Chapter 8
Architecturally Significant Requirements Identification, Classification and Change Management for Multi-tenant Cloud-Based Systems

Muhammad Aufeef Chauhan and Christian W. Probst

Abstract Involvement of numerous stakeholders in cloud-based systems' design and usage with varying degrees of nonfunctional requirements makes Architecturally Significant Requirements (ASRs) identification and management a challenge undertaking. The aim of the research presented in this chapter is to identify different types of design-time and run-time ASRs of the cloud-based systems, provide an ASRs classification scheme and present a framework to manage the requirements' variability during life cycle of the cloud-based systems. We have used a multifaceted research approach to address the ASRs identification, classification, and change management challenges. We have explored findings from systematic as well as structured reviews of the literature on quality requirements of the cloud-based systems including but not limited to security, availability, scalability, privacy, and multi-tenancy. We have presented a framework for requirements classification and change management focusing on distributed Platform as a Service (PaaS) and Software as a Service (SaaS) systems as well as complex software ecosystems that are built using PaaS and SaaS, such as Tools as a Service (TaaS). We have demonstrated applicability of the framework on a selected set of the requirements for the cloud-based systems. The results of the research presented in this chapter show that key quality requirements of the cloud-based systems, for example, multi-tenancy and security, have a significant impact on how other quality requirements (such as scalability, reliability, and interoperability) are handled in the overall architecture design of a cloud-based system. It is important to distinguish tenant-specific run-time architecturally significant quality requirements and corresponding cloud-based systems' components so that run-time status of the

M.A. Chauhan (✉) · C.W. Probst
Department of Applied Mathematics and Computer Science (DTU Compute), Technical University of Denmark, Kongens Lyngby, Denmark
e-mail: muac@itu.dk

C.W. Probst
e-mail: cwpr@dtu.dk

M.A. Chauhan
Software and Systems Section, IT University of Copenhagen, Copenhagen, Denmark

M. Ramachandran and Z. Mahmood (eds.), *Requirements Engineering for Service and Cloud Computing*, DOI 10.1007/978-3-319-51310-2_8

tenant-specific architecture quality requirements can be monitored and system configurations can be adjusted accordingly. For the systems that can be used by multiple tenants, the requirements change management framework should consider if the addition or modification (triggered by a specific tenant) of a quality requirement can impact quality requirements of other tenants, and whether or not a trade-off point should be introduced in the architecture (corresponding to the requirements). The trade-off point can also be referred as a variability point, that is, a compromise has to be made among the number of quality requirements and only some of the requirements can be satisfied. System analysts and software architects can use the proposed taxonomy and the management framework for identifying relevant quality requirements for multi-tenant cloud-based systems, for analyzing impact of changes in the requirements on the overall system architecture, and for managing variability of the architecturally significant requirements.

Keywords Cloud computing · Platform as a service (PaaS) · Software as a service (SaaS) · Architecturally significant requirements (ARSs) · Requirements classification · Requirements change management · Architecture quality

8.1 Introduction

Cloud computing's utility and service provisioning model offers on-demand scalability and flexible acquisition of computing and storage resources [6]. The cloud resources are offered as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [18]. IaaS provides virtualization of underlying hardware infrastructure, whereas PaaS and SaaS utilize IaaS for providing platforms for cloud-enabled software development or on-demand software systems for end users. Cloud computing adoption can be broadly classified into three categories: (i) Utilizing Infrastructure as a Service (IaaS) [43] cloud environments (as a hosting platform) to deploy software applications. (ii) Migrating existing applications to the cloud to offer the applications as Software as a Service (SaaS) following pay-per-use model [7, 16, 17]. (iii) Developing new SaaS applications using IaaS and PaaS [43] cloud resources.

Development of the cloud-based systems for each of the above-mentioned purposes have associated challenges in terms of Architecturally Significant Requirements (ASRs) identification, analysis and management. Nonfunctional requirements that can have a significant impact on architecture of a software system are referred as ASRs [29]. Each cloud-based system can have a specific set of ASRs, which are more relevant to that system. In the cloud-based systems that use IaaS as a mean to acquire flexible and on-demand infrastructure resources, requirements such as scalability, elasticity, and security are important. For the systems that are to be migrated from old infrastructure to the cloud, the requirements such as interoperability, security, and privacy are more important [16]. The ability of the selected IaaS and PaaS clouds to support the future enhancements in the system are critical for

developing new applications as well as migrating existing applications on the cloud [7, 17]. Some of the ASRs are equally important for different types of the cloud-based systems. For example, as cloud-based systems are aimed to serve many tenants, a characteristic that is referred as multi-tenancy [10] is critical for each of the IaaS, PaaS, and SaaS systems. Moreover, each tenant of the system can have its specific design-time and run-time architecturally significant requirements, including but not limited to security, privacy, availability, scalability, elasticity, and portability [18, 47]. Hence, managing different quality requirements for different tenants is also important. Specific types of cloud-based systems can have additional ASRs, for example, the systems that provision Tools as a Service (TaaS) need to support semantic and process-centric integration [19, 45]. Furthermore, as the data and services are hosted on geographically distributed locations, the cloud-based systems have to comply with additional constraints and regulatory requirements that can directly or indirectly impact architecture of the systems. Last, but not the least, the ASRs can change during life cycle of a cloud-based system. The changes can be either because of involvement of the new stakeholders or modifications in the requirements of the existing stakeholders.

To adequately address the above-mentioned challenges, there is a need to have a specialized approach for identification, classification, analysis, and management of the design-time and run-time architecturally significant requirements of the cloud-based systems and for variability management of the requirements. In particular, we aim to address the following objectives:

- Discuss important Architecturally Significant Requirements (ASRs) of the cloud-based systems, different dimensions of the requirements and the impact that the requirements can have on difference life cycle phases of the cloud-based systems (i.e., system design, system instantiation, system operation, and system evolution). The discussion on ASRs and their respective dimensions can facilitate analysis of the run-time and design-time architecture quality of a cloud-based system.
- Provide a classification scheme to group the ASRs into different categories based upon their impact on the life cycle phases of the cloud-based systems. The classification scheme can help to identify the requirements that should be focused during each phase (including the systems' deployment and operational phases with respect to the multi-tenancy configurations).
- Propose a quality requirements management approach so that the requirements corresponding to the specific tenants can be managed and their impact on each other can be analyzed when existing requirements are changed or modified, or the new requirements are added. The proposed management approach can facilitate to keep track of the changes in the requirements and to control the architecture quality of a cloud-based system in terms of inclusion of the desired ASRs in a specific cloud-based system.

This chapter is organized as follows. Section 8.2 provides an insight to the ASRs for the cloud-based systems. Section 8.3 explores the relation of the ASRs with

multi-tenancy quality of the cloud-based systems. Section 8.4 describes a classification scheme that can be used for classifying the ASRs into different groups using the presented classification parameters. Section 8.5 presents a probabilistic analysis method to analyze the impact of the included ASRs on overall architecture quality of the systems. Section 8.6 describes the related work and Sect. 8.7 concludes this chapter.

