Chapter 2

Business-to-Consumer Pattern Analysis

An Enterprise Business Pattern is not my invention any more than a design pattern or architecture pattern is the invention of those finally documenting them. Most patterns are invented long before authors capture them into a useful reference text. My efforts in capturing the Enterprise Business Patterns in this book are the result of much analysis. My analysis had the express purpose of recognizing what patterns already existed in the enterprise computing industry. In this and the two chapters that follow, I walk through the process that I used in identifying what patterns already existed. I then boil them down to the ones I believed to be most deserving of forming into a pattern catalog. I present my analysis in a top-down manner; that is, I present the outer layer and peel away until I find the patterns that are the most important to capture. In this chapter I specifically examine the Business-to-Consumer macro pattern and the Enterprise Business Patterns that exist within.

Being the most popular of all the XBPs (Extraprise Business Pattern), many enterprise architects and developers easily relate the *B2C* pattern. That's because just about every e-business provides some form of business interface to their product and/or service offerings via a Web site, and architects and developers have spent an incredible amount of time building them over the last several years.

Sometimes an e-business Web site offers nothing more than a way to view information such as "brochureware." Many others, however, offer sophisticated e-commerce engines with product catalogs, a shopping cart, personalization, and special promotional offer processing. From the basic to the complex, all such solutions constitute the popular B2C pattern. In this chapter I step through the B2C pattern from the most basic example and work my way through to the complex.

Basic B2C

To begin our analysis of the B2C pattern, let's first look at the most basic form and then gradually advance to the more complex. Here's what you are likely to find in the way of a simple "brochureware" or information portal Web site. My examination is presented in the form of requirements analysis, employing use cases as the means of exploring what each pattern's characteristics are. **Figure 2.1** provides a use case diagram for a basic B2C system.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

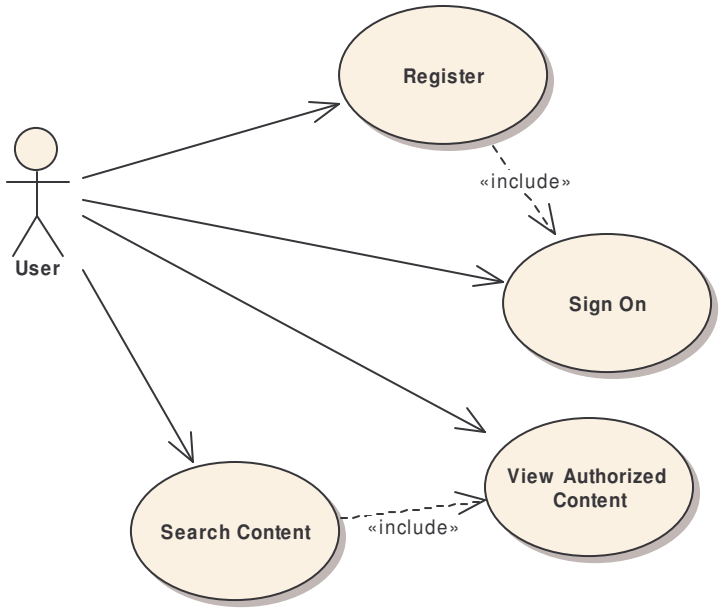


Figure 2.1: A diagram showing a minimum set of use cases likely found in a true B2C system.

As seen in **Figure 2.1**, I consider four basic use cases to be part of the basic B2C pattern. The user may always view some kind of content per the *View Authorized Content* use case. If a user wants to view more sensitive or valuable content they may be asked to register and sign on to the system using the *Register* and *Sign On* use cases. To quickly find pertinent content the user will likely want to employ the *Search Content* use case.

As you can see, whether or not the user has registered and signed on they can always view some kind of content. Regardless of what they are viewing at any given moment, however, their level of authorization has to be determined on the fly. If the user is not signed on, the content they can view is probably limited. If they are signed on, they can probably access another level of information. In some designs each anonymous user that browses to the Web site is assigned the default "guest" identity, which has limited access. Other designs assume that a user that has not signed in is simply denied any special access. These two approaches accomplish the same thing.

If the anonymous user attempts to access some level of privileged content then they will be prompted to register. It is up to the business to determine how much personal user information is required of the user. Following registration the user is automatically signed on to the system and granted access to the content previously denied them. On future occasions when the same user returns to the site they may sign on again and gain access to more content according to the level of authorization granted them.

Various levels of authorization may be granted based on certain criteria. The criteria may be that a sales person has spoken to the user about purchase expectations and determines that granting them a higher privilege of access is warranted. Or perhaps the user has paid for a higher level of access, which leads us to a more sophisticated level of the B2C pattern (discussed in detail below). (Frankly, in the technical arena the publishers of the information do well to grant full access to any and all users if they plan to sell their products into a broad market. But that's not a technical decision.)

The *Search Content* use case allows the user to find specific content over a broad range of available information in a short period of time. Any documents that have search

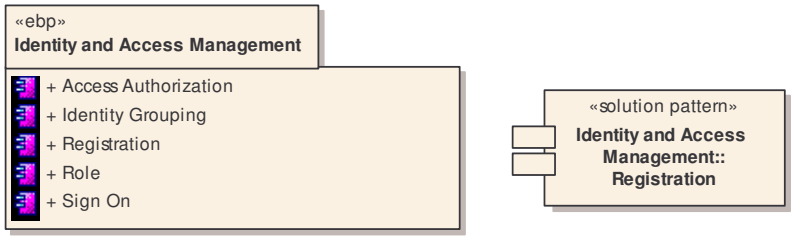
© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

“hits” in them will be presented to the user. They may be ranked according to relevance or presented in some other order. Generally it is good practice to present a text snippet from each document containing hits. Such a snippet should show the searched words in context. It is a business decision whether or not to present found documents (with snippets) as search results if the user is not authorized to view the full document. Generally if the documents are available to anyone at all and access to the documents is available for some sort of fee, it is a good idea to include them in search results and then deny access if the user requests to view it. If a pay-for document is what the user is looking for, seeing a snippet of the document may entice them to pony up the fee, especially when they are first denied access after selecting it.

Let’s now fully identify EBPs that reside within this very limited B2C XBP. The *View Authorized Content* use case infers two important B2C concepts. First, and most important to the consumer, it states that content is viewable by users. Secondly it qualifies that for a user to view content they must be authorized to do so. Really then this use case is divided into two different primary concerns: viewing and security.

Registration, authentication and sign on—the other two use cases—as well as authorization are all security concerns that are addressed by what is commonly known as identity and access management. These concerns are generally widespread in even the most basic B2C environment. “Widespread” is not overstating the facts as three out of the four use cases directly rely on or support identity and access management.

Here I have discovered only a limited definition of the *Identity and Access Management* EBP, but a full suite of solution patterns are provided in the pattern catalog. For now we have found the following overarching EBP and its Solution Patterns. Each of the components inside the EBP package are stereotyped `<<solution pattern>>`, as is shown in the *Registration* pattern (stereotyped component element) next to the package in the following diagram:



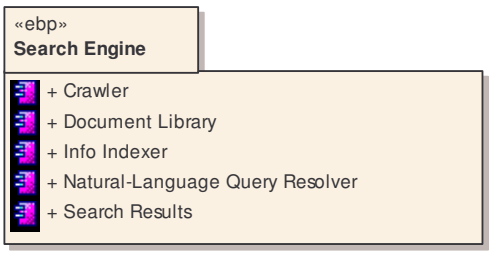
The purpose of the *Registration* and *Sign On* patterns define familiar concepts. The *Role* pattern defines permissions are administered to secure system resources. *Identity Grouping* captures how identities are gathered into groups, and all users within a group can share a particular set of permissions or privileges assigned to the group itself. The *Access Authorization* solution pattern defines how users are granted access to system resources via the *Role* they are in.

These patterns are essential for authorizing a given user the ability to view specific content. There are a few other solution patterns identified in the EBP. You can find full treatment of these patterns under the *Identity and Access Management* EBP in section 2, the Pattern Catalog. You should also note that this EBP is fundamental to each of the

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

XBPs; in other words, *Identity and Access Management* actually resides within B2C, B2B, and B2E, as you will see later. It is just that I happened to find it here first.

Clearly a search mechanism is central to this analysis, so it makes sense that I recognize the *Search Engine* EBP at this level of analysis. What isn't clear at this point is a notion of all the solution patterns within the *Search Engine*. The techniques behind search engines in general are very broad, and many are proprietary trade secrets (Google, for example). So there's really no way to capture every available *Search Engine* solution pattern. But we can deduce this much: There must be a way to index content and there must be away to search the index for sets of terms. There must be a way of sorting the search results (e.g. per relevancy). And there must be a way to display to results and link to the full text of any given result. So here is what the *Search Engine* EBP consists of:



The *Natural-Language Query Resolver* solution pattern is of particular interest. It is different from a SQL-type query analyzer because the search expression itself has few if any syntax rules. It allows a human to type what they are looking for. In response it uses several special techniques to deduce what the person means, and then searches the pre-generated index for documents that fit the bill. The index is also far different from something that belongs to a relational database, and it's the job of the *Info Indexer* solution pattern to generate an index that the *Natural-Language Query Resolver* uses as input. The *Info Indexer* gets its data form the *Crawler*, which knows how to scan a data source, such as a Web Site or other document hierarchy. The *Search Results* solution pattern is also unique because it doesn't deal with relational database rows and columns. Rather, it is document-centric, and may be assembled using a variety of sorting/ranking algorithms. Finally the *Document Library* is the means of browsing the indexed content as if it were a hierarchical catalog, rather than a big blob of nebulous documents.

