20th ISPE International Conference on Concurrent Engineering
C. Bil et al. (Eds.)

doi:10.3233/978-1-61499-302-5-40

Cloud Automatic Software Development

Hind BENFENATKI[a,1], Hamza SAOULI[b], Nabila BENHARKAT[c],
Parisa GHODOUS[a], Okba KAZAR[b], Youssef AMGHAR[c]
[a] *Université de Lyon, CNRS*
Université Claude Bernard – Lyon 1- France, LIRIS UMR 5205
[b] *Computer Science Department – University of Biskra*
[c] *INSA-Lyon, LIRIS, UMR5205*
F-69621, France

Abstract. Software Engineering must face the new challenges imposed by the Cloud Computing paradigm. New methodologies for software development must be proposed. For this purpose, this paper presents a specific methodology for collaborative software development in the Cloud, and then describes the architecture of Automatic Software Development as a Service (ASDaaS). The goal of ASDaaS is to popularize software development in the Cloud and make it accessible to non-IT professionals. In fact, with Cloud Computing and the convergence toward "Everything as a Service", we no longer consider the classical context of software development, where IT teams or integrators are solicited to perform software development. ASDaaS allows a stakeholder, without computer skills to perform automatic developments from functional requirements, SLA (Service Level Agreement) requirements, and business rules definition. ASDaaS promotes the discovery and composition of web services. It is itself composed of a set of services which can carry out and cover the whole process of software development. ASDaaS also allows the automatic development on Cloud platforms of undiscovered services by model transformation. Indeed, for each new development, a choice of PaaS (Platform as a Service) is performed by matching development constraints imposed by the stakeholder, with the features and services offered by the Cloud Platform.

Keywords. Collaborative Development, Cloud Computing, Business rules, Business Process

Introduction

The main ideas of the paradigm "Cloud Computing" is to enable companies to acquire Computing resources on demand, to make payment according to use, and to discharge the concern of the resources provenance [1].

Cloud Computing offers many advantages for software development [2], particularly because it offers the possibility of an elastic resource allocation. It promotes a simple and ergonomic use, without having to worry about the underlying infrastructure or the deployed development environment. Another advantage of this new paradigm for software engineering is that companies and organizations can develop mas-

[1] hind.benfenatki@universite-lyon.fr

sively distributed software systems by dynamically assembling services. These services may come from different providers [3].

With the Cloud Computing paradigm, traditional software development methodologies gradually give way to the composition of services that represent business software artifacts. However, the immaturity of the Cloud model in the field of software engineering induces the lack of specific and adapted methodologies to develop appropriate software. Indeed, the methodologies experienced by software engineering do not meet the distributed and collaborative nature, simplistic and user-friendly approach and flexibility in the allocation of resources offered by Cloud Computing.

In this paper, we propose a methodology for collaborative development of Cloud-based service-oriented software, and an architecture of Automatic Software Development as a Service (ASDaaS). ASDaaS allows the development of business process with minimal intervention of the business stakeholder and without the intervention of developers. It promotes the discovery and the composition of services described in the business process; and the automatic development of undiscovered services in different Cloud platforms according to the APIs (Application Programming Interface) and services proposed by the Cloud platforms, and the SLA (Service Level Agreement) and KPIs (Key Performance Indicators) described by the user. ASDaaS also allows the selection of a Cloud infrastructure for the software deployment.

The rest of this paper is structured as follows. Section 1, describes the software engineering issues vis-à-vis the Cloud. Section 2 addresses related works. Section 3 describes the proposed methodology. Section 4 introduces ASDaaS's architecture. Section 5 draws final conclusions and describes our future works.

1. Cloud Computing : Software Engineering challenges

Cloud Computing is a model for access through the network, and on demand to a set of shared and configurable Computing resources (eg networks, servers, storage, applications, and services) that are rapidly deployable and releasable with minimal administration effort or provider interaction. This paradigm considers three main service models: SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service) [4] on which other services such as "Data as a Service" and "Desktop as a Service" are grafted. Four deployment models are possible with Cloud Computing: public, private, community and hybrid.

