Chapter 5
Software Project Management as a Service (SPMaaS): Perspectives and Benefits

Muthu Ramachandran and Vikrant Chaugule

5.1 Introduction

Cloud computing has evolved to address the availability of computing resources which can be accessed from anywhere and anytime. In particular, computing hardware and software often gets outdated, and hence, it is wise to outsource computing resources and to manage their IT infrastructures outside of their company premises, which is more cost-effective than is the case at present. Applications can be leased (like pas-as-you-go service) rather than being purchased, and companies have increased their data centers due to demand (Amazon, Microsoft, and IBM) [1]. Cloud computing is heavily based on "software as a service" concept and needs high-speed web access. It provides services on demand utilizing resources more effectively within the cloud environment. The cloud architecture, its layers, and its composition of components and services need to be designed for scalability, security, and re-configurability as they support services and its agreements (e.g., service level agreements). In this scenario, the resource management of cloud computing is the key to achieving potential benefits.

Cloud computing, one of the greatest developments in the field of computing, has the ability to transform and change the work of an IT industry. It has definitely helped in making the way software can be offered more attractively and also changing the way hardware is purchased and designed. It has led to a complete

M. Ramachandran (✉)
School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Leeds LS6 3QS, UK
e-mail: M.Ramachandran@leedsbeckett.ac.uk

V. Chaugule
Department of Computer Science and Engineering, National Institute of Technology, Surathkal, Karnataka 575025, India
e-mail: vikrant.chaugule@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_5

change especially that developers coming up with new Internet services need not require a large investment in hardware or the human resources to operate the hardware. The developers need not waste costly resources and face losses in case the product fails, and on the other hand, they do not need to worry about scalability if the idea turns out to be successful and popular. This versatility about resources, without paying a premium for vast scale, is phenomenal for the IT industry.

Cloud computing consists of both applications provided as services over the Internet and the hardware and systems software which provide such services in the data centers. The services themselves have long been referred to as software as a service (SaaS) [2]. Together, the hardware and software at the datacenter comprises the cloud. When this is made available in the form of a pay-as-you-go fashion, to the people, it is called a public cloud, whereas utility computing is the service being sold. If the cloud is owned privately by an organization only for storing their information and is not made available to the public, it is called a private cloud. Thus, the addition of SaaS and utility computing is called cloud computing. People can be users or providers of utility computing, or users or providers of SaaS. We would like to focus on SaaS project management with the help of improving the way services are provided such that it is more convenient for users to use and benefit from them.

Some of the benefits of using cloud computing are [3]:

- It leads to lowering of project costs. Since the model used for billing is pay as per usage, maintenance is reduced since infrastructure required is not purchased.
- A massive infrastructure is provided by all the cloud service providers, and therefore, managing large volumes of data has become a reality. The cloud can be scaled dynamically, and sudden workload spikes can be handled very efficiently.
- It is very flexible. With enterprises having to adapt and adjust very rapidly, delivery speed becomes very critical. Hence, more emphasis is given on getting applications to market very quickly.

With the emergence of cloud computing, the focus moves to the interface, that is, interface between the service consumers and service providers. Some areas like distributed services, risk assessment, procurement, and service negotiation will demand expertise from enterprises, but most of them are only modestly equipped to take care of them.

Cloud computing is based on web access; therefore, we need to design web applications which are designed for security. Hence, it is essential to design cloud applications as web service components based on well-proven software process, design methods, and techniques such as component-based software engineering (CBSE). Wand and Laszewski [4] define cloud computing as a set of network-enabled services which provides scalable, guaranteed QoS (quality of service), inexpensive computing platforms on demand, customizable (personalized), and all of which can be accessed in a simple and pervasive way. An overview of the different cloud computing paradigms is discussed and presented with definitions, business models, and technologies by Wand and Laszewski [4] and by many others.

Software components provide a good design rationale supporting various requirements of application developments, design flexibility, system composition, testability, reusability, and other design characteristics. Component-based designs are customizable, and interfaces can be designed supporting SLA (service level agreement). SLAs vary between service providers which need to be customized without much effort. This can only be achieved using a component which has been designed for flexible interface that links to a number of SLAs. Each SLAs and business rules can be represented as a set of interfaces that can be mapped onto knowledge-based database or a data server. This also allows the reuse of SLAs for any individual service providers. Some of the important characteristics of the cloud computing mentioned are:

- On-demand service
- Handling multi tenancy service requirements
- Resource grouping
- Efficient elasticity
- Measurable service delivery

Our earlier work described by Ramachandran [5] on component model for web services and service-oriented architecture (SOA), grid computing, and various other systems can become an integrated aspect of any cloud computing architectures and application design. We also need to understand the basic differences among SOA (service-oriented architecture), grid, and cloud computing. SOA is to offer services which are based on open standard Internet services and virtualization technology and have been running in a different environment, *grid* offers services from multiple environments and virtualization, and *cloud* combines both. We also need to identify a specific development process for capturing requirements, design and implementation strategies, security, and testing cloud applications. Cloud computing paradigm has lots to offer, but at the same time, we need to consider building a secured and resilient architecture and services that are reliable and trustworthy. In this chapter, a generic component model and a web service component model have been developed meeting the design demands for building cloud application architectures. In this research, we have also proposed architectural composition strategies which can be customized for various cloud services.

This chapter presents our work on software development process model for building cloud services as it is necessary to follow a systematic approach. The organization is as follows. Section 5.2 gives a detailed explanation on the software development process for Cloud computing, Sect. 5.3 talks about the service development process, Sect. 5.4 compares classical and cloud-based project management tools, Sect. 5.5 provides a critical evaluation of some existing project management tools, Sect. 5.6 discusses the integrated software development process, and Sect. 5.7 gives a detailed explanation on the service-oriented architecture. Conclusions are summarized in Sect. 5.8.

5.2 Software Development Process for Cloud Computing

In order to define a process model, it is useful to capture some of our thoughts on understanding the very nature of cloud characteristics and its type of services that aims to provide. Identifying characteristics of a service-oriented system is vital for designers such that they can select, design, and evaluate those characteristics that are applicable to their applications. Service-oriented computing (SoC) [6] involves integration of several disciplines and subject areas, and therefore, some of the characteristics will overlap. Some of the identified services and component characteristics are:

- Reusable web services and some other core services
- Enterprise integration services
- Dynamic binding and reconfigurable at run-time
- Granularity
- Publish, subscribe, and discover
- Open world where components must be able to connect and plug to third party software systems or components.
- Heterogeneity supporting cross-platform applications
- Reconfigurable
- Self-composable and self-recoverable
- Cloud infrastructure and resources management
- Autonomic framework
- Middleware
- QoS

This is illustrated in Fig. 5.1, which shows some of the above characteristics that are the key to developing software components. In the modern software development, characteristics such as open world where components can be customizable and connectable to third party systems and their components and heterogeneity are crucial to developing highly reusable web services that will apply across domains and services.

The main reason for presenting such characteristics is to understand the basis for service-oriented systems and hence providing good practice design guidelines. The next section looks at the distinct features and differences between services and components. Again these characteristics need to overlap as we are also interested in applying component-based development for service-oriented systems. In particular web services need to possess both services and component characteristics. After looking at service-oriented computing and the characteristics of SaaS systems in this section, we shall look at the service-oriented development process in detail in the next section.