Most of the propriety designs and implementations of *Search Engine* are found within the *Info Indexer* and *Natural-Language Query Resolver* solution patterns. While I do not attempt to provide any special insight into such algorithms, I will mention a few popular public domain strategies and how they work.

The other primary concern of my first use case diagram is that of viewing content. So there is always a user interface appropriate for use in a basic B2C environment. Users are presented some default information, and if they register and sign on to the system they can view some special information. They may also search and view the search results in a condensed format, and then in detail. As far as user interface is concerned this level of B2C pattern requires what appears to be a very unsophisticated presentation component. Nonetheless, there are some very important patterns inside the slightly more complex EBP called *Dynamic Web Site*.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

Peering inside the *View Authorized Content* use case we find the essence of the *Dynamic Web Site* EBP. A dynamic business web site is based on the need to supply up-to-date views of live data. It also must be able to serve the needs of the consumer in versatile ways. In fact, the following solution patterns are reused in many of the sophisticated presentation EBPs discussed below. They are *Dynamic Web Page*, *Stylized Page*, *Printer-Friendly Article*, *Image Producer*, *Business Document Publisher*, and *Intellectual Property Distribution Facility*.

I summarize the solution patterns in logical order. First, the *Dynamic Web Page* defines how "live" data is served to consumers. *Stylized Page* pattern supports the fact that all consumer-centric Web pages must look good. Web pages should be attractive to the intended audience, whether attractive means stoically conservative or GenX flashy. Turning a *Dynamic Web Page* into a *Stylized Page* addresses that challenge. Some of your pages may require the presentation of graphs. If the graphs represent live changing data you will want to use the *Image Producer* pattern.

Since *Stylized Page*, the most heavily used of the presentation patterns, may include numerous extraneous graphic elements (such as related links or even banner ads), and a given article may span many such pages, it may not be suitable for print. Therefore, it's always courteous to provide a single page that houses the exact same document or article content in a printer-friendly layout. This is the job of the *Printer-Friendly Article* pattern.

There are many uses for the *Business Document Publisher* solution pattern. Sometimes printable material must have the official corporate look. Many times such a document will have a standard layout and format, but contain live data from one or more data sources, either of the production or reporting variety, or both. For example, an insurance quote or policy will be based on data that was submitted by the insured's agent and that generated by underwriting workflows. An airline maintenance instructions manual will have information from the aircraft manufacturer merged with optimized and up-to-date steps sanctioned by the owning airline organization. A corporate earnings report will be based on up-to-the-minute business intelligence data. These and many other scenarios require that live data be merged into a standard template document on the fly, and that the document be highly readable and print in a WYSIWYG manner. The *Business Document Publisher* pattern addresses these important requirements.

Finally, whether you are distributing business documents or software solutions online, you may need to shield such distributions using licensing or other legal protection. The *Intellectual Property Distribution Facility* allows you to secure sensitive and proprietary information by distributing it only to those you intend to distribute it to, and to alert and inform the consumer that they must agree to your specific legal terms in order to obtain what you are distributing.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

I have only scratched the surface of the *Business-to-Consumer* XBP. There are much more complex requirements to analyze and glean from. Let's next explore a more complex presentation facility that will have broad uses in the B2C space, and beyond.

B2C Portal

Content-rich Internet businesses attract consumers to their site by publishing their content into aggregated miniature views. An example of this is the popular Yahoo! site with its My Yahoo! portal. Within the My Yahoo! home page are housed a suite of applications of various kinds. You may reach your email account and your calendar. You can glance at up-to-the-minute stock ticker values of your customized portfolio and read the latest news headlines. The weather report for your area and for the nation can be consulted. You may check the line scores of your favorite sports teams and even read the daily comics. And it's all right there on a single page.

Of course if a win for the Colorado Avalanche was an extremely important one there are probably more details available in an attached article. When you click on the article link you are zoomed in to the full article. The My Yahoo! home page with the rich content set is no longer visible, but you can return to it with a single click. The Avalanche victory article (provided by the *Stylized Page* pattern note above) fills most of the page, but a menu at the top of the article allows you to focus elsewhere easily enough.

Hence, a portal is a collection of applications in a centralized presentation space. The miniature views of applications on a portal home page are called portlets. The best portals are dynamic, personalized, and configurable.

Dynamic: Portals are dynamic when their content is current and potentially reactive. Current content means that your stock tickers and weather reports are up-to-the-minute or better, not a summary of what happened last week. Reactive content is context sensitive and will "change shape" as surrounding influences have an impact. For example, say that Yahoo! acquires an online travel agency and provides it inside the portal. You use the travel portlet to find roundtrip airfare to Boston right on the home page without zooming. Now, if the weather portlet adjusts itself to show you the weather forecast for the next seven days in Boston, you'd consider that pretty handy. That's the power of reactive content. Generally reactive content is supported through an event management framework, where other portlets in the portal receive events when something major happens to another portlet.

Personalized: A portal can be personalized in several ways. One such way is by simply displaying the user's name when they arrive at the site. Another such way is that the user may only be able to use certain kinds of portal content. If, for example, stock market analysis is available for a fee, those who have not paid for the service will not be able to view it. Still another way that a portal can be personalized is by the user determining what portlets they want access to, enabling some and disabling others.

Customizable: Closely associated with personalized portals are customizable portals. You might even think that personalization and customization are the same things. We distinguish the two because customization is under user control, while personalization includes both user selections and portal-only control. One common customization is where the user determines they color scheme they like best. They may also determine the layout of portlets on the home page. Further they may determine what information is viewable in each portlet, and other preferences such as layout.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

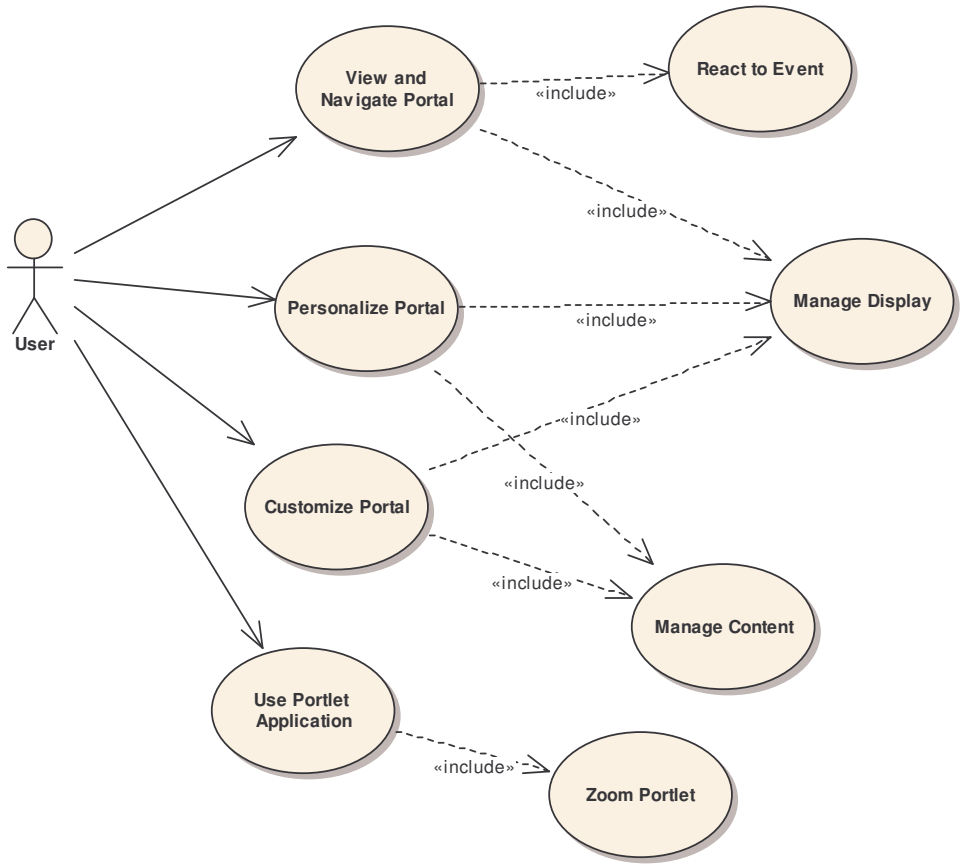


Figure 2.2: The use cases common to a standard business portal.

