# Chapter 1
# Impact of Semantic Web and Cloud Computing Platform on Software Engineering

**Radha Guha**

**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 uses 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 among themselves. Another business model is the cloud computing platform, where hardware, software, tools, and applications will be leased out as services to tenants across the globe over the Internet. There are many advantages 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 tenants. Because of these advantages, cloud computing may be the prevalent computing platform of the future. To realize all the advantages of these new business models of distributed, shared, and self-provisioning environment of Web services and cloud computing resources, the traditional way of software engineering has to change as well. This chapter analyzes how cloud computing, on the background of Semantic Web, is going to impact on the software engineering processes to develop quality software. The need for changes in the software development and deployment framework activities is also analyzed to facilitate adoption of cloud computing platform.

**Keywords** Software engineering • Semantic Web • Cloud computing platform • Agile process model • Extreme Cloud Programming

R. Guha (✉)
ECE Department, PESIT, Feet Ring Road, BSK III Stage,
560085, Bangalore, India
e-mail: radhaguha@pes.edu

Z. Mahmood and S. Saeed (eds.), *Software Engineering Frameworks for the Cloud Computing Paradigm*, Computer Communications and Networks,
DOI 10.1007/978-1-4471-5031-2_1, © Springer-Verlag London 2013

## 1.1 Introduction

Since the inception of the World Wide Web (WWW) in 1990 by Tim Berners-Lee, there has been a large warehouse of documents on the WWW, and the number of documents is growing very rapidly. But, unless the information from these documents can be aggregated and inferred quickly, they do not have much use. Human readers cannot read and make decisions quickly from large number of mostly irrelevant documents retrieved by the old search engines based on keyword searches. Thus, Tim Berners-Lee's vision is to transform this World Wide Web into an intelligent Web system or Semantic Web [1–8] 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 databases 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 data in the Web for linking. Then these linked data in the global database will collectively produce intelligent information by software agents on behalf of the human users, and the full potential of the Web can be realized.

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 location, functions, messages containing the parameter list to execute the functions, and communication protocols for accessing the service using it 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. Applications like Hadoop and Mashup [9, 10] can combine data and functionalities from multiple external sources hosted as Web services and 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 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 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 the 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.

Now, with the inclusion of the Free/Libre/Open Source Software (FLOSS) [11] pieces, Web services, 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 chapter is that the prevalent software process models should involve the cloud providers in every step of decision-making of 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 compliance checking of various requirements during project planning, design, development, testing, and verification can be automated. Semantic artifacts will also give their product a competitive edge for automatic discovery and integration with other applications and efficient maintenance of their artifacts.

This chapter explores how Semantic Web can reduce software development work with automatic discovery of distributed open source software components. Also, Semantic Web techniques are explored that need to be incorporated in software development artifacts to make them Semantic Web ready. Then, to realize the many advantages of the cloud computing business model, how the well-established software engineering process models have to adapt is analyzed. 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? As a whole, cloud computing paradigm on Semantic Web background makes software development project more complex.

In Sect. 1.2, background literatures on transformation to Semantic Web, cloud computing platform, and software engineering are surveyed. In Sect. 1.3, 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 Sect. 1.4, an agile process model which incorporates interaction with cloud provider is proposed and analyzed. Section 1.5 concludes the chapter.

## 1.2 Literature Survey

### *1.2.1 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 URL (Universal Resource Locator) of any Web site (document) is known, then that resource can be accessed or browsed over the Internet. Domain name service (DNS) registry was developed to discover a machine on the Internet which hosts a Web page URL. This capability of Web 1.0 published information pages which were 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. In Web 1.0, the contents of the Web pages are static and the meanings of the Web pages are deciphered by the people who read them. Web contents are developed by HTML and user input is captured in Web forms in the client machine and sent to remote server via a common gateway interface (CGI) for further processing.

In Web 2.0, XML (eXtensible Markup Language) was designed to give hierarchical structure to the document content, to transform it into data, and to transport the document as data. Where HTML tags prescribe how to display the Web content in client computer, the XML tags add 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 (Personal Digital Assistant) 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, and AJAX (Asynchronous Java Script and XML) allowed two-way communications with dynamic Web contents and created social communities like Facebook, MySpace, and Twitter. Web 2.0 has also seen the revolution of using the Web as the practical medium for conducting businesses. An increasing number of Web-enabled e-commerce applications like e-Bay and Amazon have emerged in this trend to buy and sell products online.

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 anymore, 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 (Universal Resource Identifier) mechanism, processed and linked to other data from other documents, and inferred by machine automatically. URI is an expansion on the concept of Universal Resource Locator or URL and can both be a name and location. 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.

To this end, we need a standardized knowledge representation system [12, 13]. 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 (General Formal Ontology), OpenCyc/SUMO (Suggested Upper Merged Ontology), DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), WordNet, FOAF (Friend of a Friend), SIOC (Semantically Interlinked Online Communities), SKOS (Simple Knowledge Organization System), DOAP (Description of a Project), vCard, etc., are the much used well-known ontology libraries of RDF vocabularies. For example, implementation of DublinCore makes use of XML and a Resource Description Framework (RDF).

RDF triples describe any data in the form of subject, predicate, and object. Subject, predicate, and object all are URIs which can be individually addressed in

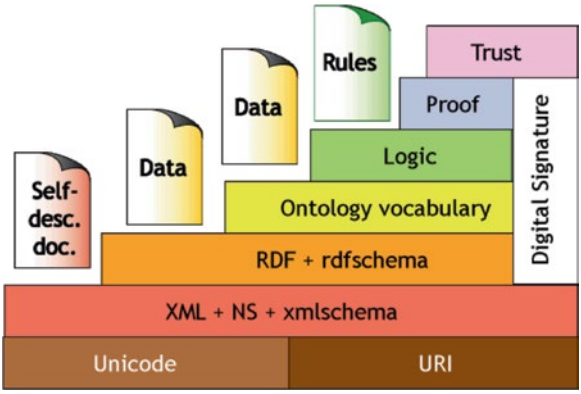


**Fig. 1.1** Semantic Web Wedding Cake [8]

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 and 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 semiautomatically by tools like RDFizers [7], D2R (Database to RDF), JPEG → RDF, and Email → RDF. Linked data browsers like Disco, Tabulator, and Marbles are getting designed to browse linked data Semantic Web. Linked data search engines like Falcon and SWSE (Semantic Web search engine) are getting designed for human navigation, and Swoogle and Sindice are getting designed for applications.

Figure 1.1 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 realizing the full potential of the Web.

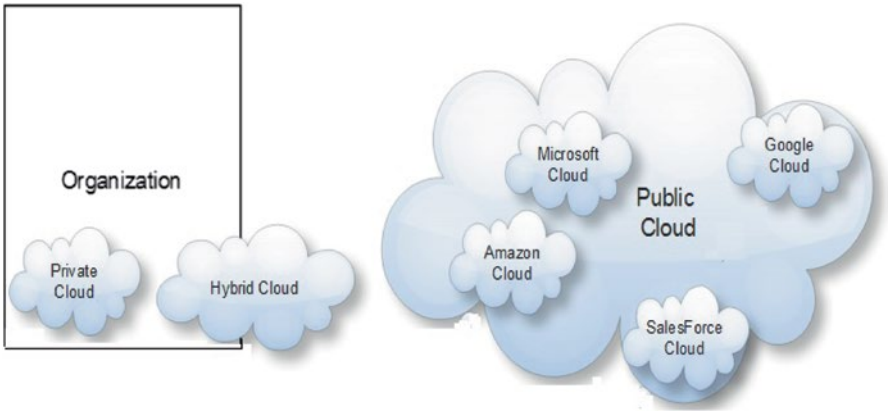


**Fig. 1.2** Cloud computing platform

### *1.2.2 Cloud Computing Platform*

Cloud computing [14–16] is the most anticipated 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).

