The Dynamic Business Object Pattern

Russ Rubis (Florida Atlantic University), Dr. Ionut Cardei (Florida Atlantic University)

***Proceedings of the 20th Conference on Pattern Languages of Programs(PLoP)*. 2013.**

Presenter: 李易庭 Yi-Ting Li
Date: 2022-06-17

Dynamic Business Object Pattern

Intent

Describes an ***extensible*** design for business objects used for business applications and a structure for their ***platform-neutral*** specification.

Example

Consider a shopping cart. It is a customer ***request*** for goods or services performed using a web based application.

After selecting items and adding them to a shopping cart, the customer provides the billing and shipping information, then ***submits*** the request for processing.

The vendor then ***receives*** the shopping cart request, verifies the billing information, fills out the order and ships it to the customer.

Finally, the vendor ***notifies*** the customer via email that the shopping cart request has been **completed**.

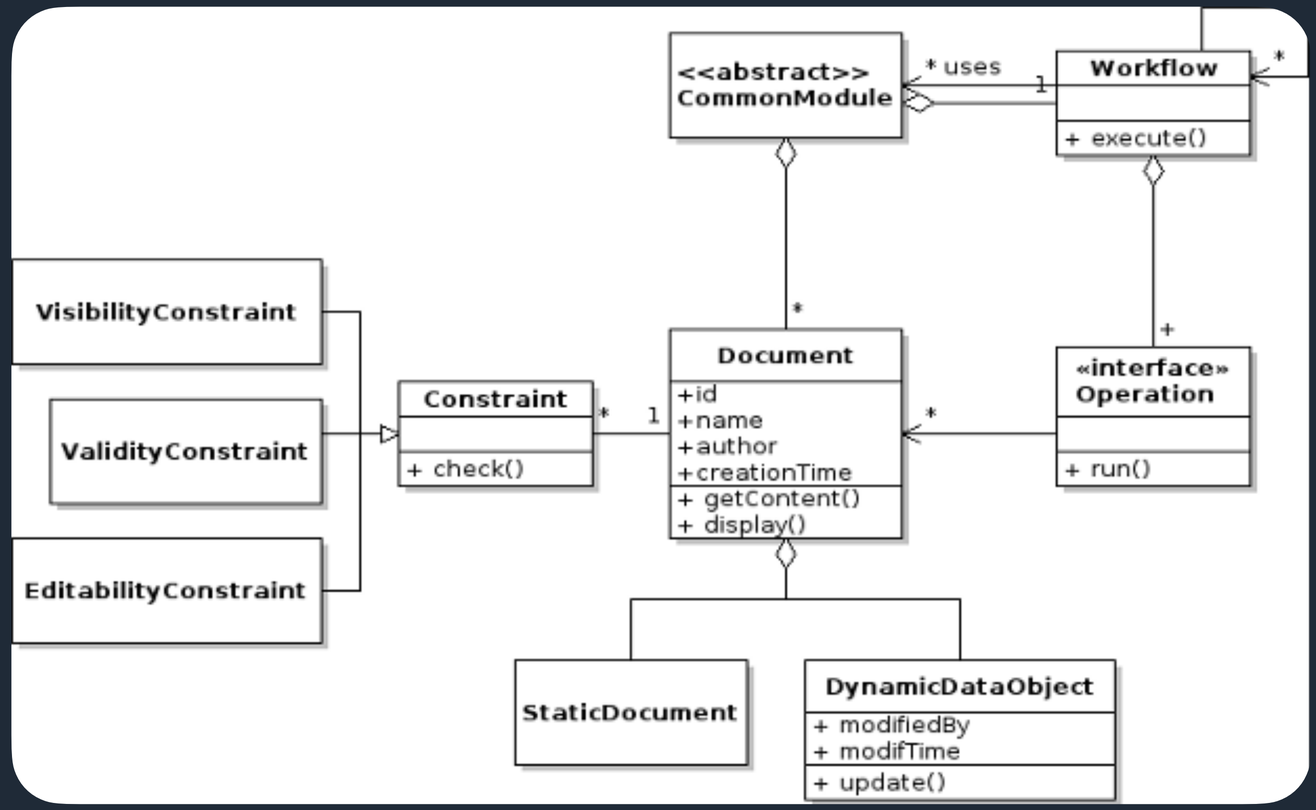
Problem

- Business applications involve ***complex*** distributed operations orchestrated on client and server side.
- Applications rely on a ***variety*** of documents with static and dynamic content for presentation and for data storage.

