A Stepwise Self-adaptive Model for Improving Cloud Efficiency Based on Multi-Agent Features

Amr Tolba[1, 3*], Ahmed Ghoneim[2, 3]

[1] Riyadh Community College, King Saud University, Saudi Arabia.
[2] College of Computers and Information Sciences, King Saud University, Saudi Arabia.
[3] Mathematics and Computer Science Department , Faculty of Science, Menoufia University, Shebin El-Kom, Egypt.

* Corresponding author. Tel : +966506511223; email: atolba@ksu.edu.sa.
Manuscript submitted January 10, 2015; accepted June 8, 2015.
doi: 10.17706/jsw.10.8.1037-1044

Abstract: Today, the multi-agent systems are the most common systems that have intelligent behavior and able to adapt features based on the environmental changes. On the other side, cloud applications enable the stakeholders to customize their resources and software they need based on the requested domain. These applications face many challenges such as how to handle the changes of the stakeholder requirements at run-time, how to reconfigure the constituted architecture dynamically to be in consistency with the new services, and how to cope with the highly inherent expensive cost. To deal with these challenges, we proposed a new model that uses basic agent features for enhancing the cloud infrastructure functionalities by reconfiguring their allocated resources and software at run-time. The proposed model is composed of three levels. The first level is the cloud level which via its functionalities the consistent image that maps the user's requests through its manual components can be created & established. The second level is the intermediate level, which is responsible for two issues: playing the role of connector between the cloud level and the multi-agent level, and verifying the consistent of outputs for both of the upper and lower levels. The third level is the multi-agent, which is responsible for improving the quality of the constituted cloud images by co-operating the information, reasoning, learning and mobile agents. Finally, the urban transportation system is used to proof the applicability and usage of the proposed model.

Key words: Cloud computing, multi-agent system (MAS), resource allocation.

1. Introduction

Technologies such as grid computing, cluster computing, and cloud computing, are designed to allow access to large amounts of computing in the power of entire virtual manner through the pooling of resources and providing a single-vision system. It is worth mentioning that providing computing as a service is considered one of the most important goals of using the said technologies. Cloud is a large group of virtual resources which are characterized by its practical usability & easy accessibility (such as hardware, software development and / or services). These resources dynamically reconfigured to adapt its scale, and to get the optimal use of their structure and behavior.

Amongst the most demanding difficulties that the merging cloud computing environments & multi- agent systems have to deal with are the intelligent resources, the independent & dynamic adaptation, the condition changes and the effective visualization. The cloud computing is considered a highly favorable

active trend in the world of today because of its edges as regards trustworthiness, anticipation and susceptibility of expansion of computing infrastructure which is required for establishing & finalizing the agent-based systems. In addition to that the software agents help cloud computing systems to be more accommodated, manageable and autonomous in resource management, service provisioning and in operating huge applications and programs [1].

The software agent is defined as a computer entity that is equipped with miscellaneous distinguished traits such as intelligence, dynamics, reasoning, autonomous and dependability which help together at enhancement of inter-cooperation among other agents and carrying out its own targets by itself. The idea of the cloud computing relies on sharing the services and information among network nodes. The primary objective behind using the concept of cloud environment is to set and combine together the whole requested services in one single place (the cloud) with a view to enabling the users/nodes to come together into one access that they can approach at any time or at any place [2], [3].

The rapid growth of technologies as well as the increasing number of users enforce cloud computing systems to provide large-scale infrastructures for high performance computing that can adapt to user and application needs. This cloud infrastructure can be integrated with software agents that have intelligence, dynamics, reasoning, and autonomous features, and make these agents take a proper decision at run time in an automatic way. Therefore, adding software agents to cloud computing environment will increase the cloud performance and making them more robust, flexible and autonomic.

In this paper, we have proposed a new cloud framework by employing the self-governing & learning traits of the software agents to enable the reconfiguration of the behavior of both allocated resources and software at run-time. The output of the proposed framework was fully coherent resources and software image.

The rest of the paper is organized as follows: Section 2 describes the proposed approach in details. Section 3 illustrates the Urban Transportation system as a case study. Section 4 presents and discusses related works. Finally, Section 5 concludes the paper.

2. Proposed Approach

In this section, we introduced our proposed model that improves the quality of the cloud infrastructure using multi-agents system. The model is composed of three levels as shown in Fig. 1.

The first level named cloud level; is about the operational level of the model, which has direct connection with the group of users through its representative agent (user agent). Cloud infrastructure enables the user to choose his required resources and software, after that constitute the matched image, then pass the execution to the view controller who can use the agent feature to dynamically adapt both the structure and behavior of the generated image based on the agent capability and the run-time changes (user changes or environmental changes). The functionality of the cloud level has been fulfilled by using the following components: First, the cloud user service GUI is directly in contact with the user agent for detecting and extracting the user's request or views. Second, the message adapter is responsible for parsing the received requests and generates XML-based message that covers the original requests. Third, allocate/reallocate resources component is responsible for identifying the suitable resources that match user requirements after that allocate/reallocate software component is responsible for detecting required software to match the user's software requirements. Fourth, establish and reestablish the communication between the resources and software. Finally constitute a consistent image that includes the match resources and software. The constituted image generated by the component Generated User views. Note that the allocate function of the resources and software is used for the normal flow for preparing the image, and the reallocate function is used to enhance the image based on the reflective feedback for the multi-agent level.

The second level named intermediate and control level. Its main functionality, which is recognized in the shape of a middleware layer component, is to guarantee the connection between the upper & higher levels and the user agent. It consists of two components (agent interface and view controller). Agent interface has direct connection with the following: multi-agent level through its agent bus, the user agent, the view control component and cloud level. This agent is responsible also for creating the higher message with the same format for multi-agent level. The view controller component starts its process by checking the usability and consistency of this view. In addition, it checks the satisfactory level of these views. The output after processing these components is a request for changes that includes both the user feedbacks and the inconsistency issues. Moreover, it reconfigures both the constituted image and views based on the generated request of changes and facing environmental changes.

The third level named multi-agent level. This level is responsible for improving the performance of the cloud image using the agent features. The main interface of this level is the agent bus interface that integrates with all the other components in this level. The level has four components. The reasoning agent used for applying the reasoning techniques for software image comprehension. The information agent is responsible for extracting the inconsistent paths within the image and collecting the suitable alternative information using the functionality of the reasoning agent. Learning agent is responsible for representing the rule engine for this level based on the best practice of the experience of dealing with the set of similar software images. Mobile agent is responsible for searching to identify different solutions within different applications outside or within the domain by using its mobility features. The control flow that illustrates the core model components functionalities is shown in Table 1.

Table 1: Pseudo Code: Procedure Flow of the Proposed Model

```
Inputs:          User request, User profile.
Out puts:     Cloud Image
BEGIN- Main
            USER_GUI (user_login);
            USER_Agent (select (HW, SW));
            Message_adapter(extract, HW-Specifications);
            Message_adapter(extract, SW-Specifications);
          Call Cloud_Bus (HW-Specifications, SW-    Specifications, OutPut: Cloud Image)
            For Each Cloud Image Component
                  Generated_User_View (GUV. generates (stractural_view);
                  Generated_User_View (GUV. generates (behavioral_view);
             EndFor
             For Each View
                  View_Controller (VC.generate(request_of_changes, Inconsistency));
                  View_Controller (VC.generate(request_of_changes, user_statisfactory_level));
                  Merge_request_of_changes(Inconsistency, User_satisfactory_level); // for each view
             EndFor
        While request_of_changes exists Do
            Agent_interface (request_of_changes, current image, OutPut: Up-to-date Cloud Image);
            End Do
            For Each changes
              Reasoning_Agent(request_of_changes, current image; Apply(AI_reasoning
                                  techniques));
              information_Agent(request_of_changes, current image; Extract(bad smile parts ));
              Learning_Agent(request_of_changes, current image; Add (new rules, rule engine of
                              this level));
              Mobile_Agent(request_of_changes, current image; Search (? Solution, mobile
                                                  featues(imagration)));
            END For
END- Main
Subroutine Cloud_Bus(HW, SW, OutPut: Cloud_Image)
   BEGIN- subroutine
              Allocate_HW(HW, consistent_HW);
              Allocate_SW(SW, consistent_SW);
              Constitute_Image( consistent_HW, consistent_SW, OutPut: Cloud_Image);
   END- subroutine
```

