

7. The method of claim 1, wherein the manifest file is Open Software Description format.

8. The method of claim 1, wherein the distribution unit is a compressed file.

9. The method of claim 1, wherein the manifest file and the distribution unit comprise a distribution unit file and the manifest file and the distribution unit are acquired by extracting each from the distribution unit file.

10. The method of claim 9, wherein the distribution unit file comprises a dependency distribution unit.

11. The method of claim 9, wherein the distribution unit file comprises a plurality of distribution units.

12. The method of claim 9, wherein the distribution unit file is a compressed file.

13. The method of claim 1, wherein the manifest file and the distribution unit are distributed on a computer-readable medium.

14. The method of claim 1, wherein the components in the software package and the dependencies comprise native code components.

15. The method of claim 1, wherein the components in the software package and the dependencies comprise Java classes.

16. The method of claim 1, wherein the components in the software package and the dependencies comprise a mixture of native code components and Java classes.

17. The method of claim 1, wherein the step of extracting the components comprises the steps of:

copying the components from the distribution unit into a file structure having a file directory;

updating the file directory with information about each component in the file; and

storing the file structure into the directory on the computer.

18. The method of claim 1, wherein the code store data structure resides in an existing system data structure and the step of updating the code store data structure comprises the step of interfacing with an existing system data structure manager.

19. The method of claim 1, further comprising the steps of:

determining when a component in the software package already exists on the computer;

comparing the versions of the components; and

updating the code store data structure with the location for the component which is the later version.

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

determining if the manifest file specifies a that the software package cannot shared components with software packages previously installed on the computer; and

isolating the components in the software package by associating the components in the code store data structure with a unique execution space specified by the manifest file.

21. A computer-readable medium having computer-executable instructions to a cause a client computer to perform a method comprising:

downloading a manifest file and a distribution unit for a software package described by the manifest file from a server computer;

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

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

causing the client computer to install the software package; and

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

22. The method of claim 21, wherein the step of resolving the dependencies comprises the steps of:

acquiring, from a server computer 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;

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

causing the client computer to install the dependency software package; and

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

23. The method of claim 21, wherein the software dependencies are nested so that the client computer processes the step of resolving the software dependencies recursively.

24. The method of claim 21, wherein the code store data structure resides in an existing system data structure managed by the operating system in the client computer and the step of updating the code store data structure comprises the step of interfacing with the operating system.

25. A computer system comprising:

a plurality of clients; and

a server, communicatively coupled to each of the clients through a transport medium and having local storage holding a manifest file and a distribution unit,

wherein each client acquires the manifest file and the distribution unit through the transport medium.

26. The computer system of claim 25, wherein the manifest file and the distribution unit comprise a distribution unit file, and each client acquires the distribution unit file through the transport medium.

27. A computer-readable medium having stored thereon a code store data structure comprising:

a name field containing data representing a name of a distribution unit for a software package installed on a computer;

a unit version field containing data representing a version for the distribution unit represented by the first field;

a component field containing data representing a list of components in the distribution unit represented by the first field;