Chapter 2
Envisioning the Cloud-Induced Transformations in the Software Engineering Discipline

Pethuru Raj, Veeramuthu Venkatesh, and Rengarajan Amirtharajan

Abstract The software engineering field is on the move. The contributions of software solutions for IT-inspired business automation, acceleration, and augmentation are enormous. The business values are also rapidly growing with the constant and consistent maturity and stability of software technologies, processes, infrastructures, frameworks, architectural patterns, and tools. On the other hand, the uncertainty in the global economy has a direct bearing on the IT budgets of worldwide organizations. That is, they are expecting greater flexibility, responsiveness, and accountability from their IT division, which is being chronically touted as the cost center. This insists on shorter delivery cycles and on delivering low-cost yet high-quality solutions. Cloud computing prescribes a distinguished delivery model that helps IT organizations to provide quality solutions efficiently in a manner that suits to evolving business needs. In this chapter, we are to focus how software-development tasks can get greatly simplified and streamlined with cloud-centric development processes, practices, platforms, and patterns.

Keywords Cloud computing • Software engineering • Global software development • Model-driven architecture • MDA • Lean methodology • Distributed computing

2.1 Introduction

The number of pioneering discoveries in the Internet space is quite large. In the recent past, the availability of devices and tools to access online and on-demand professional and personal services has increased dramatically. Software has been

P. Raj (✉)
Wipro Technologies, Bangalore 560035, India
e-mail: peterindia@gmail.com

V. Venkatesh • R. Amirtharajan
School of Electrical and Electronics Engineering, SASTRA University,
Thanjavur, Tamil Nadu, India

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_2,

pervasive and persuasive. It runs on almost all kinds of everyday devices that are increasingly interconnected as well as Internet-connected. This deeper and extreme connectivity opens up fresh possibilities and opportunities for students, scholars, and scientists. The devices at the ground level are seamlessly integrated with cyber applications at remote, online, on-demand cloud servers. The hardware and software infrastructure solutions need to be extremely scalable, nimble, available, high-performing, dynamic, modifiable, real-time, and completely secure. Cloud computing is changing the total IT landscape by presenting every single and tangible IT resource as a service over any network. This strategically sound service enablement decimates all kinds of dependencies, portability, interoperability issues, etc.

Cloud services and applications are becoming very popular and penetrative these days. Increasingly, both business and IT applications are being modernized appropriately and moved to clouds to be subsequently subscribed and consumed by global user programs and people directly anytime anywhere for free or a fee. The aspect of software delivery is henceforth for a paradigm shift with the smart leverage of cloud concepts and competencies. Now there is a noteworthy trend emerging fast to inspire professionals and professors to pronounce the role and responsibility of clouds in software engineering. That is, not only cloud-based software delivery but also cloud-based software development and debugging are insisted as the need of the hour. On carefully considering the happenings, it is no exaggeration to say that the end-to-end software production, provision, protection, and preservation are to happen in virtualized IT environments in a cost-effective, compact, and cognitive fashion. Another interesting and strategic pointer is that the number and the type of input/output devices interacting with remote, online, and on-demand cloud are on the climb. Besides fixed and portable computing machines, there are slim and sleek mobile, implantable, and wearable devices emerging to access, use, and orchestrate a wider variety of disparate and distributed professional as well as personal cloud services. The urgent thing is to embark on modernizing and refining the currently used application development processes and practices in order to make cloud-based software engineering simpler, successful, and sustainable.

In this chapter, we discuss cloud-sponsored transformations for IT and leveraging clouds for global software development and present a reflection on software engineering. The combination of agility and cloud infrastructure for next-generation software engineering, the convergence of service and cloud paradigms, the amalgamation of model-driven architecture, and the cloud and various mechanisms for assisting cloud software development are also discussed. At the end, cloud platform solutions for software engineering are discussed, and software engineering challenges with respect to cloud environments are also presented.

2.2 Cloud-Sponsored Transformations for IT

The popularity of the cloud paradigm is surging, and it is overwhelmingly accepted as the disruptive, transformative, and innovative technology for the entire IT field. The direct benefits include IT agility through rationalization, simplification,

higher utilization, and optimization. This section explores the tectonic and seismic shifts of IT through the cloud concepts.

- *Adaptive IT* – There are a number of cloud-inspired innovations in the form of promising, potential, and powerful deployment; delivery; pricing; and consumption models in order to sustain the IT value for businesses. With IT agility setting in seamlessly, business agility, autonomy, and adaptivity are being guaranteed with the adoption and adaption of cloud idea.
- *People IT* – Clouds support centralized yet federated working model. It operates at a global level. For example, today there are hundreds of thousands of smartphone applications and services accumulated and delivered via mobile clouds. With ultrahigh broadband communication infrastructures and advanced to compute clouds in place, the vision of the Internet of devices, services, and things is to see a neat and nice reality. Self-, surroundings-, and situation-aware services will become common, plentiful, and cheap; thereby, IT promptly deals with peoples' needs precisely and delivers on them directly.
- *Green IT* – The whole world is becoming conscious about the power energy consumption and the heat getting dissipated into our living environment. There are calculated campaigns at different levels for arresting climate change and for sustainable environment through less greenhouse-gas emission. IT is being approached for arriving at competent green solutions. Grid and cloud computing concepts are the leading concepts for green environment. Especially the smart energy grid and the Internet of Energy (IoE) disciplines are gaining a lot of ground in order to contribute decisively for the global goal of sustainability. The much-published and proclaimed cloud paradigm leads to lean compute, communication, and storage infrastructures, which significantly reduce the electricity consumption.
- *Optimal IT* – There are a number of worthwhile optimizations happening in the business-enabling IT space. "More with less" has become the buzzword for both business and IT managers. Cloud enablement has become the mandatory thing for IT divisions as there are several distinct benefits getting accrued out of this empowerment. Cloud certainly has the wherewithal for the goals behind the IT optimization drive.
- *Next-Generation IT* – With a number of delectable advancements in wireless and wired broadband communication space, the future Internet is being positioned as the central figure in conceiving and concretizing people-centric discoveries and inventions. With cloud emerging as the new-generation compute infrastructure, we will have connected, simplified, and smart IT that offers more influential and inferential capability to humans.
- *Converged, Collaborative, and Shared IT* – The cloud idea is fast penetrating into every tangible domain. Cloud's platforms are famous for not only software deployment and delivery but also for service design, development, debugging, and management. Further on, clouds, being the consolidated, converged, and centralized infrastructure, are being prescribed and presented as the best bet for enabling seamless and spontaneous service integration, orchestration, and

collaboration. With everything (application, platform, and infrastructure) are termed and touted as publicly discoverable, network-accessible, self-describing, autonomous, and multitenant services, clouds will soon become the collaboration hub. Especially business-aware, process-centric, and service-oriented composites can be easily realized with the cloud-based collaboration platform.

- *Real-Time IT* – Data's variety, volume, and velocity are on the climb. The current IT infrastructures are insufficient in order to extract actionable insights out of pouring data. Hence, the emergence of big data computing and analysis technologies are given due diligence and attention. These fast-maturing technologies are able to accomplish real-time transition from data to information and to knowledge. Cloud is the optimized, automated, and virtualized infrastructure for big data computing and analytics. That is, with the infrastructure support from clouds, big data computing model is to see a lot of improvements in the days ahead so that the ultimate goal of real-time analytics can be realized very fluently and flawlessly.

2.3 Leveraging Clouds for Global Software Development (GSD)

Globalization and distribution are the two key concepts in the IT field. Software development goes off nations' boundaries and tends toward places wherein quality software engineers and project managers are available in plenty. On-site, off-shoring, near-shoring, etc., are some of the recent buzzwords in IT circles due to these developments. That is, even a software project gets developed in different locations as the project team gets distributed across the globe. With the sharp turnarounds in a communication field, a kind of tighter coordination and collaboration among team members are possible in order to make project implementation successful and sustainable. In-sourcing has paved the way for outsourcing with the maturity of appropriate technologies. As widely known, software sharply enhances the competitive advantage and edge for businesses. Hence, global software development (GSD) has become a mandatory thing for the world organizations. Nevertheless, when embarking on GSD, organizations continue to face challenges in adhering to the development life cycle. The advent of the Internet has supported GSD by bringing new concepts and opportunities resulting in benefits such as scalability, flexibility, independence, reduced cost, resource pools, and usage tracking. It has also caused the emergence of new challenges in the way software is being delivered to stakeholders. Application software and data on the cloud are accessed through services, which follow SOA principles.

