# AN ARCHITECTURE FOR AN OBJECT MANAGER FOR STRUCTURAL ENGINEERING DESIGN SYSTEM

**Jamal A. Abdalla**[1]

## ABSTRACT

The Object Manager (OM) provides facilities for extracting objects from the Central Data Base (CDB), for instancing active objects in the Application Data Base (ADB), for committing object data back to the CDB, for making active objects persistent, for making passive objects active, for managing the object identifiers and for managing objects' memory and disk spaces. The Object Manager carries out these responsibilities of managing objects as a whole irrespective of their type and content. In this paper an architecture of the Object Manager will be shown and the features and capabilities which are required of an Object Manager will be presented. The Object Manager is part of an integrated system. The architecture of the integrated engineering system resembles that of the Object Management Group (OMG) Reference Model Architecture. The Object Manager and its Application Data Base act as the Object Request Broker (ORB) of the integrated system. The interaction of the Object Manager with its ingredient elements and with the ADB together with its operations will be presented using the Unified Modeling Language (UML). A note on the implementation of the Object Manager components will be given.

## KEY WORDS

Object Manager, Central Data Base, Application Data Base, UML.

## INTRODUCTION

An integrated object-oriented software system for structural engineering usually consists of objects belong to different domains such as geometric modeling, conceptual design, preliminary design, structural analysis, detailed design, and drafting [Fenves et al. 1988, Abdalla 1991, Rivard 2000, Mora 2004]. In a central data base architecture for integrated structural engineering system, objects of shared data are usually stored in this common data repository and are shared among several application programs [Abdalla 1991]. These objects have different structure and content and they need to be managed differently than regular data. Since objects are encapsulation of data and operations, the database management demand for objects is different from that of simple data types. Therefore, an Object Manager (OM), which is less elaborate than a full Data Base Management System (DBMS) can be used to manage objects in the Application Data Base (ADB). To perform design operations, objects need to be extracted from the Central Data Base (CDB) and stored in the ADB. Once

[1] Dept. of Civil Engineering, American University of Sharjah, POBox 26666, Sharjah, UAE
Phone 971/65152959, FAX 971/65152979, jabdalla@aus.edu.

objects are extracted from the CDB and placed in the ADB, it is the responsibility of the OM to keep track of these objects. Also, objects need to be activated, de-activated and permanently stored into the disk. Active objects in memory need to be tracked such that communication among design objects becomes possible [Abdalla 2004]. The Object Manager carries out these responsibilities of managing objects as a whole irrespective of their type and content.

This work builds on previous research work [Abdalla 1991, Powell et al. 1989] and extends it to incorporate interoperability using the Object Management Group Reference Model Architecture and its data modeling language [OMG 2006], among others. The paper is mainly concerned with the architecture of the Object Manager and the interaction among its elements and among the Object Manager and the Application Program. The paper will first present background about the object data model and object management. It then outlines the features of the Object Manager, followed by the architecture and ingredients of the Object Manager. Operations supported by the Object Manager will then be noted, together with details of two sample operations and how they are carried out and the steps involved. A note on the implementation of the Object Manager will be given.

## OBJECT DATA MODELS AND OBJECT MANAGEMENT

The issue of object data models and object management has been addressed by several researchers in the last two decades [King 1986, Stonebraker 1988, Atkinson et al. 1989, CADFC 1990, Ozsu et al. 1994, Darwen et al. 1995, Loomis 1995, Ahn et al. 1998, Catell et al. 2000, OMG 2006]. Atkinson et al. (1989) presented the first manifesto paper on which they emphasized pure object-oriented features while Darwen et al (1995) presented the third manifesto paper on which they merry the relational and object oriented database systems. The second manifesto paper was presented in 1990 by the Committee for Advanced DBMS Function Corporate (CADFC 1990) in which the relational model and its features were more emphasized. The Object Management Group (OMG 2006) was formed to establish specifications, standards and guidelines for object data modeling and object management. The result is the Object Management Group Reference Model Architecture as a framework for distributed objects and the Unified Modeling Language (UML) as a data modeling language. Ahn et al. (1998) presented a survey of architectural features of contemporary object storage systems. They explored object managers in combination with storage systems by focusing on architectural issues and proposed a general guidelines for the design and implementation of object storage systems.