From a software engineering point of view, Cloud Computing faces several challenges related in particular to:

- **Distributivity :** the modularity imposed by the Cloud Computing is a challenge in itself, which leads to dividing the application into units that can be deployed in a distributed environment [5]. Therefore, the debug and test operations can therefore be difficult. The distribution of data in the Cloud must also meet the challenge of access to data [6].
- **Security:** storing data on the Cloud infrastructure creates security challenges with respect to data access and confidentiality due partially to the multi-tenant architecture [7].
- **Cloud service composition:** developers are brought to discover and compose web services to avoid reinventing existing services [2].

- **Interoperability:** the migration of developed applications from a Cloud platform to another poses interoperability and portability problems, due to the advocated architecture and the APIs compatibility between the different Cloud providers [6].
- **Dependence on PaaS:** the technological advancement heavily relies on the PaaS support technologies [6].
- **Elasticity and scaling:** another challenge is that the hardware architectures are not fixed but rather flexible due to the property of elasticity in the Cloud Computing technologies.
- **Evaluation:** evaluation of applications developed on the Cloud is difficult due to the inability to assess the complexity of the APIs provided by the PaaS, since the implementation of these is ignored by users [6].

2. Related works

Due to the recent nature of the Cloud Computing paradigm, few methodologies and approaches exist in literature. In [8], the authors proposed an approach that used the Domain Specific Languages (DSL) within the process of development and deployment of software on the Cloud. This approach relies on two basic steps. The first concerns the development of a DSL for a given domain in order to facilitate the modeling of an application. In the second step, the DSL language is used by Application Designer to model applications. These models are automatically translated into specific executable code to a target Cloud platform, and then this code is automatically deployed on the Cloud platform. This approach can be costly in DSL development time during the first phase.

In [9], Ardagna et al. proposed the MODACLOUDS system issued from an European project [10] that uses the principle of MDD (Model Driven Development) for the development of applications on the Cloud. MODACLOUDS comprises a design environment and a runtime environment. The design environment includes the modeling and code generation part, and includes a DSS (Decision Support System) module that allows risk analysis for the selection of Cloud providers and evaluation of the impact of the adoption of Cloud in the internal business processes. The runtime environment allows to observe the system running and to provide feedback to the design environment. This allows developers to react to performance fluctuations and redeploy applications on different Clouds over the long term. This approach is very interesting; however, it does not exploit the APIs and services provided by Cloud platforms, and returns the final choice of Cloud platform to the developer (human intervention).

The work proposed by [11] describes the SaaS Development Life Cycle (SaaSDLC) and outlines six phases. During the “envisioning” phase, an identification of new business opportunities and applications that can benefit from the characteristics of the Cloud is made. A “platform evaluation” is then performed on the basis of cost and capabilities. The “planning” phase includes overall functionality requirements and design specifications, and establishes project plans. Once the Cloud platform is selected, a subscription is made. The “Subscription” phase involves the Cloud provider and the client in the feasibility assessment of the application security architecture, and data architecture on the Cloud platform. Then comes the “service devel-

opment" phase which is composed of a series of iterations. Deployment and testing are performed continuously throughout each iteration. The last phase "operations" includes the creation of process deployment and operation for the functioning of the service hosted on the Cloud. The SaaSDLC does not consider reuse of Cloud services. It promotes the development to a specific platform, making application portability more difficult.

As part of IBM Research to provide a Software Development as a Service SDaaS, the authors in [12] propose a hybrid development method (agile & workflow) for the large projects. Two separate tracks characterize this method: Prototype and Release Tracks. Prototype Track comprises a pure agile development, with short segments of development of specific features, customer demonstrations and iterations. The Release Track includes integration and testing of all the features that have been prototyped and completed in the previous cycle prototype. This work does not consider the Cloud Computing aspect in the development process, but is mainly focused on the adaptation of agile and workflow methods for projects development.

In [13] Guha et al. advocate the intervention of Cloud provider in the Agile eXtreme Programming software development process, especially in planning, design, building, testing and deployment phases to mitigate the challenges associated with Cloud software development, and makes it more advantageous. The roles and activities of the Cloud provider and developers are pre-defined. In this paper, the authors consider the aspect of roles between the various stakeholders in the agile development process of Cloud applications, and do not consider the other characteristics of Cloud applications.