8.2 Architecturally Significant Requirements of the Cloud-Based Systems

Architecturally Significant Requirements (ASRs) play a critical role in architecture design, development, and adoption of a software system [29]. The ASRs' impact on a software system raises the need to incorporate the ASRs at early stages of the software architecture design. If the ASRs are not analyzed and anticipated during initial phases of architecture design, a major architecture refactoring may be needed during later stages, which can result in multifold increase in development cost of a software system [29]. Hence, it is important to analyze different types of the ASRs (that can be important for the cloud-based systems) and their impact on different parts of the systems. A summarized view of the ASRs for the cloud-based systems is presented in Fig. 8.1. The details of the ASRs critical for the cloud-based systems are discussed in the following subsections.

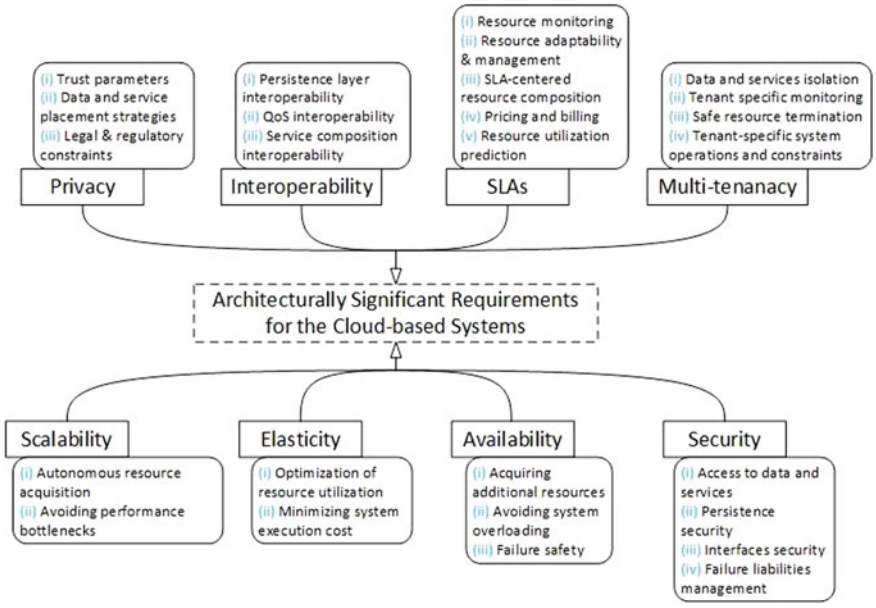

Fig. 8.1 An overview of Architecturally Significant Requirements (ASRs) for the cloud-based systems

8.2.1 Scalability

Scalability is one of the fundamental quality requirements of the cloud-based systems. Scalability supports on-demand provisioning of a cloud-based system by acquiring additional resources, as the number of users using the system grow [18]. Scalability needs for the cloud-based systems can be classified into two main groups. (i) Scalability requirements associated with identifying the system usage and bottlenecks so that a prediction (an estimation) can be made for the needed resources [35]. (ii) Autonomous resource acquisition requirements along with resource acquisition rules so that the resources from the public, private, or hybrid clouds can be acquired on-demand [5, 67].

The most commonly used system monitoring metrics are active profiling of CPU and identification of bottlenecks associated with response time using heuristics algorithms [35]. The cloud resources can be acquired from private, hybrid, or public clouds depending upon constraints on the data and the service. For more competitive and cost-effective resource acquisition approaches, different auction schemes such as Modified Vickery Auction (MVA) and Continuous Double Auction (CDA) can be adopted for sufficient resource availability and unsufficient resources availability, respectively [67]. Vertical scaling (when scalability patterns are adopted in multiple layers of a cloud-based system) and horizontal scaling (when scalability patterns are adopted in only one layer of a cloud-based system) can also be adopted in the cloud-based systems using private, hybrid, and public IaaS clouds [5]. Therefore, the scalability requirements for the cloud-based systems should consider all the above-mentioned factors.

8.2.2 Elasticity

While scalability handles acquisition of additional resources, elasticity deals with optimizing the resource acquisition process so that the extra IaaS resources can be disposed off when not needed [39, 41]. The elasticity requirements can be associated with different types of Quality of Service (QoS) parameters. Observing a time lapse between a request for a particular operation and response of the request is one of the frequently used QoS parameters [39]. Throughput of the layers and data retrieval time is another QoS parameter that is considered for elasticity [51]. Minimizing execution cost of a cloud-based system is also an important parameter to measure elasticity of the system [30]. Last but not the least, providing lowcost computing cycles is an important dimension to consider for achieving elasticity in a PaaS system [52].

8.2.3 Availability

Availability quality requirements guarantee that a cloud-based system and its constituting services are available for utilization as specified in a Service Level Agreement (SLA) [18]. Availability quality characteristics in the cloud-based systems can be achieved in the following ways: (i) Acquiring additional infrastructure resources for hosting the system's services and data for redundant deployment to avoid complete system failures [26]. (ii) Avoiding overloading of the system's resources by replicating the components and services to distribute load among the replicated resources [66]. (iii) Achieving increased application performance by replicating the computing resources to distribute computing intensive tasks and workflows [4, 37].

There are a number of additional constraints that need to be examined corresponding to the availability requirements. The specific location constraints on the IaaS cloud resources that can be used for hosting the system's components (e.g., computing and storage resources available on the IaaS cloud nodes or geographical regions where the physical infrastructure is available) should be examined [26]. The nature of the required availability approaches, i.e., active or passive approaches should be considered [4]. Moreover, the requirements associated with the questions such as whether to make the cloud monitoring mechanism an integral part of the system or to carry out the monitoring via external monitoring agents should also be considered [37].

8.2.4 Security

As the cloud-based systems are accessible via Internet, security becomes an essential quality requirement of the cloud-based systems [18]. The security quality characteristics of the cloud-based systems can be classified into four categories: (i) Access to the cloud-hosted data and services. (ii) Security of the persisted data on the cloud. (iii) Security of the APIs through which the cloud services are exposed to the external world. (iv) Security liabilities of the cloud providers and cloud-hosted Virtual Machines (VMs).

Each of the security quality requirements categories can be further broken down into a number of sub-quality requirements with respect to the nature and types of security attacks that can target a cloud-based system. To restrict the users so that they can have access to the system only according to the desired privileges, different types of the authorization requirements can be incorporated, such as authorization based upon users' roles (e.g., if a specific user is authorized to perform certain operations in the system or can have access to a specific type of data) or users' hierarchy in the users access tree structure (e.g., administrators and super users have more privileges than normal users) [18]. Quality requirements associated with data persistence on the cloud deal with data confidentiality and integrity [55].

