Stock Manager: An Analysis Pattern for Inventories

Eduardo B. Fernandez
Dept. of Computer Science and Engineering,
Florida Atlantic University, Boca Raton, FL 33431
ed@cse.fau.edu

Abstract

Inventories keep track of what an institution has, such as parts, finished goods, furniture, machinery, etc. A good inventory system is a necessity for any modern business or manufacturing system. We present an analysis pattern for an inventory system that keeps track of quantity and location of items in stock, and updates these quantities according to the different stages of manufacturing or production, from component ordering to product shipping. This is a generic model defined from the abstraction of a real workable inventory and can be extended to fulfill more detailed requirements or similar applications. This pattern is a composite pattern and we identify two atomic patterns as components.

1. Introduction

In modern manufacturing systems, the management of the information involved in the manufacturing process has become a key factor to reduce production costs and improve product quality. Many companies and research institutions have invested large amounts of resources in this field, and Manufacturing Resource Planning Systems (MRP) have become important [Salv92]. The inventory is one of the most important parts of an MRP system, and keeps track of information about quantities of units of interest and their locations.

We develop here an inventory pattern, Stock Manager, which satisfies the expected requirements of reusability and extensibility. This model considers not only the static view of the system, but also dynamic aspects. Because an inventory control system cannot be modeled completely unless several other aspects of the whole manufacturing system are taken into account, the effects of some other functional parts of manufacturing systems are also included. This model can be used as starting point to develop a complete model for the manufacturing system or as an example of a Semantic Analysis Pattern (SAP) [Fer00a], a minimal application that can be used in a range of domains. This pattern is also a composite pattern [Rie96] and we identify two atomic patterns as components: Basic Inventory and Item Distribution.

2. Problem

How can businesses, manufacturing shops, libraries, etc., keep track of their stocks of items of different types and their locations?

3. Context

All institutions, large or small, use components to build products or supplies to perform their work. They need to be able to track the quantities of the items that they have. In a manufacturing plant these items are components and products; in a library they are books, tapes, or records; in a clothing store they are clothes and accessories, etc. These institutions also need to be able to find these items when needed.

4. Forces

- In a company, items can be materials used for manufacturing or finished products. There is the possibility of losses or destruction of this stock. The institution must be able to keep track of the actual number of items in stock.
- Other functional units may change the stock quantities; i.e., any transfer or use of items anywhere should update the corresponding inventory quantities.
- The solution must describe a fundamental semantic unit. This means the solution must be simple enough to apply to a variety of situations. This is the basis for reusability.
- The solution must include representations of real-life documents.

5. Solution.

5.1. Requirements

The basic functions or Use Cases of an inventory system can be summarized as:

- Separate different types of stock, e.g., materials or components versus finished products.

- Keep track of the quantities of each item in stock. Several kinds of quantities may be needed. The *onHand* quantity indicates the existing quantity of an item. The *onOrder* quantity indicates how many items will be delivered to the stock in the future. The *reserved* quantity indicates the amounts reserved for orders and it implies that the quantity that an employee can use later is the difference of the *onHand* and the *reserved* quantities (the *available* quantity). Some other quantities may be used in more elaborate systems.

- Keep track of the locations of items. The inventory should record the distribution of items in specific locations.

.
Because of the nature of these functions, almost all the other subsystems have an effect on the quantities held in the inventory. Some of the functions that have a direct effect and update data on the inventory system are: purchasing, receiving, distribution of materials, auditing, scrapping, shipping, and manufacturing.

5.2 Atomic patterns

We start by defining an inventory pattern for keeping track of items, the Basic Inventory pattern. A basic model of an inventory system is shown in Figure 1. The items are anything of whose existence and quantity we want to be aware. Each item belongs to a unique type, which provides an identifier such as item number (item code, model). The Inventory class keeps quantities of interest for each item. This can be considered an atomic pattern that can be used on its own or as part of a larger model, e.g., a SAP. Its dynamic aspects are shown in the composite pattern.

Figure 2 shows a class model for the Item Distribution pattern. This is a pattern that describes the locations of a set of items as well as their distributions in those locations. It is another atomic pattern and will be combined with the Basic Inventory pattern to form the Stock Manager pattern. To indicate distribution we need to break down the inventory quantity into location quantities (usually there exist several locations where an inventory item can be stored). This requires to define a many-to-many association between the Inventory and Location classes (one type of item can be distributed into several locations and one specific location can store several types of item), and use an association class Distribution to indicate the distribution of items in locations.

