

mation as the name and version of the distribution unit, a list of the components and their location on the computer, and the source of the distribution unit. Additional fields in the code store data structure can also contain a component version, a component data type, and a digital signature if one was affixed to the distribution unit.

[0014] During the installation, the package manager can optionally scan the code store data structure to determine if a component to be installed already exists on the computer and updates the code store data structure with the location of the later version of the component.

[0015] When a user requests execution of software, the package manager uses the code store data structure to locate the appropriate components for the operating system to use. When the user requests the uninstallation of a software package, the package manager deletes the appropriate components from the computer and updates the code store data structure accordingly.

[0016] The manifest file and distribution unit optionally are combined into a distribution unit file.

[0017] The manifest file format is common across all types of code and operating systems and easily extended to embrace new code types as they arise. The manifest file and distribution unit can be stored on all types of media from traditional magnetic and optical disks to networked servers. The distribution units for dependencies do not have to reside on the same type of media as the distribution unit or the manifest file that refers to the dependency. More than one distribution unit can be resident in a distribution unit file and a distribution unit file can contain a mixture of distribution units containing different code types.

[0018] Thus, the software package manager, the manifest file, the distribution unit and the code store data structure of the present invention solve the problems with existing distribution mechanisms. The manifest file is not particular to a particular code type or operating system and allows for the specification of nested software dependencies. Because the manifest file contains the location of the distribution units for any dependencies, the software package manager can acquire and install the dependencies without requiring manual intervention by the user. Different types of distribution units can be mixed in a distribution unit file so that a single mechanism is used to acquire and install all types of code.

[0019] The code store data structure maintained by the software package manager contains information about the installed software such as version and installation location, and is used to resolve version discrepancies among software programs that share components. The code store data structure is used by the package manager to locate necessary component when the software is executed so that a component stored in one directory can be readily shared by software programs with components in different directories. Finally, the code store data structure eases the uninstallation process by centralizing all the information about installed components.

[0020] The present invention describes systems, clients, servers, methods, and computer-readable media of varying scope. In addition to the aspects and advantages of the present invention described in this summary, further aspects

and advantages of the invention will become apparent by reference to the drawings and by reading the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] FIG. 1 shows a diagram of the hardware and operating environment in conjunction with which embodiments of the invention may be practiced;

[0022] FIGS. 2A, 2B and 2C are diagrams illustrating a system-level overview of an exemplary embodiment of a package manager of the invention;

[0023] FIGS. 3A, 3B, 3C and 3D are flowchart of methods to be performed by a client according to an exemplary embodiment of the package manager of the invention; and

[0024] FIG. 4 is a diagram of an exemplary embodiment of an entry in a code store data structure suitable for use by the methods shown in FIGS. 3A, 3B and 3C.

DETAILED DESCRIPTION OF THE INVENTION

[0025] In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

[0026] The detailed description is divided into five sections. In the first section, the hardware and the operating environment in conjunction with which embodiments of the invention may be practiced are described. In the second section, a system level overview of the invention is presented. In the third section, methods for an exemplary embodiment of the invention are provided. In the fourth section, a particular Open Software Description implementation of the invention is described. Finally, in the fifth section, a conclusion of the detailed description is provided.

Hardware and Operating Environment

[0027] FIG. 1 is a diagram of the hardware and operating environment in conjunction with which embodiments of the invention may be practiced. The description of FIG. 1 is intended to provide a brief, general description of suitable computer hardware and a suitable computing environment in conjunction with which the invention may be implemented. Although not required, the invention is described in the general context of computer-executable instructions, such as program modules, being executed by a computer, such as a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types.

[0028] Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer