Title: MAPPING APPLICATION DEPENDENCIES AT RUNTIME


(57) Abstract: Systems, methods, and machine-readable and executable instructions are provided for mapping application dependencies. Mapping application dependencies can include defining an application model with a number of application dependencies that enable an application to run and defining a platform model that defines a number of capabilities that are provided by a server. Mapping application dependencies can also include creating a topology model that maps the application model to the platform model at runtime by mapping the number of application dependencies to the number of capabilities while the application remains independent from the server.

Fig. 4

[Continued on nextpage]
Declarations under Rule 4.17:
— as to the identity of the inventor (Rule 4.17(i))

Published:
— as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(H))
— with international search report (Art. 21(3))
MAPPING APPLICATION DEPENDENCIES AT RUNTIME

Background

[0001] Cloud services, be it private or public clouds, are gaining momentum. Maintaining availability of applications running on cloud systems and other types of systems is important. Hybrid cloud systems are becoming increasingly popular as private cloud systems seek to expand into public cloud functionality. The binding of private cloud systems to public cloud systems can affect the availability of applications running on the hybrid cloud system.

Brief Description of the Drawings

[0002] Figure 1 illustrates an example of a hybrid cloud system implemented using a topology model according to the present disclosure.

[0003] Figure 2 illustrates an example of a deployment stack according to the present disclosure.

[0004] Figure 3 illustrates an example of application dependency mapping according to the present disclosure.

[0005] Figure 4 illustrates an example of tag resolving at deployment time according to the present disclosure.

[0006] Figure 5 is a flow chart illustrating an example of a method for mapping application dependencies at runtime according the present disclosure.
Figure 6 illustrates a block diagram of an example of a machine-readable medium in communication with processing resources for mapping application dependencies at runtime according to the present disclosure.

Detailed Description

Examples of the present disclosure may include methods and systems for mapping application dependencies at runtime. An example method for mapping application dependencies at runtime may include defining an application model with a number of application dependencies that enables an application to run and defining a platform model that defines a number of capabilities that are provided by a server. An example method for mapping application dependencies at runtime can also include creating a topology model that maps the application model to the platform model at runtime by mapping the number of application dependencies to the number of capabilities while the application remains independent from the server.

The figures herein follow a numbering convention in which the first digit or digits correspond to the drawing figure number and the remaining digits identify an element or component in the drawing. Similar elements or components between different figures may be identified by the use of similar digits. For example, 108 may reference element "08" in Figure 1, and a similar element may be referenced as 308 in Figure 3.

As used herein, "a" or "a number of" something can refer to one or more such things. For example, "a number of widgets" can refer to one or more widgets.

Hybrid cloud systems can combine two or more cloud systems. For example, a hybrid cloud system can combine a private cloud system and a public cloud system. A public cloud system can include a cloud system that makes applications, storage, and/or other resources available to the general public through a service provider. A private cloud system can include a cloud system that is operated solely by one entity for the use of that entity.
[0012] A hybrid cloud system can combine two or more cloud systems by combining the resources of the two or more cloud systems. For example, a hybrid cloud system can combine the hardware resources of a private cloud system with the monitoring resources of a public cloud system.

[0013] Hardware resources and software resources associated with a first cloud system can limit the use of an application in a second cloud system. For example, if a first cloud system is associated with a first operating system (OS), then a second cloud system can only use monitoring resources, e.g., application, that are compatible with the second cloud system. In a number of examples of the present disclosure, a hybrid cloud system can combine two or more cloud systems that are independent of each other without limiting the use of an application.

[0014] Figure 1 illustrates an example of a hybrid cloud system implemented using a topology model according to the present disclosure. Hybrid cloud system 100 can include public cloud system 102 and private cloud system 104.

[0015] A public cloud system 102 can include a number of capabilities. For example, a public cloud system 102 can include an run book automation (RBA) server 106-1 and a monitor server 106-2. An RBA server 106-1 can define, build, orchestrate, manage and report on workflow, e.g., capabilities. Additionally an RBA server 106-1 can customize automation of a system. A monitor server 106-2 can manage a number of objects. For example, a monitor server 106-2 can manage an application, a server, and/or any other objects from which the monitor server can receive a number of messages.