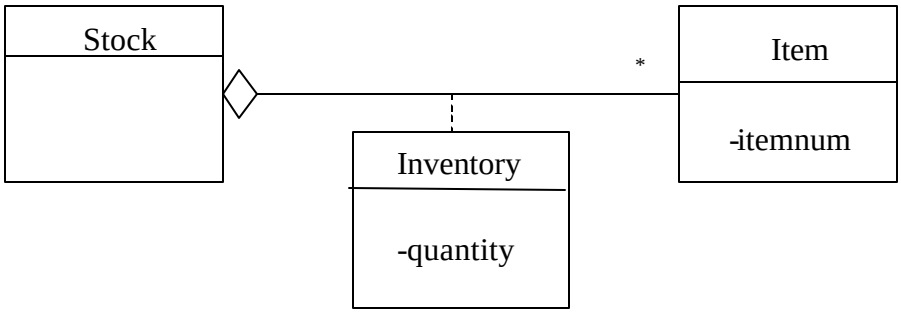

Figure 1. Class diagram for a basic inventory

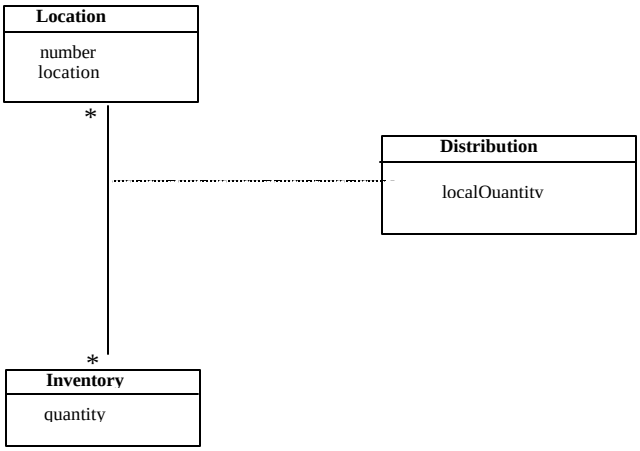

Figure 2. Class diagram for the Item Distribution pattern.

5.3. Class model for Stock Manager

A model for an inventory that satisfies the requirements of Section 5.1 is shown in Figure 3. Classes **Stock** and **Component/Product** are related by a composition association [1]. The quantities, described by the Inventory class, are a joint property of **Stock** and **Component/Product** that has different values for different links. This model permits a designer to define different types of stock as separate collections; e.g., stock of components, stock of products. Different types of inventories can be generalized into a class **Inventory**.

Items are classified into two varieties: finished products and components (used in product manufacturing), but other varieties are possible. The **Product** class has attribute 'model'

[1] This is a loose type of aggregation, where both classes can have independent existence [Booc99].

as a unique identifier and other attributes that describe what a customer may select, such as *color*, etc.. The **Component** class has *itemnum* as unique identifier and other attributes such as *description*, *type*, etc. **Product** and **Component** usually are in a many-to-many aggregation relationship (one model of a product uses several component types and one type of component can be used to manufacture several models of a product), but that is not relevant for the inventory model and not shown in this figure.

The reason the Item class is decomposed into subclasses **Product** and **Component** is that there exists many differences in the management of these two entities; for example, a product is made of several components and the inventory needs to keep track of the dynamic changes of component quantities in the manufacturing process. In other words, the component inventory is more complex than the product inventory. On the other hand, there exist similar aspects in both inventories, e.g. both keep track of *onHand* quantity (the total amounts) of corresponding products or components and their physical locations. Generalization can be applied here defining **Inventory** as a superclass to hold similarities, and **ComponentInventory** and **ProductInventory** as subclasses to preserve unique aspects of each variety of inventory.

The model of Figure 3 includes operations derived from dynamic analysis, as shown in the next section.

5.4 Dynamic Analysis

The class model shown provides a static view of the inventory. This is not enough, we also need a dynamic model that shows how the inventory changes along time. Here, we model the general features that an inventory should have in order to perform common operations. Other features for specific inventories can be derived from this basic configuration.

When components or products are moved to stock, their *onHand* values are increased by operation *add_to_stock*. When components or products are moved out of the stock, their *onHand* values are reduced by applying operation *remove_from_stock*. Operation *add_to_stock* determines the distribution of items into local stockrooms based on some predetermined criteria. When components and products are moved to specific stockrooms operation *add_to_localStock* increases the *localonHand* quantity. Similarly *remove_from_localStock* reduces the *localonHand* quantity. The corresponding sequence diagram is given in Figure 4.