The confidentiality requirements for the persisted data deal with threats to the stored data, undesired use of the stored data and availability of the stored data. There can also be data encryption requirements for using different types of encryption algorithms [25, 32], e.g., using ElGamal public key cryptosystems [23]. The requirements for protection against undesired or illegal use of data can require embedding certain types of auditing schemes for data usage history [21, 63]. The requirements concerning the security of the Application Programable Interfaces (APIs) encompass protection from code-centric or SQL injection attacks, hijacking of user sessions, and XML/SOAP wrapping or flooding attacks (the attacks in which huge volume of XML data are sent to APIs to fail the access control and authentication mechanisms) [1]. The requirements for cloud providers security liabilities include handling of plausible service deniability, anonymizing data, and service indexes, introducing intermediate security services to protect direct access to the cloud-hosted data and catering oblivious routing of the data [14]. Protecting the internal application services and providing standards-based end point abstractions for secured communication among the services are also important security considerations [59]. The integrity requirements encompass inclusion of Byzantine Fault Tolerance approaches in the persistence services of the cloud-based systems [3].

8.2.5 Privacy

Privacy requirements of the cloud-based systems are closely related to the security requirements. Privacy on the cloud means that the data is stored and processed on the cloud as defined in privacy specifications. The privacy quality requirements can be classified into three categories: (i) The requirements for specifying trusted cloud parameters and identifiers (that can be used to capture stakeholders' privacy constraints and to select the cloud resources according to the specified constraints). (ii) The requirements for data storage and service placement strategies corresponding to the privacy constraints. (iii) The privacy requirements to comply with legal and regulatory constraints.

The requirements for identification of parameters that can characterize trusted cloud services (and when the services should be opened to remote users) and trusted external services are derived from further refinements of the privacy requirements [8]. Similarly, the privacy specific location parameters that specify where the data and services can be hosted (and which can be driven by legal or regulatory constraints) are a critical part of the privacy requirements [8]. The requirements for services matching process (to facilitate service composition) are a core factor that can influence the design of a cloud-based system. For example, if the end users are allowed to specify their privacy parameters and select the services that are to process the data, a market-oriented cloud-broker infrastructure can be helpful. The cloud-broker can facilitate the users to interact with a cloud market in which the users can specify their privacy constraints and the cloud-broker selects the services with optimal match to the privacy constraints [11]. Hence, in order to incorporate

the security in a cloud-based system, the requirements for the brokerage infrastructure should be considered.

8.2.6 *Interoperability*

Cloud interoperability enables multiple cloud-enabled systems to collaborate with each other [15]. Cloud interoperability can be classified into multiple dimensions as follows: (i) Interoperability of the data persistence layer so that the data can be stored on the cloud resources satisfying location, security, and privacy constraints on the data. (ii) Interoperability among different layers of the cloud service model (i.e., IaaS, PaaS, and SaaS) so that the underlying cloud infrastructure satisfying location, security, and privacy constraints can be selected. (iii) Interoperability of the cloud-hosted services so that the services can be composed at runtime according to the desired Quality of Service (QoS) parameters.

The requirements for the above-mentioned interoperability dimensions can be broken down further into multiple sub-requirements. Cloud services and persistence interoperability requirements deal with how to handle multiple collaborative cloud services, how to select the appropriate persistence store of the data, details on the mechanisms of storing and retrieving the data from the data persistence units, and on-the-fly migrating of the data and services among heterogeneous clouds [31, 57]. The requirements for the brokerage process among the clouds deal with defining and executing the mechanisms for selection of the desired cloud resources via cloud brokerage [64]. Defining interlayer mappings among the cloud resources to categorize the resources that can be replaces with one another is vital for interoperability [15]. The requirements of defining, identifying, and selecting interoperable cloud services can facilitate not only service selection process but also run-time composition of the services [56]. Decentralized deployment of the cloud infrastructure can facilitate satisfaction of the security and privacy constraints on the data and the services, hence the cloud-based systems' requirements for the decentralized deployment should be matched with the infrastructure support of the underlying cloud [50]. The requirements for autonomous selection and composition of the underlying cloud resources and the hosted cloud-based systems' services explore different parameters needed for resources or services identification and the attributes for which the search queries can be run [48]. The selection and composition requirements can also facilitate the resources and services matching and portability of the services among the clouds [49]. These requirements can also be used for discovering and composing heterogeneous cloud services on the fly [68].

8.2.7 Service Level Agreement (SLA) Compliance

The compliance of the cloud-hosted data and services with Service Level Agreements (SLA) between the cloud-resource providers and the cloud-resource consumers is vital; especially when a large number of tenants with varying service quality needs are being served [40]. Hence, the requirements related to SLAs compliance focus on the following dimension: (i) Monitoring requirements for the cloud resources and cloud-hosted services to monitor the quality attributes of interest. (ii) Resources' adaptability and management requirements corresponding to the monitoring parameters. (iii) Service composition requirements for satisfying SLAs. (iv) Billing requirements for managing pricing variability with respect to the SLAs. (v) Requirements for predicting the system's behavior with respect to the run-time quality requirements in order to enable the SLA compliance for unforeseen scenarios.

For the cloud-resource providers to comply with SLAs, the providers have to monitor the resources for quality attributes of interest such as scalability [38]. The monitoring requirements need to be focused on key performance indicators of the system (e.g., elasticity, scalability, and performance) and the monitoring should be nonintrusive so that the monitoring mechanism do not affect normal operations of the system [34]. Adaptability requirements should be focused on quality of service parameters for services transmission and communication environments, and should focus on key performance indicators [34]. SLA requirements should also include requirements associated with availability of the qualified candidate services [53], optimal service composition approaches to be adopted for QoS specific services' composition [46] and requirements for license management of the virtualized cloud resources [12].

For services and data management in the cloud-based systems, the focus of the requirements engineering and management effort should be on characterizing SLA compliance and regulatory requirements for data retention, intercloud migration of the services and data, and confidentiality constraints on the data and the services [42, 60]. The requirements for run-time management of the SLAs (including enforcement of fine-grained SLA compliance policies for managing data and handling run-time services operations, enforcement of data retention policies on data persistence objects, and management of billing corresponding to the run-time quality requirements) are also critical [11]. Moreover, the requirements for resource discovery and monitoring in accordance with SLAs are also important. To conclude, the SLA compliance requirements need to focus on consistency, scalability [20], workload management driven by applications and users behavior [40], monitoring of the resources deployed on different platforms [70], anticipation of the system behavior for desired QoS parameters [27], customization of the monitoring parameters for different types of the systems following users' specifications [13], and optimization of profit margins while satisfying SLAs [9].