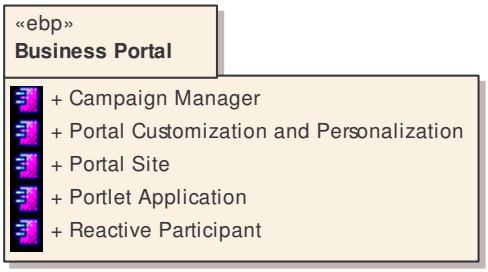
Figure 2.2 draws attention to the basic user requirements I noted as belonging in the standard business portal. There are broader uses of portals than content aggregators/providers in the B2C XBP. Portals make sense for e-commerce based B2C businesses. Different product venues may each be showcased in a different portlet, for example.

More recently there has been a greater use of portals in the corporate enterprise. Sites with HR-sponsored information and other content of interest to the company's employees have grown in popularity. Further, internal and mission-critical applications may appropriately be hosted in a corporate portal.

For example in the healthcare industry remote clinics may interact with a portal to administer patient and doctor data on a centralized hosted application. There may be different applications for billing and accounts receivable, clinic-patient stats, and doctor/practitioner reporting systems. Each application would be hosted in a separate portlet, and be available only to those with the authorization to use it. None of these examples have anything to do with B2C. These are all B2E (*Business-to-Enterprise*) XBP uses of portal. Again, I have identified overlap into other XBPs, although I just happened to discover the existence of portal here first. Hence I chose not to name this EBP in a B2C-centric way, such as Consumer Portal. Rather I named it more generically the *Business Portal* EBP.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

The *Dynamic Web Site* EBP is applicable within the *Business Portal* EBP. In fact, all previously identified EBPs, including *Identity and Access Management* and *Search Engine* are all reusable here. The following identifies the solution patterns within *Business Portal*:



In logical order the *Business Portal* solution patterns provide the following characteristics and behavior. The *Portal Site* pattern encapsulates the overall characteristics of a portal. It defines how the portal home page is laid out (e.g. rows and columns). It defines the menu or navigation controls for the applications. It defines the storyboard or page flow that determines what happens when the user selects a menu option or some selection within a portlet. In general it defines the overall look and feel of the portal and the applications that function within it.

The *Portlet Application* pattern defines how an application is presented within the implementation of a *Portal Site*. A portlet identifies both how the featured application is presented on the home page and how it is presented when it is zoomed. Both the *Portal Site* and the *Portlet Application* instances may be customized and personalized via the *Portal Customization and Personalization* pattern. A *Reactive Participant* is a portlet application that handles events provided by other portlets. This pattern is more than a renamed *Publish-Subscribe*. While publishers, subjects, and receivers define the participants in the pattern, the pattern also describes how participants react to events; that is, the facilities by which they morph when an event occurs.

Finally, the *Campaign Manager* pattern expresses how Web campaigns of various kinds are carried out. A campaign is any business operation with tactics to guide the user through to making a choice. In an e-commerce setting the campaign might be to introduce the shopper to a new product or a new store feature or partner. In a business domain a campaign might be to make employees aware of a new product line and how to handle questions about the product line in order to properly promote it. The diversity of its use and the special use of portal features explains why the *Campaign Manager* is part of the *Business Portal* EBP and not an e-commerce engine asset.

Common e-Commerce B2C

Next I analyze the e-commerce Web site. I'm not at this point taking on the most complex of e-commerce sites, but my analysis has definitely advanced beyond "brochureware." The more advanced e-commerce site analysis follows this subsection.

The e-commerce Web site is at the heart of the B2C XBP and it is certainly what first comes to mind when B2C is mentioned. Some may believe that with the downfall of the .COM boom there's not much point in e-commerce patterns. No doubt e-commerce sites are missing the limelight of that era, but there are still many uses for them. Of course

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

today e-commerce must make business sense. So they've settled into the role of supporting real businesses, which is what they were always intended to support.

Today we see e-commerce engines weaved into a business story, using a soft sell approach. They don't appear until they are absolutely needed; that is, a "shopper" is not necessarily aware that they are a "shopper" until the time comes for the site to "close the deal." Businesses featuring real, established products for sale online are stronger today than ever before. The e-commerce site has survived and has real merit in the new century.

As we analyze the requirements domain of an e-commerce site, note where the use cases from the basic B2C pattern reside in the use case diagram below. The *Register* and the *Sign On* use cases are at the top of the diagram, and *View Authorized Content* lives at the bottom of the diagram. *Register* and *Sign On* have nearly identical behavior as in the Basic B2C use case diagram. In addition, *Register* now extends a portion of the *Manage Shopper Account* use case. That is, when the user registers with the system the *Register* use case is responsible for requesting that a shopper account be created and initialized.

The account is created by a flow in the *Manage Shopper Account* use case. The *View Authorized Content* use case is extended by the *View Store Catalog* use case. In other words, whenever sensitive content may be displayed for and/or requested by the user, the *View Store Catalog* will know because it uses a flow in the *View Authorized Content* use case to execute the necessary security behavior.

Further, the *Search Content* use case from the Basic B2C analysis is found near the bottom left of the diagram. This use case remains intact. Since search engines and their indexing component are fed data in an ad hock fashion, the search consumer—the content managers of the e-commerce site—need only submit candidate search data to the implementer of *Info Indexer* solution pattern. There's no need for the *Info Indexer* to understand the location and shape of the information to be indexed. Additionally there must be an integration established between the e-commerce site and the *Search Results* solution pattern. The integration allows activated (e.g. clicked) search results to jump into a section of the product catalog as easily as it does to normal site information pages.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

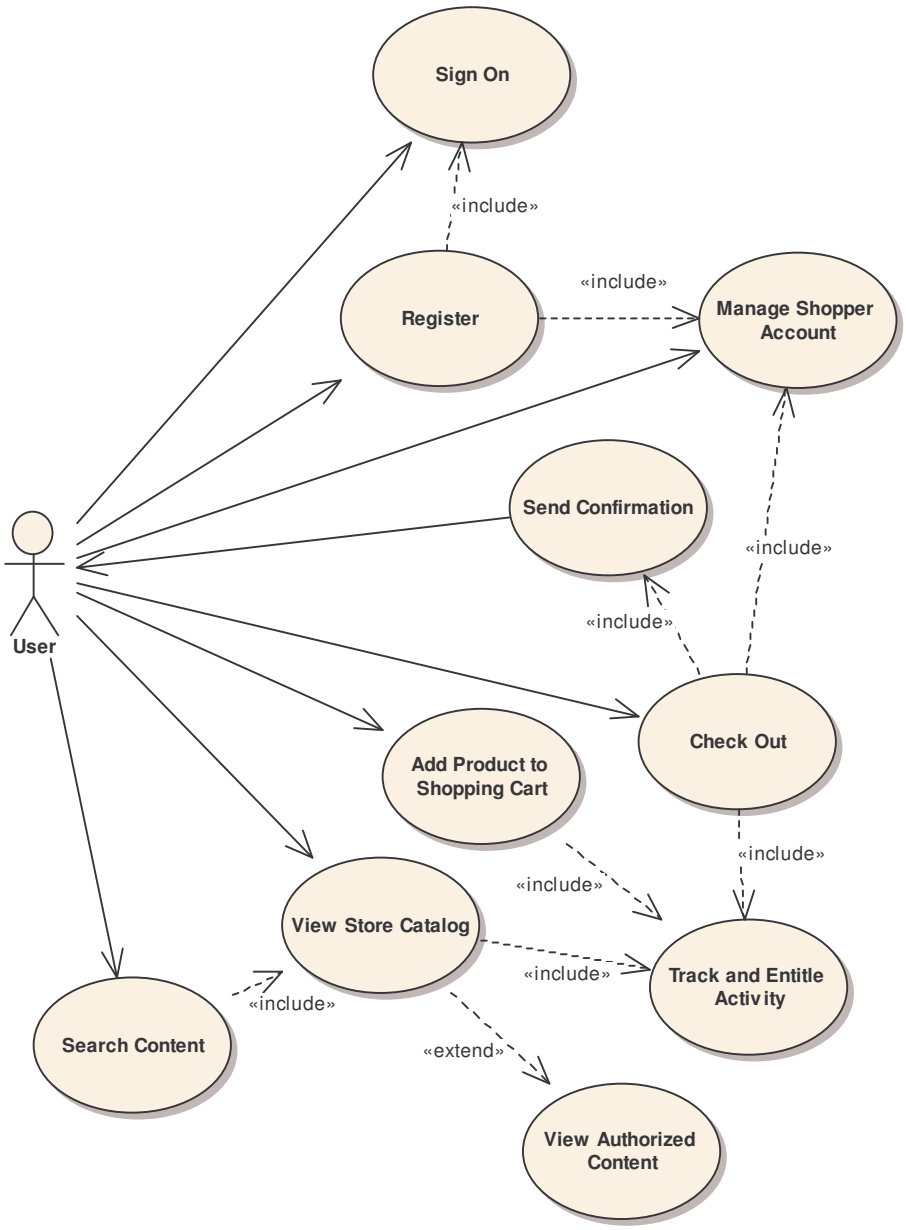


Figure 2.3: The uses cases found in a system that features a business portal.