GSD is actually the software-development process incorporating teams spread across the globe in different locations, countries, and even continents. The driver for this sort of arrangement is by the fact that conducting software projects in multiple geographical locations is likely to result in benefits such as cost reduction and

reduced time to market, access to a larger skill pool, proximity to customer, and 24-h development by following the sun. But, at the same time, GSD brings challenges to distributed software-development activities due to geographic, cultural, linguistic, and temporal distance between the project development teams.

Because of the distance between the software-development teams, GSD encounters certain challenges in terms of collaboration, communication, coordination, culture, management, organizational, outsourcing, development process, development teams, and tools. The real motive for using the cloud for supporting GSD is that the cloud idea thrives as it is closely related to the service paradigm. That is, services are created, provisioned, and delivered from cloud-based service platforms. Since SOA runs a mechanism for development and management of distributed dynamic systems, and it evolved from the distributed-component-based approach, it is argued that cloud has the innate potential and strength to successfully cater for the challenges of GSD where a project is developed across different geographical locations. GSD challenges can be overcome through SOA. This will contribute to increased interoperability, diversification, and business and technology alignment. Cloud as the next-generation centralized and service-oriented infrastructure is capable of decimating all the internal as well as externally imposed challenges.

- Global Software Development (GSD) in Cloud Platforms [1] – Clouds offer instant resource provisioning, flexibility, on-the-fly scaling, and high availability for continuously evolving GSD-related activities. Some of the use cases include.
- Development Environments – With clouds, the ability to acquire, deploy, configure, and host development environments become "on-demand." The development environments are always on and always available to the implementation teams with fine-grained access control mechanisms. In addition, the development environments can be purpose-built with support for application-level tools, source code repositories, and programming tools. After the project is done, these can also be archived or destroyed. The other key element of these "on-demand" hosting environments is the flexibility through its quick "prototyping" support. Prototyping becomes flexible, in that as new code and ideas can be quickly turned into workable proof of concepts (PoCs) and tested.
- Developer Tools – Hosting developer tools such as IDEs and simple code editors in the cloud eliminates the need for developers to have local IDEs and other associated development tools, which are made available across time zones and places.
- Content Collaboration Spaces – Clouds make collaboration and coordination practical, intuitive, and flexible through easy enabling of content collaboration spaces, modeled after the social software domain tools like Facebook, but centering on project-related information like invoices, statements, RFPs, requirement documents, images, and data sets. These content spaces can automate many project-related tasks such as automatically creating MS Word versions of all imported text documents or as complex as running work flows to collate information from several different organizations working in collaboration. Each content space can be unique, created by composing a set of project requirements. Users can invite

internal and external collaborators into this customized environment, assigning appropriate roles and responsibilities. After the group's work is "complete," their content space can be archived or destroyed. These spaces can be designed to support distributed version control systems enabling social platform conversations and other content management features.

- Continuous Code Integration – Compute clouds let "compile-test-change" software cycle on the fly do continuous builds and integration checks to meet strict quality checks and development guidelines. They can also enforce policies for customized builds.
- APIs and Programming Frameworks – Clouds force developers to embrace standard programming model APIs where ever possible and adhere to style guides, conventions, and coding standards in meeting the specific project requirements. They also force developers to embrace new programming models and abstractions such as .NET Framework, GWT, Django, Rails, and Spring Framework for significantly increasing the overall productivity. One more feature of using clouds is that they enforce constraints, which push developers to address the critical next-generation programming challenges of multicore computing, parallel programming, and virtualization. As explained earlier in the chapter, global software development is picking up fast, and the emergence of clouds is to boost the GSD activities further.

2.4 A Reflection on Software Engineering

Radha Guha writes in [2] that over the last half-century, there have been robust and resilient advancements in the hardware engineering domain. That is, there are radical and rapid improvisations in computers, memory, storage, communication networks, mobile devices, and embedded systems. This has been incessantly pushing the need for larger and more complex software. Software development not only involves many different hardware elements, it also involves many different parties like end users and software engineers. That is why software development has become such an inherently complicated task. Software developers are analyzing, articulating, and adopting the proven and prescribed engineering disciplines. That is, leveraging 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 the stipulated time and agreed budget and by its throughput, user-friendliness, consumability, dependability, and modifiability.

Typically, a software engineering engagement starts off with an explicit and elegant process model comprising several formally defined and synchronized phases. The whole development process of software from its conceptualization to implementation to operation and retirement is called the software-development life cycle (SDLC). SDLC goes through several sub-activities like requirement's gathering, planning, design, coding, testing, deployment, maintenance, and retirement.

These activities are well 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. The development of an avionic embedded system will adopt a different process model from development of a Web application.

Even though software engineering [3] takes the engineering approach, the success of software products is more difficult than products from other engineering domains like mechanical engineering or civil engineering. This is because software is intangible during its development. Software project managers use a number of techniques and tools to monitor the software building activities in a more visible way. These activities include software project tracking and control, risk management, quality assurance, measurements, configuration management, work product or document's generation, review, and reusability management.

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. Therefore, there is a continued and consistent focus on simplifying and streamlining software implementation. In this chapter, we are to see some of the critical and crucial improvements in software engineering process with the availability of cloud infrastructures.

The Evolutions and Revolutions in the Software Engineering Field – There are a number of desirable and delectable advancements in the field of software engineering in order to make the tough task of software construction easier and quicker. This section describes the different levels and layers in which the software engineering discipline and domain evolve.

At the *building-block level*, data, procedures, classes, components, agents, aspects, events, and services are the key abstraction and encapsulation units for building and orchestrating software modules into various types of specific and generic software. Services especially contribute in legacy modernization and migration to open service-oriented platforms (SOPs) besides facilitating the integration of disparate, distributed, and decentralized applications. In short, building blocks are the key ingredient enabling software elegance, excellence, and evolution. In the recent past, formal models in digital format and service composites are evolving fast in order to further simplify and streamline the tough task of software assembly and implementation. As software complexity is on the rise, the need for fresh thoughts and techniques is too on the climb.

On the *language level*, a bevy of programming languages (open source as well as proprietary) were produced and promoted by individuals, innovators, and institutions. Even, there are efforts underway in order to leverage fit-for-purpose languages to build different parts and portions of software applications. Software libraries are growing in number, and the ideas of software factory and industrialization are picking up fast lately. Service registry and repository are an interesting phenomenon for speeding up software realization and maintenance. Programming languages

and approaches thrive as there are different programming paradigms such as object orientation, event- and model-driven concepts, componentization, and service orientation. Further on, there are script languages in the recent past generating and getting a lot of attention due to their unique ability of achieving more with less code. Formal models in digitalized format and service composites are turning out to be a blessing in disguise for the success and survival of software engineering. There are domain-specific languages (DSLs) that could cater to the specific demands of domains quite easily and quickly.

As far as *development environments* are concerned, there are a number of diverse application building platforms for halving the software developmental complexity and cost. That is, there are a slew of integrated development environments (IDEs), rapid application development (RAD) tools, code generators and cartridges, enabling CASE tools, compilers, debuggers, profilers, purpose-specific engines, generic and specific frameworks, best practices, key guidelines, etc. Plug and play mechanism has gained a lot with the overwhelming adoption of eclipse IDE for inserting and instantiating different language compilers and interpreters. The long-standing objectives of platform portability (Java) and language portability (.NET Framework) are being achieved at a middleware level. There are standards-compliant toolkits for process modeling, simulation, improvement, investigation, and mapping. Services as the well-qualified process elements are being discovered, compared, and orchestrated for partial or full process automation.

At the *process level*, waterfall is the earliest one, and thereafter there came a number of delicious variations in software-development methodology with each one having both pros and cons. Iterations, increments, and integrations are being touted as the fundamental characteristics for swifter software production. Agile programming is gaining a lot of ground as business changes are more frequent than ever before and software complexity is also growing. Agility and authenticity in software building are graciously achieved with improved and constant interactions with customers and with the enhanced visibility and controllability on software implementation procedures. Agility, being a well-known horizontal technique, matches, mixes, and merges with other paradigms such as service-oriented programming and model-driven software development to considerably assist in lessening the workload of software developers and coders. Another noteworthy trend is that rather than code-based implementation, configuration-based software production catches up fast.