In [14], the authors describe Service-Oriented Software Development Cloud (SOSDC), a Cloud platform for developing service-oriented software and a dynamic hosting environment. The SOSDC adopts a system architecture covering the three levels of Cloud services. The IaaS level contains "Dynamic Provisioning Software Appliance" and is primarily responsible for providing software appliances. The PaaS level "App Engine: Dynamic Hosting Environment for Service Oriented Software" provides App Engine for testing, implementing and monitoring the deployed application without having to consider the technical details. SaaS level aims to provide "Online Service-Oriented Software Development Environment". Once an application is developed, the developer may request an App Engine hosting environment by specifying the deployment requirements. This approach aims to provide a dynamic development environment by providing on demand appliance for developers, but does not exploit public Cloud platforms.

The methodology proposed in this work, (i) maintains interoperability through the use of web services and the modeling of functionalities to be developed; (ii) meets the requirements of the distributed nature of Cloud; and (iii) does not depend on a particular platform.

3. The proposed methodology

We no longer consider the traditional way of development, where the customer goes to a software integrator, or mobilizes his/her IT department to develop software that meets their needs. In this work, we do not consider developers as key stakeholders,

but business stakeholder who feel the need to automate a Business Process Management (BPM). Unlike other applications, BPM considers the organization of the company. Therefore, the definition of business rules is paramount.

Our methodology benefits from Cloud Computing and Web services to automatically build and deploy a service-oriented software. It promotes the discovery and composition of services, but also entails the development of new services on Cloud platforms if undiscovered.

The result of this methodology is an Automatic Software Development as a Service (ASDaaS). ASDaaS is a development environment in which the different services are developed and deployed on various Cloud platforms, amounting to interacting decentralized and interdependent systems. The ASDaaS uses three types of data input (requirements in terms of functionality, the SLA and the business rules that describe the business constraints) to generate a software release.

ASDaaS is an upper layer of the SaaS. This positioning allows users of ASDaaS not to depend on a particular platform. In addition, it allows us to proceed to the choice of an appropriate platform for each development. Each platform has different APIs, and the choice of platform should depend on the need for development.

The idea is that via a browser a business stakeholder can access an environment of automatic development of business processes (ASDaaS). ASDaaS incorporates the composition of web services, the automatic development of services on Cloud Platforms (PaaS: Google App Engine, Windows Azure…), and the deployment of services on Cloud infrastructures (IaaS: Amazon Web Services…).

The originality of our approach lies in the following facts: (i) It does not require great knowledge in software engineering to gain access to development; (ii) It promotes the composition of developed services on multiple Cloud platforms : Inter-Clouds; (iii) It allows selection of a development platform (PaaS) according to the development; (iv) It allows selection of IaaS for deployment that meets the SLA and pre-defined KPIs; (v) and it automatically deploys developed applications on a preselected Cloud infrastructure.

Figure 1 illustrates the different phases of the proposed methodology. We note that the business stakeholder can, at any time, introduce changes following his feature needs.

3.1. ASDaaS Subscription .

To initiate the project, a project identity sheet is created by the project creator. This sheet contains an identification of the different organizations involved in the project, and several stakeholders of each organization. A profile is assigned to each stakeholder. The creator of the project has an administrator profile. This allows him to allocate tasks to different stakeholders and have a rise of information feedback through operating reports, showing the progress of the work.

3.2. Requirements expression.

This phase consists of describing the inputs of ASDaaS described by the business stakeholder and translating the service needs following the WSDL file for searching purposes. Three types of inputs are identified:

- The requirements and expectations of the customer expressed in terms of functionality (services) in a business process. It consists of defining the inputs / outputs of each service, without having to write the algorithms. This phase involves to i) modeling the Workflow Description , ii) defining the number of processes, iii) describing the features included in the process, in addition to the input / output, and iv) defining the context and launch architecture.
- SLA: the stakeholder describes his/her requirements in terms of quality of service, Cloud platforms and infrastructures security.
- Business rules that define business constraints.

3.3. Application interface creation.

The client creates the interface through an easy-to-use tool. The business stakeholder has at his disposal a variety of forms (windows, tabs, buttons, check boxes ... etc). The creation of the interface allows the stakeholder to see his requirements more clearly.

3.4. Service discovery.

In this phase, through a web services search engine, we have to discover services corresponding to several features included in the different business processes. The advantage of this phase is to reducing the workload and improving reusability of web services. The complex features that have not been discovered as a web service, are broken down, if possible, to allow start a finer new search. If no further decomposition is possible, and no web service has been found for this feature, we move on to the phase of service automatic development.