[0016] A private cloud 104 can include a number of capabilities. For example, a private cloud can include a deploy server 106-3 and an infrastructure as a service (laaS) server 106-4. A deploy server 106-3 can deploying an application, e.g., capabilities. An laaS server 106-4 can include the capability of provisioning system infrastructure.

[0017] A server can include any system that provides or requests a number of resources. A server can include a hardware infrastructure. A server can also include a software infrastructure. Capabilities associated with a server
can reference the services and/or resources that a server can provide. A server can provide services and/or resources through a hardware infrastructure associated with a server and/or through a software infrastructure associated with a server.

[0018] A number of capabilities can also include properties associated with a server. Disclosing properties can include providing a number of parameters associated with a server. The parameters can include parameters that originate with a first server that provides specific capabilities or parameters that originate with a second server that does not provide the specific capabilities. For example, a first server that is hosting a webpage can query a second server for a port number. The second server can provide the port number to the first server. The first server can then provide the port number, e.g., parameter, to an application regardless of where the port number originated. That is, the first server can provide a capability regardless of where the port number, e.g., capability, originated.

[0019] In a number of examples of the present disclosure, a topology model 108 can allow an application to use the capabilities of the public cloud system 102 and the private cloud system 104. That is, a topology model 108 can facilitate interactions between the different servers in the public cloud system 102 and the private cloud system 104 without requiring an application to be configured to any particular system.

[0020] Figure 2 illustrates an example of a deployment stack according to the present disclosure. A deployment stack 220 can include a hardware stack 222, an OS stack 224, a middleware stack 226, and/or an Application stack 228. A deployment stack 220 can include more or less stacks. Furthermore, a deployment stack 220 can include different stacks that convey other functionalities. For example, a deployment stack 220 can include multiple applications that share application stack 228.

[0021] A hardware stack 222 can include hardware infrastructure. Hardware infrastructure can include a number of hardware components that are associated with computing. For example, hardware infrastructure can include a physical server, a physical network, a processor, memory, and/or any number of
hardware components that assist in computing. Hardware infrastructure can include more or less components than those listed. Hardware infrastructure can also include a hardware infrastructure that is simulated in software. For example, hardware infrastructure can include hardware virtualization. Hardware virtualization simulates a hardware infrastructure, e.g., a simulated computer environment.

An OS stack 224 can include a number of OSs that run on a hardware system, e.g., hardware infrastructure. An OS can provide a number of basic services to an application. The basic services can provide access to hardware infrastructure. However, an OS is not limited to basic services and can include a number of services.

Middleware stack 226 can include middleware. Middleware includes software that provides services to an application beyond those provided by the OS stack 224. For example, middleware can include a number of applications, libraries, and/or configurations that provide services and/or resources to an application beyond those provided by a hardware stack 222 and/or an OS stack 224.

An application stack 228 can include a number of applications. A number of applications can include applications that are supported by the middleware stack 226, the OS stack 224, and/or the hardware stack 222. In a number of examples of the present disclosure, an application stack 228 can include applications that can be categorized as middleware. For example, an application such as a web hosting application, can under a first set of circumstances, be categorized as middleware because it can provide service and/or resources to a different application. However, an application such as a web hosting application can, under a second set of circumstances, be categorized as a standalone application because the web hosting application can depend on middleware.

A deployment stack 220 can define a number of dependencies between sub-stacks, e.g., a hardware stack 222, an OS stack 224, a middleware stack 226, and/or an application stack 228. For example, a deployment stack 220 can define the dependencies that an OS in an OS stack
224 can have on hardware infrastructure in a hardware stack 222. A
deployment stack 220 can also define a number of dependencies that
middleware in a middleware stack 226 can have on an OS and/or on hardware
infrastructure. Moreover, a deployment stack 220 can define a number of
dependencies that an application in an application stack 228 can have on
middleware, an OS, and/or hardware infrastructure.

