Title of the Invention: Application installation
Abstract Title: Installing applications at selected runtime instances

Disclosed is a method of installing an application at a selected runtime instance. The method starts by receiving a request to install an application, then it accesses data defining the shared resources required by the application, and accesses data defining shared resources already installed at existing runtime instances. Then, an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application is selected. Finally the application is installed at the selected runtime instance. The method may maintain a table storing identifiers of the shared resources installed at existing runtime instances. The method may also determine the free capacity available at the existing runtime instances and select the instance with sufficient capacity to install the application. The step of installing the application may include removing from the application those shared resources already present and installing the residual components of the application at the selected runtime instance.
Fig. 6

CLOUD CONTROLLER

501: DETECT APPLICATION MODULE TYPE
502: IDENTIFY UTILITY MODULES
603: CREATE RESOURCE IDENTIFIERS
604: SEARCH LUT FOR BEST MATCH
605: INSTALL ON BEST MATCH
APPLICATION INSTALLATION

DESCRIPTION

FIELD OF THE INVENTION

This invention relates to a method of, and system for, installing an application at a runtime instance.

BACKGROUND

“OSGi” is a modularity technology for Java™. Within OSGi, a bundle is a unit of modularity. One example of a bundle is a raw artefact of a binary Java jar file with additional bundle metadata that describes the identity and externals of the bundle. Herein, the bundle is referred to as the “bundle resource”, the raw artefact is referred to as the “artefact resource”, and the metadata for the bundle referred to as the “metadata resource”. Resources are also known as objects.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

It is common for software applications to be composed of individual modules. Most application modules, such as Java Enterprise Edition (EE) EAR files or OSGi applications, will make use of one or more utility modules which provide reusable functionality. Typically utility modules are packaged as part of the application module and referenced by the application module using metadata that is also part of the application module. Installation of the application module into an application framework typically involves copying both the application module and utility module to a single directory on a hard disk. Thus, if several different application modules are installed which make use of a common set of utility modules there will be multiple copies of the same utility module stored on the hard disk, and if the application modules are running in the same application framework runtime instance, there might also be several versions of the same utility module loaded into memory. This is obviously wasteful in terms of disk and memory usage, as well as in terms of the time spent loading the utility modules into memory.
A known solution to this problem is to make use of the concept of shared resources, known as shared libraries (for Java Platform, Enterprise Edition or Java EE (JEE) applications) or shared bundles in a common bundle repository (for OSGi applications). Utility modules which are likely to be common to several application modules are extracted from the application modules and installed as a shared library or common bundle in an application framework runtime instance. At application module installation time the application module is linked to the required shared libraries or bundles. As a result of this process, multiple application modules can make use of the same shared resource, even though there is only one copy of the shared resource on disk. Depending on the application framework, there may also only be one version of the shared resource loaded into memory.

The shared resource approach works well when application module developers and application framework administrators can work together, prior to application installation, to decide which utility modules should be made available as common resources. However, if an application module is to be deployed to an application framework that is provided using a PAAS (Platform As A Service) cloud then it is not possible to plan in advance those resources that can be shared.

Therefore, there is a need in the art to address the aforementioned problem.

**BRIEF SUMMARY OF THE INVENTION**

According to a first aspect of the present invention, there is provided a method of installing an application at a runtime instance, the method comprising the steps of receiving a request to install an application, accessing data defining shared resources required by the application, accessing data defining shared resources already installed at existing runtime instances, selecting an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, and installing the application at the selected runtime instance.

According to a second aspect of the present invention, there is provided a system for installing applications at runtime instances, the system comprising a plurality of servers each arranged to run one or more runtime instances, and a management component connected to the servers and arranged to receive a request to install an application, access data defining shared
resources required by the application, access data defining shared resources already installed at existing runtime instances, select an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, and install the application at the selected runtime instance.

According to a third aspect of the present invention, there is provided a computer program product on a computer readable medium for installing an application at a runtime instance, the product comprising instructions for receiving a request to install an application, accessing data defining shared resources required by the application, accessing data defining shared resources already installed at existing runtime instances, selecting an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, and installing the application at the selected runtime instance.

