Impact of Web 2.0 and Cloud Computing Platform on Software Engineering

Radha Guha
Dept. of Engineering Mathematics and Science
Texas A&M International University, TX 78041
Radha.guha@tamiu.edu

David Al-Dabass
School of Computing and Informatics
Nottingham Trent Univ., NG11 8NS
David.al-dabass@ntu.ac.uk

Abstract— **Current era of Web 2.0 is enabling new business models for using the semantic web. One such business model is leasing out computing platform of hardware and software over the internet to the tenants and is dubbed as Cloud Computing. The anticipated future trend of computing is believed to be this cloud computing as it promises a lot of benefits like no capital expenditure, speed of application deployment, shorter time to market, lower cost of operation and easier maintenance for the tenants. This paper analyses how cloud computing on the background of Web 2.0 is going to impact the software engineering process to develop quality software. As the cloud provider is an external entity or third party, how difficult will be the interaction with them? How to separate the roles of SW engineers and cloud providers? SW engineering should include framework activities to leverage all the benefits of cloud computing systematically and strategically. This paper extends the traditional agile process model named Extreme Programming (XP) and integrates interaction with the cloud provider to facilitate acceptance of cloud computing.**

Keywords- *Web 2.0; Cloud Computing; Software Engineering; Agile Process Model*

I. INTRODUCTION

In the era of semantic web or Web 2.0 [1], [2], [3], [4] emergence of several web technologies are enabling innovative use of the web. In Web 2.0, metadata written in XML (extensible markup language) describing the web content can be read and processed by the computers automatically. Other XML based web protocols like service oriented architecture (SOA), simple object access protocol (SOAP), web service description language (WSDL) and universal description, discovery and integration (UDDI) of web are capable of integrating applications developed on heterogeneous computing platforms, operating systems and with varities of programming languages. With this capability of data integration and data exchange between heterogeneous applications, new business models of application deployment and delivery over the internet have been conceptualized. Applications can be hosted on the web and accessed via the internet by geographically dispersed clients. These interoperable applications hosted on the web for use by multiple clients remotely are called Web Services which can even be discovered on the fly with no prior knowledge of their existence. As the same service will be catered to multiple clients they can even be customized according to clients' likes. Application architecture and delivery architecture will be two separate layers for providing this flexibility.

Applications like Hadoop and Mashup [5], [6], which combine data and functionalities from multiple external sources hosted as web services are producing valuable aggregate new information and creating new web services. Hadoop and Mashup can support high performance computing involving distributed file system with petabytes of data.

In another business model, the application development infrastructure like processors, storage, memory, operating system and application development tools and software can all be delivered as utility to the clients over the internet. This is what is dubbed as cloud computing where a huge pool of physical resources hosted on the web will be shared by multiple clients as and when required. Because of the many benefits of this business model like no capital expenditure, speed of application deployment, shorter time to market, lower cost of operation and easier maintenance of resources for the clients, cloud computing may be the prevalent computing platform of the future.

This paper analyzes impact of Web 2.0 and cloud computing platform on software engineering process to develop quality software (SW). Economies of all developed countries depend on quality SW and SW cost is more than hardware (HW) cost. Moreover because of the involvement of many parties, SW development is inherently a complex process and most of the SW project fails because of lack of communication and coordination between all the parties involved.

The main thesis of this paper is that the prevalent SW process models should involve the cloud provider in every steps of decision making in software development life cycle to make the software project a success. In Section II, background literature on cloud computing and software engineering is surveyed. How the software developer is coping with the changing trend of application development with Web 2.0 protocols and application deployment over the web is reported. In Section III, challenges of cloud computing platform for software engineering is analyzed. In Section IV and V, an agile process model which incorporates interaction with cloud provider is proposed and analyzed. Section VI concludes the paper.

II. LITERATURE SURVEY

A. Cloud Computing

Cloud computing [7], [8], [9] is the future trend of computing. Cloud computing is the idea of renting out

978-0-7695-4294-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ISED.2010.48

server, storage, network, software technologies, tools and applications as utility or service over the internet as and when required in contrast to owning them permanently.

Depending on what resources are shared and delivered to the customers, there are 4 types of cloud computing. In cloud computing terminology when hardware such as processors, storage and network are delivered as a service it is called infrastructure as a service (IaaS). Examples of IaaS are Amazon's Elastic Cloud (EC2) and Simple Storage Service (S3). When programming platforms and tools like Java, Python, .Net, MySQL and APIs are delivered as a service it is called platform as a service (PaaS). When applications are delivered as a service it is called software as a service (SaaS).

