

[0080] Additionally, the manifest file can specify an XML tag, “namespace,” that causes the package manager to isolate software packages from one another even if the same component is used by multiple packages:

[0081] <JAVA>

[0082] . . .

[0083] <NameSpace>Fred’s Software Company</NameSpace>

[0084] Thus the use of namespaces avoids version mismatches among software packages.

[0085] A namespace is analogous to a directory in a file system, such as implemented in the Windows family of operating systems, in that installing applications in different directories provides isolation for the applications. Previously a namespace was global to all applications installed on a system so that all files and components in the namespace were accessible by the applications. The global nature of previous namespaces presents difficulties in naming files and components because an application programmer has to avoid common names to prevent potential conflicts with identically named files and components for another application installed in the same namespace. Installing the second of the two applications would likely cause one or both applications to fail, just as placing all files for all applications installed on a computer into a single directory frequently causes conflicts between the applications.

[0086] In the present invention, the presence of a namespace XML tag in the manifest file causes the package manager to associate the files and components of the corresponding application in the code store data structure with the unique namespace specified in the tag. When an application is executed, the package manager passes the associated namespace name to the computer’s runtime environment so that any files and components installed in that namespace are visible to the application while files and components installed in other namespaces are not. Using the example XML tag above, Fred’s CoolestApp is associated with a namespace called “Fred’s Software Company,” would execute in the “Fred’s Software Company” namespace, and have access to any files or components installed in the “Fred’s Software Company” namespace. Similarly, an XML tag for Bob’s identically named “CoolestApp” would specify “Bob’s Software Company” as the namespace, execute in the “Bob’s Software Company” namespace, and have access to any files or components installed in the “Bob’s Software Company” namespace. Neither Bob’s CoolestApp nor Fred’s CoolestApp can access a common component or file installed in the other’s namespace. Therefore, because of the isolation that namespaces provide, both Fred and Bob are assured their applications will function correctly even though identically named and having common components or files, and that the applications will continue to function correctly irregardless of the number of CoolestApps using the same components or file which may be installed on the computer.

[0087] Continuing with the distribution of Fred’s Software Company’s CoolestApp, the manifest file in a first exemplary embodiment in this section is stored separately from the distribution unit at <http://www.fsc.comncoolestapp.osd>. The corresponding distribution unit is stored in a cabinet file at <http://www.fsc.org/coolestapp.cab>. The CoolestApp’s

dependency on the components of the earlier CoolApp is indicated with a “DEPENDENCY” tag and refers the package manager to the CoolApp manifest file at <http://www.fsc.org/coolapp.osd> (not shown). The CoolApp manifest file directs the package manager to the location of the distribution unit for the CoolApp.

```
<SOFTPKG NAME="com.fsc.www.coolestapp" VERSION="1,0,0,0">
  <TITLE>CoolestApp</TITLE>
  <ABSTRACT>CoolestApp by Fred’s Software Company
  </ABSTRACT>
  <LICENSE HREF="http://www.fsc.com/coolestapp/license.html" />
  <!--FSC’s CoolestApp is implemented in native code -->
  <IMPLEMENTATION>
    <OS VALUE="WinNT"><OSVERSION VALUE="4,0,0,0"/></OS>
    <OS VALUE="Win95"/>
    <PROCESSOR VALUE="x86" />
    <LANGUAGE VALUE="en" />
    <CODEBASE HREF="http://www.fsc.org/coolestapp.cab" />
    <!--CoolestApp needs CoolerApp -->
    <DEPENDENCY>
      <CODEBASE HREF="http://www.fsc.org/coolapp.osd" />
    </DEPENDENCY>
  </IMPLEMENTATION>
</SOFTPKG>
```

[0088] Had the CoolApp manifest file been stored in a cabinet distribution unit file along with the CoolApp components, the location of the distribution unit file would have been <http://www.fsc.org/coolapp.cab>.

[0089] In a second exemplary embodiment of the invention for purposes of this section, components contained in a distribution unit file are caused to be installed by OSD tags embedded on a Web page. If Fred’s Software Company’s Web page requires additional software to be downloaded and installed for viewing the page, FSC can use the OSD vocabulary within HTML commands to have the user’s browser download the necessary components as shown in the two examples below.

```
<OBJECT CLASSID=
  "clsid:9DBAFCCF-592F-101B-85CE-00608CEC297B"
  VERSION="1,0,0,0"
  CODEBASE="http://www.fsc.com/coolestapp.osd"
  HEIGHT=100 WIDTH=200 >
</OBJECT>
-or-
<APPLET code=myapplet.class id=coolestapp width=320 height=240>
  <PARAM NAME=useslibrary VALUE="coolestapp">
  <PARAM NAME=useslibraryversion VALUE="1,0,0,0">
  <PARAM NAME=useslibrarycodebase VALUE=
    "http://www.fsc.com/coolestapp.osd
">
</APPLET>
```

[0090] The HTML <OBJECT> or <APPLET> tag informs an OSD-aware client browser, such as Microsoft Explorer 4, that there is additional software required to view the Web page. The browser invokes the package manager to execute the software package if it is already installed or to install it if not. If not already installed, the package manager instructs the browser to download the distribution file unit and proceeds with the installation as described in the previous