3.5. Service composition.

This phase composes the discovered and developed services. It occurs according to the discovery and implementation of the features.

3.6. Automatic development of undiscovered services.

This phase identifies and develops the features described in the requirements but that have not been discovered as web services. This involves three key steps:

1. Undiscovered services modeling: undiscovered services are modeled according to the UML (Unified Modeling Language) in order to allow automatic generation of a code for a targeted Cloud platform with full knowledge of the tools it offers. The model-driven approach allows one to “model once and run everywhere.”[9].
2. Discovery and selection of development platforms for each service: this step allows us to choose a Cloud platform based on the module to be developed and technologies proposed by the PaaS platform. It includes the participation of the Cloud provider in the process of establishing the contract between the customer and the provider.

3. Automatic deployment and publication of the software artifact as a web service.

3.7. Tests and validation.

The tests are made once the services are discovered and developed. Two types of tests are considered: (i) the test of application features and the respect of business rules, with the aim of validating or bringing modifications to the expression of needs; (ii) application availability tests done automatically at various moments. Validation is made by the customer. It takes place after the deployment of the application and after the last tests.

3.8. IaaS selection for application deployment.

A Cloud infrastructure is selected according to performance indicators (KPIs) and SLA required by the client.

3.9. Automatic deployment.

This phase consists in automatically deploying the application on a Cloud infrastructure (IaaS).

In our methodology, the maintenance is not an occasional service; it takes place in a continuous way. Maintenance includes both changes made at any time by the business stakeholder, at any time, in the expression of its needs and web service changes (adding, deleting or modifying features). In other words, the phases of service discovery and composition do not stop. In the next section, we describe an architecture for Automatic Software Development as a Service.

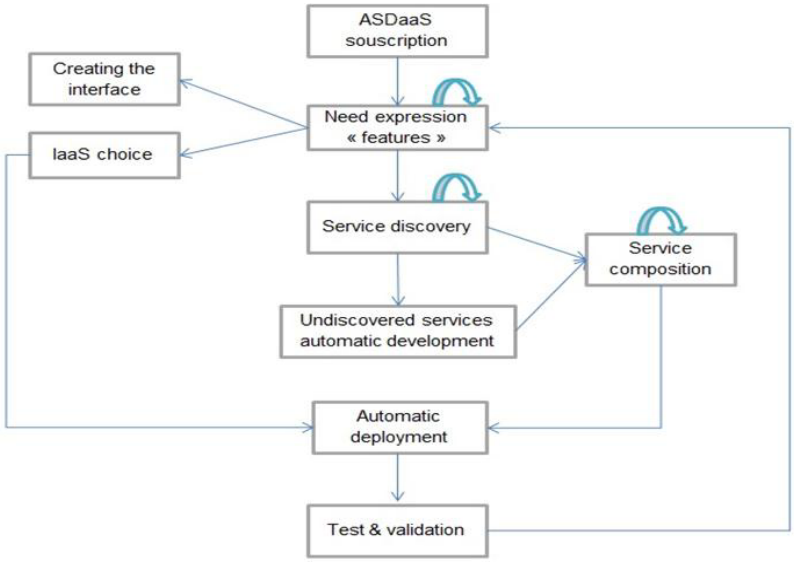

Figure 1. Cloud-based Service-Oriented software development methodology.

We propose an example to illustrate our methodology. We want to implement a service that can generate an itinerary of sightseeing tours, from a chosen destination and period and depending on the weather. As a first step, we define (i) the functional requirements and business rules, (ii) the SLA, (iii) the KPIs for IaaS selection for deployment, and for PaaS selection for automatic development of services. Figure 2 shows the functional requirements of a process.

These rules must be respected: (i) if the weather is good, we will favor outdoor tours; (ii) otherwise we will favor covered visits.

We put at the disposal of the user a tool to create an interface that can, in this case, generate a questionnaire where the user can enter the destination and the dates for the trip. Then based on KPIs and SLA, we proceed to the selection of a Cloud Infrastructure for deployment. In parallel, we discover services: we search for service S1 to estimate the weather for given period and destination, and service S2 to identify the sights of the destination, and finally, S3 to establish a visit itinerary. The service S3 was not found, so we will model the service and generate the code automatically on a Cloud platform, previously chosen, depending on the parameters (SLA, pre-cited KPIs, and PaaS APIs). These services are composed according to the discovery and the development. Then the resulting application is deployed on the preselected Cloud infrastructure.