At the *infrastructural level,* the cloud idea has brought in innumerable transformations. The target of IT agility is seeing a neat and nice reality and this in turn could lead to business agility. Technically, cloud-inspired infrastructures are virtualized, elastic, self-servicing, automated, and shared. Due to the unique capabilities and competencies of cloud IT infrastructures (in short, clouds), all kinds of enterprise IT platforms (development, execution, management, governance, and delivery) are being accordingly manipulated and migrated to be hosted in clouds, which are extremely converged, optimized, dynamic, lean, and green. Such meteoric movement decisively empowers application platforms to be multitenant, unified, and centralized catering to multiple customers and users with all the enhanced productivity,

extensibility, and effectiveness. In other words, cloud platforms are set to rule and reign the IT world in the days to unfold. In other words, platforms are getting service-enabled so that any service (application, platform, and infrastructure) can discover and use them without any barriers. Service enablement actually expresses and exposes every IT resource as a service so that all kinds of the resource's incompatibilities are decimated completely. That is, resources readily connect, concur, compose, and collaborate with one another without any externally or internally imposed constrictions, contradictions, and confusions. In a nutshell, the unassailable service science has come as a unifying factor for the dilapidated and divergent IT world.

In summary, the deeply dissected, discoursed, and deliberated software-development discipline is going through a number of pioneering and positive changes as described above.

2.5 Combination of Agility and Cloud Infrastructure for Next-Generation Software Engineering

As indicated previously, there have been many turns and twists in the hot field of software engineering. It is an unquestionable fact that the cloud paradigm, without an iota of doubt, has impacted the entire IT elegantly and exceedingly. Besides presenting a bright future on the aspect of centralized deployment, delivery, and management of IT resources, the cloud idea has opened up fresh opportunities and possibilities for cloud-based software design, development, and debugging in a simplified and systematic fashion. That is, with the overwhelming adoption and adaption of cloud infrastructures (private, public, community, and hybrid), producing and preserving enterprise-scale, mission-critical, and value-added software are going to be definitely distinct. There are four key drivers that unanimously elevate the software development to be advanced to an accomplished in a cloud. These are:

- *Time, Cost, and Productivity* – The developer community is being mandated to do more, quicker, and with fewer resources.
- *Distributed Complex Sourcing* – Due to various reasons, IT project team members are geographically dispersed.
- *Faster Delivery of Innovation* – The focus is on enabling architects and developers to think ingeniously in order to deliver business value.
- *Increasing Complexity* – In today's world, an enterprise-scale project easily consumes several million lines resulting in more complexity.

In order to reduce complexity, resources, cost, and time considerably, professionals and professors are vigorously and rigorously striving and searching for incredibly inventive solutions. Newer concepts, process optimization, best practices, fresh programming models, state-of-the-art platforms, design patterns and metrics, and advanced tools are being increasingly unearthed and utilized for lessening the software development workload. Researchers are continuously at work in order to

discover competent and compact methods and mechanisms for simplifying and streamlining the increasingly multifaceted tasks of constructing and conserving next-generation software systems. The major benefits of agile methodology over the traditional methods are:

- Faster time to market
- Quick return on investment
- Shorter release cycles
- Better adaptability and responsiveness to business changing requirements
- Early detection of failure and immediate correction

There are several agile development methods such as Scrum, extreme programming, test-driven development, and lean software development [4]. With agile models, business houses expect that services and solutions are being delivered incrementally earlier rather than later, and delivery cycle time period comes down sharply. That is, one delivery cycle takes up from 2 to 4 weeks. However, in the midst of these turnarounds, there arise a number of critical challenges, as mentioned below:

- High effort and cost involved in setting up infrastructures
- Lack of skilled resources
- Lack of ability to build applications from multiple places across the globe

There are a few popular cloud platforms available in order to enable software development in cloud environments. Google App Engine, salesforce.com, cloudfoundry.org, cloudbees.com, corenttech.com, heroku.com, windowsazure.com, etc., are the leading platforms for cloud-based application development, scaling, and sustainability.

Collabnet (http://www.collab.net/), a product firm for enabling software development in cloud-based platforms, expounds and enlightens on the seamless convergence of the agile programming models, application lifecycle management (ALM) product, and clouds for a precise and decisive answer for the perpetual software engineering challenges, changes, and concerns. It convincingly argues that cloud technologies reduce development barriers by providing benefits in the following critical areas:

- *Availability* – Code is centralized and infrastructure is scalable and available on demand.
- *Access* – Ensures flexible access to test environments and transparency to project data for the entire team.
- *Overhead* – Reduced support overhead, no upgrade latency – teams use an on-demand model to get what they need, quickly and easily.

Agile processes set the strong and stimulating foundation for distributed teams to work closely together with all the right and relevant stakeholders to better anticipate and respond to user expectations. Agile teams today are empowered to clearly communicate with users to act and react expediently to their feedback. That is, they are able to collaboratively and cleverly iterate toward the desired state and user satisfaction. Cloud intrinsically facilitates open collaboration across geographies

and time zones with little investment or risk. With more and more development and test activities moving toward clouds, organizations are able to save time and money using virtual and shared resources on need basis. Developers could save time by leaving configuration, upgrades, and maintenance to cloud providers, who usually employ highly educated and experienced people. Anytime anywhere access is facilitated for those with proper authentication and authorization, and assets are completely centralized and controlled.

Agile and cloud are being positioned together and prescribed as a powerful and pathbreaking combination for the software-development community. This might seem counterintuitive to those entrenched in waterfall processes or those comfortable with the idea of a daily stand-up and colocated teams. The reality is altogether different. That is, there are a number of technical and business cases emerging for using the agile methods in the cloud. The agility concepts make development teams responsive to the changing needs of businesses and empower them to be adaptable and flexible. Further on, proven agile processes help to break down all sorts of barriers and blockages between development and production, allowing teams to work together to concentrate on meeting stakeholder expectations. The synchronization of agile and cloud paradigms fully free up developers from all kinds of difficulties to achieve more with less, to innovate fast, and to ultimately bring value to the business.

2.6 Convergence of Service and Cloud Paradigms

The service idea has matured and stabilized as the dominant approach for designing, developing, and delivering open, sustainable, and interoperable service-oriented systems for enterprise, Web, embedded, and cloud spaces. Even many of the modules of packaged business software solutions are modified and presented as services. Services are publicly discoverable and accessible, reusable, and composable modules for building distinct and specific applications through configuration and customization, runtime matching, selection and usage of distributed, disparate and decentralized services, replacement of existing service components through the substitution of new advanced service components, and service orchestration. Services as process elements are supporting and sustaining process-oriented systems, which are generally more flexible. That is, operation and controlling of software solutions at process level considerably reduce the software development, management, and maintenance tasks.

Thus, the process propensity of the service paradigm and cloud-centric service-oriented infrastructures and platforms bring a number of distinct advantages for software engineering. Services and cloud computing have garnered much attention from both industry and academia because they enable the rapid and radical development of enterprise-scale, mission-critical, high-performance, dynamic, and distributed applications. Agility, adaptivity, and affordability, the prime characteristics of next-generation software systems, can be realized with the smart leverage of

processes, services, and cloud platforms. Precisely speaking, the service paradigm is to energize futuristic software design, whereas cloud platforms are being tipped and touted as the next-generation service-centric platforms for service development, deployment, management, and delivery.

Service-Oriented Software Development – It is to see a lot of delectable and decisive shifts with the adoption of cloud platforms. The smooth and seamless convergence of services and clouds promises shining days for software-development community. Of course, there are a few challenges that need utmost attention from scholars, scientists, and students. Security, visibility, controllability, performance, availability, usability, etc., need to be obviated in order to fast-track service-based software implementation in clouds.

