Mohamed Fayad

# Accomplishing Software Stability

The kitchen proves an ideal setting to illustrate how stability and change work together within software.

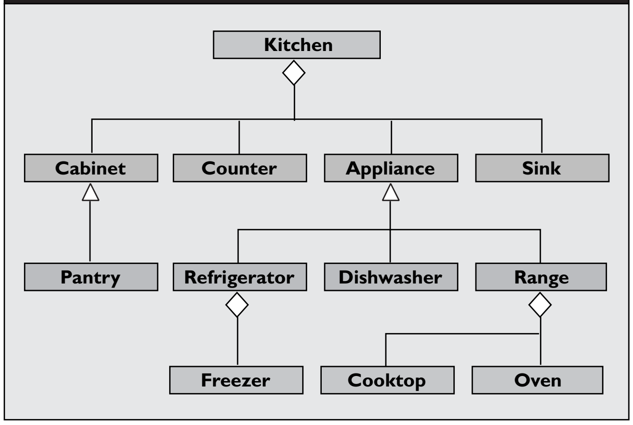
A descriptive example of the use of Enduring Business Themes (EBTs) and Business Objects is the typical kitchen. The concept of a kitchen depends on a variety of objects that do not remain constant over time. These objects can be considered "Industrial Objects."

Our kitchen model can be thought of as a tree of aggregations and generalizations. Many of the branches or leaves of the tree can change and the kitchen model breaks down. In this sense, the kitchen model is not stable over time.

Similarly, the problem with software systems today is that Industrial Objects like those found in the kitchen model (see Figure 1) are frequently observed to be within the core of design. Likewise, Industrial Objects are too often finely intertwined within the system implementation. In such systems, changes over time usually result in changes in the Industrial Objects. The results are unstable designs and an unstable code base.

If the designer properly considers those aspects of a kitchen that do not change over time the design of our kitchen is radically different. A proper design centers on enduring themes and Business Objects. Enduring themes are those concepts of the system that would remain constant over time. Externally, Business Objects also remain constant over time, but may have internal, intangible processes that change. In the case of the kitchen, the model centers on concepts such as lighting, storage and preservation, heating, and convenience. Objects such as the range and the refrigerator become secondary and are replaced easily as new technologies become available.

**Figure 1. An unstable classical model of a kitchen.**



To find the EBTs and Business Objects for any model, you must have enough experience in the domain of the problem to decide whether or not a concept is core to the system. Experience working in an area builds an innate feel for what is and what is not enduring. Often, this is not enough. A professional chef might be a good person to consult when trying to find the EBTs of a kitchen. A very experienced, professional chef might say, "I've always had pots, pans, a stove, an oven, a set of cabinets, and a refrigerator. These must be enduring themes of the kitchen." This chef would be wrong. The objects that he or she identified are, in fact, Industrial Objects—not EBTs. Despite the fact this chef has never been without these objects, these objects are nowhere near enduring enough to be considered EBTs.

To find the EBTs of a kitchen, one must distance oneself from the material items thought to be typical kitchen items. Instead, look at the "why's" of these items. Why does the kitchen have a refrigerator? Why does the kitchen have a range? Attaching objects to the answers to these questions yields the EBTs of a kitchen.

Although the model of the kitchen using these EBTs is more complex than the traditional kitchen model, it more accurately represents the essence of a kitchen and is a more stable model. Those objects that can change in the not-so-distant future are moved toward the periphery of the kitchen model while the hubs of the model are replaced with stable EBTs and Business Objects.

The EBTs of the kitchen not only include cooking, but livability, food storage, convenience, cleanliness, and cuisine. If any of these themes are changed or removed from the model, the model and the kitchen itself change dramatically. For example, if the livability theme is taken out of the model, the model is no longer a useful design for a kitchen in a home, but possibly for a kitchen in the back of a restaurant. If the cuisine changes, it may also be necessary to change some structural features of the kitchen. In other words, the kitchen is only as stable as its EBT constituents.