Modeling of engineering data for integration of the design process usually involves the modeling of form, content, function and behavior of design objects [Abdalla et al. 1991]. Object Management Group [OMG 2006] had suggested a distributed object management framework [Beeharry et al. 1994, Vinoski 1997] and specified standards that concentrate on the modeling of structure, behavior and interaction of objects. OMG uses the Unified Modeling Language [OMG 2006] as its data modeling language and it is used in this study to

model interaction among objects. The Unified Modeling Language [Siegel 2006] defines several types of diagrams for modeling objects' structure, behavior and interaction as follows: (a) static application structures diagrams – this includes Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram; (b) general types of behavior diagrams – this includes Use Case Diagram, Activity Diagram, and State Machine Diagram; and (c) different aspects of interaction diagrams – this includes Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram. The sequence diagram is used in this paper to model the interaction among the elements of the Object Manager and other elements of the integrated system.

Management of objects usually involves several management planes as shown in Figure 1. In this study only management of objects within the object manager plane shown in Figure 1 is considered. Management of design objects' versions is addressed elsewhere [Abdalla et al. 2000].

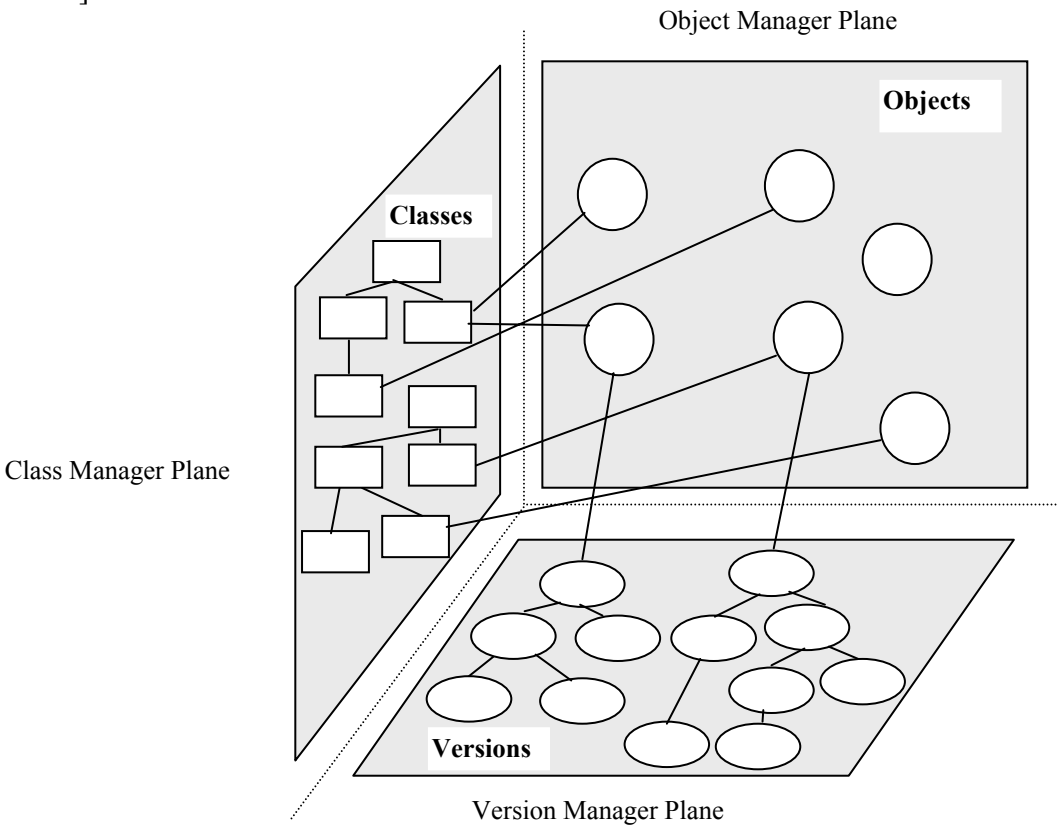


Figure 1: Management planes in integrated engineering systems

## FEATURES OF THE OBJECT MANAGER

Although there are similarities in the functions of the OM and a DBMS, there are differences in their role, the type of data they manage and the authority each has over the data it manages. Some of these differences are as follows [Abdalla 1991].

