

section. The "CODEBASE" element in <OBJECT> and the "useslibrarycodebase" tag in <APPLET> can point to the manifest file or to the distribution unit file.

[0091] In a third exemplary embodiment of the invention for purposes of this section, a distribution unit file is used to automatically distribute software from Fred's Software Company's server to the user's computer. This automatic distribution across a network employs "channels" to which the user subscribes to automatically "push" software components through a client agent such as a browser. The channel is described using a Channel Definition Format (CDF) which is also based on XML. A CDF file uses the OSD elements to inform a CDF-aware client agent as to what software components should be downloaded and installed.

```
<CHANNEL HREF="http://www.fsc.com.intropage.htm">
  <SELF="http://www.fsc.com/software.cdf" />
  <TITLE>A Software Distribution Channel</TITLE>
  <SOFTPKG
    HREF="http://www.fsc.com/aboutsoftware.htm"
    AUTOINSTALL="yes"
    NAME="{D27CDB6E-AE6D-11CF-96B8-444553540000}"
    VERSION="1,0,0,0">
  <IMPLEMENTATION>
    <OS VALUE="WinNT"><OSVERSION VALUE="4,0,0,0">
      </OS>
    <OS VALUE="Win95">
    <PROCESSOR VALUE="x86" />
    <CODEBASE HREF="http://www.fsc.com/coolestapp.cab" />
  </IMPLEMENTATION>
</SOFTPKG>
</CHANNEL>
```

[0092] This section has described a particular implementation of the package manager which is directed to install software by OSD elements embedded in an XML document. The processing of a manifest file described in previous section when written as XML document is described. In addition, alternate embodiments in which a separate XML document resides on a Web page to direct a browser to invoke the package manager to install a software package is also described in this section.

Conclusion

[0093] A software package manager has been described which manages the installation, execution and uninstallation of software packages acquired through various media. The software manager uses a manifest file, a distribution unit, and a code store data structure to accomplish its functions. Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention.

[0094] For example, those of ordinary skill within the art will appreciate that the file and data structures described herein can be easily adapted to future distribution media. Furthermore, those of ordinary skill within the art will appreciate that future extensible languages which are platform and operating system independent can be used to direct the software package managers actions.

[0095] The terminology used in this application with respect to is meant to include all hardware and software platforms. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.

We claim:

1. A computerized method for managing software packages on a computer comprising the steps of:

acquiring a manifest file that describes a distribution unit for a software package;

resolving software dependencies for the software package as specified by the manifest file;

acquiring the distribution unit for the software package;

extracting components in the software package from the distribution unit into a directory on the computer;

causing the installation of the software package on the computer; and

updating a code store data structure on the computer with information from the manifest file pertaining to the software package.

2. The method of claim 1, wherein the step of resolving the dependencies comprises the steps of:

acquiring, for each of the software dependencies in turn, a dependency manifest file that describes a dependency distribution unit for a dependency software package;

resolving software dependencies for the dependency software package as specified by the dependency manifest file;

acquiring the dependency distribution unit for the dependency software package;

extracting components in the dependency software package from the dependency distribution unit into a directory on the computer;

causing the installation of the dependency software package on the computer; and

updating a code store data structure on the computer with information from the dependency manifest file pertaining to the dependency software package.

3. The method of claim 1, wherein the software dependencies are nested so that the step of resolving the dependencies is processed recursively.

4. The method of claim 1 further comprising the step of:

locating the directory containing the components for the software package using the code store data structure when the execution of software in the software package is requested.

5. The method of claim 1 further comprising the steps of:

modifying the code store data structure to reflect the removal of the software package when the uninstallation of the software package is requested; and

deleting each component for the software package from the directory when the code store data structure indicates no other software package uses the component.

6. The method of claim 1, wherein the manifest file specifies the directory into which the components of the software package are extracted.