SOFTWARE ENGINEERING ON SEMANTIC WEB AND CLOUD COMPUTING PLATFORM

Radha Guha, Ph.D, UCI
radhaguha@yahoo.com
Anaheim, CA 92807

Abstract: Tim Berners Lee's vision of the Semantic Web or Web 3.0 is to 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 grand vision of the web is materializing many innovative use of the web. New business models like interoperable applications hosted on the web as services are getting implemented. These web services are designed to be automatically discovered by software agents and exchange data amongst themselves. Another business model is the cloud computing platform, where hardware, software, tools and applications all will be leased out to tenants across the globe over the internet. The advancement of the Semantic Web and the many advantages of the new business models are changing the way software needs to be developed. This makes one wonder how software engineering has to adapt to these anticipated futuristic trends and how much it will benefit from them. This paper analyses how cloud computing on the background of Semantic Web is going to impact software engineering processes to develop quality software. As the cloud provider is an external entity or third party, how difficult will be the interactions with them? How to separate the roles of software engineers and cloud providers? 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. This paper also explores how Semantic Web will facilitate software development with automatic discovery of distributed open source software components. Also Semantic Web techniques are explored that needs to be incorporated in software development artifacts to make them Semantic Web ready.

1. Introduction

Since the inception of the World Wide Web in 1990 by Tim Berners Lee, it has been a large warehouse of documents and the number of documents is growing very rapidly. But unless the information from these documents can be aggregated and inferred quickly, they don't have much use. Human readers cannot read large number of documents (mostly irrelevant) retrieved by the old search engines based on keyword searches and make decisions quickly. Thus Tim Berners Lee's vision is to transform this World Wide Web into an intelligent web system or Semantic Web [1], [2], [3], [4], [5], [6], [7], [8] which will allow concept searches rather than keyword searches. First, Semantic Web or Web 3.0 technologies will transform disconnected text documents on the web into a global database of structured, linked data. These large volumes of linked data in global database will no longer be only for human consumption but for quick machine processing. Just like a relational database system can answer a query by filtering out unnecessary data, Semantic Web technologies will similarly filter out information from the global database. This capability requires assigning globally accepted explicitly defined semantics to the linked data. Then these linked data in the global database will collectively produce intelligent information by software agents on behalf of the users and the full potential of the web can be exploited.

Anticipating this transition of the web where data integration, inference and data exchange between heterogeneous applications will be possible, 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 XML (eXtensible Markup Language) based, interoperable applications are called Web Services which can publish their functions and messages (which contain the parameter list to execute the functions) for getting discovered on the web and used 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. Other 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. Applications like Hadoop and Mashup [9], [10] can combine data and functionalities from multiple external sources hosted as Web Services, are producing valuable aggregate new information and are creating new Web Services. Hadoop and Mashup can support high performance computing involving distributed file system with petabytes of data and parallel processing on more than hundreds to thousands of computers.

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.

On the other hand, economies of all developed countries depend on quality software and software cost is more than hardware cost. Moreover 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 for better software development and for 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 continuous changing requirements of the software. This is more so when the software has to be maintained and no one has the comprehensive knowledge about the whole system.

Now with the inclusion of the Free/Libre/Open Source Software (FLOSS) pieces and cloud computing platform, software development complexity is going to increase manifold because of the synchronization needs with third party software and the increased communication and coordination complexity with the cloud providers. The main thesis of this paper is that the prevalent software process models should involve the cloud providers in every steps of decision making in software development life cycle to make the software project a success. Also the software developers need to change their software artifacts from plain text documents to machine readable structured linked data, to make them Semantic Web ready. With this semantic transformation knowledge management in software engineering will be much easier and it will also give their product a competitive edge for automatic discovery and integration with other applications and efficient maintenance of their artifacts.

