An approach for developing an interoperability mechanism between cloud providers

Meriem Thabet* and Mahmoud Boufaida

LIRE Laboratory,
Software Technologies and Information Systems Department,
Constantine 2 University,
25000, Algeria
E-mail: thabet.meriem@hotmail.fr
E-mail: mboufaida@umc.edu.dz
*Corresponding author

Fabrice Kordon

LIP6 Laboratory,
Pierre and Marie Curie University,
4, place Jussieu, 75252 Paris, Cedex 05, France
E-mail: fabrice.kordon@lip6.fr

Abstract: Due to the presence of numerous cloud service providers, the requirement is emerging for interoperability between them so that companies can choose multiple suppliers to fit their needs. This paper provides an approach to address the problem of cloud interoperability. We aim at facilitating the collaboration among providers by proposing an architecture based on an agent society, to support and ensure the data portability and interoperability. For that, we define a two-phase migration protocol that enables data portability by permitting providers to exchange data regardless of their infrastructure, tools and platforms, according to a specific demand in order to satisfy companies' needs.

Keywords: cloud provider; interoperability; data portability; agent; heterogeneous clouds; migration protocol; space-based situated computing.

Reference to this paper should be made as follows: Thabet, M., Boufaida, M. and Kordon, F. (2014) 'An approach for developing an interoperability mechanism between cloud providers', *Int. J. Space-Based and Situated Computing*, Vol. 4, No. 2, pp.88–99.

Biographical notes: Meriem Thabet is preparing her PhD thesis in the field of cloud interoperability at the Software Technologies and Information Systems Department of Constantine 2 University in Algeria. She is affiliated to LIRE Laboratory. She received her Master degree in 2010. She is currently working on multi-agent systems, web service, data portability and interoperability issues in cloud computing environments.

Mahmoud Boufaida is a Full Professor at the Software Technologies and Information Systems Department of Constantine 2 University in Algeria. He heads the Research Group 'Information Systems and Knowledge Bases'. He has managed and initiated multiple national and international level projects including interoperability of information system and integration of applications in organisations. His research interests include cooperative information systems, web databases, software engineering and service oriented architectures.

Fabrice Kordon is a Full Professor at the University of Pierre et Marie Curie and the Head of the Team 'Modelling and Verification in the 'Network and Distributed Systems' at LIP6 Laboratory. His research interests mainly concern: distributed systems, software engineering and formal methods.

This paper is a revised and expanded version of a paper entitled 'An agent-based architecture and a two-phase protocol for the data portability in clouds' presented at 27th International Conference on Advance Information Networking and Applications Workshops, Barcelona, Spain, 2013.

1 Introduction

Despite the increasing use of clouds technology by companies, their interoperability remains a problem due to the risk of becoming trapped in proprietary approaches that could contradict the objectives of the system. Therefore, interoperability plays an important role in making the providers' services more reliable. Moreover, companies often need more than one service provider to work together to meet their needs (Kim, 2009). Unfortunately, existing cloud computing solutions have not been built with interoperability in mind (Sheth and Ranabahu, 2010a). They usually lock customers into a single cloud provider, those hinder the portability of data or software relying on them (Joe, 2010).

Among the problems arising from the numerous cloud computing solutions is the lock-in, where data and applications can hardly be moved to other systems (thus locked-in) forcing cloud users to rely on one service provider (Sheth and Ranabahu, 2010b). Users must then decide in advance the cloud service provider they will use. This hinders the use of clouds. So, solving this provider lock-in relies on an ability to let different cloud service providers cooperate.

The work presented in this paper proposes an approach, which aims at overcoming the lock-in problem and ensures interoperability between cloud service providers. To do so, we claim that data portability is a fundamental requirement and a necessary precondition to be fulfilled (Loutas et al., 2011), because it lets different services from several providers share data.

The proposed approach provides an architecture that relies on a registry describing all the offered services via a 'description and publication agent' and on an intermediate layer managing services invocation by companies through a coordinator agent. This architecture also relies on the use of mobile agents in order to move data from one cloud to another. We specify a two-phase migration protocol, which shows the data transfer among clouds. Each cloud defines its own data format, platforms and tools. Since data format exchanged among cloud providers still suffer from clouds' heterogeneity, we advocate the intervention of adapter agents to convert data formats moved from one environment to another.

This paper is organised as follows. Section 2 presents a short overview of our approach. Section 3 describes the designed architecture and the interoperability mechanism focusing on the used components. The migration protocol of the mobile agent and the different actions of the adapter agent are defined in Sections 4 and 5. Section 6 demonstrates how the migration protocol can be used by means of a case study, processed on a prototype implementation. Some related works are discussed in Section 7. Finally, Section 8 draws a conclusion of this paper and shows future directions of this research work.

2 Overview of the proposed approach

We describe in this section our approach to ensure interoperability between different cloud service providers. The essence of this approach is to ease the collaboration among clouds and let them share data regardless their platforms, structure and tools.

Data portability is a key factor to achieve interoperability, because data sharing remains the most critical aspect of getting the clouds to work together. So, the common linking thread among these clouds will be the data. As long as the data can be interchanged between different cloud services (Shahab, 2010), the needs of the companies will be largely met.

Our main contribution resides in the use of agent technology. The agent paradigm brings some useful characteristics such as intelligence, autonomy, cooperation and mobility. The mobility aspect allows an agent to process information on the server and to send only the relevant. This minimises communication costs and reduces significantly the execution time of tasks.

