Evolution of Software Engineering in the Changing Scenario of Modern Hardware Architecture, Semantic Web and Cloud Computing Platform

Radha Guha
PES Institute of Technology/Bangalore, India
radhaguha@pes.edu

***Abstract*—**Traditional way of software engineering is no longer fully suitable in the changing scenario of modern hardware and software architecture of parallel and distributed computing on Semantic web and Cloud computing platform. A parallel hardware architecture can support high performance computing but needs changes in programming style. Also the capability of Semantic web can link everything on the internet making an interoperable intelligent system. And with this capability many beneficial business models like Web services and Cloud computing platform have been conceptualized. Cloud computing is the most anticipated future trend of computing. These changes in hardware and software architecture means we need to re-visit the traditional software engineering process models meant for a single computer system. This paper first surveys the evolution of hardware architecture, newer business models, newer software applications and then analyses the need for changes in software engineering process models to leverage all the benefits of the newer business models. This paper also emphasizes the vulnerability of the web applications and cloud computing platform in terms of risk management of web applications in general and privacy and security of customer information in shared cloud platform which may threaten the adoption of the cloud platform.

***Keywords/Index Terms*—** Agile Process Model , Cloud Computing Platform, Privacy and Security Issues, Risk Management, Semantic Web, Software Evolution.

1. INTRODUCTION: CHANGES IN HW INFRASTRUCTURE

Since the inception of computers in the 1950's hardware technologies have improved rapidly and cost of computing has become cheaper. Rapid changes of hardware technologies in all fronts like CPU, memory, hard disk, communication network and embedded systems made requirements for diversified software applications. In 1950's software program was for a particular hardware platform and a super computer will serve to scientific computations only for big companies with million dollars contract. These super computers were expensive and needed large floor area, electricity consumption and large number of administrative staff. Thus cost of computing was very high in those days. Programmers used these super computers on a time sharing basis and used machine codes to write their programs and job control languages to run their programs. Then from 1970's came the era of minicomputers and desktop computers where business applications will be installed in-house for a company and several end-users will operate it. Customer call-centers were established so that customers can get the required services through the end users of these business applications.

Since the inception of World Wide Web by Tim Barner Lee's in 1990's (Christian Bizer et al. 2012), business applications can be accessed over the internet by the customers themselves for getting the required services. Also many applications can be downloaded from the internet to run on one's personal computer. Not only big businesses, many individuals and home grown businesses had their web presence either for information sharing, social networking or for e-commerce. They designed their web applications ad-hoc with scripting languages like HTML, XML, Java Script, PHP, ASP etc. These web applications are designed quickly by armatures and not by IT professionals following SW engineering processes models and thus are of poor quality. This era also gave rise to pervasive and ubiquitous computing with several smart embedded systems with very tiny computers embedded in small hand-held and portable devices requiring no human interactions but still providing complex intelligent services.

Now a day many commercial applications as well as scientific applications need to process huge amount of data real time with complex algorithms for modeling and simulation purposes requiring more processing power of the computers. Advancement in network infrastructure made parallel and distributed computing a norm instead of sequential processing by a single computer. These high processing computing (HPC) needs can also be supported by application specific integrated circuit (ASIC) or reconfigurable field programmable gate array (FPGA) platform as process technology can integrate huge number of high speed

transistors on the same chip where a number of processors can be configured to run concurrently.

With this changing scenario of hardware infrastructure, requirements for various software delivery architectures have emerged. Thus traditional SW process models are needed to be updated which is the topic of this paper. Section II highlights the emergence of newer business models for application delivery. Section III highlights the evolvement of generation of programming languages. Section IV emphasizes the importance of quality software development in today's economy and explores traditional SW processing models. Section V analyzes need for newer SW process models. Section VI concludes the paper.

2. SEMANTIC WEB AND CLOUD COMPUTING PLATFORM

Tim Barner Lee's vision of the development of Semantic Web or Web 3.0 (Handler et al., 2008; Brand Niemann et al., 2005; Erin Cavanaugh, 2006) can transform the World Wide Web into an intelligent web system of structured, linked data which can be queried and inferred as a whole by the computers themselves. This Semantic Web capability is materializing many innovative use of the web such as hosting Web services and the cloud computing platform. Applications can be hosted on the web and accessed via Internet by geographically dispersed clients. These XML (eXtensible Markup Language) based, interoperable applications are called Web Services which can publish their location, functions, messages containing the parameter list to execute the functions and communication protocols for accessing the service for using them correctly by all. 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 these web applications for providing this flexibility. XML based Web 2.0 and Web 3.0 protocols like Service Oriented Architecture (SOA), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) registry are designed to discover Web Services on-the-fly and to integrate applications developed on heterogeneous computing platforms, operating systems and with varieties of programming languages.