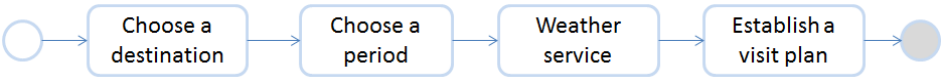

Figure 2. Functional requirements.

4. Automatic Software Development as a Service (ASDaaS)

ASDaaS is an automatic business process development environment. Its architecture is illustrated in figure 3.

The “project management” service basically takes care essentially of two activities: (i) The creation of a new project, and (ii) The monitoring of the existing projects.

“Prototype as a Service” allows the creation of a prototype from needs expression and interface creation. Features described by the business stakeholder are discovered by “Discovery as a Service”. For undiscovered services, “PaaS Discovery as a Service”, offers the possibility of their development on a specific PaaS platform according to: (i) PaaS characteristics; (ii) APIs offered by PaaS; and (iii) Respect for the SLA imposed by the client. Once the PaaS is selected, the service “Automatic Development as a Service” provides an automatic development from models. An automatic generation of source code is made thanks to a code generator which can transform models towards the code (approach MDD). Code generation is made by exploiting the APIs of chosen PaaS and by respecting the architecture imposed by the Cloud provider. The developed services are then published.

The service “Composition as a Service” composes discovered services and developed services. Service discovery and composition are continuous processes which do not stop even after the deployment of the application, with a view to ongoing im-

provement of the application. A CVS will manage different versions of composition in order to allow backtracking, if necessary.

A selection of IaaS for the deployment is made by the service "IaaS Discovery as a Service". This depends on a number of parameters such as (i) the characteristics of different IaaS, dependent on KPIs previously defined; and (ii) respect of the SLA imposed by the client. The various prototypes corresponding to the various compositions are deployed on the preselected infrastructure. The deployment is done throughout the development process to allow stakeholders to perform tests on the resulting prototypes.

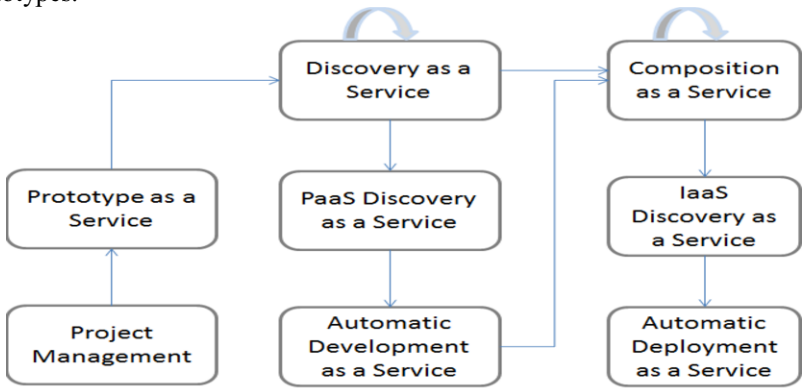

Figure 3. ASDaaS architecture.

5. Conclusion and future works

In this paper, we have described a methodology for Cloud-based collaborative software development, and then presented the ASDaaS architecture. This architecture is composed of eight services: Project Management, Prototype as a Service, Service Discovery as a Service, Composition as a Service, PaaS Discovery as a Service, Automatic Development as a Service, IaaS Discovery as a Service, and Deployment as a Service. These services collaborate to provide a composite software based on discovered and developed web services. We believe that the paradigm ASDaaS will change the role of software developers who will no longer have to worry about developing functionalities required by the client, but rather about ensuring compliance with security settings.

In our future work, we will focus on providing a safety layer in the "Service Discovery as a Service". Then we will use the approach based on QoS of Raluca and Florica [24] to select the web services that have the same functionalities, to enhance the results of discovery engine.

References

[1] Michael Armbrust , Armando Fox , Rean Griffith , Anthony D. Joseph , Randy H. Katz , Andrew Konwinski , Gunho Lee , David A. Patterson , Ariel Rabkin , Matei Zaharia: Above the Clouds: A Berkeley view of Cloud Computing. Technical Report No. UCB/EECS-2009-28 . (2009).

[2] Bharat Chhabra, Dinesh Verma, Bhawna Taneja: Software Engineering Issues from the Cloud Application Perspective. International Journal of Information Technology and Knowledge Management, pp 669-673. (2010).