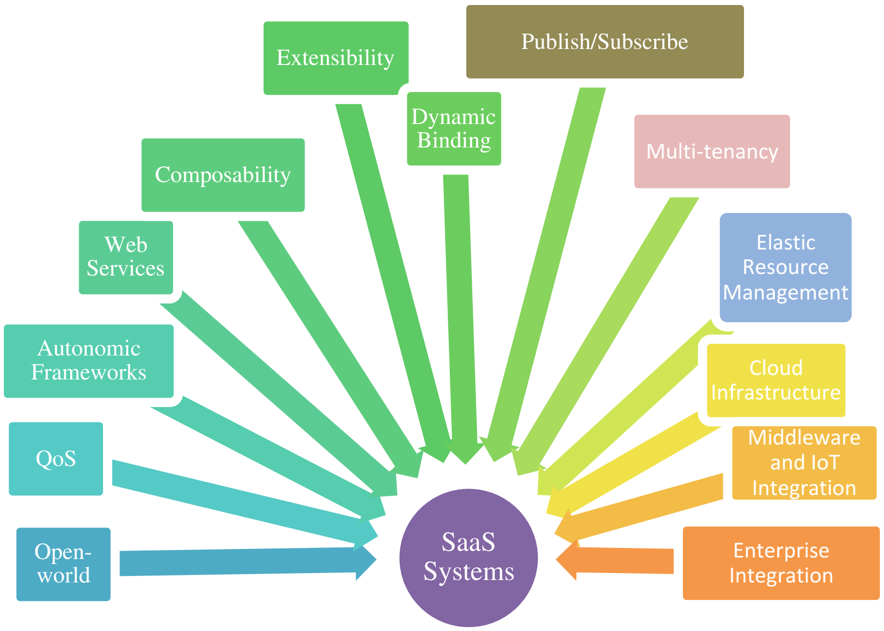


Fig. 5.1 Characteristics of software as a service systems (SaaS)

5.3 Service Development Process

The identification of service requirements [6] needs a new RE process and modelling techniques as it is highly dependent on multilevel enterprises across corporation. Identifying and knowing all requirements for all expected and even unexpected services is very hard. The idea in software as a service is to publish automatically new services whereby service agents can then be able to request and take advantage of required services for their customers. Figure 5.2 shows a development process model for service-oriented computing where initial requirements are captured based on enterprise-wide techniques and perhaps using domain analysis which should focus on a family of products and services. The second phase (RE services) involves identifying a set of requirements of system services. This process involves service modelling and service specification for which we can use any well-known techniques such as use case design and a template for software as a service (SaaS).

The third phase (categorizing services) involves classifying and distinguishing services into various categories such as enterprise integration services (services across corporations, departments, other business services); software services which represents core functionality of software systems; business logic services which represents business rules and its constraints; web services (a self-contained and web-enabled entity which provides services across businesses and customizable at

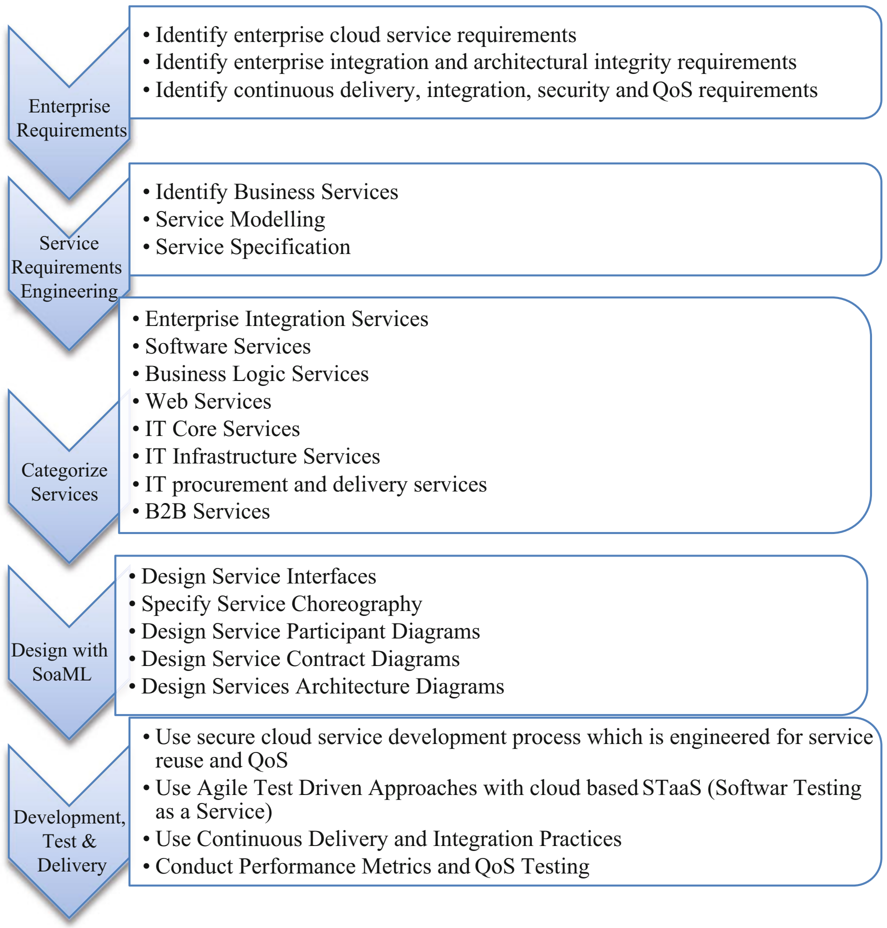

Enterprise Requirements
• Identify enterprise cloud service requirements
• Identify enterprise integration and architectural integrity requirements
• Identify continuous delivery, integration, security and QoS requirements
Service Requirements Engineering
• Identify Business Services
• Service Modelling
• Service Specification
Categorize Services
• Enterprise Integration Services
• Software Services
• Business Logic Services
• Web Services
• IT Core Services
• IT Infrastructure Services
• IT procurement and delivery services
• B2B Services
Design with SoaML
• Design Service Interfaces
• Specify Service Choreography
• Design Service Participant Diagrams
• Design Service Contract Diagrams
• Design Services Architecture Diagrams
Development, Test & Delivery
• Use secure cloud service development process which is engineered for service reuse and QoS
• Use Agile Test Driven Approaches with cloud based STaaS (Softwar Testing as a Service)
• Use Continuous Delivery and Integration Practices
• Conduct Performance Metrics and QoS Testing

Fig. 5.2 Service-oriented software development process

run-time); IT core services which include resource management, help desk systems, IT infrastructure, and procurement; and delivery services, B2B and B2C services, data services, QoS services, middleware services, transaction management services, process integration services, re-configurability services, and grid services which include grid resource management and re-configurations. Service design stage has been proposed with designing services using OMG standard design notation known as SoaML which consists of a five-stage design:

- Design Service Interfaces: This offers services to other services through well-designed interfaces (the value provided), it allows design for service reuse, and it allows modelling of service specification.

- Specify service choreography which defines the interaction between the provider and consumer in completing a service, and this can be modelled using UML sequence diagrams.
- Design Service Participant Diagrams: The concept of a participant, in SoaML, represents certain party or component that provides a transaction through its interface to a consume service(s). Participants can be software components, organizations, system, or individuals. The participant design should be designed with SoaML participant diagrams which are similar to a UML component service with provider and require interfaces designed through the concept of a port. Service participant diagram allows for modeling primarily the participants that play role(s) in services architectures. It also presents the services provided and used by these participants.
- Design Service Contract Diagrams: As we have discussed, there are three approaches to specify a service: the above two interface-based approaches – simple interface and service interface – and, thirdly, through a service contract. Service contract defines the agreement between parties about how a service is to be provided and consumed. "Agreement" here refers to interfaces, choreography and any terms and conditions. Interacting participants MUST agree to the agreement in order for the service to be enacted. In SoaML, this is designed using service contract diagrams.
- Design Services Architecture Diagrams: This stage of the design offers features to express the complete list of services and their interactions. A service-oriented architecture, abbreviated as SOA, shows the participant roles that provide and consume services to fulfill certain purpose. In SoaML, this is represented as large globe with all interacting services connected.

We discuss in detail, in the last section of this chapter, SoaML design for SPMaaS. Based the above finding, we can propose a new paradigm for cloud application engineering as shown in Fig. 5.3. This illustration provides a relevant link to classical software engineering process.

As shown in Fig. 5.3, the requirements phase is linked to identifying cloud requirements which should in particular identify service requirements and relevant software security requirements so that cloud services are built with security rather than adding security batches after release. The design phase is linked to designing services for cloud environment and reuse as services are loosely coupled and have high potential for reuse. The code/implementation phase is linked to service development. Likewise testing and QA are related to cloud testing strategies and quality engineering. The next section discusses classical software project management activities and the need of cloud-based project management tools and highlights the advantages of cloud-based project management tools [11–17].