According to a further aspect, the present invention provides a computer program stored on a computer readable medium and loadable into the internal memory of a digital computer, comprising software code portions, when said program is run on a computer, for performing the steps of the invention.

Owing to the invention, it is possible to provide a mechanism whereby application modules can be automatically targeted to individual machines and application framework runtimes instances in a PAAS cloud based on the utility modules that they use. This approach saves disk space, memory and Central Processing Unit (CPU) cycles compared with installing each application module with its own copy of any utility modules. This saving benefits both individual machines in the cloud and the cloud as a whole. This is particularly beneficial in the case of commonly reused third party modules such as, for instance, common logging utilities that may be shared by many different applications.

When an application module is deployed to a PAAS cloud it is analysed, by the cloud management software, to determine the utility modules it contains. Then existing application framework runtimes instances in the cloud are checked to see if any of them have matching utility modules already installed as, for instance, shared libraries or as bundles in an OSGi bundle repository. The application module is preferentially installed on an existing application
framework runtime instances which already has the shared resources installed that match the utility modules of the application module.

At installation time all utility modules from the new application module are either converted into new shared resources or linked to existing ones. The application is targeted at the server in the cloud that provides as many of the requirements as possible. The conversion of the application from one including utilities by value to one including utilities by reference allows for a more flexible deployment. This approach means that application modules are automatically deployed so as to best make use of the existing runtime capabilities and configuration of the cloud. In particular disk space and process memory is reduced as multiple copies of a utility module can be replaced with a single shared resource.

Preferably, the method further comprises maintaining a table storing an identifier for each shared resource installed at an existing runtime instance and wherein the step of accessing data defining shared resources already installed at existing runtime instances comprises accessing the table. The management component within the computing network can maintain a lookup table that records all of the shared resources in use across all of the existing runtime instances. This provides a simple method of determining the shared resources currently deployed in the cloud computing environment when a new application is needed to be installed within the cloud. The table can be accessed to see which shared resources are currently installed at which specific runtime instance.

Following from the use of the table as outlined in the previous paragraph, the method can advantageously further comprise, following the step of installing the application at the selected runtime instance, updating the table with identifiers for each new shared resource installed during the installation of the application at the selected runtime instance. Each time a request to install an application is made and the best runtime instance has been selected for that application, then the table can be updated to reflect any new shared resources that will become available at the selected runtime instance, as a result of the installation. This provides an ongoing update of the table of shared resources which can lead to future application installations being adapted to reflect the updated table.

Ideally, the method further comprises determining the free capacity available at existing runtime instances and wherein the step of selecting an existing runtime instance with the
greatest number of installed shared resources common to the shared resources required by the
application comprises selecting only an existing runtime instance with sufficient free capacity
to install the application. The selection of the runtime instance to use for the new application
being installed can also take into account the amount of free memory available at each of the
available runtime instances. The selection of the appropriate runtime instance can then be
limited to only those that have sufficient free memory.

Advantageously, the step of installing the application at the selected runtime instance
comprises removing from the application those shared resources already present at the
selected runtime instance and installing the residual components of the application at the
selected runtime instance. The installation of the application can be limited to installing only
those modules needed over and above the shared resources already present at the selected
runtime instance. This is the most efficient method of installing the application and will save
the most processor and bandwidth usage.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the present invention will now be described, by way of example
only, with reference to the following drawings, in which:-

Figure 1 is a schematic diagram of a runtime instance,
Figure 2 is a further schematic diagram of the runtime instance, following installation of a
new application,
Figure 3 is a schematic diagram of an application and a cloud computing environment,
Figure 4 is a diagram of a shared resource lookup table,
Figures 5, 6 and 7 are further schematic diagrams of a cloud computing environment, and
Figure 8 is a flowchart of a method of installing an application.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Figure 1 shows an example of a runtime instance 10, which comprises two specific locations
12 and 14. Application specific modules are installed in the location 12 and shared resources
are installed in the location 14. Two applications 16 (labelled AM1 and AM2) are shown as
being installed in the application area 12. Each application 16 comprises one or more
application specific modules 18 and one or more shared resources 20. The application AM1 comprises an application specific module M1 and the shared resources R1 and R2. Similarly, the application AM2 comprises an application specific module M2 and the same shared resources R1 and R2.

