Chapter 6
Business Requirements Engineering for Developing Cloud Computing Services

Muthu Ramachandran

Abstract Cloud computing is an emerging paradigm that is becoming rapidly popular with business organisations. The software-as-a-service (SaaS) delivery approach is increasing in demand for yet more cloud-based services. However, this new trend needs to be more systematic with respect to software engineering (design and development) and its related processes. In this case, a valid question is: How do we change our existing user-based requirements capturing methodologies to a suitable service-based business requirements engineering? In this chapter, we present an approach to cloud requirements engineering that is based on business-oriented analysis as this is the key to a successful cloud service. This chapter explores the new requirements engineering process and relevant techniques for capturing cloud-based services. The process and techniques have been explained using a large-scale case study based on Amazon Cloud EC2.

Keywords Cloud computing • Software engineering • Requirements engineering • Cloud services • Service-oriented computing

6.1 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 In particular, computing hardware and software often gets outdated, and, hence, it is wise to outsource computing resources and to manage their IT infrastructures outside of their company premises, which is more cost-effective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 the case at present. Applications can be

M. Ramachandran (✉)
School of Computing and Creative Technologies, Faculty of Arts,
Environment and Technology, Leeds Metropolitan University, Leeds LS6 3QS, UK
e-mail: M.Ramachandran@leedsmet.ac.uk

leased (as pay-per-use services) rather than being purchased. Also, companies have increased their data centres due to demand (Amazon, Microsoft and IBM). Cloud computing is heavily based on 'software-as-a-service' concept and needs high-speed Web access. It provides services on demand, utilising 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 reconfigurability as they support services and its agreements (e.g. service-level agreements). In this scenario, the resource management of cloud computing is key to achieving potential benefits.

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). Wang and Laszewski [1] define cloud computing as a set of network-enabled services which provide scalable, guaranteed QoS (quality of service) and inexpensive computing platforms on demand, which are customisable (personalised), and all of which can be accessed in a simple and pervasive way. An overview of different cloud computing paradigms is presented with definitions, business models and technologies by Wang and Laszewski [1] and by many others [1–34].

Traditionally, requirements engineering is defined as a set of activities involving various stakeholders to elicit requirements for a software system. This process is further refined to provide clear classes of requirements such as functional, non-functional, governance and business. Requirements validation is another process of making sure that the requirements are clear, consistent and contextual (3Cs). Business requirements is often not clearly identified and captured as this is directly related to business level. Therefore, we can define business requirements as a process of discovering, analysing, defining and documenting the requirements that are related to enterprise-wide business objectives. This process involves identifying and capturing key business stakeholders who are mainly investors (use interviews, focus groups, ethnographic studies and current market analysis), conducting business feasibility analysis using ROI (return on investment) strategies, studying organisational objectives that should represent true value for long-term investment, analysing the impact of business change to the enterprise and forecasting profit with respect to a set time period, prioritising business requirements and producing a business requirements document to sign off.

SaaS design process involves identifying service components and artefacts that can all be mapped onto service-oriented architecture (SOA). 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 customisable and interfaces can be designed supporting SLA (service-level agreement). SLAs vary between service providers which need to be customised without much effort. This can only be achieved using components which have been designed for flexible interfaces that link to a number of SLAs. Each SLA and associate business rule can be represented as a set of interfaces that can be mapped onto knowledge-based database or a data server. This also allows reuse of SLAs for any individual service providers. Some of the important characteristics of cloud computing are:

- On-demand services and pay per use
- Handling wide area multiple network addresses
- Resource grouping and management
- Efficient elasticity vs. costing
- Measurable service delivery and QoS

The first characteristic of on-demand service and pay-per-use cost efficiency model poses tremendous challenges to provide efficient support and a trustworthy cost model (provided by cloud service providers) for pay per use for every resource used by customer services automatically. Cloud computing is based on clients with high bandwidth for Internet access, and each client may have *N* number of end users or cloud application users. Therefore, it will create *N***N* multiple network addresses which need to be managed accurately as it has a strong dependency for costing users. The second characteristics of cloud computing is based on clients with high bandwidth for Internet access, and each client may have *N* number of end users or cloud application users. Therefore, it will create *N***N* multiple network addresses which need to be managed accurately as it has a strong dependency for costing users. The third characteristic on resource grouping and management has to be monitored and managed efficiently by cloud service providers both reasons of efficiency and costing. The fourth cloud characteristic on elasticity, scalability and costing poses huge challenge for cloud service providers as part of the cloud service management system. The final cloud characteristic on measurable service delivery and quality of service (QoS) has long-term implications for cloud service providers to measure and improve service quality continuously.

Our earlier work described by Ramachandran [22] 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 amongst SOA (service-oriented architecture), grid and cloud computing. *SOA* is to offer services which are based on open standard Internet services and virtualisation technology and have been running in a different environment, *grid* offers services from multiple environments and virtualisation 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.

This chapter has proposed a model which is based on the notion of design for scalability of the cloud architecture which is driven by business requirements. We have also identified a set of business as a service for Amazon EC2 cloud. The result shows that 20 % represents BPaaS services from business requirements.

6.2 Design for Cloud Applications

The idea of design for reuse and design for testability have emerged to address how best design can be represented in the system which reflects expected design characteristics (based on design principles) such as reusability, testability, securability (building software security in) and scalability. These are the four basic architectural characteristics that are prevalent in most systems. The main purpose of identifying them during requirements stage is to build them right from the beginning; therefore, they can exhibit themselves on-the-fly. In order to define a process model for cloud computing applications, it is useful to capture some of our thoughts on understanding the very nature of cloud characteristics and the type of services that it 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) involves integration of several disciplines and subject areas, and, therefore, some of the characteristics will overlap. Some of the identified service components characteristics are:

- Reusable Web services and some other core services
- Enterprise integration services
- Dynamic binding and reconfigurable at runtime
- 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 recoverable
- Cloud infrastructure and resources management
- Autonomic framework
- Middleware
- QoS
- Controllability
- Visibility and flexibility
- Security and privacy
- High performance and availability
- Integration and composition
- Standards