(1) The major role of the OM is to provide objects with addresses of other objects, to assist them in sending messages to each other. Objects then communicate with each other for querying about more detailed information.

(2) Since the Object Manager manages objects as a whole irrespective of their structure and internal details, it does not know specific details of the objects it manages,.

(3) The OM can not enforce consistency and integrity constraints on the object data since it has no knowledge of object details.

(4) The OM usually supports only one AP with its ADB and both acts as the Object Request Broker.

Therefore, the OM provides facilities for extracting objects from the CDB and instancing them as active objects in the ADB; for committing object data back to the CDB; for making active objects persistent; for making passive objects active; for managing the object identifiers, memory addresses and disk locations; and for managing the memory and disk spaces. The OM manages complete objects, and has no knowledge about the attributes or messages associated with any object. However, the OM provides support for certain queries about objects, including the object class and whether the object was created by the AP or extracted from the CDB.

## ARCHITECTURE AND INGREDIENTS OF THE OBJECT MANAGER

The concept of a central data base (CDB) as a repository for objects common to several applications is adopted as a basis for the work presented here. Each application program (AP) has its own local data base, termed an application data base (ADB). Some of the objects in the ADB are extracted from the CDB, whereas others are created by the AP for the specific application. Objects are transferred between the ADB and the CDB as needed. Objects reside permanently in the CDB, and are managed by and accessed through a Data Base Management System (DBMS).

The duties of the OM are divided among a number of specialized ingredients as shown in Figure 2. These ingredients are as follows.

(1) The Active Object Manager (AOM): it manages active objects in the ADB memory space. The AOM locates active objects in memory (i.e., given an object identifier it returns the memory address) by maintaining and consulting an active object dictionary. It also supports the swapping of objects into and out of memory.

(2) The Passive Object Manager (POM): it manages passive objects in the ADB disk space. The POM locates passive objects on disk (i.e., given an object identifier it returns the disk location). It also supports the swapping of objects between memory and disk. The POM also maintains and consults a passive object dictionary.

(3) The Transfer Object Manager (TOM): it manages transfer objects. These are Container objects which are used to transfer objects from one form to another, in particular from active to passive form and vice versa.

(4) The Object Factory Manager (OFM): it manages creation of new objects. The OFM sends messages to object classes to create new instances of themselves. It then supervises the population of these objects (i.e., the assigning of values to their attributes).

(5) The Object Extraction Manager (OEM): it manages extraction of object attributes from the CDB. Extraction involves creating a new design object in the ADB, then populating it with attribute values obtained from the corresponding object in the CDB.

(6) The Object Commitment Manager (OCM): it manages commitment of object attributes back to the CDB.