In Section II, background literatures on transformation to Semantic Web, cloud computing platform and software engineering are surveyed. In Section III, first emphasis is given on the need for producing software artifacts for the Semantic Web. Secondly how the software developers are coping with the changing trend of application development on cloud platform with Web 2.0 and Web 3.0 protocols and application deployment over the web is reported. Thirdly, challenges of cloud computing platform for software engineering are analyzed. In Section IV, an agile process model which incorporates interaction with cloud provider is proposed and analyzed. Section V concludes the paper.

II. Literature Survey

1. *Transformation to Semantic Web*

World Wide Web was invented in 1990 by Tim Barners Lee. Since then the transformation of the web has been marked with Web 1.0, Web 2.0 and Web 3.0 technologies. In Web 1.0, the HTML (hypertext markup language) tags were added to plain text documents for displaying the documents in a specific way on web browsers. Each document on the web is a source of knowledge or a resource. In the World Wide Web, with the hypertext transport protocol (HTTP), if the URI (Universal Resource Identifier) of any web site (document) is known then that resource can be accessed or browsed over the internet. URI is an expansion on the concept of Universal Resource Locator or URL, which is the commonly known identifier for websites and webpages. URI can be name and location both. This capability of Web 1.0 published information pages which ware static and read only. HTML's <href> tag (a form of metadata) links two documents for human readers to navigate to related topics. In Web 1.0, for quick search and retrieval metadata, (data about data) that describes the contents of electronic documents or resources are added in the document itself, which has the same purpose as indexes in a book or catalogues in a library. Search engines like Google and Yahoo create metadata databases out of those metadata in web documents to find the documents quickly.

Then in Web 2.0, XML (eXtensible Markup Language) was designed to give hierarchical structure to the document content, transform it into data and to transport the document as data. Where HTML tags, prescribes how to display the web content in client computer, the XML tags adds another layer of metadata to query the web document for specific data. XML documents can be read and processed by computers (by a parser) automatically and can be exchanged between applications developed on heterogeneous computing platforms, operating systems and varieties of programming languages once they all know the XML tags used in the documents. As for example in order to use text generated by a Word Processor and data from spreadsheets and relational databases together they all need to be transformed into a common XML format first. This collaboration of applications is possible in a closed community when all the applications are aware of the common XML tags. Web 2.0 technologies also enabled pervasive or ubiquitous web browsing involving personal computers, mobile phones and PDA running different operating systems like Windows, Macintosh or Linux, connected to the Internet via wired or wireless connections. Web 2.0 technologies like XML, DHTML, AJAX (Asynchronous Java Script and XML) allowed two way communications with dynamic web contents and created social communities like Facebook, MySpace, Twitter etc.

But for collaboration in the open, ever-expanding World Wide Web by all, everybody on the web has to agree on the meaning of the web contents. XML alone does not add semantics to the web content. Thus in Web 3.0, Resource Description Framework (RDF) protocol is designed to add another layer of metadata to add meaning or semantics to the data (text, images, audio or video) inside the document with RDF vocabularies understood by machines. As computer memory is not expensive any more this metadata can be verbose even for human understanding instead of being only for machine understanding. Authors, publishers and users all can add metadata about a web resource in a standardized format. This self-describing data inside the document can be individually addressed by HTTP URI mechanism, processed and linked to other data from other documents and inferred by machine automatically. Search engines or crawlers will navigate the links and generate query response over the aggregated linked data. This linked data will encourage reuse of information, reduce redundancy and produce more powerful aggregate information.

But first of all we need a standardized knowledge representation system. Modeling a knowledge domain using standard, shared vocabularies will facilitate interoperability between different applications. Ontology is a formal representation of knowledge as a set of concepts in a domain. Ontology components are classes, their attributes, relations, restrictions, rules and axioms. DublinCore, GFO, OpenCyc/SUMO, DOLCE, WordNet, FOAF, SIOC, SKOS, DOAP, vCard etc. are the much used well known ontology libraries of RDF vocabularies. Implementation of DublinCore makes use of XML and a Resource Description Framework (RDF).