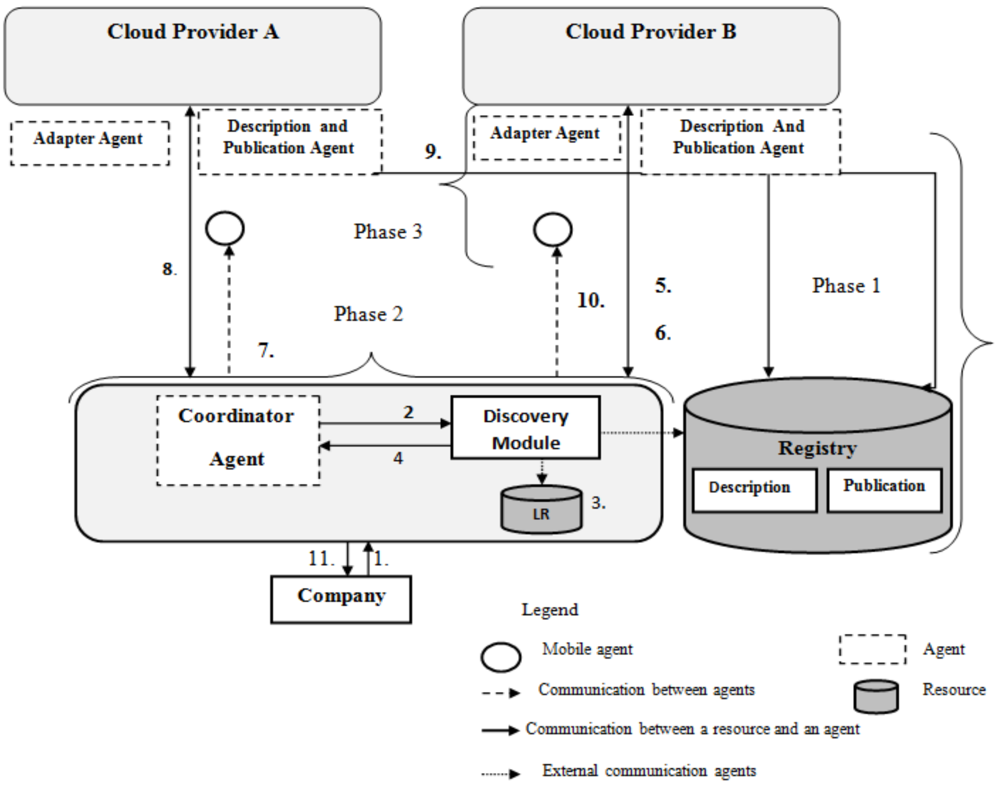
As a matter of fact, our agent-based architecture relies on a service registry, where the providers' services are described and published via a 'description and publication agent' in this registry. This latter provides all the services as atomic services that will be discovered and invoked through an intermediate layer.

In order to ensure the data portability among different clouds, we use the concept of mobile agent that can carry the company's data from one provider to another if such a need arises, due to their ability to move between different environments. Moreover, the heterogeneity of clouds is another significant issue, because when the data has to be interchanged between clouds, the format becomes important and needs to be appropriate according to providers. This issue can be solved by using the adapter agents. These last convert the data format exchanged from one model to another.

3 An agent-based architecture

The interoperability system proposed in this paper is divided into two parts: static and dynamic (see Figure 1). The static part presents the different used components in our architecture (resources and agents) that are described in Section 3.1. The dynamic part shows how these components are invoked during the three interoperability process phases. The description and publication of the services phase permits to define the services offered by providers and to publish them in the registry. The phase of services discovery relies on the charge of identifying and invoking services by companies. Finally, the phase of portability and adaptation of the exchanged data is related to data sharing and their translation from one provider's model to the model used by another one. More details of these phases are explained in Section 3.2.

Figure 1 Global overview of the architecture



In fact in our approach, the concepts of 'registries', 'description and publication agent', and 'discovery module' have been inspired from Ishak et al. (2008). We have integrated these concepts in our architecture in order to facilitate the collaboration among cloud providers. Also, we will not develop these aspects in this work because we are only focusing on the problem of the portability and adaptation of data.

3.1 *Components of the architecture*

This section describes the components used in our architecture, which is based on two kinds of components: resources and agents.

3.1.1 *Resources*

They are of two types:

- *Service registry (global registry)*: it is an entity that contains the description of services offered by different suppliers. It consists of two modules. The publication module enables the cloud providers to publish the descriptions of their services. The discovery module allows companies to discover and identify services and their providers.
- *Local registry*: it is a limited copy of the global registry. This registry stores information about the most requested services by companies, in order to speed their discovery and to guide companies in choosing them.

3.1.2 *Agents*

Four types of agents are defined in our system (see the list below). They cooperate to provide services with a high added value. Each agent is capable of flexible autonomous actions in order to achieve its objectives.

- *Description and publication agent*: it handles the discovery and publication of services and consists of two modules. The description module describes all the services provided. The publication module saves all these descriptions in the registry. The internal tasks of this agent are shown in Thabet and Boufaida (2013).
- *Coordinator agent*: this agent coordinates all communications between agents and manages the accesses to the registry through its communication module. It also contains an activation module that triggers the mobile agent when it is necessary by sending it a message. The transfer of results to the company is made at the level of the exploitation module.

Figure 2 Coordinator agent and its internal tasks

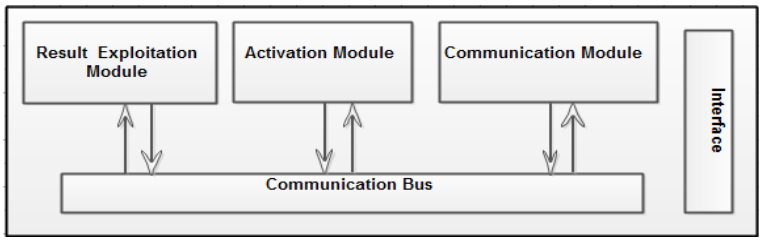
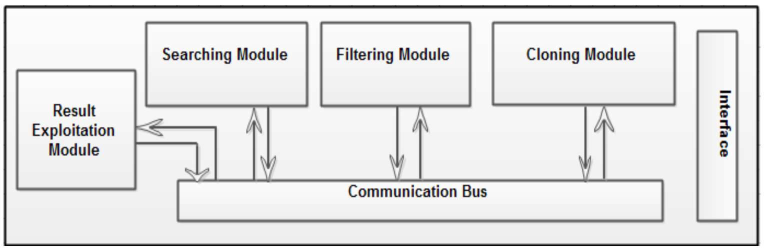


Figure 3 Mobile agent and its internal tasks



- *Mobile agent*: in this context, the concept of mobile agent appears as a solution to facilitate the data portability. These agents search and collect data by performing some processing (search, collect, filter). This processing reduces the amount of information carried by agents, therefore the network traffic.
- *Adapter agent*: this type of agent enables the conversion of data formats retrieved from one cloud to another. Thus, each cloud must have its own adapter agent, which collects and reconstructs all the results received from the mobile agents and converts them according to the format of the provider recipient.

3.2 *Different phases of the interoperability process*

As mentioned above our system is based on three phases. The aims of these phases are as follows.