8.3 Relationship of the Architecturally Significant Requirements with Multi-tenancy Quality Characteristics

Multi-tenancy quality characteristic (requirement) of the cloud-based systems facilitates secured sharing of the resources among multiple tenants and adoption of the systems with respect to tenant-specific configurations [62]. Multi-tenancy characteristic affects the design of the cloud-based systems from two-different perspectives. First, multi-tenancy determines security to provide isolation among different services belonging to different tenants in a cloud-based system. Services' isolation is also referred as security dimension of multi-tenancy. Second, multi-tenancy determines a specific configuration of a cloud-based system for a specific tenant with respect to the quality requirements discussed in Sect. 8.2. The security requirements of the multi-tenancy can be classified into three broad categories: (i) Isolation among the data and services belonging to different tenants. (ii) Monitoring of the resources for their compliance with QoS parameters and their usage (so that the tenants can be billed accordingly). (iii) Safe termination of the resources once tenant-specific operations are completed so that run-time state of tenant-specific configuration of the system cannot be exploited via a cross tenant attack. Specification and management of the run-time quality requirements for different tenants are determined by the nature of the run-time system's operations and constraints on the data processing services and need for exposure of the data to external systems.

Security requirements of the multi-tenancy focus on the following dimensions. To control access to the multi-tenant systems, the requirements focus on hierarchical Role-Based Access Control (hRBAC) mechanisms or conditional Role-Based Access Control (cRBAC) mechanisms [10]. For hRBAC mechanisms, different users and external systems are grouped into hierarchical clusters of users, and access rights are determined based upon the position of a user or an external system in the hierarchy. For cRBAC mechanisms, the users and systems are granted access to the system to perform a specific operation if all the preconditions are satisfied. The preconditions include not only authentication and authorization but also if the prerequisite operations have been completed and the data needed for the current operations (to be performed) is available. To have a centralized security control mechanism for all the system provided by a particular cloud platform, an aspect-oriented security mechanism can be adopted [2]. The requirements for the aspect-based security control mechanism focus on database requirements to maintain architecture description of the hosted systems and security constraints desired by different tenants, management system requirements to define and integrate security in the hosted systems, and interface requirements through which the security aspects can be integrated in the hosted systems.

The focus of handling generic quality requirements for multi-tenant cloud-based systems is on feature-based resource management, cost-based resource optimization, tenant distribution over the resources, and monitoring of the services and

hosted platforms for their compliance with SLAs. The requirements associated with feature-based resource management focus on models that can be used to share instances of the services among the tenants with similar quality requirements [44]. These requirements also focus on resource allocation model to analyze failure cost of wrong service placement strategy and cost of successful service placement strategy in terms of energy footprint and price of the used resources. For distributing tenant-specific resources on hybrid clouds to satisfy the privacy and security constraints, the scheduling and routing algorithms should focus on context of the operations and data requests [24]. For monitoring the deployed resources on the hybrid clouds, observers on all layers of the cloud service and deployment model can be needed [28]. Moreover, to satisfy the run-time performance parameters for SLA compliance, the multi-tenancy requirements should focus on monitoring, scheduling, load balancing and provisioning of the components, and services and data according to available computing resources for each specific tenant [61].

8.4 A Classification Scheme for Management of Architecturally Significant Requirements

Traditionally, architecturally significant quality requirements are classified into two broad categories: (i) design-time quality requirements and (ii) run-time quality requirements [29]. However, in order to organize the requirements for complex software systems, such as cloud-enabled systems, the requirements need to be further classified into sub-groups. The sub-groups facilitate to establish the relationship among different types of the requirements' classes and analyze the impact of changes in the requirements across the sub-groups. In this section, we identify different attributes that can be used to classify the architecturally significant quality requirements of the cloud-based systems into different groups and discuss a selected set of the requirements to explain the classification approach. An overview of the requirements classes and the classification parameters is shown Fig. 8.2 and the details of the sub-classes along with description of the classification parameters are summarized in Table 8.1.

8.4.1 System Management Requirements for Hosted Services and Data

The requirements that can be classified into this group are associated with provisioning of the cloud-hosted services following the desired run-time quality parameters, providing communication among the hosted services, and handling the security and privacy constraints. In the following subsections, we describe the details on classification parameters for different dimensions of the system management requirements.

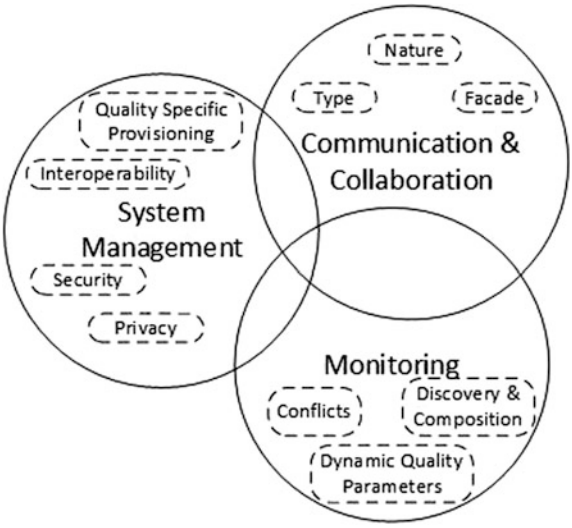

Fig. 8.2 Architecturally Significant Requirements classes and key classification parameters for the cloud-based systems

8.4.1.1 Quality Specific Provisioning

The requirements of the cloud-enabled systems can be classified into *Quality Specific Provisioning* group if the requirements satisfy to the following conditions:

- The requirements defining the parameters for initialization and deployment of the services, data, or a combination of these on the cloud.
- The requirements that can have an impact on run-time behavior of a cloud-based system.
- The requirements that either specify the parameters for SLAs management or deal with the system compliance with the SLAs.
- The requirements classifying the nature and types of the client systems or services that are to interact with a cloud-based system.
- The requirements specifying the nature and type of the end user devices that are to interact with the system.

8.4.1.2 Interoperability and Integration Requirements

The *Interoperability and Integration Requirements* group encapsulates the requirements satisfying the following conditions:

- The requirements that deal with specifying system interfaces. For example, REST-based interfaces, SOAP-based interfaces, or translucent callback interfaces.

Table 8.1 Requirements classes and parameters used for classification

Requirement classes	Classification parameters
Quality specific provisioning	Controlling initialization and deployment
	Effecting run-time system behavior
	Facilitating SLAs achievement
	Dealing with nature and type of client systems and services
	Dealing with end-user devices
Interoperability and integration	Enabling translucent system interfaces
	Governing cloud federation
	Specifying interoperability of the hosted services
Security and privacy	Accessing data and services
	Dealing with multi-tenancy
	Specifying data encryption requirements
	Trusting cloud and the hosted services
	Complying with legal and regulatory requirements
	Handling data and service placement strategies
	Managing liabilities of the cloud-hosted services
Communication and collaboration	Enabling communication and interaction with heterogeneous cloud environments and externally services
	Securing inter service communication
	Providing services interface facades
	Abstracting service end points
Monitoring	Checking compliance with respect to dynamically changeing quality characteristics
	Adapting run-time quality parameters
	Monitoring quality parameters
	Managing run-time quality conflicts
	Optimizing system configuration with respect to quality characteristics
	Enabling autonomous services selection and composition
	Managing services distribution and scheduling mechanisms
	Enabling dynamic cloud resource discovery
	Handling redundant service deployments

- The requirements that handle specification, management, and governance of the federated cloud. That is, what kind of cloud resources should be combined to form a federated cloud, when the cloud federation should take place, and under which conditions specific services from a federated cloud should be selected.
- The requirements which deal with nature and type of the data that should be exchanged among the services. For example, whether XML-based data structures or a language-specific data structures are to be used for interoperability and integration among the services.