Components or products can be transferred from one stockroom to another and operation *transfer* performs this action. Due to the fact that all the movement operations can apply to both components and products, we put them in the superclass **Inventory.** When components or products are moved out of one stockroom, operation *remove_from_localstock* is used to update the *localonHand* quantities. In the opposite case, *add_to_localstock* is applied. A sequence diagram for transfer of items is given in Figure 5.

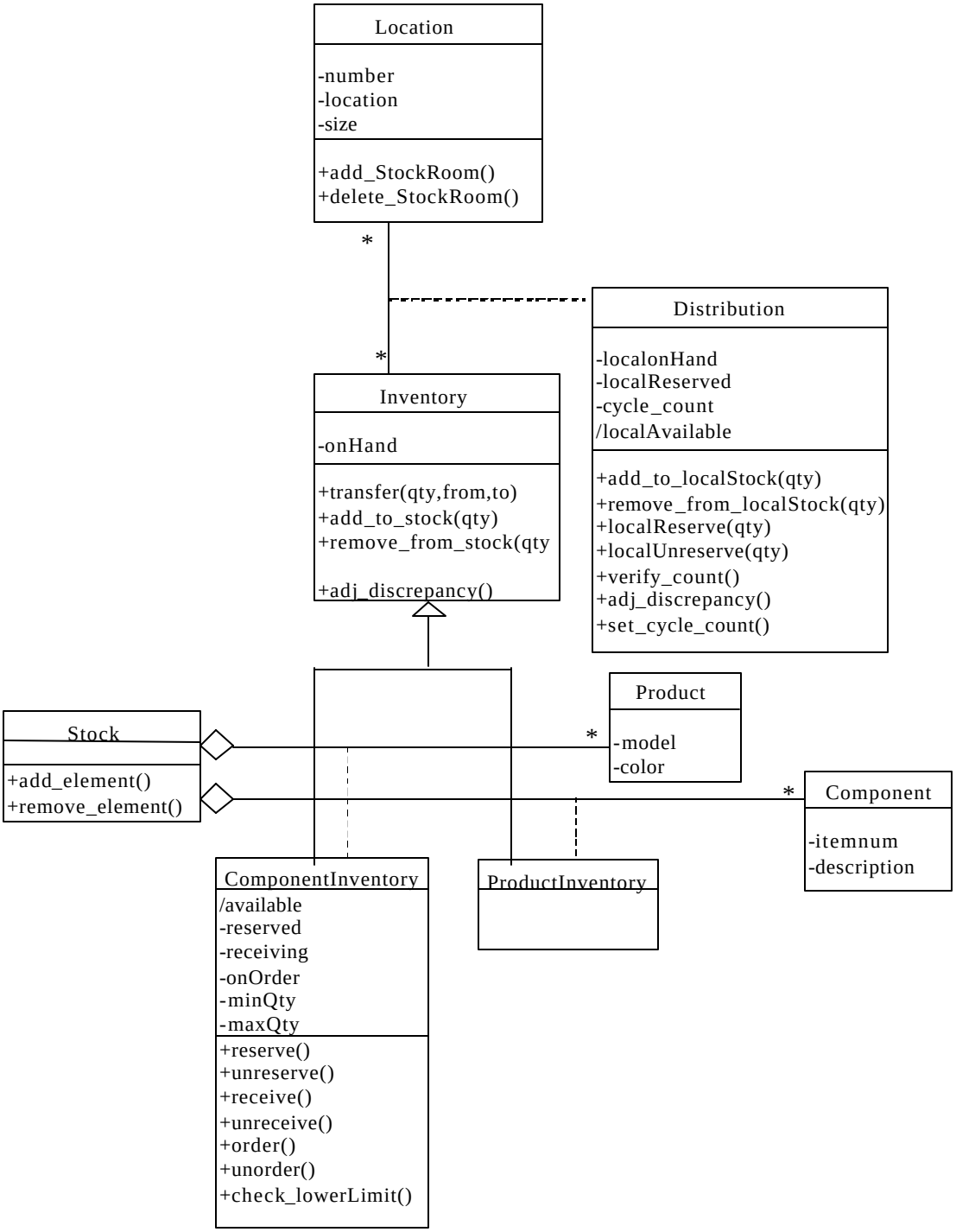

Figure 3. Class diagram for the Stock Manager analysis pattern.