3.2.1 *Description and publication of services*

This phase shows how the provided services are described and defined. It requires the use of a 'description and publication agent' and a 'service registry' to register the web services. This agent describes the services and stores these descriptions in the global registry. In other words, it can publish the providers' services or modify them by using the publication module.

3.2.2 *Discovery of services by companies*

It is the process of identifying the web services that have been previously published. In this phase, all the services are sought and located from the registry. Only those that are potential to meet companies' needs are discovered by using the intermediate layer. This latter contains three components: the coordinator agent, the discovery module and the local registry. Following the call of the discovery module by the coordinator agent, this module searches in the local registry in order to find the most requested services by companies. In case of certain services are not found in the local registry, another research is done in the global registry. After the discovery process is complete, the results will be transmitted to the coordinator agent that will invoke the service provider and send the results to the company.

3.2.3 *Portability and adaptation of exchanged data*

This phase is related to the interoperability aspect. It constitutes the originality of our approach. When a company invokes a service provider that requires the use of data from one or more other cloud providers, a set of generated mobile agents is activated. These agents allow searching, collecting and filtering data by crossing the cloud. However, the heterogeneity of cloud service providers is another problem because each provider uses its own data structure. So, this requires the intervention of adapter agents for converting the data format exchanged from one environment to another. Each cloud has its own adapter agent.

4 A two-phase migration protocol of the mobile agent

The sequence of all the messages that are exchanged between agents forms the migration protocol of the mobile agent. Figure 4 shows a set of transitions between the mobile agent 'MA' the coordinator agent 'CA' and the adapter one 'AA'. The migration protocol of our mobile agent is composed of two essential phases: activation and migration.

- *Activation of the mobile agent*: after having received the request from the provider and after having performed a research in the registry by the discovery module, the coordinator agent sends a message to activate the mobile agent: *activate (API, data, MA)*.

- *Migration of the mobile agent*: after having activated the mobile agent, this latter extracts the different parameters of the message that have been received at the interface. After their transmission, at the level of the searching module, the mobile agent migrates within the cloud to search the required information.

The mobile agent clones itself and sends its replica to agents to inform them about the sites that it has already visited. This method does not eliminate all redundancy of research, but allows for a compromise between, the loss of time in existing research and the excessive messages that can sometimes be penalised without satisfactory result.

Each agent that does not lead to a definitive result in a predefined time is ignored.

The filtering module selects the received results and sends them to the exploitation module. Filtering can be done progressively from the reception of data, to improve performance in terms of time. Once the mobile agent produces the results, which the address is known by all the other mobile agents, it sends a direct message to the adapter agent. This message contains the corresponding data according to the specified request.

In the next section we present the different actions of the adapter agent.

Figure 4 Migration protocol

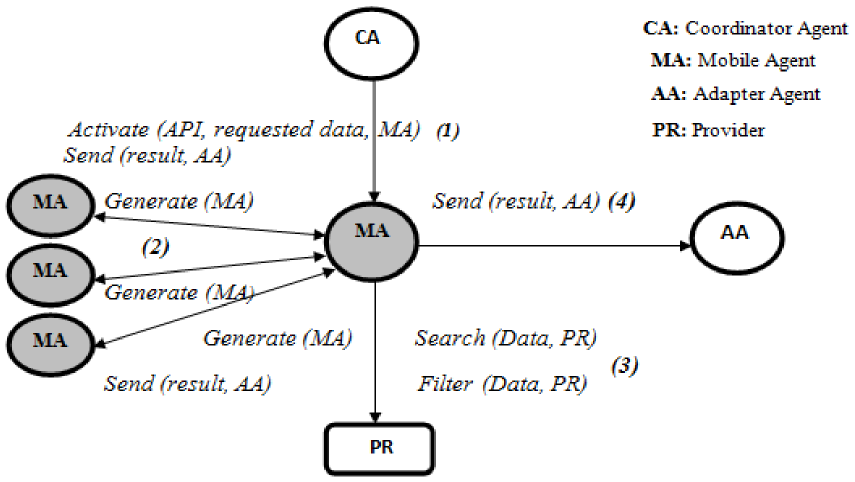
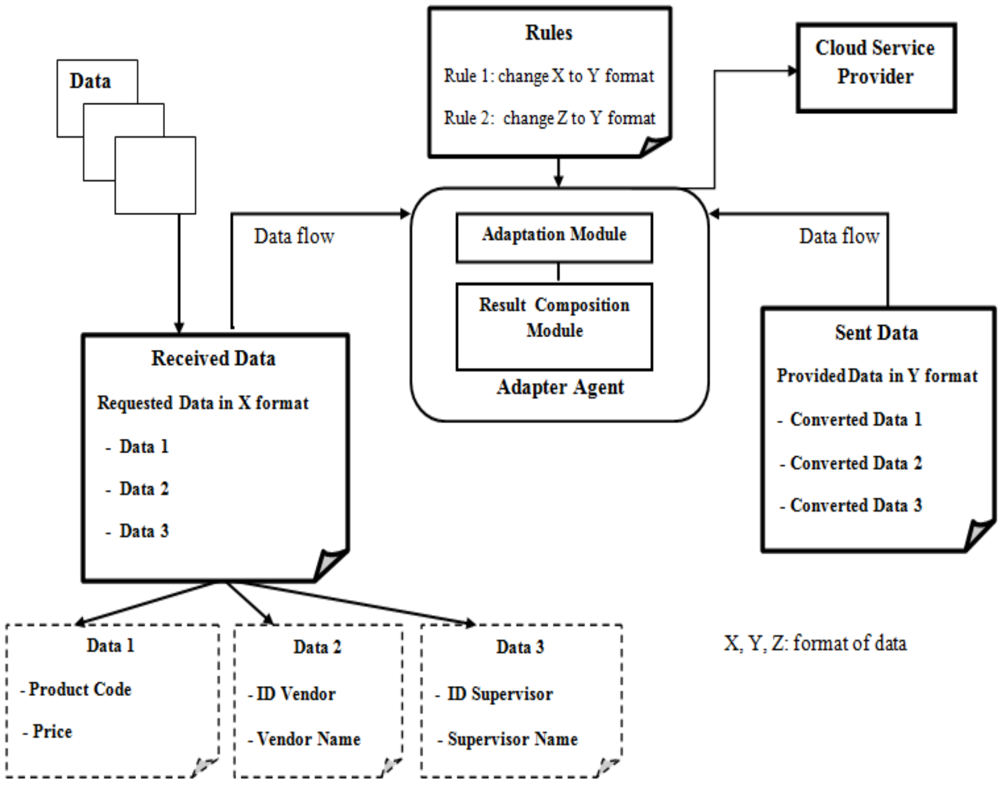


