

**SOFTWARE PACKAGE MANAGEMENT****RELATED APPLICATION**

This application is related to U.S. patent application Ser. No. 08/764040, titled AUTOMATIC SOFTWARE DOWNLOADING FROM A COMPUTER NETWORK, filed on Dec. 12, 1996, and assigned to the assignee of the present application.

**COPYRIGHT NOTICE/PERMISSION**

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever. The following notice applies to the software and data as described below and in the drawing hereto: Copyright© 1997, Microsoft Corporation, All Rights Reserved.

**FIELD OF THE INVENTION**

This invention relates generally to software distribution, and more particularly to the management of software packages after distribution.

**BACKGROUND OF THE INVENTION**

Historically, the primary medium for software distribution has been either the traditional floppy disk or the more recent compact disc (CD-ROM). However, more and more individuals are acquiring software by downloading it from remote server computers connected to the client computers through the Internet. Additionally, companies and organizations are distributing software to their users across their local area networks. The physical medium is the network cable itself and the supporting communication hardware, a fixed cost associated with the establishment of the network. Therefore, distributing and installing software over an existing network bypasses the cost overhead of producing CDs or floppy disks.

In addition, using the network as the distribution medium profoundly reduces the software's total cost of ownership to an extent that cannot be achieved by CDs or floppies even when the media cost almost nothing to manufacture. Software distribution via CDs and floppies obey the "pull" paradigm, where every action is user-initiated. Distribution over the network has the ability to apply a "push" paradigm which provides three main benefits.

First, the installation is "hands-free" in that the user does not have to manually install the software. Second, the software can be easily and timely upgraded from a designated location because the burden of upgrading is borne by the software itself. Third, because different types of computer hardware and operating systems can connect to a common network, software distributed over the network can be made to work across platforms or intelligent so that only the correct version of platform-specific software is pushed down to the user.

However, current methods of software distribution over a network do not fully exploit the benefits. Existing distribution of platform-specific, or "native code," software relies on installation file formats that are hard to create, not extensible, and specific to a particular operating system. Although most current software is written in modules, there is no current mechanism that handles the situation where one

component in a software program requires the presence of another to operate. If a user downloads software from a Web page, the user may discover that the program requires an external library which necessitates another network session to download, assuming the user can find the right location, and then the user must manually install the library before installing the software.

Software programs written in the popular platform-independent Java language require that the Java classes be "packaged" for distribution but the package does not contain persistent information so once Java software is installed on a client computer, all information about it is lost. It is impossible to tell what the version number is, where it came from, or whom the author is. Additionally, the current network distribution methods make it difficult to digitally sign a Java package for security purposes.

More problems arise when a user wants to execute an application which depends on both native code components and Java components since the distribution methods are completely different. Finally, once the software is downloaded and successfully installed on the client computer, no mechanism exists to track all of the components so that older versions can be easily superceded when newer version are available or that all the related components can be readily uninstalled when necessary.

Therefore, there is a need for a software distribution and tracking mechanism that handles cross-platform software, specifies the component dependencies, and is applicable to both the older distribution media as well as to the network distribution paradigm.

**SUMMARY OF THE INVENTION**

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

A software package manager uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. For installation, the package manager acquires the manifest file and parses it to learn if the software package depends on any additional components. The package manager resolves any dependencies by acquiring a distribution unit containing the needed component and installs the dependency's distribution unit as described below. Because dependencies can be nested within dependencies, the package manager recursively processes all the dependencies before finishing the installation of the software package that depends upon the additional components.

The software package manager acquires the distribution unit and extracts the components in the distribution unit into a directory on the computer. The package manager causes the operating system of the computer to install the software. The package manager then updates a code store data structure with information in the manifest file. The fields in the code store data structure contains such information 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.

During the installation, the package manager can optionally scan the code store data structure to determine if a