RDF triples describes any data in the form of subject, predicate and object. Subject, predicate and object all are URIs which can be individually addressed in the web by the HTTP URI mechanism. Subject and object can be URIs from the same document or from two separate documents or independent data sources linked by the predicate URI. Object can also be just a string literal or a value. RDF creates a graph based data model spanning the entire web which can be navigated or crawled following the links by software agents. RDF schema (RDFS), web ontology language (OWL) and simple knowledge organization system (SKOS) are developed to write rules, express hierarchical relations, inference between web resources. They vary in their expressiveness, logical thinking and hierarchical knowledge organization from being more limited to more powerful in RDFS to SKOS. For querying the RDF data written in RDFS, OWL or SKOS, RDF query language named SPARQL has been developed.

RDF tags can be added automatically or semi-automatically by tools [7] like RDFizers, D2R (Database to RDF), JPEG->RDF, Email->RDF etc. Linked data browsers like Disco, Tabulator, Marbles are getting designed to browse linked data Semantic Web. Linked data search engines like Falcon, SWSE are getting designed for human navigation and Swoogle, Sindice are getting designed for applications.

Figure 1 [8], shows the Semantic Web protocol stacks (Wedding Cake) proposed by Tim Barners-Lee in 2000. The bottom of the Wedding Cake shows standards that are well defined and widely accepted whereas the other protocols are yet to be implemented in most of the web sites. Unicode is a 16-bit code word which is large enough (2^{16}) for representing any characters in any languages in the world. URI (Universal Resource Identifier) is the W3C’s codification for addressing any objects over the web. XML is for structuring the documents into data and RDF is the mechanism for describing data which can be understood by machines. Ontologies are vocabularies from specific knowledge domain. Logic refers to making logical inferences from associated linked data. Proof is keeping track of the steps of logical inferences. Trust refers to the origin and quality of the data sources. This entire protocol stack will transform the web into a Semantic Web global database of linked data for exploiting the full potential of the web.

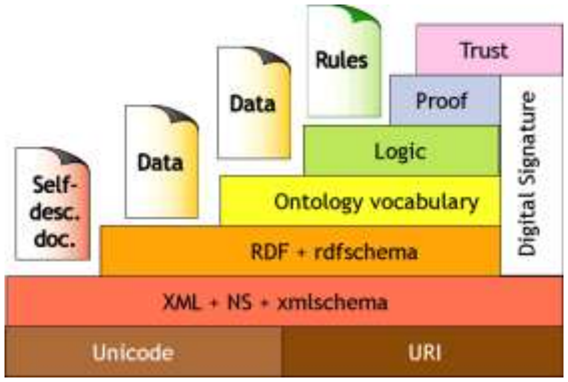


Figure 1: Semantic Web Wedding Cake [8]

2. Cloud Computing Platform

Cloud computing [11], [12], [13] is the future trend of computing. Cloud computing is the idea of renting out 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 four 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).

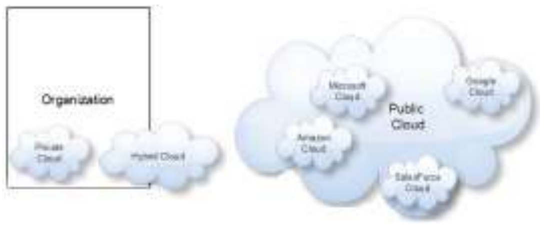


Figure 2: Cloud Computing Platform

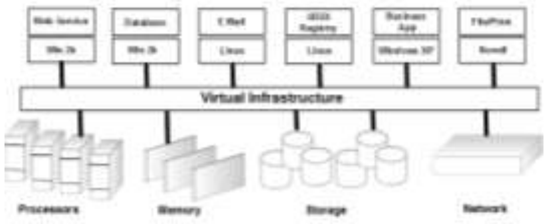


Figure 3: Virtual Infrastructure [13]

Depending on the amount of self-governance or control on resources by the tenant there are three types of cloud like internal or private cloud, external or public cloud and hybrid cloud (Figure 2). 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, Web 3.0 and virtualization [13], [14], [15] of hardware and software resources (Figure 3). 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, 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. 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 2]. 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.