As widely pronounced, services are being positioned as the most flexible and fertile component for software production. That is, software solutions are made of interoperable services. It is all about the dynamic discovery and purposeful interactions among a variety of services that are local or remote, business or IT-centric, and owned or subscribed from third-party service providers. Services are standards-compliant, self-describing, and autonomous entities in order to decimate all kinds of dependencies and incompatibilities, to promote seamless and spontaneous collaborations, and to share each of their capability and competency with others over networks. Process and workflow-based service compositions result in dynamic applications that are highly portable. XML is the key data representation, exchange, and persistence mechanism facilitating service interoperability. Policies are being framed and encouraged in order to achieve automated service finding, binding, usage, monitoring, and governance. The essence of service governance is to explicitly establish pragmatic policies and enforce them stringently. With a consistent rise in automation, there is a possibility for deviation and distraction, and hence the service governance discipline is gaining a lot of ground these days.

As there is a clear distinction between service users and providers, service-level agreement (SLA) and even operation-level agreement (OLA) are becoming vital for service-centric business success and survival. Furthermore, there are geographically distributed several providers providing identical or similar services and hence SLA, which unambiguously describes runtime requirements that govern a service's interactions with different users, has come as a deciding factor for service selection and utilization. A service contract describes its interface and the associated contractual obligations. Using standard protocols and respective interfaces, application developers can dynamically search, discover, compose, test, verify, and execute services in their applications at runtime. In a nutshell, SOA-based application development is through service registration, discovery, assessment, and composition, which primarily involves three stakeholders:

- A service provider is one who develops and hosts the service in cloud platforms.
- A service consumer is a person or program that finds and uses a service to build an application.
- A service broker mediates between service providers and consumers. It is a program or professional in helping out providers publishing their unique services and guiding consumers to identify ideal services.

The service science is on the growth trajectory. There are service-oriented platforms, patterns, procedures, practices, products, and packages. Service management has become a niche area of study and research. The knowledge-driven service era is to dawn with the availability of competent service-centric technologies, infrastructures and processes, toolsets, architectures, and frameworks. Service engineering is picking up fast with the sufficient tweaking and tuning of software engineering principles, techniques, and tips. Everything related to IT is being conscientiously manipulated and presented as a service for the outside world setting the context and case for IT as a service (ITaaS). In other words, any service can connect and cooperate with other services individually or collectively to make bigger and better things for the total humanity.

The Synchronization Between Service and Cloud Ideas – As explained and elucidated above, the service and cloud computing models together signal a sunny and shining days ahead for software building. A combined framework comprising the service and the cloud concepts goes a long way in halving the application development drudgery. Cloud-centric application development gets a consolidated, centralized, virtualized, and shared IT infrastructure for efficiently constructing and preserving applications. Multitenancy, auto-provisioning, and elasticity features are the strong business and technical cases for embracing the cloud idea.

Now with the concepts of the Inter-cloud that are fast emerging and evolving, cloud integration and federation aspects are bound to grow significantly. That is, connected and federated clouds will become the common, casual, and cheap thing for next-generation enterprise IT. The federation of multiple types of clouds (mobile, device, sensor, knowledge, information cloud, high-performance cloud, etc.) is to enable distributed, global, and collaborative software development [5]. The open and industry-strength interoperability standards of SOA empower service-sponsored cloud integration and, on the other hand, cloud-hosted service integration. In short, the cloud grid is not an option but a necessity considering the growing complexity of IT toward sustaining the business dynamism.

The concept of designing and developing applications using SOA and delivery through cloud is to explode. Cloud brokerage firms could maintain cloud-hosted service registry and repository that works out as a single point of contact for global application developers. The service metadata offers the exact location, interface, and contract of services being probed for use. Service developers could host their services in service platforms of worldwide cloud providers, and this enables application developers to search and choose right and relevant services based on the business requirements. Service providers could also host integrated development environments and rapid application development tools, code generators and cartridges, debuggers, simulators, emulators, etc., in their own clouds or in third-party cloud infrastructures. Furthermore, they could publish software artifacts such as modifiable and extendible business processes, workflows, application templates, user interfaces, data schema, and policies to facilitate software development and generation. Developers can find viable and value-added services from multiple service providers and leverage these artifacts in order to come out with service-oriented

applications. The fast-maturing federation science is to dictate the future of software engineering. In short, there are cloud-based components such as:

- Application development artifacts such as templates, processes, and workflows
- Service development environments and tools
- Service registry repository
- SCA-compliant application implementation platforms with service discovery, integration, and orchestration features and facilities leveraging the application artifacts
- Application delivery as a service via the Internet as the cheap and open communication infrastructure

Service-Based Software Design and Development – Development of service systems remains a quiet big challenge because services are being developed by different entities and deposited in geographically distributed locations. For an application to fructify, diverse services need to be smartly collected and consolidated. Different services are covered up with disparate policies. Varying capabilities decorate services. Also application development process is increasingly diversified because application developers, service brokers, and application service providers are distributed. The coordination here is very important for the SOA-based IT and business successes. Standardized protocols, messaging mechanisms, and interfaces are very essential services to be linked remotely and resiliently.

Software engineering revolves around two main activities: decomposition and composition. As business problem evolves and enlarges, the act of decomposition of business problem is required as our mental capability is limited. Once an appropriate solution for the business problem is designed, then identify those solution building blocks and compose them to develop the solution.

Similar to other development methodologies, service-oriented software development starts with requirements extraction, elucidation, and engineering. During this phase, the application developer develops a business model; works with the customer to articulate, analyze, authenticate, and refine requirements; designs a workflow for the business model; and finally decomposes the requirements into smaller and manageable modules. Then the application developer sends each of the disintegrated and disengaged requirements to a service brokerage to find suitable services that satisfy the enshrined requirements. Once the right services are identified for each of the requirement parts, the application developer simply composes them into an application. Service component architecture (SCA) is a recent architectural style enabling application componentization into service modules that in turn get assembled to form a single entity. There are SCA-compliant IDEs from different product vendors. In some cases, correct services might not be available and hence one has to develop those services from the scratch.

Cloud-Based Software Delivery – Software engineering encompasses not only the software-development processes but also the effective delivery of the developed software to users, which includes software deployment and maintenance. However, SOA does not prescribe any specific methods for software deployment, management, governance, and enhancement. These can be decided and activated by software service organizations differently. Clouds as the standardized and smart infrastructure

come to the rescue here by ensuring effective application delivery. Applications can be affordably deployed and maintained in advanced cloud platforms. Application capabilities can be provided as a service. All kinds of non-functional (quality of service (QoS)) attributes are effortlessly accomplished with clouds. Anytime anywhere resource access is being facilitated. Centralized monitoring and management are remarkably simplified here. That is, clouds as the next-generation service-oriented infrastructures (SOIs) have emerged in correct time in order to take the service idea to greater heights. It is therefore no exaggeration to proclaim that the software engineering field is greatly and grandiosely empowered by evolving cloud concepts.

Agile Service Networks (ASNs) [6, 7] – Cloud computing's high flexibility needs novel software engineering approaches and technologies to deliver agile, flexible, scalable, yet secure software solutions with full technical and business gains. One way is to allow applications to do the computing in cloud, and the other is to allow users to integrate with the applications. Agile service networks (ASNs) are themselves an emerging paradigm envisioning collaborative and dynamic service interactions (network edges) among global service-oriented applications (network nodes). ASNs can be used as a paradigm for software engineering in the cloud, since they are indeed able to deliver solutions which are both compliant to the cloud's needs and able to harness it, bringing about its full potential.

Context adaptation is used in ASNs to achieve agility. The concept of ASN is defined as a consequence of "late service binding." In the context of services' dynamism, which is achieved through late service binding, ASNs become a perfect example of how agility can be achieved in SOA systems. Adaptation is presented as one of the main tenets of SOA. This paradigm regards highly dynamic systems within a rapidly changing context to which applications must adapt. In this sense, ASNs are used to exemplify industrial needs for adaptive, context-aware systems.

ASN Key Features – ASNs are dynamic entities. Dynamism is seen as an essential part of the service interactions within collaborative industries (i.e., industrial value networks). Dynamism in ASNs is the trigger to service rearrangement and application adaptation. For example, an ASN made of collaborative resource brokering such as distributed stock markets is dynamic in the sense that different partners may participate actively, others may be dynamically added while brokering is ongoing, others may retire from the brokering process, and others may dynamically change their business goals and hence their brokering strategy. ASNs are business-oriented: ASNs are borne out of business corporative collaborations and represent complex service applications interacting in a networked business scenario involving multiple corporations or partners in different sites (i.e., different geo-locations). Within ASNs, business value can be computed, analyzed, and maximized.

