MANAGED OBJECT PATTERN

Class Behavioral

Intent
Provide direct control of individual resources in a managed system. The Managed Object provides a one-to-one mapping to the manageable entities, as well as a more uniform interface.

Motivation
You need to manage a large collection of resource entities. Each entity can have an interface that widely varies from the interface of other entities such that there is a great number of different interfaces having somewhat similar features. Such variation can be caused by the nature of the entity, its manufacturer, etc. In addition, there is often a hierarchical and/or containment relationship amongst the entities being managed.

The problem here is to provide a unified interface for representing and controlling the resource, and yet to provide a fine degree of control. While attempting to solve this problem, the developer must deal with a number of conflicting forces. For instance, the resources have widely diverging interfaces, and yet they need to be uniformly controlled while taking into account the relationships amongst the resources.

The proposed solution is to use an instance of the Adapter pattern for each individual resource to translate its particular interface into one shared by all resources. This Adapter is called a Managed Object. It enables management operations to be performed using a single interface. The Managed Objects can then be aggregated using the Composite pattern. The composition is called a Management Information Base (MIB). It can be sensitive to the hierarchical or containment relationship that exists amongst the resources. The layout and interface of the MIB follows a specification which describes the content to the outside world.

The result is a flexible structure that appropriately mirrors the organization of the resources and uses a single interface for management. Individual resources can be located by traversing the MIB (with the Iterator pattern and/or the Visitor pattern) and applying operations on the Managed Object's interface. It will in turn apply the corresponding operation to the resource using the appropriate interface.

Applicability
The Managed Object pattern works best when:

- There is a need to give structure to the information in a sizeable information base.
- There is a large number of widely differing resources.
- The resources have an inherent hierarchical relationship.
- There is a need to provide a unified interface to managed resources within an application (as opposed to within a distributed system, in which case the Manager-Agent pattern might be more appropriate).
Structure

- **MIB**
  - Provides a complete representation of the subsystem.
  - Entirely composed of Managed Objects.
- **Specification**
  - Published information about the MIB and its interface.
  - Used by remote management systems to learn about the contents of the MIB.
- **Managed Object**
  - Represents a single resource
  - Handles calls from the management system to the particular resource it represents.
- **Resource**
  - Vendor-specific component of the system that is accessible through the management interface.

Collaborations

- The MIB presents an aggregate view of the Managed Objects.
- The Managed Object represents the resource being managed, it translates the resource’s specific interface into the interface that is expected by the management system, and it implements management-specific operations (e.g. the \texttt{VisitMe()} operation in the \texttt{Visitor} pattern).

Consequences

The Managed Object pattern has the following benefits and liabilities.

1. *The MIB and the Managed Objects offer a single interface to communicate.* This interface encapsulates the proprietary protocols of the resources and simplifies the management of the system. A managing application is then able to communicate with any and all the resources in the MIB, regardless of their nature or particular characteristics.

2. *The Managed Object provides indirect access to the resource.* This pattern adds another level of indirection to the processing of a management operation. This will further
degrade the system’s performance, unless care is taken to keep the overhead to a minimum.

3. The Managed Object must offer an interface that can gracefully support a large variety of proprietary interfaces. It must allow for the access to internal parameters as well as the application of specific functions to the resource. The specification should express the details of the MIB in a manner as neutral and as precise as possible. This way, the interoperability of the components is ensured regardless of their underlying source or implementation.

Implementation
Here are some useful techniques for implementing the Managed Object pattern.

1. Use accessors functions to grant access to the attributes. The internal state of the Managed Object is encapsulated within the class but must still be accessible to the management system. There are two possible approaches at getting to those state attributes: one set of accessors per attribute or generic accessors. In the first case, the Managed Object has a separate accessor method for each access type (read, write, ...) for each attribute. The alternative is to have a single method per access type and have it handle all attributes by name. This is the method used by CMIS. The Inherited Behavior pattern allows for the reuse of the attribute access semantics across the inheritance hierarchy of Managed Objects.

2. Use a generic invocation method to call management methods on the Managed Objects. Rather than having a public method for each operation, the Managed Object can publish a single method through which all operations are invoked. When coupled with the single accessor method per access type scheme described above, the interface to Managed Object is frozen in the root of the hierarchy and enables a high degree of polymorphism. The downside is that either strong type semantics must be sacrificed, or extra measures must be taken to manually ensure they are preserved, a task that would normally be handled by the programming language’s compiler.

Known Uses
The Managed Object pattern is used in CMIS, the OSI Network Management standard. It has been implemented in Layla, a C++ framework for CMIS applications.

A diluted version of the Managed Object pattern can be found in SNMP, the Internet Network Management standard. In SNMP, only the leaves of the Composite structure are mapped to actual resources. The state of the Managed Object instance is kept in relational tables while the functionalities are implemented as standalone functions operating on the MIB.

Related Patterns

Adapter: The relationship between a Managed Object and its resource follows the Adapter pattern. The Managed Object plugs an expected interface onto the managed resource.

Composite: The relationships between the Managed Objects can be modeled using the Composite pattern. Each Managed Object can be seen as being composed of the Managed Objects under its control / contained in it, if any.

Contacts
http://www.iro.umontreal.ca/~keller/Layla
Jean Tessier, AT&T Laboratories, Advanced Technologies Division, Holmdel, NJ.
Jean.Tessier@att.com

Rudolf K. Keller, Département d’informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal (Québec) H3C 3J7, Canada.
keller@iro.umontreal.ca