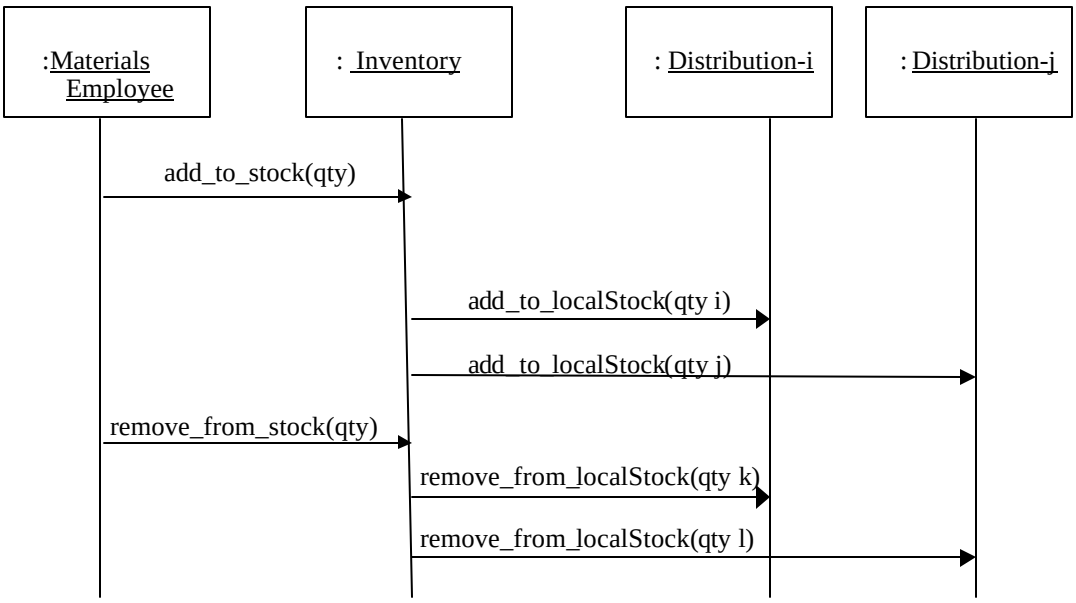

Figure 4. Sequence diagram for moving items in or out of stock

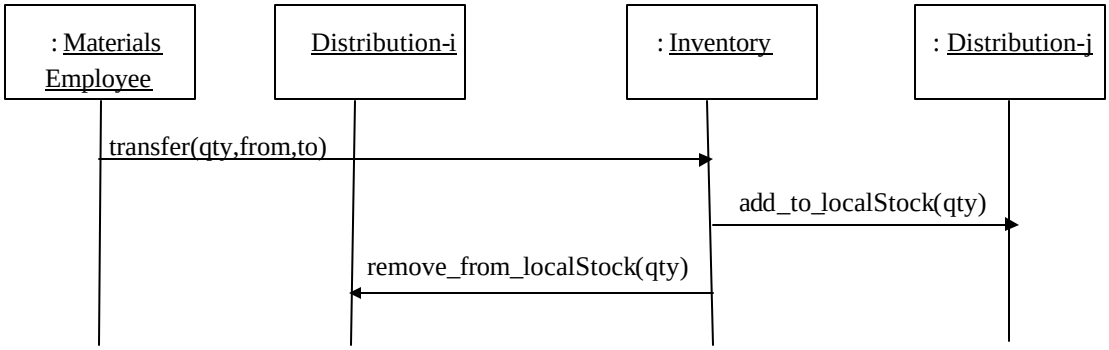

Figure 5. Sequence diagram for item transfer

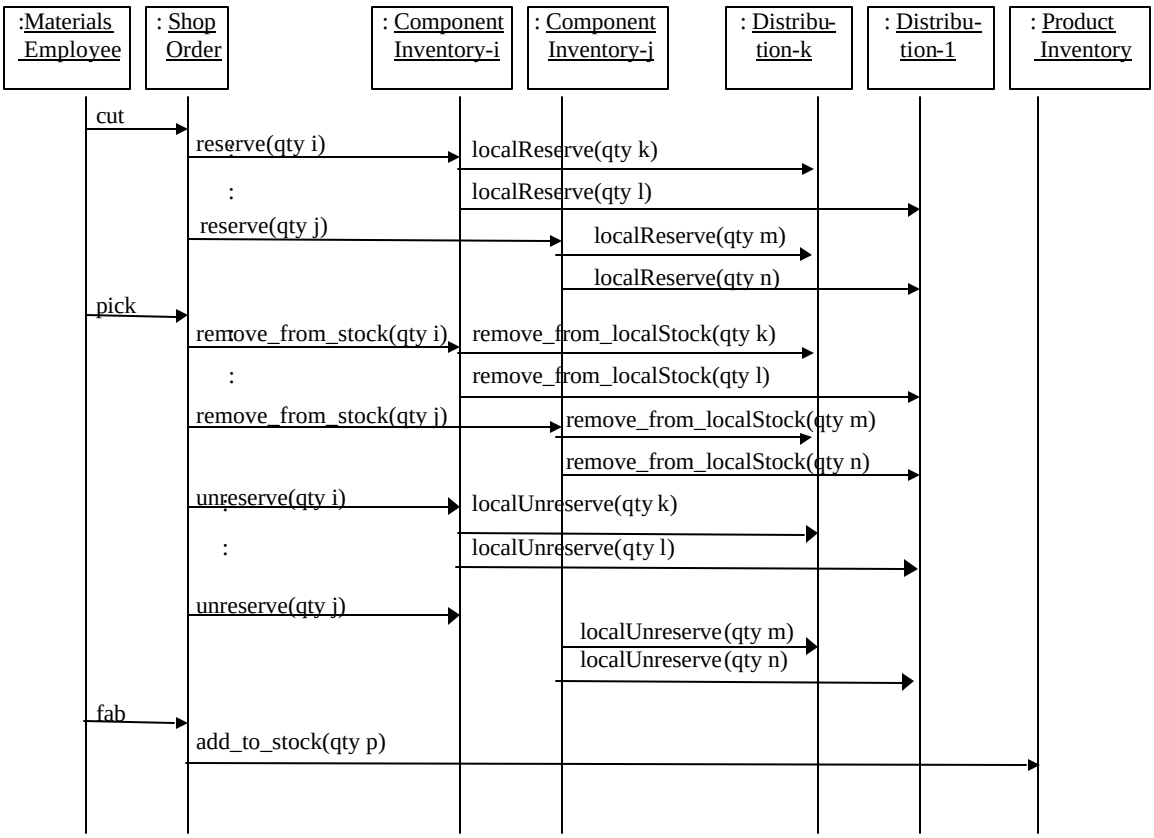

Figure 6. Sequence diagram for manufacturing a Shop Order

The most complex changes to the inventory happen during manufacturing (Figure 6). We first assume that Customer Orders have already been processed into a form that indicates in detail what and how many components are needed in the manufacturing for a type of product; this is a Shop Order. When a Materials Employee cuts the Shop Order, the values of *reserved* components are increased based on the quantity indicated by this Shop Order. When components have been physically picked from the stockroom, the values of *reserved* and *onHand* for these components are reduced. When fabrication is finished, **ProductInventory** quantities are updated by increasing their *onHand* value. Usually, a Shop Order takes several days from cut to finish. The stages cut, pick and fab let people know what is the status of the Shop Order.

From all these sequence diagrams we can find the operations needed for the classes of Figure 3.

6. Application of the Pattern to a Library Inventory

To show the use of the pattern in a different domain we apply it to a library inventory (Figure 7). Class **WorksStock** now defines the stocks of library materials. Class **Distribution** indicates quantities in specific locations. **Location** here can indicate the circulation desk, storage shelf, microfilm center, new publications reading area, etc. We show here a stock of **Works**; however, this would probably be separated into stocks of books, videotapes, CDs, etc., because these categories are handled differently. The Use Cases for this system are activities such as borrow/return a book. Some of the sequence diagrams for these Use Cases are similar to the ones shown, e.g., transfer of books between locations; however, some don't have a counterpart, e.g , shop order processing.