Cloud-Induced Software Engineering Challenges – As widely reported, there are some important concerns with public clouds. Security, controllability, visibility, performance, and availability are the major issues. Virtualization, the central technology for the massive uptake and incontestable success of the cloud idea, has introduced new security holes. Typically, public clouds are more or less accommodating several customers to be economical, and there are real dangers and risks in a shared environment. If a cloud is not available for a few minutes, the resulting

loss would be very enormous necessitating the sharp increment in guaranteeing cloud availability. Cloud reliability is another central and crucial factor not to be sidestepped easily. The security of data in rest or in transit has to be infallibly secure, and cryptography is the major source of inspiration for data security in a cloud environment. Identity and access management solutions are being conceived and concretized for the more open and risky cloud systems. Besides, application and service security and network and physical security aspects are also critical in a cloud environment.

Smartphone applications are becoming very popular and very large in the number with the massive production and release of slim and sleek, handy and trendy, yet multifaceted mobile phones. As there are literally more mobile devices compared to desktop and other powerful compute machines, application development for the fastest-growing mobile space is gaining unprecedented importance. Mobile technologies, application architectures and frameworks, toolsets, service delivery platforms, hypervisors for mobile devices, unified and integrated application development environments, etc., are being produced in plenty by competing parties in order to score over others in the mind and market shares. There are specific cloud infrastructures for securely storing a variety of mobile data, content, mails, services, and applications. Besides cell phones and smartphones, other mobile and portable devices incessantly capturing the imagination of people are the powerful tablets. Thus, there are several dimensions and directions in which the nifty and niche content and application development activities for the mobile landscape are proceeding.

With cloud emerging as the centralized place for mobile services, the days of anywhere anytime information and service access and upkeep are bright. Especially form builder applications for smartphones are being made available so that users could creatively produce their own forms in order to indulge in commercial and financial activities on the move. Hundreds of thousands of smartphone applications are being built, hosted, and subscribed by various smartphone vendors. Games are the other prominent and dominant entities for the mobile world. Precisely speaking, mobiles and clouds are increasingly coming closer for context-aware, customer-centric, and cognitive applications.

In summary, the penetration of cloud idea is simply mesmerizing and momentous. The cloud-based platforms are being positioned as the dynamic, converged, and fit-for-purpose ones for application engineering not only for enterprise IT but also for embedded IT, which incidentally includes mobile, wearable, portable, fixed, nomadic, wireless, implantable, and invisible devices. Extremely and deeply connected applications and services are bound to rule the IT in the coming days, and the cloud paradigm is the definite and decisive contributor for the future IT.

Although, the service and cloud concepts have greater affinity in strengthening software development and delivery, there are some serious issues to be addressed urgently in order to eliminate all kinds of doubts of in the minds of enterprise executives in order to reach into the promised land of cloud-sponsored service era.

2.7 Amalgamation of Model-Driven Architecture and the Cloud Paradigms

Modeling has been a fundamental and foundational activity for ages. Before a complex system gets formed, a model of the system is created as it could throw some light about the system's final structure and behavior. Models could extract and expose any kind of hidden risks and lacunae in system functioning and give a bit of confidence for designers and developers to plan and proceed obviating all kinds of barriers. Models give an overall understanding about the system to be built. In short, models decompose the system into a collection of smaller and manageable chunks in order to empower engineers to have a firm grip and grasp of the system under implementation. Modeling is one of the prominent and dominant complexity-mitigation techniques as systems across domains are fast-growing in complexity.

As IT systems are growing complexity, formal models are presented as the next-generation abstraction and encapsulation unit for them. In the recent past, models have been used as building blocks for having portable, sustainable, and flexible IT systems. Models are created digitally, stored, refined, and revitalized as per the changing needs. There are formats such as XML Metadata Interchange (XMI) for exporting models over the network or any other media to other systems as inputs for further processing. There are unified and visual languages and standardized notations emerging and energizing compact and formal model representation, persistence, manipulation, and exchange. Product vendors and open source software developers have come out with innumerable software tools for facilitating model creation, transformation, verification, validation, and exporting. For object orientation, unified modeling language (UML) has been the standard one for defining and describing models for various constructs and activities. For component-based assembly and service-orientation programming, UML profiles have been created in order to keep UML as the modeling language for software engineering. Further on, there are processing modeling and execution languages such as BPML and BPEL and notations such as BPMN in order to develop process-centric applications. That is, process models act as the building blocks for system engineering.

Model-driven architecture (MDA) is the associated application architecture. Model-driven software engineering (MDSE) is being presented as the most dynamic and drastic method for application engineering. Emerging and evolving MDSE techniques can automate the development of new cloud applications programmatically. Typically, cloud applications are a seamless union of several unique services running on different IT platforms. That is, for producing competent cloud applications, all the right and relevant services from diverse and geographically distributed servers have to be meticulously found, bound, and linked up in order to build and sustain modular (loosely coupled and highly cohesive) cloud applications. Precisely speaking, services have been overwhelmingly accepted as the most productive and pliable building block for realizing adaptive, mission-critical, and enterprise-scale applications.

For building service-oriented cloud applications, there is a need for modernizing all the legacy software modules into services. Model-driven reverse engineering techniques are capable of discovering and generating standardized models out of legacy software modules. The overall idea is to use such techniques and enabling frameworks such as MoDisco framework to speed up the task of model creation from legacy modules. These formal models can be subjected to further transformation to derive corresponding services that in collaborate with other cloud-based services in order to craft fresh cloud applications quickly. That is, just as software as a service (SaaS) paradigm, the notion of modeling as a service (MaaS) is to see brighter days ahead especially in assisting the formation of cloud applications out of existing non-cloud applications. As there are billions of legacy code still contributing extensively for fortune corporations across the globe, MaaS is to grow exponentially. There will be processes to be defined, frameworks to be produced, cloud platforms to be immensely utilized, etc. Reverse engineering of application modules into a PIM and then into one or more PSMs to automate the service realization out of old software components is the cleverest and clear-cut approach for the forthcoming cloud era. It is keenly anticipated that similar to SaaS, MaaS will become a pioneering initiative. Here are some possible applications of MaaS [8]:

- Creation of collaborative and distributed modeling tools to allow the specification and sharing of software models among team members in real time.
- Definition of modeling mash-ups as a combination of MDSE services from different vendors.
- Availability of model transformation engines in the cloud to provide platform-independent model management services.
- Improving Scalability of MDSE – Models of real-life applications (especially those obtained by reverse engineering of running systems) are usually very large. Modeling services in the cloud would ensure the scalability of MDSE techniques in those scenarios.
- Facilitating Model Execution and Evolution – Moving code-generation and simulation services to cloud would facilitate the deployment and evolution of software applications (regardless of whether those applications were implemented as SaaS) and substantially reduce the time to market. The cloud service providers (CSPs) with their infrastructure administration experts could set up the relevant infrastructures to compile and deploy the applications quickly.
- Solving Tool Interoperability Problems – Exchanging data (and metadata) among MDSE tools is one of the major challenges nowadays. So far, the problem is being addressed by defining bridges among the tools, but MaaS is to offer a more transparent and global solution to this problem. For instance, bridges could be defined as services and executed on demand automatically by other services when incompatibility issues surface.
- Distributed Global Model Management – Complex MDSE projects involve several models (possibly conforming to different metamodels), model transformations, model injectors and projectors, etc. The MaaS paradigm is to facilitate the manipulation of all these modeling artifacts in a distributed environment.

Model-Driven and Cloud-Sponsored Legacy Enablement Toward Mainstream Computing – Long-living software systems [9] constantly undergo a number of changes during their lifetime. These are triggered by a changing system context (system usage and technology stacks) and/or changing system requirements. The changes include functional and/or non-functional attributes, for example, the capability and capacity of the system to deal with increasing system workload. The latter is often a direct consequence of providing the access to existing systems over the Internet, for example, for the integration of the systems into novel service compositions.