When the application AM1 was installed into the runtime instance 10, the application specific module M1 was installed in the location 12 and the shared resources R1 and R2 were installed in the location 14. The actual shared resources within the application AM1 were replaced with links to the shared resources R1 and R2. Subsequently, when the application AM2 was installed at the runtime instance 10, only the application specific module M2 was installed in the location 12 and the shared resources R1 and R2 within the application AM2 were not installed as they were already present in the location 14. The application AM2 was installed with links to the previously installed shared resources 20.

In the same way, when a new application 16’ is to be installed, then the modules within that application 16’ are examined to see how they compare to the shared resources 20 already installed within the location 14 of the runtime instance 10. In this case, the new application AM3 comprises an application specific module M3 and a shared resource R2 (such as a standard library). When this new application 16’ is to be installed at the runtime instance, then only the application specific module M3 needs to be installed in the location 12 and the shared resource R2 within the application AM3 can be replaced with a link to the already installed copy of R2 that is present within the location 14.

Figure 2 shows the runtime instance 10 after the installation of the third application 16’, which is labelled AM3 in the Figure. The application specific module 18, labelled M3, will be installed in the location 12 and the shared resource 20, labelled R2, will not be installed, as there is already a copy of this specific shared resource 20 installed in the location 14. The shared resource R2 in the application is replaced with a link to the location of the shared resource R2 within the installation area 14. This means that duplicate copies of the same shared resources are not installed at the same runtime instance 10. Only new shared resources will be installed at the location 14.

The methodology described with respect to Figures 1 and 2 provides an efficient process for installing applications at a runtime instance 10 and also provides efficient use of the resources
available at the runtime instance 10. Memory usage is reduced and less bandwidth is consumed when applications are installed. All future applications installed at the same runtime instance 10 will go through the same process of only installing those shared resources 20 that are not already present installed at the location 14. Each new application 16 that is to be installed will be examined to identify those modules within the application 16 that are application specific and those that are shared resources that might already be present.

The runtime instance 10 is present on a server that is providing client devices with access to the applications 16 that are being hosted by the runtime instance 10. A client device will make a request to run an application 16 and a copy will be installed at a runtime instance 10 on an available server to provide the user of the client device with access to the application 16. The application 16 that is installed will comprise one or more application specific modules 18 and one or more shared resources 20. A shared resource 20 is a component that is essentially standardised in some sense, such as a library, and can be used by different applications in the same computing environment.

Figure 3 shows a more complex computing environment, which is essentially a cloud 22 comprised of a network of servers each providing one or more runtime instances 10. The cloud 22 is a computing resource that provides processing and storage functionality to client devices, commonly via sessions that users will initiate to access the computing functionality available. A management component 28 will manage the setting up of the sessions and the installation of applications that a user will access via their local client device. The underlying hardware resources that make up the cloud 22 are invisible to the client devices and the users are not aware of the underlying processes that provide their computing resources.

In this Figure, a new application 16 is shown that is to be installed somewhere within the cloud 22. This application 16 will have to be installed in an existing runtime instance 10 or a new instance will have to be created specifically for the new application 16. This application 16 comprises modules such as the application specific module 18 and the shared resources 20. The question mark shown on the arrow from the application 16 to the cloud 22 indicates the open question of where best to install the application 16. The management component 28 within the cloud 22 must perform an analysis of the runtime instances 10 within the cloud 22.
This analysis determines the most logical location for the installation of the new application 16 within the cloud 22, and this process is described in more detail below. In summary, after receiving a request to install an application 16, a management component 28 accesses data that defines the shared resources 20 required by the application 16 and accesses data that defines shared resources 20 already installed at existing runtime instances 10. The management component 28 selects an existing runtime instance 10 with the greatest number of installed shared resources 20 common to the shared resources 20 required by the application 16 and installs the application 16 at the selected runtime instance 10.