Next we delve into pre-existing software development methodologies to develop quality software products in traditional environment not involving Web Services and cloud computing platform.

3. Software Engineering Process

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 software 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 software project is measured by whether it is developed within time and budget and by its efficiency, usability, dependability and maintainability [16], [17].

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. 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 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 the 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. 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.

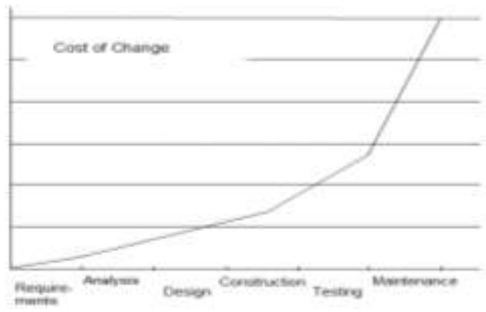


Figure 4: Economics of Software Development

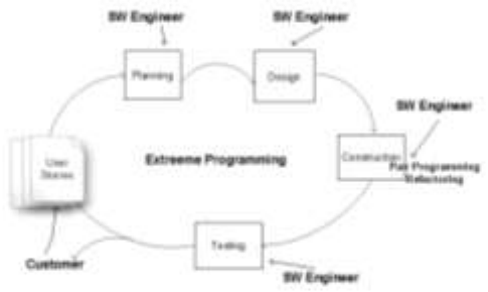


Figure 5: Extreme Programming Process Model

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 4). 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 5), 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.

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 key to the success of a project.

Figure 5 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.

In the next sections we analyze the need for 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. Finally we suggest methodologies to develop quality software that will push forward the advances of the cloud computing platform.

III. Analysis

1. *Need for Semantic Web Enabled Software Artifacts*

Semantic Web effort has just started and not all are aware of it even the IT professionals. The linked data initiative [7] that was taken in 2007 by small groups of academic researchers from universities now have participants of few large companies like BBC, Thompson Reuters and Library of congress who have transformed their data for the Semantic Web. DBpedia is another community effort to transform the Wikipedia documents for Semantic Web. Sophisticated queries can be run on DBpedia data and link to other Semantic Web data. Friend of a Friend (FOAF) is another project to link social websites and their people and describe what they create or do. Federal and State governments are also taking initiatives to publish public data online. US Census data is one such semantic data source which can be queried and linked with other semantic data sources. Unless all government public data can be transformed for the Semantic Web they will not be suitable for interoperable web applications.

Figure 6 [7], shows the current size of the linked data web as of March 2009. Today there are 4.7 billion RDF triples which are interlinked by 142 million RDF links. Anybody can transform their data in linked data standards and can link to the existing linked data web. In Figure 6, the circles are nodes of independent data sources or web sites and the arcs are their relationship with other data sources. The thicker links specify more connections between the two data sources and bi-directional links means both data sources are linked to each other.