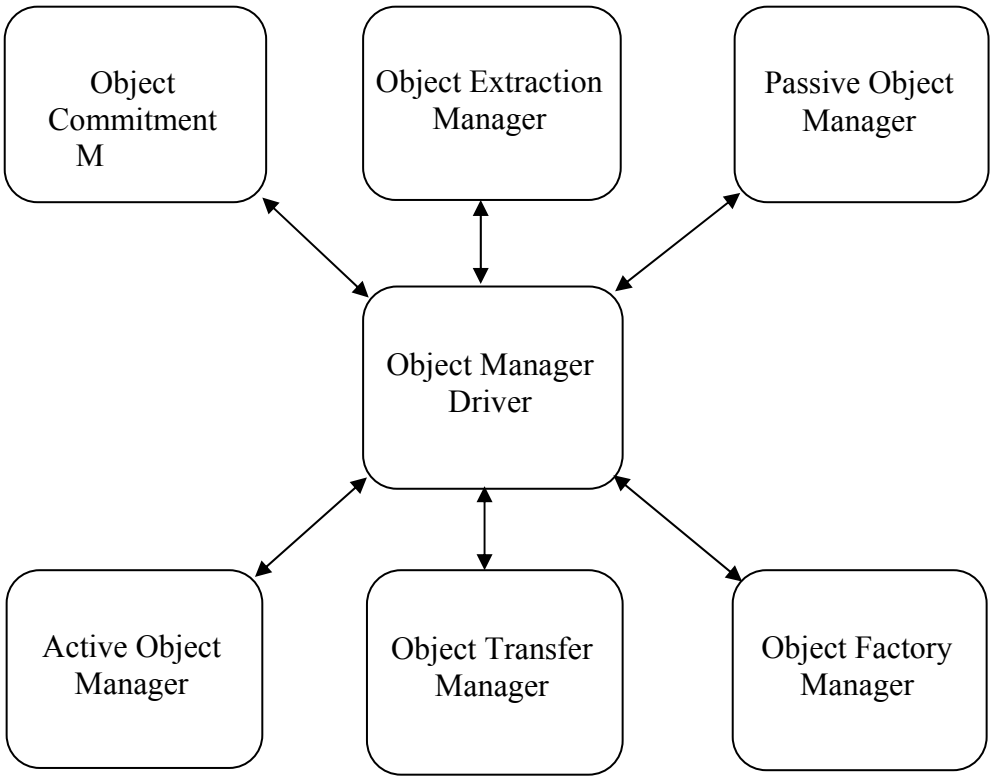


Figure 2: Elements of the Object Manager

## OPERATIONS OF THE OBJECT MANAGER

The OM performs its operations by commanding its ingredients to perform specialized operations. There are many operations that the Object Manager performs on the objects. As noted, the operations apply to objects as a whole not to individual attributes. The operations also apply to objects uniformly, irrespective of their classes. The operations supported by the OM can be classified into two main categories:

- Action operations which result in actions on objects.
- Query operations which provide information about objects.

There are several operations, belong to these two categories, that are supported by the Object Manager [Powell et al. 1989, Abdalla 1991]. Two sample operations, activate and de-activate are presented here for illustration.

- Activate: As previously indicated an Application Data Base (ADB) consists of objects in memory and on disk. An object in memory is “active”, and can be operated on directly by an Application Program (AP). Thus the Activate operation brings an object into memory, i.e., making it active.
- De-activate: An object is made persistent by saving it in the ADB disk space. Persistent objects which do not have active copies in memory are termed "passive" objects. Thus the Deactivate operation removes an active object from memory.

Details of these two operations are given the following sections.

## EXAMPLE 1: ACTIVATE OPERATION

For the Application Program (AP) or other objects to communicate with an object, the object must be active (i.e., in memory). For activating an object, the steps are as follows. These steps are illustrated using the Sequence Diagram of the UML as shown in Figure 3.

(1) The AP determines that a design object is to be activated. The AP sends a message to the OM, identifying the object and requesting that it be activated.

(2) The OM sends a message to the Passive Object Manager to locate the file record defining the object.

(3) The Object Manager sends a message to the Transfer Object Manager to create an empty Transfer object, and instructs the Transfer object to read the record into itself.

(4) The OM sends a message to the Object Factory Manager to create a new instance of the object to be activated.

(5) The Object Factory Manager sends a message to the object class to instance a new object of its type.

(6) The Transfer Object Manager sends a message to the created object. This message includes a reference to the Transfer object. The new object copies its instance variables from the Transfer object.

(7) The OM sends a message to the Active Object Manager to add the object to its active object dictionary.

(8) The OM returns control to the AP, with a response informing it about the success of the operation.

