Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2014 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

2014

A CLOUD DEPLOYMENT APPROACH FOR CONSUMER SUPPORT SYSTEMS

Jyhjong Lin
Ming Chuan University, jlin@mail.mcu.edu.tw

Maria R. Lee
Shih Chien University, maria.lee@mail.usc.edu.tw

Tsairyuan Chang
Ming Chuan University, tychang@mail.mcu.edu.tw

Shinjer Yang
Soo Chow University, sjyang@csim.scu.edu.tw

A CLOUD DEPLOYMENT APPROACH FOR CONSUMER SUPPORT SYSTEMS

Jyhjong Lin, Department of Information Management, Ming Chuan University, Taiwan, jlin@mail.mcu.edu.tw

Maria R. Lee, Department of Information Technology and Management, Shih Chien University, Taiwan, maria.lee@mail.usc.edu.tw

Tsairyuan Chang, Department of Information Management, Ming Chuan University, Taiwan, tychang@mail.mcu.edu.tw

Shinjer Yang, Department of Computer Science and Information Management, Soo Chow University, Taiwan, sjyang@csim.scu.edu.tw

Abstract

Customer relationships have been commonly recognized as a critical factor for enterprises to succeed their business. Effective customer relationships could help enterprises deliver services to customers based on their needs, preferences, or past transactions. This model however emphasizes on the use of customer information for benefiting enterprises; customers in contrast receive less information from enterprises. To address this issue, the paradigm Consumer Support Systems (CSS) is proposed to support the provision of service information for customers to help on their decision making. In addition, since Cloud Computing has been popularity for its rich set of features and their practical applications on business can be expected, it is thus an envisioned trend for CSS to be deployed on clouds to enhance further its effectiveness on customer relationships. For this need, we present in this paper a cloud deployment approach for such a cloud-based CSS. The approach starts from the consideration of CSS and cloud characteristics, through the recognition of the architectural components in CSS and clouds, and finally ends with the deployment of the architectural components in CSS on clouds. The approach is illustrated by a cloud-based CSS for travel arrangement where decision support for travels is provided for satisfying the arrangement needs from travelers.

1 INTRODUCTION

Customer relationships have been commonly recognized as a critical factor for enterprises to succeed their business. Effective customer relationships could help enterprises deliver services to customers based on their needs, preferences, or past transactions. In this context, many discussions have been presented: (1) Customer Decision Support (CDS) (Ba 1997; Baker 2000); (2) Customer Relationship Management (CRM) (Galbreath and Rogers 1999; Lin and Lee 2004; O'Keefe and Mceachern 1998; Woodruff 1997); (3) Customer Knowledge Management (CKM) (Bueren, et al. 2004; Davenport and Klahr 1998; Garcia-Murillo and Annabi 2002; Lin 2007; Mobasher, et al. 2002; Thomke and Hippel 2002; Wilkestrom 1996); (4) Recommendation System (O'Mahoney, et al. 2004); and (5) Intelligent Agent (Wagner and Turban 2002). In general, these approaches focus on benefiting enterprises by collecting customer information; their usefulness on enhancing customer relationships has already been demonstrated (Bueren, et al. 2004). However, from the viewpoint of information flow, the reverse delivery of service information from enterprises to benefit customers is somehow insufficient. This presents a notable problem of information asymmetry that hurdles customer relationships since customers have no sufficient service information to assist on their decision making about what they really need. To address this problem, the paradigm Consumer Support Systems (CSS) has been introduced (Orman 2007) with a 4-layer framework of collaborative mechanisms to support the provision of service information from enterprises to customers. In addition, a method for its development has been presented in (Lin 2009) that considers its architecture and characteristics to provide guidance on its construction.

In terms of the architecture for Web information systems (WIS), client-server patterns were most commonly used in the past decades; almost all kinds of existing WIS were constructed using this style of architectures. However, a new paradigm, Cloud Computing and its three SaaS (Software-as-a-Service), PaaS (Platform-as-a-Service), IaaS (Infrastructure-as-a-Service) service models (Dubey and Wagle 2007; Hayes 2008; Hutchinson, et al. 2009), has been established and popularity in recent years for its rich set of features. Its advantage includes that users can utilize software services in a low cost-, threshold-, and risk-way; these services can be quickly deployed on the Internet without duplication of work such that developers can focus on their QoS (Quality of Service) to improve core competitiveness. Therefore, its practical applications on business with promising values can be expected. As such, a cloud-based version is recognized as a trend for the next generation of WIS, and hence how to deploy existing WIS on clouds becomes a desired field in the literature (Hutchinson, et al. 2009). As a WIS, therefore, CSS can be envisioned to evolve into its next generation on clouds to enhance further its effectiveness on customer relationships.

Thus, we extend in this paper the previous work for CSS by focusing on the deployment of CSS on cloud environments. The resultant cloud-based CSS is identified from the consideration of CSS and cloud characteristics, through the recognition of the architectural components in CSS and clouds, and finally ends with the deployment of the architectural components in CSS on those in clouds. To illustrate, the approach is applied to a cloud-based CSS for travel arrangement where decision support for travels is provided for satisfying the arrangement needs from travelers.

This paper is organized as follows. Section 2 presents the approach with respective description diagrams, including package, deployment, and sequence ones. The approach is then illustrated in Section 3 by applying it to a cloud-based CSS for travel arrangement. Finally, Section 4 has the conclusions and our future work.

2 THE DEPLOYMENT APPROACH

The approach has the following three steps where respective diagrams are used for descriptions:

1. **Requirement Identification**, described in an abstract package diagram, that clarifies CSS and cloud characteristics, and then identifies the desired cloud requirements for CSS (i.e., the desired requirements for the cloud-based CSS).
2. **Architecture Identification**, described in a detailed package diagram that determines the architectural components in CSS and prospective clouds.
3. **Deployment Specification**, presented in a deployment diagram that specifies the deployment of the architectural components in CSS on the intrinsic configuration elements in selected clouds.