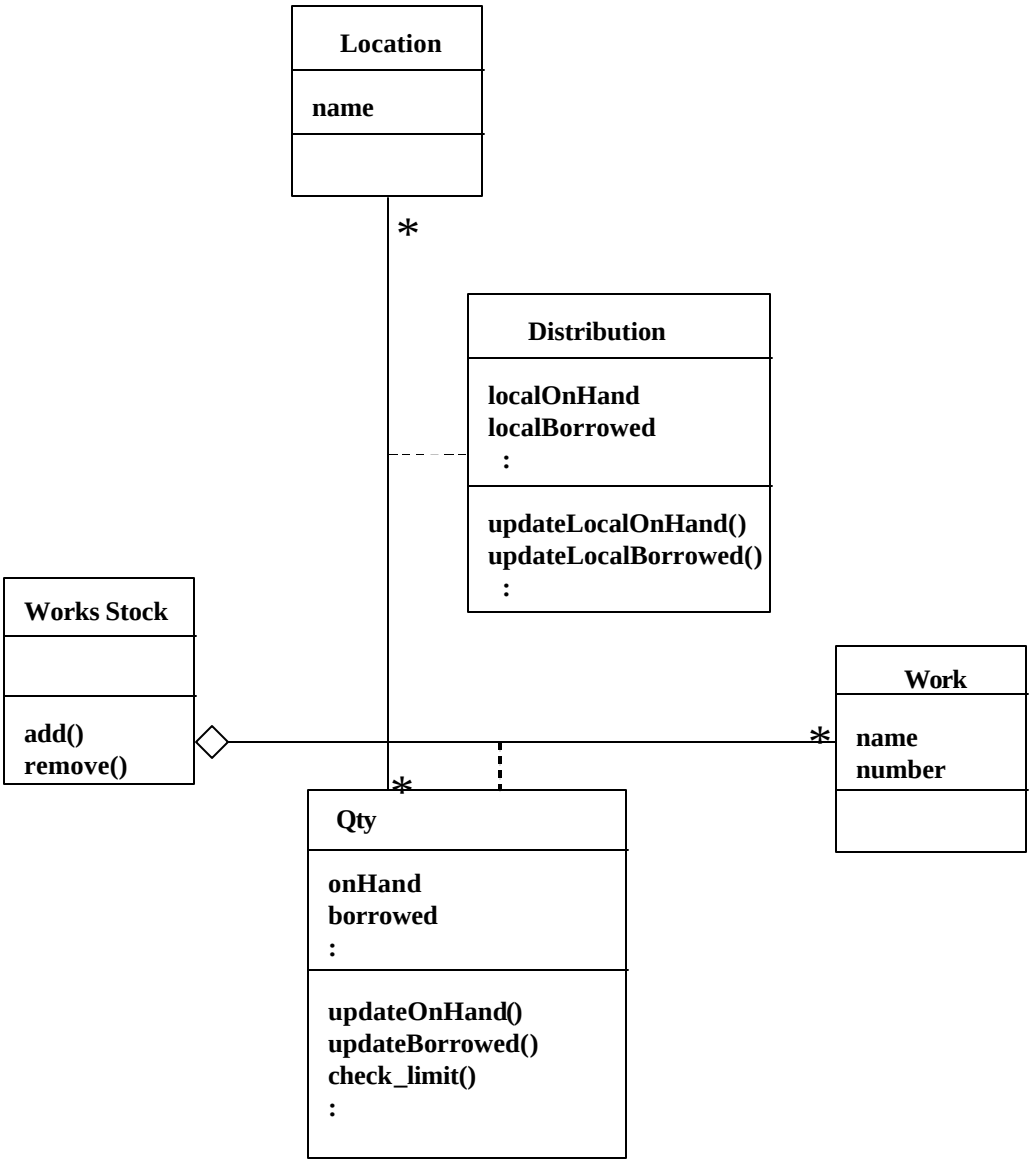

Figure 7. Library inventory

7. Consequences

The models generated satisfy the forces:

- The pattern can be used to keep track of quantities of interest for different types of stock and their distributions.
- The effect of other activities can be reflected through appropriate operations. As indicated for the library example, the activities that affect the inventory may be different in different applications.
- Although the pattern is described in manufacturing terms, it can be applied to represent the inventory of a library, a business, or similar places.
- Documents such as Shop Orders and others are considered part of the external systems that interact with the inventory and are not represented in the pattern.

In order to make the pattern applicable to several domains we have left out:

- Details of the items, such as structure and description.
- Details of the storage locations
- Documents for internal actions, e.g. for transfer of stock within locations.
- Exceptions, e.g., a reservation for an amount larger than the amount available, order cancellation..
- Alarms to indicate low inventory levels.
- Historical information.
- Money movement.
- Handling of inventory items such as liquids, measured in different ways.

These aspects should be added through other patterns or by developing a complete framework that includes them.