Figure 5 Adapter agent's actions



5 Data flow management using the adapter agent

After receiving the request (needed data) from the mobile agents (it can be more than one mobile agent depending on the demand), the adapter agent makes the necessary processing on data and sends the results (converted data) to the recipient provider, as shown in Figure 5. The structure of the adapter agent is mentioned in Thabet and Boufaida (2013). The adapter agent has two principal functions:

- *Extraction and adaptation of data*: through the adaptation module of the adapter agent. First, this latter extracts the data from the request and converts them to the format used by the database of the recipient provider (Provider B). Finally, it sends the data to the result composition module that will transmit them to the provider B.
- *Composition of results*: once the input data have been converted to the data model of the provider B, they are sent to the result composition module, which sorts and sends them to the provider B. This latter, executes and offers the requested services and sends the result to the coordinator agent that transmits the response to the company.

6 Application to a case study

In this section, we present a case study of interoperability between two cloud providers in order to create the dashboard of the company.

6.1 *Prototype implementation*

AUML modelling language is used to represent different levels of abstraction. During the design of class diagram, two levels of abstraction are commonly used: conceptual and implementation level.

- *Conceptual level*: this level is a view of multi-agent system eliminating all the superficial information (attributes, operations), in order to understand the structure of the architecture.
- *Implementation level*: it provides the content of the classes (attributes and operations).

Figure 6 Class diagram (see online version for colours)

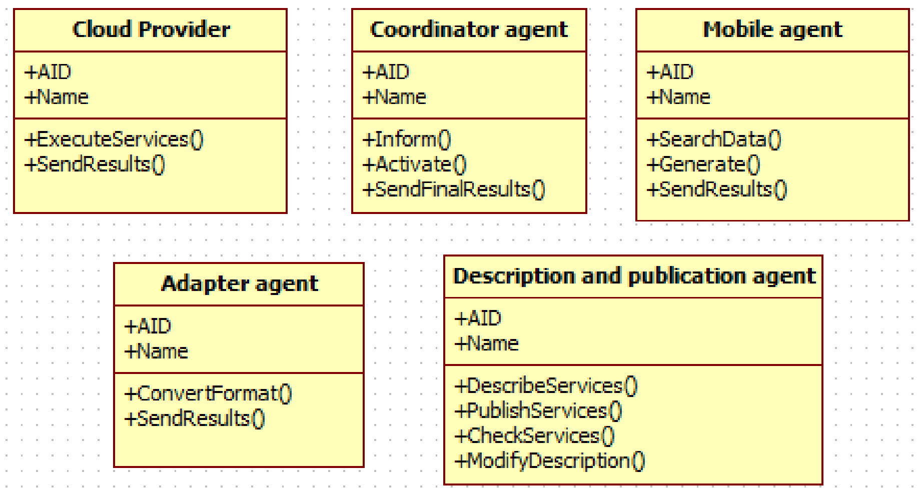
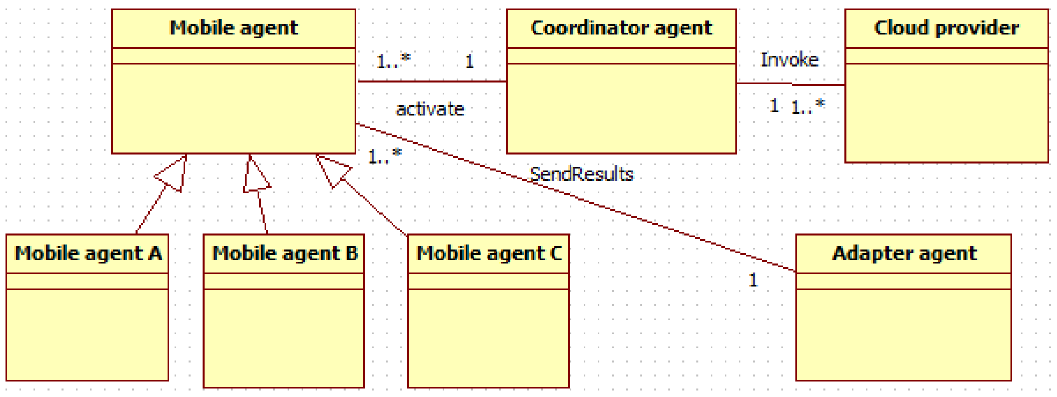


Figure 7 Classes used in our architecture (see online version for colours)



6.2 *The case study*

We illustrate our solution via a case study of using two cloud service providers. Our contribution is based on the achievement of the interoperability at the software as a service (SaaS) level. In our case, the end-user is a company that needs to choose different providers in order to realise its dashboards.

In fact, it is reasonable that companies use multiple cloud providers to satisfy their purposes and customers' needs, in order to perfect their offerings and develop themselves. But the problem that arises is where most of the companies will end up with heterogeneous environment of different types of cloud providers. So, the common linking thread among these disparate cloud services will be the data. These last can be interchanged among heterogeneous clouds (Shahab, 2010).

In our case, the company uses the software of 'customer relationship management' (CRM), 'human resources' (RH) of the cloud provider A and the dashboard software of the cloud provider B, in order to create its dashboards for sales tracking (CRM) and monitoring of absenteeism (RH). To do so, the company needs its data in the provider A. In other words, this data must be imported by the provider A and integrated in the cloud provider B.