Figure 4 illustrates a shared resources lookup table (LUT) 24, which may be used to store an identifier 26 for each shared resource 20 that is installed at an existing runtime instance 10. Essentially, the lookup table 24 comprises rows, where each row is dedicated to a specific shared resource 20 that is represented in the table 24 by its identifier 26. Each row also contains a list of the runtime instances 10 that contain the specific shared resource 20 of that row. Although the table 24 of Figure 4 shows each shared resource 20 as only present at a single runtime instance 10, it is highly likely, in a practical implementation that each shared resource 20 will be present at multiple runtime instances 10.

The shared resource identifier 26 that is used within the lookup table 24 may be the name of a file, module or library that is commonly used in a specific technology environment. Additionally or alternatively, the shared resource identifier 26 could be created using a predefined hashing function on the shared resource 20, if there is likely to be more than one name used for the same shared resource 20. This will allow the management component 28 to identify any shared resources 20 that constitute part of a new application 16 that is received. The management component 28 will perform the hashing function on each new shared resource to obtain the shared resource identifier 26.

The management component 28 stores and maintains the table 24 and can constantly update the table 24 as new applications 16 are installed. For example, a new application 16 may be received that comprises four shared resources 20, two of which are already present in a specific runtime instance 10, but two of which are completely new to the management component. The new application 16 will be installed at the specific runtime instance 10 that already has the two common shared resources 20 present and the table 24 will be updated to
show that two new shared resources 20 that were not previously known are now installed at that specific runtime instance 10.

Figure 5 illustrates the first process steps that are to be taken when a new application 16’ is to be installed somewhere within the cloud 22. A user deploys an application module to the cloud 22. At deployment time the cloud 22 controlling software being run by the management component 28 performs the following steps, starting by detecting 501 the type of application module, for example as a JEE ear file or as an OSGi application file. File extensions can be used or knowledge about the structure of the programming environment in order to detect the different application module types that are present within the new application 16’ that is to be installed at a runtime instance 10 that is already installed within the cloud 22.

Using the knowledge of the application module types, the management component 28 identifies 502 utility modules, for example jar files referenced from a manifest of an Enterprise Java Beans (EJB) jar file in the ear file, or bundles included by value in an OSGi application. All utility modules will be converted to become a shared library or shared bundle even if only used by a single application module. These modules can be considered to be shared resources 20. Any module within the application 16’ that is generic in nature and not specific to that particular application can be a shared resource 20. These are the modules that can be reused between applications 16 within the same runtime instance 10.

The management component 28, which runs the controlling software for the cloud 22, is neutral as far as the programming language used for the new application 16’ is concerned. Any new application 16’ installed with the cloud 22 can be handled in the methodology described here, as long as the application 16’ comprises discrete modules, more than one of which can be shared between applications 16. In steps one and two shown in Figure 5, the management component 28 examines the modules that make up the new application 16’ and identifies the “utility modules” within that application 16’, in order to perform a more efficient installation of that application 16’ within the cloud 22.

Figure 6 shows the next steps in the process. The management component 28 creates 603 an identifier 26 for each shared resource 20 in the new application 16’. This resource identifier 26 will be used for comparison with other shared resources 20 to check for equality. This may be generated via a hash function or could be name based or could just be the bytes that make
up the shared resource 20. The management component 28 searches 604 for all application framework runtimes 10 in the cloud 22 that already have at least one of the shared resources 20 installed using the shared resource lookup table 24, which maps shared resource identifiers 26 to the runtimes 10 on which they are installed, for example, by searching the lookup table 24. These runtimes 10 are considered potential candidates for installation of the new application 16’.

The management component 28 then installs 605 the application module 18 on the runtime 10 which best satisfies the following set of criteria in that it has as many common shared resources 20 as possible and has sufficient free capacity to meet the performance requirements of the application module 18. At installation time the management component 28 creates any new shared resources/link from the application module 18 to existing shared resources 20 as needed. In this way, the management component 28 selects the most efficient location for the new application 16’. This efficiency reduces the amount of data that has to be installed and also reduces the bandwidth requirement of the installation process, with minimal additional effort by the management component 28.

