

Methods of an Exemplary Embodiment of the
Invention

[0048] In the previous section, a system level overview of the operation of an exemplary embodiment of the invention was described. In this section, the particular methods performed by a client or local computer of such an exemplary embodiment are described by reference to a series of flowcharts. The methods to be performed by the client computer constitute computer programs made up of computer-executable instructions. Describing the methods by reference to a flowchart enables one skilled in the art to develop such programs including such instructions to carry out the methods on suitable computerized clients (the processor of the clients executing the instructions from computer-readable media).

[0049] The software package manager of the present invention is described as providing three major functions to the runtime environment of the local computer on which it runs as illustrated in FIGS. 3A-3D. It manages the installation of software packages, it locates necessary components when software is executed, and it supports the uninstallation of software. The package manager uses the installed package database, also called a "code store" data structure, to track components of software packages that have been installed on the local computer. One of skill in the art will, upon reading the following description of the code store data structure, recognize that any type of organized data structure, including various type of data bases, is suitable for use with the package manager.

[0050] As in the exemplary embodiment described in the previous section, the manifest file contains dependency entries specifying locations of distribution units containing required software components. The distribution unit file is suitable for distributing software packages on traditional media, such as CD-ROM or floppy disk, as well as over a wide area network, such as the Internet. The package manager extracts the manifest file and the distribution unit from the distribution unit file. In an alternate embodiment, the distribution unit and the manifest file can be stored separately on a network and the manifest file contains the network location of its corresponding distribution unit.

[0051] Installation

[0052] Referring first to FIGS. 3A and 3B, a flowchart of methods to be performed by a client according to an exemplary embodiment of the invention when installing new software is shown. This method is inclusive of the steps or acts required to be taken by the package manager.

[0053] When the distribution unit file for a software package is loaded onto a computer for installation, the software package manager running in the computer acquires the manifest file for processing (step 301). In an embodiment in which the manifest file is distributed in a distribution unit file, the package manager acquires the distribution unit file and extracts the manifest file from the distribution unit file. The package manager checks the name and version of the software package contained in the manifest file against the code store data structure to determine if the software package has already been installed (step 303). If so, the package manager exits (step 321).

[0054] If the software package is not installed, the package manager checks the manifest file to determine if the software

package requires the installation of other software components (dependencies) not supplied as part of the distribution file unit (step 305) before installing the software package from the distribution unit. Such is frequently the case when a software package is written in a common programming language such as Java or C++ which depend on object class or language libraries being present.

[0055] If there are dependencies, the package manager checks the code store data structure to determine if the dependencies are installed on the local computer (step 327). If not, the package manager uses information stored in the manifest file about the dependencies to locate a source for the dependencies. The package manager then acquires a dependency from the source specified in the manifest file (step 329). In one embodiment, the source is a remote server designated by a uniform resource locator (URL) path name. In an alternate embodiment, the source is a server on a local area network and the path name is a network drive.

[0056] After acquiring the dependency from the source, the package manager installs the dependent software components on the local computer. The installation of the dependent components is identical to the installation process for the original software package which will be described in detail below in conjunction with steps 307 through 325. Because each dependency can itself include dependencies, the package manager will install all the nested dependencies prior to finishing the installation of the original software package.

[0057] Once all the dependencies are installed, the package manager determines if a directory for the software package exists (step 307) and creates one if it does not (step 309). The package manager assigns the directory a unique name that has a high probability of being different for every computer on which the software package is installed. In one embodiment, the unique name is generated using a standard hashing algorithm having the software package name as input.

[0058] The package manager extracts the components for the software package from the distribution unit file into the directory (step 311). In one embodiment, the components are gathered into an uncompressed "zip" formatted data file which contains a directory of the files within it. Other embodiments which use similar file structures to group the components together with a directory structure will be readily apparent to one skilled in the art. The package manager then invokes the installation facility provided by the operating system to install the components (step 313).

[0059] When the installation is successful (step 315), the package manager updates the code store data structure with the information contained in the manifest file (step 317 and FIG. 3B).

[0060] The package manager creates an entry in the code store data structure for the software package that includes the name of the software package, the version number, and the name of the directory (step 331). The names of the components in the software package are stored in the corresponding software package entry in code store data structure (step 333).

[0061] In one alternate embodiment shown in FIG. 3B, as part of the update process for the code store data structure, the package manager scans the code store data structure to