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 (Fig. 1.2). 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 and Web 3.0 and virtualization [16–18] of hardware and software resources (Fig. 1.3). 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 among all users, and there is efficient resource management which can provide higher resource utilization and on-demand scalability. Increased resource utilization brings down

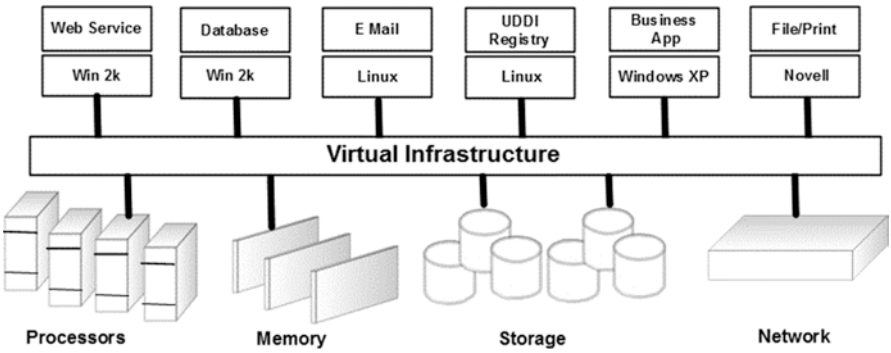


**Fig. 1.3** Virtual infrastructure [13]

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 user.

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 provides 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 (Fig. 1.2). Web services and applications like Hadoop and Mashup can run on these clouds. Though there are many advantages of cloud computing platform, there are few challenges regarding safety and privacy of tenant's information in cloud platform which can threaten the adoption of cloud computing platform by the masses. If these few challenges can be overcome, because of many of its advantages, this cloud computing model may be the prevalent computing model of the future.

#### 1.2.2.1 Safety and Privacy Issues in Cloud Computing Platform

All the resources of the cloud computing platform are shared by multiple tenants (Fig. 1.4) over the Internet across the globe. In this shared environment, having trust of data safety and privacy is 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

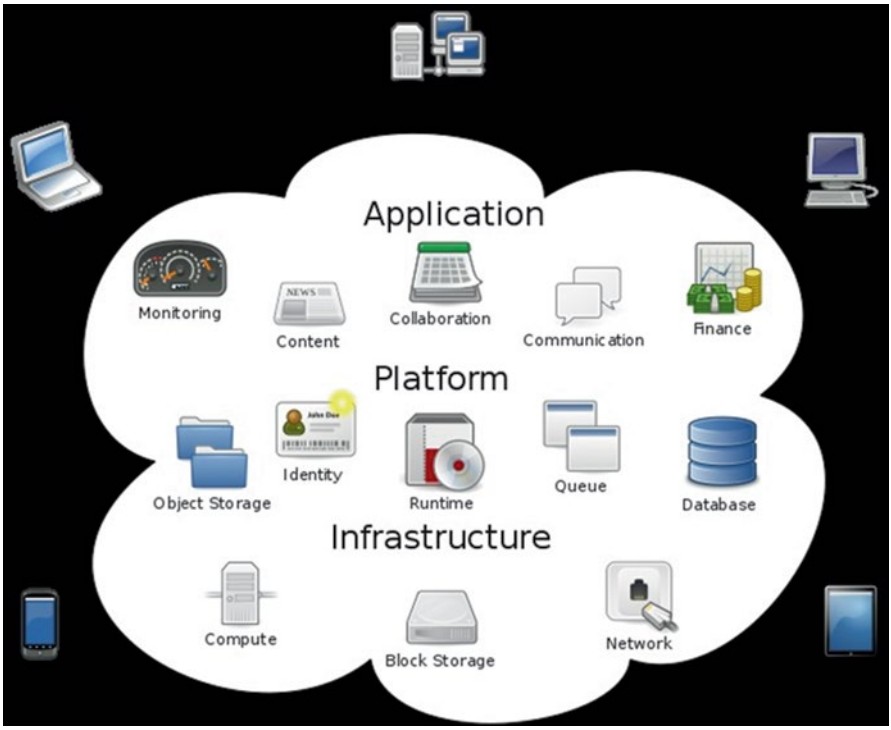


**Fig. 1.4** Shared resources in cloud computing

no unauthorized 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 [19, 20]. Software engineer must build their applications as Web services which can guarantee to lessen this risk of exposure of sensitive data of cloud customers.

Next, we look into the preexisting software development methodologies to develop quality software products in traditional environment not involving Web services and cloud computing platform.