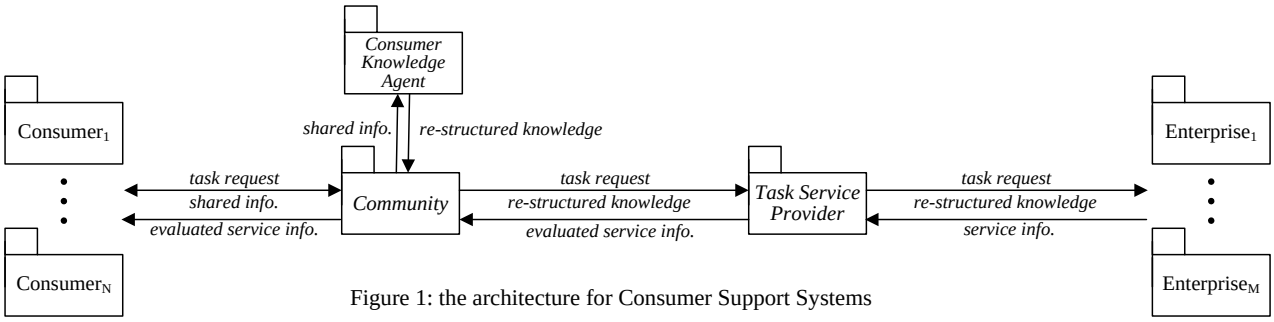

Figure 1: the architecture for Consumer Support Systems

2.1 Requirement Identification

2.1.1 Identification of CSS characteristics

Initially, consider the characteristics of CSS (Orman 2007) as follows.

1. Consumers would enroll into a community where they may share information about their desired tasks (e.g., buy or rent services from enterprises).
2. Such shared information is re-structured by a knowledge agent into specific styles of knowledge (Mobasher, et al. 2002) which will be captured by prospective enterprises (via a cooperative task service provider) to provide information about task-relevant services useful for these consumers.
3. The community cooperates with the task service provider to accomplish the task requests issued by consumers.
4. Based on the requests received, the task service provider helps to collect and evaluate information about task-relevant services from prospective enterprises, and then presents (via the community) to consumers for their possible recognition and comparisons.
5. With the evaluated information about task-relevant services, consumers can recognize and compare available services that may satisfy their desired tasks.

As a result, CSS has the architecture as shown in Figure 1.

1. It has a 4-layer architecture of five collaborative components where ***Consumers*** interact with ***Enterprises*** via three intermediaries: ***Community***, ***Consumer Knowledge Agent***, and ***Task Service Provider***.
2. ***Community*** is organized for ***Consumers*** to share information about their desired tasks. In addition, it is also responsible for forwarding the shared information to ***Consumer Knowledge Agent*** for re-structuring into specific styles of knowledge. It then sends the re-structured knowledge to ***Task Service Provider*** for forwarding to ***Enterprises*** to catch their needs (e.g., provide services satisfying their desired tasks). Finally, it also cooperates with ***Task Service Provider*** to receive service information relevant to these requests for ***Consumers*** to make recognition and comparisons.
3. ***Task_Service_Provider*** is used to receive the knowledge from ***Community*** and then forwards it to ***Enterprises*** for providing service information useful for ***Consumers***. In addition, based on the task requests received from ***Community***, it cooperates with ***Enterprises*** to provide information about these tasks. Furthermore, with the task-relevant information, it also helps to evaluate the information in a comparative model for presenting to ***Consumers*** (via ***Community***) to aid on their decision making.

2.1.2 Identification of cloud characteristics

Based on the architecture and service models of clouds (Hutchinson, et al. 2009), their characteristics can be identified as follows.

1. The architecture of a cloud may have a wide variety of configuration elements, including for example virtual machines, data storages, a/synchronous message queues, Web service interfaces, and user action portals/gadgets. Cloud applications may integrate the use of these elements to provide services.
2. Most of the elements in this architecture are dynamic and leverage a SOA (Service-Oriented Architecture). It is therefore possible for clouds to interoperable among each other (i.e., interoperability via Web services among clouds).

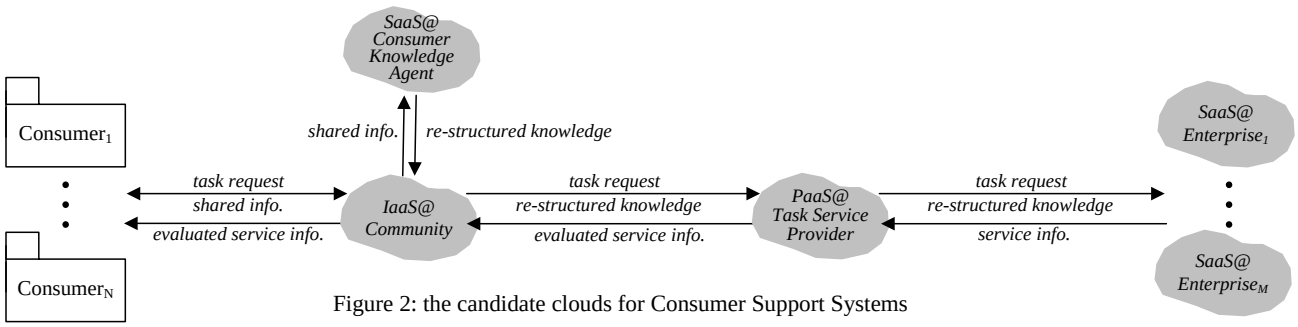

Figure 2: the candidate clouds for Consumer Support Systems

3. Service elements such as virtual machines, data storages, a/synchronous message queues, and Web service interfaces may be used together to enable the customizations of services by encapsulating the desired services from prospective clouds (either local or interoperable clouds).
4. Interface elements such as user action portals and gadgets may be used to provide rich user interface controls that enable the customizations of user interfaces by encapsulating the desired portals or gadgets in different user panels.

2.1.3 Identification of requirements for cloud-based CSS

With CSS and cloud characteristics, it is then good time to identify the desired requirements for the cloud-based CSS; that is, considering the five architectural components in CSS, various cloud environments may be selected for their respective deployments on these clouds to support the functional/non-functional purposes of CSS. Further, for the purposes of collecting consumer knowledge for enterprises and delivering service information to benefit consumers, it may require such QoS about the usage and user actions on these clouds as customized user interfaces and access modes, performance, reliability, security, and scalability.