A deployment stack 220 can also define a number of
dependencies within individual sub-stacks. For example, a first subset of
hardware infrastructure can depend on a second subset of hardware
infrastructure wherein the first subset of hardware infrastructure and the second
subset of hardware infrastructure can be part of a hardware stack 222.

A deployment stack 220 can order the dependencies between
sub-stacks and the dependencies within sub-stacks. An order of the
dependencies can include a number of different orders and is not limited to a
single methodology of ordering. In an example of the present disclosure, an
order of the dependencies can include giving the hardware stack 222
dependencies a first priority, the OS stack 224 dependencies a second priority,
the middleware stack 226 dependencies a third priority, and the application
stack 228 dependencies a fourth priority.

Figure 3 illustrates an example of application dependency
mapping according to the present disclosure. A topology model 308, which can
be analogous to topology model 108 in Figure 1, can map a number of
application model dependencies to a number of platform model capabilities at
runtime.

An application model 332 can define a number of configuration
units of an application and the dependencies of the number of configuration
units. A number of configuration units can define a number of application layers
that compose an application. For example, an application model 332 can
include an application layer 336-1 and a database layer 336-2 (referred to
generally as layers 336). An application layer 336-1 can define the core
components of an application while a database layer 336-2 can define an
interaction between the application and a database. The Layers 336 can define
a number of dependencies between themselves. For example, an application
layer 336-1 can depend on a database layer 336-2. The Layers 336 can also
define a number of dependencies outside of the application model 332.

A platform model 334 can describe an environment that can
support a number of applications by providing a number of capabilities. A
platform model can be composed of a number of tiers. A tier can include a
configuration of servers that provides hardware infrastructure, an OS, and/or
middleware capabilities. The servers within a tier can reside in a common
network configuration. A number of tiers can include a number of different
configurations of servers. For example, platform model 334 can include a
tomcat tier 338-1 and a MySQL tier 338-2. A tomcat tier 338-1 can include the
capabilities of hosting a web application. A MySQL tier 338-2 can include the
capabilities of initializing and managing a database. The capabilities that a tier
provides can be provided by a configuration of servers at runtime. Furthermore,
the capabilities that a tier provides can be provided after infrastructure
provisioning, e.g., after runtime.

A topology model can provide the fulfillment of a number of
application dependencies such that an application can be deployed on a
specific tier, e.g., a configuration of servers. A topology model can be specific
to a tier and an application. For example, a first topology model can map an
application model to a first platform model and a second topology model can
map an application to a second platform model wherein the first platform model
and the second platform model can differ.

In Figure 3, topology model 308 can map a dependency
associated with an application layer 336-1 to a capability associated with a
tomcat tier 338-1. For example, if the dependency is defined as the installation
of a Java web application, then the Tomcat tier 338-1 can provide the capability
of installing a java web application. Topology model 308 can map a
dependency associated with a DB layer 336-2 to a capability associated with a
MySQL tier 338-2. For example, if the dependency is defined as initializing a
database, then a MySQL tier 338-2 can provide the capability of initializing a
database.
[0033] A topology model 308 can fulfill the dependencies through the use of a deployment stack. The deployment stack can define an order in which dependencies can be fulfilled. For example, in an example of a deployment stack, an installation of a java web application dependency can be fulfilled before an initialization of a database dependency. Alternatively, an initialization of a database dependency can be fulfilled before an installation of a java web application dependency.

[0034] A topology model can map and fulfill a number of dependencies at runtime. In a number of examples of the present disclosure, runtime can include a time when an application initiates execution. In some examples, runtime can include an operation execution time. For example, an operation execution time can include a time at which an application is deployed.

[0035] Figure 4 illustrates an example of tag resolving at deployment time according to the present disclosure. An application model 432, which can be analogous to an application model 332 in Figure 3, can define a number of dependencies through a number of tags 440-1. A platform model 434, which can be analogous to a platform model 334 in Figure 3, can define a number of capabilities through the number of tags 440-1.