The structural components of the proposed MAS-Cloud model are shown in Fig. 2. This figure represents the core model classes such as UserAgent, CloudGUI, CImageAllocator, CImageGenerator, ViewHundler, ViewController, AgentDispather, MobileAgent, LearningAgent, ReasoningAgent, and InformationAgent. All these classes operate together to create the image that match the user request. Moreover, the Agent dispatcher and its related agents are responsible for adapting the created image to face the environmental changes.

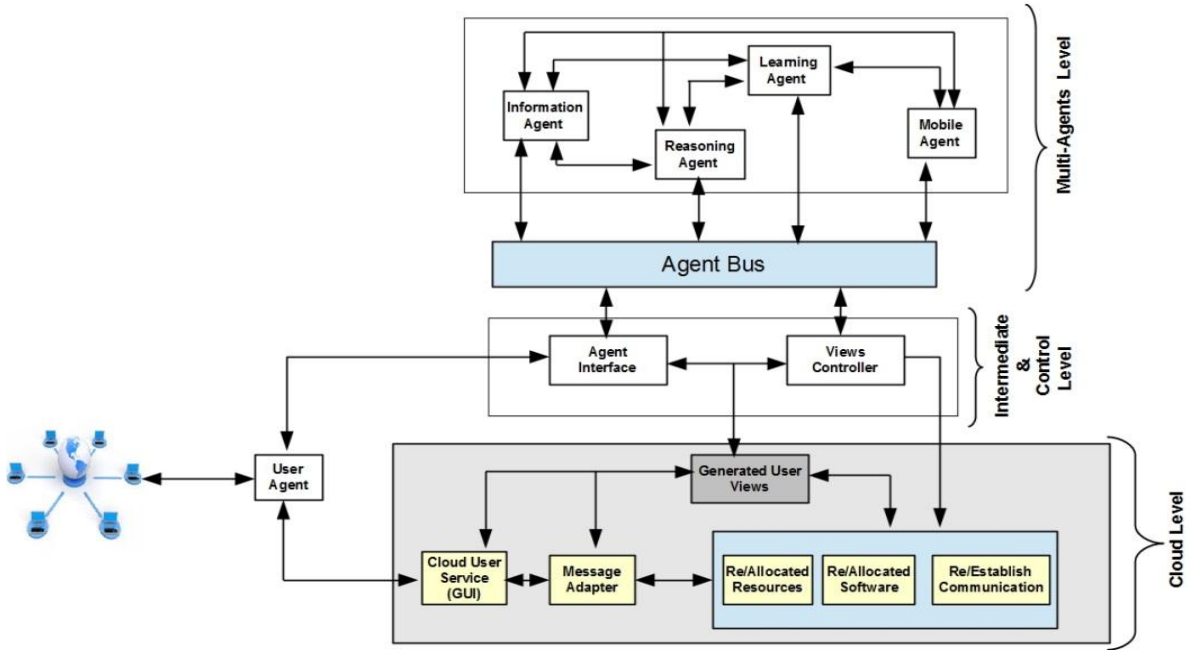

Fig. 1. Proposed MAS-cloud model.

3. Case Study: An Urban Cloud for Transportation System

In this section, we introduce the capability and the usage of the proposed model using urban transportation system as an agent system works on cloud environment. The user uses its mobile application to select his destination. This urban cloud simulator is running into Google Application Engine (GAE) [4], [5], which builds the blackboard for all client requests and uses Java Network Launch Protocol (JNLP) Link to launch the current location for each client and all necessary java applications within the GAE.

In Fig. 3, a part of the static functions of the model is presented as use cases. This diagram includes six use cases and four actors. The data flow of this diagram starts with the actor named “urban users” which able to register and add destination, then the system creates an urban image based on the user request by adding software and resources into this urban image. The constituted image can use the external resources and the external software to fulfill the user requirements. The last use case is named “adaptedCloudImage” that is responsible for adapting the image to face runtime change through the actor named environment sensor. This actor is able to cover all unusual events and pass them to this use case.