### *1.2.3 Traditional Software Engineering Process*

Here, we delve into preexisting software development methodologies first to develop quality software products in traditional environment not involving Web services and cloud computing platform. Over the last half-century, rapid advances

of hardware technology such as computers, memory, storage, communication networks, mobile devices, and embedded systems are 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, stakeholders, end users, and software developers. That is 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 [21, 22].

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 requirements 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, and type of the project. As an 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 requirements 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. Oftentimes volume of paper documents is too large for aggregating information by humans. More than 50 % of software projects fail due to various

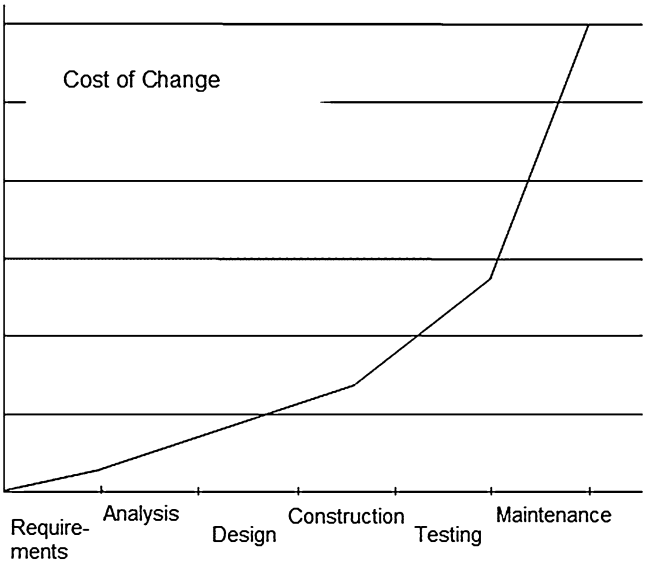


**Fig. 1.5** Economics of software development

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.

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 the cost of the project exponentially (Fig. 1.5). 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 are done involving all the parties from the beginning.

That is the reason why several agile process models like Extreme Programming (XP) (Fig. 1.6), Scrum, Crystal, and Adaptive 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 a 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.

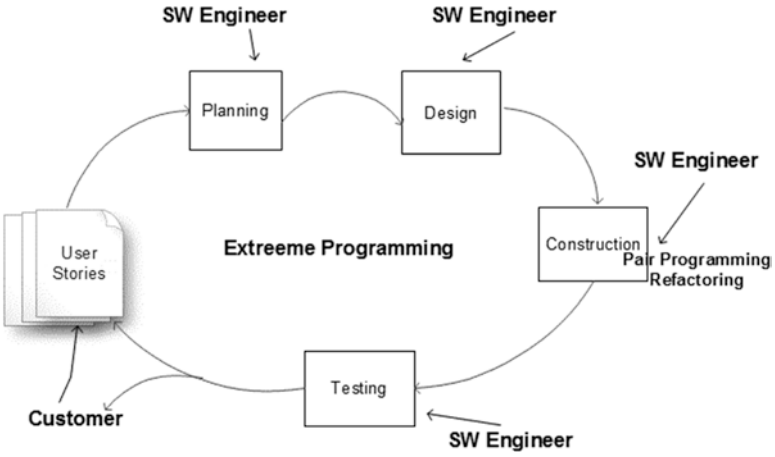


**Fig. 1.6** Extreme Programming process model

Figure 1.6 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 and CRC (class, responsibility, collaboration) card narrated during the requirements 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 refactored or improved to make it more efficient.

In the following sections, analysis for 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 and Web 3.0 technologies and cloud computing platform are reported. Finally, methodologies to develop quality software that will push forward the advances of the cloud computing platform have been suggested.

