Patterns for Internet shops

Eduardo B. Fernandez, Yi Liu, and RouYi Pan
Dept. of Computer Science and Engineering.
Florida Atlantic University, Boca Raton, FL 33431
ed | yliu | rpan@cse.fau.edu

Abstract
Shopping on the Internet has become common and any web site must provide convenient user interfaces for this purpose. An appropriate infrastructure is needed to support a variety of navigational views. We present two patterns that are part of the infrastructure of web shops: the Catalog pattern and the Shopping Process pattern. The Catalog pattern describes how to organize the information about products for a web shop, the Shopping Process pattern describes the steps required to buy something in the Internet. We also show the combination of these patterns in a web shop.

Introduction
Web shops have become a common activity because they bring many opportunities for shop owners and customers. There are many patterns that can be discovered in this type of applications. We can divide them into three types:

- Patterns for navigational views.
- Patterns for building interfaces.
- Patterns for infrastructure.

A variety of patterns of the first two types have been presented [Lya98, Lya99, Pal, Ros00]. Good infrastructure can improve the response of the server and also provides extensibility and flexibility for building good navigational views and interfaces. This type of patterns should be clear, flexible, efficient, extensible, and independent of the other two types.

We present in this paper two patterns for the infrastructure needed to build a web shop: the Catalog pattern and the Shopping Process pattern. The Catalog pattern organizes information about products in an efficient and flexible way. It also provides mechanisms to help customers make decisions. The goal of the Shopping Process pattern is to simplify the shopping process and to improve the efficiency and convenience of the buying process.

These patterns are examples of Semantic Analysis Patterns (SAPs) [Fer00a], a simple semantic unit that applies to a variety of situations. They are also composite patterns, where their components have value on their own right. The combination of these two patterns can be used to define an Internet shopping framework.

Catalog Pattern

Intent

The catalog pattern organizes information about the products sold in a web site.

Context

Web shops where companies sell products of different types. There may be other applications in the server to provide process structure, billing, inventory, and other related functions.

Problem

Web shops sell a variety of products, sometimes totally unrelated, e.g., books and food. An important problem is: How to organize product information, provide on-line guidance to the users, and improve the attraction of the web site so that users are willing to visit and return?

Forces

- There is a large variety of products that can be sold.
- Products and their descriptions change frequently.
- Without good support for product search and selection customers will not return to the site, the infrastructure should support a variety of these functions.
- Related products should be indicated to the customer to entice him to buy more items.
- Catalogs are frequently built in an ad hoc manner and cannot be reused, which results in duplications and waste of time and effort.

Solution

Define a catalog class as central point to collect products. Separate the different aspects of interest such as detailed descriptions of products and related products in distinct classes. Use an Observer pattern to look for changes and notify customers of changes.

Class Diagram

A **Catalog** is a collection of products. Each product belongs to at least one catalog. The **Product** class defines the type of product being sold, buyers usually buy a typical product, not an individual product. This class contains the basic attributes of each product. In particular, a status attribute indicates special aspects, e.g., a new product. The navigational views may place a special product in an outstanding position on the screen. Also, new products may be separated from the regular products and made known to the customers [Lya98].

The **ProductInfo** class provides more detailed information about a product. It may also include comparison among different varieties of the same product, different brands, or

provide the best price/performance ratio. For example, for a TV set, it may provide comparison information from several brands. For travel agencies, it may provide additional choices for customers, such as letting them choose a nearby departure airport, so that air tickets will be cheaper. Class **SimilarProduct** provides links to other similar products.

Modifications to the products are notified to the customers by email to let them know there is some new or interesting product. A class **ProductObserver** watches for changes and notifies customers. Note that these classes are a type of Observer pattern. Class **Notification** keeps records of notifications sent to users.