Next let's look at the unique aspects of the e-commerce use cases. **Figure 2.3** is a diagram containing the use cases likely to be found in a business portal environment. In logical order, a User (consumer) approaches this special form of the *Dynamic Web Site* EBP and is presented the option to browse available products by interacting with the *View Store Catalog* use case. As the User browses, the site navigation components may track their click activity by invoking the *Track and Entitle Activity* use case. This is an optional use case and one that the user does not directly interact with. If you do decide to implement click tracking there is an additional behavior that is implied. Based on previously noted trends, either by the specific user or similar shoppers, you may want to show the user targeted advertising and entitle them to particular offers or benefits. This

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

analysis and targeted presentation flows are also found in the *Track and Entitle Activity* use case.

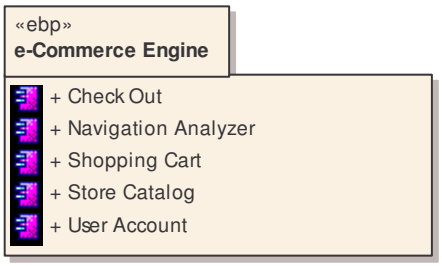
As the User shops for products the site must provide a means for them to indicate what items they are interested in buying. Traditionally this provision has been in the form of an electronic shopping cart. Hence, the *Add Product to Shopping Cart* use case allows the user to select products for purchase.

Following all shopping activities, the user hopefully decides to buy the products they have placed in their cart. They do so by selecting an option on the e-commerce site likely named "Check Out." This user experience leads to the execution of the *Check Out* use case. Several things may occur at this point. If the user has not registered before, or if they have never purchased from the site before, the *Check Out* use case may need to interact with the *Manage Shopper Account* use case. This will establish a consumer relationship with the business. Additionally, however, if the consumer will be using a different payment method than in previous shopping experiences, or if their billing or shipping address, or some other identity information has changed, the supporting *Manage Shopper Account* use case flows may be employed as necessary.

Whether or not any identity or payment information in the consumer's account changes, the *Check Out* use case always interacts with *Manage Shopper Account* if the check out process completes. It saves a reference to the generated invoice so the consumer may view it after the transaction is processed.

Lastly the *Check Out* use case invokes a flow in the *Send Confirmation* use case to reassure the user that their purchase is on its way. Confirmation is generally sent via email. Even if the consumer receives their merchandise immediately (for example, through download) the confirmation is still a necessary courtesy. Attaching a copy of the invoice to the email message is a very good reason to send a confirmation to the user regardless of how the product is received. The *Check Out* use case obtains the consumer's email contact address through a flow in the *Manage Shopper Account* use case. The "pretty" invoice sent to the user may be generated using the *Dynamic Web Site*'s (EBP) *Business Document Publisher* solution pattern.

We are now ready to identify our next cataloged EBP and the solution patterns within it:



The *Store Catalog* solution pattern defines the means of implementing a consistent user experience around browsing for products. There are actually a variety of different ways to publish product information, and we examine a few different strategies in this pattern. As the user navigates the *Store Catalog* the site may use the *Navigation Analyzer* pattern to capture specific and general consumer trends.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

There are actually two primary responsibilities of the *Navigation Analyzer* pattern. The first is to collect the clicks of individual shoppers and augment the trending data. The second is to analyze the trends of the clicks and then target advertising, provide shopping hints, and entitle special features or offers. The targeted advertising and shopping hints are based on clicks the shopper has just made and potentially on activities they have had in the past.

At first glance you might think that the *Shopping Cart* and *Check Out* patterns are really one and the same. In actuality they are two patterns. There are detailed strategies around the *Shopping Cart* alone and that of the *Check Out* pattern that strongly suggest unique patterns. But just like the use of the *Singleton* and *Factory Method* patterns [GoF], you won't generally use *Check Out* without the *Shopping Cart* pattern.

The *User Account* pattern expands on the *User* pattern found under the *Identity and Access Management* EBP. The *User* pattern provides the essential structure necessary to maintain a simple user of a system. The *User Account* pattern adds on what's necessary for the user to have a sophisticated experience with, for example, an e-commerce site.

Advanced e-Commerce B2C

Finally I address the more complex form of the B2C pattern. What exactly is an advanced B2C system? That can be difficult to put your finger on because the kinds of e-commerce systems vary widely. However, one hint to understanding this problem domain is that an advanced system would most likely stand in for a more sophisticated acquisition process that traditionally took place between consumers and some sort of sales person.

What I am trying to identify as "advanced" is not just about the presentation itself. If a Web application has a Flash or Flex [Macromedia] user interface, or streaming video presentations of products, it doesn't necessarily mean that the acquisition process itself is complex. The business behind the Web pages may be very simplistic, which in my opinion does not merit the application as being advanced in terms of e-commerce B2C. Just because one aspect of the implementation requires a greater than an average amount of work to design and implement does not make the *business processes*, and hence the patterns describing them, more complex. On the other hand providing an advanced user interface in front of an advanced set of automated business commerce processes is a very good thing.

Think about some complex acquisition processes (no, not just getting your management to feed your insatiable appetite for more and more technical books!) that have traditionally transpired in a brick-and-mortar setting compared to really simple ones. Buying groceries is simple. Buying insurance is complex. Buying a book is simple. Buying an airline ticket, including making reservations and specifying personal preferences, is complex. Buying a pair of jeans is simple. Buying shares of a stock at just the price you want to pay is complex. Buying a music CD or video DVD is easy. Buying a property in the mountains is complex. Well, you get the point.

What I've noted is that the more complex commerce processes traditionally occurred with the help of an agent or broker, or some other sales assistant. When you check out at a grocery or department store a checker scans your product selections and collects your funds. When you buy insurance, for example high-end property insurance, much more is involved. (There's more on this example below.)

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

Auctioning[1]
Automobile Sales
Financial Services
Banking
Insurance
Stock and Securities Brokerage
Real Estate Brokerage
Reservations and Ticketing
Airline and Other Transportation
Hotel and Vacation Rental
Time-Share Trading

Table 2.1: Advanced commerce-based business processes that have e-commerce counterparts.

Table 2.1 summarizes some of the complex business processes that I've thought about and analyzed, which have e-commerce counterparts. Most of the business processes listed in Table 2.1 deal with expensive or relatively expensive, volatile, complex, time-constrained commodities, or possibly all of the above. Therefore, there are generally multiple humans involved in the acquisition process for each of these items. By default the person attempting to acquire one of the goods approaches the offering company and requests information, a proposal, a quotation, or to out rightly make a purchase. And then the fun begins.

All of these consumer requests may cause a series of rather complex processes, and generally at least one other person gets involved. The other people involved usually include at least one agent, auctioneer, broker, clerk, teller, underwriter, or general sales person. For the remainder of this section I will use the term "agent" when referring to any one of those roles.

The complexity of the business processes often times results from the need to negotiate satisfactory deals. There is generally also complexity because of the required in-depth knowledge of the product and the agent's ability to convey it in an understandable way to the consumer. Once the consumer is educated, they and the agent might barter a bit over optional inclusions and pricing. This requires communication and time. On a computer system it represents a long-lived transaction. I am not suggesting that there should be a database transaction that must span hours or days of time. That would not be a good thing. Rather, I am describing a *business transaction* that must be tracked over a relatively long period of time and with enough precision so that accounts can be settled at some future time. If and when the deal can be completed, a "contract" of some kind is sealed between the consumer and the agent's organization. The "contract" may be as simple as providing the credit card, or as complex as a 100-page legal document. **Figure 2.4** demonstrates the basic business process involved in an advanced business transaction.

[1] While a traditional auction may be a somewhat simplistic acquisition experience—outside of trying to make sense of the environment, trying to follow the auctioneer, and competing with other bidders—mimicking this experience using computer software makes for a rather complex application.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

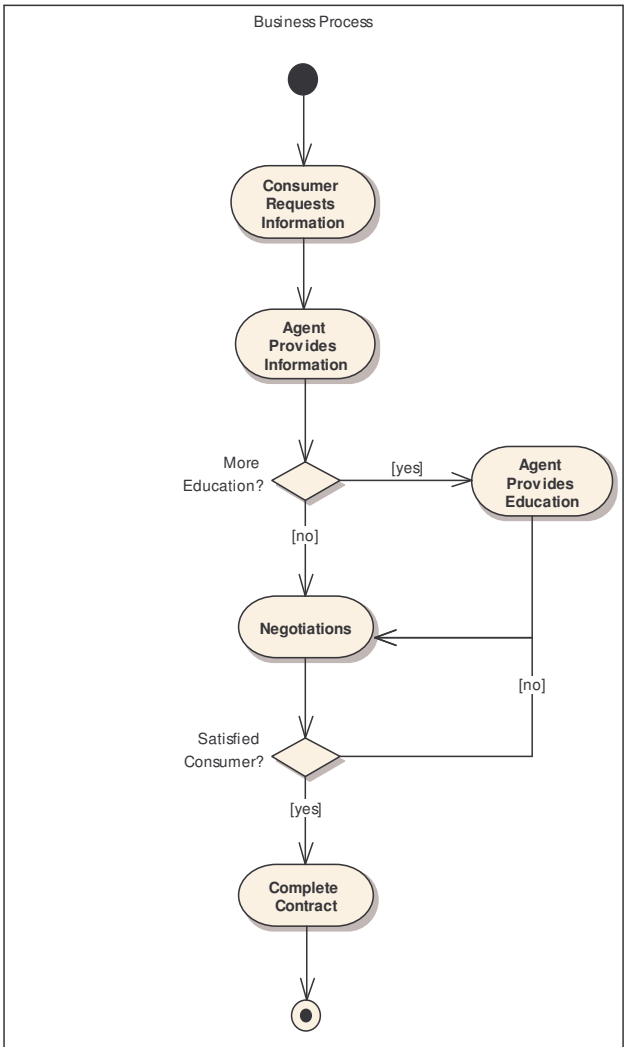


Figure 2.4: The high-level Consumer-Agent activities in an advanced business process.