In the following we will illustrate overall scenario for the proposed model. The scenario starts by allowing the user agents to register, after valid registration the users have the ability to select or write their destinations as shown in Fig. 4. The cloud level within the model automatically builds urban cloud for user (UID-a203), by allocating all the necessary information related to resources and software's. The allocated resources and software’s are used to generate the path from the source to destination. The next step is to check the consistency of the allocated parts whether they are communicated by checking the Extensible Messaging and Presence Protocol (XMPP) messages between them. The generated path includes

information's getting from google earth system, tourist guide system, restaurant guide system and hospital system. The impact of the third level of the model for this case is to implement the rerouting algorithm to provide intelligent path. Moreover, the Multi agent level is using reasoning agents to dynamically recognize the best resources and software's to communicate with based on the change of resources and software status and the user changes his destination or asks for providing information about the historical places during the generated path. Finally, the urban user gets the created urban cloud. The created urban cloud dynamically reshaped based on the changes on the environment.

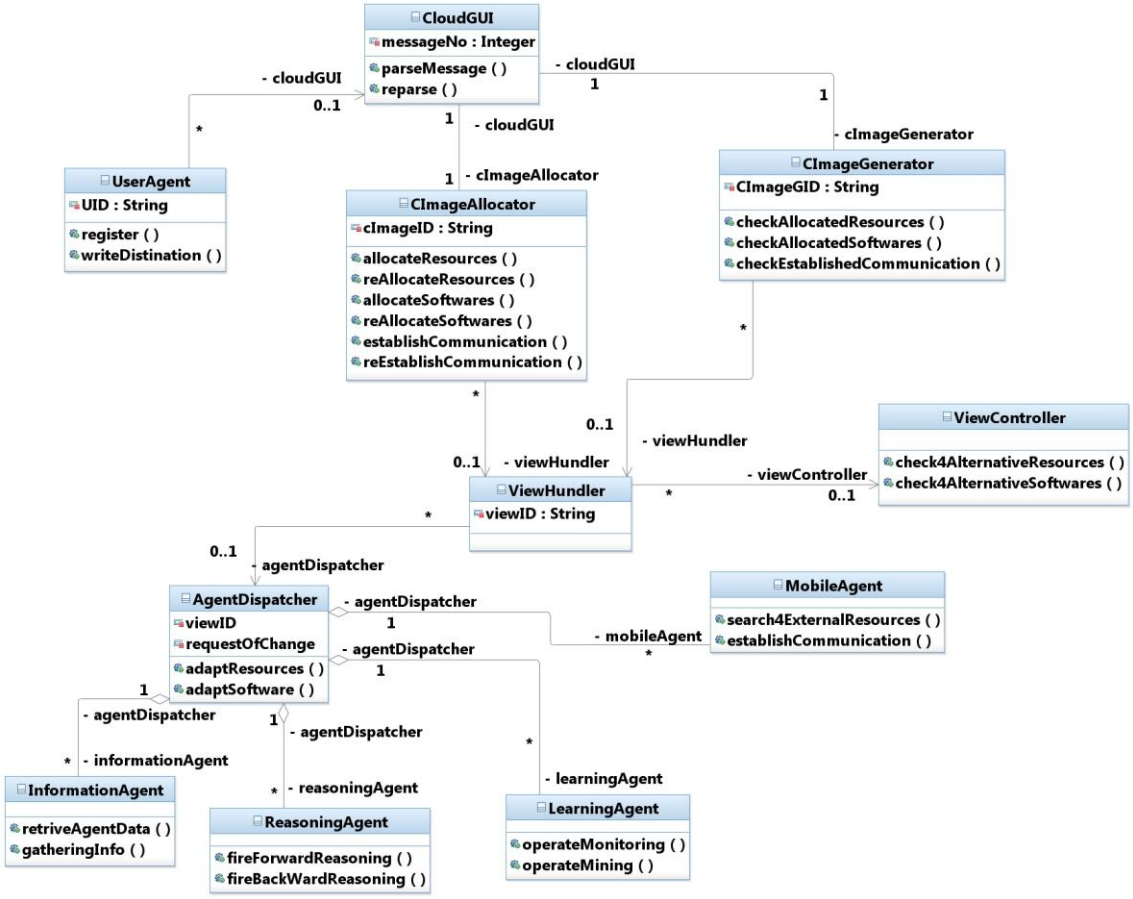

Fig. 2. Main components of the model — class diagram.