Solution

3 main elements:

1. **Dynamic Data Object (DDO)**
2. **Static Document**
3. **Workflow**



Dynamic Data Object (DDO)

- Could ***change over time*** as a result of some business process or user input.

- Unlike a static document, its contents can be (and often are) changed, and unlike a workflow, it is not a process, but rather an object (or an entity) ***representing a specific set of data.***

- DDO in the Common Module is an ***abstract object type*** and is assumed to contain a ***limited set of attributes*** within the Common Module.

- The DDO employs a set of ***constraints*** to control what access rules and specific behaviors are permitted on the object and how they should be accomplished.

Static Document

- Any document whose data ***cannot be changed by end users*** (i.e. clients) of the application

- Only the ***visibility constraint*** can be applied to it.

- Within the Common Module the static document can be divided into 2 groups: ***print media*** and ***electronic media***.

Constraints

Provides a way to ***limit the access*** to the dynamic data object, and to some extend to the static data object as well.

Visibility Constraint (ViC)

Contains logic that dictates the ***visibility*** of a component or attribute of that object.

The ViC is not required for every object or entity; it should only be used when needed by the application logic.

Validity Constraint (VaC)

Determines the ***validity*** of an element (sub-object or attribute) of a DDO.

- A VaC is not required for every object or entity by default.

Editability Constraint (EC)

Contains logic that determines whether an element (component or attribute) of a DDO is ***editable***, i.e. changeable by some user interface.

- The EC is not required for every object or entity
- Element editability can be qualified (parameterized) with ***contextual information***, such as user role.

Workflow

Describes the ***logic associated*** with the Dynamic Business Object, the sequence of operations that the Dynamic Business Object goes through during its *lifetime.*

The runtime for a workflow is ***platform-neutral***, ***generic***, and is ***interpreted*** by the various actors involved in its execution.

Common Module’s workflow includes just ***two trivial*** no-op operations, Start and Stop, that can be specialized in sub-modules.

Figure 2. Default workflow for the Common Module.

Implementation: a shopping cart

Implementation

This example will show how a Dynamic Business Object Pattern employs an existing module to create (or extend) a new module to address a specific business need.

In this example we'll create and apply pattern to a Shopping Cart Request Module.

Our shopping cart module will be based on an existing module (Abstract Customer Request Module).

The most basic customer request can be broken down into the following steps:

1. Customer initiates a request
2. Request is received and processed by the business
3. Customer is informed of the request completion