Now I have reached a dilemma. Do I take a deep dive into each of the specific business processes listed in Table 2.1, or do I stay high level? It is fair to give each of the specific business processes serious analysis. What I have found is that each of the processes has very similar properties and workflows. Thus, it makes good sense for this presentation to stay at the highest level of abstraction possible. That will enable me to present the best Enterprise Business Pattern from the broad spectrum of business domains. I will first present use case analysis for some of the key business domains found in Table 2.1. Following that I will be in a better position to explain the common and unique patterns found in each domain. The use case analysis stays high-level, but the pattern analysis is a deep dive.

Auctioning

The characteristics of an auction are as follows. An entity (person or organization) registers an item to be sold on auction. The seller establishes an opening price for the item, and declares a minimum acceptable sales price reserve either publicly or privately.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

The auction house also sets a price for their service to the seller. It may cost a set price just to register the item for auction, and the house may want to keep a percentage of the final sale. The house may also stipulate that they keep a higher percentage of the sale if they are able to drive the final sales price higher than the seller's minimum acceptable price. This collectively establishes the terms of the sale.

Then the item goes up for auction. Bids are placed until such time that no more bids are made. In the process it's possible for the seller to alter their offering terms if it seems that they could make purchase more attractive. The auctioneer generally has a good sense for how long an auction should continue before closing the bidding process. For practicality in an electronic, on-line auction, the seller generally chooses a set period of time. If the offers reach an acceptable financial level per the terms of the auction, the sale is made. The highest bidder then pays for the item and the item is turned over to them. The seller receives the monies from the sale minus fees for services rendered. There may be a grace period offered for items that are not well known to the buyer before the auction, which allows the buyer to back out of the deal if they can conclusively prove that the seller has grossly misrepresented the auctioned item.

Figure 2.5 contains a use case diagram that shows the requirements for auctioning an item. The scenarios are fairly self-explanatory. A *Seller* registers an item for auction, which includes establishing terms for the sale. Once the item goes up for auction the *Seller* has three primary system functions at their disposal. They monitor the progress of the auction by examining and evaluating bids. If they think they can improve the chances for a sale the may alter the price, or sales terms. If the *Seller* uses this feature then all previous *Bidders* receive notification that the terms have changed, which may "incentivate" them to bid again.

All along the process *Sellers* may answer questions for *Bidders* using the *Collaborate* use case. When a *Bidder* is ready to place their bid they use the *Place Bid with Strategy* use case. Hence, a bid does not have to end with a single offer. A *Bidder* may elect to run up their offer to a pre-determined amount if any other Bidders outbid them intermittently. The agent role in the *Auction House* has the responsibility to establish the terms of sale for each *Seller*. Additionally the *Auction House* monitors the bidding process transactions and finally closes the auction after the agreed upon time expires. The *Auction House* agent is an automated system component in an on-line auction.

As you read on, note how this example has overlapping patterns with the following examples.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

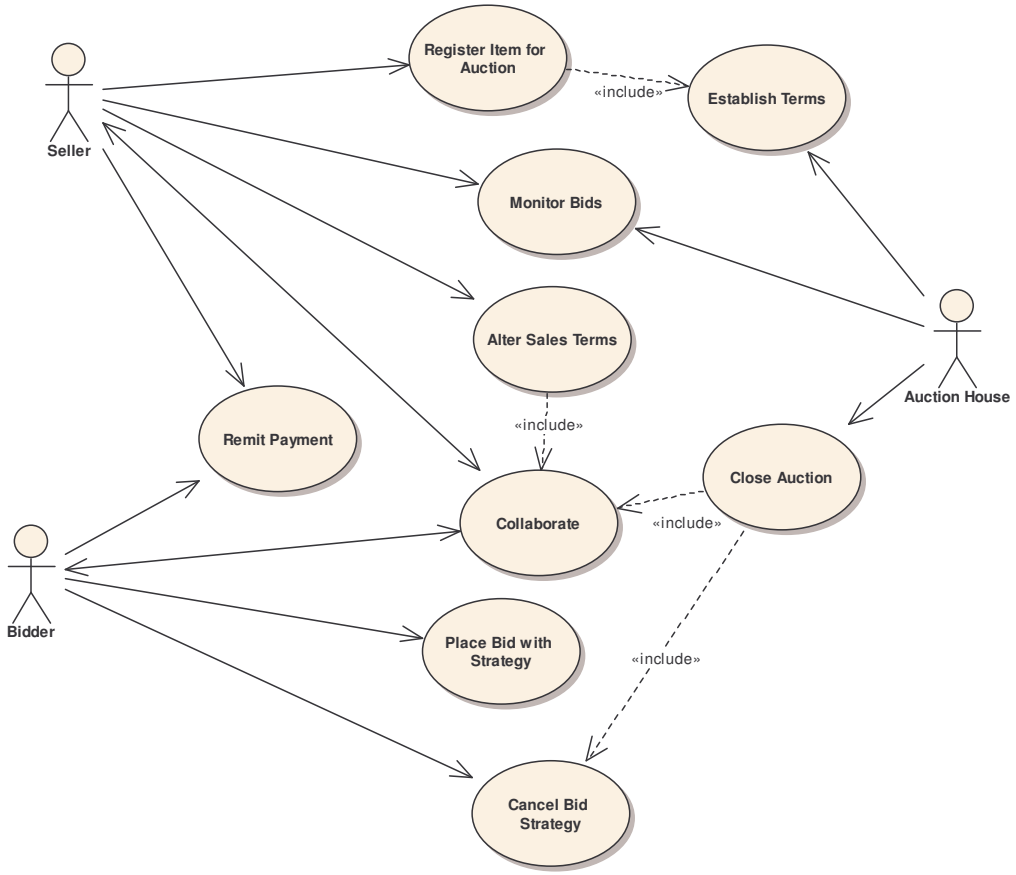


Figure 2.5: The use cases generally supported by an online auction system.

Financial Services, Insurance

Let's use the example of a high-end property insurance broker. The consumer must approach the insurance organization and supply them with information about the risk, which in this case is property. They must declare the construction type of the building(s) for which insurance is sought, the condition of the building(s), the total value of all the property, how much insurance already exists for the property, and how much new insurance is required. Then the underwriter (perhaps via an insurance agent) must provide one or more initial quotes, each quote being based on including and excluding certain options. Generating a quote may involve running some risk management software to calculate how much risk is involved in insuring the particular property, and whether or not the insurer can afford to take more risk in that particular region of the nation. The region probably has one or more major potential disaster threats such as earthquake, hurricane, tornado, and the like. Taking on too much risk in one region (zip code or set of connecting counties) could in turn spell disaster for the insurer if a landfall event would take place there.

At the point where the consumer receives the quotes, both consumer and agent may dicker over price or policy options. The dickering leads to two-way correspondence, all of which must be tracked for regulatory and legal purposes. Also, more and more quotes must be sent to the consumer along with the correspondence.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

Once a deal is struck the insurance agent may have to send out an inspector to make sure the property measures up to the consumer's description. This involves collecting inspector data and photographs of the building(s) and other property features. Finally the consumer must either be covered by a policy or denied coverage.

All along this process lots of information has been generated and collected. What happens when the deal completes? What about if the insured would soon after like to add or remove some coverage options? What is done if the insured decides to cancel the policy just after the policy is bound? If the policy is bound, the information must either be promoted forward and premium must be calculated and invoiced, likely on a monthly basis. The policy must be printed and mailed or generated electronically and emailed. If an agent (other than the underwriter) was involved, commissions must be paid. If the consumer decides to cancel their policy the system must compensate for the action, as if to rollback events that took place over days or weeks or even several months of time. "Complex and advanced" may be an understatement, and I haven't even touched on policy renewals or claim management!

