

cation” in a World Wide Web browser, such as Microsoft Internet Explorer 4. A plug-in application is often employed to provide additional capabilities, such as multimedia or interactive controls, to browsers. One type of control application is that written to conform with Microsoft’s ActiveX specifications. The plug-ins are usually written in object-oriented languages such as C++ or Java and are typically used on Web pages. The user may be prompted to download the plug-in or it may be automatically downloaded when needed.

[0038] Referring to FIG. 2A, Fred’s Software Company wants to distribute the CoolestApp over the Internet from Fred’s Software Company’s Web server 201 to a user’s computer 203. Fred’s Software Company logically groups the components for the CoolestApp together into a “distribution unit” 209. The components can include platform-specific compiled binary files such as dynamic linking library (.dll) files used by the Microsoft Windows family of operating systems, Java bytecode (.class) files, or files that contain optional installation instructions for how to use certain components contained in the distribution unit, for example, ActiveX controls may need to be registered before use. The distribution unit 209 can be a separate file or can be a portion of a “distribution unit file” 205 as explained below.

[0039] Fred’s Software Company also creates a “manifest” file 207 describing the CoolestApp. The CoolestApp manifest file 207 contains information about CoolestApp, including the name of the CoolestApp distribution unit 209, the version number of the software package (all components in the distribution unit 209 have the same version number in this embodiment), and the operating systems under which the CoolestApp executes. Fred’s Software Company bundles the CoolestApp distribution unit 209 and manifest file 207 into a distribution unit file 205 for storage on the server 201.

[0040] The names of other files in the distribution unit file 205, such as a text file containing licensing information or a “readme” file containing special instructions for the software package, are listed in the manifest file 207. The manifest file 207 also contains entries for software that is required to run CoolestApp but which is not included in the distribution unit file 205. Such required software represent “dependencies” and frequently include such items as language libraries and common object class libraries. A dependency can also be another software package. The manifest file 207 provides the ability to describe the software dependencies in a recursive tree format, also known as a “directed graph.”

[0041] In the present example, CoolestApp is an enhanced version of a software program named “CoolApp” previously distributed by Fred’s Software Company. Rather than completely rewriting CoolestApp, Fred’s Software Company used the CoolApp components as a base and created additional components for the new features in CoolestApp. In the interest of minimizing download time, Fred’s Software Company does not include the original components for CoolApp in the CoolestApp distribution unit 205. Instead Fred’s Software Company inserts a dependency entry in the manifest file 205 which directs a user’s browser to the location on Fred’s Software Company server 201 holding the distribution unit file 215 for CoolApp as illustrated in FIG. 2B.

[0042] The browser begins the installation of the CoolestApp software package to the local computer by downloading the CoolestApp distribution unit file 205. A software package manager 211 running in the underlying operating system on the user’s computer 203 extracts the manifest file 207 from the distribution unit file 209 and accesses an installed package database 213 to determine that Fred’s Software Company’s CoolestApp is not already installed. The dependency entry in CoolestApp manifest file 207 alerts the package manager 211 that the CoolestApp depends on Fred’s Software Company’s CoolApp. The package manager 211 determines that CoolApp has not been previously installed and directs the browser to download the CoolApp distribution unit file 215 from the server location specified in the dependency entry.

[0043] Once the CoolApp distribution unit file 215 has been downloaded to the user’s computer 203, the package manager extracts the CoolApp manifest file 217 and determines that CoolApp does not have any dependencies. The package manager 211 creates a private directory 221 for Fred’s Software Company applications, named FSC, extracts the CoolApp components from the distribution unit 219 into the FSC directory, and calls the underlying operating system installation facility to install the CoolApp components. The package manager 211 registers the CoolApp components in the installed package database 213 when the installation is successful.

[0044] Referring to FIG. 2C, the package manager 213 extracts the CoolestApp components from the CoolestApp distribution unit 209 to the FSC directory 221, calls the installation facility, and registers the CoolestApp components in the installed package database 213. The browser now can run the CoolestApp helper application from the FSC directory 221.

[0045] If the user downloads additional Fred’s Software Company applications that depend upon the components in either CoolApp or CoolestApp, the package manager 211 will use the already installed components to satisfy any dependencies that reference installed software package unless the additional applications require versions later than that installed.

[0046] If after running the CoolestApp helper application, the user decides that CoolestApp is not needed, the user employs the underlying operating systems uninstall facility to uninstall CoolestApp. The uninstall facility invokes the package manager 211 which determines if the CoolApp and CoolestApp components are being used by other applications and deletes the software packages from the FSC directory 221 if not. The package manager 211 also deletes the package entries from the installed package database 213 when the packages have been deleted from the FSC directory 221.

[0047] The system level overview of the operation of an exemplary embodiment of the invention has been described in this section of the detailed description. The package manager and its supporting files have been described in relation to installing a software package having a single dependency. While the invention is not limited to any particular distribution media, for sake of clarity a simplified version of Internet software distribution has been described.