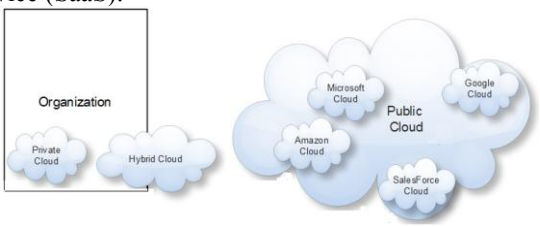

Figure 1: Cloud Computing Platform

Depending on the amount of self governance or control on resources by the tenant there are 3 types of cloud like internal or private cloud, external or public cloud and hybrid cloud (Figure 1). In private cloud an enterprise owns all the resources on-site and shares them between multiple applications. In public cloud the enterprise will rent the resources from an off- site cloud provider and these resources will be shared between multiple tenants. Hybrid cloud is in the middle where an enterprise owns some resources and rents some other resources from a third party.

Cloud computing is based on service oriented architecture (SOA) of Web 2.0 and virtualization [10], [11] of hardware and software resources (Figure 2). Because of the virtualization technique, physical resources can be linked dynamically to different applications running on different operating systems. Because of the virtualization technique, physical resources can be shared amongst all users and there is efficient resource management which can provide higher resource utilization and on-demand scalability. Increased resource utilization brings down the cost of floor space and cost of power and cooling. Power savings is the most attractive feature of cloud computing and is the renewed initiative of environment friendly green computing or green IT movement of today.

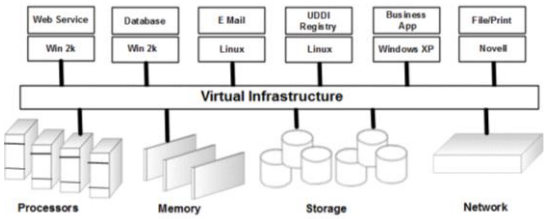

Figure 2: Virtual Infrastructure [10]

Cloud computing not only reduces cost of usage of resources but also reduces maintenance cost of resources for the users.

Cloud computing can support on-demand scalability. An application with occasional demand for higher resources will pay for the higher resources only the time it is used instead of leasing all the resources from the very beginning in anticipation of future need. This fine-grained (hourly) pay-by-use model of cloud computing is going to be very attractive to the customers.

There are many other benefits of cloud computing. Cloud infrastructure can support multiple protocols and change in business model for applications more rapidly. It can also handle increased performance requirements like service scaling, response time and availability of the application, as the cloud infrastructure is a huge pool of resources like servers, storage and network and provide elasticity of growth to the end users.

With this business model of catering multiple clients with shared resources, world's leading IT companies like Microsoft, Google, IBM, SalesForce, HP and Amazon are deploying clouds [Figure 1]. Web services, applications like Hadoop and Mashup can run on these clouds. Because of all its advantages, this cloud computing model may be the prevalent computing model of the future.

In the next sections we first delve into software development methodologies to develop quality software products in traditional environment not involving web services and cloud computing platform. We then analyze the challenges of the current business model of application development and deployment involving web 2.0 and cloud computing platform. Finally we suggest methodologies to develop quality SW that will push forward advances of the cloud computing platform.

B. Software Engineering

Over the last half-century rapid advances of hardware technology such as computers, memory, storage, communication networks, mobile devices and embedded systems is pushing the need for larger and more complex software. Software development not only involves many different hardware technologies, it also involves many different parties like customers, end users and software developers. That's why SW development is an inherently complex procedure. Since 1968 software developers had to adopt the engineering disciplines i.e. systematic, disciplined and quantifiable approach to make software development more manageable to produce quality software products. The success or quality of a SW project is measured by whether it is developed within time and budget and by its efficiency, usability, dependability and maintainability [12], [13].

Software engineering starts with an explicit process model having framework of activities which are synchronized in a defined way. This process model describes or prescribes how to build software with intermediate visible work products (documents) and the final finished product i.e. the

operating SW. The whole development process of SW from its conceptualization to operation and retirement is called the software development life cycle (SDLC).

SDLC goes through several framework activities like requirements gathering, planning, design, coding, testing, deployment, maintenance and retirement. These activities are synchronized in accordance to the process model adopted for a particular software development. There are many process models to choose from like water fall model, rapid application development (RAD) model, and spiral model depending on the size of the project, delivery time requirement and type of the project. As for example development of an avionic embedded system will adopt a different process model from development of a web application.