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 instead of clients owning them permanently. This is what is dubbed as cloud computing (Duane Nickull et al., 2007; Sun Microsystem, 2009; Sun Microsystem, 2012; Radha Guha et al., 2010; Radha Guha, 2013) 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.

3. GENERATIONS OF PROGRAMMING LANGUAGES

Over the years many programming languages have evolved to have better programming style and manage complexity of larger program. In the beginning i.e. in 1950's assembly language which is symbolic binary machine code were used to write hardware specific code. This can be identified as first generation of programming language. One benefit of hardware specific assembly language is that it is very efficient in its performance as it can fine-tune its operation on a specific hardware. But the problem with machine code is that it is not human readable. It is extremely laborious to write a program with machine code and thus is error prone.

Problems using machine codes gave rise to high level programming languages which is easy to understand and manageable by the programmers. Programs became much larger and more complex than that in the previous generation. System software and application software needed to be segregated to make things more manageable. Where system software known as the operating system managed the hardware resources more efficiently, application software took care of the business needs.

Second generation of programming languages are all high level languages to make it human readable and more manageable. Fortran (Formula Translation) was the first high level human readable imperative language designed in 1950's which needed a compiler to translate it to machine code. Fortran was mainly for scientific computations. Other second generation languages that evolved in 1950's were LISP (list programming), COBOL (COmmon Business Oriented Language) and ALGOL (ALGOrithmic Language). Though these languages were high-level, they were still dependent on hardware domain and flow of control of these languages were with "jump", "goto" statements and was loopy. Also there were no construct for recursion and local variables. Dynamic memory creation was also not an option. LISP had recursion. In 1960's ALGOL60 and BASIC (Beginner's All-purpose Symbolic Instruction Code) were developed.

In third generation programming languages the program was structured with hierarchy of blocks and function calls. Code reuse was possible because of the sub-routines or function calls. Programmers had to write fewer amounts of codes and thus made fewer

mistakes. All previous generation of programming languages were retrofitted to incorporate this structured programming feature. In 1970's PROLOG (PROgramming LOGic) was developed as a logic processing language. Mid-level programming language C was developed for structured programming with function calls to reuse code. In C language it is also possible to use assembly code and so it is called mid-level language. To manage the complexity of large program object oriented programming (OOP) concept was proposed in Simula67. Later in 1980's C language was extended to C++ to incorporate OOP concepts like abstraction, encapsulation, inheritance, polymorphism to enhance code reuse and better organizing a program.

In 1990's many fourth generations programming languages were designed for reducing programmer's effort further and to reduce software development cost. Unlike the general purpose languages of third generation these fourth generations programming languages are for specific domains such as database query language SQL, Oracle Forms, report generator, Visual Basic, GUI creator, FoxPro and web development languages. They are often used for business development quickly such as with Visual Basic, PowerBuilder and ColdFusion in the front end and SQL for connecting to the backend database. With the popularity of Internet many scripting languages are developed. These are JavaScript, ASP, (application server pages), VBScript, Python, PHP etc.

4. SOFTWARE IS INDISPENSIBLE

Economies of all developed countries depend on quality software and today software cost is more than hardware cost. 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. Systems like manufacturing, monitoring, surveillance, computer aided design, diagnosis and various other decision support systems are software controlled. Thus cost of software failure is dire. Software development not only involves many different hardware technologies, it also involves many different parties like customers, stakeholders, end users and software developers. That's why software development is an inherently complex procedure. Because of the involvement of many parties, software development is inherently a complex process and most of the software projects fail because of lack of communication and coordination between all the parties involved.

Knowledge management in software engineering has always been an issue which affects better software development and its maintenance. There is always some gap in understanding about what the business partners and stakeholders want; how software designers and managers design the modules and how software developers implement the design. As the time passes, this gap in understanding increases due to the increased complexity of involvement of many parties and continuously changing requirements of the software. This is more so at the later stage when the software has to be maintained and no one has the comprehensive knowledge about the whole system. In the next section we delve into existing software development methodologies first to develop quality software products in traditional environment not involving Web Services and cloud computing platform.