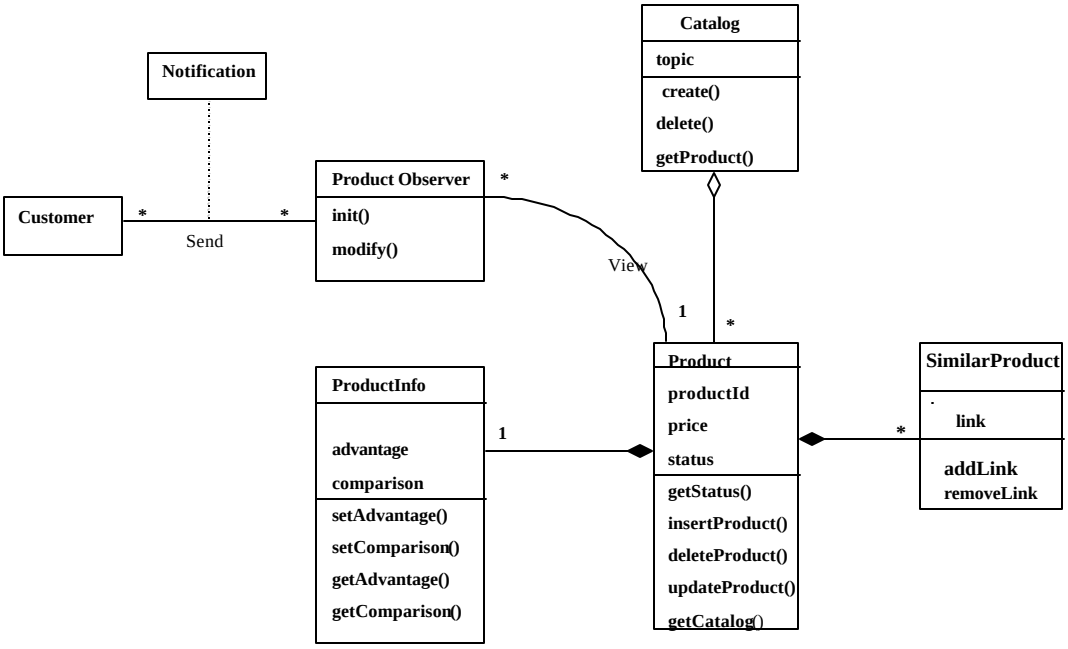

Figure 1: Class diagram for Catalog pattern

Dynamic analysis
The sequence diagram of Figure 2 shows the collaborations triggered when products are modified. This is very similar to the standard Observer pattern sequence.

Consequences

This pattern provides the following benefits:

- It provides the needed infrastructure to describe products conveniently. Navigational classes can be added to show this information in attractive ways.
- It is reusable. The pattern is suitable for various kinds of web shops, from big malls to personal stores.

- It can be combined easily with other patterns, such as Personalization patterns [Pal, Ros01], our Shopping Process pattern shown later, or Inventory patterns [Fer00b].

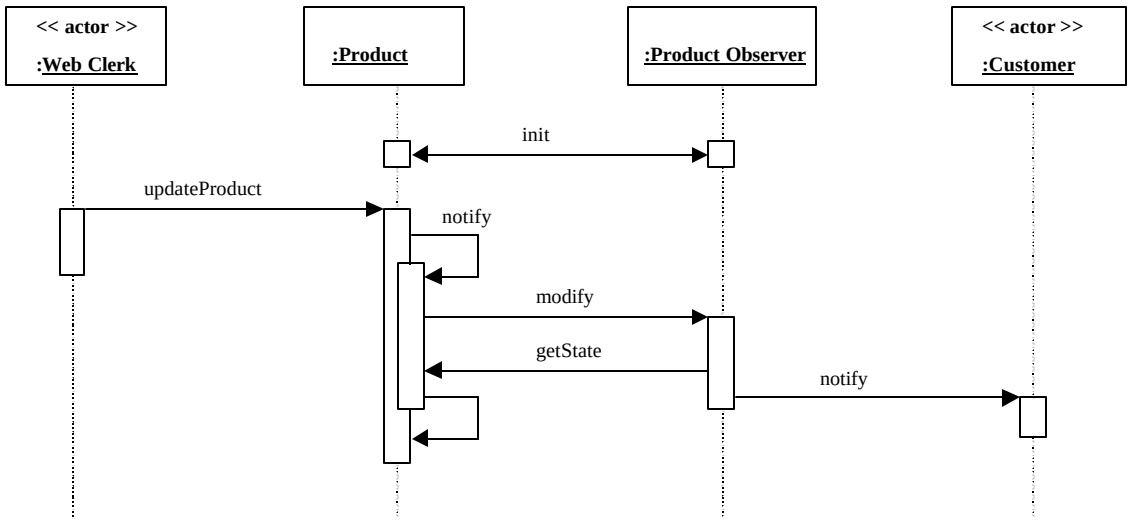

Figure 2: Sequence diagram for update of a product

Other considerations include: While the basic pattern refers to product types, some sites do sell individual products, e.g., a used-car site, an auction site. In this case the **Product** class becomes a set of individual products and we need an additional class to indicate the product type.

