

Mohamed E. Fayad

How to Deal with Software Stability

All that exist are heuristics for determining an EBT, a BO, or an industrial object.

In my previous column I introduced the concepts of enduring business themes (EBTs) and business objects (BOs) as essential artifacts used to achieve software stability [2, 3]. This third case study compares and contrasts a classical model built by an expert in the field of software engineering, Peter Coad, and a model of the same problem based on Enduring Business Themes and Business Objects.

The project is to create a warehouse tracking system allowing goods storage and order filling to be performed more quickly and more efficiently. The warehouse is physically organized into aisles of bins. Each bin contains items, and each pair of bins and items has a line item associated with them for clerical purposes. Items arrive at the warehouse on pallets. Since pallets do not have to contain the same item, they also have line items for clerical purposes. When a customer places an order, that order and the line items on it are translated into a pick list and list line items. The items indicated by the pick list are finally retrieved from their appropriate locations in the warehouse and moved to the

loading dock for shipment. The tracking application must also keep track of customer information and customer organizations to better serve repeat customers.

Coad's model of this problem [1] has several aggregations and finely intertwined relationships between classes. Many of these classes could be classified as Industrial Objects (IOs). Consequently, minor changes to this system could result in major changes to this model and would require a reengineering process to fix. Finding EBTs for a warehouse tracking system is no trivial task. However, once it is done and if it is done properly, the resulting model will be far more stable and robust to change.

Warehouses are used, of course, for storing goods. Thus, one of the obvious EBTs that should be incorporated into the model should be goods storage. People also want their goods stored and retrieved in a timely manner. Therefore, the warehouse must be run and the storage systems must be structured efficiently. It is easy to see that efficiency is another EBT worth consideration. Other EBTs that immediately come to

mind are shipping, receiving, order handling, customer service, and movement of goods.

There are several BOs involved in a warehouse. Several of them exist in [1]. First and foremost, the warehouse itself is a BO. The goods to be stored are also BOs. It does not matter to the system what goods are being stored. To the system, goods are goods. In [1], the "Item" class represents goods, so, for consistency's sake, they will remain members of the "Item" class in the EBT-based model. Customer, organization, and order, from [1], are also BOs. The rest of the objects in his model, however, are IOs. They can all be changed completely without changing the basic premise of a warehouse.

Although this may not appear obvious, the EBT-based model is far more adaptable. Most of the associations between classes have been moved to associations between EBTs and between BOs, which will never change in such a way as to affect the system. All the changes will occur in the IOs, which have been moved toward the periphery of the system.

Thinking Objectively

Identification Heuristics

Some heuristics for finding enduring themes immediately come to mind. Identification of EBTs and BOs does not come easily. Engineers used to classical methods of object-oriented modeling will have to change their

words, they must be stable over time. In many cases, determining whether or not something is enduring simply takes experience in a given field. However, most of the time it takes a little more. Field experience allows one to know how long given concepts or

sense. It is obvious that a new process introduced to a system is less likely to be enduring than a process that has had a place in the system for years. However, as evidenced by the now infamous “millennium bug,” not even a millennium of time can always be considered enough time for something to become enduring.