4.1. Traditional Software Engineering Process

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 software project is measured by whether it is developed within time and budget and by its efficiency, usability, dependability and maintainability (M. Brambilla et al., 2006; R. Presman, 2009; Sommerville, 2006; Salesforce.com, 2009).

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 software. The whole development process of software 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. Software requirements are categorized as functional, contractual, safety, procedural, business and technical specification. Accuracy of requirements gathering is very important as errors in requirement gathering will propagate through all other subsequent activities.

Requirements arising from different sectors need to be well documented, verified to be in compliance with each other, optimized, linked and traced. All software engineering process 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 changes and type of the project. As for example, development of an avionic embedded system will adopt a different process model than development of a web application. Another criterion for choosing a suitable process model is its ability to arrest errors in requirement gathering.

Even though software engineering takes engineering approach, success of software 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 software framework activities in a more visible way. These umbrella activities are software 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 for sticking to the plan and giving much importance to document generation for project tracking and control, many software projects failed. Often time volume of paper documents is too large for aggregating information by humans. More than 50% of software projects fail due to various reasons like schedule and budget slippage, non-user friendly interface of the software and non-flexibility for maintenance and change of the software. And the reasons for all these problems are lack of communication and coordination between all the parties involved.

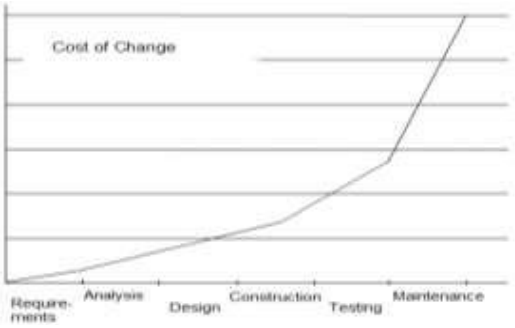
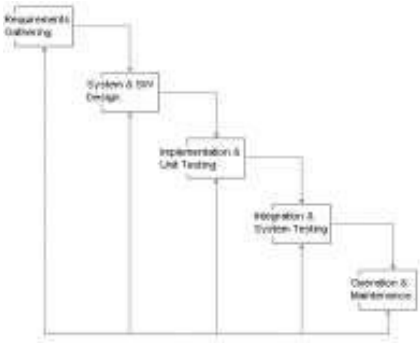


Fig. 1. Economics of Software Development

Requirement changes of a software 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 1). 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 software is done involving all the parties from the beginning. That's why several agile process models like Extreme Programming (XP) (Figure 2), Scrum, Crystal and Adaptive etc. have been introduced in mid 1990s to accommodate continuous changes in requirements during the development of the software instead of Water Fall process model which had no such scope. These agile process models have shorter development cycles where small pieces of work are "timeboxed", 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. Agile software development is possible only when the software 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 software projects as changes are inevitable and responding to the changes is the key to the success of a project.

Figure 2 depicts the steps of agile process model named extreme programming (XP) for a traditional software development where the customer owns the developing platform or software developers develop in- house and deploy the software 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 software 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 software 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.



(a) Water Fall Process Model

(b) Extreme Programming Process Model

Fig. 2. Software Engineering Process Models

In the next sections analysis for the need for changing programming style for parallel and distributed computing is emphasized first. Also producing software development artifacts for the

Semantic Web and the challenges of the current business model of application development and deployment involving Web 2.0, Web 3.0 technologies and cloud computing platform are analyzed. Finally methodologies to develop quality software that will push forward the advances of the cloud computing platform have been suggested.