Cloud computing brings a new ray of hope of addressing this issue very deftly by providing almost unlimited amount of compute or storage resources. In order to utilize this new offer, long-living software systems have to be migrated to cloud. Often this implies major changes (invasive) to the system structure for which no systematic engineering process is available today. This vacuum can lead to high risks or even project failures. There has to be a bridge between the conventional and classic computing and the cloud computing architectures. That is, the age-old architectural styles and patterns such as three-tier client/server architecture do help in building business applications. With cloud's emergence, new-generation architectural styles emerge for the efficient use of the almost unlimited computational resources in the cloud. There is a new architectural style (the so-called SPOSAD style: Shared, Polymorphic, Scalable Application and Data) allowing massive replication of the business logic, which is enabled by a smart physical data distribution. This evolution in different directions and dimensions has to be bridged through a systematic engineering support for facilitating the movement from the old to new architecture. The authors have focused on supporting performance and scalability predictions.

They have proposed a formal process. First, existing systems have to be reverse-engineered to obtain a performance prediction model. These models contain both static as well as dynamic aspects such as contributing components and their interactions. Second, the software architect has to select a set of potential target architecture styles or patterns, which have to be appropriately formalized. For example, the architect plans to evaluate the impact of the classical system architecture movement to MapReduce or to the SPOSAD style, and, thus, he/she automatically adapts the reverse-engineered performance prediction models by the selected architectural styles.

Third, the performance of the target architectures is evaluated to get a final ranking and to come to a recommendation for the migration. Finally, based on the analyzed target architecture, the system's implementation has to be adapted. The major foundations for the sketched process are already in place (software architectural patterns, software performance engineering, architecture evolution, and model transformations).

2.8 Mechanisms for Assisting Cloud Software Development

Today, not only development processes but also environments have to be very agile [10] and anticipative as software development becomes more sophisticated. Cloud-induced agile IT environments are being presented as the viable and valuable

resources for new-generation software construction. The unique capabilities of clouds are being succinctly indicated below:

- On-demand provisioning and de-provisioning of resources in minutes through a self-service access.
- Non-function requirements of servers, storages, and network products are being ensured.
- Implicit support for virtual development environments and multi-tier application architectures.
- Easier migration of the existing virtual-server images and workloads into the cloud.

Clouds can accelerate the development cycle by creating multiple development environments that enable several software activities to be carried out simultaneously. Testing can be accomplished along with development. The unique on-demand resource provisioning capability of clouds makes this parallelization possible. Cloud supports different levels of quality of service (QoS). Developers could choose the appropriate QoS level as per the applications. This means that a higher level of performance, security, and availability needs to be assigned to a development environment for performance and scalability testing. In exchange, the hourly cost of such environment goes up. The QA process will also benefit from on-demand up and down scaling of cloud resources, as this finally solves the problem of testing performance and scalability of applications at a large scale, but without indefinitely reserving and paying for resources when they are unused.

Cloud virtual machines (VMs) support multi-tier application development and testing. That is, presentation tier, business logic tier, and data tier are being deployed in different VMs. When the development in a virtual cloud environment is finished, the images of virtual servers can be easily transferred to the production environment.

The advantage is to avoid problems related to configuring a new application for transfer from the development to the production environment, which again affects the speed of the application time to market.

The Lean Thinking Principles for Cloud Software Development – There are lean approaches and principles being sincerely and seriously examined and expounded by professionals and pundits for optimally implementing a variety of industrial systems. Software engineers are also vigorously following the same line of thinking for producing high-quality software solutions for a variety of business and societal problems. The core elements of the lean principle are "eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, respect people and optimize the whole." This set of well-intended tasks definitely creates a sound case for contemporary cloud enterprises. As corporates are planning and assimilating cloud technologies as a part of their business transformation initiative, there are other mandatory things to be accomplished in parallel in order to reap the envisioned advantages.

Here is what a few software companies have achieved by applying lean principles to their development process [11]:

- *Salesforce.com* has improved time to market of major software releases by 61 % and boosted productivity across their R&D organization by 38 % since adopting agile development.

- *BT Adastral,* the largest telecommunications company in the UK, completed its first major lean software project 50 % sooner than expected and incorporated many product changes along the way. The product yielded 80 % ROI in the first year.
- *PatientKeeper,* specializing in software for the healthcare industry, puts out weekly maintenance releases, monthly new feature releases, and quarterly new application releases. This company completes 45 development cycles in the time it takes their competitors to do 1 cycle.
- *Timberline Software* (now part of The Sage Group), serving the construction and real estate market, estimates that improvements in quality, costs, and time to market were all greater than 25 % as a result of switching to lean software development.

Lean thinking is important for scaling agile in several ways [12]:

- Lean provides an explanation for why many of the agile practices work. For example, Agile Modeling's practices of lightweight, initial requirements envisioning followed by iteration modeling and just-in-time (JIT) model storming work because they reflect deferment of commitment regarding what needs to be built until it is actually needed, and the practices help eliminate waste because we are only modeling what needs to be built.
- Lean offers insight into strategies for improving our software process. For example, by understanding the source of waste in IT, we can begin to identify it and then eliminate it.
- Lean principles provide a philosophical foundation for scaling agile approaches.
- It provides techniques for identifying waste. Value stream mapping, a technique common within the lean community, whereby we model a process and then identify how much time is spent on value-added work versus wait time, helps calculate overall time efficiency of what we are doing. Value stream maps are a straightforward way to illuminate our IT processes, providing insight into where significant problems exist.

The lean manufacturing with its emphasis on eliminating waste and empowering employees shook up the automotive industry. Lean principles are revolutionizing software development industry as well. Lean developers can build software faster, better, and cheaper than competitors using traditional bulky and bulging methods. By adopting agile practices and test-driven development, a software firm can go a long way toward leaning out its operations and serving its customers better.

Lean Agile Methodologies Accentuate Benefits of Cloud Computing [13] – Lean and agile are two different production methodologies that are used extensively in business. The lean approach is derived from the production processes adopted by Toyota, Japan. It focuses on a demand-driven approach with an emphasis on:

- Building only what is needed
- Eliminating anything that does not add value
- Stopping production if something goes wrong

The agile approach is focused on the notion that software should be developed in small iterations with frequent releases, because neither the end-user requirements

nor the exact amount of efforts can be accurately finalized upfront. Even the end users themselves cannot fully articulate what they need. Hence, the requirements must be collaboratively discovered, analyzed, and finalized. Agile processes [14] involve building software in small segments, testing those segments, and then getting end-user feedback. The aim is to create a rapid feedback loop between the developers and the actual users.

Lean agile development methodologies and the cloud model complement each other very well. Cloud services take pride in meeting user requirements rapidly, delivering applications whenever and to whatever extent they are needed. Agile methods give high credence to user collaboration in requirements discovery. The lean agile system of software development aims to break down project requirements into small and achievable segments. This approach guarantees user feedback on every task of the project. Segments can be planned, developed, and tested individually to maintain high-quality standards without any major bottlenecks. The development stage of every component thus becomes a single "iteration" process. Moreover, lean agile software methods place huge emphasis on developing a collaborative relationship between application developers and end users. The entire development process is transparent to the end user and feedback is sought at all stages of development, and the needy changes are made accordingly then and there.

Using lean agile development in conjunction with the cloud paradigm provides a highly interactive and collaborative environment. The moment developers finalize a feature, they can push it as a cloud service; users can review it instantly and provide valuable feedback. Thus, a lengthy feedback cycle can be eliminated thereby reducing the probability of misstated or misunderstood requirements. This considerably curtails the time and efforts for the software development organization while increasing end-user satisfaction. Following the lean agile approach of demand-driven production, end users' needs are integrated in a more cohesive and efficient manner with software delivery as cloud services. This approach stimulates and sustains a good amount of innovation, requirement discovery, and validation in cloud computing.

2.9 Cloud Platform Solutions for Software Engineering

Compared to on-premise applications, cloud-based software as a service (SaaS) application are delivered through the Web, billed on a subscription basis, and service providers themselves are responsible for delivering the application at acceptable service levels. As a consequence, the economics of delivering SaaS is different from traditional software applications. Companies delivering SaaS/Cloud applications need to realize economies of scale and keep the application delivery costs low. These issues have a significant impact on how SaaS applications are architected, developed, and delivered. For the paradigm of SaaS to succeed, issues like application scalability, cost of delivery, and application availability had to be resolved comprehensively. A new set of architectural, development, and delivery principles have emerged and strengthened the spread of the SaaS model.