In the example of Figures 5 and 6, the new application 16’ includes two shared resources 20, which are labelled R1 and R2. The runtime instance 10 that is labelled AFR1 has the shared resource R1 already present, according to the lookup table 24. However, the runtime instance 10 that is labelled AFR2 has both the shared resources R1 and R2 present, again as can be seen from the lookup table 24. This means that this runtime instance AFR2 is the better location for the installation of the new application 16’ and therefore the management component 28 installs the new application 16’ at this runtime instance 10, as shown by the arrow in Figure 6.

If a matching runtime 10 cannot be found for the installation of the new application 16’, the management component 28 provisions a new runtime 10 from the cloud 22 as normal and the application module 18 and new shared resources 20 are installed onto it. Finally, the management component 28 update the shared resource lookup table 24 with any newly created shared resource identifiers 26 and the runtime 10 on which they are installed. This provide a dynamic table 24 that will be continually updated each time a new application 16’ is installed and the efficiency of runtime installation will continue for each new application 16’ as user requests are received to install applications 16 within the cloud 22.
Figure 7 shows the management component 28 as a server within the cloud computing network 22. The management component 28 is in communication with other servers 30 that constitute the cloud computing environment 22. A large number of different machines will make up the cloud 22 including servers 30 and dedicated storage devices that serve multiple components within the cloud 22. The management component 28, in relation to the installation of a new application 16, can be controlled by a computer program product that can be provided from a computer readable medium such as a CD-ROM 32. The instructions with the computer program product are used to control the processor of the management component 28, when handling the installation of a new application 16.

The process embodied by the instructions on the computer program product is summarised in the flowchart of Figure 8. The method of installing a new application 16 at a runtime instance 10 comprises the steps of receiving 801 a request to install an application 16, accessing 802 data defining shared resources 20 required by the application 16, accessing 803 data 24 defining shared resources 20 already installed at existing runtime instances 10, selecting 804 an existing runtime instance 10 with the greatest number of installed shared resources 20 common to the shared resources 20 required by the application 16, and installing 805 the application 16 at the selected runtime instance 10.

In a preferred embodiment of the present invention, a record of where shared resources are to be found is stored in a lookup table 24. It will be appreciated by the skilled person that such records could be stored in and accessed from a number of different artefacts, for example, in a relational database.

As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method, computer program product or computer program. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module” or “system.” Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.
Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java®, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter
scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Aspects of the present invention are described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which
comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

For the avoidance of doubt, the term “comprising”, as used herein throughout the description and claims is not to be construed as meaning “consisting only of”.
CLAIMS

1. A method of installing an application at a runtime instance, the method comprising the steps of:
   o receiving a request to install an application,
   o accessing data defining shared resources required by the application,
   o accessing data defining shared resources already installed at existing runtime instances,
   o selecting an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, and
   o installing the application at the selected runtime instance.

2. A method according to claim 1, and further comprising maintaining a table storing an identifier for each shared resource installed at an existing runtime instance and wherein the step of accessing data defining shared resources already installed at existing runtime instances comprises accessing the table.

3. A method according to claim 2, and further comprising, following the step of installing the application at the selected runtime instance, updating the table with identifiers for each new shared resource installed during the installation of the application at the selected runtime instance.

4. A method according to claim 1, 2 or 3, and further comprising determining the free capacity available at existing runtime instances and wherein the step of selecting an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application comprises selecting only an existing runtime instance with sufficient free capacity to install the application.

5. A method according to any preceding claim, wherein the step of installing the application at the selected runtime instance comprises removing from the application those shared resources already present at the selected runtime instance and installing the residual components of the application at the selected runtime instance.

6. A system for installing applications at runtime instances, the system comprising:
a plurality of servers each arranged to run one or more runtime instances, and
• a management component (28) connected to the servers and arranged to:
  ▪ receive a request to install an application,
  ▪ access data defining shared resources required by the application,
  ▪ access data defining shared resources already installed at existing runtime instances,
  ▪ select an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, and
  ▪ install the application at the selected runtime instance.