For this, therefore, it is common to consider all available cloud environments that may satisfy the cloud requirements. As one may conceive, there is usually more than one cloud that satisfies the requirements; such clouds hence become the candidates from which specific ones are then selected for the realization of the deployment. As an example, Figure 2 shows some candidate clouds identified for CSS where

1. For ***Consumer_Knowledge_Agent*** and ***Enterprises***, some SaaS clouds (i.e., clouds denoted as ***SaaS@....*** that provide SaaS services) are considered since their desired behaviors may be satisfied by the SaaS services provided by these clouds.
2. For ***Task_Service_Provider***, some PaaS clouds are identified (i.e., clouds denoted as ***PaaS@....*** that provide PaaS services) since its desired behaviors may be well supported by the platform services provided by these clouds with the following features: (1) the inter-cloud interaction capability for the cooperation with ***Enterprises*** to forward knowledge/receive information; and (2) the intra-cloud analysis capability for the evaluation of the received service information into a comparative model for consumers to make recognition and comparisons.
3. For ***Community***, some IaaS clouds are identified (i.e., clouds denoted as ***IaaS@....*** that provide IaaS services) since its desired behaviors may be well supported by the infrastructure services provided by these clouds with the following features: (1) the storage and manipulation capability for the large volume of information shared among ***Consumers***; and (2) the inter-cloud interaction capability for the cooperation with ***Consumer_Knowledge_Agent*** for re-structuring the shared information into specific styles of consumer knowledge.

2.2 Architectural Identification

With the above mentioned requirements, the architectural components in CSS should be deployed on selected cloud environments that address these requirements by the interactions between $Consumer_{1...N}$ and $Enterprise_{1...M}$ through collaborative clouds; it is therefore necessary to identify the constituents in the CSS architectural components and also the configuration elements in candidate cloud environments such that the constituents in CSS can be smoothly deployed on the configuration elements in selected clouds. The following illustratively presents the constituents in CSS and also the elements in specific cloud environments.

2.2.1 The ***Community*** component in CSS

As mentioned above, ***Community*** is organized for ***Consumers*** to share information about their desired tasks (e.g., buy or rent services from ***Enterprises***). In addition, it is also responsible for forwarding the shared information to ***Consumer Knowledge Agent*** for re-structuring into specific styles of knowledge (i.e., knowledge of consumers). It then sends the re-structured knowledge to ***Task Service Provider*** for forwarding to ***Enterprises*** to catch their needs (e.g., provide services satisfying their desired tasks). Finally, it also cooperates with ***Task Service Provider*** to receive service information relevant to these requests for ***Consumers*** to make recognition and comparisons.

In summary, these requirements for ***Community*** can be described as: (1) **Share consumer information** that helps on the information sharing among $\text{Consumer}_{1...N}$; (2) **Process shared information** that forwards the shared information to ***Consumer_Knowledge_Agent*** for re-structuring into the consumer knowledge, and then send the re-structured knowledge to ***Task_Service_Provider***; (3) **Process task request** that receives the task requests from and return the evaluated information about task-relevant services to $\text{Consumer}_{1...N}$; (4) **Cooperate with task service provider** that cooperates with ***Task_Service_Provider*** to receive the evaluated information about task-relevant services; and (5) **Present services information** that provides $\text{Consumer}_{1...N}$ with the rich user interface controls for visualizing the information from ***Task_Service_Provider***.

Based on the above requirements for ***Community***, Figure 3 shows its five constituents that realize these requirements. In particular, 'Interface Manager' is imposed to realize the customization of user interfaces for $\text{Consumer}_{1...N}$ where *Consumer Profiles* are used to determine which interface portals are preferred by them; in addition, with such customized user interfaces, their containing gadgets may withhold by 'Portal Manager' the visualized information for sharing or about the task-relevant services to form the customized portals (under available *Portal Frameworks*) that deliver to $\text{Consumer}_{1...N}$ their desired information according to their interactive requirements. Further, 'Info. Manager' accesses *Community Member Profiles* and *Shared Consumer Info.* to help on information sharing among the interested consumers; *Shared Consumer Info.* is also retrieved for re-structuring into specific styles of knowledge by ***Consumer_Knowledge_Agent***. In addition, 'Task Request Manager' forwards the task requests from $\text{Consumer}_{1...N}$ to 'Cooperation Manager' that cooperates with ***Task_Service_Provider*** to receive the evaluated information about these requests; the evaluated information is then visualized and returned to $\text{Consumer}_{1...N}$ through 'Portal Manager'. Finally, 'Web Service Manager' is responsible for interoperating with two external architectural components through *Web Service Client* APIs for accessing any remote services provided from these two components.

2.2.2 The ***Task_Service_Provider*** component in CSS

For another example, ***Task_Service_Provider*** receives the consumer knowledge from ***Community*** and then forwards that knowledge to ***Enterprises*** that utilizes it to provide service information useful for ***Consumers***. In addition, based on the task requests received from ***Community***, it cooperates with ***Enterprises*** to provide information about these tasks. Furthermore, with the task-relevant information, it also helps to evaluate the information in a comparative model for presenting to ***Consumers*** (via ***Community***) to aid on their decision making. In summary, these requirements for ***Task_Service_Provider*** can be described as: (1) **Process consumer knowledge** that receives the consumer knowledge from ***Community*** and forwards the knowledge to $\text{Enterprise}_{1...M}$; (2) **Process task request** that receives the task requests from and return the evaluated information about task-relevant services to ***Community***; (3) **Cooperate with enterprise** that cooperates with $\text{Enterprise}_{1...M}$ to provide the information about task-relevant services; and (4) **Evaluate service information** that evaluates into a comparative model the information about task-relevant services from $\text{Enterprise}_{1...M}$.

Based on the above requirements for ***Task_Service_Provider***, Figure 4 shows its five constituents that realize these requirements. Particularly, 'Task Request Manager' forwards the task requests to 'Cooperation Manager' that cooperates with $\text{Enterprise}_{1...M}$ to provide information about task-relevant services; the service information from $\text{Enterprise}_{1...M}$ is then evaluated by 'Evaluation Manager' for returning to ***Community***. Finally, 'Web Service Manager' is responsible for interoperating with various external architectural components through *Web Service Client* APIs for any remote services provided in these components.