## 1.3 Need for Modification of Software Engineering: Analysis

### *1.3.1 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 a small group of academic researchers from universities now has participants of few large companies like BBC, Thompson Reuters, and Library of Congress who have transformed their

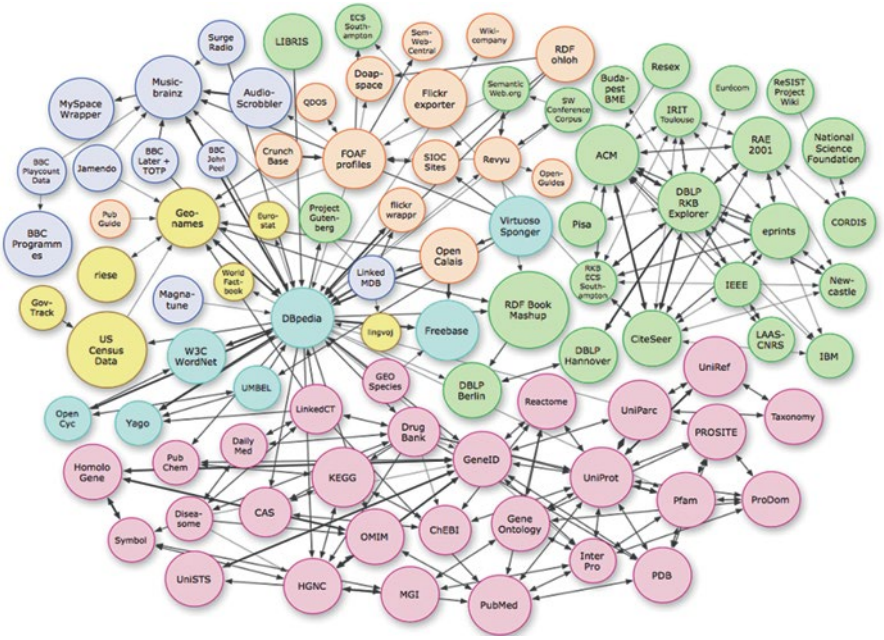


**Fig. 1.7** Linking open data cloud diagram giving an overview of published data sets and their interlinkage relationships [7]

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 Web sites 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 1.7 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 Fig. 1.7, 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 bidirectional links mean both data sources are linked to each other.

Once the software engineers grasp the Semantic Web technologies and 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

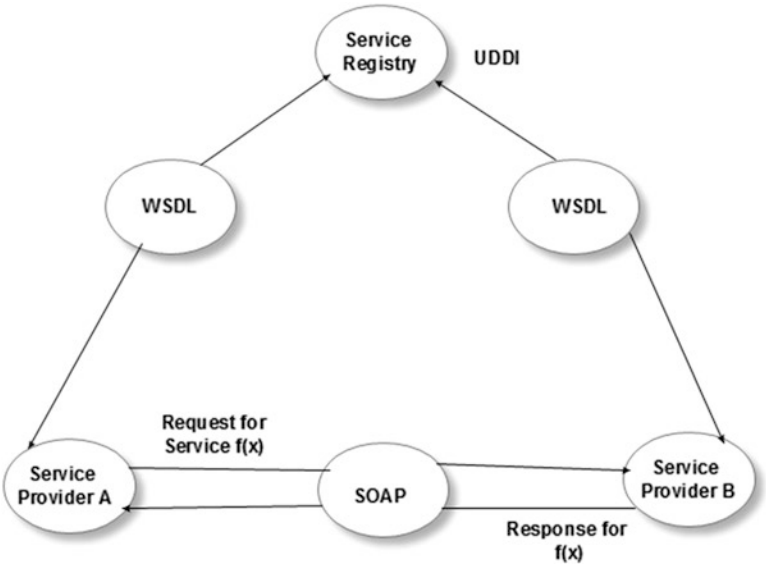


**Fig. 1.8** Service-Oriented Architecture for interoperability of services

much more efficient and cheaper. 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 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.

### *1.3.2 Creating a Web Service*

Benefits of Web services [23–26] are code reuse and speedy development of software projects. But in order to use Web services from the Web, the application must create a Web client which can interface with the Web services and request for services and receive services. In Fig. 1.8, the Service-Oriented Architecture (SOA) that has emerged to deliver software as a service (SaaS) business model is illustrated.

An application programming interface (API) of Web service is first created as WSDL document using XML tags, for advertising to the world over the Internet. WSDL documents have five major parts. It describes data types, messages, port, operation (class and methods), binding (SOAP message), and location (URL). WSDL documents need not be manually created. There are automatic tools like Apache Axis [25], which will create the API from a Java programming code. Apache Axis is an open source, XML-based Web service framework.