With increase in the use of service-oriented architecture, software projects and systems can get very complex. With the aim of managing this complexity, a number of SDLC models have been used:

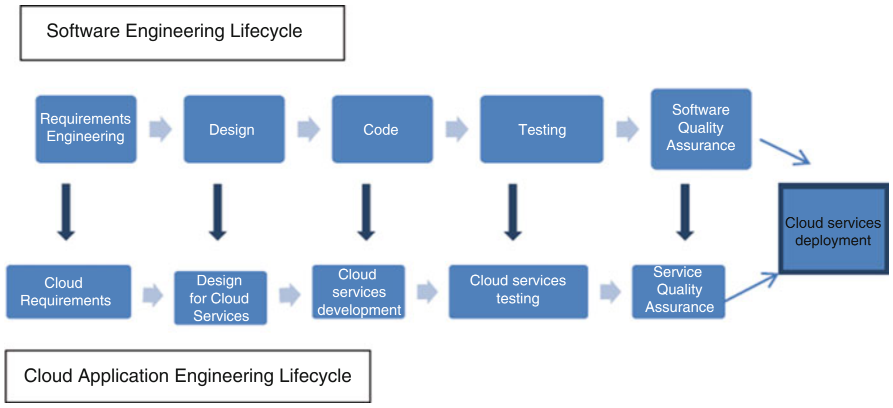

Software Engineering Lifecycle
Requirements Engineering
Design
Code
Testing
Software Quality Assurance
Cloud services deployment
Cloud Requirements
Design for Cloud Services
Cloud services development
Cloud services testing
Service Quality Assurance
Cloud Application Engineering Lifecycle

Fig. 5.3 Software engineering vs. cloud application engineering lifecycle

- Waterfall model
- Spiral model
- Rapid prototyping
- Agile method
- Incremental
- Synchronize and stabilize

The above mentioned models have been in existence for many years, and each of them has their own advantages. For instance, the waterfall model is simple to understand, use, and manage. High amount of risk analysis is done with the spiral model, and hence avoidance of risk is enhanced. The agile method allows regular adaptation to changing circumstances, and even late changes in requirements are welcomed. The incremental model provides benefits like easier testing and debugging during smaller iterations and also lowers initial delivery cost. Though these models have their own advantages, there are many issues which exist:

- Fulfilling compatibility criteria of numerous services from various vendors
- Sufficient bout of resources and accordingly manage them
- Lack of required coordination between client and provider to provide what was required
- Managing responsiveness and streamline changes as requested
- Deviation from anticipated product
- Difficult to come back to initial stages in case they had not turned out as expected

Understanding and keeping all of the above issues in mind, mainly relating to the vendor-based SDLC, the cloud-based service would typically offer the following:

- Requirements Engineering as a cloud service (REaaS) menas it supports reuse of requirements, traceability, and commonality and variability analysis for a familay of related systems. At this pahse of any project management activities

using REaaS can also support contractual regulations, project initiion, acceptance testing and planning.
- Product design based on the collected requirements and documented artifacts.
- Implementation through cloud service based on customer chosen environment.
- Un-optimized and deviation from the required product needs to be checked, and hence testing is performed.
- Option of upgrading and updating the services to incorporate changing and dynamic requirements of the client.

Similar work done previously in this area can be seen in paper [7] where authors have focused on a cloud-based management of projects through the means of software as a service and its various augmentable utilities. They have proposed a model on the cloud which provides SDLC phases as coordinated services. In this proposed model, services will be able to interact with one another and either providers or consumers of data and behavior, instead of letting the client collect all the data and put it altogether after gathering from numerous vendors. A technique and approach for the implementation of the model is also stated in detail. The internal working of the model makes use of two services – IaaS and SaaS. SaaS (software as a service) borrows services and resources from IaaS (Infrastructure as a Service) providers and in turn leases those services to the users. This maximizes resource utilization and also results in increasing customer satisfaction level (CSL). The SaaS contains two vital layers, namely, platform layer and application layer. While the platform layer would be responsible for the admission control depending on how many projects are already admitted, scheduling process, etc., the application layer is required to assemble the service from IaaS and integrate the resources with it to perform the job which conventionally is done by a third party system. The model proposed by them can be implemented not only for small term developer level projects but also higher level project management for which the number of resources to be utilized is a long-term and non-ephemeral function of usage and maintenance.

5.4 Classical vs. Cloud-Based Software Project Management

As management is said to both science and the art, so is project management. The careful process of bringing together economics, software technology, and human relations for a software project is not a simple task. A software project is an extremely people-intensive exertion that traverses an exceptionally long period, with crucial ramifications on the work and execution of a wide range of classes of individuals.

A software project can be regarded as the assembling of tasks that create an identifiable and valuable outcome. In its fundamental form, project management [8–9] consists of planning, executing, and monitoring these activities (Fig. 5.4). Notwithstanding, the high expenses and failure rates of software projects keep on

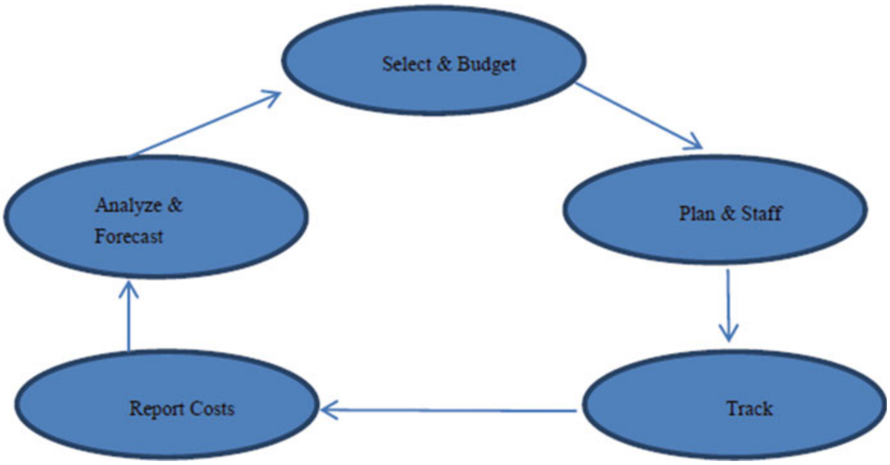


Fig. 5.4 Project management lifecycle

engaging analysts and specialists, and regardless of a few advances, the successful administration of the project is still a challenging process. Dealing with the one of a kind and complex procedures that constitutes a task includes the execution of particular administration exercises. In programming improvement, as in most different organizations, there has been an inclination toward institutionalizing these exercises by method for formalized, nonexclusive project management methodologies like PRINCE2 [8], which was created and championed by the UK government. In spite of the fact that there is a worldwide origination of the project management marvel, there is no brought together hypothesis of project management or very much characterized measure of project success. It is beyond the capabilities of project teams of large software projects to decide the technological, environmental, and organizational states which might have an influence on the outcome of the desired product. Another challenge faced is that the information required to extrapolate most software problems depends upon the individual's idea for solving them. The sort of issues that software projects manage have a tendency to be exceptional and hard to define, and arrangements have a tendency to advance constantly as designers pick up a more prominent appreciation of what must be settled. Adding to the many-sided quality of the issue and its answer is the quick changing and very questionable environment, for instance, market turbulence and changes in client prerequisites and project goals [10]. It is vital along these lines to acknowledge that our suspicions and forecasts about future occasions will, by nature, be indeterminate. While overseeing software projects, we should be to a great degree careful of extrapolating past patterns or depending too vigorously on past experience. The more noteworthy the instability inborn in a project, the more the project needs to move from customary methodologies that depend on an altered succession of exercises to methodologies that permit to reclassify the exercises – or even the structure of the project arrangement – in mid course. Hence, as the uncertainty and complexity of a project increases, managers need take on roles toward flexibility and learning rather than the traditional risk management.

Some of the important project management software features to be considered are [20]:

1. *Task Management*: To simplify managing and achieving goals, they are broken down into a set of tasks. Tasks are created and managed during the entire process. Tasks such as creating tasks, managing subtasks from larger tasks, set tasks to recur or repeat should be handled by the software.
2. *Team Collaboration*: This forms one of the most important features especially in a distributed team environment. A virtual space needs to be created for discussions among team members. It should allow creation and sharing of documents as well as sending messages to one or more people.
3. *Email Integration*: Integration of the project management software with email turns out to be very beneficial as well as powerful. It can be used for sending updates, information about new tasks, and status reports to a predefined list of members.
4. *File Management*: The online application can provide storage space to manage the files and documents easily with or without the help of a third party. Features like adding notes to files, uploading files, having a version control, and organizing files can also be provided.
5. *Scheduling*: This feature of the software deals with setting time lines and creating milestones for completion of various tasks and also identifying dependencies between resources. This might not be very important for a small team or simple project.
6. *Project Management*: Project management is very crucial for larger organizations where templates need to be created, issues need to be managed, and prioritization among projects is required [9].
7. *Time Management*: Project management software can help in providing a certain degree of control in accepting submitted reports, timesheets, etc. This is valuable to project teams handling many resources and running for longer duration of time.