[0036] A tag can allow an application model to remain independent from a platform model. A tag can allow for the communication of dependencies and capabilities without requiring communication between an application and a tier. For example, a tag can allow an application model to be re-mapped to a number of platform models without changing the application model because the application is not configured to any specific tier.

[0037] When an application model places a tag on a dependency it can create a contract with a platform model. A platform model can guarantee the application model, e.g., the application layer, that a number of capabilities that are associated with a tag will be available at runtime, e.g., deployment time. A tag can allow an application layer to create a reference variable that references a capability, e.g., service and/or resource, that is provided by a platform model. A tag can allow a reference variable to reference a capability even though the application layer does not know where and/or how the capability will be fulfilled.
A reference variable can be late binding. Late binding includes resolving a reference variable at runtime, e.g., deployment time and/or operation execution time. Furthermore, an application layer can reference parameters through a number of tags without being bound to a specific platform.

For example, an application layer 436 can tag 442-1 a reference variable Param_Port with an http_port tag. A tomcat tier 438 in platform model 434 can tag 442-2 a port number, e.g., 8080, representing a port. The application layer 436 can tag the reference variable Param_Port without being bound to a specific platform tier. The tags can be resolved at deployment time 444 by mapping a Param_Port reference variable that is tagged, e.g., http_port tag, to a port number 8080 that is tagged with a corresponding tag, e.g., http_port tag.

Figure 5 is a flow chart illustrating an example of a method for mapping application dependencies at runtime according the present disclosure. At 550, an application model can be defined with a number of application dependencies that enable an application to run. At 550, a platform model can be defined a number of capabilities that are provided by a server. At 554, a topology model can be created, wherein the topology model can map the application model to the platform model at runtime by mapping the number of application dependencies to the number of capabilities while the application remains independent from the server.

In a number of examples of the present disclosure, an application model can be mapped to a platform model at run time. A mapping can include ordering a number of dependencies and resolving the dependencies at runtime. An application model can create a number of dependencies using a number of tags. Creating a number of dependencies through a number of tags can allow an application to remain independent of a server. The dependencies can be created through reference variables. A reference variable can be tagged without binding a reference variable to a specific server. The application can remain independent because the application can create dependencies without requiring the application to communicate with the server.
[0040] Figure 6 illustrates a block diagram of an example of a computer-readable medium in communication with processing resources for platform runtime abstraction according to the present disclosure. The computer readable medium 688 (e.g., a tangible, non-transitory medium) and/or the memory resource 686 can store a set of instructions executable by the processing resource 684 to define 690 an application model. The instructions can be executed to define 692 a platform model. The instructions can be executed to create 694 a topology model that maps the application model to the platform model.

[0041] An application model can be defined 690 with a number of applications dependencies that enable a number of applications to run. The number of application dependencies that enable the number of applications to run can include application dependencies within the number of applications and application dependencies between the number of applications and a number of servers. A platform model can be defined 692 with a number of tiers that define a number of capabilities wherein a tier includes a group of servers. A tier can include a group of servers that have the same OS and the same middleware. A topology model can map the application model to the platform model by mapping the number of application dependencies to the number of capabilities while the application model remains independent form the platform model. The topology model can resolve the application dependencies by using a deployment stack. A deployment stack can define an ordering of the application dependencies wherein the ordering of the application dependencies can define an order in which the number of application dependencies can be resolved. A number of applications can define the priority of the dependencies. The priority of the dependencies can provide a basis for the ordering of the application dependencies. The topology model can create a mapping between the number of application dependencies and a number of capabilities that a platform model provides.

[0042] The methods, techniques, systems, and apparatuses described herein may be implemented in digital electronic circuitry or computer hardware, for example, by executing instructions stored in computer-readable storage.
media. Apparatuses implementing these techniques may include appropriate input and output devices, a computer processor, and/or a tangible computer-readable storage medium storing instructions for execution by a processor.  