In order to achieve the acceptable levels of maturity, companies need to address issues in three core areas [15]:

- They need to build applications that support a multitenant architecture that enables a single instance of the application to be shared among multiple customers. Multitenancy has a significant impact on all layers of the application stack and is challenging to achieve. This architectural principle is a significant contributing factor in reducing application delivery costs.
- SaaS vendors need to address a significant number of non-functional application concerns that are essential for the success of the service. For example, traditional software vendors were not concerned with issues like metadata management, tenant customization and configuration, scalability, fault tolerance to meet SLAs, metering, monitoring, robust security in distributed environments, and a host of other concerns.
- As applications grow and scale, companies need to address automation of operations and application management. Automation of operations and application management is among the primary contributing factors in reducing application delivery costs. Despite emerging automation in areas like the infrastructure cloud, 75–80 % of the issues arising in operations are best solved at the application design and development level. Furthermore, it is difficult and expensive to achieve operational and administrative automation once the service is designed and developed. SaaS providers can achieve significant benefits if application architecture takes automation of operations into account early in the application life cycle.

The cloud idea is everywhere and engineers, executives, exponents, and evangelists are trying different ways and means of adopting and adapting the cloud concepts as per their organizational needs. Data centers are being pruned and tuned to be cloud centers, traditional applications are getting modernized and migrated to local as well as remote cloud environments, centralized delivery and management of IT resources are being insisted and illustrated, innovative and disruptive ideas get quickly concretized by renting needed compute and storage servers from public cloud providers, server systems exclusively for backup and disaster recovery to guarantee business continuity are being subscribed out of cost-effective cloud servers, all kinds of customer-centric applications such as collaboration software are unhesitatingly moved to cloud systems in order to reap their distinct advantages (technical as well as business), etc. In the recent past, cloud is being prescribed as the most productive solution for software coding and testing. That is, platform as a service (PaaS), which has been dormant and dumb for quite a long time, gets a fresh life with the realization across the globe that cloud-based platforms are much more effective, simpler, and quicker for software building.

How Azure Helps Cloud Software Development to Be Agile? – Microsoft Azure is an application platform on the cloud that provides a wide range of core infrastructure services such as compute and storage along with building blocks that can be consumed for developing high-quality business applications. Azure provides platform as a service (PaaS) capabilities for assisting application development, hosting,

execution, and management within Microsoft cloud centers. Windows Azure is an open cloud platform that enables to quickly build applications in any language, tool, or framework. The advantages of Azure cloud are:

- Azure provides staging and production environments on the cloud which provide resource elasticity on demand, and this agility factor helps a lot for any Windows application development team.
- Only the development and unit testing is carried out on-premise systems.
- Cloud staging environment can be used to create different test environments on cloud such as integration, system, and UAT.
- Application source code can be maintained in Azure cloud storage.
- Developers test their application with a production-like environment as setting up a real production environment for testing involves more investment, planning, time, and resources. That is, all kinds of infrastructure-intensive software testing can be accomplished in Azure cloud with high dependability cost-effectively due to the inherent elastic nature of Azure. This enables application providers to ensure the SLA to their customers and consumers.
- A couple of integrated development environments such as Visual Studio.NET are provided by Microsoft in order to simplify and speed up cloud application development activities.
- Source code can be promoted from one environment to another rather seamlessly without developers having to write verbose deployment scripts or instruction manuals to set up the application in the target environments.

How Azure Helps Software Delivery to Be Agile? – Delivery is also facilitated by Azure cloud. By providing flexible infrastructures just in time, cloud software delivery is made agile. All kinds of fluctuations of infrastructure needs are being automatically taken care of Azure cloud. All kinds of plumping works are being delegated to cloud center experts so that designers, developers, and testers can focus on their core activities.

As Visual Studio IDE is tightly integrated with the cloud environment, application development and deployment happen faster and are hugely simplified. The cloud provides all the libraries and APIs upfront in order to lessen the developmental cost and complexity. Further on, in the Azure cloud, deployment and upgrade processes are completely automated to minimize or eliminate some of the lengthy and tedious steps while planning and executing the traditionally accomplished deployment process. Working prototypes built by geographically dispersed developers and centrally deployed in Azure can be made available and accessible immediately to prospective customers in order to elicit and extract their feelings and feedbacks as this arrangement sharply reduces time especially for contemplating any major or minor corrections to take the products to market quickly.

The Alice Platform [15] – In order to help companies with the challenges of building and delivering successful SaaS services, the authors have developed the first open SaaS platform called Alice. As a company focused on developing cloud-based SaaS services, it became quite evident that traditional JEE, .NET, and Ruby on Rails platforms were not designed to address base level architectural

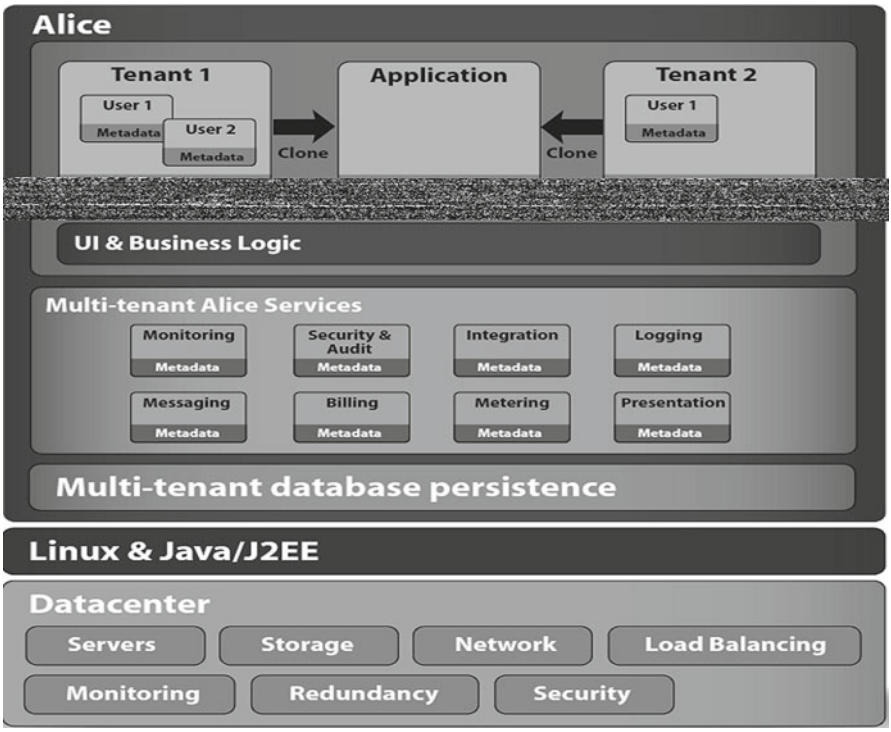

Fig. 2.1 The architectural diagram of the Alice platform

concerns of large and scalable SaaS applications. While building applications for our clients, developers had to address multitenancy, data management, security, scalability, caching, and many other features. Many of the most successful SaaS companies had themselves built their own platforms and frameworks to address their specific applications and cost needs. Companies like Salesforce and NetSuite, first and foremost, built platforms to meet their application needs and lower delivery costs, rather than building them to be sold as a platform as a service (PaaS).

Release of SaaS application platforms by companies like Salesforce has not made a significant difference in the development and delivery of commercial SaaS applications. Currently, many PaaS/SaaS platforms on the market are suitable for development of only small situational applications, rather than commercial business applications that are of interest to startups, independent software vendors (ISVs), and enterprises. These platforms use proprietary languages, are tied to a specific hardware/software infrastructures, and do not provide the right abstractions for developers. Alice was developed to address the above concerns and provide a robust and open platform for the rapid development of scalable cloud services applications. Figure 2.1 illustrates the reference architecture of the Alice Platform for SaaS application development and delivery.

2.10 Software Engineering Challenges in Cloud Environments

With the coherent participation of cloud service providers, the software development complexity is to climb further [3]. In the ensuing cloud era, software development process will start to involve heterogeneous platforms, distributed services, and multiple enterprises geographically dispersed all over the world. Existing software process models are simply insufficient unless the remote interaction with cloud providers is a part and parcel of the whole process. Requirements gathering phase so far included customers, end users, and software engineers. Now it has to include cloud service providers (CSPs) as well, as they will be supplying the computing infrastructure, software development, management, maintenance platforms, etc. As the cloud providers are only conversant with the infrastructure utilization details, their experts can do the capacity planning, risk management, configuration management, quality assurance, etc., well. Similarly, analysis and design activities should also include CSPs, who can chip in with some decision-enabling details such as software-development cost, schedule, resource, and time.