These characteristics and their underpinning design principles embody a large variety of best practices that exist widely. These best practices have evolved over the last two decades of software engineering. For example, software requirements engineering, software reuse concepts and practices have been widely adopted and are used in the industry. Therefore, the main aim of this chapter is to consider a systematic approach to capturing business requirements that can be applied to the cloud paradigm. Service design is based on the principle of loosely coupling and therefore is a good candidate for achieving service-level reuse such as business services,

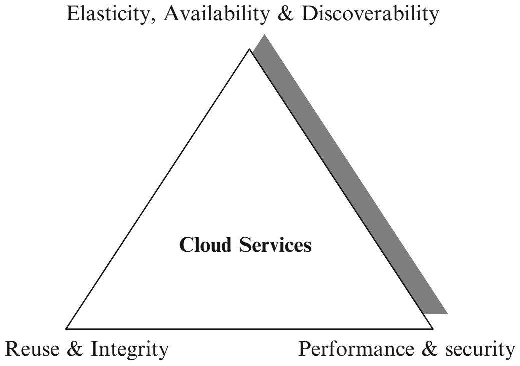

Fig. 6.1 Design for cloud applications

infrastructure services, composite services (as services are designed based on the principle autonomous), co-operation services, information services, task-oriented services, and orchestration services). Therefore, service level reuse has potential to save service development cost and cloud resource utilisation cost. The notion of design for reuse, design for test (also known as testability) and design for security exists in software engineering literatures more widely (Ramachandran [22]). Controllability, visibility and flexibility are design characteristics that can help to build and recover new services more widely. High performance, standards and availability characteristics can provide required service quality. In order to make a design for cloud applications, we need to understand various required cloud characteristics and provide a clear set of design guidelines that can be used by cloud applications engineers. Some of such guidelines are presented as:

- Make applications loosely coupled using SOA principles.
- Design for cloud will provide a value for money in the longer term.
- Use cloud and SOA design principles and characteristics as strictly as possible as discussed by Erl [10].
- Leverage three-tiered SOA architecture which will even allow you to design a database service linking to two different cloud providers.
- Make use of asynchronous messaging wherever possible as discussed by Linthicum [35].
- Avoid cloud-specific APIs wherever possible so as allowing portability across clouds.

Our work on best practice software guidelines provides a disciplined approach to service-driven software development life cycle [22]. Our previous work on this has identified guidelines on good requirements representation using use case models for identifying common requirements across a range of software product lines [22]. Reuse of service-level business requirements can yield higher-level reuse across cloud service. Making business service requirements can develop reuse across different levels in the SOA model. Therefore, Fig. 6.1 shows a model for design for reuse which focuses on elasticity, availability and discoverability, reuse and integrity and performance and security. For each business requirement, we need to conduct analysis based on main six criteria identified in this model with view to future business and its sustainability.

For simplicity, we can define some of the terms very briefly. Elasticity directly represents business focus for services which provide value proposition, and, therefore, service should be able to expand and contract resources based on demand and be able to charge pay per use. Availability can be defined with respect to business focus to ensure that the services are available by creating multiple data centres, proper disaster recovery planning and providing service recovery and failover mechanisms in place. Discoverability is one of the key criteria as part of service-oriented design principle, meaning that the service should be designed in such a way that it can be discovered automatically and should be able to be adopted by service requesters automatically or with a minimum human input. Elasticity, availability and discoverability are part of quality of service (QoS).

Service reuse can be defined as the process of linking business service together to solve an end-to-end business problem or a business process. Although this looks simple but can create reuse across cloud services with automatic discoverability and composability with strict integrity in place. Oh et al. [36] states that the reusability is a key intrinsic characteristic of cloud services and can yield a high return on investment (ROI). Services can be reused and composed to create new cloud services and applications from a set of common service directories across different cloud providers. Service integrity can be defined as the degree to which a service can be provided without excessive impairment and the degree to which it provides fair value to the business. Service reusability and integrity are part of the key criteria for measuring the quality of service (QoS).

One of the main reason for moving cloud is the cost benefit. Therefore, it is paramount for cloud providers to ensure performance is effective. There are a number of performance characteristics such as network throughput and latency. Service availability is another key factor in measuring cloud performance. This is also known as uptime. Other parameters include scalability of service applications, pay per use, load balancing, elastic load balancing, number of cloud computing created per service instance, number of cloud images created per instance, number of cloud resources created per instance and cloud profiling.

Cloud security is paramount amongst all other characteristics as cloud service is internet based. Therefore, we need to make sure that network security, denial of service attacks, software service security and other forms of security are well protected. Other aspects include cloud content management, privacy, business continuity and data recovery.

6.3 Business-Oriented Cloud Service Development Process

Identification of service requirements 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

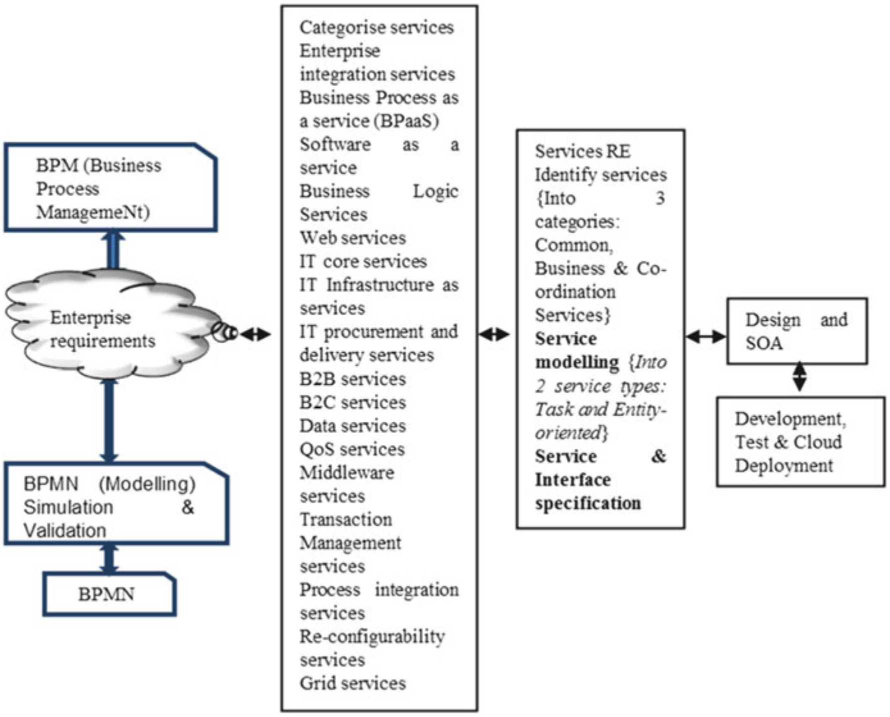
BPM (Business Process ManagemeNt)
Enterprise requirements
BPMN (Modelling) Simulation & Validation
BPMN
Categorise services
Enterprise integration services
Business Process as a service (BPaaS)
Software as a service
Business Logic Services
Web services
IT core services
IT Infrastructure as services
IT procurement and delivery services
B2B services
B2C services
Data services
QoS services
Middleware services
Transaction Management services
Process integration services
Re-configurability services
Grid services
Services RE
Identify services {Into 3 categories: Common, Business & Co-ordination Services}
Service modelling {Into 2 service types: Task and Entity-oriented}
Service & Interface specification
Design and SOA
Development, Test & Cloud Deployment