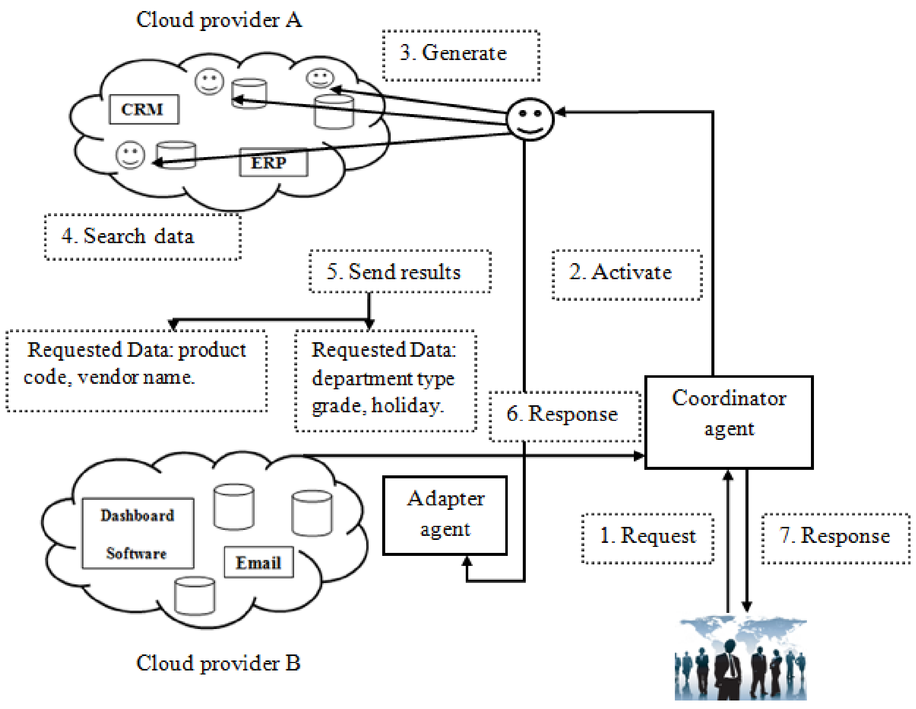
In this instance, the invocation of providers A and B should be ordered. It should be managed and provided by the coordinator agent. It should also be performed in a specific order. First, the provider B must be invoked because it needs data of the provider A that will be invoked later. So, the provider B invokes the coordinator agent by a request including the needed data to integrate them and offer the required services.

After having received the request by the provider B, the coordinator agent activates the mobile agent of the provider A that migrates within the cloud to accomplish its task. This agent searches, collects, filters the requested data and sends them to the adapter agent.

In order to create the first dashboard, the mobile agent must search the data that concern all the products like: product code, description of products, and all the information that has a relation with vendor: commands, price, sale date and current stock level. For the second dashboard, the mobile agent searches the needed data, which are always sought by the provider B, like the personnel data: age, sex, grade, holiday and their working time.

Indeed, our mobile agent generates other agents to speed and facilitate the research of data, to process the requests of clients as soon as possible. After that, it sends a direct message to the adapter agent of the provider B. This message includes a document containing the required data. In the example below, we show how the mobile agent is used to ensure the data portability. Our interaction diagram is inspired from Rabia and Amjad (2012).

Figure 8 Interactions between the mobile agent, the coordinator agent and the adapter one (see online version for colours)



In this instance, we have based the realisation of our approach on a case study that we did by integrating agents, with using database management systems (DBMS): the relational MySQL cluster database for the provider A and the column family database (HBase) for the provider B. These databases are heterogonous. Each database has its own structure and type of data. To do so, the adapter agent is used to solve the problem of heterogeneity and to facilitate data integration by the cloud provider B.

Table 1 Different DBMS used in the study case

Cloud service providers	*DBMS*	*Data model*	*Exchange function*
Provider A (CRM, RH)	MySQL cluster	Relational	MySQL cluster to HBase
Provider B (dashboard software)	HBase	Column family	/

As mentioned in Table 1, the provider A uses the MySQL Cluster database to store all information concerning products, vendors and so on.

The database used in the provider A is called 'Provider A' that contains several tables, which represent all the information such as: products, customers, vendors.

In our case, we rely on the tables: Product, Catalogue and ControlSupervisor.

The table 'Product' concerns all the data about product like: ID-product, price, quantity, description and so on.

The table 'ControlSupervisor' concerns all the data 'about supervisor like: ID-supervisor, supervisor name, report and so on.

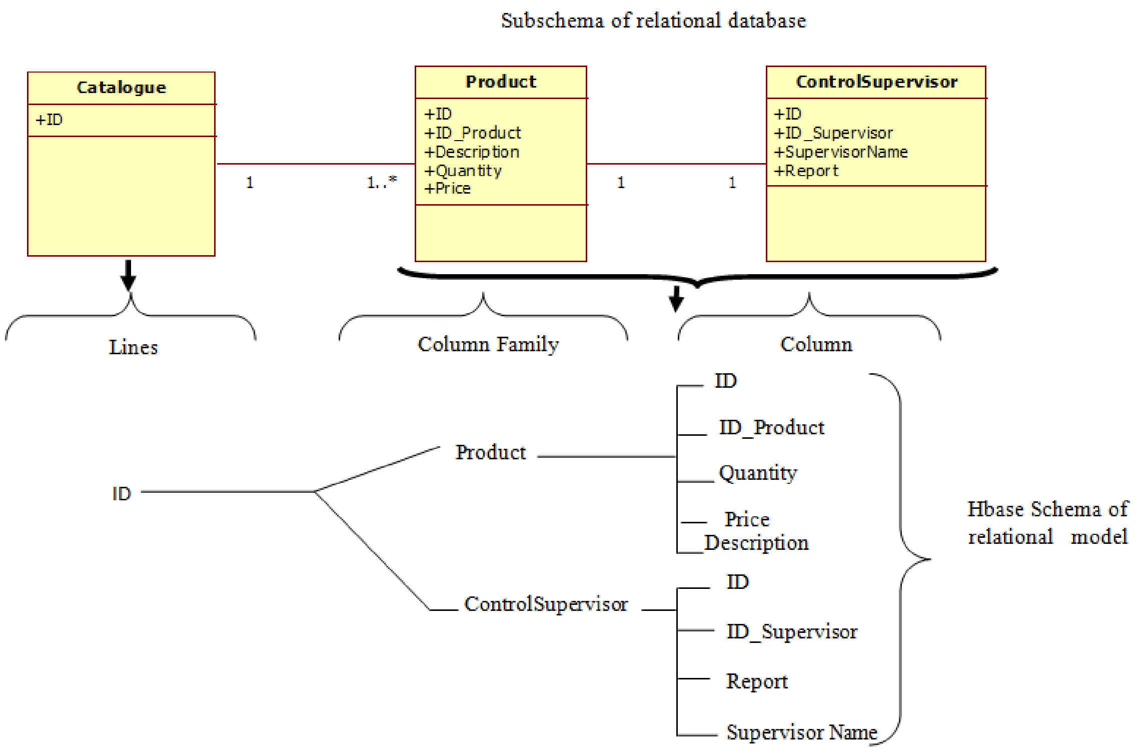
The table 'Catalogue' is used for saving all the identifiers of products that are already controlled by the supervisor. In other words, to know the products that are sold and those that are stored.