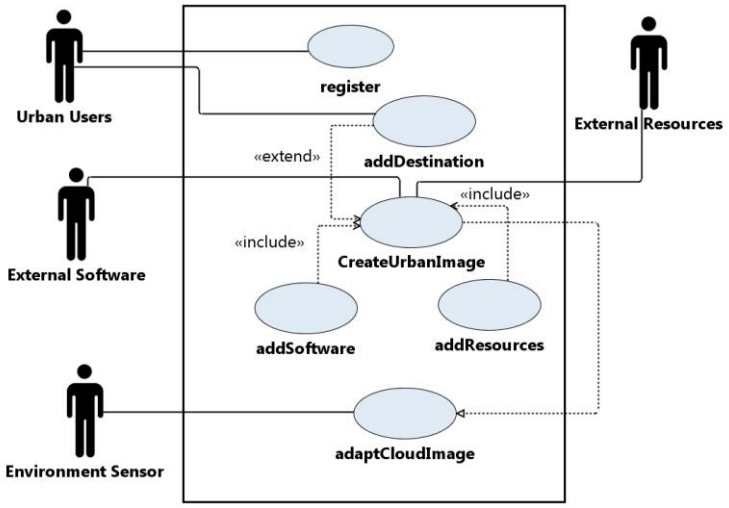

Fig. 3. UCTS use cases.