Fig. 6.2 Business-oriented cloud service development process

services is very hard. The idea in service-oriented engineering is to publish automatically new services whereby service agents can then be able request and take advantages of required services for their customers. Figure 6.2 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 (Services RE) 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 service-level specifications.

The third phase (Categorising services) involves classifying and distinguishing services into various categories such as enterprise integration services (services across corporations, departments, other business services), BPaaS (which represents process related to businesses), software services (which represents core functionality of software systems), business logic services (which represents business rules and its constraints) and Web services (a self-contained and Web-enabled entity which provides services across businesses and customisable at runtime). IT core services include resource management, help desk systems, IT infrastructure, procurement, delivery services, B2B and B2C services, data services, QoS services, middleware

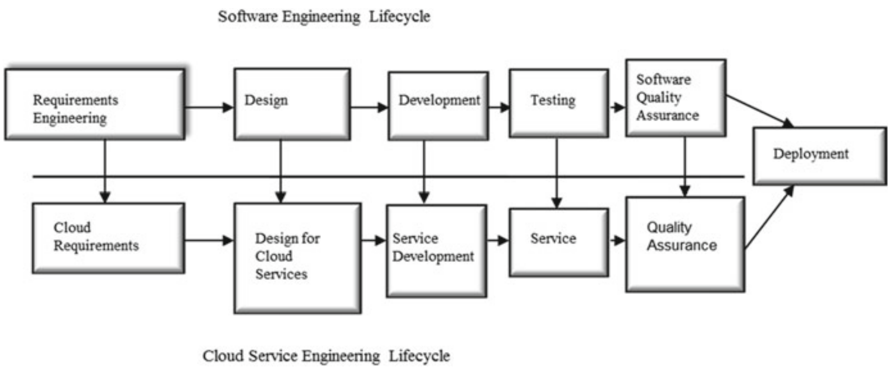

Fig. 6.3 Software engineering vs. cloud service engineering life cycle

services, transaction management services, process integration services, reconfigurability services and grid services which include grid resource management and reconfigurations. Based on the above finding, we can propose a new paradigm for cloud applications engineering as shown in Fig. 6.3. This illustration provides a relevant link to classical software engineering process.

As shown in Fig. 6.3, 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 in rather than adding security batches after release. The design phase is linked to designing services for cloud environment which are reusable. Services are designed as loosely coupled allowing 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 key difference in cloud SE life cycle is service quality engineering/assurance (SQoS). Service quality engineering/assurance represents quality of service aspects which is different from software engineering quality. SQoS should consider parameters such as workflow management which helps to manage resources instantly, accuracy and accountability of pay-per-use, throughput, latency and service satisfactory index.

6.3.1 Business Process as a Service Paradigm

Business process as a service (BPaaS) is a top-level part of the service-level architecture (BPaaS → SaaS → PaaS → IaaS) for cloud platform. This refers to any business process such as payroll, multivendor e-commerce, advertising, printing, enterprise-wide applications and common business processes and could include contract negotiation services [37]. BPaaS services can also be designed to automate

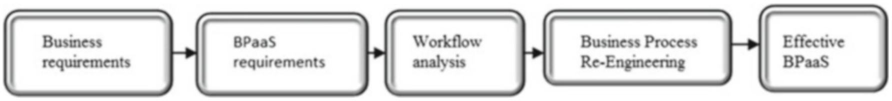

Fig. 6.4 BPaaS process scenario

certain business utility services such as billing and shipping. BPaaS can be a part of internal cloud services as well as external services from different cloud vendor types such as public, hybrid and virtual private. Gandhi [37] addresses some of the key questions that need to be addressed:

- What are the key attributes of BPaaS services to negotiate and gain new business strategies?
- How can BPaaS partnering services accelerate new businesses?
- What are the implications if we don't act now?

These are some of the key strategies and business analysis to be considered for designing BPaaS services. We should be able to use and transfer knowledge gained in business strategies and business process re-engineering and enterprise-wide applications. Figure 6.4 provides a process for capturing and designing BPaaS.

As shown in Fig. 6.4, we should be able to identify and extract business processes and business-related functions as candidate for BPaaS from business requirements capturing process. The second step is to conduct a detailed workflow and task analysis for each suitable BPaaS service. The third step is to conduct business process re-engineering (BPR) for each service which aims to identify ROI, business needs analysis, market analysis and business negotiation strategies for each task that is identified in the workflow analysis. Finally, conduct business effective analysis which interlinks internal and external cloud environment.

BPaaS's most important aspect of the service is to integrate scattered and embedded business rules together in many organisations. Often business rules are scattered and some embedded in different places within the organisations. Therefore, organisations have difficulties in dealing with constant change and evolution of new businesses. BPaaS will also act as business rule management system (BPMR).