5. NEED FOR MODIFICATION OF SOFTWARE ENGINEERING: ANALYSIS

In latest hardware technology the computers are multi-core and networked. SW engineers should train themselves in parallel and distributed computing to complement this advances of hardware and network technology. Many modeling and simulation software needs to process huge amount of streaming data with complex algorithms real time which is possible only by this parallel hardware architecture. Though need for HPC and the parallel hardware architecture such as grid computing, ASIC and reconfigurable FPGA computing are evolving over the last three decades, the application developers are still not familiar with the parallel programming styles and exploration of the parallel resources of the modern hardware platform. One of the most important challenges of HPC is to transform the sequential program written for classic Von Neumann architecture for parallel processors of the hardware. Task level parallelism can be discovered by the programmers and exploited by parallel hardware architecture. POSIX thread programming can fork multiple child threads with sub tasks assigned to them and can synchronize operations by joining the threads after their executions. OpenMP is another application programming interface (API) where the programmers can give directives to the compiler to identify the task level parallelism and schedule task on multiple processors according to the programmer directives. This involves additional libraries and tools as part of the OS for process synchronization, control and data communication between processes. The well-known examples of such libraries are POSIX thread for thread programming, OpenMP, Massage Passing Interface (MPI), Parrallel Virtual Machine (PVM) to provide application developers with distributed communication and synchronization facilities.

5.1. Need for Semantic Web Enabled Software Artifacts

Inclusion of the Free/Libre/Open Source Software (FLOSS) (Sun Microsystem, 2012), Component of the Shelf (COTS) pieces and Web Services, software development complexity is going to increase manifold because of the synchronization needs with third party software. Automatic discovery and integration with Web Services will reduce the amount of work in terms of kilo-line of code (KLOC) or function points (FP) required for developing software on semantic web but there will be added semantic skill requirements. Software developers need to change their software artifacts from plain text documents to machine readable structured linked data, to make them Semantic Web ready. Once the software engineers grasp the Semantic Web technologies, understand their capabilities and their many advantages like interoperability, adaptability, integration ability of open and distributed software components with other applications, they will make their software artifacts Semantic Web ready. With this semantic transformation knowledge management in software engineering will be much easier and compliance checking of various requirements during project planning, design, development, testing and verification can be automated. All requirements can be optimized, linked and traced. Aggregating of information from requirements document will be easy and impact analysis before actual changes are made can be done more accurately. Increased maintainability of software will also increase reliability of the software. Semantic artifacts will also give their product a competitive edge for automatic discovery and integration with other applications and efficient maintenance of their artifacts. Semantic Web services which can be linked with other web services will create new and more powerful software applications, encourage reuse and reduce redundancy.

5.2. Impact of Cloud Computing on Software Engineering

Like cloud computing platform, software development process will also involve heterogeneous platforms, distributed web services and 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 and synchronization with web services are taken care of. Software process models should involve the cloud providers in every steps of decision making of software development life cycle to make the software project a success. First of all roles of software engineers and cloud providers are needed to be separated. As the cloud provider is an external entity or third party, interactions with them will be difficult. As a whole Cloud computing paradigm on Semantic Web background makes software development project more complex.

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 percentage of the infrastructure, planning and design phases of software development also have to include the cloud providers. The cloud providers can help in answering these questions about: 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.

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. Software engineers should train themselves in internet protocols, XML, web service standards and layered separation of concerns of SOA architecture of internet, Semantic Web technologies 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.

Complexity of SW development project will also 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.

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 higher priority tenants. The amount of interactions between software engineers and cloud providers will depend on type of cloud like public, private or hybrid cloud involvements (Figure 3). 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. 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.

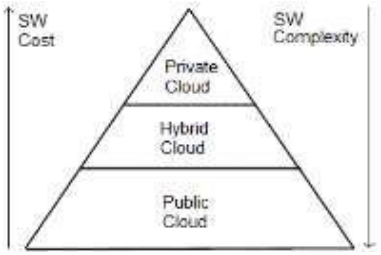


Fig. 3. Economics vs. Complexity of Software

5.3. Risk Management for Web Applications and Safety and Privacy Issues in Cloud Computing Platform

In comparison to desktop applications, web applications (T. DeMarco et al., 2003; T. Emilia Mendez et al., 2006; Siani Pearson, 2009; Wayne A. Jansen, 2011) need to be more reliable as they are supposed to be for multi-tenants and multi-user environment. Reliability of software applications developed by integrating COTS and Web Services should be taken care of. COTS are executable binary codes need to be trust worthy as they do not provide source code or not enough documentation are available for them. Also when resources are rented from cloud providers, misuse of data by cloud providers needs to be protected also. Web applications are also more vulnerable to security threats as they are exposed to the whole world over the internet. There can be attack from viruses, hackers, phishing like credit card fraud, malicious code injected by botnets which can cause performance lags or crashing of the system etc. Security requirements of web applications also need to be analyzed and incorporated from the early planning and design phase. Data encryption is a standard practice to secure transmission of sensitive data over the internet. Also any of such attack should be constantly monitored and in case of data loss immediate action should be taken to mitigate the problem. All these threats can be classified as external threats.