The cooking EBTs must be present, of course, as it is an aggregate of those tools used within a kitchen to prepare food. The cuisine theme represents the food cooked within a given kitchen. Of course, it is an aggregate of food and the recipes used to prepare the food. Food storage and preservation is an aggregation of those objects that perform the cold and dry storage duties of a kitchen. Livability is the theme that makes the kitchen part of the home. Cleanliness is something that must be maintained within a kitchen to ensure food does not become tainted with disease. Finally, convenience is what makes the kitchen easy to use and keeps the cook working in the kitchen happy.

The two Business Objects in the model are "light" and "recipe." Both are Business Objects instead of EBTs because they both contain internal processes that may change without changing their external relationship to the problem. Recipes for given cuisine, for example, can contain many different processes that result in the same dish. Light can be produced in many different ways yielding the same, comfortable, homey light.

**A summary of the identification criteria for enduring business themes, business objects, and industrial objects.**

| | Enduring Business Themes | Business Objects | Industrial Objects |
|---|---|---|---|
| **Stability Over Time** | Stable over time | Externally stable | Unstable |
| **Adaptability** | Adaptable without change | Adaptable through internal change | Not necessarily adaptable |
| **Essentiality** | Essential | Essential | Replaceable |
| **Intuition** | Intuition | Intuition and reading | Reading only |
| **Explicitness** | Implicit | Implicit or explicit | Explicit |
| **Commonality to the Domain** | Core | Core | Peripheral |
| **Tangibility** | Conceptual | Semi-tangible | Tangible |

## Identification Criteria

Experience points to seven tests as criteria for distinguishing EBTs, Business Objects, and Industrial Objects. While there may be exceptions, these tests generally provide a very reliable method for identification (see the table).

**Stability.** Probably the most important criterion for identifying EBTs and Business Objects is they must be enduring. Both must be stable over time. Where they differ is in *how* they are stable. EBTs are completely stable both internally and externally. They have never and will never change. All the

EBTs identified in Figure 2 are completely stable. Kitchens must always facilitate cooking and provide a livable environment. Likewise, friends must always consider both their friendship and their finances when reconciling loans, and warehouses must always store goods in an efficient manner and serve their customers if they expect to stay in business.

Business Objects can change internally. Externally, however, they must remain the same. The warehouse case study has a perfect example of this. In the warehouse study, the storage schema is a Business Object. Externally, it must always be a method of storing goods in the warehouse. Internally, however, it is an aggregate of Industrial Objects. Any of these Industrial Objects can change or be replaced. As long as the storage schema is still externally the same, the model is still valid. Item is another Business Object in this model. Internally, it does not matter if the item is a bathroom slipper or a telephone directory. Externally, it is a storable good and that is all that matters to the system.