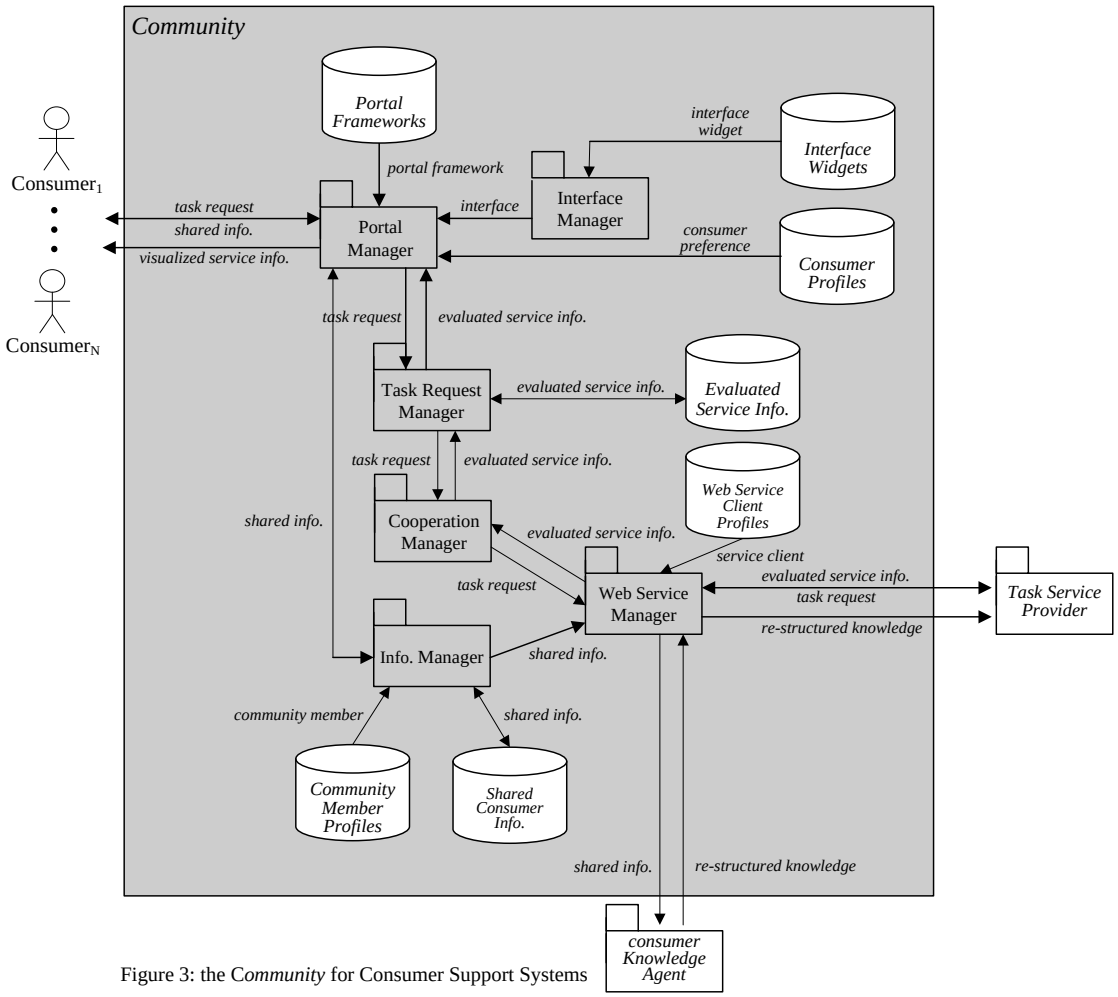

Figure 3: the C*ommunity* for Consumer Support Systems

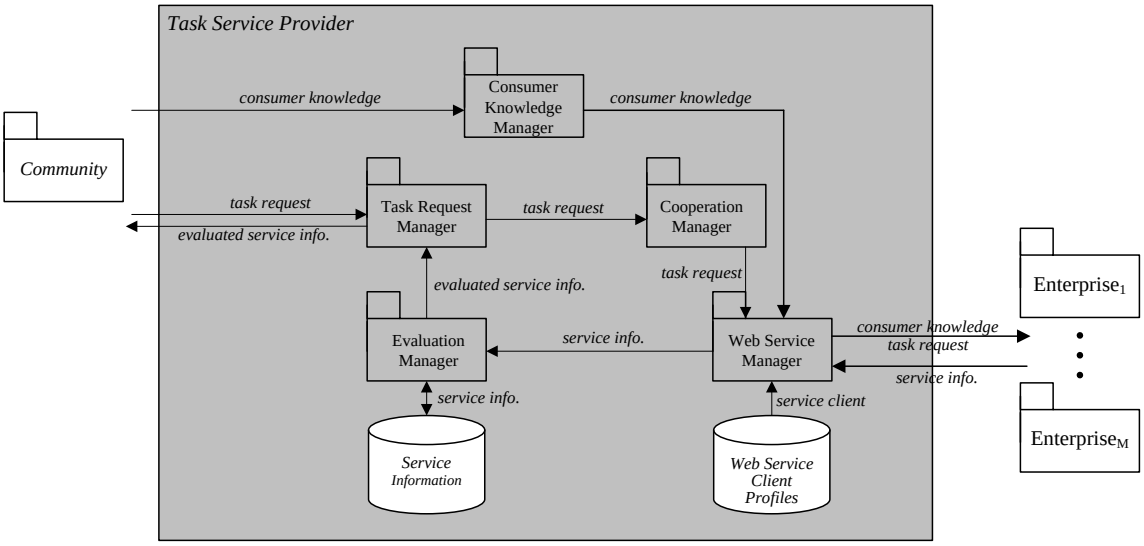

Figure 4: the *Task Service Provider* component for Consumer Support Systems

2.2.3 The elements in candidate clouds

With the constituents in the CSS architectural components, the configuration elements in candidate clouds also need to be identified such that the constituents in CSS can be smoothly deployed on the elements in selected clouds. For this, three situations are necessarily discussed:

1. For those SaaS clouds on which ***Customer_Knowledge_Agent*** and ***Enterprises*** are considerably deployed, their SaaS services would satisfy the desired behaviors of these two CSS components. The deployment of these two CSS components on the SaaS clouds is thus not necessary and hence the identification of the configuration elements in the SaaS clouds is also extra.