6.4 Business Requirements Engineering Process and Framework

Businesses are striving through tough market competition to deliver value-driven products and services. The pace of business delivery has rapidly changed since well-established business practices, nature of business service with advancement and demand for technology-based business services such as e-commerce, e-government, Web services and cloud services. People are looking for value for

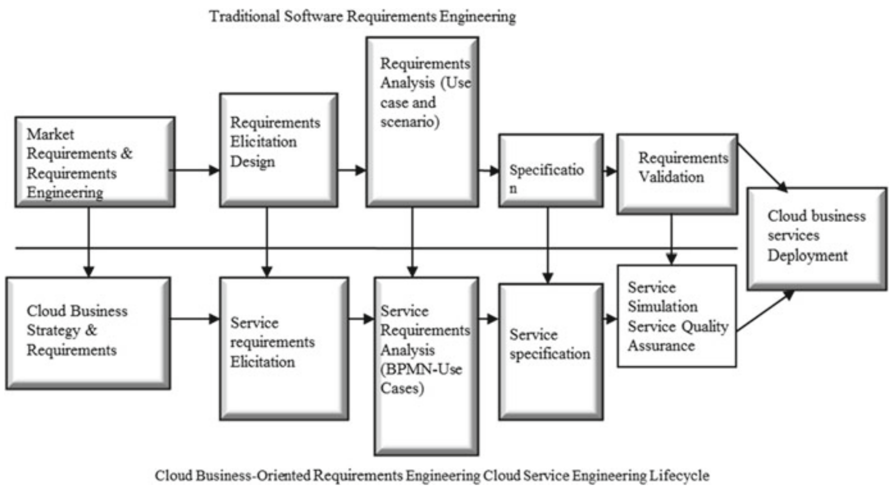
Traditional Software Requirements Engineering
Market Requirements & Requirements Engineering
Requirements Elicitation Design
Requirements Analysis (Use case and scenario)
Specificatio n
Requirements Validation
Cloud business services Deployment
Cloud Business Strategy & Requirements
Service requirements Elicitation
Service Requirements Analysis (BPMN-Use Cases)
Service specification
Service Simulation Service Quality Assurance
Cloud Business-Oriented Requirements Engineering Cloud Service Engineering Lifecycle

Fig. 6.5 Cloud business-oriented requirements engineering

money as well as automated results (self-driven services). Cause [38] discusses a concept known as PRAISED which is defined as follows:

- P → Productivity gains
- R → Reduced cost
- A → Avoided cost
- I → Increased revenue
- S → Service-level improvements
- E → Enhanced quality
- D → Differentiation in the marketplace

Cause [38] argues that many companies force technologies to be sold as their way of improving business value without understanding of business and market needs. Cause [38] has also proposed a feature-driven development (FDD) approach to identifying business need to drive business value as it captures required features of a business and a product. Our approach to identifying BPM using PRAISE model will enhance BPM to drive market and business values. Figure 6.5 shows cloud business-oriented requirements engineering which compares with classical requirements engineering process.

As shown in Fig. 6.5, classical market requirements process needs to be used for conducting business requirements for cloud services which will include business strategies, identifying business services requirements, market analysis and ROI. The second phase is the requirements elicitation and specification which aims to identify stakeholders and conduct requirements analysis and validation which will derive service requirements elicitation, evaluation and validation. This phase will also derive business process modelling using BPMN, and business process simulation will form the basis for service requirements validation. The final process will deliver hand-picked candidates for business services requirement.

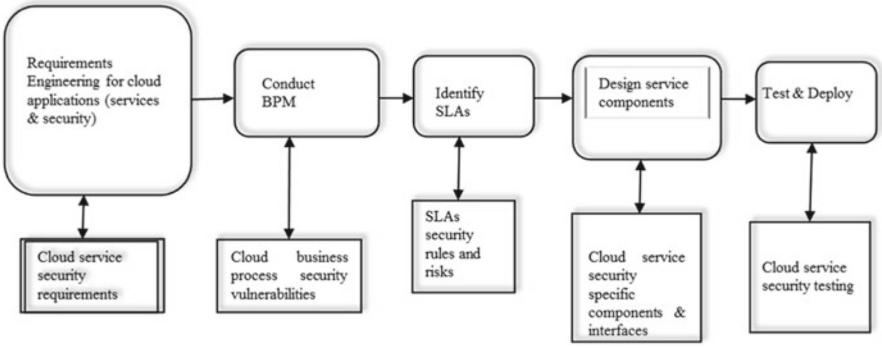
Requirements Engineering for cloud applications (services & security)
Conduct BPM
Identify SLAs
Design service components
Test & Deploy
Cloud service security requirements
Cloud business process security vulnerabilities
SLAs security rules and risks
Cloud service security specific components & interfaces
Cloud service security testing

Fig. 6.6 Cloud service security development process with built-in security

Software security has emerged to build security in from requirements through to testing. Security assessment and analysis needs to be applied for each phase of the life cycle [39]. Software engineering has established techniques, methods and technology over two decades. However, due to the lack of understanding of software security vulnerabilities, we have been not successful in applying software engineering principles when developing secured software systems. Therefore, software security can't be added after a system has been built as seen in today's software applications. However, the issue here is to apply software security techniques to cloud services. Services are application system and therefore we should be able to apply those techniques to develop cloud services with built-in security.

Figure 6.6 shows a process model for the development of cloud services with built-in security. As shown in the diagram, the cloud development process model consists of a number of phases such as RE for cloud, conducting BPM modelling and specification (using BPMN 2 standard and BPEL), identifying and specifying SLAs, building software security in, designing services and testing and deploying.

As part of the cloud service requirements engineering process, we can apply software security engineering techniques all identified cloud services. This includes using security analysis tree and various other techniques specified by Ramachandran [39]. The second step is on identifying BPM (business process modelling) which should include software security analysis for each business process identified to allow us to identify potential security threats. This has been illustrated in Fig. 6.6 which starts with service requirements and business requirements (as shown in Fig. 6.5) as the input to conduct service security analysis using techniques such as Secure Quality Requirements Engineering (SQUARE) and Microsoft Secure Development Lifecycle (SDL). The outcome of this process should yield a set of cloud services security requirements with clear indication of software security issues. The second phase is to conduct business process management during this process should identify a set of business process requirements with security vulnerabilities.