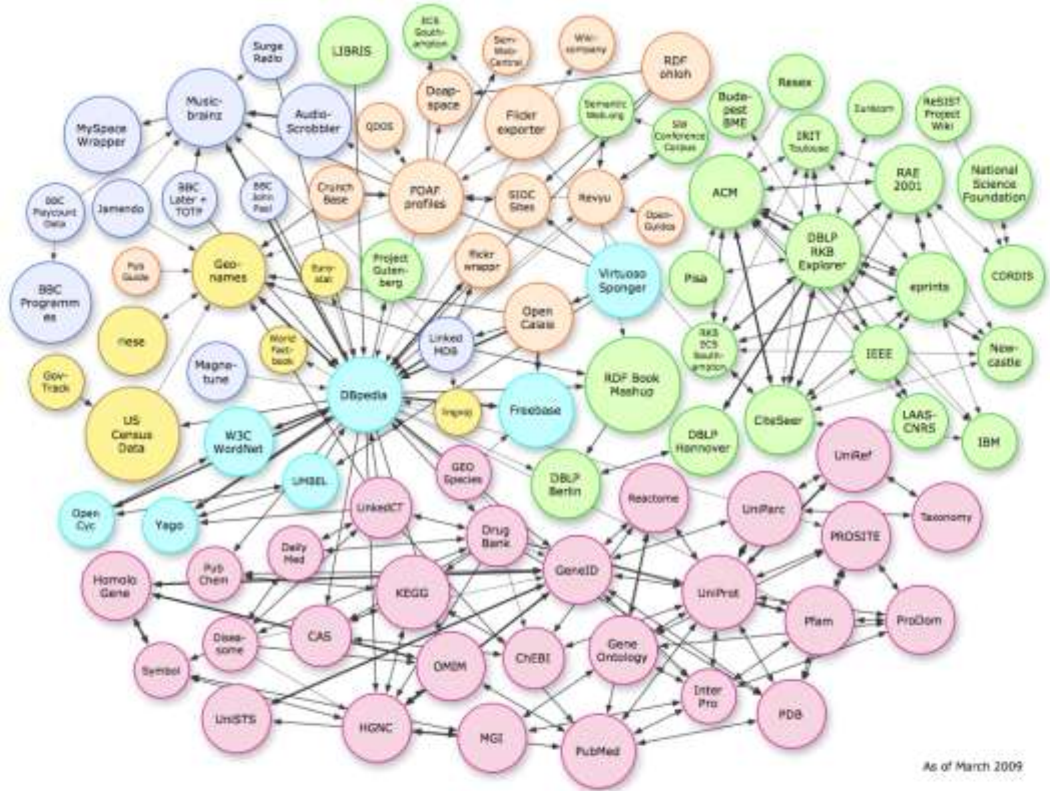


Figure 6. Linking Open Data cloud diagram giving an overview of published data sets and their inter-linkage relationships [7].

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. Once the software artifacts are transformed into semantic artifacts software maintainability will be much more efficient and cheaper. Increased maintainability of software will also increase reliability of the software. Semantic Web services will be easy to discover on the web and that will give a competitive edge to their products. Semantic Web services which can be linked with other web services will create new and more powerful software applications, encourage reuse and reduce redundancy.

2. *How Software Development Industry is Surviving in the Cloud Computing Age?*

This section surveys how software development industry is trying to survive in the era of Web 2.0 and Web 3.0 with Web Services and cloud computing. In reference [18], 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 [19], Salesforce.com finds that agile process model works better on cloud computing platform. Before cloud computing, release of the software 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 software 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 software development process and suggest extending the existing agile process model, named extreme programming to mitigate all the challenges in Section V below.

3. *Impact of Cloud Computing on Software Engineering: Challenges*

In the rapidly changing computing environment with web services and cloud platform, software development is going to be very challenging. Software 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 software 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 software 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 software engineers should train themselves in parallel and distributed computing to complement this advances of hardware and network technology. 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 [13].

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.

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 or 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 7]. 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.

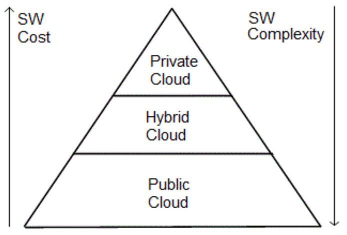


Figure 7: Economics vs. Complexity of Software Development on Cloud Computing Platform

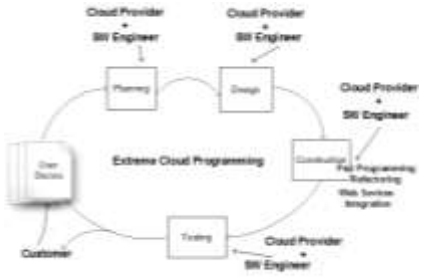

Cloud Provider + SW Engineer
Cloud Provider + SW Engineer
Planning
Design
Cloud Provider + SW Engineer
Extreme Cloud Programming
Construction
Pair Programming Refactoring
Web Services Integration
Testing
Cloud Provider + SW Engineer
Customer

Figure 8: Extreme Cloud Programming

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 8].