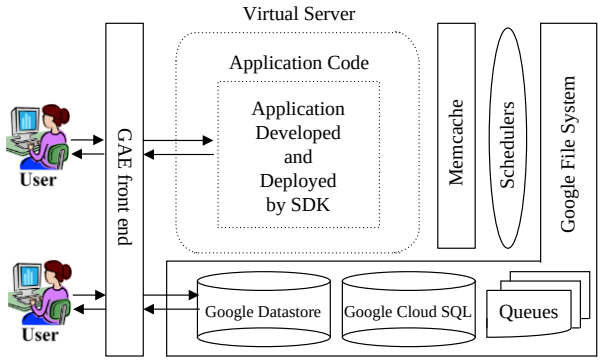

Figure 5 : the Google GAE PaaS cloud

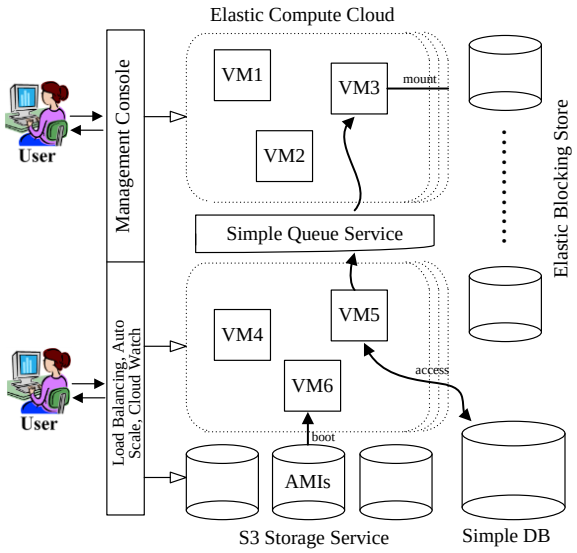

Figure 6 : the Amazon EC2 IaaS cloud

2. For those PaaS clouds on which ***Task_Service_Provider*** is considerably deployed, their PaaS services would support well the desired behaviors of the CSS component. The deployment of this CSS component on the PaaS clouds is thus necessary and hence the configuration elements in the PaaS clouds need to be identified. Currently, there are plenty of available PaaS clouds such Google GAE (GAE 2013) and Microsoft Azure (Azure 2013); Figure 5 shows the elements in GAE that supports well the inter-cloud interaction and intra-cloud analysis capabilities.
3. For those IaaS clouds on which ***Community*** is considerably deployed, their IaaS services would support well the desired behaviors of the CSS component. The deployment of this CSS component on the IaaS clouds is thus necessary and hence the configuration elements in the IaaS clouds need to be identified. Currently, there are also plenty of available IaaS clouds such Google GCE (GCE 2013) and Amazon EC2 (EC2 2013); Figure 6 shows the elements in EC2 that supports well the storage manipulation and inter-cloud interaction capabilities.

2.3 Deployment Specification

With the constituents in CSS components and the elements in candidate clouds, it is good time to determine from the candidate clouds which ones are selected for the deployment of these components. In general, this can be achieved by means of specific evaluation criteria that may rank these candidates in the context of the satisfaction of the cloud requirements. For example, based on the QoS features of the service models provided by these clouds, a candidate whose service models gain best weighted assessments may be selected as the cloud environment for a CSS component to be deployed.

1. For those SaaS clouds on which ***Customer_Knowledge_Agent*** and ***Enterprises*** are considerably deployed, any ones hosted by the prospective agents/enterprises with SaaS services most satisfying the desired behaviors of these two CSS components would be selected for replacing the provision of the services provided by these two CSS components.

2. For those PaaS clouds on which ***Task_Service_Provider*** is considerably deployed, many available ones such as Google GAE and Microsoft Azure can be considered. Among them, however, GAE as shown in Figure 5 would be selected since its platform supports well the inter-cloud interaction capability for the cooperation with ***Enterprises*** to forward knowledge/receive information and the intra-cloud analysis capability for the evaluation of the received service information into a comparative model for consumers to make recognition and comparisons. Figure 7 shows the deployment of the five constituents in ***Task_Service_Provider*** on the three virtual servers (i.e., Mashup, Analysis, and Web Service) in GAE where some data storages such as Datastore and Cloud SQL are specifically used.