The third phase is to identify service-level agreements (SLAs) which should derive a set of security specific rules. It is also a well-known best practice that eliciting and

Table 6.1 Cloud security risk analysis framework

Service layer	Known types of security threats and attacks on the cloud service that will affect your network	Weighting factors for requirements prioritisation – High = 10; Medium = 5; Low = 1
SaaS (Software as a Service)	DDoS (distributed denial of service attack)	8
	Data stealing	3
	Wrapping attack	4
	Accountability attack	4
	Man in the middle attack	6
	Botnet attack	7
PaaS (Platform as a Service)	SQL injection	6
	SSL attack	3
	Spoof attack	5
IaaS (Infrastructure as a Service)	Blackout/outage	1
	Malware injection attack	3

validating service-level requirements early can save as much as 70 % of the overall test and development costs. Typically, SLA refers to a part of service contracts defining performance attributes, message passing constraints, problem management, customer duties, warranties, disaster recovery, service termination agreements and required local and international laws etc., all of which can be embedded as part of the WSDL specifications. In the context of business-oriented requirements, we need to identify SLA with regard to B2B, B2C and business process and operational constraints. This allows services to make decision on acquiring new businesses. This can further be classified into new and existing business services, customer-driven services, market-driven services, corporate-level services and enterprise-level services. In general, we can define a good business process as

$$\text{Business Process} = \text{Business Rules} + \text{Process}$$

that results in simple processes, higher agility, trust, business integration and better risk management. This will also help business processes to define service trust which is the higher form of business quality as part of QoS performance characteristics. Building trust is the basic means of creating a branding which has been historically successful for major business across the world. Cloud security risks analysis should also be part of this process to identify risk associated with each security and business requirement. Therefore, we propose a framework for conducting security risk assessment. This is shown in Table 6.1, a risk analysis framework which can be used to systematically analyse cloud security risks. The framework provides a comprehensive structure for analysing cloud security risks. This framework consists of service layers and their type of service security attacks that are well known. For each of those security attributes, we need to assign a weighting factor from 10 to 1. The weighting factor 10 (high) has higher risks, 5 (medium) and 1 (low). At the SaaS level, the well-known security risks are DDoS, data stealing, wrapping attack, accountability attack etc.

At the PaaS level, the well-known attacks are SQL injection, SSL attack and spoof attack. At the IaaS level, the well-known attacks are blackout and malware attacks. These lists are not limited to security risks shown in our framework which are commonly known and the discovery of such security risk identification should continue to grow as we gain more user experiences. The above weighting factor for prioritising security requirements is the average of total score against its known frequency of threats, loss of business days (in terms of technical challenges associated to recover), financial loss and predictability. Ramachandran [39] discusses more detailed approaches to vulnerability analysis.

The next phase is on service design which starts with business and service requirements in order to design cloud service components, service interfaces and architecture. During this stage, we need to identify security-driven approach to design of interfaces, message descriptions and handing vulnerabilities that are identified in the previous phase. The final phase is on cloud testing and deployment. During this phase, the main aim is to identify security test strategies such as penetration testing, attack tree testing and other forms of testing. Numerous test strategies have been discussed by Ramachandran [39].

To help manage business process requirements, we have identified a generic enterprise requirements framework (ERF) as shown in Fig. 6.7. The concept of enterprise requirement is based on IT service management, business process management and software development. The main aim is to identify business goals, service concept, change management, organisational rules, enterprise economics, business analysis and software development. *Business analysis* can be defined as a set of tasks, knowledge and techniques that are required to identify business needs and to determine solution to business problems. The solutions often include system development, software development, organisational change and process improvement [40].

The ERF, as presented in Fig. 6.7, consists of three major categories:

- Customer requirements aim to identify service needs, business goals and business types. This further classified into B2B, B2C and C2C business types. Secondly, it aims to identify service requirements and, thirdly, to identify governance requirements.
- Market analysis aims to identify clear rationale for a business service and to analyse return on investment strategies. This further classified into industry strategies, opportunities, competitor analysis and business assets.
- Investment analysis aims to identify required systems, services and infrastructures. The application system refers to identifying cloud infrastructure services, content management services and service types such as SaaS, PaaS and IaaS. This further classified into identification of business application systems; *dynamic scaling* is the key basic rationale behind elasticity, the ability of a cloud to be able to add and remove capacity as and when it is required. This can also be referred as elastic scaling. Secondly, to identify infrastructure services refers to management services required to manage IaaS. Thirdly, to identify service security rationale, risk analysis, availability and resiliency is the ability to withstand security attacks and vulnerability.

The ERF framework provides a structured approach to capturing enterprise requirements. The ERF can also be used to document enterprise-wide requirements as it provides a template. This should also identify peak user performance metrics,

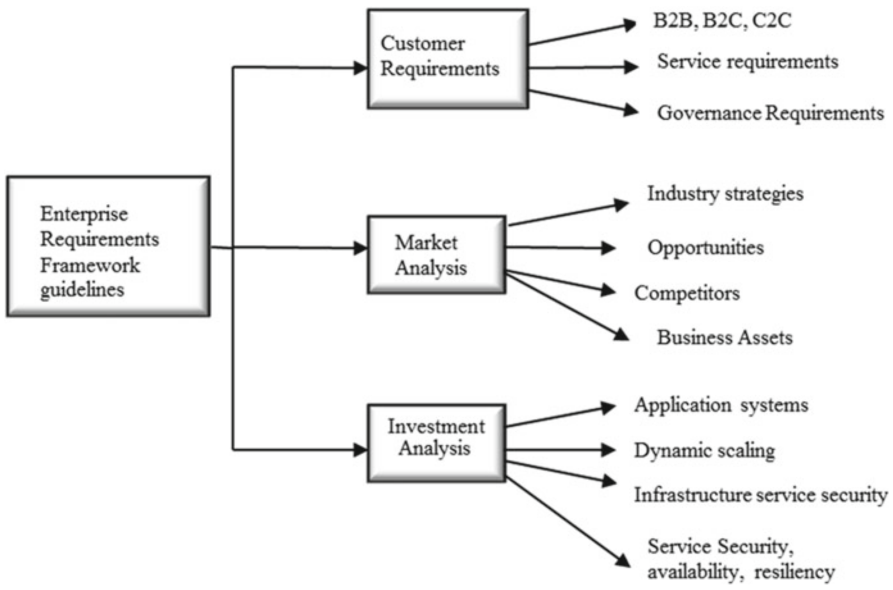

Fig. 6.7 Enterprise requirements framework

capacity planning, security and privacy, availability, response time, hours of operation, pay-per-use calculations, server load, load balancing and cloud management.