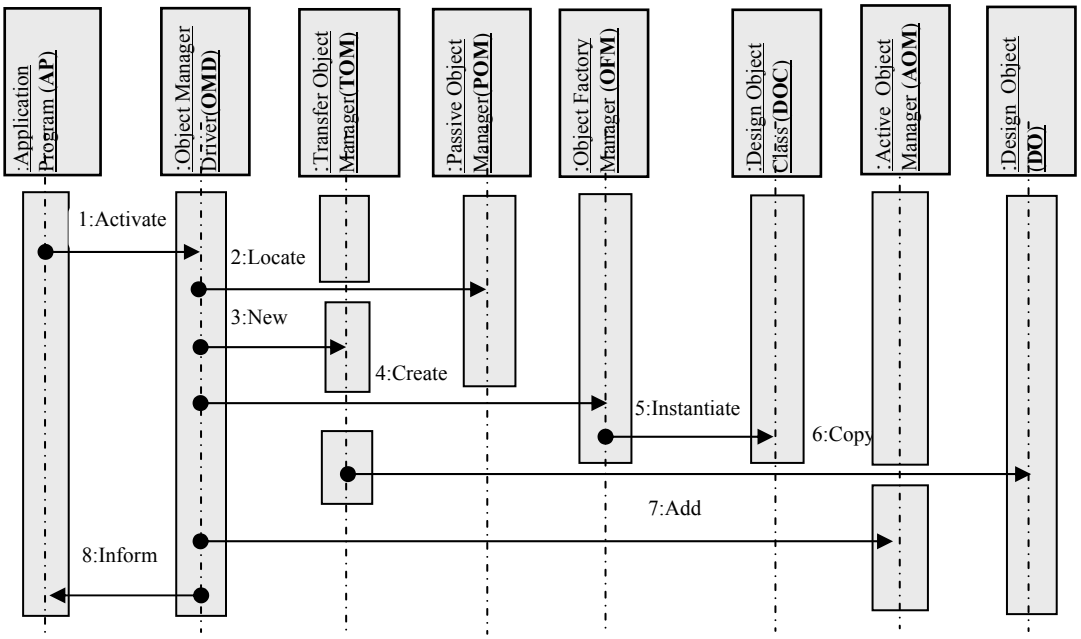


Figure 3: Sequence Diagram for the Activate operation

## EXAMPLE 2: DE-ACTIVATE OPERATION

For de-activating an object (i.e., removing it from memory) the steps are as shown in Figure 4 and outlined below. These steps are illustrated using the Sequence Diagram of the UML as shown in Figure 4.

(1) The AP sends a message to the OM requesting de-activation of a certain object.

(2) The OM sends a message to the Active Object Manager to locate the given object.

(3) The OM sends a message to the Active Object Manager to remove the object from the active object dictionary.

(4) The OM returns control to the AP, with a response informing it about the success of the operation.

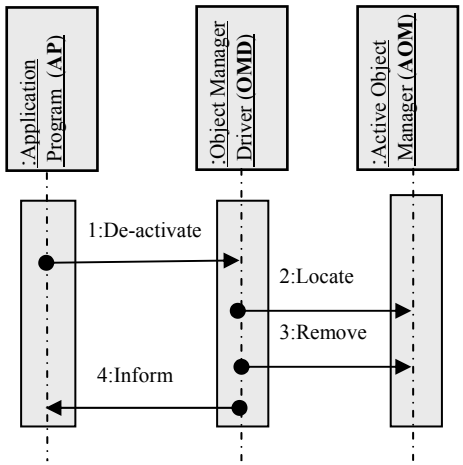


Figure 4: Sequence Diagram for the De-activate operation

## NOTE ON IMPLEMENTATION OF THE OBJECT MANAGER

The Object Manager class is a subclass of a general class called Manager class and it is implemented as an aggregation of its ingredient objects. Some of the ingredients of the Object Manager are themselves composed of other ingredients. Each ingredient of the OM (Object Factory Manager, Object Extractor Manager, Object Committer Manager, Active Object Manager, Passive Object Manager and Transfer Object Manager) supports a set of operations for object manipulation. Each object is identified by a user-defined name, termed its object identifier (OID). When an object is active it also has a memory address, which is used by the Application Program to locate the object in memory. When an object is saved on disk, it is assigned a disk location. The OID, memory address and disk location are all references to the object. The OID is fixed, whereas the memory address and disk location can vary. Implementing the OM class involves implementing these ingredient classes of the Object Manager and all the operations they support. The detailed design of these classes is a major task that is currently underway.

## SUMMARY AND CONCLUSIONS

In this paper an architecture for an Object Manager is presented, the features required of the Object Manager which manages objects in the Application Data Base have been identified and specific operations for managing objects in memory and on disk have been defined. Details of the steps of executing some of the operations of the Object Manager such as activate and de-activate have been presented using the Sequence Diagram of the Unified Modeling Language.

Research work on designing an architecture for the integrated engineering system based on the Object Management Group Reference Model Architecture together with the Object Manager, its operations and their implementations is currently underway. It is anticipated that such development will in integrating the structural engineering design process.