Even though software engineering takes engineering approach, success of SW product is more difficult than products from other engineering domain like mechanical engineering or civil engineering. This is because software is intangible during its development. Software project managers use a number of umbrella activities to monitor the software framework activities in a more visible way. These umbrella activities are SW project tracking and control, risk management, quality assurance, measurements, configuration management, work-product or documents generation, review and reusability management. CMMI (Capability maturity model integration) is a software process improvement model for software development companies by comparing their process maturity with the best practices in the industry to deliver quality software products.

Even after taking all these measures of sticking to the plan and giving much importance to document generation for project tracking and control, many SW projects failed. More than 50% of software projects fail due to various reasons like schedule and budget slippage, non user friendly interface of the SW and non-flexibility for maintenance and change of the SW. And the reasons for all these problems are lack of communication and coordination between all the parties involved.

Requirement changes of a SW are the major cause of increased complexity, schedule and budget slippage. Incorporating changes at a later stage of SDLC increases cost of the project exponentially (Figure 3). Adding more number of programmers at a later stage does not solve the schedule problem as increased coordination requirement slows down the project further. It is very important that requirements gathering, planning and design of the SW is done involving all the parties from the beginning.

That's why several agile process models like Extreme Programming (XP), Scrum, Crystal and Adaptive etc. have been introduced in mid 1990s to accommodate continuous changes in requirements during the development of the software. These agile process models have shorter development cycles where small pieces of work are "time-boxed", developed and released for customer feedback, verification and validation iteratively. One time-box takes few weeks to maximum a month of time. Agile process model is communication intensive as customer satisfaction is given the utmost importance.

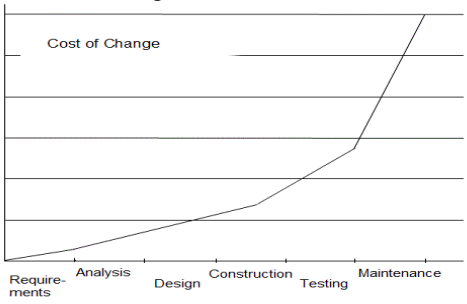

Figure 3: Economics of Software Development

Agile software development is possible only when the SW developers are talented, motivated and self-organized. Agile process model eliminates the exponential increase of cost to incorporate changes as in the waterfall model by keeping the customer involved throughout and validating small pieces of work by them iteratively. These agile process models work better for most of the SW projects as changes are inevitable and responding to the change is key to the success of a project.

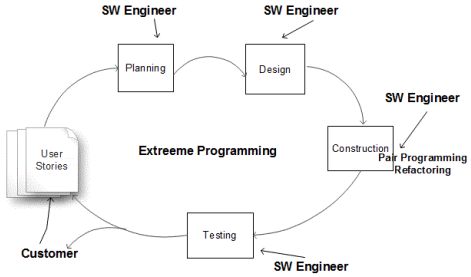

Figure4: Extreme Programming Process Model

Figure 4 depicts the steps of agile process model named extreme programming (XP) for a traditional SW development where the customer owns the developing platform or SW developers develop in- house and deploy the SW to the customer after it is built. XP has many characteristics like user story card, CRC (class, responsibility, collaboration) card narrated during the requirement gathering stage jointly by the customer and the SW engineers. Customer decides the priority of each story card and the highest priority card is only considered or "time-boxed" for the current iteration of SW development. Construction of code is performed by two engineers sitting at the same machine so that there is less scope of errors in the code. This is called pair programming. Code is continuously re-factored or improved to make it more efficient.

C. How SW development industry is surviving in the cloud computing age?

This section surveys how SW development industry is trying to survive in the era of Web 2.0 with web services

and cloud computing. In reference [14], they present framework activities for designing applications based on discovery of semantic web service using software engineering methodologies. They propose generating semiautomatic semantic description of applications exploiting the existing methodologies and tools of web engineering. This increases design efficiency and reduces manual effort of semantically annotating the new application composed from web services of multiple enterprises.

In reference [15], Salesforce.com finds that agile process model works better on cloud computing platform. Before cloud computing, release of the SW to the user took time and getting feedback from the customer took more time which thwarted the very concept of agile development. Whereas now a new releases of the SW can be uploaded on the server and used by the users immediately. Basically in this paper what they have described is the benefits of software as a service hosted on the internet and how it complements agile computing methodology. They have not considered the challenges of cloud computing in developing new business software.