8. Known uses

Some examples of inventory models are:

- The model presented here was applied in the development of an inventory prototype for the Information Technology Dept. of the Pager Products Division of Motorola in Boynton Beach, FL [Fern97].

- In Hay's inventory model [Hay96], an ASSET is a physically-existing thing, that must be(currently) at a particular SITE, such as a warehouse location or a manufacturing work center. That is, an ASSET is the physical occurrence of an ASSET TYPE at a SITE. Two kinds of ASSET are considered: DISCRETE ITEM and INVENTORY. A DISCRETE ITEM is an asset that can be identified by individual pieces(that is, by serial member or company tag number). The inventory only keeps track of assets in bulk quantities. ASSET TYPEs are broken down into PRODUCT TYPE, MATERIAL TYPE, and PART/EQUIPMENT TYPE, because physical products, parts and materials are kept track of in different ways in the author's view. In summary, Hay includes some aspects which we left out. However,

his model doesn't include dynamic aspects, attributes, or operations; it doesn't separate either stock from inventory.

- Fowler dedicates a chapter (Chapter 6 in [Fowl97]), to Inventory and Accounting. He emphasizes the movement of money and goods through accounts. Another important aspect in his model is the separation of metadata (knowledge level) from operational data.

- Hellenack [Hell97] discusses inventories as part of other business patterns. She separates Finished Goods Inventory from Raw Material Inventory (equivalent to our **Product** and **Component** classes), and also includes some money movement. That model includes only static aspects and no distribution

9. Related Patterns

The Reservation and Use of Entities pattern [Fern99], and the Order and Shipment pattern [Fern00b], complement this pattern for several applications, e.g., manufacturing. The patterns in [Brag98, Brag99] apply to resource control and will appear together with this pattern in many applications. A pattern that may use the Stock Manager pattern is the Dependent Demand pattern [Hau97]. Because the products and components may be types, the Type Object Pattern [John98], is also relevant. Finally, the Stock/Component and Stock/Product aggregations are examples of the Container/Context pattern [Coa97]..

Acknowledgements

The Motorola prototype was built also by Michael Lynch, Chin-Shu Lee, Zhiwei Peng, and Mahbub Anwar, they all contributed to these ideas. Rosana T.V. Braga, our shepherd, significantly improved the quality of this paper with her insightful comments. The Writers' Workshop at PLoP 2000 provided valuable suggestions to improve this paper

References

[Booc99] G.Booch, J.Rumbaugh, and I.Jacobson, " *The Unified Modeling Language User Guide*", Addison-Wesley 1999.

[Brag98] R.T.V. Braga, F.S.R.Germano, and P.C. Masiero, "A confederation of patterns for resource management", *Procs. of PLoP'98*, http://jerry.cs.uiuc.edu/~plop/plop98

[Brag99] R.T.V. Braga, F.S.R.Germano, and P.C. Masiero, "A pattern language for business resource management", *Procs. of PloP'99*, http://st-www.cs.uiuc.edu/~plop/plop99

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

[Fern97] E.B. Fernandez , M. Lynch, and C.S. Lee, *"The development of an object-oriented prototype for an inventory control system"*, Report TR-CSE-97-32, Dept. of Computer Science and Engineering, Florida Atlantic University, April 1997.

[Fern99] 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://st-www.cs.uiuc.edu/~plop/plop99

[Fern00a] E.B. Fernandez and X. Yuan, "Semantic Analysis Patterns", *Procs. of the 19th Int. Conf. on Conceptual Modeling (ER2000),* 183-195.

[Fern00b] E. B. Fernandez, X. Yuan, and S. Brey, *"An Analysis Pattern for Order and Shipment of a Product"*, *Procs. of PLoP 2000*.

[Fowl97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

[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.C.Hay, *Data model patterns -- Conventions of thought*, Dorset House Publishing, New York, 1996.

[Hell97] L. J. Hellenack, "Object-oriented business patterns", *Object Magazine*, January 1997, pp. 23-30 and 70.

[John98] R. Johnson and B. Woolf, "Type Object", Chapter 4 in *Pattern Languages of Program Design 3,* Addison-Wesley, 1998.

[Rie96] D. Riehle, "Composite design patterns", *Procs. of OOPSLA'97*, 218-228.

[Salv92] G. Salvendy, *Handbook of Industrial Engineering*, Wiley-Interscience Pubs. , 1992.