Known uses

The pattern can be applied to many applications, such as:

- On-line book stores. Each bookstore organizes books by catalog, such as computers, medical, history, and literature. The products are books. Product-info will provide reviews and descriptions of each book. The status attribute could indicate special deal, new book, or a best selling book. If some customers are interested in some kind of books, whenever such books appear or some special deal is available, an email will notify them.
- Music stores, shoe stores, wine stores…

Most Web Application servers , e.g., IBM's WebSphere Commerce Suite, incorporate catalogs [ibm].

Related patterns

The Catalog pattern includes the Container/context pattern [Coa97]. The Stock manager pattern complements this pattern by keeping track of the quantities of products in stock

[Fer00b]. Other patterns usually found together with the catalog are the Search patterns [Lya99].

Shopping Process Pattern

Intent

Models the structure needed for selecting and buying a product from a web shop.

Context

Web shops contain many products. Customers can select and purchase different products in the same session. A customer looks for the product he is interested, chooses something he wants to buy and adds it to his list of purchases. He can inspect and modify his purchases at any time. Some customers may be known to the shop and treated differently. The web shop will also include catalogs, billing, shipping, and other applications.

Problem

The shopping process must have well defined steps. This is necessary because we need to show the customer where he is in the process. The problem is now: How to describe the shopping process in a precise way?

Forces

- Customers may get disoriented through the steps of the shopping process. We want to keep the shopping process as convenient and clear for the user as possible. Clearly, this will depend on the specific web pages shown to the user, here we worry about the underlying structure to make this job convenient.
- We need to show to the customer where he is in the buying process.
- The steps of the process may need to be changed for new products or new business models. This may imply new entities as well as new steps.
- Customers like to have several ways of paying.
- Some customers must be treated differently.
- If the shopping structure is not secure, customers will not return.

Solution

The most common metaphor for the shopping process is based on the concept of shopping cart, analogous to the carts used in supermarkets. A customer may have several shopping carts and each shopping cart will contain his selections. An order is produced when the customer decides to buy (checkout) his cart selections. An invoice is produced for each order.

Class Diagram

A class **ShoppingProcess** is a single entry point for the complete process. The **ShoppingCart** class collects information about all the products a customer has selected. A **CartItem** object indicates the quantity and the product selected by a customer. A

customer can query the products in his cart and remove products from the cart. The **Customer** class indicates the customer responsible for a shopping cart. This class also provides operations to allow the customer to modify his information. Several forms of payment are provided by the system, such as credit card and e-check. When the shopping cart is checked out, an **Order** and an **Invoice** will be generated.