Figure 2.6 is a use case diagram for quoting, binding, and maintaining an insurance policy. In this use case diagram there's a mix of traditional and automated business process. The primary difference between these processes is that the *Consumer Agent* is always a person in the traditional process, but includes a major system component in the automated process. The component is a front end Web interface that potentially replaces the need for a human agent, allowing the *Consumer* to act as his or her own agent. Of course any given *Consumer* seeking insurance may decide to use a human agent, who in turn uses the automated component agent in behalf of the *Consumer*.

Do you see a pattern forming? If you look back at the *Auctioning* business process note the similarities—and much more than *Remit Payment*! The domain model of each business appears to be quite a bit different, but are they that much different? In the auction environment there's an *item* for sale and *Bidders* try to obtain it for a satisfactory price. In the insurance business the *item* for sale is a policy.

Note: It's interesting to flip the tables a bit on the insurance business. From the perspective of the insurance company we might conclude that the insurance company is really the buyer. They may agree to buy the insured's monthly premium in exchange for replacement coverage on a property. This becomes clear when you consider that several insurance companies may bid on the insured's monthly premium. However, for obvious reasons I won't view the insurance company as the *Consumer* in this case.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

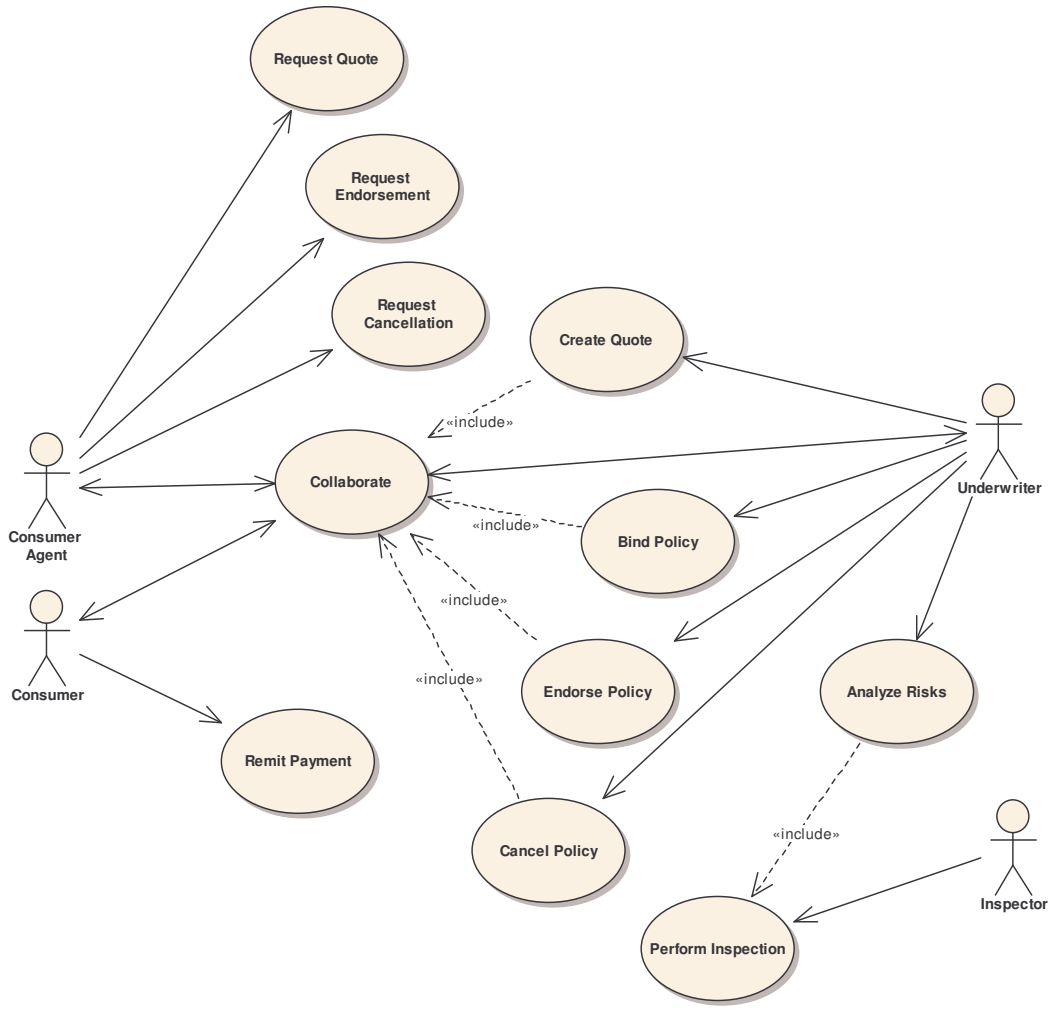


Figure 2.6: A use case diagram highlighting use cases for an online insurance quoting and policy binding system.

Collaboration is a 1:1 match between the two. The topics being discussed are different, but the mechanisms are similar or identical. In the insurance example the "dickering" could be handled via emails, or they could be managed in a much more organized collaboration. If the *bid model* where used, it would give the process much more uniformity. Tracking the collaboration would be much easier than using emails, especially if regulators require all collaboration to be archived for some specific period of time. Of course there are only two *Bidders* in the insurance example, the *Consumer* and the *Underwriter*, and bids can go higher or lower, but the model works well. Even if some of the collaboration must take place over the telephone, the *bid* would still serve as the official statement of record. When the phone conversation takes shape and terms are settled, for example, the *Underwriter* would conclude the collaboration by entering the mutually agreed-upon *bid* into the system. Following this an email would be generated by the system containing a link to the official record, the *bid*. This step encapsulates the auctioning system's *Alter Sales Terms* use case.

Other similarities are that the item for sale must be described. In the auction business the *Seller* describes the item. In the insurance business the description is partly described

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

by the *Consumer* and partly by the *Underwriter*. The fine-grained domain objects are different, but most of the course-grained domain objects can be shared.

There are also similarities in the arbitration period allowed by the auction and the insurance examples. In the case of the auction the *Bidder* (buyer) may have the opportunity to check out the item they purchased. In the insurance example the *Underwriter* has the opportunity to examine the property they have agreed to insure with the use of their inspection process. Both of the business models allow one of the parties to turn the deal around if the other party has grossly misrepresented part of the agreement. Granted, there are sound business reasons for not emphasizing the inspection process in an auction environment. But there must nonetheless be a means for facilitating it when necessary.

Financial Services, Stock Brokering

The consumer requests a quote for a particular stock ticker symbol. The agent answers with a quote that is either real-time or delayed (perhaps 20 minutes). The consumer requests the purchase of a number of shares at a particular price. The purchase price may be specified or it may be open to market value. If the consumer specifies a non-market price they may also configure a time limit for purchase. If the time limit is reached before the stock is valued at their specified price, the order is canceled. After the order criteria are specified, the consumer submits their order.

The agent attempts to fill the request. If a specific price was given, no action will be taken until the stock's price reaches the consumers specified price, or until the time limit threshold for watching the stock is reached. If the price of the stock satisfies the consumer's order criteria, the stock is purchased in the requested quantity of shares.

Besides getting stock quotes and ordering stock purchases, the consumer may also request an alert to be sent to them when specific stock criteria are met. This can help the consumer in buy/sell decisions by allowing them to react just in time. **Figure 2.7** is a use case diagram for a basic stock brokering service. The *Broker* agent is human in a manual system, but in the depicted automated system the agent (*Broker* actor) is a software component.

The emerging pattern can be observed in this business process as well. When the *Consumer* uses the *Submit Order Criteria* use case it is not at all unlike the *Place Bid with Strategy* use case in the auction example. The strategy here might be to buy a stock at a specific price within a set timeframe, while the *Bidder* in the auction process may set an upper limit for counter bids when other *Bidders* raise the bar. And like the insurance example the *Consumer* here also requests and receives quotes from a back-office agent (*Broker* vs. *Underwriter*).