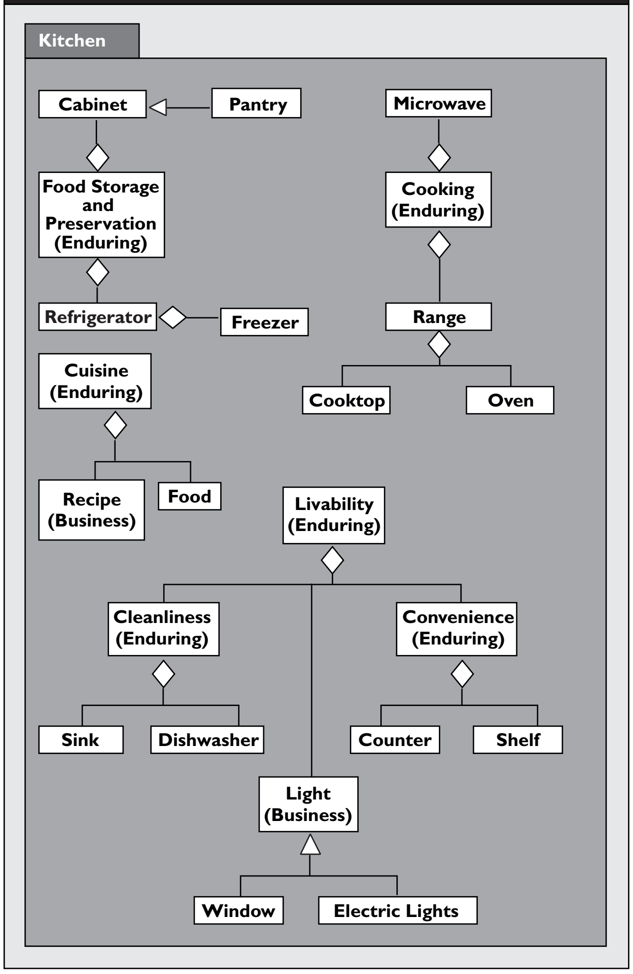
Industrial Objects can be changed at will. If the system is designed correctly, changing these objects should not cause changes to ripple through the entire system. For example, in a kitchen, the range can be replaced with either a barbecue or a food replicator. As long as it still facilitates the theme of cooking, the model remains intact. Do not take this to the extreme, however. In the warehouse example, one cannot replace the loading dock with a banana and expect this to work. The Industrial Objects must continue to fit with the Business Objects and EBTs to which they are attached.

**Adaptability.** An object is considered adaptable if it can be used in a system despite changes the system may go through. Some objects are adaptable without any changes to the objects themselves. Others require some changes to the internal processes of the objects. Still others must change completely.

EBTs cannot change at all. How they function internally as well as how they interact with the system externally cannot change. Therefore, EBTs are adaptable without making any changes to the EBTs themselves. It does not matter what changes the system goes through. The EBTs of that system remain constant. They must adapt to the changes without changing themselves [1–3].

Business Objects can only change internally. The processes used inside Business Objects and the other classes that make up Business Objects can change. Everything else about the Business Object must stay the same. Thus, Business Objects are adaptable through internal change.

**Figure 2. A more stable model of the kitchen problem.**



A very experienced, professional chef might say, "I've always had pots, pans, a stove, an oven, a set of cabinets, and a refrigerator. These must be enduring themes of the kitchen." This chef would be wrong.

Finally, all aspects of Industrial Objects can change. As long as the Industrial Object is reasonable, the system should still be able to work with it. Industrial Objects do not have to be adaptable to change. If a change to a system warrants the removal or replacement of an Industrial Object, that should be fine and the system model should account for this possibility.

**Essentiality.** EBTs and Business Objects are both essential features of any system. For example, if one removes the storage EBT from the warehouse model, the warehouse is destroyed. A warehouse cannot function without the theme of storage. Similarly, if the cooking theme is removed from the kitchen, the kitchen turns into a room that can store food. The same thing would happen if Business Objects are removed from any of the systems. If the loan object is removed from the loan model, then the model is no longer of a loan between friends. If the customer object is removed from the warehouse model, then the warehouse cannot turn a profit and will go out of business.

The only nonessential objects are Industrial Objects. For example, you can easily remove the range from the kitchen model. As long as there is still some facility to cook food in the kitchen, the model is still a model of a kitchen. Aisles and bin combinations may be removed from the warehouse model and replaced with some other storage medium. As long as there is a way to store things in the warehouse, the warehouse concept remains intact.

**Implicitness/Explicitness.** A common error in systems requirements analysis is EBTs are almost never explicitly defined in the problem statement. EBTs are frequently left out of all supporting documentation. For example, there is no mention in the problem statement for the warehouse system that the warehouse must serve its customers. The customer service theme must come from knowledge about how a warehouse works and how it survives. The customer service theme is implicit.

Business Objects are sometimes stated in the problem statement or supporting documentation, but with equal frequency, they are not identified. Again, in the warehouse example, the customer Business Object is mentioned and it is explicitly stated that customer information must be stored. However, there is no mention in the problem statement of the storage schema Business Object. Modes of stocking products change over time, but the need to model each storage schema that is used is enduring. This, again, comes from knowledge of how a warehouse works and what must be provided to make a warehouse run efficiently. Frequently, Business Objects are left out of the analysis and design because they are implied.