Indeed, the relation between 'Catalogue' and 'Product' tables is 'one to many because each catalogue is made for multiple products, while the relation between 'Product' and 'ControlSupervisor' tables is 'one to one because each product has one and only one control per day.

In order to execute the service of dashboard for Sales Tracking by the provider B, we have to migrate the subschema, which is composed of the tables 'Product' and 'ControlSupervisor' of the MySQL Cluster database to the Hbase database. So that the company can achieve its dashboard. Figure 9 shows an example that illustrates the performed conversions between two cloud providers when a migration of data occurs. It presents a conversion of subschema of the relational database to the column family database.

According to this conversion, we note that the linked tables in the relational model with relation 'one to one' are transformed into a column family, the operations are transformed into a column in HBase and each primary key (ID) is transformed into a line.

Figure 9 HBase conceptual model of relational schema (see online version for colours)



6.3 Using the implementation of the case study

In this section, we simulate the migration protocol of the mobile agent. In order to implement our protocol, we should use an agent-based platform that permits the implementation of their different agents. Several platforms are supplied as software packages, such as JADE, ZEUS.

In our work, we use the JADE platform (Bellifemine et al., 1999) because of its known facilities. Among them we mention the following (Fabio et al., 2003):

- This platform can be split among several hosts. Only one Java application and therefore only one Java Virtual Machine, is executed on each host.
- The JADE platform provides security mechanisms for its proper use.
- The agents are implemented as Java threads and live within agent containers that provide the runtime support to the agent execution.
- The platform offers an efficient transport of ACL messages inside the same agent platform.
- It provides a library of FIPA interaction protocols.

In JADE (Bellifemine et al., 1999), the agents are represented as instances of the Java class. Each java class is an extension of the basic agent class (included in jade.core). For our protocol, we define an interoperability package that includes two classes: Coordinator and Mobile.

1 the coordinator class that describes the coordinator agent behaviour, which initiates the first phase of the migration protocol

2 the mobile class that describes the mobile agent behaviour, which performs the second phase related to the migration protocol.

Regarding the actions of agents defining their behaviours, these last are described by *action method* of behaviours class. Figure 10 shows the skeleton of the myAgent class that can be a coordinator class, or a mobile class.

We create the different agents in the main container. All the messages sent and received by the agents can be captured and displayed on the interface 'Introspector'. Figure 11 presents an example of the agent Introspector interface that shows the messages received by the coordinator and the mobile agent.

In our architecture, we use cognitive agents that are able to plan their behaviours and remember their actions, in order to evaluate the offers that are available to them.

As a matter of fact, we distinguish two communication modes: indirect communication that is done by transmitting signals through the environment and direct communication, which permits the exchange of messages among agents. We focus on a multi-agent system in which the interaction is direct. The communication among agents is done through the ACL messages. We remind that the JADE platform uses FIPA-ACL as a language of inter-agent communication. The ACL messages exchanged are represented in Figure 12.

Figure 10 Extension of the agent and behaviours classes

```
import jade.core.Agent;
import jade.core.behaviours.*;
public class myAgent extends Agent
{
//…myAgent could be Coordinator or Mobile
protected void setup()
{
addBehaviour(new myBehaviour(this) );
}
class myBehaviour extends SimpleBehaviour
{
public myBehaviour(Agent a)
{
super(a);
}
public void action()
{
//...this is where the real programming goes!!
}
private boolean finished = false;
public boolean done()
{
return finished;
}
} // ---------- End myBehaviour
}//end class myAgent
```

We use the performatives given below, in order to implement the two-phase migration protocol (activation and migration)

1 Performatives used by the coordinator agent:

- *Request*: it corresponds to the request of the company
- *Request whenever*: it is used by the coordinator agent to require the discovery module to perform a research about the requested services in the registries. It corresponds to the message *search* (*service*).
- *Inform*: it is used by the coordinator agent to inform the mobile agent that it has been selected to search data. It corresponds to the message *activate* (*API*, *requested data*, *MA*).

2 Performatives used by the mobile agent:

- *Confirm*: it is used by the mobile agent to confirm that it has received the request of the coordinator agent.
- *Propagate*: it is used by the mobile agent to generate other mobile agents. It corresponds to the message *generate* (*MA*).
- *Request*: it describes a response containing all the data searched by the mobile agent. It corresponds to the message *send* (*result, AA*).
- *Confirm*: it is used by the adapter agent for the acknowledgement of the result sent by the mobile agent (Required data).

Figure 11 Agent introspector (see online version for colours)