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

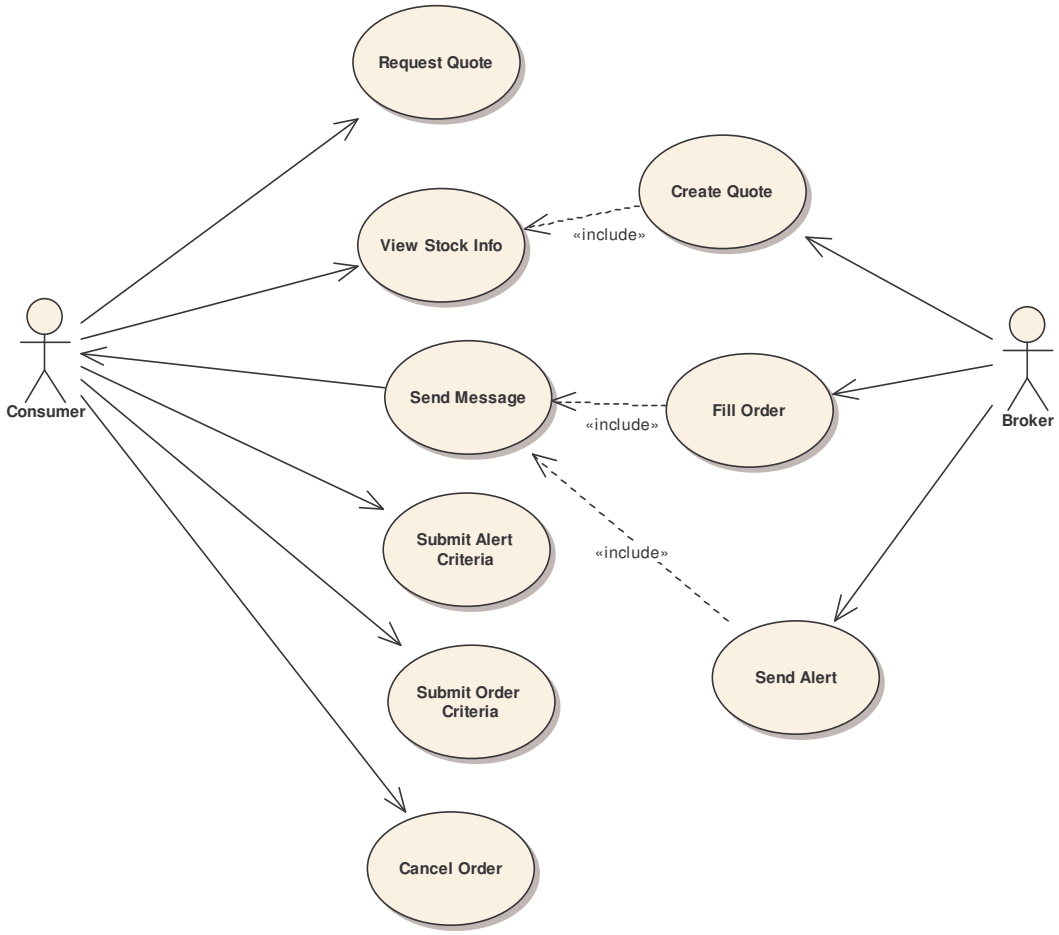


Figure 2.7: The functionally significant use cases of an online stock trading system.

Collaboration is managed in this business using *Send Message*. The *Send Alert* use case coupled with *Send Message* has very similar characteristics as the *Alter Sales Terms* and *Collaborate* use cases of the auction business. The *Fill Order* use case here is very similar to the *Bind Policy* in the insurance example. The *Cancel Order*, *Cancel Bid*, and *Cancel Policy* use cases also overlap.

Next let's look at a few reservations and ticketing examples.

Reservations and Ticketing, Airline Transportation

Booking airline travel has become a very easy process thanks to online reservation and ticketing services. The consumer specifies the airport (or city) of departure and the destination, as well as the date of departure and date of return, and submits their request. The agent responds with a list of available flights including itineraries and costs. The flights may be on the same day but with varying times and layovers (there may also have been an option to list flights on surrounding dates).

If the consumer finds a desirable itinerary and price, they may select the itinerary and book the flight. Booking the flight allows the consumer to enter their frequent flyer information for mileage credit, to request a special meal, and to select their seat. It also provides the means to pay for the ticket using a credit card. The use case diagram in **Figure 2.8** shows the requirements for an airline booking system.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

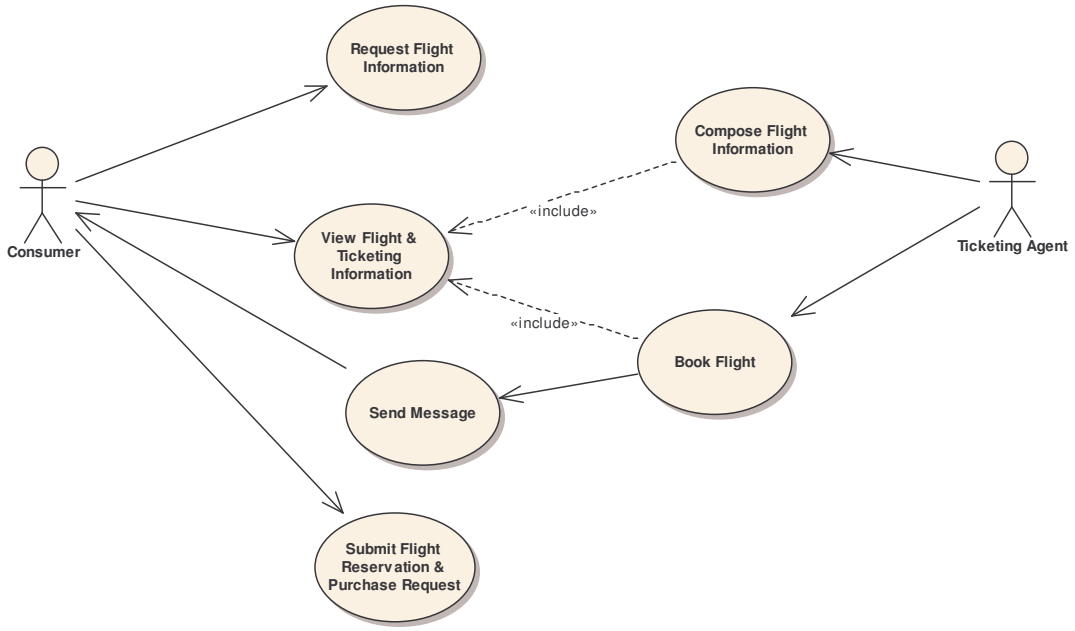


Figure 2.8: The high-level use cases commonly found in an airline reservation and ticketing system.

Have you already identified the overlapping functionality in this business model? Like the insurance and stock trading examples, the airline transportation example has *Request Quote* and *Create Quote* use cases. In the airline example it's more advantageous to name the corresponding use cases *Request Flight Information* and *Compose Flight Information*, respectively. However, they are the same basic pattern. The same goes for *Place Bid with Strategy*, *Submit Order Criteria*, and *Submit Flight Reservation & Purchase Request*. Comparing the other business models, the fine-grained domain objects are different, but the course-grained domain model is part of a macro, Enterprise Business Pattern.

The pattern continues to emerge with my last detailed example, *Time-Share Trading*.

Reservations and Ticketing, Time-share Trading

The time-share industry allows individuals to purchase a portion of a vacation property. The vast majority of the properties are condominiums of various sizes. Generally a property is offered for sale for one-week time blocks, and buyers may purchase as many of the time blocks as they like (or can afford). One buyer time-shares the condo with other buyers. An added value has to do with where in the year the time block is situated. For example, there may be high seasons, shoulder seasons, and low seasons for each time-share location. The higher the season you purchase, the higher the cost. If you want to purchase a time-share in a popular location, say a warm winter region, the location will also add value. An example is purchasing two-weeks of a two-bedroom condo in Puerto Vallarta, Mexico, that you can use during the months of December through March.

Some time-share owners don't desire to return to the same location year after year. And even though they could store weeks in their account, they would sometimes like to take vacations in other locations while using the power of their previous investment.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

Sometimes owners just can't make the trip and they'd simply like to allow someone else to use their slot for a price. Enter the time-share broker and exchange concept.

The time-share broker industry allows owners to place their time-share into a pool and exchange their time block for cash[2] or to trade their equity for a condo of similar value in another location. Usually a time-share broker has access to properties operated by many different real estate management companies. The time-share broker makes money from annual fees, and/or for an owner exchanging their time blocks for another property. One system used for the brokering process assigns a number of points, *P*, to a given property. Each week, *W*, owned represents a fixed number of points. So the trading value of the property, *V*, is *P* x *W*.

You can imagine the manual process associated with this brokering function. An owner calls a toll-free number, tells the agent their member identification, and informs the agent that they'd like to sell or exchange all or a portion of their property's yearly value. The broker then calculates the value to the owner and makes the property available for bidders. An interested party may now book the property, either for a cash fee or by exchanging any points they have. Meanwhile the owner will receive the value of their points either as cash or trade for another location (both minus broker fees).

Now imagine a surrogate agent service that automates the manual functions of the time-share brokering industry. What you are imagining is probably a lot like one of my colleagues architected, designed, and built for time-share mogul RCI. **Figure 2.9** contains a use case diagram that captures the major actors and scenarios of a time-share trading environment.