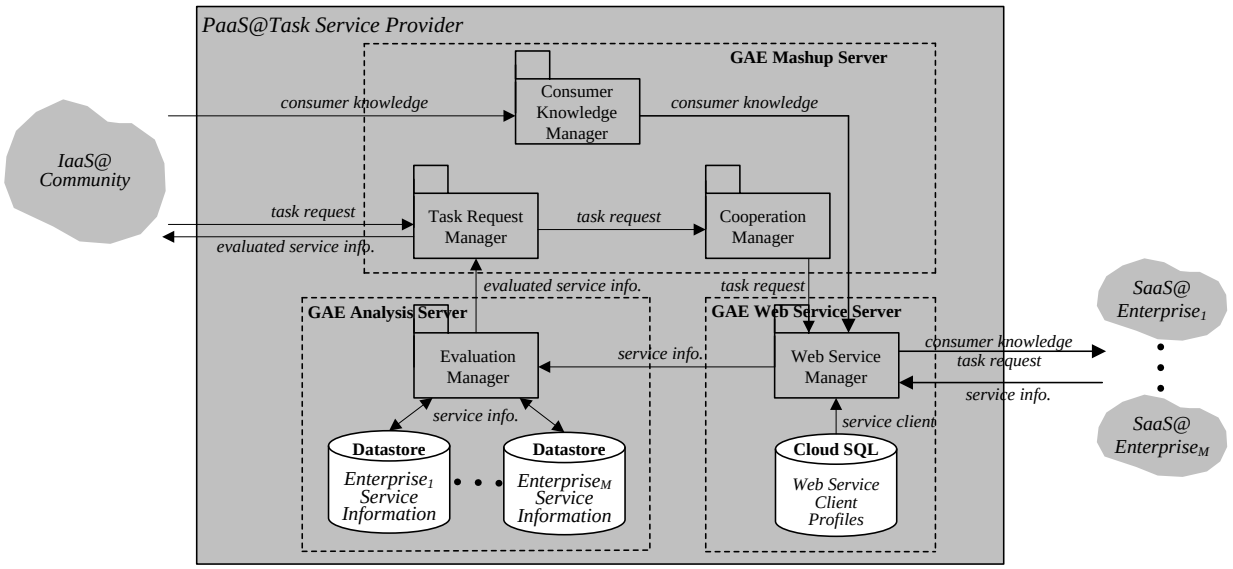

Figure 7: the Google GAE-based deployment of the *PaaS@Task Service Provider*

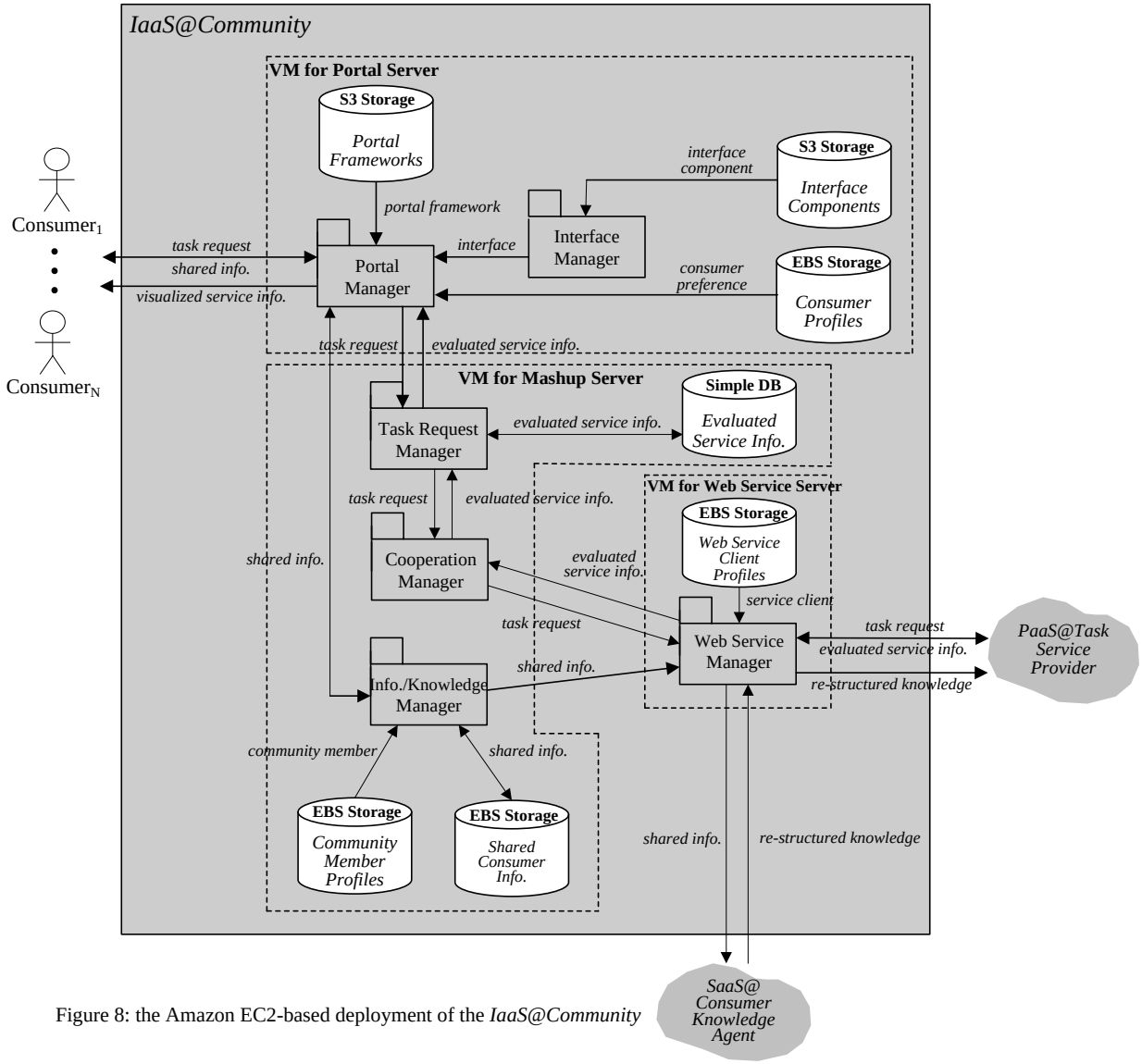

Figure 8: the Amazon EC2-based deployment of the *IaaS@Community*

3. For those IaaS clouds on which ***Community*** is considerably deployed, many available ones such as Google GCE and Amazon EC2 can also be considered. Among them, however, EC2 as shown in Figure 6 would be selected since its infrastructure supports well the storage and manipulation capability for the large volume of information shared among ***Consumers*** and the inter-cloud interaction capability for the

cooperation with ***Consumer_Knowledge_Agent*** for re-structuring the shared information into specific styles of consumer knowledge. Figure 8 shows the deployment of the six constituents in ***Community*** on the three virtual machines (VMs) in EC2 and some storages such as S3 storage, EBS storage, and Simple DB are specifically used.

3 APPLYING TO TRAVEL ARRANGEMENT

Providing travelers with services about travel arrangement is very important for travel agents. Many existing agents offer relevant services such as searching, viewing, and ordering tours via their own web sites or intermediary agents (e.g., EzTravel). However, these ordinary functions are not sufficient for many travelers who desire more sophisticated services (e.g., knowledge-based decision support) for their advanced purposes such as arranging new tours after sharing related experiences with other travelers. For these limits, a CSS for travel arrangement is devised to provide these sophisticated services; further, for enhancing its effectiveness on satisfying the arrangement needs from travelers, the aforementioned deployment approach is applied to deploy it on cloud environments.

3.1 The Cloud Requirements of CSS for Travel Arrangement

In our example, prospective travelers may desire sophisticated knowledge-based decision support services about travel arrangement after sharing information with other travelers. Therefore, four functions below are often desired to satisfy their needs.

1. ***Share travel experiences*** - a traveler shares travel experiences with other travelers.
2. ***Share tour site thoughts*** - a traveler shares tour site thoughts with other travelers.
3. ***Arrange a new travel*** - a traveler arranges a new travel by organizing its transportation, accommodation, tour sites, and so on.
4. ***Recommend a new tour site*** - a traveler recommends a new tour site by specifying its specialties.

With the above desired functions, the characteristics about CSS for travel arrangement can be identified as below.

