Analysis Patterns for the Order and Shipment of a Product

Eduardo B. Fernandez, Xiaohong Yuan, and Sandra Brey

Dept. of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, FL 33431

Abstract

These analysis patterns describe how a customer places an order for a product, and the subsequent shipment of the product. We describe first two elementary patterns, the Order and Shipment patterns. We then combine them to form what we have called a semantic analysis pattern because it emphasizes semantic aspects of the application model as opposed to improving flexibility. The purpose of this type of pattern is to serve as a starting point when translating requirements into an actual design. This pattern represents a minimum application so that it can be applied to a variety of situations and it can be combined with other related patterns to describe more complex applications. The order and shipment of a product is a very common real-life problem. The pattern focuses on the basic aspects of the order and its delivery, and the correspondence between an order and its fulfillment.

1. Introduction

We present analysis patterns that describe how a customer places an order for a product, and the subsequent shipment of the product. We describe first two elementary patterns, Order and Shipment. We then combine them to form a pattern of the category that we have called Semantic Analysis patterns [Fer00] because they emphasize semantic aspects of the application model as opposed to improving flexibility. The purpose of this type of pattern is to serve as a starting point when translating requirements into a conceptual model. This pattern represents a minimum application so that it can be applied to a variety of situations and it can be combined with other related patterns to describe more complex applications.

The order and shipment of a product is a very common real-life problem. It involves a customer placing an order for certain kind of product or service; for example, food, book, tapes, etc., and subsequently taking delivery of the product and paying for it. The Order pattern focuses on the basic aspects of the order, without detailing the specific type of product or customer. The Shipment pattern describes delivery of the ordered product. An important point of the combined pattern is tying together an order and its corresponding fulfillment. Details of the product such as manufacture and availability are left for the specific application or for complementary patterns.

2. Order pattern

The most basic pattern is the description of the order itself.

Context

Numerous practical situations require requesting a product or a service, e.g., ordering dinner in a restaurant, ordering books from an e-commerce company, etc.

Problem

How to describe a request for a product or service

Forces

- The request must be captured in a precise way
- The status of the request must always be known.
- The order must be correlated with its final result, a shipment or delivery.
- Customers want usually a type of product, not a specific individual product

Solution

The class model of Figure 1 shows the required information, including classes to describe the order, the customer who placed the order, and the product. The association between classes Customer and Order shows that each order is placed by a specific customer, but a customer may generate many orders. An order consists of a collection of line items, each of which describes a particular type of product and the quantity ordered. Figure 2 shows the states of an order along time. State changes in the order trigger other events; completion creates a shipment object, closing the order puts its information in a log. A sequential diagram describing these actions is given in Figure 8.

Consequences

- This pattern describes only the request for a product, but it can be related to its delivery.
- Order requests may be for products or services.
- The customer may be a person or another system; for example, a Shop Order is created from each line of a Customer Order for products that need to be manufactured.
- While the order must include complete information, only some aspects need to be specified for the customer and the product.
- The requested product is a product type, not an individual product. It can be easily extended for this case.
- Information about a customer and credit checking are performed only the first time a customer places an order.