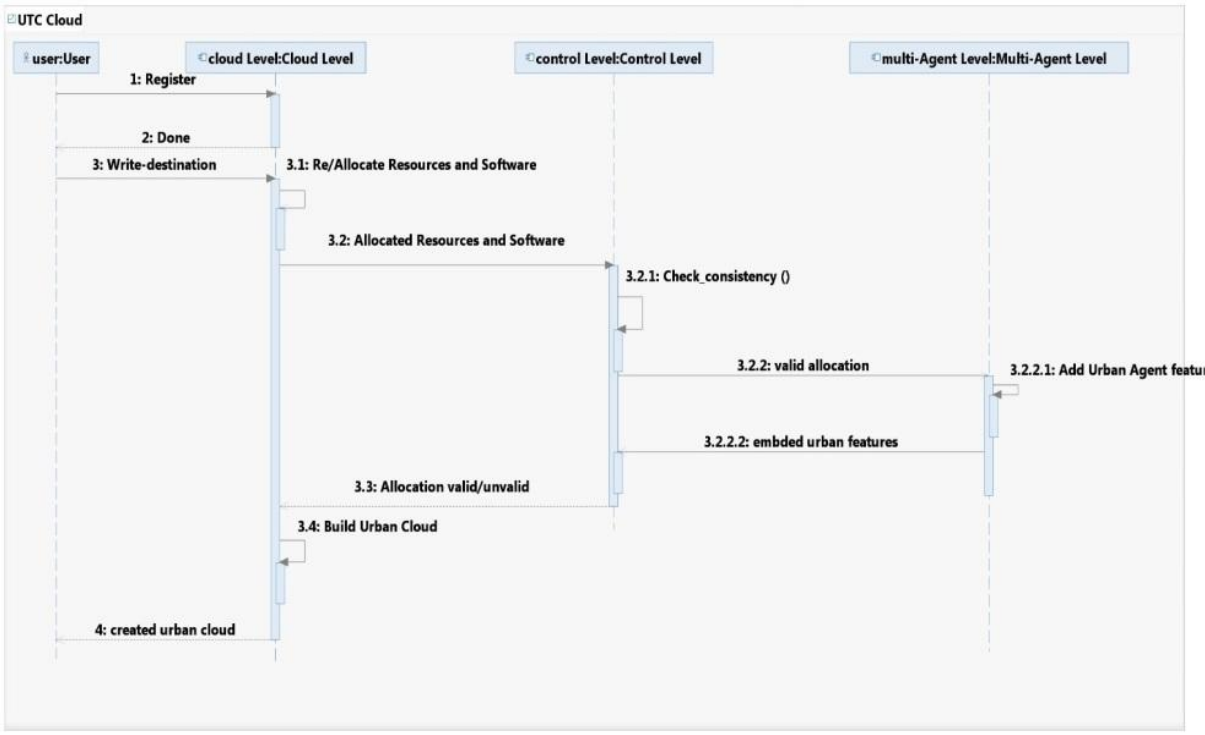

Fig. 4. Scenario for creating urban cloud.

4. Related Work

In the following, we will illustrate some related works that focus on embedding software agent into cloud computing environment. In our previous work [6], we proposed a cloud framework in which the autonomous and learning features of software agents were used to able to reconfigure the behavior of both allocated resources and software at run-time (IABCF Smarter). This Cloud framework composed of four main components: cloud infrastructure, agent features, view controller and agent adapter. In this paper we tried to couple agent features in the field of cloud computing to improve the behavioral attitude of the cloud. Vishal Jain *et al*. [7] proposed a model for the information retrieval using Multi-Agent System with Data Mining technique in a Cloud Computing environment. The multi software agent is used in a cloud environment to allow the users to retrieve information from virtual resources easily and with less infrastructure and storage cost. Uchibayashi, Toshihiro *et al*. [8] introduced a framework of an agent-based support system for discovering services in a public cloud. They use agents implemented by JADE only to measure network usage against user requirements information. Yue-Shan *et al* [9] proposed an ontology-based agent generation framework that allows users submit a request to the cloud and retrieve information from it. Messina, Fabrizio, *et al*. [10] proposed a protocol for supporting the SLA negotiation process, involving semantic issues which do not require the use of a common global agent ontology sharing knowledge in advance. In detail, this protocol considers that each agent is able to partition the other agents based on both their expertise in cloud services and similarity with their own ontologies.

Domenico Talia *et al*. [11] discussed some kind of coupling between software agents and clouds to produce high performance intelligent cloud service. Also, they discussed how to improve features for both cloud and agent, by making clouds more flexible and providing software agent a reliable environment to execute a large application. Myougnjin Kim, *et al*. [12] introduced another kind of direct relationship between software agents and cloud. They proposed an intelligent multi- agent model for resource Virtualization (IMAV) that automatically allocates suitable services to mobile node. Ignacio Lopez-Rodriguez, *et al* [13] introduced a framework where agents portrayed as a new cloud computing

services that will represent clients in a virtual environment. On the other hand our previous work [14] introduced an ontology-based cloud framework for discovering their external agent's interoperability (COBE framework). This framework designed based on blackboard design style and employs software agents to enhance the dynamic behavior of cloud computing environment.

Aarti S, *et al* [15], introduced a kind of coupling between software agent and cloud infrastructure for increasing scalability in cloud environment, the proposed model is supported by a set of algorithms that enable the system to search for resources and software in another cloud, inside or outside the domain, when the local cloud becomes overloaded or unavailable. Taekgyeong H. *et al* [16], proposed other kind of agent-based Cloud Service Discovery System (CSDS) that can be used to consult an ontology when gathering information about cloud services. A Cloud Service Reasoning Agent (CSRA) and Cloud Ontology (CO) are presented as new components that able to drive reasoning about the relations of cloud services using reasoning methods and rate the search results. The authors attempted to prove that CO makes the CSDS more successful in finding cloud services that meet users' requirements.

Based on the above related works, we can conclude that most of the above models and frameworks are implicitly structured and lack some components to control the constituted cloud environment. In addition to that, most of them lack of a method for facing and applying the runtime changes; this case implies us to adapt both the structure and behavior of the constituted cloud images.

5. Conclusion and Future Work

In this paper, we introduced the new model for improving the quality of the cloud image using the multi-agent features. The proposed model has three levels. The following summarizes the functions of the model: First constituting the cloud image to meet the user's request; then checking the image's consistency and linking the level and finally modifying the current image based on the multi-agent capabilities. The main component of the implemented classes is shown in the designed class diagram. In the future works we are planning to use the graph theory aspects to optimize the framework execution as we can improve the functionality of the multi-agent level by adding new agents and using refactoring techniques to improve the updated design of the cloud image within this level.