Cloud computing being the newest hype of the IT industry, the challenges of software engineering on cloud computing platform have not been studied yet and no software development process model for cloud computing platform has been suggested yet. We analyze the challenges of the cloud computing platform on SW development process and suggest extending the existing agile process model, named extreme programming to mitigate all the challenges in Section III below.

III. ANALYSIS

A. Impact of Cloud Computing on Software Engineering: Challenges

In the rapidly changing computing environment with web services and cloud platform, SW development is going to be very challenging. SW development process will involve heterogeneous platforms, distributed web services, multiple enterprises geographically dispersed all over the world. Existing software process models and framework activities are not going to be adequate unless interaction with cloud providers is included.

Requirements gathering phase so far included customers, users and software engineers. Now it has to include the cloud providers as well, as they will be supplying the computing infrastructure and maintain them too. As the cloud providers only will know the size, architectural details, virtualization strategy and resource utilization % of the infrastructure, planning and design phases of SW development also have to include the cloud providers. The cloud providers can help in answering these questions on: 1) How many developers are needed, 2) Component Reuse, 3) Cost estimation, 4) Schedule Estimation, 5) Risk Management, 6) Configuration Management, 7) Change Management, and 8) Quality Assurance.

Because of the component reuse of web services the size of the software in number of kilo- lines of code (KLOC) or number of function points (FP) to be newly developed by the SW engineer will reduce but complexity of the project will increase many folds because of lack of documentations of implementation details of web services and their integration requirements. Only description that will be available online is the metadata information of the web services to be processed by the computers automatically.

Only coding and testing phases can be done independently by the software engineers. Coding and testing can be done on the cloud platform which is a huge benefit as everybody will have easy access to the software being built. This will reduce the cost and time for testing and validation.

But software developers have to use the web services and open-source software freely available from the cloud instead of procuring them. Software developers should have more expertise in building software from readily available components than writing it all and building a monolithic application. Refactoring of existing application is required to best utilize the cloud infrastructure architecture in a cost effective way. In latest hardware technology the computers are multi-core and networked and the SW engineers should train themselves in parallel and distributed computing to complement this advances of HW and network technology. SW engineers should train themselves in internet protocols, XML, web service standards and layered separation of concerns of SOA architecture of internet to leverage all the benefits of Web 2.0. Cloud providers will insists that software should be as modular as possible for occasional migration from one server to another for load balancing as required by the cloud provider [9].

Maintenance phase also should include the cloud providers. There is a complete shift of responsibility of maintenance of the infrastructure from software developers to cloud providers. Now because of the involvement of the cloud provider the customer has to sign contract with them as well so that the “Software Engineering code of ethics” are not violated by the cloud provider. In addition, protection and security of the data is of utmost importance which is under the jurisdiction of the cloud provider now.

Also occasional demand of higher resource usage of CPU time or network from applications may thwart the pay-by-use model of cloud computing into jeopardy as multiple applications may need higher resource usage all at the same time not anticipated by the cloud provider in the beginning. Especially when applications are deployed as “Software-as-a-Service” or “SaaS” model, they may have occasional workload surge not anticipated in advance.

Cloud provider uses virtualization of resources technique to cater many customers on demand in an efficient way. For higher resource utilization occasional migration of application from one server to another or from one storage to another may be required by the cloud provider. This may

be a conflict of interest with the customer as they want dedicated resources with high availability and reliability of their applications. To avoid this conflict cloud providers need to introduce quality of service provisions for high priority tenants.

Now we analyze how difficult will be the interaction between cloud providers and the software engineers? The amount of interactions between software engineers and cloud providers will depend on type of cloud like public, private and hybrid cloud involvements. In private cloud there is more control or self governance by the customer than in public cloud. Customer should also consider using private cloud instead of using public cloud to assure availability and reliability of their high priority applications. Benefits of private cloud will be less interaction with cloud provider, self governance, high security, reliability, availability of data [Figure 5]. But cheaper computing on public cloud will always outweigh the benefits of less complexity of SW development on private cloud platform and is going to be more attractive.