Known uses and related patterns

See Sections 5 and 6

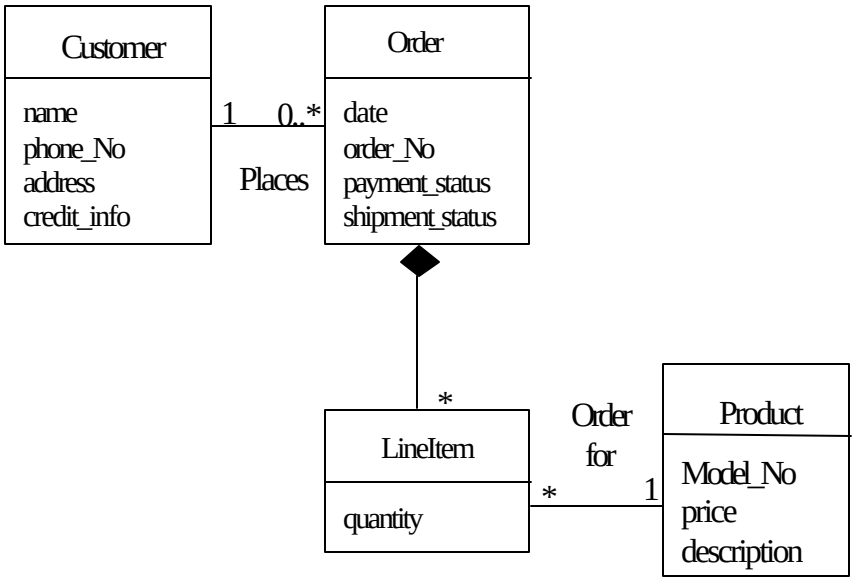

Figure 1. Class model for the Order pattern

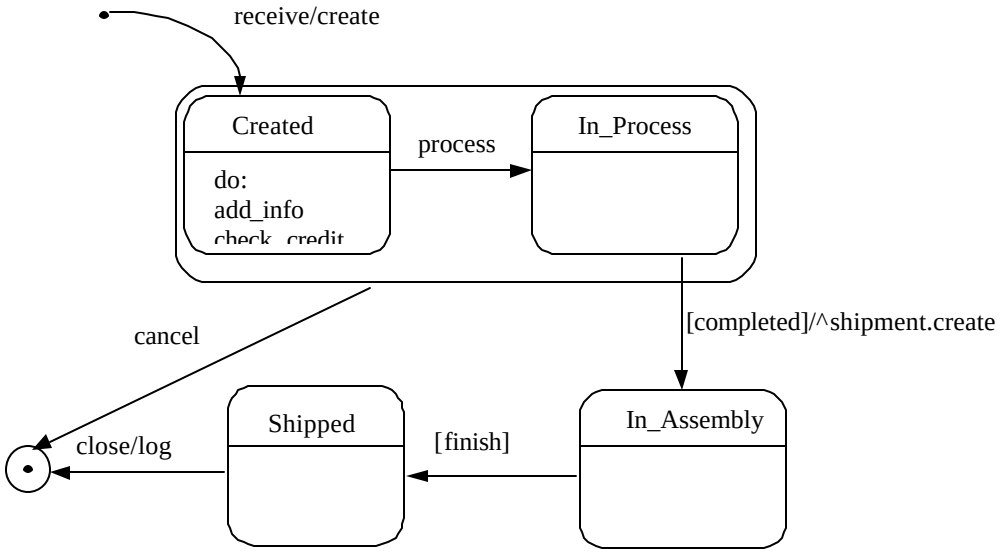

Figure 2. Statechart for class Order

3. Shipment pattern

Context
An order for some product has been fulfilled and the product must be delivered to the customer. The product to be delivered must correspond to the ordered product.

Problem
How to describe the shipment of an ordered product or delivery of a requested service.

Forces
- The shipment must correspond to some request expressed in some order.
- The shipment should describe the products shipped and the amount to be paid (an invoice document).
- There is a responsible customer who will receive the shipment.
- The ways in which the product may be delivered or paid are not relevant to the pattern.

Solution
A shipment is going to a customer and this should receive an invoice for a payment. This is shown in the class model of Figure 3, where classes Shipment, Customer, and Invoice describe these facts.

Statecharts for classes Shipment and Invoice describe the states of these classes along time (Figures 4 and 5). When the shipment is completed, an invoice is created, and closing the shipment or the invoice enters them in a log. Figure 7 shows a sequential diagram for these actions.

Consequences
- Each shipment can be related to its corresponding order.
- Customers may be a person, an institution, or another system
- Delivery and payment details are not included.
- The receiver may or may not be the same customer who placed the order.
- The Invoice class describes the amount to be paid and corresponds to a real document.
- The model also applies to services, with class Shipment representing the delivery of the requested service.