[0043] A process implementing techniques disclosed herein may be performed by a processor executing instructions stored on a tangible computer-readable storage medium for performing desired functions by operating on input data and generating appropriate output. Suitable processors include, by way of example, both general and special purpose microprocessors. Suitable computer-readable storage devices for storing executable instructions include all forms of non-volatile memory, including, by way of example, semiconductor memory devices, such as Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and flash memory devices; magnetic disks such as fixed, floppy, and removable disks; other magnetic media including tape; and optical media such as Compact Discs (CDs) or Digital Video Disks (DVDs). Any of the foregoing may be supplemented by, or incorporated in, specially designed application-specific integrated circuits (ASICs).  

[0044] Although the operations of the disclosed techniques may be described herein as being performed in a certain order and/or in certain combinations, in some implementations, individual operations may be rearranged in a different order, combined with other operations described herein, and/or eliminated, and the desired results still may be achieved. Similarly, components in the disclosed systems may be combined in a different manner and/or replaced or supplemented by other components and the desired results still may be achieved.  

[0045] The above specification, examples and data provide a description of the method and applications, and use of the system and method of the present disclosure. Since many examples can be made without departing from the spirit and scope of the system and method of the present disclosure, this specification merely sets forth some of the many possible embodiment configurations and implementations.
What is claimed:

1. A method for mapping application dependencies at runtime comprising:
   defining an application model with a number of application dependencies that enables an application to run;
   defining a platform model that defines a number of capabilities that are provided by a server; and
   creating a topology model that maps the application model to the platform model at runtime by mapping the number of application dependencies to the number of capabilities while the application remains independent from the server.

2. The method of claim 1, wherein creating the topology model that maps the application model to the platform model at runtime includes mapping the application model to a platform model when the application initiates execution.

3. The method of claim 1, wherein the application remains independent from server includes the topology model resolving the application dependencies without requiring the application to communicate with the server.

4. The method of claim 3, wherein topology model resolves the application dependencies by providing a number of values to a number of reference variables that are used by the application to reference the number of capabilities.

5. A non-transitory computer-readable medium storing instructions for mapping application dependencies at runtime executable by a computer to cause the computer to:
   define an application model with a number of application dependencies that enable a number of applications to run;
define a platform model with a number of tiers that define a number of capabilities wherein a tier includes a group of servers; and

create a topology model that maps the application model to the platform model at runtime by mapping the number of application dependencies to the number of capabilities while the application model remains independent from the platform model.

6. The medium of claim 5, wherein the number of application dependencies that enable the number of applications to run include application dependencies within the number of applications and application dependencies between the number of applications and a number of servers that define the number of tiers.

7. The medium of claim 5, wherein the topology model that maps the application model to a platform model at runtime resolves the application dependencies using a deployment stack.

8. The medium of claim 7, wherein the deployment stack includes an ordering of the application dependencies wherein the ordering of the application dependencies defines an order in which the number of application dependencies are resolved.

9. The medium of claim 5, wherein the topology model includes a mapping between the number of application dependencies and the number of tiers.

10. The medium of claim 5, wherein the tier that includes the group of servers include a group of servers that have the same operating system (OS) and the same middleware.

11. An system that maps application dependencies at runtime, comprising:
define an application model with a number of requested parameters that enable a number of applications to run;

define a platform model that includes a number of tiers that include a number of server parameters wherein a tier includes a group of servers that provide the same operating system (OS) and the same middleware; and

create a topology model that maps the application model to the platform model at runtime by mapping the number of requested parameters to the number of server parameters through a number of tags while the application model remains independent from the platform model.

12. The system of claim 11, wherein the number of applications define a number of requested parameters by placing a number of dependencies on a platform that supports the platform model through a number of tags.

13. The system of claim 12, wherein the number of tags create a contract between the number of applications and a platform wherein the contract guarantees the number of applications that the number of server parameters that correspond to the requested parameters will be available at deployment.