1. **Travelers** would enroll into **Traveler Community** where they may share such information about travel arrangement as travel experiences, tour site thoughts, travel arrangement, and tour site recommendation.
2. The shared information is re-structured by **Traveler Knowledge Agent** into specific styles of knowledge that is then captured by **Travel Agents** (via **Travel Service Provider**) to provide **Travelers** with useful travel arrangement information.
3. **Traveler Community** cooperates with **Travel Service Provider** to accomplish the travel arrangement requests issued by **Travelers**.
4. Based on the requests received, **Travel Service Provider** helps to collect and evaluate the information about travel arrangement from prospective **Travel Agents**, and then presents (via **Traveler Community**) to **Travelers** for their possible recognition and comparisons.
5. With the evaluated information about travel arrangement, **Travelers** can recognize and compare available services that may satisfy their desired tasks.

As a result, CSS for travel arrangement has the architecture as shown in Figure 9.

1. It has five architectural components where **Travelers** are interacting with **Travel Agents** through three intermediaries, **Traveler Community, Traveler Knowledge Agent,** and **Travel Service Provider**.
2. It helps **Travelers** to share the information about travel arrangement via **Traveler Community** and then re-structure it by **Traveler Knowledge Agent** into summarized knowledge for **Travel Agents** to catch their needs.
3. It helps **Travelers** to make possible recognition and comparisons by delivering the evaluated information about travel arrangement from **Travel Agents**.

Therefore, based on the above architecture, the desired requirements for the cloud-based CSS for travel arrangement can be identified. For example, as shown in Figure 10, four clouds ***IaaS@Traveler_Community***, ***SaaS@Traveler_Knowledge_Agent***, ***PaaS@Travel_Service_Provider***, and ***SaaS@Travel_Agent*$_{1...M}$**, may be considerably imposed to provide the two kinds of clients, Traveler$_{1...N}$ and Travel Agent$_{1...M}$, with respective travel arrangement information and traveler knowledge.

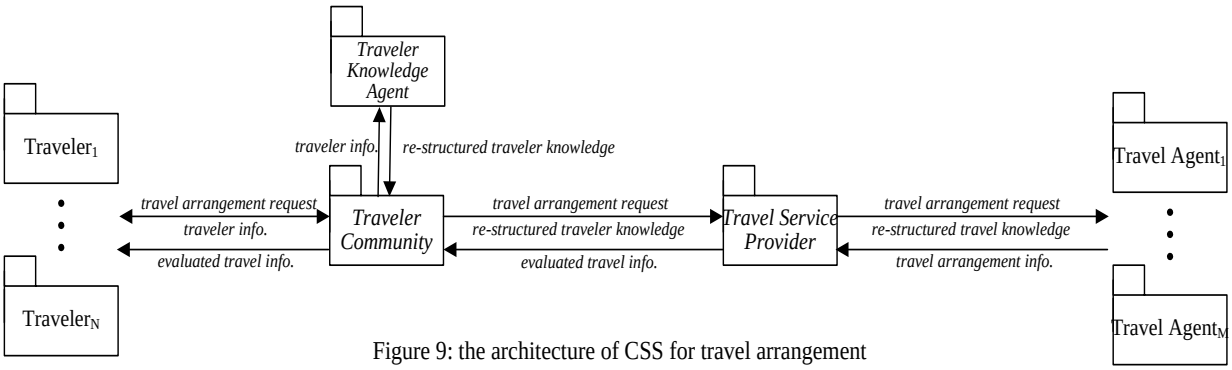

Figure 9: the architecture of CSS for travel arrangement

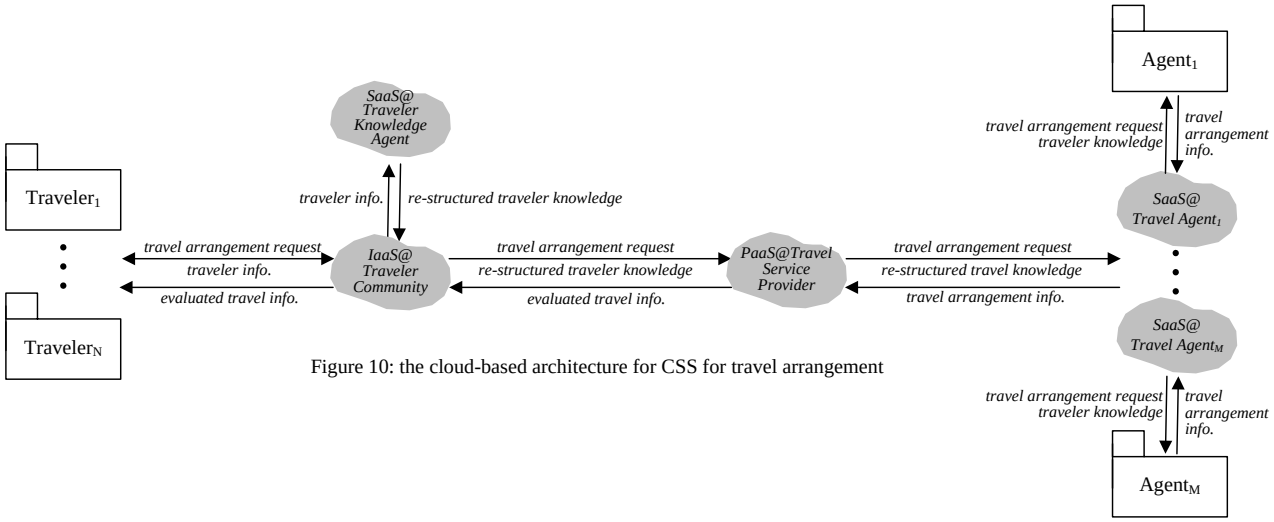

Figure 10: the cloud-based architecture for CSS for travel arrangement

3.2 The Deployment of CSS for Travel Arrangement

In the cloud-based CSS for travel arrangement, $\text{Traveler}_{1...N}$ may issue the travel arrangement requests via ***IaaS@Traveler_Community*** to acquire from prospective ***SaaS@Travel_Agent*$_{1...M}$** the information about travel arrangement that is earlier collected and evaluated by ***PaaS@Travel_Service_Provider***. In particular, bi-directional information flows between $\text{Traveler}_{1...N}$ and $\text{Agent}_{1...M}$ where (1) the information shared among $\text{Traveler}_{1...N}$ is re-structured into summarized knowledge and then captured by ***SaaS@Travel_Agent*$_{1...M}$** (via ***PaaS@Travel_Service_Provider***); and (2) reversely, information about the travel arrangement from ***SaaS@Travel_Agent*$_{1...M}$** is structured and evaluated by ***PaaS@Travel_Service_Provider*** for possible recognition and comparisons (via ***IaaS@Traveler_Community***) by $\text{Traveler}_{1...N}$. Therefore, based on the generic CSS architecture in Figure 2, Figure 10 shows the cloud-based architecture of CSS for travel arrangement where the two clouds ***PaaS@Travel_Service_Provider*** and ***IaaS@Traveler_Community*** would contain the same deployed CSS constituents as those in Figures 7 and 8).