Known uses and related patterns
See Sections 5 and 6.

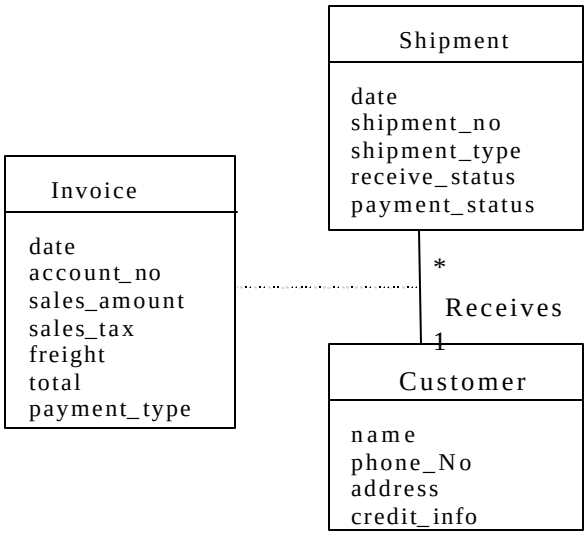

Figure 3. Class model for Shipment pattern

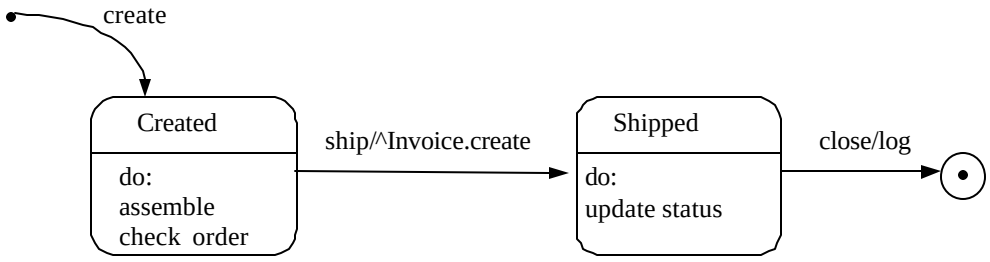

Figure 4. Statechart for class Shipment

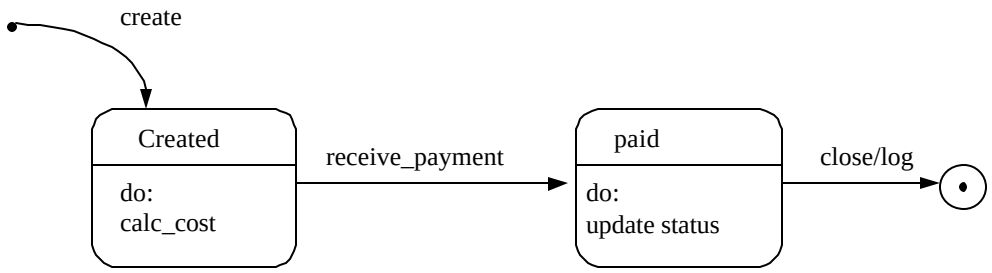

Figure 5. Statechart for class Invoice

4. Order/Shipment pattern

Intent

This pattern describes the placement of an order for some product or service and its corresponding fulfillment.

Context

A specific example is shown in Figure 6, where a Customer orders a shipment of pagers from a pager manufacturer. The classes contained in the model include Customer, Order, LineItem, Shipment, Invoice, and Pager, with their obvious meanings.

The association between Shipment and Order shows that each shipment has a corresponding order, but an order does not necessarily result in a shipment (e.g. the order could be cancelled). The objects of the association class between classes Customer and Shipment describe the invoices created for each shipment.

This example is a particular case of a more general problem of order and shipment, which appears in a variety of contexts, e.g., when ordering some product, when ordering a dinner in a restaurant, when ordering a repair job to be done somewhere.