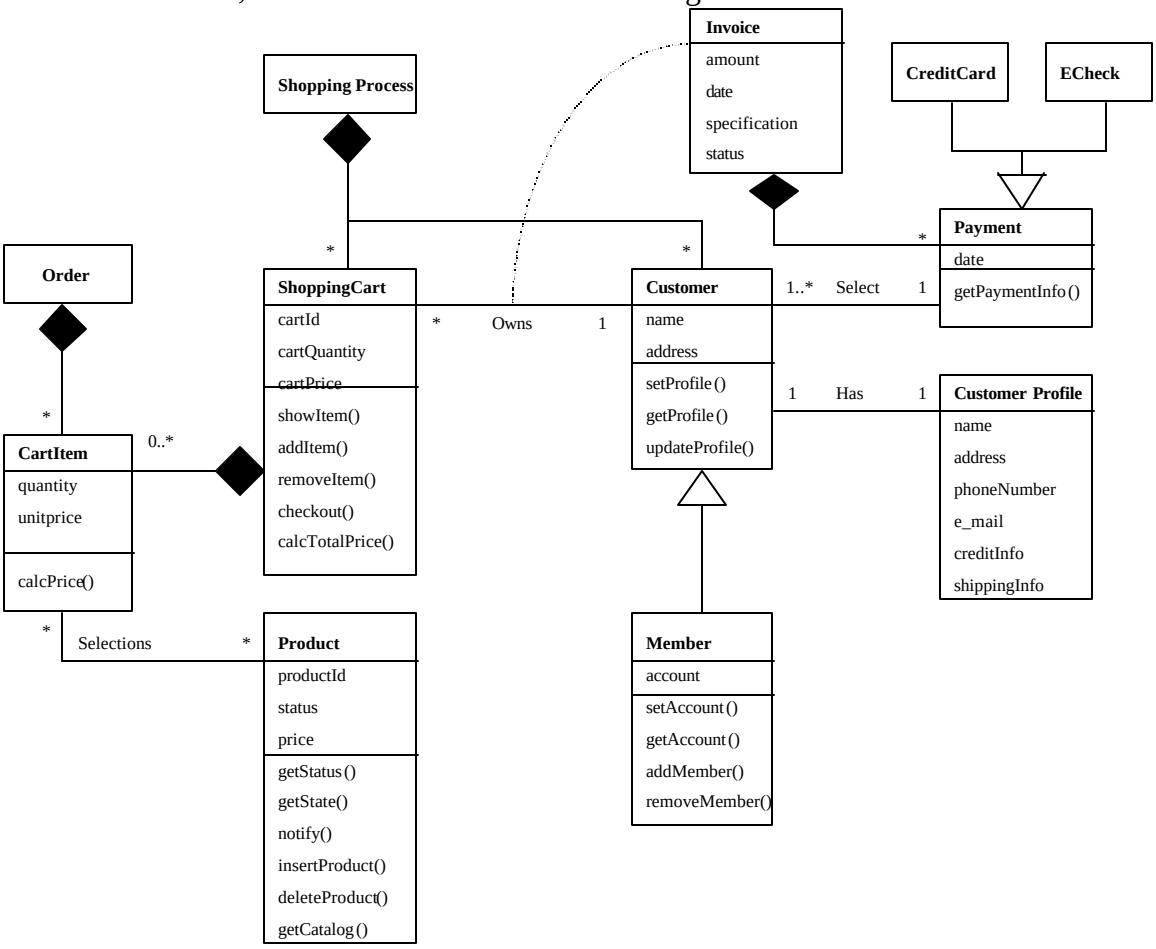

Figure 3: Class diagram for Shopping Process pattern

Dynamic aspects

The following interactions can be found in this pattern (Figure 4):

- *Selection* : When a customer performs a selection operation on some kind of product, a new cartItem object is created and added to the cart.
- *Checkout* : When a customer checks out a product, the prices of all products selected are calculated, the customer billing and shipping information is retrieved, and an invoice is created.

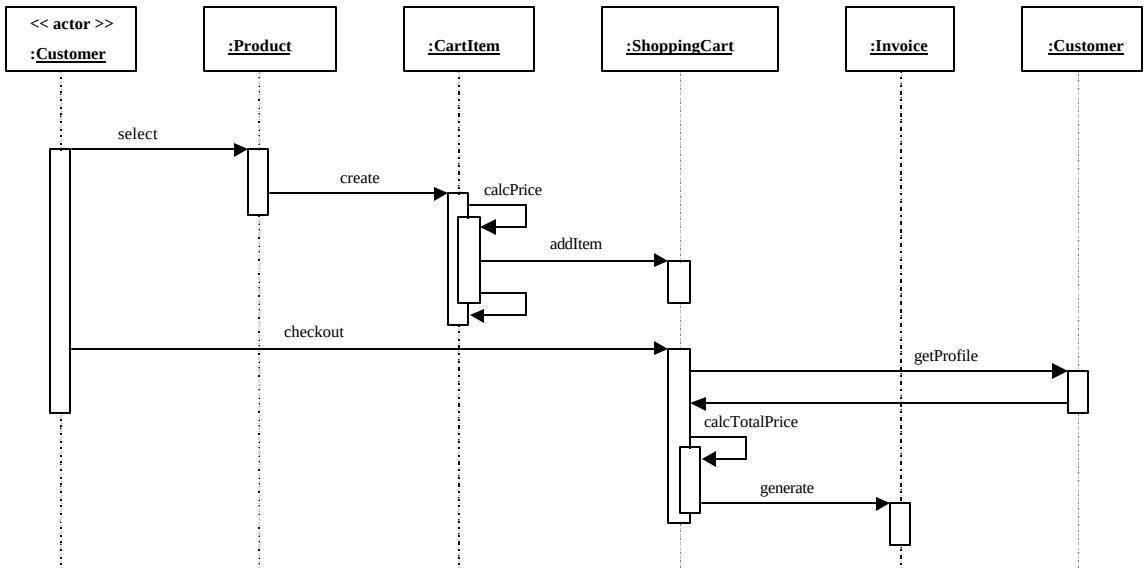

Figure 4: Sequence diagram for buying and checking out a product

Consequences

The following benefits can be found from use of this pattern:

- The pattern describes an abstract shopping process. It can be applied to various web shops from those selling shoes to those selling software.
- The pattern provides common elements for building shops in the Internet.
- The pattern should help reduce the complexity of the shopping process in the Internet. All or some of the process steps can be displayed to the user for his guidance. Customer information need not be reentered for member customers.
- It is easy to add authorization to the conceptual structure; for example, access to the process is restricted only to some process managers.