4 CONCLUSIONS

In this paper, we present a method for the deployment of CSS on cloud environments. The method addresses first the identification of CSS and cloud characteristics. After then, the architectural components in CSS and candidate clouds are recognized in a step-by-step manner to support a smooth deployment. Finally, the deployment is realized by imposing the constituents in CSS components on the configuration elements in selected clouds. As a result, for such a deployed cloud-based CSS, four clouds are imposed with inter-cloud interactions to support both of the information from consumers to enterprises (i.e., for enterprises, capturing knowledge from consumers) and the reverse delivery of information from enterprises to consumers (i.e., for consumers, receiving service information from enterprises). For illustration, the method is applied to a cloud-based CSS for travel arrangement. In the CSS, a IaaS-based traveler community is imposed for information sharing among travelers. Further, a SaaS-based knowledge agent is used to re-structure the shared information into various styles of traveler knowledge that is then captured by prospective SaaS-based travel agents. Since travelers may issue the travel arrangement requests to these SaaS-based travel agents, a PaaS-based travel

service provider is imposed to help on collecting and evaluating the information about travel arrangement from SaaS-based travel agents, and then presenting (via the IaaS-based traveler community) to travelers for their possible recognition and comparisons.

Since cloud applications have been recognized in recent years as a trend for the next generation of business applications, how to deploy the many existing business applications on the clouds for taking advantage of cloud computing has thus become a desired field in the literature. However, current discussions about this mainly focus on some important issues about the deployment on a cloud environment and then present respective tips for addressing such issues. For a formal process, any methods that take into considerations of the architecture and characteristics of both business applications and clouds to provide guidance on their deployment are still missing. Such methods, in our opinion, should not be negligible since a well-guided process is critical for directing the deployment of the many existing applications in a systematic and managed manner. The method presented herein provides an effort on this need.

As our future work, we will continue to explore the realization of the cloud-based CSS for travel arrangement. Thereafter, in addition to travel arrangement, we will look also forward to the practical use of our work in other domains like e-Book management and e-Learning arrangement; its usability on such cloud-based decision support systems will also be carefully experienced. As one may conceive, while deploying these systems, experiences about the application can be collected correspondingly for validating the usefulness and effectiveness of the method.

Acknowledgements

The work in this paper is supported by Ministry of Science and Technology in Taiwan under Grant NSC 102-2410-H-130 -032.

References

Ba, S., et al. (1997). Using client-broker-server architecture for Intranet decision support, Decision Support Systems, 19(3), 171-192.

Baker, M. (2000). Creating An Alliance between Employees and Customers, Knowledge Management Review, 3(5), 10-11.

Galbreath, J. and Rogers, T. (1999). Customer Relationship Leadership: A Leadership and Motivation Model for the 21th Century Business, The TQM Magazine, 11(3), 161-171.

Lin, J. and Lee, M. (2004). An OO Analysis Method for CRM Information System, Journal of Information and Software Technology, 46(7), 433-443.

O'Keefe, R. and Mceachern, T. (1998). Web-based Customer Decision Support Systems, Communications of the ACM, 41(3), 71-78.

Woodruff, R. (1997). Customer Value: The Next Source for Competitive Advantage, Journal of the Academy of Marketing Science, 25(2), 139-153.

Bueren, A., et al. (2004). CKM – Improving Performance of Customer Relationship Management with Knowledge Management, 37th Annual Hawaii International Conference on System Sciences, p. 70172.2.

Davenport, T. and Klahr, P. (1998). Managing Customer Knowledge, California Management Review, 40(3), 195-208.

Garcia-Murillo, M. and Annabi, H. (2002). Customer Knowledge Management, Journal of the Operational Research Society, 53(8), 875-884.

Lin, J. (2007). An OO Development Method for Customer Knowledge Management Information Systems, Journal of Knowledge-Based Systems, 20(1), 17-36.

Mobasher, B., et al. (2002). Five Styles of Customer Knowledge Management, and How Smart Companies Use Them To Create Value, European Management Journal, 20(5), 459-469.

Thomke, S. and Hippel, E. (2002). Customers as Innovators, Harvard Business Review, April 2002, 5-11.

Wilkestrom, S. (1996). The Customer as Co-Producer, The European Journal of Marketing, 30(4), 6-19.

O'Mahoney, M., et al. (2004). Collaborative Recommendation: A Robustness Analysis, ACM Transactions on Internet Technology, 44(11), 344-377.

Wagner, C. and Turban, E. (2002). Are Intelligent e-Commerce Agents Partners or Predators?, Communications of the ACM, 45(5), 84-90.

Orman, L. (2007). Consumer Support Systems, Communications of the ACM, 50(4), 49-54.

Lin, J. (2009). An Object-Oriented Development Method for Consumer Support Systems, International Journal of Software Engineering and Knowledge Engineering, 19(7), 933-960.
Dubey, A. and Wagle, D. (2007). Delivering Software as a Service, The McKinsey Quarterly, http://ai.kaist.ac.kr/~jkim/cs489-2007/Resources/DeliveringSWasaService.pdf.
Hayes, B. (2008). Cloud Computing, Communications of the ACM, 51(7), 9-11.
Hutchinson, C., et al. (2009). Navigating the Next Generation Application Architecture, IEEE IT Pro, 11(2), 18-22.
Google App Engine (GAE). (2013). https://appengine.google.com/
Windows Azure (Azure). (2013). http://www.windowsazure.com/zh-tw/
Google Compute Engine (GCE). (2013). https://cloud.google.com/products/compute-engine
Amazon EC2 (EC2). (2013). http://aws.amazon.com/ec2/