After creating the WSDL document, a Web client to consume the Web service is needed. Web client is created using SOAP to communicate request and response messages between the two applications. SOAP is an XML messaging format for exchanging structured data (XML documents) over HTTP transport protocol and can be used for remote procedure call (RPC). SOAP structure has three parts: (1) envelop, (2) header, and (3) body. Body defines the message and how to process it.

Software engineers have to master XML language and other Web technologies like WSDL and SOAP in addition to knowing a programming language like Java or C++ in order to use or create a Web service.

### *1.3.3 How SW Engineers Are Coping in Cloud Platform*

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 [27], the authors 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 [28], 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, new releases of the software can be uploaded on the server and used by the users immediately. Basically in this chapter, 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 Sect. 1.3.4.

### *1.3.4 Impact of Cloud Computing on Software Engineering*

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, 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.

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.

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 manyfold 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.

However, software developers need 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 the latest hardware technology, the computers are multi-core and networked, and the software engineers should train themselves in parallel and distributed computing to complement these 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, and Semantic Web technologies to leverage all the benefits of Web 2.0. Cloud providers will insist 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 [16].

Maintenance phase should also 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 a contract with them as well so that the "Software Engineering code of ethics" is 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

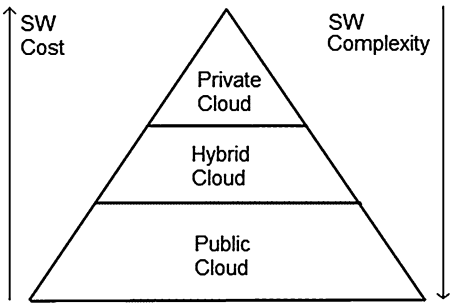


**Fig. 1.9** Economics vs. complexity of software

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 resource 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 the 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, and availability of data (Fig. 1.9). 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.

## 1.4 Proposed SW Process Model for Cloud Platform

Innovative software engineering is required to leverage all the benefits of cloud computing and mitigate its challenges strategically to push forward its advances. Here an extended version of Extreme Programming (XP), an agile process model for cloud computing platform named Extreme Cloud Programming (Fig. 1.10), is proposed. All the phases like requirements gathering, 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 1.1. Resource accounting on cloud platform will be done by the cloud

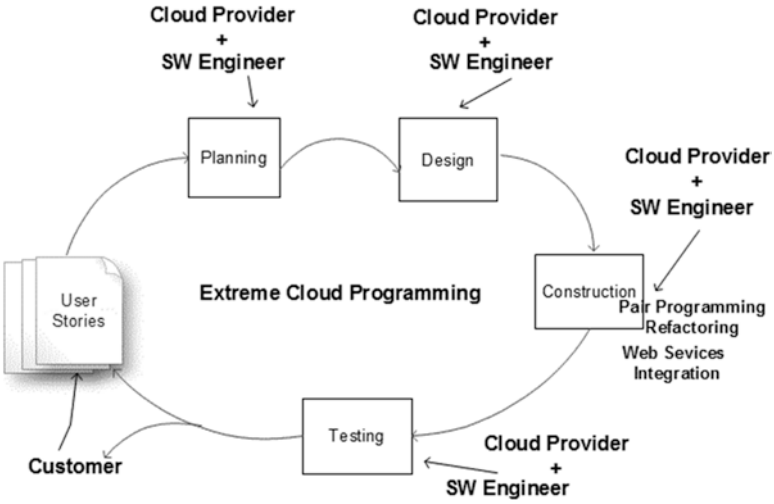


**Fig. 1.10** Extreme Cloud Programming development on cloud computing [29]

**Table 1.1** Software engineering-role separation [29]

| | Roles | |
|---|---|---|
| Activity | Software developer | Cloud provider |
| Requirements gathering | Elicitation | Resource accounting |
| | | Virtual machine |
| Analysis | SW modules | SW/HW architecture |
| Design | Interface design | Component reuse |
| | Data types | |
| | Cost estimation | |
| | Schedule estimation | |
| Construction | Coding | Implementation details |
| | Integration of Web services | |
| Testing | Unit test | Integration test |
| | Integration test | |
| Deployment | | Operation and maintenance |

provider in the requirements 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.