Software projects have many properties and attributes which make them different from any other engineering project. For instance, the product is intangible due to which we can say that a product is 90% complete even though there are not any visible outcome. Due to such issue, it is very important to have proper project management to ensure a quality product to the clients. The core activities involved in software project management are project planning, project scheduling, risk management, control, and managing people. An efficient software project management focuses on people, problem, and the process. People must be organized into teams and motivated to do quality work and should coordinate well to achieve effective communication and results. The problem must be communicated clearly from customer to developer which must then be decomposed into goals and assigned to the respective teams. Finally, the process

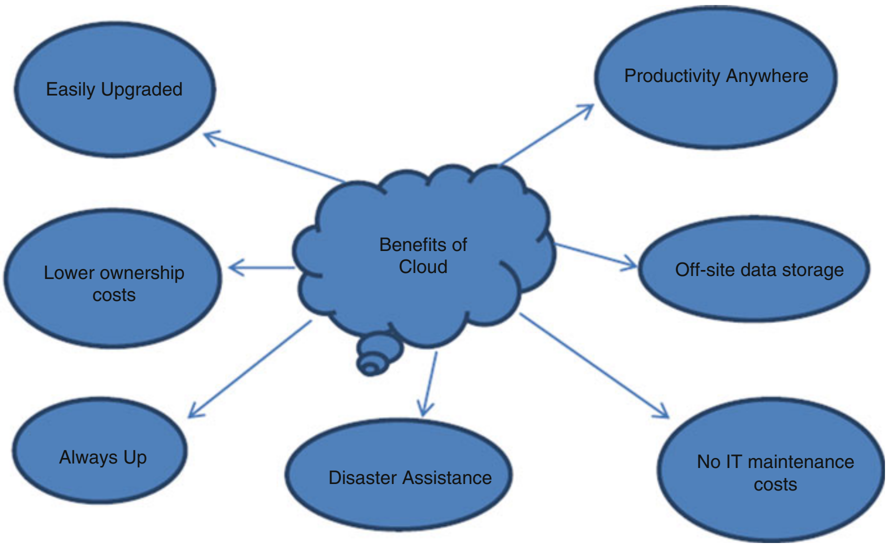


Fig. 5.5 Advantages of using cloud-based PM tools

should consist of a set of work tasks chosen which must be adapted to the people and problem.

With all of the above in mind, we can see that the use of cloud computing [18, 19] in software project management will prove to be immensely beneficial. Cloud-based project management tools can be used to set priorities and align teams to work faster and smarter across the organization. Business is moving faster, becoming increasingly collaborative, and embracing more remote workers every day. Hence, a system is needed which allows us to plan and adjust in real time. With the help of the cloud, it is possible to have a central tool to manage the entire software development process and track progress of the project and also monitor whether the employees are working toward the goals of the project and company. Another prime reason for shifting to cloud-based tools would be cost-effectiveness. Some advantages of using a cloud-based tool are shown in Fig. 5.5.

As it can be seen, the major advantages of using cloud-based project management tools include lower maintenance cost, and also it can be easily upgraded. Maintenance of servers and systems is one area where organizations tend to spend a lot of money. This cost can be drastically reduced with the use of the cloud. The other advantage cloud provides is that it can be easily upgraded and no extra hardware and systems need to be set up in case there is a need for scaling the services. The next section gives a critical evaluation of some of the popularly used cloud-based project management tools which have been compared using certain criteria.

5.5 Evaluation of Cloud-Based Software Project Management Tools

In the manner in which programs can be written either in editors such as notepad through Vi to eclipse, online project management tools range from shared to-do lists to multimedia collaborative environments. In order for a portal to be considered as a software project management tool, it must have a few specific features. Some of the criteria for evaluating the tools are bug tracker, to-do lists or some kind of task management support, and a document repository which helps the stakeholders to share and modify content and understand what work have been done so far. Another important feature is conversational tools like emails, chat, wikis, blogs, etc. which provide a mechanism for stakeholders to communicate and collaborate. All the other components like calendar sharing, report generation, tagging mechanism, and time tracking tools which may be provided are offshoots of the core features, namely, task management, document repository, and conversational tools [20].

There are many existing project management tools available online/in the cloud. Some of the popular tools are listed with the features they offer and the different industries they are used in:

1. *Freshdesk* [21]: It is the most recent in cloud-based support tech that comes with everything needed to manage and track projects. They follow a simple goal of making the process of brands talking to their customers and also making it easier for customers to get in touch with their businesses. An array of features like issue tracking, SLA management, smart automations, SEO ready FAQ section, knowledge base, and customizable self-service portals are provided by Freshdesk which helps increase agent productivity and reduce burnout.

Used by: Real estate, professional service providers, healthcare, and insurance

2. *Zoho Projects* [22]: Zoho projects are the project management software from Zoho. It provides features like project planning, assigning tasks, effective communication, update reminders, and detailed reports on progress. Unlimited users can be added to all plans with no extra cost.

Used by: Small and large teams across various industries

3. *TouchBase* [23]: TouchBase is totally a state-of-the-art, web-based project management software. It offers an incorporated bundle with management, asset tracking, purchasing, contract management, self-service portal, and knowledge base at a reasonable cost point. TouchBase gives all that you need an undeniable IT Help Desk and a beneficial Help Desk staff for your project management team. TouchBase can be easily customized as per your industry requirements.

Used by: Global corporations around the world

4. *SpiraPlan* [24]: SpiraPlan provides a complete agile project management system in one package that manages your project's requirements, releases, iterations, tasks, and issues in one environment, fully synchronized. Designed specifically to support agile methodologies such as Extreme Programming (XP),Kanban, Scrum, DSDM, and Agile Unified Process (AUP), it allows teams to manage all their information in one environment.

Used by: Project managers and IT professionals

5. *Easy Redmine* [25]: Easy Redmine is an open-source software for complex project management with extensions for resources, finance, and customer management. In the cloud or on your own server, all comes with professional implementation and support. Easy Redmine supports whole project lifecycle, so you can start with an area where you feel the most urgent need. Afterward, Easy Redmine can grow with you, thanks to the features which work as separately installable extensions. Over 20,000 users worldwide

Used by: Software developers, education, healthcare, media, government

6. *eXo* [26]: eXo platform is an open-source social-collaboration software designed for enterprises. It is full featured, based on standards, and extensible and has an amazing design. eXo helps companies connect their employees, customers, and developers through social, collaborative, and content-driven intranets, websites, and dashboards.

Used by: Large enterprises, mid-size businesses, public administrators

7. *Basecamp* [27]: Web-based software that makes it simple to communicate and collaborate on projects. It is used by millions of people, and 98% of its customers recommend it, primarily for its simplicity. It supports multiple languages and can be accessed on your mobile phone.

Used by: Freelancers, entrepreneurs, small businesses

8. *Genius Project* [28]: Genius inside offers its prime solution Genius Project since 2008 as cloud-based as well as on-premise solution for its project management software. Apart from the typical project management features, some of its noteworthy features include simulator which gives visual representation of the what-if scenario's in the project and phase and gate review support process for new product development, and it also provides Agile and SCRUM support. Genius Project is available in three deployment options: hosted on premise, SaaS, or installed on IBM's Lotus Notes. Extremely user friendly and customizable user interface with built in social collaboration platform, Genius Project fits the needs of every industry and provides benefits for everyone in the organization: PMO, executive, project manager, and team member.

Used by: Project centric companies of all industries

9. *Trello* [29]: Trello lets you organize anything with anyone. It is a flexible project management solution that fits into your workflow in a visual, collaboratively

focused way. Trello replaces post-it notes in a digital whiteboard format that can be used for anything from redesigning a website, to posting company updates regularly for management, to complex projects with many participants.

Used by: Project management teams of any size

10. *Kanzen* [30]: Kanzen is a project management and collaboration tool that focuses on but is not limited to Kanban method to improve business processes. A unique set of features allows user to view their workload in three views – a Kanban board, task list, personal task list, and a calendar. Intuitive interface and the ease of use will allow you to concentrate on your tasks rather than struggling with the software. Features include e-mail notifications, analytics, access rights, and more.

Used by: Businesses, project management teams, and individuals

11. *Salesforce* [31–32]: Uses the world's best CRM for small businesses in combination with a top project management tool from our AppExchange to better manage and gain visibility into all stages of your company's projects. The power of Salesforce plus a project management partner on our AppExchange will allow your company to reach its peak efficiency and productivity. Salesforce's Sales, Service, and AppExchange applications help companies connect with customers, partners, and employees in entirely new ways.