6.5 Design of Service Components

Component models and their architecture provide a framework for system composition and integration. A generic component model that is presented in this chapter provides a unique concept of two distinct set of services: *provide and requires*. Software components are the basic unit of artefact that supports service composition with the cloud computing architecture and its environment. However, each development paradigm and application demands customisable and extendable component architectures that suit the needs of their applications. Each Web service component interface is mapped onto different ports within architectural layers to request for services and offer services as and when required at runtime.

The aim is to map business requirements onto a service component that can be designed and implemented. A *service component* can be defined as the one that configures a service implementation. A service component model (UML-based service model) is shown in Fig. 6.8 which reflects service component design principle with a number of plug-in-type interfaces that allow to connect other service components, service provider type of interfaces (IServiceInterface1, 2 etc.) and IServiceContract interface which is a unique concept in our design that allows you to build and reuse business rules. The other types of interface include EntryPort, RejectedMessagePort and ExitPort. These interfaces reflect WSDL descriptions and can be automatically generated.

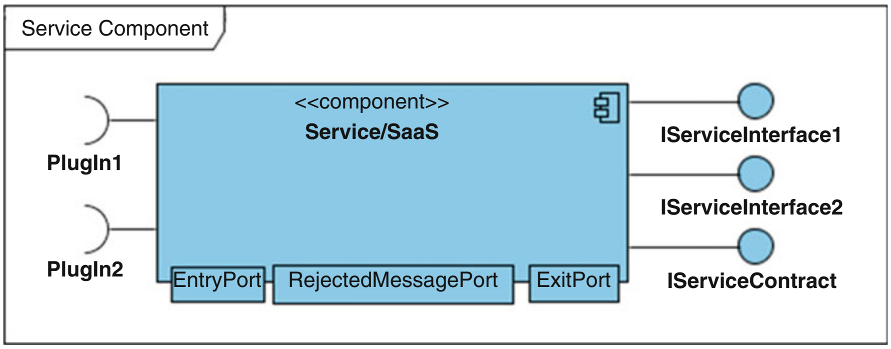

Fig. 6.8 Component model for SaaS

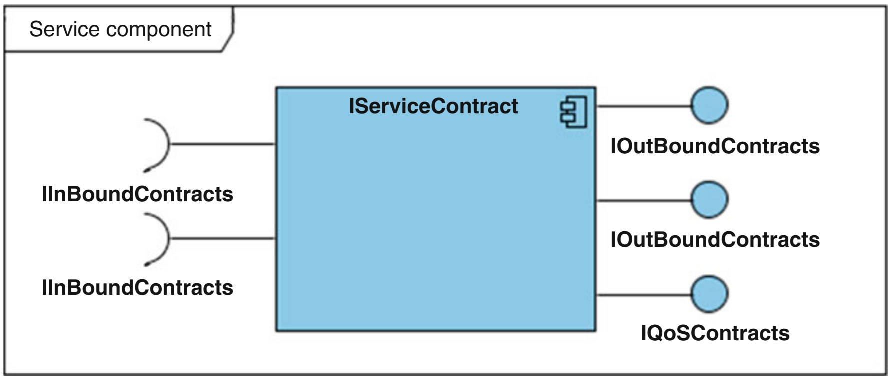

Fig. 6.9 Component model for service contract interface

The service contract interface IServiceContract is a complex class as it allows us to build component rules and be able to reuse them in another service implementation where the similar design contract applies. Due to its nature of complexity, we have designed a separate service component as shown in Fig. 6.8. The service contract component model provides plug-in interfaces such as IInBoundContracts which allows a service component to take business contracts/rules as input to the component, whereas the provider interface such as IOutBoundContracts provides business contract services to other service components. The IQoSContracts service provides services contracts on quality of service rules that are embedded within the service component implementation (Fig. 6.9).

The service component modelling and design provides a systematic approach to building cloud service components to allow on-the-fly configuration, to discover new business services and to be able to connect and disconnect service compositions. *Service composition* is one of the key principles of service design which can't be achieved without a component-based approach. The design principle of component interface allows service flexibility, elasticity and scalability. A service composition is defined as the development of customised services by discovering, integrating and executing existing services. Design of service composition is not only to consume services but also to provide services. Cloud service orchestration layer and its principle can also be addressed and achieved using service composition when services are designed as components based on the model as shown here.

Service composition and orchestration allows service-level reuse to happen. *Service reuse* is a notion of designing services as generic as possible to be reused in another service invocation. Designing services for reuse is based on SOA design principles:

- Loose coupling is to limit dependency between service consumers and service providers. This can be achieved by service interface design which has been part of a service component model as discussed.
- Autonomy is the key principle that enables service reuse. This can be achieved by designing services that can manage their own resources as database and legacies and to maintain by themselves without depending on other services. Service autonomy facilitates service adoption, scalability, QoS, SLA and virtualisation.
- Statelessness is the property of a service to have a context, but it will not have any intermediary state waiting for an event or a callback.
- Granularity has been a prominent design principle of reuse. A large granularity of service component which is self-autonomous can yield higher level of service reuse through service composition. However, a balance must be struck when designing service components and interfaces.
- Composability is the process by which services are combined and integrated to provide a comprehensive and composite service. This principle is also the key to achieving cloud orchestration. A composite service consists of an aggregation of services that can produce another reusable service (s).
- Discoverability is an important means of mandating service time (design time reuse and runtime discoverability) notion when designing service components so that component can be called on when required. Service component interface concept allows components to be discovered and connected.

Designing reusable services can save cost as it is a well-known benefit of reuse. Cost reduction is one of the key aspects of cloud computing which aim to reduce cost for consumers by allowing pay-per-use cost model. The design rationale and service component model discussed in this section will help to improve cloud service reuse experiences.