8.4.1.3 Security and Privacy Requirements

The *Security and Privacy* group includes the requirements dealing with one or more of the following specifications.

- The requirements dealing with access to the cloud-hosted data and services. These requirements include both authentication and authorization requirements.
- The requirements handling different aspects of the multi-tenancy characteristic of a cloud-based system. For example, whether the tenant-specific service instances should be isolated from each other or not, what quality of service parameters are desired by each tenant, and how multiple services can be composed to meet QoS constraints of a specific tenant.
- The requirements associated with security of the data. For example, whether the data should be encrypted or not, what kind of encryption algorithm should be applied on the data, and how the data should be persisted.
- The requirements specifying constrains on the trusted execution of the services on the cloud. These requirements can include the characteristics of the trusted cloud environments and the parameters to be used for the selection of a trusted cloud environment.
- The requirements specifying legal and regulatory constrains on the systems. For example, for how long the data history of a system should be maintained before permanently deleting the data.
- The requirements specifying the strategies for hosting the services and persisting the data on different cloud environments. For example, a constraint specifying that sensitive data should always be stored on a private cloud in an encrypted format.
- The requirements specifying system liabilities and penalties for cases in which a cloud-based system fails to comply with desired operational conditions. These requirements can also include how to handle the exceptional cases in which desired security and privacy constraints could not be satisfied.

8.4.2 Communication and Collaboration Requirements

Communication and collaboration among the services and hosting cloud environments, when a cloud-based system is operational, are a critical run-time property of the cloud-based systems. The requirements associated with *communication and collaboration* group can be classified based upon the following properties.

- The requirements that specify the parameters for communication and interaction among heterogeneous cloud environments as well as interaction among the services hosted on the heterogeneous cloud environments.
- The requirements defining nature and type of the communication. For example, whether the communication is to be encrypted or not, or whether a specific

communication protocol (e.g., publisher subscriber pattern) should be followed to exchange the notifications and data.
- The requirements defining services' facade.
- The requirements defining types of interfaces and signatures of interfaces for services' end points.

8.4.3 Monitoring Requirements

Services monitoring requirements handle observation of the run-time system behavior and adaptation based on the analysis of the monitoring parameters. The requirements can be classified into this category based upon the following properties.

- The requirements monitoring system compliance with respect to the dynamically changing architecture quality attributes. For example, security and privacy requirements of the data can be different for different types of tenants and can vary according to the nature and type of the data.
- The requirements dealing with monitoring of specific run-time quality attributes of a cloud-based system.
- The requirements specifying mechanisms to handle conflicting run-time quality requirements.
- The requirements associated with optimization methods based upon the systems' monitoring metrics.
- The requirements specifying concrete methods to achieve the quality attributes. For example, services distribution and scheduling mechanisms.
- The requirements specifying details of resource discovery and composition.

8.5 A Probabilistic Analysis Method to Analyze Impact of Changes in Architecturally Significant Requirements

The different types of the architecturally significant requirements discussed in Sects. 8.2 and 8.3 can be classified into different groups as discussed in Sect. 8.4. A change in one of the requirements can impact one or more of the related or dependant requirements. As a result, to manage and track changes in the requirements, a systematic approach is required that can be used to analyze the impact of the changes with reference to the nature and type of relationships that exist among different requirements and the degree of impact that the requirements can have on each other. In this section, we describe a probabilistic analysis method to analyze impact of the changes in architecturally significant requirements.

The requirements can be related to each other with different types of relations to represent dependency, composition, complementation, contradiction, and proportionality relationships. These relationships are explained as follows:

- ***Dependency relation*** represents a relationship among the architecturally significant requirements such that a requirement B is dependent upon a requirement. In other words, in order to satisfy the requirement B, the system first has to satisfy the requirement A.
- ***Composition relation*** represents a relationship among the architecturally significant requirements such that the composition of a number of sub-requirements is needed in order to satisfy a higher order requirement.
- ***Complementation relation*** represents a relationship among the architecturally significant requirements such that incorporation of a requirement A in the system can complement a requirement B, i.e., incorporating the requirement A in the system can make it easy to incorporate the requirement B.
- ***Contradiction relation*** represents a relationship among the architecturally significant requirements such that a requirement A is contradictory to a requirement B, i.e., it is not possible to completely satisfy both the requirements (A and B) in the system at the same time. In other words, either the requirements A and B are mutually exclusive or a trade-off has to be made for degree of satisfaction of each requirement in the system.
- ***Inverse proportionality relation*** represents a relationship among the architecturally significant requirements such that the degree of achieving a requirement A can have an inverse impact on the degree of achieving a requirement B. For example, if a cloud-based system satisfies a requirement A 90% of the time, the requirement B can only be satisfied 10% of the time, and vice versa.

Each of the defined relationships has a probabilistic value, which represents the degree of strength of the relationship between two requirements. For example, a probabilistic value of 50% with ***dependency relation*** between the requirements A and B shows that the requirement B is at least 50% dependent on the requirement A. Figure 8.3 shows the symbolic representation of the relationships that can exist among the requirements and describes an example scenario. The relationships can be used for not only to establish a link among the requirements of the cloud-based systems but also to analyze impact of changes in one of the requirements on the other system requirements.

Figure 8.3 shows the details of the proposed approach for the analysis of two types of high-level Architecturally Significant Requirements (ASRs), i.e., high response time and security of a cloud-based system. The probability score associated with the different relations in the diagram is based upon the expert opinion of the authors of this chapter. An alternative approach to the expert opinion can be to seek input from the stakeholders of the system and calculate the probability score using weighted averages (e.g., for cases in which some of the stakeholders have more control on the requirements engineering and management than others). The relations to achieve the change management shown in Fig. 8.3 are linked to each