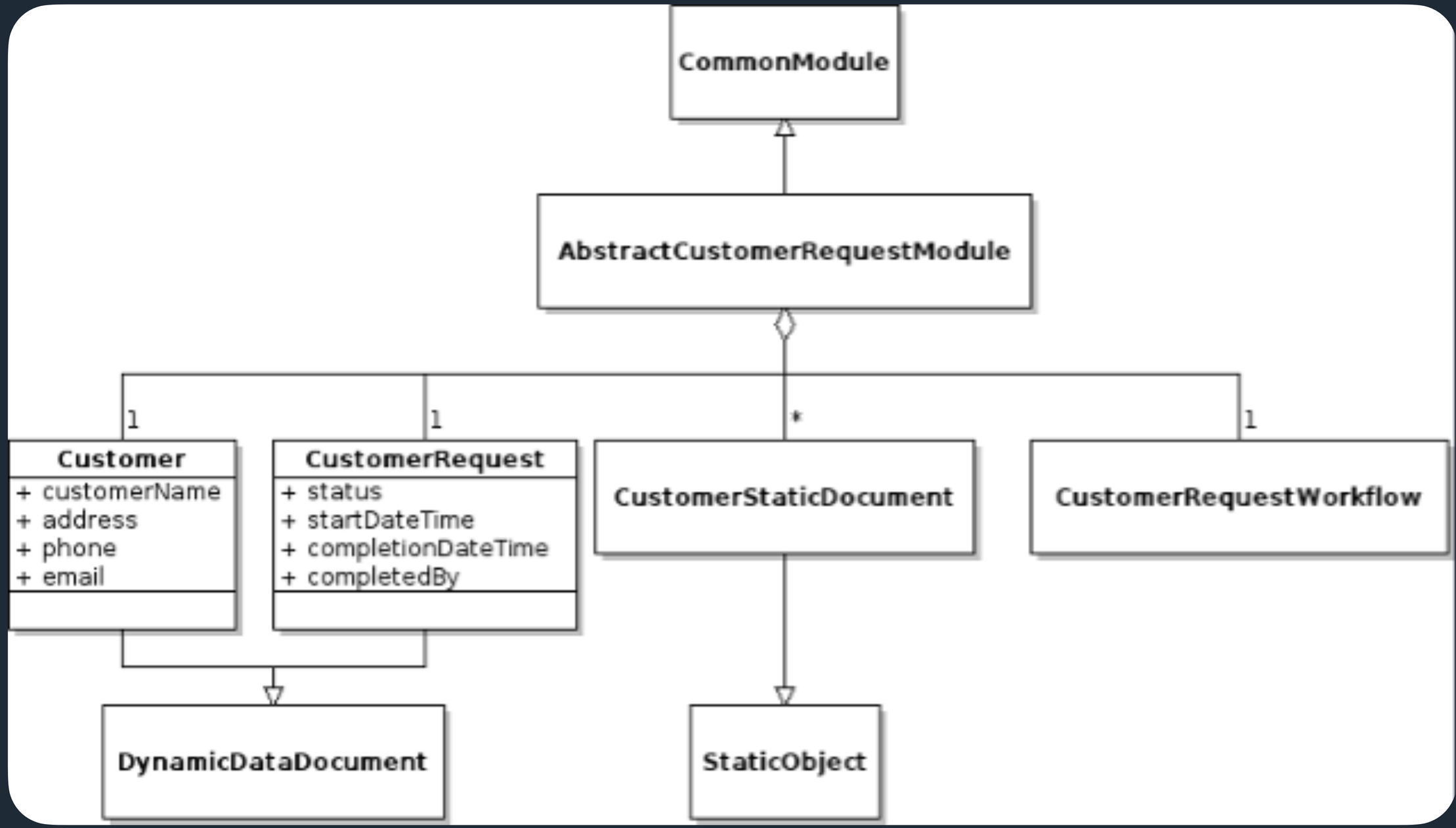
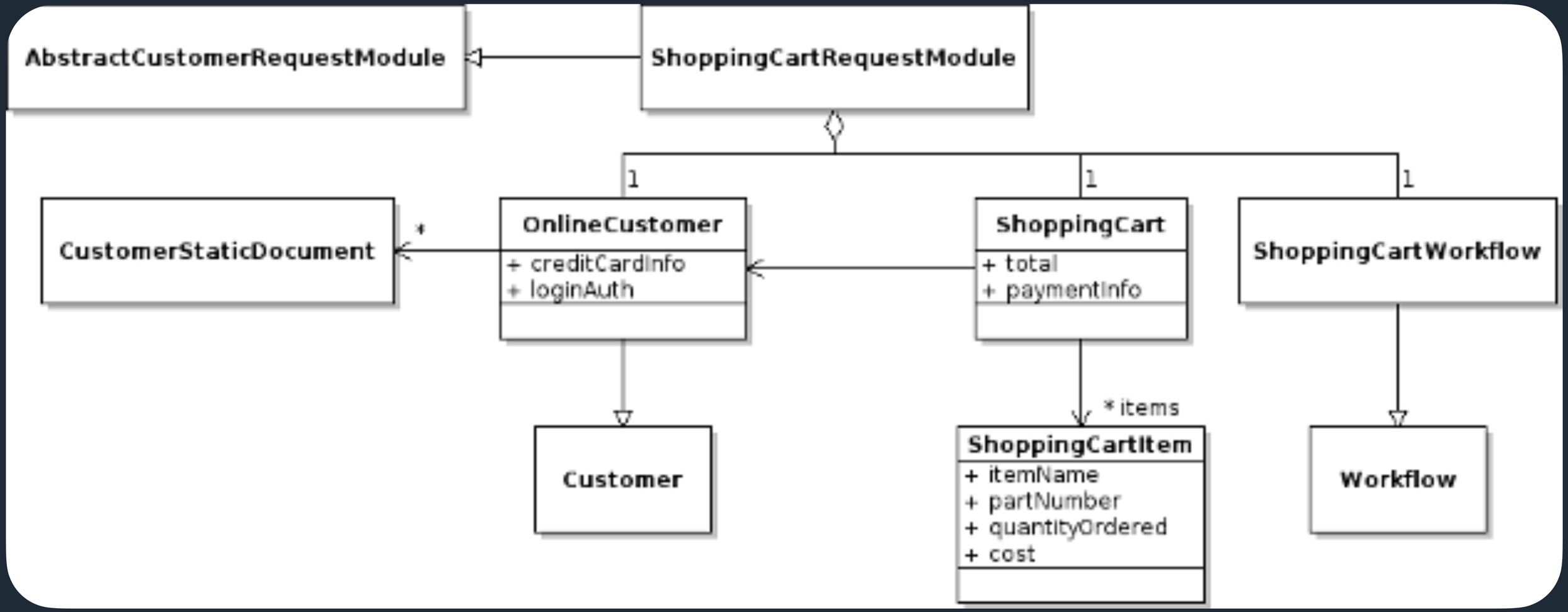
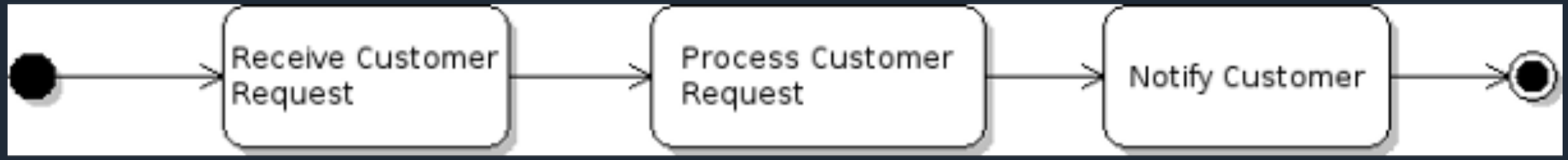


Diagram showing Common Module specialization for a Customer Request.



Shopping Cart Request Module specialization:
support for online customers, shopping cart, and its items



The workflow of a shopping cart Dynamic Business Object. Receive Customer Request Abstract customer request workflow

The workflow of a shopping cart Dynamic Business Object. Receive Customer Request and Notify Customer are operations inherited from the Abstract Customer Request Module

Consequences

Benefits

Dynamic Business Object Pattern can greatly simplify the development of business objects, especially if a growing library of dynamic business objects is accumulated and employed over time.

The pattern also takes into account the processes (via workflows) that are required to support the business object's life cycle. The pattern is also flexible enough to allow for future changes and additions, if any should be required.

Liabilities

Dynamic Business Object Pattern does not take into account the specifics of dynamic data object support at the user interface and data persistence levels.

In addition, until a Dynamic Business Objects library is amassed, much of the work will be required to be performed from scratch.