Development and debugging can be done on cloud platforms. There is a huge cost benefit for individuals, innovators, and institutions. This will reduce the cost and time for verification and validation. Software developers should have gained more right and relevant expertise in building software from readily available components than writing them from the scratch. The monolithic applications have been shunted out and modular application has the future. Revisiting and refactoring of existing application is required to best utilize the cloud paradigm in a cost-effective manner. In the recent past, computers are fit with multicore processors. Another trend is computers are interconnected as well as with the Web. Computers are becoming communicators and vice versa. Computers are multifaceted, networked, and shrinking in size, whereas the scope of computing is growing. Therefore, software engineers should train themselves in parallel and distributed computing to complement the unprecedented and inescapable advances in hardware and networking. Software engineers should train themselves in Web protocols, XML, service orientation, etc. Web is on the growing trajectory as it started with a simple Web (Web 1.0). Today it is the social Web (Web 2.0) and semantic Web (Web 3.0) attracting the attention of professionals as well as people. Tomorrow definitely it will be the smart Web (Web 4.0). The cloud proposition is on the fast track and thereby there will be a scintillating synchronization between the enlarging Web concepts and the cloud idea.

Cloud providers also have the appropriate infrastructure and methods in hand in order for application maintenance [14]. There is a service-level agreement (SLA) being established as a contract between cloud users (in this case, software engineers) and cloud providers. Especially the advanced cloud infrastructure ensures non-functional (scalability, availability, security, sustainability, etc.) requirements. Other serious challenges confronting the cloud-based software development include the following. As we see, the development of software is multilateral in a cloud environment unlike the collocated and conventional application software development.

The difference between these two radical approaches presents some of the noticeable challenges to software engineering:

- *Software Composition* – Traditionally, application software engineers develop a set of coherent and cohesive modules and assemble them to form an application, whereas in the fast-expanding cloud landscape, finding and composing third-party software components is a real challenge.
- *Query-Oriented Versus API-Oriented Programming* – MapReduce, streaming, and complex event processing require developers to adopt a more functional query-oriented style of processing to derive information. Rather than a large surface area of OO APIs, these systems use an extension of SQL-like operations where clients pass in application specific functions which are executed against associated data sources. Doing complex join queries or function composition such as MapReduce is a difficult proposition.
- *Availability of Source Code* – In the current scene, full source of the code is available. However, in the multilateral software development, there is no source code available because of third-party components. Therefore, the challenge for software engineers is the complete comprehension of the system.
- *Execution Model* – The application software developed generally is executed on single machine, whereas the multilateral software developed for cloud environment is often distributed between multiple machines. Therefore, the challenge for software engineers is the traceability of state of executing entity and debugging.
- *Application Management* – The challenges are there as usual when there is an attempt to embrace newer technologies. Application lifecycle management (ALM) is quiet straightforward in the traditional setting, whereas globally, collaborative and cloud-based application management is beset with definite concerns and challenges.

The need of the hour to make the cloud concepts more beneficial to all sections of the world is to activate the innovation culture; thereby, a stream of inventive approaches can be unearthed to reinvigorate the sagging and struggling software engineering domain. Here is one. Radha Guha [2] has come out with an improved cost estimation model for the cloud-based software development.

2.11 Conclusion

Nowadays, for most business systems, software is a key enabler of their business processes. The software availability and stability directly impact the company's revenue and customer satisfaction. Software development is therefore a critical activity. Software development is undergoing a series of key changes. A growing number of independent software vendors (ISVs) and system integrators (SIs) transform themselves into service providers delivering their customers' and partners' applications in the form of services hosted in the cloud.

The cloud technology could reduce the time needed for the development of business services and to take them to the market. Each additional month or quarter in which the cloud services are accessible to users has a direct impact on increasing revenues, which affects the final financial statements. The speed at which software applications can be developed, tested, and brought into production is definitely one of the critical success factors for many companies. Therefore, any solution accelerating the application time to market has an immediate and measurable impact on return on investment (ROI).

Application developers are regularly confronted with a request to establish special environments for developing, debugging, and compiling appropriate software libraries for making software solutions. Typically, these environments are established for a limited period of time. Accessing appropriately configured development environments with an adequate processing power and storage space on demand is very crucial for software engineering. To perform their tasks, the programmers should be able to quickly configure servers, storage, and network connections. Here comes the significance of cloud environments for taking software to market quickly. In this chapter, we primarily discussed the pathbreaking contributions of cloud infrastructures for realizing sophisticated and smart services and applications.

References

1. Yara, P., Ramachandran, R., Balasubramanian, G., Muthuswamy, K., Chandrasekar, D.: Global software development with cloud platforms. In: Software Engineering Approaches for Offshore and Outsourced Development. Lecture Notes in Business Information Processing. http://link.springer.com/chapter/10.1007/978-3-642-02987-5_10. vol. 35, pp. 81–95 (2009)
2. Guha, R.: Software engineering on semantic web and cloud computing platform. http://www.cs.pitt.edu/~chang/231/y11/papers/cloudSE (2011). Accessed 24 Oct 2012
3. Chhabra, B., Verma, D., Taneja, B.: Software engineering issues from the cloud application perspective. Int. J. Inf. Technol. Knowl. Manage. **2**(2), 669–673 (2010)
4. Kuusela, R., Huomo, T., Korkala, M.: Lean Thinking Principles for Cloud Software Development. VTT www.vtt.fi. A Research Summary of VTT Technical Research Centre of Finland (2010)
5. Hashmi, S.I.: Using the cloud to facilitate global software development challenges. In: Sixth IEEE International Conference on Global Software Engineering Workshops, 15–18 Aug 2011, pp. 70–77. IEEE XPlore Digital Library, IEEE, Piscataway (2011)
6. Tamburri, D.A., Lago, P.: Satisfying cloud computing requirements with agile service networks. In: IEEE World Congress on Services, 4–9 July 2011, pp. 501–506. IEEE XPlore Digital Library, IEEE, Los Alamitos (2011)
7. Carroll, N., et al.: The discovery of agile service networks through the use of social network analysis. In: International Conference on Service Sciences. IEEE Computer Society, IEEE, Washington, DC (2010)
8. Bruneli'ere, H., Cabot, J., Jouault, F.: Combining model-driven engineering and cloud computing. http://jordicabot.com/papers/MDE4Service10.pdf (2010). Accessed 24 Oct 2012
9. Becker, S., Tichy, M.: Towards model-driven evolution of performance critical business information systems to cloud computing architectures. In: MMSM. http://www.cse.chalmers.se/~tichy/2012/MMSM2012.pdf (2012). Accessed 24 Oct 2012

10. Dumbre, A., Senthil, S.P., Ghag, S.S.: Practicing Agile Software Development on the Windows Azure Platform. White paper by Infosys Ltd., Bangalore. http://www.infosys.com/cloud/resource-center/documents/practicing-agile-software-development.pdf (2011) Accessed 24 Oct 2012
11. Lean Software Development – Cutting Fat Out of Your Diet. A White Paper by Architech solutions. http://www.architech.ca/wp-content/uploads/2010/07/Lean-Software-Development-Cutting-Fat-Out-of-Your-Diet.pdf. Accessed 24 Oct 2012
12. Tripathi, N.: Practices of lean software development. http://cswf.wikispaces.com/file/view/Practices+in+Lean+Software+Development.pdf (2011). Accessed 24 Oct 2012
13. Talreja, Y.: Lean Agile methodologies accentuate benefits of cloud computing. http://www.the-technology-gurus.com/yahoo_site_admin/assets/docs/LACC_white_paper_ed_v5.320180428.pdf (2010). Accessed 24 Oct 2012
14. Das, D., Vaidya, K.: An Agile Process Framework for Cloud Application. A White Paper by CSC. http://assets1.csc.com/lef/downloads/CSC_Papers_2011_Agile_Process_Framework.pdf (2011). Accessed 24 Oct 2012
15. Alice Software as a Service(SaaS) Delivery Platform. A Whitepaper by Ekartha, Inc. http://www.ekartha.com/resources/Alice_saas_delivery_platform.pdf. Accessed 24 Oct 2012