6.6 Case Study: Amazon EC2

Amazon has three main businesses that are consumer business, seller business and IT infrastructure business. Firstly, let's look at initial business requirements set out by Amazon to create a new cloud as a new business venture. It is aimed to build a

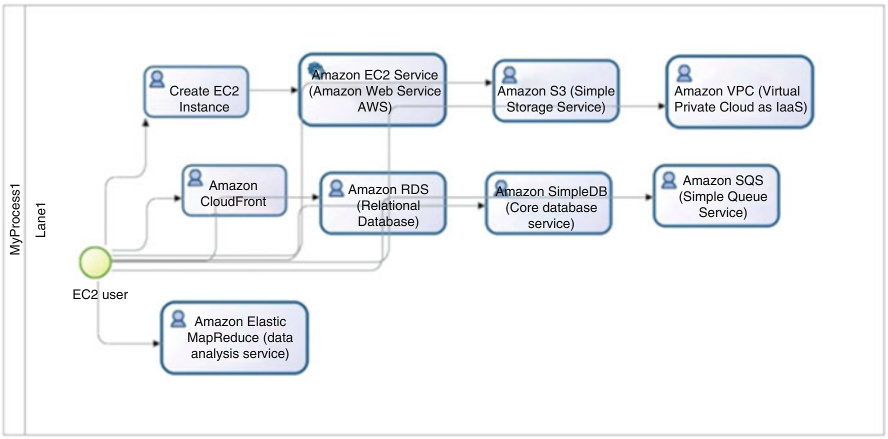
MyProcess1
Lane1
Create EC2 Instance
Amazon EC2 Service (Amazon Web Service AWS)
Amazon S3 (Simple Storage Service)
Amazon VPC (Virtual Private Cloud as IaaS)
Amazon CloudFront
Amazon RDS (Relational Database)
Amazon SimpleDB (Core database service)
Amazon SQS (Simple Queue Service)
EC2 user
Amazon Elastic MapReduce (data analysis service)

Fig. 6.10 Amazon Web services (business process modelling)

powerful cloud with features supporting scalability, failure resilient and enterprise applications including (EC2 2012 [41]):

- Elastic and scalable means users can increase or decrease computational power and other resources within minutes and are charged per use.
- Flexible means users have the choice to choose type of OS, platforms, multiple instances and applications packages.
- Designed for use with other Amazon Web services such as Amazon S3 (a simple storage service), RDS (relational DB services), SimpleDB and Amazon SQS (simple queue service).

These are the examples of non-functional requirements. There are more than 100 business processes, also known as functional requirements, identified from this study which are of typical nature such as account creation, pay-per-service metre, resource management and usage, billing and payment, data storage and maintenance and security and privacy related. Some of the currently offered Amazon Web services are, as part of the AWS, shown in Fig. 6.10 and explained as follows:

- Higher level business processes for Amazon EC2 which consists of composite business services such as RDS, MapReduce, S3, SimpleDB, VPC and SQS.
- Each of these business services can be decomposed into a number categories of business services such task-oriented, infrastructure-oriented, and business service-oriented.

We have developed a number of business services using Bonita software for business process modelling using BPMN notation.

The business process model design tool which is used in this project is Bonita OpenSolution-v5.5 (BOS 5.5). Bonita Open Solutions 5.5 is not only for modelling but we can also conduct process simulations and debugging the process. We can also conduct a range of business process modelling tasks such as service, users, call activity, script, abstract, send and receive. The final simulation process graphs are displayed in another GUI tab. To run the simulation with Bonita, we need to complete three major steps such as:

- Define the process.
- Manage the resources.
- Load profiles.

After completion of the three processes, we then should be able to generate reports of the designed process. We can generate graphs against various process and performance parameters such as execution time, time to completion, response time and raise alarm to study any intrusion during a specific time period.

Amazon S3 (Simple Storage Service) provides a simple Web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web. It provides a discoverable WSDL document describing service operations that can be implemented using RESTful HTML as well as SOAP RPC interfaces. In this experiment, we have attempted to describe its basic functionality using a subset of the available services. Basic executable SOA business models were created based on assumptions made from information provided by online Amazon AWS documents. The Amazon S3 Web Service is just one piece of entire Amazon AWS SOA structure. Other than discoverability, none of the SOA Design concepts can really be applied to the Amazon S3 service on its own. Some of the AWS business services are identified as follows (EC2 2012 [41–42]):

- Amazon S3 (Simple Storage Service) provides a simple Web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web.
- Amazon EC2 (Elastic Compute Cloud) is a Web service that provides resizable compute capacity in the cloud.
- Amazon CloudFront is a Web service for content delivery. It integrates with other Amazon Web Services to give developers and businesses an easy way to distribute content to end users with low latency, high data transfer speeds and no commitments.
- Amazon Route 53 is a highly available and scalable DNS service designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications.
- Amazon RDS (Relational Database Service) is a Web service that makes it easy to set up, operate and scale a relational database in the cloud.
- Amazon SimpleDB (Simple Database Service) is a Web service providing the core database functions of data indexing and querying in the cloud.

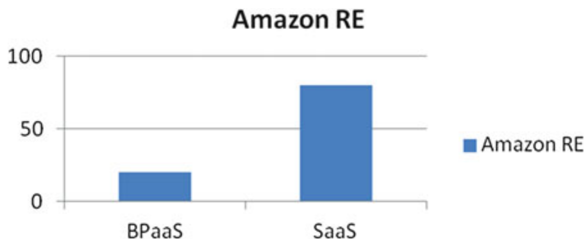

Fig. 6.11 Amazon BPaaS requirements

- Amazon SQS (Simple Queue Service) is a reliable, highly scalable, hosted queue for storing messages as they travel between computers.
- Amazon SNS (Simple Notification Service) is a Web service that makes it easy to set up, operate and send notifications from the cloud.
- Amazon Elastic MapReduce is a Web service that enables businesses, researchers, data analysts and developers to easily and cost-effectively process vast amounts of data.

These services have been considered as a whole to meet the multiple SOA Design criteria by being business-driven, enterprise-centric, loosely coupled, discoverable, stateless and flexibly contractable, and they promote vendor neutrality. The services are provided by Amazon but they can be accessed by any language running on virtually any platform. They are highly scalable and the pricing structure is set up on a cost-per-use basis. Services can be scaled almost instantly when needed and reduced just as fast providing the best of both worlds for businesses, on-demand access without the associated overhead and the delay that would otherwise be required for local on-site implementation. Figure 6.11 shows a bar chart of 100 business processes, out of which we have discovered about 20 BPaaS processes, which is about 20 %.

This is an interesting outcome for our research, in particular, how many BPaaS requirements that can be extracted to evaluate business process service exclusively. BPaaS has a growing strength in making cloud a success with respect to business as a service.