The following liabilities can be identified:

- The pattern does not provide mechanism to prevent errors, such as wrong credit card number, errors in billing address or shipping address. They should be included in a practical model.
- This is a general model for a web shop; it may need to be adapted for the characteristics of specific web shops.
- For efficiency the pattern needs support from a lower infrastructure, such as a database management system.
- The security constraints defined in the pattern must be enforced by lower-level mechanisms, such as database authorization, file permissions, cryptography, and others [Fer99a].

Known uses

Most web shops use the concept of shopping cart with an infrastructure similar to this pattern. Specific models or code for similar patterns can be found in [Bar96], [Bol00], and [Kob01].

Related patterns

This pattern can be used to support two patterns for web customers:

- The Dynamic Configuration pattern [Lya98], which helps the customers select from a variety of options and validate their selections.
- The Explicit process pattern [Ros00], which helps the users understand the buying process.

Navigational patterns, such as News [Lya98] and Advising [Ross00], can utilize these two patterns as part of their infrastructure.

The order and shipment pattern [Fer00] can complement this pattern to add details of shipment. A more detailed treatment of payments can be found in [Fow97] and [Hay96]. The Shopping Cart itself, the customer-related classes, and the payment classes are atomic patterns that can be used on their own right. Combination of the abstract versions of the two patterns presented here could be the basis of a shopping framework.

Acknowledgements

We thank our shepherd Gustavo Rossi for his insightful suggestions that made this paper considerably better.

References

[Bar96] C. Baron and B. Weil, “Implementing a web shopping cart”, *Dr. Dobbs Journal*, September 1996, 64-69 and 83-85.

[Bol00] G. Bollinger and B. Natarajan, “Build an e-commerce shopping cart”, *Java Pro*, June 2000, 38-50.

[Coa97] P.Coad, "*Object models: Strategies, patterns, and applications*" (2nd Edition), Yourdon Press, 1997.

[Con99] J. Conallen, *Building Web Applications with UML*, Addison-Wesley,1999.

[Fer99] E.B.Fernandez and X.Yuan, “An analysis pattern for reservation and use of entities”, *Procs. of Pattern Languages of Programs Conf. (PLoP’99*), http://jerry.cs.uiuc.edu/~plop/plop99

[Fer99a] E.B.Fernandez, "Coordination of security levels for Internet architectures", *Procs. 10th Intl. Workshop on Database and Expert Systems Applications*, 1999, 837-841.

[Fer00] E.B.Fernandez, X.Yuan, and S.Brey, "An analysis pattern for the order and shipment of a product", Procs.of Pattern Languages of Programs Conf. (PLoP'2000), http://jerry.cs.uiuc.edu/~plop/plop2k

[Fer00a] E.B. Fernandez and X. Yuan, "Semantic Analysis patterns", *Procs. of 19th Int. Conf. on Conceptual Modeling, ER2000*, 183-195.

[Fer00b] E.B. Fernandez, "Stock Manager: An analysis pattern for inventories", *Procs. of PLoP 2000*, http://jerry.cs.uiuc.edu/~plop/plop2k

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997

[Hay96] D.Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996.

[ibm] IBM Corp., Web Sphere Commerce Suite, http://www-4.ibm.com/software/webservers/commerce

[Kob01] C. Kobryn, *Modeling components, patterns, and frameworks with UML*, Notes for tutorial, *Software Development 2001 West*, April 2001.

[Lya98] F. Lyardet, and G. Rossi, "Patterns for dynamic websites", *Procs. PloP'98*, http://jerry.cs.uiuc.edu/~plop/

[Lya99] F. Lyardet, G. Rossi, and D. Schwabe: "Patterns for adding search capabilities to web information systems", *Procedings of EuroPloP'99*, http://www.argo.be/europlop/index.html

[Pal] M. Palmer, "A Personalization design pattern for dynamic websites", http://objectdesign.com

[Ros00] G. Rossi , F. Lyardet, and D. Schwabe, "Patterns for e-commerce applications", *Procs. EuroPLoP'2000*, http://www.coldewey.com/europlop2000/

[Ros01] G. Rossi, D. Schwabe, J. Danculovic, and L. Miaton, "Patterns for personalized web applications", *Procs. EuroPLoP'2001*, http://www.hillside.net/patterns/EuroPLoP/

[Sch98] D. Schwabe, and G.Rossi: "An object-oriented approach to web-based application design", *Theory and Practice of Object Systems (TAPOS*), October 1998.