[3] Yi Wei, Blake, M.B: Service-Oriented Computing and Cloud Computing : Challenges and Opportunities. Internet Computing, IEEE , pp 72-75. (2010).

[4] Peter MELL, Timothy Grance. The NIST Definition of Cloud Computing. Ntional Institute of Standards and Technology. (2011).

[5] Jan S. Rellermeyer, Michael Duller, Gustavo Alonso. Engineering the Cloud from Software Modules. CLOUD '09 Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. pp 32-37. (2009).

[6] Muhammad Ali Babar, Muhammad Aufeef Chauhan. A Tale of Migration to Cloud Computing for Sharing Experiences and Observations. SECLOUD '11 Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing. pp50-56. (2011).

[7] Chunming Rong, Son T. Nguyen, Martin Gilje Jaatun. Beyond lightning: A survey on security challenges in Cloud Computing. ELSEVIER : Computers and Electrical Engineering. pp 47–54 . (2012).

[8] Krzysztof Sledziewski, Bordbar, B.; Anane, R. A DSL-based Approach to Software Development and Deployment on Cloud. 24th IEEE International Conference on Advanced Information Networking and Applications. pp 414-421. (2010).

[9] Danilo Ardagna, di Nitto, E.; Mohagheghi, P.; Mosser, S.; Ballagny, C.; D'Andria, F.; Casale, G.; Matthews, P.; Nechifor, C.-S.; Petcu, D.; Gericke, A.; Sheridan, C. MODACLOUDS: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds. Modeling in Software Engineering (MISE), 2012 ICSE Workshop pp 50-56. (2012).

[10] MODAClouds : http://www.modaClouds.eu/.

[11] Hanu Kommalapati, William H. Zack. The SaaS Development Lifecycle. InfoQ: http://www.infoq.com/articles/SaaS-Lifecycle. (2011).

[12] Tobin J. Lehman, Sharma, A. Software Development as a Service: Agile Experiences. SRII Global Conference (SRII), 2011 Annual. pp 749-758. (2011).

[13] Radha Guha, Al-Dabass, D. Impact of Web 2.0 and Cloud Computing Platform on Software Engineering. International Symposium on Electronic System Design. pp 213-218 (2010).

[14] Hailong Sun, X. W, Xu Wang; Chao Zhou; Zicheng Huang; Xudong Liu. Early Experience of Building a Cloud Platform for Service Oriented Software Development. Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS). pp 1-4. (2010).

[15] http://www.auml.org/

[16] Mohammed AbuJarour, Felix Naumann, Mircea Craculeac. Collecting, Annotating, and Classifying Public Web Services. Proceeding OTM'10 Proceedings of the 2010 international conference on On the move to meaningful internet systems. pp 256-272. (2010).

[17] Joël Plisson, Nada Lavrac, Dunja Mladenic. A Rule based Approach to Word Lemmatization, SiKDD multiconference, 12-15 October, Ljubljana, Slovenia. (2004).

[18] Cohen, W. W., Ravikumar, P., Fienberg, S. E. A Comparaison of String Distance Metrics for Name-Matching Tasks, American Association for Artificial Intelligence (www.aaai.org). (2003).

[19] Lei Chen, Geng Yang, Dongrui Wang, Yingzhou Zhang. WordNet-powered Web Services Discovery Using Kernel-based Similarity Matching Mechanism. 2010 Fifth IEEE International Symposium on Service Oriented System Engineering. pp64-68. (2010).

[20] Saouli Hamza, Kazar Okba, Benharkat Aïcha-Nabila, Amghar Youssef. Web Services Discovery, Selection and Ranking Based Multi-Agent system in Cloud Computing Environment. International Journal of Information Studies, 4, pp. pp 123-147. (2012).

[21] http://www.Cloudbus.org/Cloudsim/

[22] The Aglets User's 2.0.2 manual, Aglets Development Group,(2009).

[23] Wickremasinghe, B., Calheiros, N. R., Buyya, R. CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications, http://www.gridbus.org/reports/CloudAnalyst 2009. pdf. (2010).

[24] Raluca Iordache and Florica Moldoveanu. A Conditional Lexicographic Approach for the Elicitation of QoS Preferences. On the Move to Meaningful Internet Systems: OTM 2012 pp 182-193. (2012).