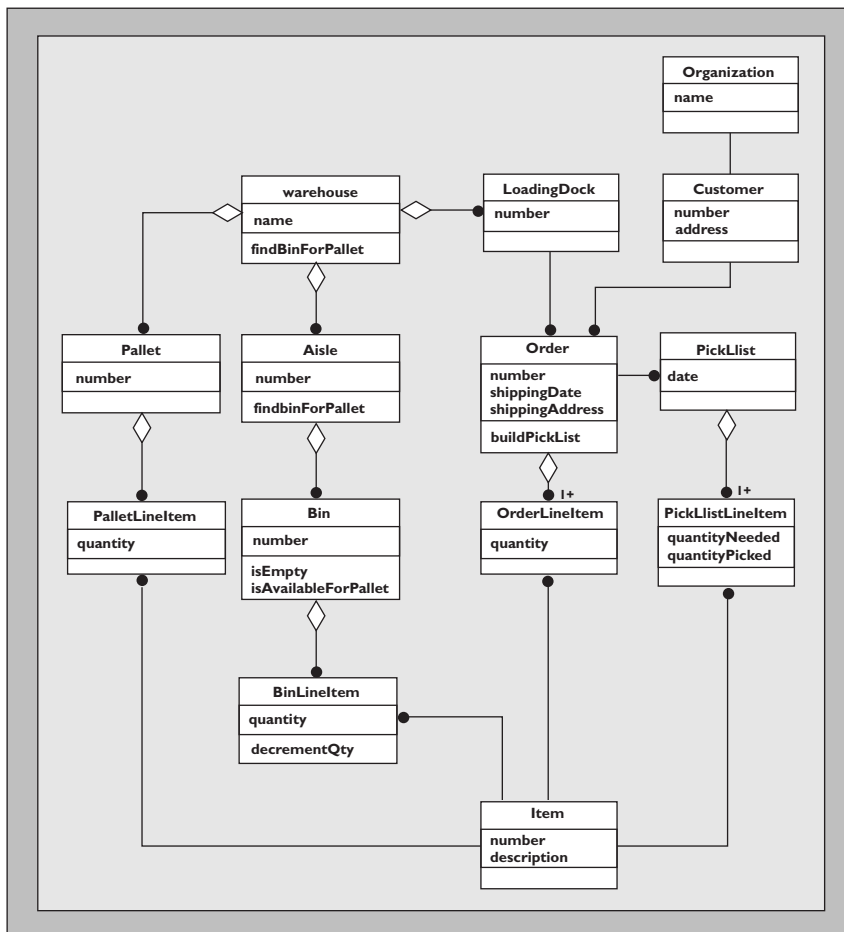


Figure 1. A model of a warehouse tracking system [1].

thought processes drastically to cope with these new concepts. These heuristics are designed to ease this transition in thought.

First and foremost, EBTs and BOs must be enduring. In other

objects have been involved with a given system. A long history in the field, however, does not necessarily translate into longevity in the future.

Determining whether or not an object will be enduring takes a knowledge of the system, a modicum of intuition, and common

Industrial Object Identification

Identifying IOs is easy. Usually, the objects one designs into a classical object model are IOs. To determine whether an object is an IO, one must simply ask: “Can this thing be replaced with something else? Will the system remain viable if some other object replaced this one?” If the answer to either of these questions is “yes,” then the object is most likely an IO.

One should also look at the “tangibility” identification criterion. If the object represents some physical entity, that object is very likely to be an IO. If it represents a piece of machinery, a button, an input device, or an output system, it must be replaced any time the physical entity it represents is replaced. Thus, it should be considered an IO and placed toward the periphery of stable designs.

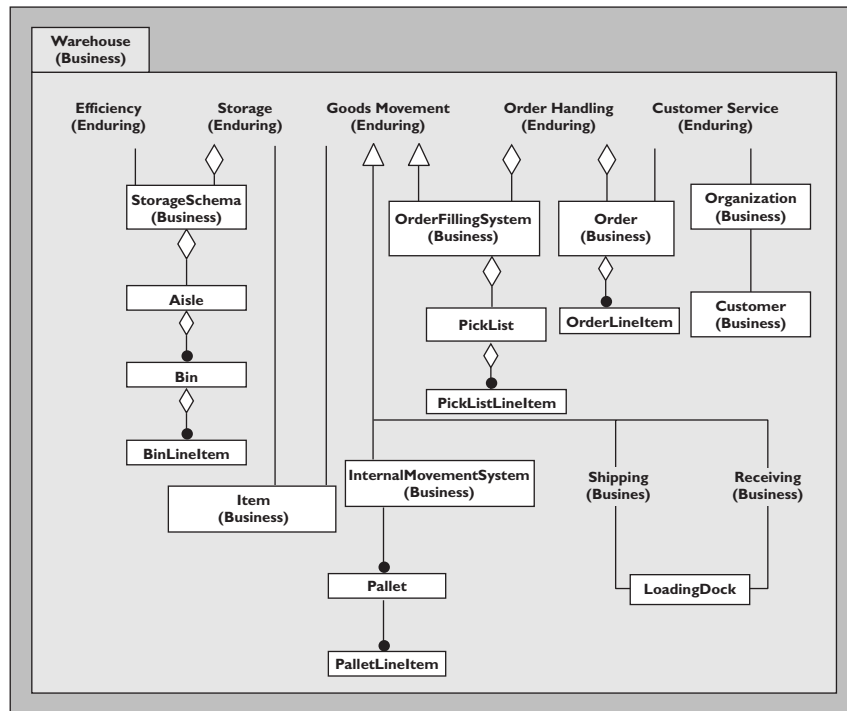
Top-Down Identification

Once one gets a feel for identifying IOs, identification of BOs and EBTs becomes far easier. There are at least two viable methods for finding BOs and EBTs. One method is to take a top-down look at the system. Start with the whole system.

Then, start breaking off conceptual pieces of the system. “What is the system used for? What must it be? Why are we building this system?” For example, in a kitchen, the system serves as a meeting area and a living area and is used for preparing food. Warehouses, as another example, store commodities, serve customers, and serve as hubs for the shipping of goods. Name each of these subunits and continue to break them down. Once IOs, which are easy to identify, are encountered, stop and back up one level. This level of “objects” contains good candidates for BOs and EBTs.

