The use of two different control models between interacting frameworks can also be gaps. A framework that uses step-by-step or batch control may not be

able to respond to requests coming from a framework using an event-driven control model. A framework may use more than one control style, such as an event-driven system with also incorporates some wizards so the states in which the frameworks interact has to be taken into account. When the user of a web-based framework can have multiple windows and multiple forms displayed at the same time, the existing system will have to be able to receive requests or information from any of them at any time.

Overlap occurs when two frameworks control the same resources. Sometimes the frameworks will expect complete control over the resource, such as both the organization's existing framework and the new electronic commerce framework they are trying to incorporate both containing the main event-dispatching loop. One or both of the frameworks then have to be factored into service providing frameworks and one of the control loops removed. Again, if the frameworks cannot be factored, then the e-commerce framework may not be suitable, or the existing system may have to be redeveloped.

Often a framework will not expect complete control over a resource or service, but only temporary control. It may request information and process it with the framework or actually hand off the thread of control to a service providing framework. A means of interleaving control, and ensuring that any dependencies between frameworks are maintained, such as a locking mechanism in the Object Management Framework, is needed.

4.3. Hooks

Hooks provide information about the ways in which a framework can be used, in terms of both what the framework expects to be provided and what it provides. As mentioned in the subsection on services, problems can occur when the service provided by one framework does not match up with the service required by another framework. In other words, the hooks that describe what the first framework provides do not map onto the hooks that describe what the second framework requires.

In object-oriented frameworks, the two main language mechanisms for providing hooks are inheritance and composition [14]. The use of inheritance allows the framework builder to embed many of the services and control flow mechanisms within the framework and only require the application developer to fill in a few methods when extending the framework. Since the developer has few details to worry about, these hooks are often easy to use. General transaction management can be handled in the Business Logic Framework with individual transactions inheriting the business logic making it simpler to quickly add new types of transactions. On the other hand, callbacks or simple method calls, types of composition, can be more difficult to use since it is hard to determine what state the system was in when the callback was invoked. In cases such as the Object Management Framework where the framework can maintain its own relevant state like the locks on data, composition is appropriate. However, hooks using composition are easier to integrate than are hooks using inheritance.

Framework gap can occur with hooks using composition, or when one hook uses composition and another uses inheritance. The framework providing the callback may not use the same interface or maintain the required conditions. An Error Handling framework may require information on the application state that the User Interface framework does not provide. Much like all gap problems, an adapter can be used to bridge the gap between the frameworks.

The more serious problem of framework overlap can occur when inheritance is used. When both the providing and requesting hook use inheritance, then multiple inheritance is required to bridge the two into a single subclass, bringing with it all of the problems of multiple inheritance. If both the User Interface and Error Handling frameworks have error classes within them which must be inherited from, then integrating them will require a third class which inherits from both if the language even allows it, and overlaps in methods and state variables must be resolved.

All of the service, control and hook concerns point towards easily factorable frameworks, similar to that presented in Section 3, as being the best candidates for integration. Of course, many systems simply cannot be so cleanly factored so some new components will need to be added to bridge the gaps between two frameworks, or some parts of a framework modified to work with another framework. However, if the framework cannot be factored, then integration with another framework becomes much more difficult and may not be feasible.

5. Case Study Experience

6. Conclusions and Future Work

The future directions of e-commerce are uncertain. An organization must be careful to choose application frameworks that not only address their current needs, but which will let them migrate to e-commerce in the future, and will be adaptable to the evolution in e-commerce itself. We are currently experimenting

with the details of how services should be factored in the extended three-tier model.

E-commerce places new demands not only on delivery technology, but on the way that business processes are designed. At present, technology is forcing organizations to embark on e-commerce before they have built a coherent model of the business processes they need. Since application frameworks need an associated business model in order to be useful, one main direction of future research is to develop business domain modelling frameworks that can support e-commerce. Without these models, developers of application frameworks will be solving the wrong problems. The San Francisco project [7] is just one example of the integration of busineess modelling and the support framework.

Acknowledgements

We wish to acknowledge the support of Natural Sciences and Engineering Research Council of Canada (Grants OGP , OGP, IOR) and Teledyne Fluid Systems - Farris Engineering.

References

[1] K. Beck and R. Johnson. *Patterns Generate Architectures.* In Proceedings of ECOOP'94, Bologna, Italy, 1994, 139-149.

[2] M. Fayad and D. Schmidt. *Object-Oriented Application Frameworks.* CACM, 40(10), October 1997.

[3] M. Fowler. *Analysis Patterns: reusable object models.* Addison Wesley, 1997.

[4] Froehlich, G., Hoover, H.J., Liu, L. and Sorenson, P. 1997. Hooking into Object-Oriented Application Frameworks. *Proceedings of the 1997 International Conference on Software Engineering*, Boston, Mass., May 17-23, 1997, pp. 491-501.

[5] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995.

[6] D. Gangopadhyay and S. Mitra. *Understanding Frameworks by Exploration of Exemplars.* In Proceedings of 7th International Workshop on Computer Aided Software Engineering (CASE-95), Toronto, Canada, 1995, 90-99.

[7] *San Francisco - Concepts & Facilities Version 2 SC41-0670-00.*
`http://www.ibm.com/Java/Sanfrancisco`

[8] R. Johnson. *Documenting Frameworks Using Patterns.* In Proceedings of OOPSLA'92, Vancouver, Canada, 1992, 63-76.

[9] G. E. Krasner and S. T. Pope. *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80.* Journal of Object-Oriented Programming, 1(3), August-September 1988, 26-49.

[10] M. Mattsson, J. Bosch, M. E. Fayad. *Framework Integration Problems, Causes and Solutions.* Preprint. `fayad@cs.unr.edu`, 1998.

[11] W. Pree. *Design Patterns for Object-Oriented Software Development.* Addison-Wesley Publishing Company, Reading, MA. 1995

[12] R. Pyle. *Electronic commerce and the internet.* Commun. ACM 39, 6 (Jun. 1996), Pages 22 - 23.

[13] Sparks, S., Benner, K. and Faris, C. 1996. Managing Object-Oriented Framework Reuse. *IEEE Computer.* 29(9), 1996, 52-62.

[14] Taligent. *The Power of Frameworks.* Addison-Wesley Publishing Company, Reading, MA. 1995.