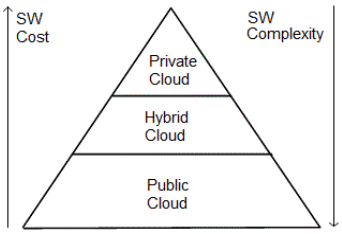

Figure 5: Economics vs. Complexity of Software Development on Cloud Computing Platform

IV. PROPOSED SW PROCESS MODEL

Innovative software engineering is required to leverage all the benefits of cloud computing and mitigate its challenges strategically to push forward its advances. Here we propose an extended version of Extreme Programming (XP), an agile process model for cloud computing platform and name it Extreme Cloud Programming [Figure 6].

All the phases like planning, design, construction, testing and deployment need interaction with the representatives from cloud provider. The roles or activities by the cloud provider and SW developers are separated and listed in Table 1. Resource accounting on cloud platform will be done by the cloud provider in the requirement gathering phase. Software architecture, software architecture to hardware architecture mapping, interface design, data types design, cost estimation and schedule estimation of the project all should be done in collaboration with the cloud provider. During the construction phase of the application if web services are integrated where many different enterprises are involved then error should be mitigated with the mediation of the cloud provider. Maintenance contract with cloud provider will be according to the Quality of Service agreement.

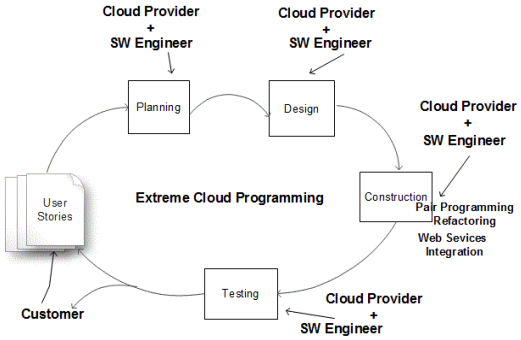

Figure 6: Extreme Cloud Programming

Table 1: SW Engineering- Role Separation

Activity	Roles	
	SW Developer	**Cloud Provider**
Requirement Gathering	Elicitation	Resource Accounting Virtual Machine
Analysis	SW Modules	SW/HW Architecture
Design	Interface Design Data Types Cost Estimation Schedule Estimation	Component Reuse
Construction	Coding Integration of Web Services	Implementation Details
Testing	Unit Test Integration Test	Integration Test
Deployment		Operation & Maintenance

A software metric is required for effort estimation of SW development using the new extreme cloud programming process model. This metric is required as American consultant Tom DeMarco aptly stated in 1997 in his book [16] about managing risk in software projects that "You cannot control what you cannot measure". Constructive cost estimation model (COCOMO) is mostly used model for cost estimation of various SW development projects. In COCOMO model [12] three classes of SW projects have been considered so far. These SW projects are classified as 1) Organic, 2) Semi-detached, 3) Embedded according to the SW team size, their experiences and development (HW, SW and operations) constraints. We extend this cost estimation model with a new class of SW project for cloud computing platform. In basic COCOMO model effort (man month), development time (months) and no. of people required are given by the following equations.

Effort Applied $= a(KLOC)^b$ [man-months]

Development Time $= c(\text{Effort Applied})^d$ [months]

No. of People = Effort Applied/Development Time [no.]

V. RESULTS AND DISCUSSIONS

In this section we experiment with the typical values of a, b, c, d for Cloud Computing class of SW projects in comparison to other classes. The typical values of the

coefficients a, b, c, d for different classes of SW projects are listed in Table 2.

Table 2: COCOMO

SW Proj.	a	b	c	d
Organic	2.4	1.05	2.5	.38
Semi-Detached	3.0	1.12	2.5	.35
Embedded	3.6	1.2	2.5	.32
Cloud Computing	**4**	**1.2**	**2.5**	**.3**

In anticipation of extra interaction complexity with the cloud providers coefficient *a* is increased to 4 for cloud computing platform. Coefficients *a*, *b* for cloud computing are determined so that the effort curve is steeper than the other 3 classes but is linear like the other 3 classes. Similarly coefficients c, d for cloud computing are determined so that the development time curve is less steeper than the other 3 classes but is linear like the other 3 classes. We adjusted coefficients *a*, *b*, *c*, *d* in cloud computing to new values of 4, 1.2, 2.5 and .3.

Because of component reuse, SW development with cloud computing will reduce KLOC (kilo lines of code) significantly. We deduce new KLOC = *i**C + (KLOC)*C where C is the % of component reuse and *i* is the coefficient adjustment for new interface design effort.