[2] I will not attempt to convince you that time-shares are a good value, either to own or to trade for cash or other locations. I am using time-share brokering as an interesting agent-based business domain. Gladly my in-laws bought the time-shares and allow me to use them. YMMV.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

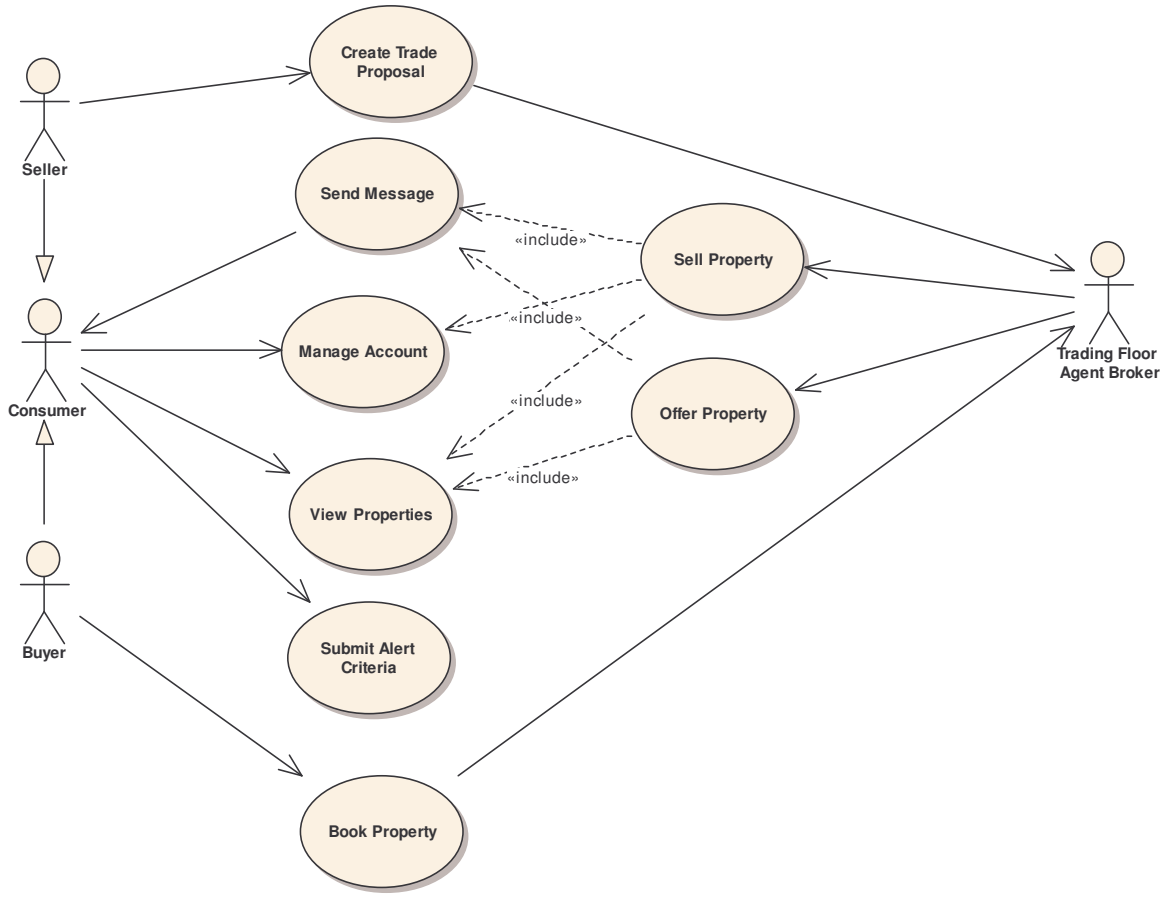


Figure 2.9: A time-share trading system is depicted by this set of use cases.

A *Seller* and a *Buyer* are both *Consumers*. The only big difference between general *Consumers* and the *Buyers* and *Sellers* is that *Sellers* use the *Create Trade Proposal* use case and *Buyers* use the *Book Property* use case. The *Trading Floor Agent Broker* is an automated system component that is responsible for taking trade proposals, turning them into property offerings, and negotiating bookings, which are sales.

The *Create Trade Proposal* use case is nearly equivalent with the *Register Item for Auction* use case in the auctioning example. The combination of *Submit Alert Criteria*, *Offer Property*, and *Send Message* is in principle the same as the *Submit Alert Criteria*, *Send Alert*, and *Send Message* use cases of the stock-brokering example.

Just about all the examples, whether shown out rightly or not, have a mechanism for viewing information. The auction system allows *Bidders* to view kinds of items. The stock brokerage, airline transportation, and time-sharing trading business all allow consumers to view detailed product information. Further, the act of being the winning bidder, binding an insurance policy, purchasing a stock, booking and ticketing a flight, and purchasing a time-share slot all require the consumer to enter into a contract with the seller/organization represented by the agent.

With the discovery of all these similarities between business domains and enterprise business systems backing them, it's time to take a deep dive into defining the Enterprise Business Pattern that encompasses them all.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

Surrogate Agent Service Enterprise Business Pattern

Clearly the overlapping functionality between business domains is striking. Of course the broader the survey of domains the more complex the patterns are to formalize. The overarching pattern, the EBP, is named for the fact that an *agent* traditionally plays a major role in these business domains. Again the agent may be an auctioneer, broker, clerk, teller, underwriter, or general sales person. But by any other name, these are all *agents* of the business domain they represent.

Some or all analyzed business domains require this functionality:
1. Register and maintain service request or product for sale; timed and open-ended sessions
2. Alerts and messaging, including agent-consumer collaboration
3. Bid or order submission with automated strategy management and execution
4. Invoice and manage payments, such as premium
5. Request for quotation, and create quote
6. Sale closer, contract binder, amendments, and cancellations
7. Risk analysis, inspections, background checks, credit checks, quality review

Table 2.2: The common functionality among advance, agent-based business domains. These similarities help identify the larger Enterprise Business Pattern and the Solution Patterns within it.

Since the *agent* in an online system is an automated, course-grained component, the agent is truly a substitute, or *surrogate* for the traditional human role. Further, the agent's role has traditionally been that of a service provider. The agent services are rendered in behalf of the seller and consumers in the respective domains. Therefore, I have named this EBP *Surrogate Agent Service*.

I recap the similarities among the advanced business domains in **Table 2.2**. Whenever we find so much overlapping functionality across several different business domains, there are definitely patterns to be harvested. The seven major functional categories identified in **Table 2.2** are the basis I have used to extract the solution patterns found within the *Surrogate Agent Service* EBP. Here's the EBP and its compliment of solution patterns:

Since an agent and consumer must exchange information, it is important to formalize the pattern that manages their interactions. The *Agent-Consumer Collaboration* solution pattern serves that purpose, and it addresses point #2 in Table 2.2 above. It's not a definition of an email client and server, but it does include email. Email may be the means through which alerts and messages are sent to the consumer, as well as other means. However, there is always a backing, traceable statement of record that the agent

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.

service can use at a later time. Recalling the statement of record may be necessary for regulatory compliance or just for customer relationship management support.

The *Collection Agent* is no muscle head. It's a solution pattern that defines how an agent-based service system calculates payments, sends invoices, and tracks remittance of such by the consumer. Payments may be for ongoing premiums, payment plans, or one-time payments. One-timers are really managed by the standard *e-Commerce Engine* EBP as part of the *Check Out* solution pattern. So you'll see some cross-referencing and reuse here, as I demonstrate how point #4 in Table 2.2 is spoken to.

The next pattern is what manages the closing of the deal between the agent's organization and the consumer. This is called the *Deal Maker* pattern and it formalizes the points noted in #6 of Table 2.2. It is responsible for binding contracts, endorsing or amending contracts, and even renewing them. The *Deal Maker* is seen across each of the business domains in which there is more than a simple product sale occurring. While there is always a sort of contract with any kind of sale, this pattern is more concerned with the generation of a binding contract between the two parties involved. This pattern makes use of the *Business Document Publisher* solution pattern from the *Dynamic Web Site* EBP.

The *Order Taker* is the solution pattern that handles the wide range of transactional orders that may take place once the contract is bound. This may occur as the result of a bid (or some use of the bid-model), a reservation, or another kind of purchase. Orders may be configured to accept a strategy along with the purchase or reservation transaction. This pattern also facilitates the cancellation of an order such as a bid. This satisfies point #3 in Table 2.2.

The next pattern, *Quote Agent*, allows a business to accept requests for quotation and to respond to them. Quoting spans many business domains and is essential when consumers must understand all surface and underlying costs associated with the goods or service of interest. Point #5 in Table 2.2 is captured by this recurring solution pattern.

The *Risk Assessor* solution pattern details how to respond to the business' need to examine all serious risks involved in offering a product or service, or obtaining them. It addresses point #7 in Table 2.2 as it hosts such risk management tools as inspections, background checks, and credit checks.

The ability to register for and/or participate in the agent-based service environment is supported by the *Sales Session* solution pattern. This pattern captures the essentials involved in registering a product or item for sale, alter the sales terms, including withdrawing and canceling the experience altogether. It also supports how an agent publishes information out to the consumer and how the customer consumes the information. This pattern reuses the *Store Catalog* solution pattern from the e-Commerce Engine EBP.

Summary

In this chapter I have taken a deep dive into the Business-to-Consumer XBP. I have conducted a fairly thorough decomposition of the B2C pattern, taking you from basic to mid-level, to complex. The EBPs that have been discovered through analysis are as follows: *Identity and Access Management*, *Search Engine*, *Dynamic Web Site*, *Business Portal*, *e-Commerce Engine*, and lastly *Surrogate Agent Service*. All of these EBPs and their supporting solution patterns are provided for you in Section II, the Pattern Catalog.

© Copyright 2003,2004 Vaughn Vernon. All rights reserved.