7. A system according to claim 6, wherein the management component is further arranged to maintain a table storing an identifier for each shared resource installed at an existing runtime instance and wherein the management component is arranged, when accessing data defining shared resources already installed at existing runtime instances, to access the table.

8. A system according to claim 7, wherein the management component is further arranged to, following the step of installing the application at the selected runtime instance, update the table with identifiers for each new shared resource installed during the installation of the application at the selected runtime instance.

9. A system according to claim 6, 7 or 8, wherein the management component is further arranged to determine the free capacity available at existing runtime instances and, when selecting an existing runtime instance with the greatest number of installed shared resources common to the shared resources required by the application, to select only an existing runtime instance with sufficient free capacity to install the application.

10. A system according to any one of claims 6 to 9, wherein the management component is arranged, when installing the application at the selected runtime instance, to remove from the application those shared resources already present at the selected runtime instance and install the residual components of the application at the selected runtime instance.

11. A computer program product on a computer readable medium for installing an application at a runtime instance, the product comprising instructions for:
• receiving a request to install an application,
o  accessing data defining shared resources required by the application,
 o  accessing data defining shared resources already installed at existing runtime
 instances,
 o  selecting an existing runtime instance with the greatest number of installed shared
 resources common to the shared resources required by the application, and
 o  installing the application at the selected runtime instance.

12. A computer program product according to claim 11, and further comprising
 instructions for maintaining a table storing an identifier for each shared resource installed at
 an existing runtime instance and wherein the instructions for accessing data defining shared
 resources already installed at existing runtime instances comprise instructions for accessing
 the table.

13. A computer program product according to claim 12, and further comprising, following
 the instructions for installing the application at the selected runtime instance, instructions for
 updating the table with identifiers for each new shared resource installed during the
 installation of the application at the selected runtime instance.

14. A computer program product according to claim 11, 12 or 13, and further comprising
 instructions for determining the free capacity available at existing runtime instances and
 wherein the instructions for selecting an existing runtime instance with the greatest number of
 installed shared resources common to the shared resources required by the application
 comprise instructions for selecting only an existing runtime instance with sufficient free
 capacity to install the application.

15. A computer program product according to any one of claims 11 to 14, wherein the
 instructions for installing the application at the selected runtime instance comprise
 instructions for removing from the application those shared resources already present at the
 selected runtime instance and installing the residual components of the application at the
 selected runtime instance.

16. A computer program stored on a computer readable medium and loadable into the
 internal memory of a digital computer, comprising software code portions, when said program
 is run on a computer, for performing the method of any of claims 1 to 5.
**Patents Act 1977: Search Report under Section 17**

**Documents considered to be relevant:**

<table>
<thead>
<tr>
<th>Category</th>
<th>Relevant to claims</th>
<th>Identity of document and passage or figure of particular relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1 - 16</td>
<td>GB 2394570 A (NCR INTERNATIONAL) See pages 11 - 17.</td>
</tr>
<tr>
<td>X</td>
<td>1 - 16</td>
<td>US 6442754 B (CURTIS) See columns 8 - 10.</td>
</tr>
<tr>
<td>X</td>
<td>1 - 16</td>
<td>US 6117187 A (STAELIN) See columns 4 - 7.</td>
</tr>
</tbody>
</table>

**Categories:**

- **X** Document indicating lack of novelty or inventive step
- **Y** Document indicating lack of inventive step if combined with one or more other documents of same category.
- **&** Member of the same patent family
- **A** Document indicating technological background and/or state of the art.
- **P** Document published on or after the declared priority date but before the filing date of this invention.
- **E** Patent document published on or after, but with priority date earlier than, the filing date of this application.

**Field of Search:**

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC:

- G06F

Worldwide search of patent documents classified in the following areas of the IPC:

The following online and other databases have been used in the preparation of this search report:

- EPODOC, WPI.
### International Classification:

<table>
<thead>
<tr>
<th>Subclass</th>
<th>Subgroup</th>
<th>Valid From</th>
</tr>
</thead>
<tbody>
<tr>
<td>G06F</td>
<td>0009/445</td>
<td>01/01/2006</td>
</tr>
</tbody>
</table>