References

[1] Cao, B. Q., Li, B., & Xia, Q, M. (2009). A service-oriented qos-assured and multi-agent cloud computing architecture. *Cloud Computing*. Heidelberg, Berlin: Springer.

[2] Zhang, Q., Lu, C., & Raouf, B. (2010). Cloud computing: state- of-the-art and research challenges. *Journal of Internet Services and Applications*, *1(1)*, 7-18.

[3] Michael, A., Armando, F., Rean, G., A, D. J., Randy, K., Andy, K., Gunho, L., David, P., Ariel, R., Ion, S., & Matei, Z. (2010). A view of cloud computing. *Communications of the ACM*, *53(4)*, 50-58.

[4] Google Apps for Work. Retrieved, from https://cloud.google.com/appengine/

[5] Li, Z. J., Cheng, C., & Kai, W. (2011). Cloud computing for agent-based urban transportation systems. *Intelligent Systems*, 73-79.

[6] Amr, T., & Ahmed, G. (2012). IABCF smarter: An intelligent agent-based cloud framework. *International Journal of Computer Science and Telecommunications*, *3(40)*.

[7] Vishal, J., Mahesh, & Kumar, M. (2012). Information retrieval through multi-agent system with data mining in cloud computing. *International Journal of Computer Technology and Applications*, *3(1)*, 62-66.

[8] Uchibayashi, T., Bernady, O. A., & Norio, S. (2013). A framework of an agent-based support system for IaaS service discovery. *Proceedings of the 13th International Conference on Computational Science and Its Applications*.

[9] Chang, Y. S., Yang, C. T., & Luo, Y. C. (2011). An ontology based agent generation for information retrieval on cloud environment. *Journal of Universal Computer Science*, *17(8)*.

[10] Messina, F., Giuseppe, P., Corrado, S., Domenico, R., & Giuseppe, M. L. S. (2014). An agent based negotiation protocol for cloud service level agreements. *Proceedings of the 3rd International Conference on Enabling Technologies: Infrastructures for Collaborative Enterprises*.

[11] Domenico, T. (2012). Clouds meet agents: Toward intelligent cloud services. *Internet Computing*, *6(12)*.

[12] Myougnjin, K., Hanku, L., Hyogun, Y., Jee, I. K., & Hyung, S. K. (2011). IMAV: An intelligent multi- agent based on cloud computing for resource Virtualization.

[13] Ignacio, L. R., & Mario, H. T. (2011). Software agents as cloud computing services. *Advances in Intelligent and Soft Computing*, 271-276.

[14] Ahmed, G., & Amr, T. (2014). COBE framework: Cloud ontology blackboard environment for enhancing discovery behavior. *International Journal on Cloud Computing: Services and Architecture*, *4(5)*.

[15] Singh, A., & Manisha, M. (2012). Agent based framework for scalability in cloud computing. *International Journal of Computer Science & Engineering Technology*, *3(4)*.

[16] Han, T., and Kwang Mong Sim. "An agent-based cloud service discovery system that consults a cloud ontology. *Intelligent Control and Computer Engineering*(pp. 203-216) . Springer Netherlands.

Amr M. Tolba received his M.Sc. and Ph.D. degrees in computer science from Menoufia University, Egypt, in the area of databases and multi-agent systems, in 2002 and 2006 respectively. He joined King Saud University (KSU) in August 2006 as an assistant professor at the Department of Computer Science. His research interests include internet of things, intelligent systems, mobile social networks, recommender systems, e-learning, and cloud computing.

Ahmed Ghoneim received his M.Sc. degree in software modeling from University of Menoufia, Egypt, and the Ph.D. degree from the University of Magdeburg (Germany) in the area of software engineering, in 1999 and 2007 respectively. He is currently an assistant professor at the Department of Software Engineering, King Saud University. His research activities address software evolution; service oriented engineering, software development methodologies, scripting languages.