Forces

- The institution needs to optimize cycle time of order fulfilling (in a qualitative sense).
- The institution needs to track order fulfillment to maintain customer satisfaction.
- The model must include representations of real-life documents, e.g., Orders, Line Items, and Invoices.
- Equivalent products may be substituted for requested products.

- The analysis model must be a faithful representation of the requirements without including implementation details. Notice that these requirements may appear in different domains.

- The pattern must describe a fundamental semantic unit. This means the pattern must be simple enough to apply to a variety of situations.

Solution

a) Requirements

The solution corresponds to the realization of the following generic Use Cases:

- Receive an order. The customer's information (name, address, etc.) is recorded and credit is checked. An Order is created.
- Cancel an order. An existing order is cancelled. Some institution policies may apply.
- Deliver an order. The shipment or service must be checked against the order. An invoice is created. The product and its corresponding invoice are delivered to the customer.

These Use Cases are generic in that they correspond to a variety of situations. In some cases some of these steps may be implicit.

b) Class Model

Figure 7 is a class diagram for the realization of these Use Cases. This diagram is an abstraction and extension of the diagram of Figure 6, where Pager has been replaced by Product. The "process" operation in class Order summarizes all the steps necessary to produce the Product, while operation "assemble" in class Shipment summarizes the actions needed to collect and put together the different portions of the order. The diagram also shows that not all products ordered may be in the final shipment or that some of these products may be different from those ordered.