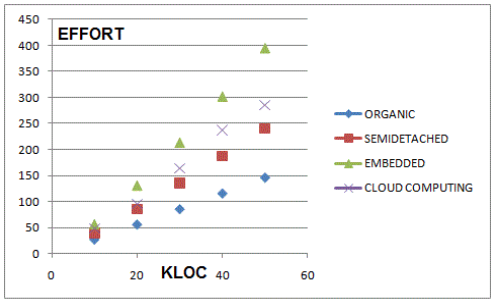

Figure 7: Extended COCOMO For SW Effort Estimation

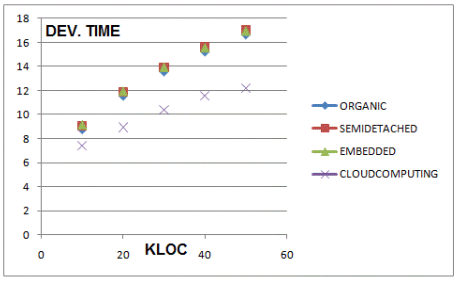

Figure 8: Extended COCOMO For SW Dev. Time

Figure 7 plots SW effort estimation for project size varying from 10 to 50 KLOC for all 4 classes of projects. We assumed 30% component reuse in cloud computing case. If more % of component reuse is possible it will mitigate the higher interaction complexity in coefficient *a* and will be beneficial for cloud computing platform. Figure 8 plots the corresponding SW development time estimation for all 4 classes of SW projects. With 30% component reuse possibility, SW development on cloud computing platform will take least amount of time.

VI. CONCLUSION

Cloud computing is a paradigm shift over traditional way of developing and deploying of software. This will make software engineering more difficult as they have to interact with a third party called the "cloud provider". The amount of work required for developing software will reduce but there will be added communication and coordination requirement with the cloud provider which makes software development project more complex. The main thesis of this paper is that the prevalent SW process models should incorporate this new dimension of interaction with the cloud provider and separate roles of SW engineers and cloud providers. A new agile process model is proposed in this paper which includes the anticipated interaction requirement with the cloud provider which will mitigate all the challenges of software development on cloud computing platform and make it more advantageous to develop and deploy software on the cloud computing platform.

REFERENCES

[1] Radha Guha. Toward The Intelligent Web Systems. In *Proceedings of IEEE CS, First International Conference on Computational Intelligence, Communication Systems and Network*, Pages 459-463, July 2009.

[2] J. Handler, N. Shadbolt, W. Hall, T. Berners-Lee and D. Weitzner. Web Science: An Interdisciplinary Approach to Understanding the Web. *Communications of the ACM, Vol. 51, No. 7*, July 2008.

[3] F. Chong and G. Carraro. Architecture Strategies for Catching the Long Tail. *Microsoft Corporation*, April 2006.

[4] J. Banerjee and S. Aziz. SOA: The missing link between Enterprise Architecture and Solution Architecture. *SETLabs briefing, Vol. 5, No 2*, Pages 69-80, March 2007.

[5] HADOOP. *http://en.wikipedia.org/wiki/Hadoop*, February 2010.

[6] D. Taft. IBM's M2 Project Taps Hadoop for Massive Mashups. *www.eweek.com,* February 2010.

[7] Sun Microsystem. Introduction to Cloud Computing architecture. White Paper, 1st Edition, June 2009

[8]Sun Microsystem. Open Source & Cloud Computing: On-Demand, Innovative IT On a Massive Scale.

[9] A. Singh, M. Korupolu, D. Mahapatra. Server-Storage Virtualization: Integration and Load Balancing in Data Centers. *IEEE/ACM Supercomputing (SC)*, 2008

[10] VMWARE. Virtualization Overview. *www.vmware.com*.

[11]Reservoir Consortium. Resources and Services Virtualization without Barriers. Scientific Report. 2009.

[12] R. Pressman. Software Engineering: A Practitioner's Approach.7th Edition. *McGraw-Hill Higher Education* (2009).

[13] I. Sommerville. Software Engineering, 8th Edition, *Pearson Education*, 2006.

[14] M. Brambilla et al. A Software Engineering Approach to Design and Development of Semantic Web Service Applications

[15] Salesforce.com. Agile Development Meets Cloud Computing for Extraordinary Results. *www.salesforce.com*

[16] T. DeMarco and T. Lister. Waltzing with Bears: Managing Risk on Software Projects, Dorset House Publishing Company, Incorporated. March 2003.