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 [30] about

**Table 1.2** COCOMO [29]

| Software proj. | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| Organic | 2.4 | 1.05 | 2.5 | .38 |
| Semidetached | 3.0 | 1.12 | 2.5 | .35 |
| Embedded | 3.6 | 1.2 | 2.5 | .32 |
| Cloud comp. | 4 | 1.2 | 2.5 | .3 |

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 (Table 1.2), three classes of software projects have been considered so far. These software projects are classified as (1) Organic, (2) Semidetached, (3) Embedded according to the software team size, their experiences, and development (HW, SW, and operations) constraints. We extend [29] 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 number of people required are given by the following equations.

$$\text{Effort Applied} = a(\text{KLOC})^b \left[\text{man} - \text{months}\right]$$

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

$$\text{No. of People} = \text{Effort Applied} / \text{Development Time}\left[\text{no.}\right]$$

The typical values of the coefficients *a*, *b*, *c*, *d* for different classes of software projects are listed in Table 1.2. 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. The coefficients *a*, *b*, *c*, *d* in cloud computing are readjusted 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 $\text{KLOC} = i * C + (\text{KLOC}) * C$, where *C* is the % of component reuse and *i* is the coefficient adjustment for new interface design effort.

Figure 1.11 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 percentage of component reuse is possible, it will mitigate the higher interaction complexity in coefficient *a* and will be beneficial for cloud computing platform. Figure 1.12 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.

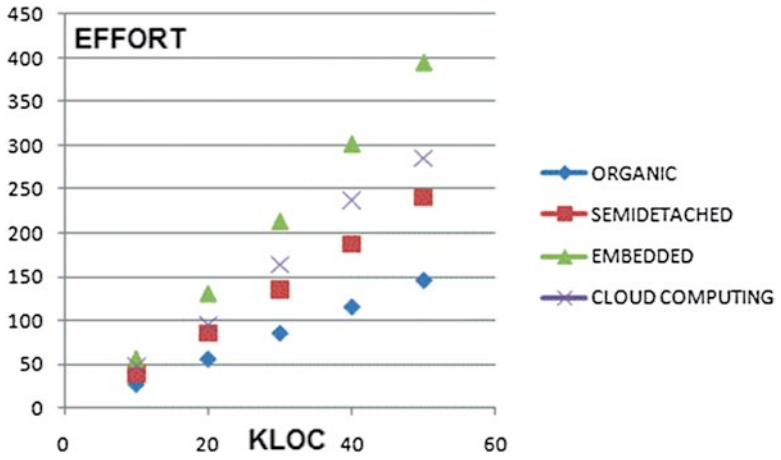


**Fig. 1.11** Extended COCOMO for SW effort estimation [29]

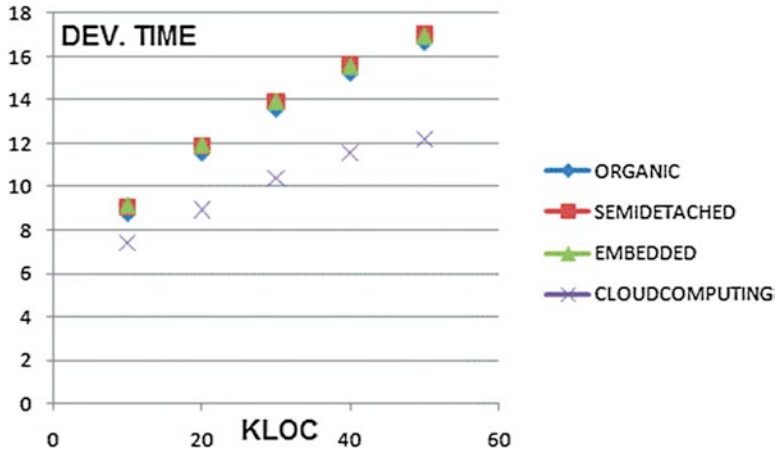


**Fig. 1.12** Extended COCOMO for SW dev. time [29]

## 1.5 Conclusion

The development of Semantic Web or Web 3.0 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 cloud computing platform. 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 software engineers have to 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 which makes software development project more complex.