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.

Table 1: Software Engineering- Role Separation

Activity	**Roles**	
	Software 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

Table 2: COCOMO

Software 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**

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 [20] 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 software development projects. In COCOMO model [16] three classes of software projects have been considered so far. These software projects are classified as 1) Organic, 2) Semi-detached, 3) Embedded according to the software team size, their experiences and development (HW, SW and operations) constraints. We extend [21] this cost estimation model with a new class of software 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]

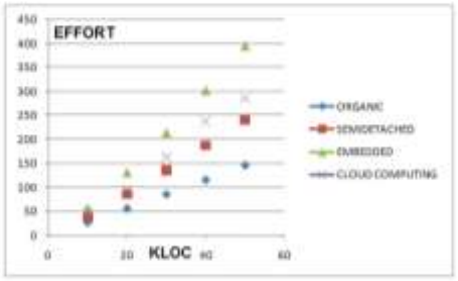
Development Time = $c(\text{Effort Applied})^d$ [months]

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

The typical values of the coefficients a, b, c, d for different classes of software projects are listed in Table 2.

In anticipation of additional 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 three classes but is linear like the other three classes. Similarly coefficients c, d for cloud computing are determined so that the development time curve is less steeper than the other three classes but is linear like the other three 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, software 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.



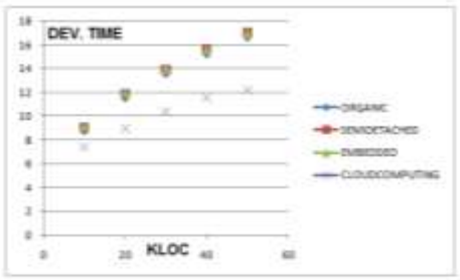


Figure 9: Extended COCOMO For SW Effort Estimation **Figure 10: Extended COCOMO For SW Dev. Time**

Figure 9 plots software effort estimation for project size varying from 10 to 50 KLOC for all four 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 10 plots the corresponding software development time estimation for all four classes of software projects. With 30% component reuse possibility, software development on cloud computing platform will take least amount of time.

V. Conclusion

Web Services and Cloud computing are paradigm shifts 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". Automatic discovery and integration with Web Services will reduce the amount of work required for developing software on cloud platform but there will be added semantic skill requirements and communication and coordination requirements with the cloud providers which makes software development project more complex. The main thesis of this paper is that the prevalent software process models should incorporate this new dimension of interactions with the cloud providers and separate roles of software engineers and cloud providers. A new agile process model is proposed in this paper which includes the anticipated interactions 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. Secondly the Semantic Web techniques are explored what the software developers needs to incorporate in their artifacts in order to be discovered easily on the web to give their product a competitive edge, and for efficient software integration and maintenance purposes.

REFERENCES

[1] Tim Barners-Lee. Future Of The Web. http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.

[2] 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.

[3] 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.

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

[5] 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.

[6] Linked Data. Tim Barners-Lee. http://www.w3.org/DesignIssues/LinkedData.html.

[7] Heath, T., Hepp, M., and Bizer, C. (eds.). Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS). http://linkeddata.org/docs/ijswis-special-issue

[8] Brand Niemann et al. Introducing Semantic Technologies and the Vision of the Semantic Web. SICoP White Paper. Feb 2005.

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

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

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

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

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

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

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

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

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

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

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

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

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