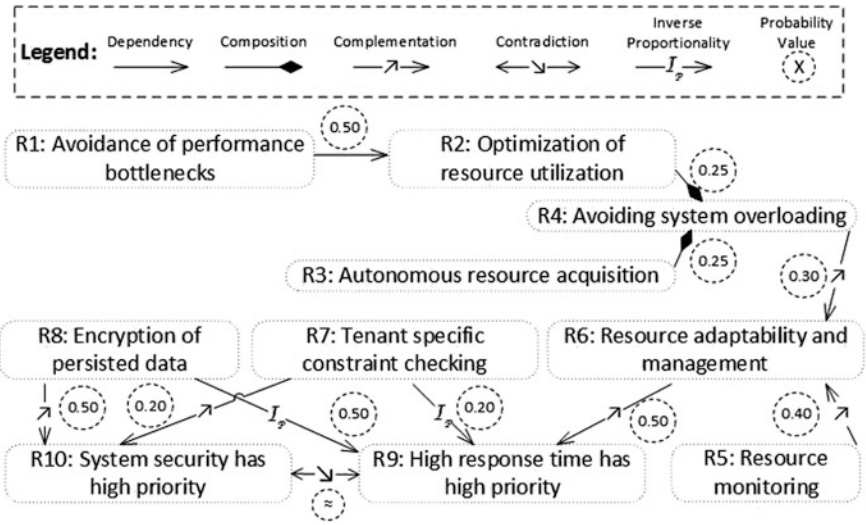

Fig. 8.3 Relationships of the probabilistic analysis method and an example application scenario

other in the following manner. The requirement *R2* for optimization of the resource utilization is dependant on the requirement *R1*, which deals with avoidance of the performance bottlenecks of the systems. A probability value score of 0.50 is assigned, indicating that *R2* is 50% dependent upon *R1*. To avoid system overloading, the requirement *R4* is composed of R2 (for optimization of the resource utilization) and *R3* (for autonomous resource acquisition). Each of the requirements *R2* and *R3* complement 25% to the achievement of *R4*, whereas remaining 50% is handled by the requirement *R4* itself. The requirement *R6* is associated with resource adaptability and management. *R6* can be complemented by *R4* by delegating system overloading avoidance capability to *R4*. The probability score of 0.30 is assigned, which indicate that 30% of the *R6* responsibility can be handled by *R4*. Similarly, the requirement *R5* handling resource monitoring contributes 40% to *R6* and subsequently *R6* contributes 50% to the requirement *R9*. *R9* describes that high response time is of higher priority for the system than security. The requirements *R7* and *R8* deal with security in terms of tenants specific security constraints checking and encryption of persisted data, respectively. *R7* and *R8* are inversely proportional to *R9* because incorporating more security measures in the system decreases response time. The probability values associated with inverse proportionality relationships have negative contributions, i.e., *R7* decreases response time by 20% and *R8* decreases response time by 50%. The requirement *R10* described that higher system security is desired. *R7* and *R8* have positive contribution to *R10* and can facilitate to incorporate *R10* in the system by the factor of 20% and 50%, respectively. As shown in Fig. 8.3, *R9* and *R10* are contradictory requirements and both cannot be fully satisfied at the same time, hence a trade-off has to be made to decide to which extend each of *R9* and *R10* should be incorporated in the system.

The quality of architecture in terms of ASR can be calculated in the following manner. A maximum score 1 can be assigned to each quality requirement indicating that it can be completely incorporated in the system. For dependency, composition and complementation relations, probability value assigned with the relation is added to the target requirement's score (for the requirements incorporation into the system). For example, in Fig. 8.3 the requirement *R1* contributes to the requirement *R2* with value 0.50. This means 50% of *R2* can be achieved by incorporating *R1* in the system. Similarly, the requirements *R4* is composed of the requirements *R2* and *R3* each with a factor of 0.25. This means that each of *R2* and *R3* contributes 25% for incorporation of *R4* in the system. For inverse proportionality, the probability value score with the relation is subtracted from the target requirement's score. For example, in Fig. 8.3, the requirements *R7* and *R8* are inversely related with the requirement *R9* by factors of 0.20 and 0.50, respectively. As a result, if *R7* and *R8* are incorporated in the system, the probability for satisfaction of *R9* is the system is only 0.30 (1–0.20–0.50). Only of the requirements associated with each other with a contradiction relation cannot be fully satisfied by the system. As a result, either only one of the requirements should be considered to be part of the system or a trade-off has to be made among the requirements for their respective degree of incorporation in the system. Table 8.2 lists the relations that can exist between the requirements versus their contributions to a system's architecture quality.

The probability value scores associated with the relations (as shown in Fig. 8.3) can be calculated in the following manner: (i) If there is a consensus among the stakeholders on the probability score of a relation, the agreed probability score can be assigned directly. (ii) If the stakeholders cannot reach a consensus, then the weighted averages for a relation k can be taken according to the following formula, in which k corresponds to identifier of each relation and N corresponds to the total number of stakeholders involved in the decision making process of the relation k.

$$Probability\,Score(k) = \frac{\sum_{i=1}^{N} Weight(k)_i \times Score(k)_i}{N}$$

The values of each of the $\text{Weight(k)}_\text{i}$ and Score(k)_i can range between 0 and 1, depending upon the value of a specific stakeholder's weight for a specific relation and the probability score assigned by the stakeholder to the relation. The weighted average of a relation k specifies probability score value of the relation k between two requirements. For example, let us assume that there are two stakeholders of the

Table 8.2 Contribution of the relations between ASRs to overall system quality

Relation	Relation's contribution
Dependency	Positive
Composition	Positive
Complementation	Positive
Contradiction	Mutually exclusive (trade-off required)
Inverse proportionality	Negative

requirements *R1* and *R2* and they cannot reach a consensus on the probability score of the dependency relation between *R1* and *R2*. If first stakeholder has 0.75 weight (stakes on the requirements) and choose a probability value score of 0.80 for the relation, and second stakeholders has 0.50 weight and choose a probability value score of 0.25. The probability score of the dependency relation between the requirements *R1* and *R2* can be calculated as follows:

$$\frac{0.75(Weight_1) \times 0.80(Score_1) + 0.50(Weight_2) \times 0.25(Score_2)}{2(N)} = 0.36$$

The weighted averages for all the relations among the requirements in case of disagreements can be calculated in the similar manner.

The relations and probabilities assigned with the relations (as shown in Fig. 8.3) can also be used to analyze impact of the changes in the requirements on overall architecture quality of a cloud-based system. Addition of new requirements or removal of existing requirements from the probabilistic analysis model can result in addition or removal of the relations and changes in the respective probabilities of the relations. Modification in the requirements can require reanalysis of the probabilities assigned to the relations. Addition, removal or modification in some of the requirements can require recalculation of the whole probabilistic analysis model. Hence, the probabilistic analysis method presented in this section provides not only the traces among different types of the requirements but also the types of the traces (in terms of the relations) and strength of the traces (in terms of the probability values), which in turn provides a mechanism to evaluate overall requirements quality of a cloud-based system.

8.6 Related Work

A number of studies have focused on requirements for cloud-based system and variability management of the requirements. Ramachandran [54] has proposed a business-oriented requirements engineering approach for the cloud-based systems. The proposed approach takes market requirements as a baseline for requirements engineering and cloud business strategy. In subsequent stages, the requirements are elicited, and cloud services are designed and tested. The business analysis is consisted of tasks, knowledge, and techniques that can be used to identify business needs and solution to the business needs. Dey and Lee [22] have proposed a requirements elicitation and variability management approach. The presented approach proposed that the requirements elicitation should focus on the social, environmental, and economic context. The proposed variability management approach focused on recording and analyzing identified conflicts, identifying key changes for the system redesign, identifying users expected behavior for different states of the system and identifying most feasible set of requirements for the cloud-based systems. Iankoulova and Daneva [33] have presented a systematic

review of the studies discussing cloud computing security requirements. The review has identified access control, integrity, auditing, privacy, and nonrepudiation as commonly reported security requirements. Kalloniatis et al. [36] have analyzed the cloud deployment scenarios with respect to security and privacy requirements. The authors have argued that the security analysis should be performed with respect to organizational needs and cloud-deployment models.