14. The system of claim 12 wherein the number of tags allows the number of applications to leverage a number platform specific properties that correspond to the platform through a number of reference variables.

15. The system of claim 11, wherein the reference variables are resolved at run time.
PROVIDED TAGS:
- home
- webapp-dir
- http_port

APPLICATION MODEL
APP LAYER: WEBAPP

PLATFORM MODEL
TOMCAT TIER (8080)

TAG parma_port WITH THE http_port TAG

AT DEPLOYMENT TIME THE VALUE IS RESOLVED: parma_port = 8080

TAG 8080 WITH THE http-port TAG
DEFINING AN APPLICATION MODEL WITH A NUMBER OF APPLICATION DEPENDENCIES THAT ENABLE AN APPLICATION TO RUN

DEFINING A PLATFORM MODEL THAT DEFINES A NUMBER OF CAPABILITIES THAT ARE PROVIDED BY A SERVER

CREATING A TOPOLOGY MODEL THAT MAPS THE APPLICATION MODEL TO THE PLATFORM MODEL AT RUNTIME BY MAPPING THE NUMBER OF APPLICATION DEPENDENCIES TO THE NUMBER OF CAPABILITIES WHILE THE APPLICATION REMAINS INDEPENDENT FROM THE SERVER

Fig. 5
PROCESSING RESOURCE

MEMORY RESOURCE

COMPUTER READABLE MEDIUM

DEFINE AN APPLICATION MODEL

DEFINE A PLATFORM MODEL

CREATE A TOPOLOGY MODEL THAT MAPS THE APPLICATION MODEL TO THE PLATFORM MODEL

Fig. 6
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 9/44(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 9/06; G06F 9/44; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: application, dependency, mapping, runtime;

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 2010-0037211 AI (ZAKINOV ALEX et al.) 11 February 2010 See abstract, paragraphs [0013], [0020], [0022], [0023], and figure 1.</td>
<td>1-15</td>
</tr>
<tr>
<td>A</td>
<td>US 2012-0072887 AI (BASAK INDRANIL) 22 March 2012 See abstract, paragraphs [0037], [0039], [0040], [0043], claim 1, and figures 5-6.</td>
<td>1-15</td>
</tr>
<tr>
<td>A</td>
<td>US 7912873 B2 (OR YAN et al.) 22 March 2011 See abstract, column 21, lines 2-11, 20-23, claim 1, and figure 6.</td>
<td>1-15</td>
</tr>
<tr>
<td>A</td>
<td>US 2010-0153916 AI (BHATKHANDE MANDAR GANESH et al.) 17 June 2010 See abstract, paragraphs [0023], [0031], claim 1, and figure 2.</td>
<td>1-15</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C.

See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

Date of mailing of the international search report
27 DECEMBER 2012 (27.12.2012)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer

BOK, Jin Yo
Telephone No. 82-42-481-51 13

Form PCT/ISA/210 (second sheet) (July 2009)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 2010-00372 A1</td>
<td>11.02.2010</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2003-251801 A1</td>
<td>09.02.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2003-251827 A1</td>
<td>09.02.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2003-251827 A8</td>
<td>09.02.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 2003-251855 A1</td>
<td>09.02.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1565811 A1</td>
<td>24.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1565814 A1</td>
<td>24.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1565815 A1</td>
<td>24.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1565851 A2</td>
<td>24.08.2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2006-0184926 A1</td>
<td>17.08.2006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7243306 B1</td>
<td>10.07.2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7337184 B1</td>
<td>26.02.2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7962590 B1</td>
<td>14.06.2011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2004-010246 A2</td>
<td>29.01.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2004-010246 A3</td>
<td>29.01.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2004-010292 A1</td>
<td>29.01.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2004-010293 A1</td>
<td>29.01.2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2004-010298 A1</td>
<td>29.01.2004</td>
</tr>
<tr>
<td>US 2010-0153916 A1</td>
<td>17.06.2010</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Form PCT/ISA/210 (patent family annex) (July 2009)