Also all the resources of the cloud computing platform are shared by multiple tenants over the internet across the globe. In this shared environment having trust of data safety and privacy are of utmost importance to customers. Safety of data means no loss of data pertaining to the owner of the data and privacy of data means no un-authorized use of the sensitive data by others. As cloud provider has greater resource pool they can easily keep copies of data and ensure safety of user data. Privacy of data is of more concern in public cloud than in private cloud. In public cloud environment as data is stored in off-premise machines users have less control over the use of their data and this mistrust can threaten the adoption of cloud computing platform by the masses. Technology and law enforcement both should protect privacy concerns of cloud customers. Software engineer must built their applications as web services which can guarantee to lessen this risk of exposure of sensitive data of cloud customers.

CONCLUSION

Traditional way of software engineering is no longer fully suitable in the changing scenario of modern hardware and software architecture of parallel and distributed computing on Semantic web and Cloud computing platform. These will make software engineering more difficult as software engineers have to change their programming style for parallel computing, master the Semantic Web skills for using open source software on distributed computing platform and they have to interact with a third party called the "cloud provider" in all stages of software processes. Automatic discovery and integration with Web Services will reduce the amount of work in terms of line of code (LOC) or function points (FP) required for developing software on cloud platform but there will be added semantic skill requirements and communication and coordination requirements with the cloud providers and web services which makes software development project more complex. Cloud computing being the anticipated future computing platform, more software engineering process models need to be researched which can mitigate all its challenges and reap all its benefits. Also safety and privacy issues of data in cloud computing platform need to be considered seriously so that cloud computing is accepted by all.

REFERENCES

Christian Bizer, Tom Heath, Tim Berners-Lee, Heath T. (2012), Linked Data- The Story So Far. Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS), Vol 5, No. 3, PP. 1-22.

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

Brand Niemann et al. (2005), Introducing Semantic Technologies and The Vision of The Semantic Web, SICOP White Paper, PP. 1-51.

Erin Cavanaugh, (2006), Web Services: Benefits, Challenges and a Unique Visual Development Solution, ALTOVA White Paper, PP. 1-21.

Duane Nickull et al. (2007), Service Oriented Architecture (SOA) and Specialized Messaging Patterns, White Paper, PP. 1-21.

http://www.itoamerica.com/media/pdf/adobe/soa_messaging_patterns.pdf.

Sun Microsystem, (2009), Introduction to Cloud Computing Architecture, White Paper, 1st Edition, PP. 1-40, www.sun.com/cloud/.

Sun Microsystem, (2012), Open Source and Cloud Computing: On-Demand Innovative IT on a Massive Scale, http://docs.huihoo.com/cloud-computing/sun-open-cloud-200906-en.pdf.

Radha Guha, David Al-Dabass, (2010), Impact of Web 2.0 and Cloud Computing Platform on Software Engineering, In Proceedings of 1st International Symposium on Electronic System Design (ISED), PP. 213-218.

Radha Guha (2013), Impact of Semantic Web and Cloud Computing Platform on Software Engineering, Book Chapter, Springer London Heidelberg New York Dordrecht, PP. 3-24.

M. Brambilla et al. (2006), A Software Engineering Approach to Design and Development of Semantic Web Service Applications, LNCS 4273, PP. 172-186.

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

Sommerville. (2006), Software Engineering, 8th Edition, Pearson Education.

Salesforce.com, (2009), Agile Development Meets Cloud Computing for Extraordinary Results, White Paper, PP. 1-8, www.salesforce.com.

T. DeMarco, T. Lister, (2003), Waltzing with Bears: Managing Risk on Software Projects, Dorset House Publishing Company, Incorporated.

T. Emilia Mendez, Nile Mosley (Eds). (2006), Web Engineering, Springer-Verlag Berlin Heidelberg.

Siani Pearson, (2009), Taking Account of Privacy When Designing Cloud Computing Services, Cloud 2009: ICSE Workshop on Software Engineering Challenges of Cloud Computing, Vancouver, IEEE, PP. 44-52.

Wayne A. Jansen, (2011), Cloud Hooks: Security and Privacy Issues in Cloud Computing, Proceedings of the 44th Hawaii International Conference on System Sciences, PP. 1-10.