Used by: Companies of various industries and any team size

As seen in Table 5.1, the tools were compared and evaluated on the abovementioned criteria. Upon the evaluation of these tools, we find that most of the portals offer some kind of repository and ticketing mechanism and support for communication tools. A role-based access control was another feature common in many of the portals. Some major differences can be seen in the target markets of these tools apart from the usual pricing and licensing differences. For instance, BaseCamp [27] primarily targets small organizations that are staffed by nonprogrammers working on short- or medium-length projects. Though there are many software developers using it, it still forms a minority of the users. This might be the reason it offers a simple and easy-to-use file upload system rather than a version-control system. Another trend observed is that companies are giving more emphasis on agile methods which also have a big influence on the features offered by them. This explains the shift from asynchronous communication like bulletin boards to synchronous communication like chats.

After studying these tools, we understand the real importance of requirements elicitation and the importance of a structured development process in the success of a software project. This is one of the critical areas where we would like to target in SPMaaS. In addition, other research issues include are whether the information provided in the portal is sufficient enough to carry out the process or do the team members need to depend on other means of communication like personal communication which might not be recorded in the portal. With such problems arising, we feel that by addressing such issues, the next generation of tools can be designed to

Table 5.1 Table comparing tools based on features provided

	Tools						
Features	Zoho Projects	TouchBase	SpiraPlan	Easy Redmine	eXo Platform	Basecamp	Trello
Budget maintenance	✓	✓	✓	✓			
Collaboration	✓	✓	✓	✓	✓	✓	✓
Prioritization				✓			✓
Email integration	✓		✓		✓	✓	
File sharing	✓	✓	✓	✓	✓	✓	
Gantt charts	✓	✓	✓	✓			
Issue management	✓	✓	✓	✓			
Milestone tracking	✓	✓	✓	✓			
Project planning	✓	✓	✓	✓	✓		
Requirements management		✓	✓	✓		✓	
Status tracking	✓	✓		✓			
Time and expense tracking		✓		✓			

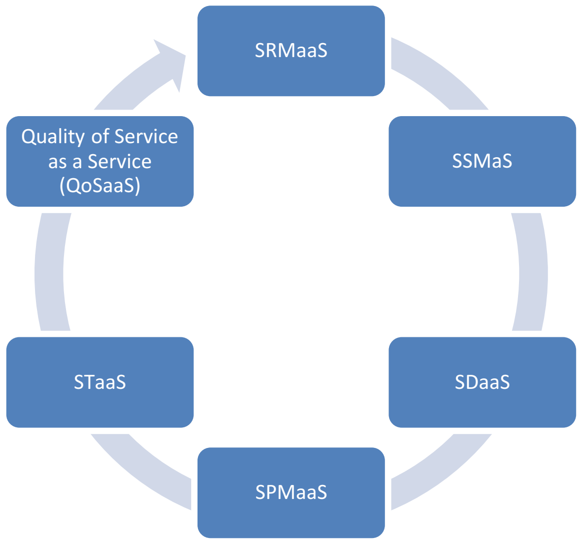


Fig. 5.6 Integrated software engineering as a service

better satisfy the real users' needs, and it is our aim to do this with the help of a cloud-based service SPMaaS.

5.5.1 *Integrated Software Engineering as a Service*

After evaluating the existing tools, we have realized the need of integrating project management in the software development cycle. The project management phase will begin immediately once the project development has begun. This can be seen clearly in Fig. 5.6.

Figure 5.6 clearly shows the overall proposed infrastructure of the proposed project management as a cloud service and how it will sit in the cloud. It will begin with software requirements as a (SRMaaS) service followed by software security management (SSMaaS). After this, the development will begin (SDMaaS), and project management will begin soon after (SPMaaS). The lifecycle will end with software testing (STaaS) and ensuring quality of the product (QoSaaS). The next section gives a detailed explanation on our proposed approach of SPMaaS and highlights its core activities.

5.6 Integrated Service Development Process and Software Project Management for SPMaaS

Project management can help companies, managers, and project teams to consummate client requirements, budget, manage time, and scope constraints. It is very important for the companies to choose the right tools so that it can help them save project cost and project time. Basically, there are two types of project management software:

- On premise [33]: These software systems reside in the data center owned by the company and runs on their own server. It is maintained by the IT employees of that company: Microsoft
- Cloud based [34]: This uses cloud technology and is offered by service providers as *SaaS*(software as a service). Many small- and medium-sized enterprises use cloud-based project management tools across different industries.

Figures 5.7 and 5.8 show the core activities which will be offered as a cloud service SPMaaS. It includes software project planning (SPPaaS), software cost estimation (SPCEaaS), software team management with support to handle virtual teams and multi-tenancy (STMaaS), and continuous delivery (CDaaS).

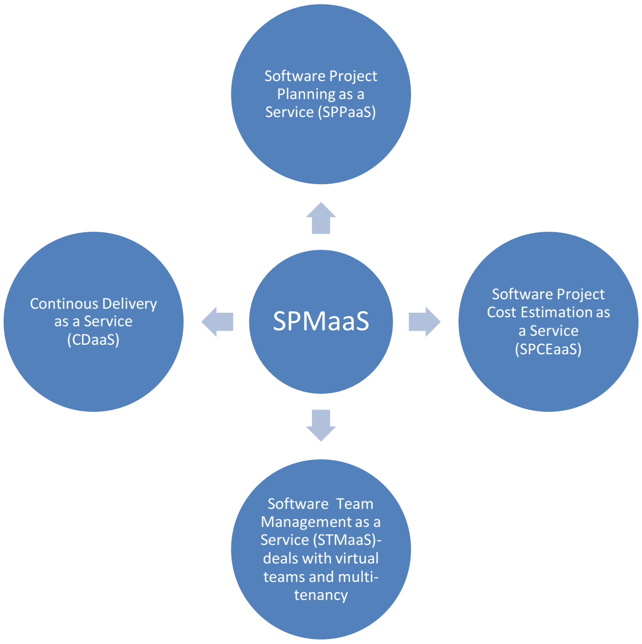

Software Project Planning as a Service (SPPaaS)
Continous Delivery as a Service (CDaaS)
SPMaaS
Software Project Cost Estimation as a Service (SPCEaaS)
Software Team Management as a Service (STMaaS)-deals with virtual teams and multi-tenancy

Fig. 5.7 Core activities of SPMaaS

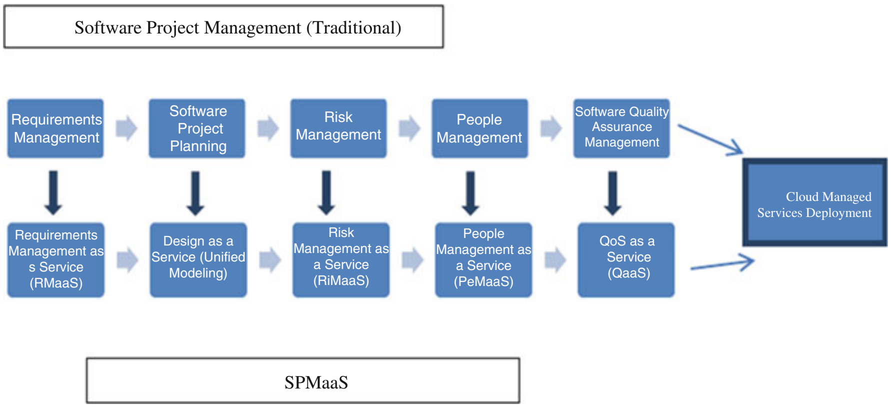

Software Project Management (Traditional)
Requirements Management
Software Project Planning
Risk Management
People Management
Software Quality Assurance Management
Requirements Management as s Service (RMaaS)
Design as a Service (Unified Modeling)
Risk Management as a Service (RiMaaS)
People Management as a Service (PeMaaS)
QoS as a Service (QaaS)
Cloud Managed Services Deployment
SPMaaS

Fig. 5.8 Software project management as a service (SPMaaS)

A detailed explanation of the features provided by SPMaaS is as follows:

- Requirements Management as a Service: Requirements management refers to the process of documenting, tracing, analyzing, agreeing, and prioritizing on requirements and then controlling change and then communicating it to the appropriate stakeholders. Any capability to which a product or service should conform is called a requirement. Poor requirements management is one of the major causes of project failure, and hence, it is a very important phase. This can help us exceed stakeholder expectations, improve performance, and meet the expected project goals. Refer to Fig. 5.9.
- Design as a Service: Unified modeling languages (UML) are used to provide a standardized way to visualize the design of a system. A set of diagrams can be drawn to visualize the system such as activities, individual components of the system, interaction among software components, external user interface, etc. The types of diagrams include structure diagrams like class diagram, component diagram, object diagram, and behavior diagrams like activity diagram, use-case diagram, interaction diagram, etc. UML diagrams help in simplifying the software development process and reducing development time.
- Risk Management as a Service: It refers to the identification, assessment, and prioritization of risks. The objective of risk management is to assure that uncertainty does not deflect the endeavor from the business goals. Risks need to be identified as early as possible to avoid any obstacles in smooth development process. Since there are infinite number of events that can have negative effect on a project, no project can ever be risk-free. Good an efficient risk management increases the likelihood of a successful project.
- People Management as a Service: People management aims at getting things done from people through effective management to produce outstanding results. It deals with understanding, managing, and delivering people's expectations. People management is one of the hardest aspect of a project management. A good team manager needs to understand strengths and wekaness of the team

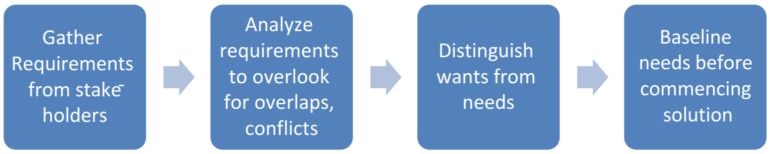

Gather Requirements from stake-holders
Analyze requirements to overlook for overlaps, conflicts
Distinguish wants from needs
Baseline needs before commencing solution

Fig. 5.9 Requirements management process

members and being able to be a motivator and should be willing to take the leadership of the team work. However, if we have a good team processe in place, it is then possible to achieve the common objectives of the team as well as the project.

- Quality as a Service: Apart from delivering the product on time, the quality of the product also plays an important role. There is a need of a process which ensures that the developed software meets and complies with standardized quality specifications. It needs to be an ongoing process within the lifecycle that routinely checks the software and hence ensures the development of a high-quality product.

With all of the above services provided as a cloud service, the process of project management is simplified, and the project can be managed very effectively and efficiently. The next section gives a detailed explanation of the architectural design with service-oriented architecture and SoaML diagram.

5.7 Architectural Design of SPMaaS with SOA

The idea of a service has been defined by many people in numerous ways. While it has been described as an encapsulated unit of functionalities, it has also been considered as a logical manifestation of some physical resources grouped as a process that an organization exposes to a network. A service has been defined as an externally observable behavior of a software/hardware component in which the internal working and processing details are well hidden and made available through a set of well-defined interfaces [35]. A service in the context of web services can also be viewed as an application or business logic that exposes its functional capabilities to clients by running on a server. Refer to Fig. 5.10. One thing which is common and is being tried to be explained in all of the definitions is that service can be viewed as conceptual identity which supports certain actions in response to a set of requests received. These requests can be in the form of messages or some kinds of programs written to trigger the internal processing at the service providers' end.

Software applications can be implemented using abstraction as the fundamental design entity. Each service clearly encapsulates certain features while at the same time hiding the underlying implementation details from the user/client. This concept greatly benefits while building systems to implement higher level services. The set of services which need to be provided are decided in the software development

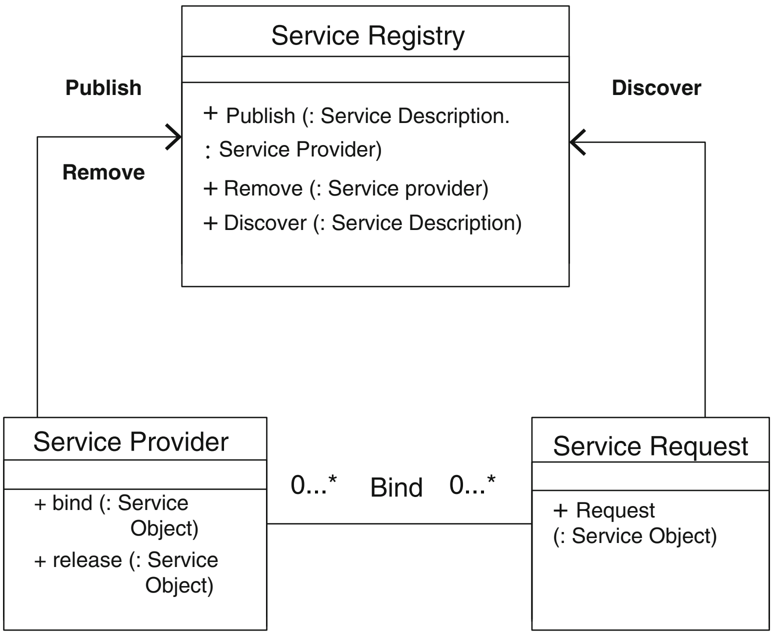

Service Registry
+ Publish (: Service Description.
: Service Provider)
+ Remove (: Service provider)
+ Discover (: Service Description)
Publish
Remove
Discover
Service Provider
+ bind (: Service Object)
+ release (: Service Object)
0...*
Bind
0...*
Service Request
+ Request (: Service Object)

Fig. 5.10 Conceptual model of service-oriented paradigm

phase. The complete system can then be built by having these services as the fundamental design entities.

In this architectural style, an interaction model between parties is defined, namely, service provider, service consumer, and service request. The provider publishes the service description and provides implementation for a service. The consumer can either use the URI for the service description directly or can find the service description in a service registry and invoke and bind a service.

5.7.1 Types of Services Offered by SPMaaS

There can be many services provided by providers in software project management, which depending on the complexity may require different levels of processing. Services can be composed into three types: elementary services, collaborative services, and composable services.

For instance, the features provided in software project management can be also broadly classified into services for planning and scheduling, services for collaboration, services for documentation, etc.

5.7.1.1 Elementary Services in SPMaaS

These refer to services which do not require complex processing and which are independent in nature. There are no additional requirements or constraints that need to be fulfilled in order to add these kinds of services. The client can add an elementary service by making a simple request in any form, and the respective service will be made available to the user. For example, services like software testing (STaaS) can be invoked separately without any previous requirements.

5.7.1.2 Composable Services in SPMaaS

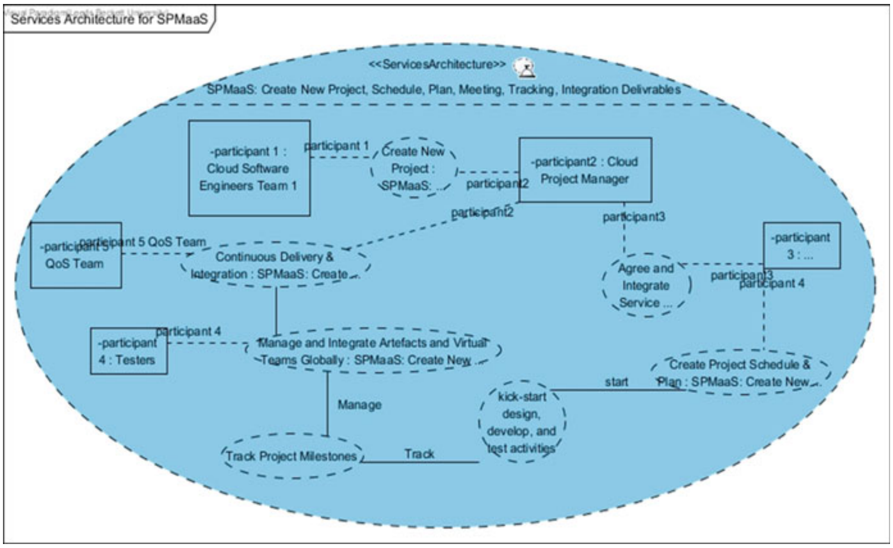
These are the services which are not readily available but can be provided by invoking a set of services belonging to the same category of services. Consider a scenario in which the client needs to add a new service. In order to provide the client with this service, it might be possible that there must be some existing services the client should already be using so that the new service can be provided with the help of them. The new service which the client might want could need the support of another service for its fulfillment or it might be an extension to some older service. For example, if a client wish to add a calender shring feature, then, this should be part of a Software Planning as a Service (SPPaaS).

5.7.1.3 Collaborative Services in SPMaaS

These services can neither be composed using the services at a service window nor can be available readily. Basically, these are the services which can be provided only if a set of conditions are followed. For example of a composite service (consists of invoking a sequence of a number of other services), taking a hypothetical situation, if you invoke a new software project cost estimation as a service (SCEaaS) of the newly created instance of a SPMaaS project, then this service will create autonomically new instance of software requirements engineering as a service (SREaaS) and software security requirements engineering as a service (SSREaaS) which in turn will also invoke software project planning as a service (SPPaaS). Hence in this sequesnce of service invokation, one services is interdependent on a set of other services and establishing service choreography.