6.7 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. This project has explored some of the process described using Amazon EC2 case study, and we have discovered that there are 20 % of the service requirements that belong to BPaaS as it is a growing business entity for cloud services.

References

1. Wang, L., Laszewski, V.G.: Scientific cloud computing: early definition and experience. http://cyberaide.googlecode.com/svn/trunk/papers/08-cloud/vonLaszewski-08-cloud.pdf (2008)
2. Creeger, M.: Cloud computing: an overview. Distributed computing. ACM Queue. http://queue.acm.org/detail.cfm?id=1554608, June 1, 2009
3. Aoyama, M., et al.: Web services engineering: promises and challenges. In: ICSE'02, Orlando, 19–25 May 2002
4. Bertolino, A., et al.: Audition of web services for testing conformance to open specified protocols. In: Stafford, J., et al. (eds.) Architecting Systems with Trustworthy Components. Springer, Berlin/New York (2006)
5. Bias, R., Cloud Expo Article, Cloud Computing: Understanding infrastructure as a service. Cloud Comput. J. http://cloudcomputing.sys-con.com/node/807481. January 2009
6. Chesbrough, H., Spohrer, J.: A research manifesto for services science, Special issue on services science. CACM **49**(7), 30–87 (2006)
7. Cobweb: http://www.cobweb.com/ (2009)
8. Curbera, F.: Component contracts in service-oriented architectures, Special issue on service-oriented computing. IEEE Comput. **40**(11), 74–80 (2007)
9. Clarke, R.: User requirements for cloud computing architecture. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, 17–20 May 2010
10. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, Upper Saddle River (2005)
11. Farrell, J., Ferris, C.: What are web services? (Special issue). CACM **46**(6), 31 (2003)
12. Khaled, L.: Deriving architectural design through business goals. Int. J. Comput. Sci. Inf. Secur. (IJCSIS) **7**(3), 20–26 (2010)
13. Helbig, J.: Creating business value through flexible IT architecture, Special Issue on service-oriented computing. IEEE Comput. **40**(11), 80–89 (2007)
14. IaaS, Cloud computing world forum. http://www.cloudwf.com/iaas.html (2010)
15. IThound Video Whitepaper. http://images.vnunet.com/video_WP/V4.htm (2010). Accessed Feb 2010
16. Lakshminarayanan, S.: Interoperable security service standards for web services, IT pro. IEEE CS Press USA (2010)
17. Nano, O., Zisman, A.: Realizing service-centric software systems, Special issue on SoC. IEEE Softw. **24**(6), 28–30 (2007)
18. Naone, E.: Computer in the cloud, technology review. http://www.technologyreview.com/Infotech/19397/?a=f (2007)
19. NIST: http://csrc.nist.gov/groups/SNS/cloud-computing/index.html (2009)
20. PaaS. Types of PaaS solutions http://www.salesforce.com/uk/paas/paas-solutions/ (2010)
21. Papazoglou, P.M., et al.: Service-oriented computing: State of the art and research challenges, Special issue on service-oriented computing. IEEE Comput. **40**(11), 38–45 (2007)
22. Ramachandran, M.: Software Components: Guidelines and Applications. Nova, New York (2008)
23. SaaS: SaaS http://www.saas.co.uk/ (2009)
24. Science Group, 2020 Science Group: Toward 2020 science, tech. report, Microsoft. http://research.microsoft.com/towards2020science/downloads/T2020S_Report.pdf (2006)
25. Serugendo, G., et al. (eds): Self-organisation: paradigms and applications. In: Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering. Springer, Berlin/New York (2004)
26. Taiyuan, S.: A flexible business process customization framework for SaaS. In: WASE International Conference on Information Engineering, Taiyuan, 10–11 July 2009
27. Tyagi, S.: RESTful web services. http://www.oracle.com/technetwork/articles/javase/index-137171.html (2006)

28. Venkataraman, T., et al.: A model of cloud based application environment. Int. J. Comput. Sci. Inf. Secur. (IJCSIS) **7**(3) (2010)
29. Verizon: http://www.zdnet.co.uk/news/cloud/2010/10/08/the-cloud-lessons-from-history-40090471/. October 2010
30. Vouk, M.A.: Cloud computing – issues, research and implementations. J. Comput. Info. Technol. (CIT) 16, 40–45 (2008)
31. Wilson, C., Josephson, A.: Microsoft office as a platform for software + services. Archit. J. (13). www.architecturejournal.net. 98–102 (2007)
32. Weiss, A.: Computing in the clouds. http://di.ufpe.br/~redis/intranet/bibliography/middleware/weiss-computing08.pdf, December 2007
33. Yang, J.: Web service componentisation. Commun. ACM **46**(10), 35–40 (2003)
34. Zhang, L-J., Zhou, Q.: CCOA: Cloud Computing Open Architecture. In: IEEE International Conference on Web Services, Bangalore, 21–25 September 2009
35. Linthicum, D.: Application design guidelines for cloud computing. InfoWorld. http://www.infoworld.com/d/cloud-computing/application-design-guidelines-cloud-computing-784?page=0,0. November (2009)
36. Oh, S.H., et al.: A reusability evaluation suite for cloud services. In: Eighth IEEE International Conference on e-Business Engineering. IEEE CS Press USA (2011)
37. Gandhi, B.: Business Process as a Service (BPaaS) delivered from the cloud. http://thoughtsoncloud.com/index.php/2011/12/business-process-as-a-service-bpaas-delivered-from-the-cloud/. December (2011)
38. Cause, G.: Delivering real business value using FDD. http://www.methodsandtools.com/archive/archive.php?id=19. Accessed April 2012
39. Ramachandran, M.: Software Security Engineering: Design and Applications. Nova Science, New York, ISBN: 978-1-61470-128-6. https://www.novapublishers.com/catalog/product_info.php?products_id=26331 (2012)
40. Longo, T., Hass, K., Cannon, D.: ITIL, business analysis and the enterprise requirements hierarchy. http://h10076.www1.hp.com/education/ITIL_BusAnalysis_Enterprise_Req_Hierarchy.pdf (2012)
41. EC2: http://aws.amazon.com/ec2/ (2012). Accessed April 2012
42. What is Cloud Computing – A complete engineering of design and implementation of cloud computing. http://www.keendirect.com/blog/cloudcomputing/. Accessed April 2012