

such a file is found, it must be the representation of this object.

The distributed environment introduces two types of delays in access to objects represented by files: (1) If the file is on a remote machine, it has to be found. (2) Once found, it has to be retrieved.

Since retrieval time is determined by the speed of file transfer across the network and the load on the file server, the modeller tries to avoid retrieving files when the information it wants about a file can be computed once and stored in a database. For example, the type of an object, which is the information needed to compute its compilation dependencies, is small compared to the object itself. The object type table stores the types of all objects of current interest; a source object in the table does not have to be examined, or even retrieved, unless it actually needs to be recompiled.

In cases where the file must be retrieved, determining which machine or computer and directory has a copy of the version desired can be very time consuming. Even when a file location hint is present and correct, it may still be necessary to determine several versions of the file to find the one with the right creation date. The modeller minimizes these problems by keeping another cache, which maps an object name into the full path name in the distributed file system of a file which represents the object. This cache is the Version Map, discussed previously. Note that both source objects, whose unique identifiers are creation dates, and binary objects, whose unique identifiers are version stamps, appear in the version map. The full pathname includes the version number of the file, which is the number after the "P". This version number makes the file name unique in the file system so that a single reference is sufficient to obtain the file.

Thus, the modeller's strategy for minimizing the cost of referencing objects has three paths:

(1) Consult the object type table or the projection table, in the hope that the information needed about the object is recorded there. If it is, the object need not be referenced at all.

(2) Next, consult the version map. If the object is there, a single reference to the file system is usually sufficient to obtain it.

(3) If there is no entry for the object in the version map, or if there is an entry but the file it mentions does not exist, or does not actually represent the object, then use the file location hint to identify a directory, and enumerate all the versions of the file to find one which does represent the object. If this search is successful, make a new entry in the version map so that the search need not be repeated.

Like the other caches, a version map is maintained on each computer or machine and in each .modelBcd object. A .modelBcd version map has an entry for each object mentioned in the model. A machine version map has an entry for each object which has been referenced recently on that machine. In addition, commonly referenced objects of the software system are added to the machine version map as part of each release.

Since the version maps are hints, a version map entry for an object does not guarantee that the file is actually present on the file server. Therefore, each successful probe to the version map delays the discovery of a missing file. For example, the fact that source file does not exist may not be discovered until the compilation phase, when the modeller tries to compile it. This means that the modeller must be robust in the face of such

errors. The release process, however, guarantees that the files are present as long as the release remains active.

While the system modeller has been described in conjunction with specific embodiments, it is evident that alternatives, modifications and variations will be apparent to those skilled in this art in light of the foregoing description. Accordingly, it is intended to embrace all such alternatives, modifications and variations as fall within the spirit and scope of the appended claims.

What is claimed is:

1. A software version management system for automatically collecting and recompiling updated versions of component software objects comprising a software program for operation on a plurality of personal computers coupled together in a distributed software environment via a local area network and wherein said objects include the source and binary files for various of said software program and are stored in various different local and remote storage means through said environment, said component software objects being periodically updated via environment editing means by various users at said personal computers and stored in designated storage means, said system including:

models comprising system objects,

each of said models representative of the source versions of a particular component software object,

each of said models containing object pointers including a unique name of the object, a unique identifier descriptive of the chronological updating of its current version, information as to an object's dependencies on other objects and a pathname representative of the residence storage means of the object, means in said editing means to notify said management system when any one of said objects is being edited by a user,

means in said management system in response to notification of object editing to track said edited objects and alter their respective models to the current version thereof,

said management system upon command adapted to retrieve and recompile said source files corresponding to said altered models and load the binary files of said altered component software objects and their dependent objects into said computers.

2. The software version management system of claim 1 wherein said system includes accelerator means to cache said object pointers in said models that never change to thereby avoid further retrieving of said objects to parse and to discern said object pointers.

3. The software version management system of claim 2 wherein said accelerator means for said models includes

an object type table for caching the unique name of the object and its object type to enhance the analysis of a model by said management system,

a projection table for caching the unique name of the source object, names of object parameters, compiler switches and compiler version to enhance the translation of objects into derived objects, and a version map for caching said pathname.

4. A method for automatically collecting updated versions of component software modules together which comprise a software program operative on a plurality of computers, said computers coupled together in a distributed software environment via a local area network and wherein said modules are stored in various different local and remote storage means