Bottom-Up Identification

Another method is to start at the bottom and work upward. Start by building a classical model of the problem at hand. Then, find the IOs in the model. Again, this should be easy. It is likely most of



Food must be stored cold to preserve it. Aha! We have found the EBT of food preservation. Continue this grouping and finding of concepts until it cannot be

Figure 2. An EBT-based model of the warehouse tracking system.

plete analysis of the problem at hand. For example, it is highly unlikely that a bottom-up identification scheme will ever have found the “livability” EBT of the kitchen model shown in the case study.

As evidenced by the now infamous “millennium bug,” not even a millennium of time can always be considered enough time for something to become enduring.

No Silver Bullets

There are no silver bullets. Just as there are no hard-and-fast rules for finding EBTs and BOs, there are no silver-bullet heuristics for separating the two. There are a few decent heuristics that can help, though. One method of separating EBTs from BOs is to examine the tangibility of the theme and object candidates. Usually, EBTs are themes—they are conceptual, overlying entities of a system. BOs, on the other hand, are far more tangible

the objects in the classical model will be IOs. Try to group these objects and find the concepts covering them. For example, in the kitchen, one might pair the refrigerator and the freezer. “What are the refrigerator and freezer used for?” one might ask. Well, they keep things cold, obviously. “Why keep things cold?”

continued. The only task remaining is to separate the EBTs from the BOs.

Note the bottom-up method of identification will only find those EBTs and BOs that cover IOs that already exist in a classical object model. Many other EBTs and BOs should probably still be considered in order to form a com-

Thinking Objectively

objects. They are overlying processes or methods or step-by-step instructions. While BOs are more conceptual than IOs, they are usually more tangible than EBTs. Consider, for example, the recipe that was documented in the kitchen case study. Recipes do not change over time even though the entities used to prepare recipes do. Also consider the “storage schema” object in the warehouse case study. In order to

process. Nevertheless, EBTs are a vital analysis artifact. The identification of EBTs may be one of the only methods a software engineer has at his or her disposal that can help design stable software over time.

Used properly, EBTs and BOs can create a stable foundation around which stable software systems can be built. The identification of these stable foundation entities will require a drastic

retraining requires an in-depth analysis of the problem domain and the business systems the problem is dealing with. This may require far more interaction between software engineers and those people who will be using the resulting software system. Finally, the identification of EBTs and BOs will require more of a “gut reaction” than most software engineers are used to relying on. Again, it will require some time to complete this mental retraining.

This is not a silver bullet. The use of EBTs and BOs is not a replacement for other good software engineering practices and common sense. It is obvious that, when analyzing problems, concentrating on those aspects of the problem that will remain stable will yield a more stable analysis of the problem. Once we learn to identify these stable foundations upon which to build our solutions, we have made the first step toward accomplishing software stability. ■

EBTs are those themes that remain constant for a given system. They should be thought of as those stable objects that should make up the core of software systems.

be efficient at storing and retrieving commodities, warehouses should have some method of storing goods in an orderly fashion. That method may change, but there will always be some efficient method of storing commodities. One cannot hear, see, or feel the recipe contained in a chef’s brain. Nor could one pick up a storage schema. However, both the storage schema and the recipe can be translated to something more tangible than a nebulous theme.

Conclusion

The heuristics for finding EBTs and BOs that were documented in this column are merely suggestions. There is no guaranteed method for determining what is an EBT, what is a BO, or what is an IO. All that exist are a series of heuristics or rules of thumb that can be used to help in the

change in the way software engineers think about problems. However, the time and money saved by engineering more stable systems far offset the cost of this mental retraining.

EBTs are those themes that remain constant for a given system. They should be thought of as those stable objects that should make up the core of software systems. BOs are also stable objects around which to build software systems, but unlike EBTs, the internal processes of BOs may change if the need arises. Externally, however, BOs are as stable as EBTs.

Problem analysis using EBTs and BOs requires far more intuition and knowledge about the problem domain than more classical problem analysis techniques. EBTs and BOs are rarely stated in problem statements or supporting documentation. Their identifica-

REFERENCES

1. Coad, P., North, D., and Mayfield, M. *Object Models Strategies, Patterns, and Applications*. Yourdon Press, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.
2. Fayad, M.E. and Alltman, A. Introduction to software stability. *Commun. ACM* 44, 9 (Sept. 2001), 95–98.
3. Fayad, M.E. Accomplishing software stability. *Commun. ACM* 45, 1. (Jan. 2002), 111–115.

MOHAMED FAYAD (fayad@cse.unl.edu) is J.D. Edwards Professor at the University of Nebraska, Lincoln.

© 2002 ACM 0002-0782/02/0400 \$5.00