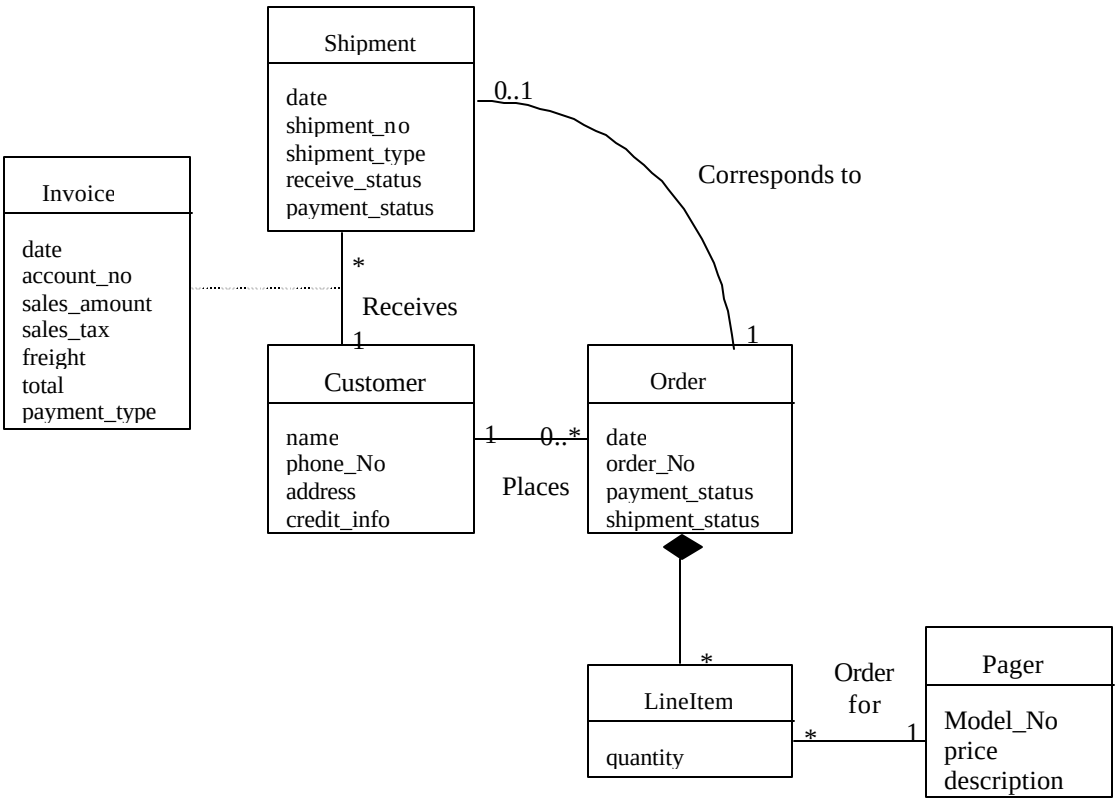

Figure 6. Order and shipment of pagers

c) Dynamic aspects

We have shown earlier statecharts for classes Order, Shipment, and Invoice. Figure 8 shows a sequence diagram showing how a customer places an order for a product and the subsequent shipment of the product. Figure 9 shows an activity diagram for the process of order and shipment. Notice that the state diagrams just show 'close' as event to close an order, without going into details of what produces the closing, while the sequence and activity diagrams assume payment as event to close the order. In Figure 8, the order of receive_shipment and receive_payment may be reversed.