5.7.2 Design of Cloud SPMaaS with SoaML

The SOA design for SPMaaS can be clearly explained with the help of the diagram shown in Fig. 5.11. The diagram has been drawn using service-oriented architecture modeling language (SoaML) which is an extension of UML 2.0 to support service



Services Architecture for SPMaaS
<<ServicesArchitecture>>
SPMaaS: Create New Project, Schedule, Plan, Meeting, Tracking, Integration Delivrables
-participant 1 : Cloud Software Engineers Team 1
participant 1
Create New Project : SPMaaS: ...
participant2
-participant2 : Cloud Project Manager
participant3
-participant 5 QoS Team
participant 5 QoS Team
Continuous Delivery & Integration : SPMaaS: Create...
Agree and Integrate Service ...
participant3
participant 4
-participant 3 : ...
-participant 4 : Testers
participant 4
Manage and Integrate Artefacts and Virtual Teams Globally : SPMaaS: Create New ...
Create Project Schedule & Plan : SPMaaS: Create New...
start
kick-start design, develop, and test activities
Manage
Track Project Milestones
Track

Fig. 5.11 SOA design for SPMaaS with SoaML

concepts. SoaML provides a standard way to architect and model SOA solutions using the unified modeling language (UML). A services architecture (SOA) is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and services that fulfill those roles.

The diagram, in Fig. 5.11, shows various participants (rectangle boxes) such as cloud software engineering team, project manager, testers, QoS teams, etc. Participants can provide as well as use services (represented with oval shape). Some of the services provided which can be seen in the diagram are creating a new project, creating a project schedule and plan, and tracking project milestones and continuous delivery and integration. The SoaML diagram also clearly shows which participants can create a service, use a service, or invoke a service with the help of messages. For example, we can see that a new project can only be created by participant1, namely, cloud software engineer team and participant2, the cloud project manager. Thus, with this diagram we can understand:

- The roles each participant plays in a service
- The message types that go between participants when a service is enacted
- Interfaces provided and used by each participant for the service
- Choreography of interactions between the participants while enacting the services

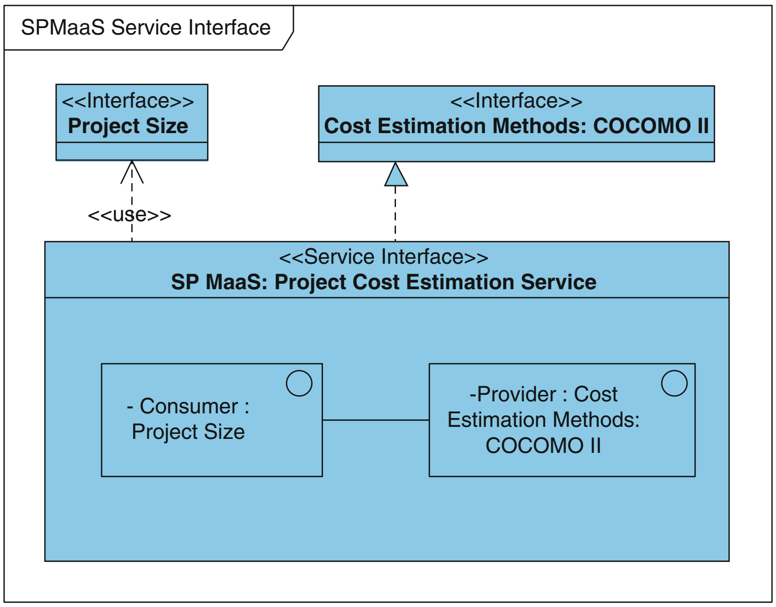

SPMaaS Service Interface
<<Interface>>
Project Size
<<Interface>>
Cost Estimation Methods: COCOMO II
<<use>>
<<Service Interface>>
SP MaaS: Project Cost Estimation Service
- Consumer : Project Size
-Provider : Cost Estimation Methods: COCOMO II

Fig. 5.12 SPMaaS service interface model

5.7.2.1 Part I – SPMaaS Service Interface Model

Service interface diagram is one of the most important SoaML diagram types. The idea of a service interface diagram is based on the core aspect being a service. A service in this case can be defined as a value delivered to another through a well-defined interface. In SoaML, a service can be specified using three approaches, namely, simple interface, service interface, and a service contract. It can be seen in Fig. 5.12 that a service interface *SPMaaS: Project Cost Estimation Service* is created. A service interface involves communication and interaction between a consumer and provider of services. In Fig. 5.12, it can be seen that the consumer is *Project Size* which is provided with *Cost Estimation Methods*. There are also two simple interfaces- Project Cost and *Cost Estimation Methods* which have beeen provided. The service interface of the project cost estimation service specifies its required needs through usage dependencies to the *Project Size* interface and the receptions and operation it receives through the *Cost Estimation Methods* interface.

5.7.2.2 Part II – Specifying SPMaaS Choreography Using UML Sequence Model

Service choreography defines the interaction between the provider and consumer in completing a service. We can specify how the consumer interacts with the provider

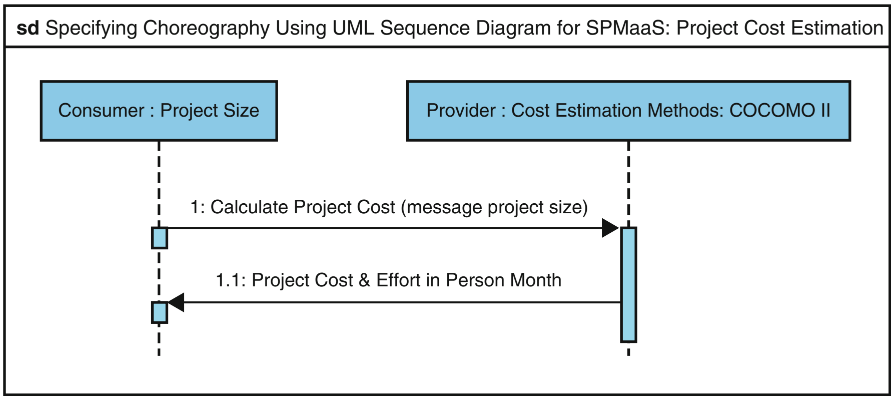

sd Specifying Choreography Using UML Sequence Diagram for SPMaaS: Project Cost Estimation
Consumer : Project Size
Provider : Cost Estimation Methods: COCOMO II
1: Calculate Project Cost (message project size)
1.1: Project Cost & Effort in Person Month

Fig. 5.13 SPMaaS choreography using UML sequence diagram

of service with the help of sequence messages between the two lifelines. In Fig. 5.13, it can be seen that the consumer begins by invoking the provider to calculate the project cost. The provider in turn reacts by replying with the person cost and effort in person month.

5.7.2.3 Part III – SPMaaS Service Participant Model

In SoaML, participant refers to a certain party or component that provides and/or consumes a service. Participants can be software components, organizations, systems, or individuals. In Fig. 5.14, globally distributed teams/ virtual teams or client has been taken as the participant. We can also see the services provided and used by the participant. A square which represents the port can be seen providing the interface for creating a new project and requiring the client requirements.

5.7.2.4 Part IV – SPMaaS Service Contract Design

A service contract specifies and defines the agreement between parties about how a service is to be provided and consumed. Interfaces, choreographies, terms, and conditions are used to define the agreement. The interacting participants must compulsorily agree and adhere to these agreements in order for the service to be enacted. Figure 5.15 shows the service contract for a new project contract service in SPMaaS in which the consumer and provider need to agree to the project contract agreement.