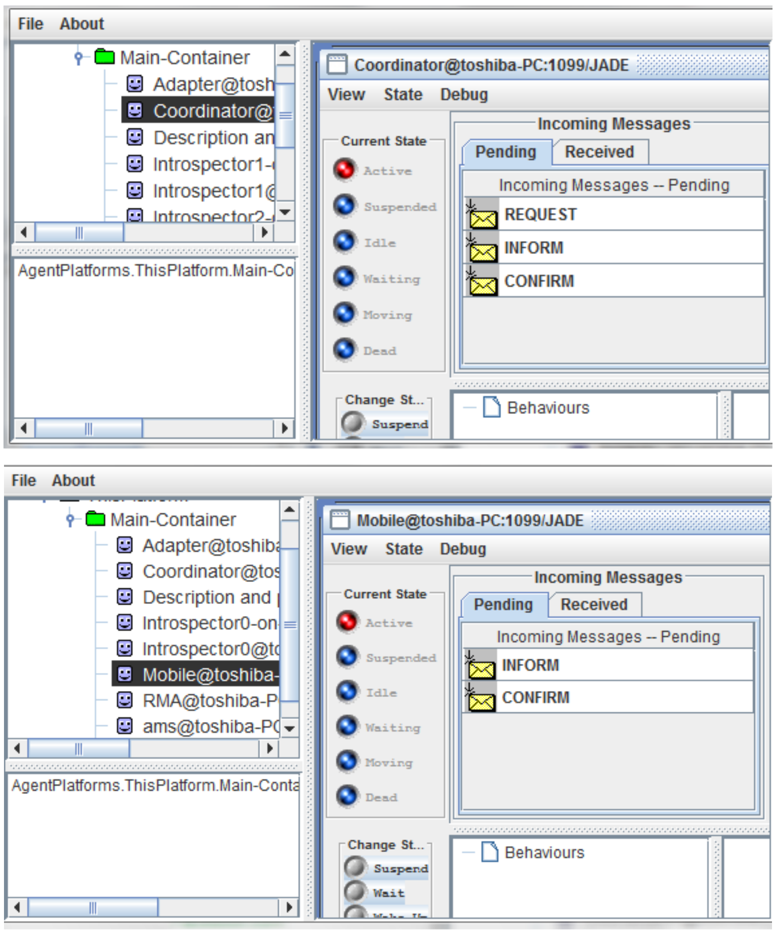
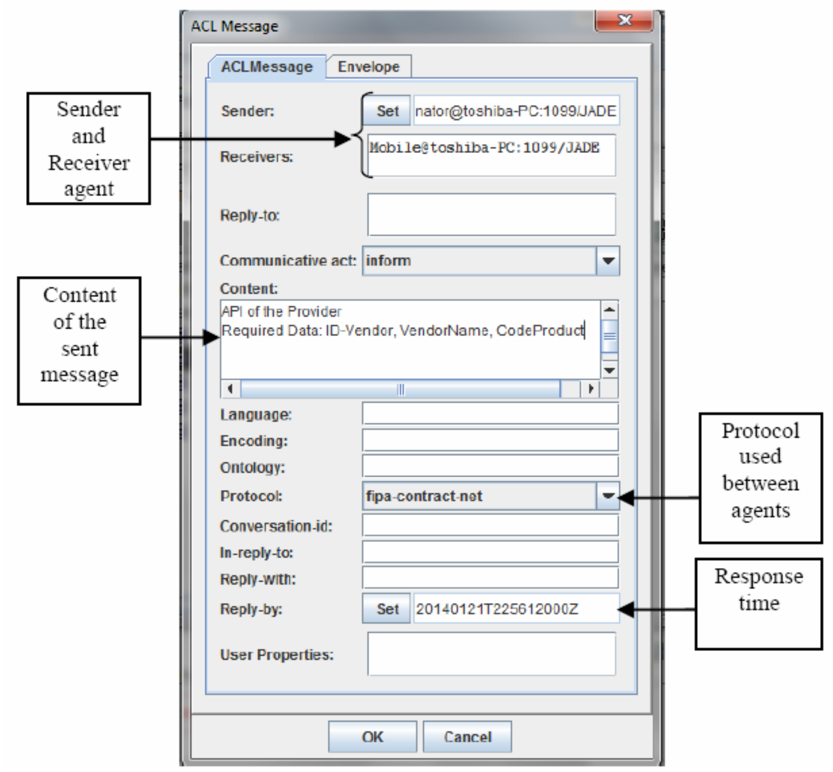


Figure 12 ACL message in JADE (see online version for colours)


ACL Message
ACLMessage Envelope
Set
Set
OK Cancel
Sender and Receiver agent
Content of the sent message
Protocol used between agents
Response time

Sender:	nator@toshiba-PC:1099/JADE
Receivers:	Mobile@toshiba-PC:1099/JADE
Reply-to:	
Communicative act:	inform
Content:	API of the Provider Required Data: ID-Vendor, VendorName, CodeProduct
Language:	
Encoding:	
Ontology:	
Protocol:	fipa-contract-net
Conversation-id:	
In-reply-to:	
Reply-with:	
Reply-by:	20140121T225612000Z
User Properties:	

7 Related work

Various studies have been made to improve the interoperability of clouds. An important part of these works has been investigated in cloud portability, focusing on the use of mobile agent at infrastructure as a service (IaaS) level (Zhang and Zhan, 2009). The mobile agent-based open cloud computing federation (MABOCCF) introduces a travelling bag mechanism in which user's tasks are encapsulated in a mobile agent, this latter carry user's tasks and move among sites in one cloud service provider, or different clouds in order to execute them on the virtual machines and to realise interoperability and portability among suppliers.

Shirazi et al. (2012) present a new technique for enabling portability by using the technology of design patterns, which provides a solution for enabling portability between column family databases and graph databases. The design pattern of data portability can be achieved by providing a way to transfer data from one cloud to another.

Loutas et al. (2011) have introduced the use of semantic web technologies in cloud computing. They propose a framework, which aims at resolving semantic interoperability conflicts that may be raised during the deployment or the migration of an application. This framework defines a three-dimensional space: fundamental platform as a service (PaaS) entities, types of semantics and levels of conflicts.

Ejarque et al. (2011) provide an approach based on a methodology in which the APIs of the suppliers are automatically modelled in ontologies. These last describe the provided functionality and the used data.

Takabi et al. (2012) propose a semantic-based policy management framework that is designed to provide to users a unified control point for managing policies. This control accesses to their data no matter where the data is stored using semantic web technologies.

Bessis et al. (2012) present the fundamental issues for developing an effective interoperable meta-scheduler for e-infrastructures in general and inter-cloud in particular, which offers additional functionalities in the area of interoperable resource management. This is because of its great agility to handle sudden variations and dynamic situations in user demands.

Some studies address the problem of interoperability by providing open standards. Bernstein et al. (2009) focus on an inter-cloud communication by introducing several problems and their solutions. The result of their work is a set of protocols, which can be used to achieve an inter-cloud communication. In their work, the use of open standards is essential in order to achieve interoperability. However, this approach focuses only on the communication part, but it does not consider current trends or other important topics, such as virtual appliance, storage and service level agreement (SLA).

Demchenko et al. (2012) propose an extensible simulation-based framework to evaluate cloud service brokers deployed on an inter-cloud environment. The main feature of the framework is the integration of several technologies and standards, which makes it easy to deploy on real production clouds.

Jrad et al. (2012) present an on-going research at the University of Amsterdam to develop the inter-cloud architecture (ICA). The ICA addresses the problem of multi-domain heterogeneous cloud-based applications integration. This architecture intends to provide a basis for building multilayer cloud services integration framework and to allow an optimised provisioning of computing, storage and networking resources.

Another research work has been made to ensure database migration to cloud computing, Belghati (2011) shows in his work the software industry's interest to this kind of projects. This work has allowed exploring the possibility of using an original hybrid model, which may be used to consider the potential of a new data model on cloud computing platforms.