Rimal et al. [58] have described the architecture requirements for the cloud-based systems in terms of provider requirements, enterprise requirements, and user requirements. The requirements describe cloud-service models, cloud deployment models, cloud quality characteristics, and billing requirements. Wind and Schrodl [65] have provided a comparison framework of the requirements engineering models for the cloud-based systems. Four well-known software development models including V Model, Rational Unified Process, Extreme Programming, and Volere are explored in terms of their suitability for requirements engineering to analyze cloud offerings with respect to suppliers and customers viewpoints, orchestration, and application components. Zardari and Bahsoon [69] have presented a goal oriented requirements engineering approach to support cloud adoption. The presented approach focuses on matching-desired goals with features of the cloud service providers. After features selection, matches are analyzed for risks and finally cloud services with least risks are selected for utilization.

The related work discussed in this section focus on higher level enterprise and business requirements for the cloud-based systems, and describe approaches to match the requirements with available cloud services. On the contrary, the research presented in this chapter focuses on multiple dimensions of the architecturally significant requirements for the cloud-based systems, relations among the requirements and an approach to manage changes in the requirements.

8.7 Conclusions

A clear understanding of the Architecturally Significant Requirements (ASRs) and relation among different dimensions of the ASRs is critical to achieve quality in architecture of the cloud-based systems. The biggest challenge for architecting quality in the cloud-based systems is to have an understanding of the details to which the ASRs should be explored and how the changes in one type of the ASRs can impact the other ASRs. In this chapter, we have presented a set of core ASRs of the cloud-based systems and have explored the requirements' relationships with the multi-tenancy quality characteristic of the cloud-based systems. The ASRs are classified into three classes including system management requirements, communication and collaboration requirements, and monitoring requirements. We have identified key classification attributes for each of the requirements classes. For example, monitoring requirements handle dynamic monitoring of the quality parameters, identification of the run-time quality conflicts, and dynamic discovery and composition of the system services (or components) to maintain the run-time

quality of a cloud-based system. We have also presented a probabilistic analysis method to analyze the impact of the ASRs on each other as well as to analyze impact of change in one of the requirements on other related and dependant requirements. The presented analysis method utilizes five different types of the relations (i.e., dependency, composition, complementation, contradiction, and inverse proportionality) to evaluate impact of the changes.

We foresee that the presented research can be used by the researchers and practitioners to identify core quality characteristics of the cloud-based systems and to use the identified dimensions of the ASRs to elicit the requirements' details. The presented probabilistic analysis method can be used to control run-time system configuration to achieve desired quality in a cloud-based system. In future, we tend to explore the presented research for its suitability for managing SaaS product lines. We also plan to extend the presented research to provide the traces among the live components of the cloud-based systems so that the quality of on-the-fly system composition for multi-tenant cloud-based systems can be determined.

Acknowledgements Part of the research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement no. 318003 (TRE_SPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.

References

1. Al-Aqrabi, H., Liu, L., Xu, J., Hill, R., Antonopoulos, N., Zhan, Y.: Investigation of it security and compliance challenges in security-as-a-service for cloud computing. In: Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 15th IEEE International Symposium on. pp. 124–129. IEEE (2012).
2. Almorsy, M., Grundy, J., Ibrahim, A.S.: Tossma: A tenant-oriented saas security management architecture. In: Cloud computing (cloud), 2012 ieee 5th international conference on. pp. 981–988. IEEE (2012).
3. AlZain, M.A., Soh, B., Pardede, E.: A byzantine fault tolerance model for a multi-cloud computing. In: Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on. pp. 130–137. IEEE (2013).
4. An, K., Shekhar, S., Caglar, F., Gokhale, A., Sastry, S.: A cloud middleware for assuring performance and high availability of soft real-time applications. Journal of Systems Architecture 60(9), 757–769 (2014).
5. Ardagna, C.A., Damiani, E., Frati, F., Rebeccani, D., Ughetti, M.: Scalability patterns for platform-as-a-service. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. pp. 718–725. IEEE (2012).
6. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM 53 (4), 50–58 (2010).
7. Babar, M.A., Chauhan, M.A.: A tale of migration to cloud computing for sharing experiences and observations. In: Proceedings of the 2nd international workshop on software engineering for cloud computing. pp. 50–56. ACM (2011).
8. Belimpasakis, P., Moloney, S.: A platform for proving family oriented restful services hosted at home. Consumer Electronics, IEEE Transactions on 55(2), 690–698 (2009).

9. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future generation computer systems 28(5), 755–768 (2012).
10. Bernabe, J.B., Perez, J.M.M., Calero, J.M.A., Clemente, F.J.G., Perez, G.M., Skarmeta, A.F. G.: Semantic-aware multi-tenancy authorization system for cloud architectures. Future Generation Computer Systems 32, 154–167 (2014).
11. Buyya, R., Pandey, S., Vecchiola, C.: Cloudbus toolkit for market-oriented cloud computing. In: Cloud Computing, pp. 24–44. Springer (2009).
12. Cacciari, C., Mallmann, D., Zsigri, C., D'Andria, F., Hagemeier, B., Rumpl, A., Ziegler, W., Martrat, J.: Sla-based management of software licenses as web service resources in distributed computing infrastructures. Future Generation Computer Systems 28(8), 1340–1349 (2012).
13. Calero, J.M.A., Aguado, J.G.: Monpaas: an adaptive monitoring platformas a service for cloud computing infrastructures and services. IEEE Transactions on Services Computing 8(1), 65–78 (2015).
14. Vera-del Campo, J., Pegueroles, J., Herna´ndez-Serrano, J., Soriano, M.: Doccloud: A document recommender system on cloud computing with plausible deniability. Information Sciences 258, 387–402 (2014).
15. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: How to enhance cloud architectures to enable cross-federation. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. pp. 337–345. IEEE (2010).
16. Chauhan, M.A., Babar, M.A.: Migrating service-oriented system to cloud computing: An experience report. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on. pp. 404–411. IEEE (2011).
17. Chauhan, M.A., Babar, M.A.: Towards process support for migrating applications to cloud computing. In: Cloud and Service Computing (CSC), 2012 International Conference on. pp. 80–87. IEEE (2012).
18. Chauhan, M.A., Babar, M.A., Benatallah, B.: Architecting cloud-enabled systems: a systematic survey of challenges and solutions. Software: Practice and Experience (2016).
19. Chauhan, M.A., Babar, M.A., Sheng, Q.Z.: A reference architecture for a cloud-based tools as a service workspace. In: Services Computing (SCC), 2015 IEEE International Conference on. pp. 475–482. IEEE (2015).
20. Chen, T., Bahsoon, R., Tawil, A.R.H.: Scalable service-oriented replication with flexible consistency guarantee in the cloud. Information Sciences 264, 349–370 (2014).
21. Daniel, W.: Challenges on privacy and reliability in cloud computing security. In: Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on. vol. 2, pp. 1181–1187. IEEE (2014).
22. Dey, S., Lee, S.W.: From requirements elicitation to variability analysis using repertory grid: A cognitive approach. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE). pp. 46–55. IEEE (2015).
23. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Advances in cryptology. pp. 10–18. Springer (1984).
24. Fehling, C., Leymann, F., Mietzner, R.: A framework for optimized distribution of tenants in cloud applications. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. pp. 252–259. IEEE (2010).
25. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Ina´cio, P.R.: Security issues in cloud environments: a survey. International Journal of Information Security 13(2), 113–170 (2014).
26. Frˆıncu, M.E.: Scheduling highly available applications on cloud environments. Future Generation Computer Systems 32, 138–153 (2014).
27. Garc´ıa, A.G., Espert, I.B., Garc´ıa, V.H.: Sla-driven dynamic cloud resource management. Future Generation Computer Systems 31, 1–11 (2014).
28. Goldschmidt, T., Murugaiah, M.K., Sonntag, C., Schlich, B., Biallas, S., Weber, P.: Cloud-based control: A multi-tenant, horizontally scalable soft-plc. In: Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on. pp. 909–916. IEEE (2015).