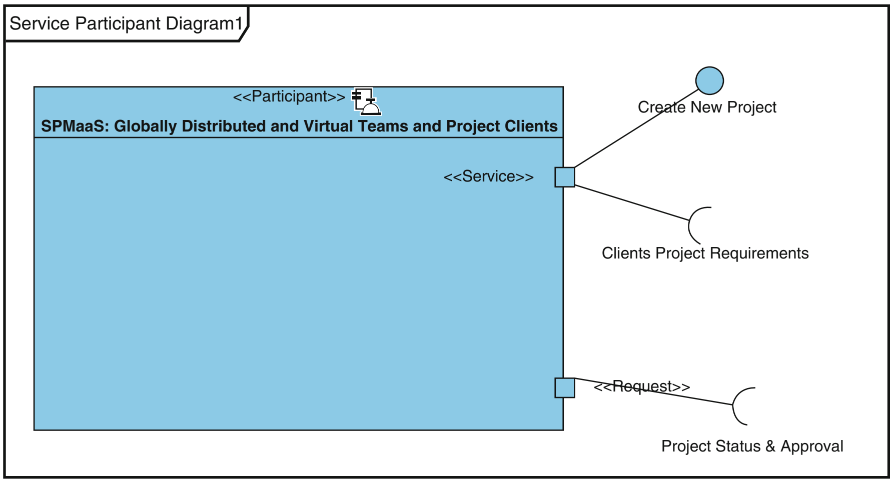

Service Participant Diagram1
<<Participant>>
SPMaaS: Globally Distributed and Virtual Teams and Project Clients
<<Service>>
Create New Project
Clients Project Requirements
<<Request>>
Project Status & Approval

Fig. 5.14 SPMaaS service participant model

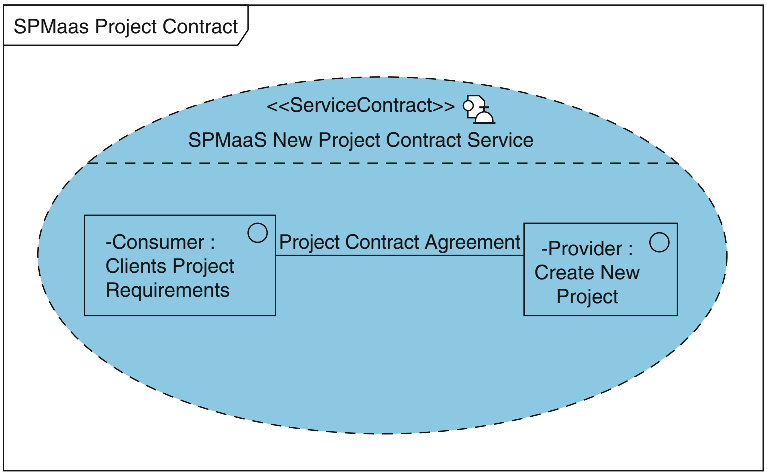

SPMaas Project Contract
<<ServiceContract>>
SPMaaS New Project Contract Service
-Consumer : Clients Project Requirements
Project Contract Agreement
-Provider : Create New Project

Fig. 5.15 SPMaaS service contract design

5.7.3 *Results and Analysis of SPMaaS Design*

The design for cloud services is challenging, and it is in its infancy for a proper and systematic approach to engineering cloud service design and development as we have shown in this chapter the importance of engineering cloud services with the current state-of-the-art tools and standards such as SoaML which has been specifically developed for cloud service engineering. Figure 5.16 shows how we have measured a number of service components for each of the SoaML stages.

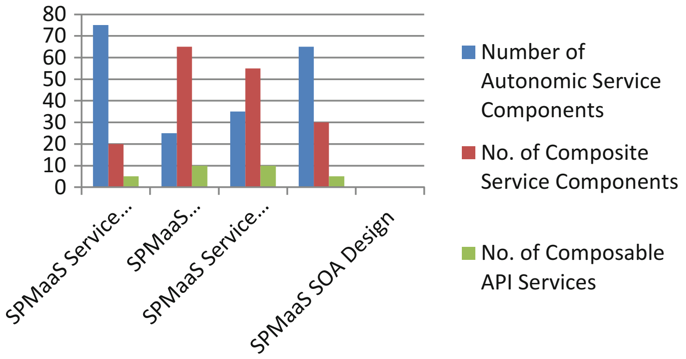


Fig. 5.16 Graph showing count of service components for different SoaML design categories

The graph shown in Fig. 5.16 shows a count of the service components for each of the different five SoaML design categories like SPMaaS service interface model, SPMaaS choreography, SPMaaS service participant model, SPMaaS service contract, and SPMaaS SOA design. Three different service components have been considered:

- *Autonomic service components*: These are the components that can work with any dependency to complete a full service.
- *Composite service components*: These depend on other services to complete a full service and cannot exist independently.
- *Composable API services*: These are the API services which can be created with the support of existing APIs available and by integrating them with some new features and providing them to the customers for easier and more specific use.

5.8 Conclusion

Cloud computing is emerging rapidly with increasing demand for service-oriented computing and associated technologies. This is the right time to explore what works better and what doesn't work for cloud environment. Therefore, the proposed model helps to understand how it should be developed to avoid classical issues related to software development projects. We believe the proposed model will help us to develop cloud applications systematically. Another major contribution of this chapter is to use SoaML to design cloud-based software project management as a service system (SPMaaS). SoaML provides a standard way to architect and model SOA solutions using the unified modeling language (UML). A services architecture (SOA) is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and services that fulfill those roles. This study discovered overall 70%

improvement of the cloud-based services by designing with SoaML by counting number of service components during the design phase of this research.

References

1. Cloud: Amazon Elastic Compute (2011) Amazon web services. Retrieved 9 Nov 2011
2. Buxmann P, Thomas H, Sonja L (2008) Software as a service. Wirtschaftsinformatik 50 (6):500–503
3. Zhang QI, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. J Int Serv Appl 1(1):7–18
4. Wang L, Laszewski VG (2008) Scientific cloud computing: early definition and experience. http://cyberaide.googlecode.com/svn/trunk/papers/08-cloud/vonLaszewski-08-cloud.pdf
5. Ramachandran M (2008) Software components: guidelines and applications. Nova Publishers, New York
6. Bichier M, Lin K-J (2006) Service-oriented computing. Computer 39(3):99–101
7. Khan A et al (2012) Cloud service for comprehensive Project Management Software. Application of Information and Communication Technologies (AICT), 2012 6th international conference on IEEE
8. Bentley C (2010) Prince2: a practical handbook. Routledge
9. Thayer RH, Yourdon E. (1997) Software engineering project management. In: Software engineering project management, pp 72–104
10. Helbig J (2007) Creating business value through flexible IT architecture, Special Issue on Service-oriented Computing. IEEE Computer 40(11)
11. IaaS (2010) Cloud computing world forum. http://www.cloudwf.com/iaas.html
12. IThound Video whitepaper (2010) http://images.vnunet.com/video_WP/V4.htm. Accessed Feb 2010
13. SaaS (2009) SaaS. http://www.saas.co.uk/
14. Science Group, 2020 Science Group: toward 2020 science, tech.report, Microsoft, 2006. http://research.microsoft.com/towards2020science/downloads/T2020S_Report.pdf
15. Vouk MA (2008) Cloud computing – issues, research and implementations. J Comput Inf Technol, CIT 16
16. Wilson C, Josephson A (2007) Microsoft Office as a platform for software + services. Archit J 13. www.architecturejournal.net
17. Zhang L-J, Zhou Q (2009) CCOA: cloud computing open architecture. In: IEEE international conference on web services
18. Armbrust M, Fox A, Grifth R, Joseph AD, Katz R et al (2009) Above the clouds: a Berkeley view of cloud computing. Technical report, University of California at Berkeley. URL http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html
19. Foster IT, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared, CoRR abs/0901.0131
20. Project Management Tools (2016) http://modeling-languages.com/survey-web-based-software-project-management-tools/. Accessed Sept 2016
21. Freshdesk (2016) https://freshdesk.com/. Accessed Sept 2016
22. Zoho projects (2016) https://www.zoho.com/projects/. Accessed Sept 2016
23. TouchBase (2016) http://www.productdossier.com/. Accessed Sept 2016
24. SpiraPlan (2016) https://www.inflectra.com/SpiraPlan/. Accessed Sept 2016
25. Easy Redmine (2016) https://www.easyredmine.com/. Accessed Sept 2016
26. eXo Platform (2016) https://www.exoplatform.com/. Accessed Sept 2016
27. BaseCamp (2016) https://basecamp.com/. Accessed Sept 2016
28. Genius Project (2016) http://www.geniusproject.com/. Accessed Sept 2016
29. Trello (2016) https://trello.com/. Accessed Sept 2016

30. Kanzen (2016) https://mykanzen.com/. Accessed Sept 2016
31. Salesforce (2016) http://www.salesforce.com/in/. Accessed Sept 2016
32. Cost Estimation Techniques (2016) http://www.computing.dcu.ie/~renaat/ca421/report.html#2.6. Accessed Sept 2016
33. Turner JR (1993) The handbook of project-based management: improving the processes for achieving strategic objectives. McGraw-Hill, London
34. Sotomayor B et al (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 13(5):14–22
35. Wan K-M et al (2006) Service-oriented architecture