Most of the research works cited have been made to ensure interoperability at IaaS model, while few studies have focused on other service models such as PaaS and SaaS model, these models remain a big challenge to achieve cloud interoperability.

In this work, we have treated the problem of data portability between suppliers and we have based the realisation of our case study at SaaS level. Moreover, there is a research work (MABOCCF) somehow similar to our approach but there are differences between them:

- MABOCCF uses the mobile agent to ensure portability at IaaS level, while we are using it to ensure data portability at SaaS level
- MABOCCF uses only one kind of agent to achieve portability and interoperability, while we are using four kinds of agents to achieve data portability and interoperability
- MABOCCF uses the mobile agent on the client side, while we are using it on the provider side
- Mobile agents used in MABOCCF migrate to run user's task on different virtual machines, while our mobile agents are used to migrate among clouds in order to transfer data.

We claim that the common point between our solution and the MABOCCF solution is the use of the mobile agent paradigm, in order to achieve the same objective, which is interoperability but on two different levels for two kinds of portability.

8 Conclusions

Today, there is no common standard for cloud computing. Hence, the interoperability will be difficult to realise. We proposed in this paper an agent-based architecture, which aims to realise the data portability and the interoperability among different cloud providers. The strength of our contribution does not reside in the architecture complexity of an agent, but in the interactions among different agents and their roles. Another important improvement of our system with regards to other approaches is that we are not bound to any data format, which enables our multi agent system to support different interoperability scenarios. For instance, the companies could choose any given cloud service provider and our system will translate the data format to the required format from the requestor. Currently, our approach deals only with syntactic heterogeneity. However, in the future it will be interesting to take into consideration the semantic heterogeneity of the exchanged data, in order to ensure a good understanding and a proper interpretation of the data. To resolve this semantic heterogeneity, new components must be added to the proposed architecture for ensuring the semantic interoperability and for improving the communication among providers.

References

Belghati, F. (2011) *Exploitation de la migration de la base de donnée relationnelle du système de gestion de processus d'affaire oryx vers la base de donnée No-SQL utilisée par la plateforme de l'informatique de nuage Hadoop*, Dissertation presented at Software Engineering Department of Superior Technical School, pp.1–96, Montreal , Canada.

Bellifemine, F., Poggi, A. and Rimassa, G. (1999) 'Jade: a fipa-compliant agent framework', in *Proc. PAAM '99*, pp.97–108, London.

Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S. and Morrow, M. (2009) 'Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability', in *Proceedings of the 4th International Conference on Internet and Web Applications and Services (ICIW)*, pp.328–336, Venice, Italy.

Bessis, N., Sotiriadis, S., Xhafa, F., Pop, F. and Cristea, V. (2012) 'Meta-scheduling issues in interoperable HPCs, grids and clouds', *Int. J. of Web and Grid Services*, Vol. 8, No. 2, pp.153–172.

Demchenko, Y., Ngo, C., Makkes, M.X., Strijkers, R. and Cees, D.L. (2012) 'Defining inter-cloud architecture for interoperability and integration', *The Third International Conference on Cloud Computing, Grids, and Virtualization*, pp.174–180, IARIA, Nice, France.

Ejarque, J., lvarez, J.A., Sirvent, R. and Badia, R.M. (2011) 'A rule-based approach for infrastructure providers interoperability', *Third IEEE International Conference on Coud Computing Technology and Science*, pp.272–279, Athens, Greece.

Fabio, B., Giovanni, C., Tizianna, T. and Giovanni, R. (2003) *Jade Programmers Guide*, University of Parma, Italy.

Ishak, I., Philippe, C. and Bernard, A. (2010) *Architecture distribuée interopérable pour la gestion des projets multi-sites. Application à la planification des activités de production*, PhD thesis, National Polytechnic Institute of Toulouse, France.

Joe, M. (2010) 'Does platform as a service have interoperability issues?' [online] http://www.zdnet.com/blog/service-oriented/doesplatform-as-a-service-have-interoperability-issues/4890.

Jrad, F., Tao, J. and Streit, A. (2012) 'Simulation-based evaluation of an intercloud service broker', *Third International Conference on Cloud Computing, Grids, and Virtualization*, pp.140–145, IARIA, Nice, France.

Kim, W. (2009) 'Cloud computing: today and tomorrow', *Journal of Object Technology*, Vol. 8, No. 1, pp.65–72.

Loutas, N., Kamateri, E. and Tarabanis, K. (2011) 'A semantic interoperability framework for cloud platform as a service', *Cloudcom*, *IEEE Third International Conference on Cloud Computing Technology and Science*, pp.280–287, Athens, Greece.

Rabia, K. and Amjad, M. (2012) 'Realization of interoperability & portability among open clouds by using agent's mobility & intelligence', *International Journal of Multidisciplinary Sciences and Engineering*, Vol. 3, No. 7, pp.7–11.

Shahab, A. (2010) *Data Portability: Key to Cloud Interoperability* [online] SSRN: http://ssrn.com/ abstract=1712565 or http://dx.doi.org/10.2139/ ssrn.1712565, pp.1–12.

Sheth, A. and Ranabahu, A. and (2010a) 'Semantics centric solutions for application and data portability in cloud computing', *2nd IEEE International Conference on Cloud Computing Technology and Science*, pp.234–241, Indianapolis, Indiana.

Sheth, A. and Ranabahu, A. (2010b) 'Semantic modeling for cloud computing, Part I & II'*, IEEE Internet Computing Magazine*, Vol. 14, pp.81–83.

Shirazi, M.N., Kuan, H.Ch. and Dolatabadi, H. (2012) 'Design patterns to enable data portability between clouds' databases', *12th IEEE International Conference on Computational Science and its Applications*, pp.117–120, Salvador.

Takabi, H., James, B.D. and Joshi, J.B.D. (2012) 'Semantic-based policy management for cloud computing environments', *Int. J. of Cloud Computing*, Vol. 1, Nos. 2/3, pp.119–144.

Thabet, M. and Boufaida, M. (2013) 'An agent-based architecture and a two-phase protocol for the data portability in clouds', *27th IEEE International Conference on Advanced Information Networking and Applications Workshop*, pp.785–790, Barcelona, Spain.

Zhang, Z. and Zhang, X. (2009) 'Realization of open cloud computing federation based on mobile agent', *IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS*, pp.642–646, Shanghai, China.