## REFERENCES

Abdalla, J A. (2004). "Elements of an Agent-based Mediative Communication Protocol for Design Objects." *Proceedings of the 10th International Conference of Computing in Civil and Building Engineering* (ICCCBE-X), Weimar, Germany, June 2-4, 2004. Paper 060.

Abdalla, J. A. and Eltayeb, T. (2000). "Towards a Version and Configuration Model for Structural Engineering Design Objects." *Proceedings of The 8th International Conference on Computing in Civil and Building Engineering* (ICCCBE-VIII), Stanford University, California, USA, August 14 – 17, 2000.

Abdalla, J. A. (1991). "An Object-Oriented Architecture and Concept for an Integrated Structural Engineering System." *Proceedings of the Second International Conference on Application of Artificial Intelligence Techniques to Civil and Structural Engineering*. Oxford, England, September 3rd-5th, 1991.

Abdalla, J. A., Phan, D. H. D., Howard, H. C. (1991). "Form, Function and Behavior for Structural Engineering Knowledge Representation." *Proceedings of the Second International Conference on the Application of Artificial Intelligence Techniques to Civil and Structural Engineering*. Oxford, England September 3rd-5th, 1991.

Ahn, J. H., Lee, S. W., Song, H. J. and Kim, H. J. (1998). "A survey of architectural features of contemporary object storage systems." *Journal of Systems Architecture*, 45(5): 363-386.

Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., Zdonik, S. (1989). "The object-oriented database system manifesto." *Proceedings of the First International Conference on Deductive and Object-Oriented Databases*, Kyoto, Japan, 1989, pp. 223–240.

Beeharry, A. and Delis, A. (1996). "On distributed persistent objects for interoperable data stores." *Informatics and Computer Science*, 91:1-32.

CADFC - The Committee for Advanced DBMS Function Corporate. (1990). "Third-generation database system manifesto." *SIGMOD Record*, 19(3): 31–44.

Catell, R.G.G., Barry, D.K., Berler, M. Eastman, J., Jordan, D., Russell, C., Schadow, O., Stanienda, T., Velez, F. (2000). *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann Publishers Inc., 2000.

Fenves, S. J., Flemming, U., Hendrickson C., Maher M. L., and Schmitt G.. (1988). "An Integrated Software Environment for Building Design and Construction." *The Fifth ASCE Conference on Computing in Civil Engineering: Microcomputers to Supercomputers*, Alexandria, VA, 1988; 21-32.

Darwen, H. and Date, C. J. (1995). "The third manifesto." *SIGMOD Record,* 24(1): 39–49.

King, R. (1986). *A Database Management System Based on an Object-Oriented Model. Expert Database Systems*. Edited by Kerschberg, L. pp. 443-468, Benjamin/Cummings, Menlo Park, CA.

Loomis, M.E.S. (1995). "Object Databases: The Essentials." *Addison-Wesley*, Reading, MA, 1995.

Mora, R., Bédard, C. and Rivard, H. (2004). "Interactive Conceptual Design of Building Structures." *10th International Conference on Computing in Civil and Building Engineering*, K. Beucke, B. Firmenich, D. Donath, R. Fruchter, and K. Roddis (Editors), Weimar, Germany, June 2-4, 2004, Paper No.: 185.

OMG (2006). The Object Management Group. http://www.omg.org.

Ozsu, M.T. Dayal, U. Valduriez, P. (1994). *Distributed Object Management*, Morgan Kaufmann, Los Altos, CA, 1994.

Powell, G. H., Abdalla, J. A. and Sause, R. (1989). "Object-Oriented Knowledge Representations : Cute things and Caveats." *Proceedings of The Sixth American Society of Civil Engineers Conference in Computing in Civil Engineering*, Atlanta, GA, USA September 1989.

Rivard, H. and Fenves, S. J. (2000). "A representation of conceptual design of buildings." *Journal of Computing in Civil Engineering*, 14(3): 151-159.

Siegel, J. (2006). Introduction to OMG UML by OMG's. http://www.uml.org/#Articles.

Stonebraker, M. (1988). "Future Trends in Data Base Systems." *Proceedings of the Forth International Conference on Data Engineering*, Los Angeles, Ca, USA.

Vinoski, S. (1997). "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments." *IEEE Communications Magazine*, 4(2): 46-55.