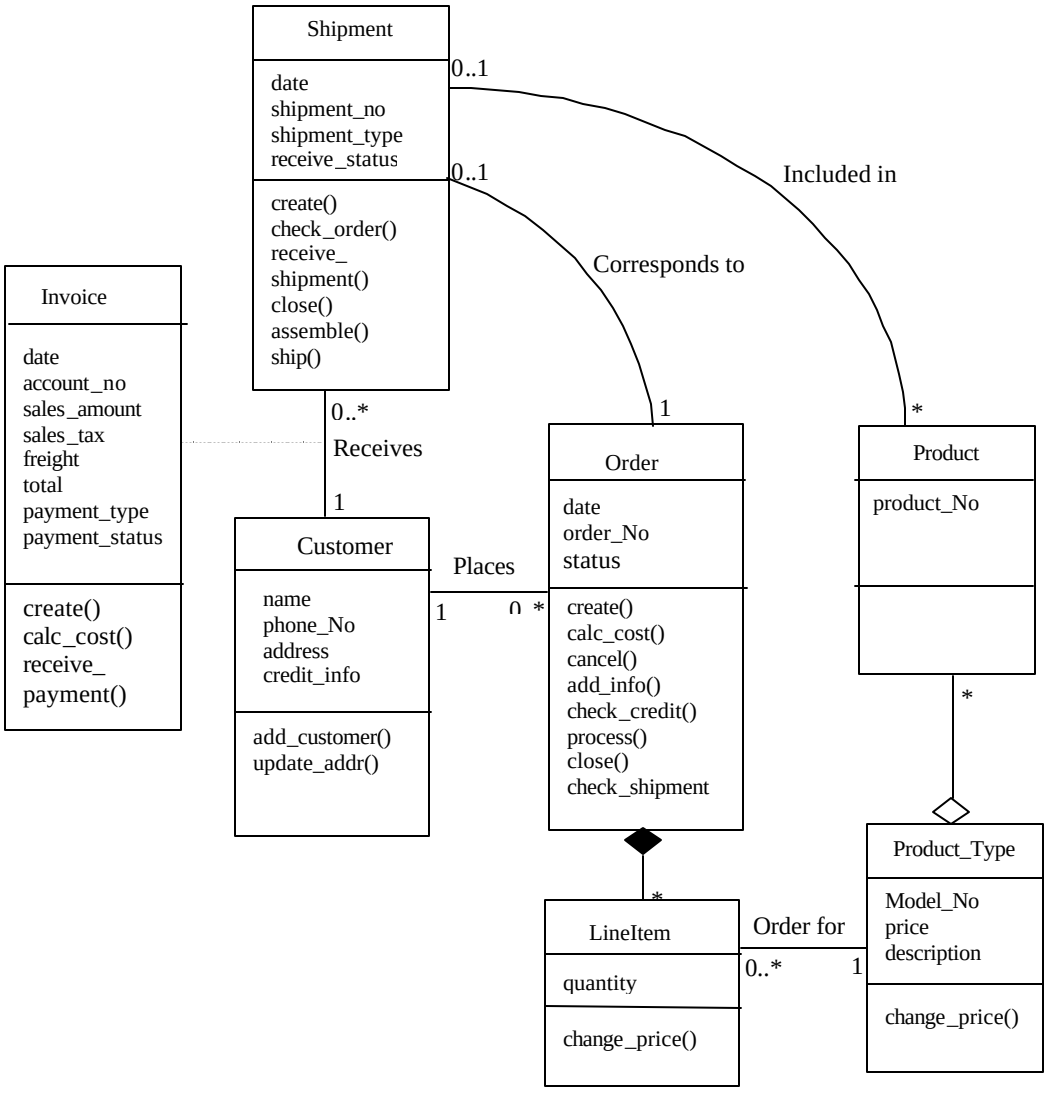

Figure 7. Class diagram for order and shipment of a product

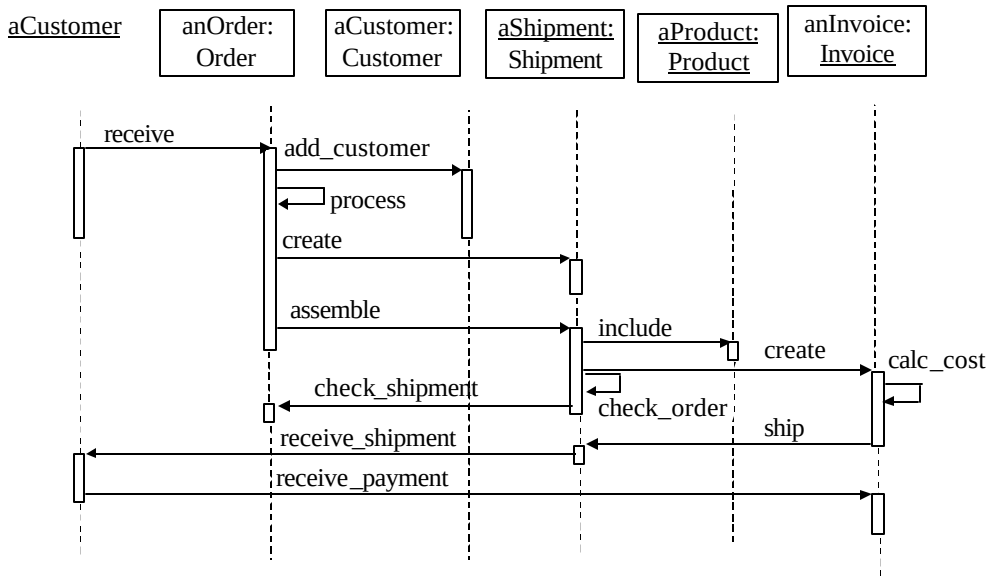

Figure 8. Sequence diagram for ordering and receiving a product

d) Consequences

The following elements are common to all implementations of this model:

- **An "order" is always generated in the "system" based on the customer's expressed need.** In a modern manufacturing facility or catalog store, the order would almost certainly take the form of an entry in a database. Likewise, at a sophisticated fast food restaurant such as Taco Bell, the order would exist as a virtual entity in the computer system, having been entered by special keystrokes at the point-of sale register. For an online vendor such as *Amazon.com*, the order (undoubtedly a virtual one) would be created automatically from the online request of the customer. At many sit-down restaurants, the order would consist of the "check", handwritten by the server based on requests from the customer.
- **An order is always linked to a customer**. That is, the customer responsible for the order must be identifiable. This may be as simple as the "rolling" system at the fast food restaurant identifying the customer by a serialized number on his printed receipt, the table number on a restaurant check, or in the form of a more sophisticated system where each order is linked via a pointer (e.g. "customer number") to a customer database containing information such as name, address, telephone, account status, discount eligibility, purchase history, etc.

