

-continued

```

public:
    CExampleObject (void);
    CExampleObject (CExampleObject *pOther);
    .
    .
private:
    CExampleMember cExampleMember;
};

```

The following code fragment illustrates that, when constructing the containing object, the address of the containing object must be passed to all member objects:

```

CExampleObject::CExampleObject (void)
    :cExampleMember (this)
{
    .
    .
CExampleObject::CExampleObject (CExampleObject *pOther)
    :CmyBaseClass ((cMyBaseClass *) pOther),
    cExampleMember (this, &pOther->cExampleMember)
{
    .
    .
}

```

Another special case occurs when an object derived from the CEngineObject class points to an object which is not derived from the CEngineObject class. This case is similar to that discussed above with member objects. In particular, if the non-CEngineObject must be changed, the CEngineObject must be checked for proper version. If a copy of the CEngineObject object must be made, a copy of the non-CEngineObject object should also be made. This can be handled by making the pointed-to object derive from the CMember class discussed above and constructing it with a pointer to the CEngineObject.

When the CEngineObject object is changed, a copy of the non-CEngineObject object must also be made. This is necessary because when one version of the CEngineObject object is deleted, there would be no way to tell if any other CEngineObject object versions were pointing to the non-CEngineObject, so there would no way to tell whether to delete the non-CEngineObject or not. However, this problem can be solved by reference counting the non-CEngineObject as discussed above.

FIG. 8 is an illustrative dialog box generated by the user interface software (214, FIG. 2) for displaying various versions of a document. The dialog box 800 has a scrolling listbox 802 which indicates each version present in a particular file. The listbox displays the version name, editors, last edit time and any remarks made by the editors. A further textbox area 804 is provided to enter additional remarks. A new version can be created by operating the "Create Version" button 806.

Operation of the create version button 806 causes the dialog box 900 to be displayed. This latter dialog box allows the user to enter a version name in editbox 902. This version name may be the name used internally to identify each version or the software may choose another name internally. A textbox area is also provided for additional comments.

Although only one embodiment of the invention has been disclosed, it will be apparent to those skilled in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the spirit and scope of the invention. These and other obvious modifications are intended to be covered by the appended claims.

What is claimed is:

1. Apparatus for constructing a document from a plurality of objects in a computer system having a memory and a non-volatile storage, a document object being accessed whenever an invocation of one of the document object's methods or the manipulation of one of the document object's attributes is required, the apparatus comprising:

a demand loader object residing in the memory and associated with each of the plurality of objects, each of the demand loader objects including program logic for loading the associated object into the memory from the storage when the associated object is accessed; and object pointer objects residing in the memory and connecting two of the plurality of objects, each object pointer object being a member of a first one of the two objects and having a pointer therein pointing to a demand loader object associated with a second one of the two objects.

2. Apparatus according to claim 1 wherein each demand loader object further includes a reference count attribute having a reference count stored therein which reference count indicates the number of object pointers which point to the each demand loader object.

3. Apparatus according to claim 2 wherein each object pointer object includes program logic for updating a reference count attribute in a demand loader object to which the object pointer points.

4. Apparatus according to claim 1 wherein each demand loader object includes an object ID for locating an associated object in the non-volatile storage.

5. Apparatus according to claim 1 wherein each demand loader object further includes a pointer to an associated object.

6. Apparatus for use with a computer system having a memory and a non-volatile storage, the apparatus creating a new version of a document, which is composed of a plurality of objects which are interconnected by object pointers, a document object being accessed whenever an invocation of one of the document object's methods or the manipulation of one of the document object's attributes is required, in response to a user request to change one of the plurality of objects, and comprising:

a demand loader object residing in the memory and associated with the one object, the demand loader object including program logic for loading the one object into the memory from the storage when the one object is accessed;

means in the demand loader object for creating a list of object versions for the one object, the list including a version ID and a pointer to a copy of the one object for each object version; and

means responsive to the request by making a copy of the one object and inserting a version ID and a pointer to the one object copy into the list.

7. Apparatus according to claim 6 wherein the new document version has a document version ID and the copying means is responsive to the request for inserting the document version ID into the list.

8. Apparatus according to claim 7 wherein the copying means is responsive to the request for checking the list to determine whether the document version ID is in the list.

9. Apparatus according to claim 8 wherein the means responsive to the request makes a copy of the one object and inserts the document version ID and a pointer to the one object copy into the list if the document version ID is not in the list.

10. Apparatus according to claim 7 wherein the demand loader object comprises an object ID for retrieving the one object from the storage into the memory.