First, the Semantic Web techniques are explored on what the software developers need 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. Then, the need for changes in the prevalent software process models is analyzed to suggest that they should incorporate the 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 chapter 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.

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 truly accepted by all.

## References

1. Barners-Lee, T.: Future of the web. http://dig.csail.mit.edu/2007/03/01 (2007)
2. Guha, R.: Toward the intelligent web systems. In: Proceedings of IEEE CS, First International Conference on Computational Intelligence, Communication Systems and Network, pp. 459–463. IEEE, Los Alamitos (2009)
3. Handler, J., Shadbolt, N., Hall, W., Berners-Lee, T., Weitzner, D.: Web science: an interdisciplinary approach to understanding the web. Commun. ACM **51**(7), 60–69 (2008)
4. Chong, F., Carraro, G.: Architecture Strategies for Catching the Long Tail. Microsoft Corporation, Redmond (2006)
5. Banerjee, J., Aziz, S.: SOA: the missing link between enterprise architecture and solution architecture. SETLabs Brief. **5**(2), 69–80 (2007)
6. Barners-Lee, T.: Linked data. http://www.w3.org/DesignIssues/LinkedData.html (2012)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. Special issue on linked data. Int. J. Semant. Web Inf. Syst. (IJSWIS). http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf (2012)
8. Niemann, B., et al.: Introducing Semantic Technologies and the Vision of the Semantic Web, SICoP White Paper (2005)
9. HADOOP: http://en.wikipedia.org/wiki/Hadoop (2010)
10. Taft, D.: IBM's M2 Project Taps Hadoop for Massive Mashups. www.eweek.com (2010)
11. Wikipedia: Free and open source software. http://en.wikipedia.org/wiki/Free_and_open-source_software. Accessed July 2012

12. Wikipedia: Web Ontology Language. http://en.wikipedia.org/wiki/Web_Ontology_Language. Accessed July 2012
13. Code{4}lib: Library Ontology. http://wiki.code4lib.org/index.php/Library_Ontology Accessed July 2012
14. Sun Microsystem: Introduction to Cloud Computing Architecture, White Paper, 1st edn. (2009)
15. Sun Microsystem: Open Source & Cloud Computing: On-Demand, Innovative IT on a Massive Scale (2012)
16. Singh, A., Korupolu, M., Mahapatra, D.: Server-storage virtualization: integration and load balancing in data centers. In: IEEE/ACM Supercomputing (SC) Conference. IEEE Press, Piscataway (2008)
17. VMWARE: Virtualization overview. www.vmware.com (2012)
18. Reservoir Consortium: Resources and Services Virtualization Without Barriers. Scientific Report (2009)
19. Pearson, S.: Taking Account of Privacy when Designing Cloud Computing Services. HP Labs, Bristol (2009)
20. Jansen, W.A.: Cloud Hooks: Security and Privacy Issues in Cloud Computing. NIST
21. Pressman, R.: Software Engineering: A Practitioner's Approach, 7th edn. McGraw-Hill Higher Education, New York (2009)
22. Sommerville, I.: Software Engineering, 8th edn. Pearson Education, Harlow (2006)
23. Cavanaugh, E.: Web services: benefits, challenges, and a unique, visual development solution. www.altova.com (2006)
24. Nickull, D., et al.: Service Oriented Architecture (SOA) and Specialized Messaging Patterns (2007)
25. Web services-Axis: axis.apache.org/axis (2012)
26. W3C: Web services Description Language (WSDL) Version 2.0 (2012)
27. Brambilla, M. et al.: A Software Engineering Approach to Design and Development of Semantic Web Service Applications (2006)
28. Salesforce.com: Agile Development Meets Cloud Computing for Extraordinary Results. www.salesforce.com (2009)
29. Guha, R., Al-Dabass, D.: Impact of Web 2.0 and cloud computing platform on software engineering. In: Proceedings of 1st International Symposium on Electronic System Design (ISED) (2010)
30. DeMarco, T., Lister, T.: Waltzing with Bears: Managing Risk on Software Projects. Dorset House Publishing Company, Incorporated, New York (2003)