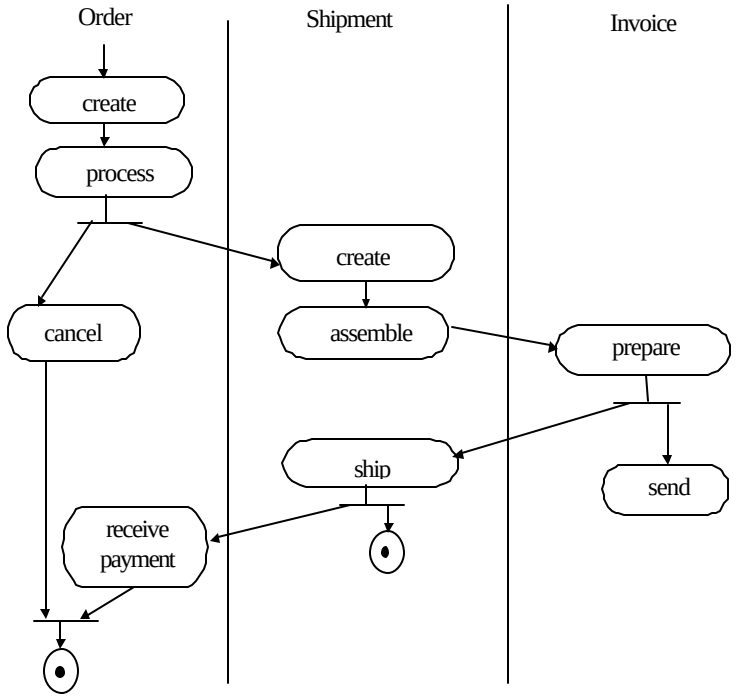

Figure 9 Activity diagram for order and shipment

- **Upon entry of an order, it is assumed that one of the following three possibilities may occur:**
  a). The product is available in stock and the order is dispatched.
  b). The product is not available, and the vendor must either order or manufacture the product.
  c). The order is cancelled before shipment.
- **Some type of documentation is always generated, with a copy archived and a copy delivered to the customer along with the product.** This will take the form of a cash register receipt, or invoice, or both, usually depending upon the sequencing of payment and delivery. It will contain important information such as the date, the product(s) delivered, and the price paid (or to be paid).
- **The ordered product could be an individual unit, not a type.**
- **The ordering or receiving customers could be another subsystem or system.**

The use of this pattern provides a systematic way of recording orders and their fulfillment and it would contribute to optimize the cycle time of order satisfaction. The pattern also applies to services; in this case class Shipment represents the delivery of the service.

The class diagram applies to an order for a generic product. It can be adapted to describe ordering specific individual products by changing the association between LineItem and ProductType to Product.

Not all the situations described by this pattern are exactly alike:

- The process of obtaining and verifying customer data may vary from lengthy in some cases, to non-existent in others.
- The customer may take delivery of the order directly from the vendor, i.e., the shipping part of the model is skipped.
- Payment may be required either before or after delivery, depending on the customer status and credit worthiness.

For generality, a good number of aspects are not represented in this pattern:

- Description of contextual and environmental aspects of the product
- Exceptions, e.g., bad credit
- How to keep track of the availability of products
- How to queue up requests when the required products are not available
- How to deal with varieties of customers, e.g., individual, corporate, preferred
- Order modification
- Returning of products
- Billing and payment policies.
- Physical details of shipping, e.g., packaging
- History

All these aspects should be completed with additional patterns or ad hoc models. One can expand these patterns into a pattern language for orders.

5. Known Uses

The following are examples of uses:

- A retailer/service provider of some type of devices, e.g., pagers, orders a quantity of devices to be resold at retail.
- A customer orders food from a restaurant.
- A customer orders a product from an e-commerce company, e.g., Amazon.com.
- A customer orders a new roof for her home.