Industrial Objects can almost always be found by reading the problem statement or its supporting documentation. In the warehouse case study, all of the Industrial Objects in the model were taken directly from the problem statement. Why is this the case? Remember the problem statement states the problem at hand. Consequently, it must state things in terms of what is happening now. It is not usually concerned with stability or change. Therefore, many of the objects mentioned in problem statements will be Industrial Objects.

However, this means the old way of documenting systems requirements must be changed. It is critical to identify both the EBTs and Business Objects early in the analysis of the problem. In short, it is important to define both the implicit and explicit features of the problem during analysis and design.

**Intuition.** Because EBTs are almost never explicitly stated in the problem statement, one must either have a great deal of experience in the field in question or one must read between the lines of the problem statement. One must use intuition when finding EBTs. For example, there is some mention of the movement of goods in the warehouse problem. However, there is no explicit statement in the problem stating this movement is important. Intuition must be used to determine this.

Business Objects are sometimes explicitly stated in problem statements. When they are not, however, one must again use intuition to find them. The storage schema and order-filling system objects of the warehouse model were found using intuition. The customer and organization objects are also Business Objects, but they are more commonly addressed and no intuition is required to find them.

Finding Industrial Objects is often a simple chore of extracting them from documentation. Since they are often explicitly stated somewhere, no intuition is required to find them. If one simply reads a problem statement and extracts all the nouns from it, one will find almost all of the Industrial Objects relevant to the system. Therefore, it is common to see objects such as product, location, and forklift in the analysis products of a warehouse system.

**Tangibility.** Tangibility helps to verify that EBTs are actually EBTs and Business Objects are actually Business Objects. EBTs are themes. As such, they are almost never tangible items. Themes are very conceptual. Take "livability" from the kitchen model as an example. This is not a concrete object. One cannot even associate a concrete object with "livability" and capture the entire essence of livability.

Business Objects are *semi-tangible.* This may be difficult to grasp at first. Take "customer" from the warehouse case study as an example. It is a Business Object. One can easily associate a human or a company or some other tangible item with a customer.

Finally, Industrial Objects are almost always concrete. One can see a range in a kitchen or walk down an aisle in a warehouse. If an object in a model represents a concrete entity, then it is most likely an Industrial Object. It does not take much thought to see why. If an object in a system represents a concrete real-world entity and that entity changes, then that object in the system must change with it. Since concrete entities usually change both internally and externally, the object in the system cannot be a Business Object. Therefore, it must be an Industrial Object.

**Commonality to the Domain.** Commonality to the domain deals with where the object should go in a system. Both EBTs and Business Objects are stable. The rest of the system should be built around these things. Therefore, the EBTs and Business Objects lie at the core of the system and should be modeled as such. Those things that can change, the Industrial Objects, should be moved as far to the periphery as possible so that changes in them do not have a ripple or cascading effect through the rest of the system.

Classifying artifacts of complex applications as EBTs, Business Objects, and Industrial Objects is a vital technique for systems analysis and design. Applying these concepts allows software engineers to change the way we define problems. Therefore, these concepts yield an important paradigm shift in systems engineering—an approach that specifically targets software stability. C

## References

1. Fayad, M. *Software Stability.* MightyWords, 2001
2. Fayad, M. and Cline, M. Aspects of software adaptability. *Commun. ACM 39*, 10 (Oct. 1996), 58–59
3. Fayad, M. and Laitinen, M. *Transition to Object-Oriented Software Development.* (Aug. 1998) John Wiley & Sons, New York, NY.

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

If the reader is interested in learning more about these concepts, he or she will find some case studies posted at www.cse.unl.edu/~fayad—check Case Studies. If the reader would like to submit a sample and specific domain problem, I would be happy to model it.