Many variations of this model have appeared in the literature:

- Hay [Hay96] presents Order and Shipment patterns but he doesn't relate the order to its shipment and does not consider dynamic aspects, attributes, or operations.
- Fowler [Fow00] uses an order as a running example in his UML book. Our activity diagram is based on his.
- Berkem [Ber99] shows more detailed activity diagrams and relates them to Use Cases.
- Ambler [Amb97] shows a class model for orders, he does not consider shipments.
- Schneider and Winters [Sch98] enumerate a variety of Use Cases that apply to orders.
- Richter [Ric99] considers orders in the context of stock trading.
- K. Brown [Bro96], applies design patterns to an order management system. He does not discuss analysis aspects.

- The Open Applications Group has defined interfaces for orders and shipments for interoperability between different systems [Ope00]

6. Related Patterns

The Reservation and Use pattern [Fer99] complements this pattern, providing the possibility of reserving a product being manufactured continuously. Fulfilling of orders requires components and affects the contents of inventories, the Stock Manager pattern [Fer00a] is also complementary. This combination could be part of a manufacturing framework. As indicated above, the Type object pattern appears here as a subpattern. The customer may participate in this process in different ways and the Role object pattern [Bau00] could be used to indicate this. A detailed treatment of money aspects can be found in [Hay96] and [Fow97]. The Dependent Demand pattern [Hau97], discusses how orders may trigger other orders in manufacturing systems. The Order/Shipment pattern is a special case of a Composite pattern [Rie96], and could be studied as such.

Acknowledgements

We thank our shepherd Dirk Riehle and the Writers' Workshop at PLoP 2000 for insightful and detailed comments that have significantly improved the quality of this paper.

References

[Amb97] S. Ambler, "Taking a layered approach", *Software Development*, July 1997, 68-70.
[Bau00] D. Baumer, D. Riehle, W. Siberski, and M. Wulf, "Role Object", Chapter 2 in *Pattern Languages of Program Design 4*, Addison-Wesley 2000.
http://st-www.cs.uiuc.edu/~plop/plop97/Workshops.html
[Ber99] B. Birkem, "Traceability management from business processes to Use Cases with UML", *JOOP*, September 1999, 29-34 and 64.
[Bro96] K. Brown, "Experiencing patterns at the design level". *Object Magazine*, January 1996, 40-48.
[Fer99] E. B. Fernandez and X. Yuan. "An analysis pattern for reservation and use of reusable entities", *Pattern Languages of Programs Conference*, *PloP99*.
http://st-www.cs.uiuc.edu/~plop/plop99
[Fer00] E.B. Fernandez and X. Yuan, "Semantic Analysis patterns", *Procs. of 19th Int. Conf. on Conceptual Modeling, ER2000*, 183-195.
[Fer00a] E.B. Fernandez, "Stock Manager: An analysis pattern for inventories", *Procs. of PLoP 2000*.
[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.
[Fow00] M. Fowler, *UML Distilled* (2nd Edition), Addison-Wesley, 2000.
[Hau97] R. Haugen, "Dependent Demand—A business pattern for balancing supply and demand", *Procs. of Pattern Languages of Programs Conf*., PloP97,

http://st-www.cs.uiuc.edu/~plop/plop97/Workshops.html
[Hay96] D.Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996.
[John98] R. Johnson and B. Woolf, "Type Object", Chapter 4 in *Pattern Languages of Program Design 3*, Addison-Wesley, 1998.
[Ope00] Open Applications Group, *Integration Scenarios,* http://www.opeanapplications.org/oagis
[Ric99] C. Richter, *Designing flexible object-oriented systems with UML*, Macmillan Tec. Publ., 1999.
[Rie96] D. Riehle, "Composite design patterns", *Procs. of OOPSLA'97*, 218-228.
[Sch98] G. Schneider and J.P. Winters, *Applying Use Cases—A practical guide*, Addison-Wesley, 1998.