29. Gorton, I.: Essential software architecture. Springer Science & Business Media (2006).
30. Han, R., Ghanem, M.M., Guo, L., Guo, Y., Osmond, M.: Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Generation Computer Systems 32, 82–98 (2014).
31. Hassan, M.M., Song, B., Huh, E.N.: A market-oriented dynamic collaborative cloud services platform. Annals of telecommunications-annales des te´le´communications 65(11–12), 669–688 (2010).
32. Huang, W., Ganjali, A., Kim, B.H., Oh, S., Lie, D.: The state of public infrastructure-as-a-service cloud security. ACM Computing Surveys (CSUR) 47(4), 68 (2015).
33. Iankoulova, I., Daneva, M.: Cloud computing security requirements: A systematic review. In: 2012 Sixth International Conference on Research Challenges in Information Science (RCIS). pp. 1–7. IEEE (2012).
34. Inzinger, C., Hummer, W., Satzger, B., Leitner, P., Dustdar, S.: Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems. Software: Practice and Experience 44(7), 805–822 (2014).
35. Iqbal, W., Dailey, M.N., Carrera, D., Janecek, P.: Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Generation Computer Systems 27(6), 871–879 (2011).
36. Kalloniatis, C., Mouratidis, H., Islam, S.: Evaluating cloud deployment scenarios based on security and privacy requirements. Requirements Engineering 18(4), 299–319 (2013).
37. Kanso, A., Lemieux, Y.: Achieving high availability at the application level in the cloud. In: Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on. pp. 778–785. IEEE (2013).
38. Katsaros, G., Kousiouris, G., Gogouvitis, S.V., Kyriazis, D., Menychtas, A., Varvarigou, T.: A self-adaptive hierarchical monitoring mechanism for clouds. Journal of Systems and Software 85(5), 1029–1041 (2012).
39. Kaur, P.D., Chana, I.: A resource elasticity framework for qos-aware execution of cloud applications. Future Generation Computer Systems 37, 14–25 (2014).
40. Kerte´sz, A., Kecskemeti, G., Brandic, I.: An interoperable and self-adaptive approach for sla-based service virtualization in heterogeneous cloud environments. Future Generation Computer Systems 32, 54–68 (2014).
41. Kirschnick, J., Alcaraz Calero, J.M., Goldsack, P., Farrell, A., Guijarro, J., Loughran, S., Edwards, N., Wilcock, L.: Towards an architecture for deploying elastic services in the cloud. Software: Practice and Experience 42(4), 395–408 (2012).
42. Li, J., Stephenson, B., Motahari-Nezhad, H.R., Singhal, S.: Geodac: A data assurance policy specification and enforcement framework for outsourced services. Services Computing, IEEE Transactions on 4(4), 340–354 (2011).
43. Louridas, P.: Up in the air: Moving your applications to the cloud. IEEE software 27(4), 6 (2010).
44. Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B., De Turck, F.: Cost-effective feature placement of customizable multi-tenant applications in the cloud. Journal of Network and Systems Management 22(4), 517–558 (2014).
45. Moser, T., Biffl, S.: Semantic integration of software and systems engineering environments. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42 (1), 38–50 (2012).
46. Nae, V., Prodan, R., Iosup, A.: Sla-based operations of massively multiplayer online games in clouds. Multimedia Systems 20(5), 521–544 (2014).
47. Nidd, M., Ivanova, M.G., Probst, C.W., Tanner, A., Ko, R., Choo, R.: Tool-based risk assessment of cloud infrastructures as socio-technical systems. The cloud security ecosystem. Syngress (2015).
48. Paik, I., Chen, W., Huhns, M.N.: A scalable architecture for automatic service composition. IEEE Transactions on Services Computing 7(1), 82–95 (2014).

49. Paraiso, F., Merle, P., Seinturier, L.: socloud: A service-oriented component-based paas for managing portability, provisioning, elasticity, and high availability across multiple clouds. Computing 98(5), 539–565 (2016).
50. Peifeng, S., Chuan, S., Xiang, Z.: Intelligent server management framework over extensible messaging and presence protocol. Communications, China 10(5), 128–136 (2013).
51. Perez-Sorrosal, F., Patin˜o-Martinez, M., Jimenez-Peris, R., Kemme, B.: Elastic si-cache: consistent and scalable caching in multi-tier architectures. The VLDB Journal—The International Journal on Very Large Data Bases 20(6), 841–865 (2011).
52. Prodan, R., Sperk, M.: Scientific computing with google app engine. Future Generation Computer Systems 29(7), 1851–1859 (2013).
53. Qi, L., Dou, W., Zhang, X., Chen, J.: A qos-aware composition method supporting cross-platform service invocation in cloud environment. Journal of Computer and System Sciences 78(5), 1316–1329 (2012).
54. Ramachandran, M.: Business requirements engineering for developing cloud computing services. In: Software Engineering Frameworks for the Cloud Computing Paradigm, pp. 123–Springer (2013).
55. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Computing (1), 69–73 (2012).
56. Rezaei, R., Chiew, T.K., Lee, S.P., Aliee, Z.S.: A semantic interoperability framework for software as a service systems in cloud computing environments. Expert Systems with Applications 41(13), 5751–5770 (2014).
57. Ribeiro, L.S., Viana-Ferreira, C., Oliveira, J.L., Costa, C.: Xds-i outsourcing proxy: ensuring confidentiality while preserving interoperability. IEEE journal of biomedical and health informatics 18(4), 1404–1412 (2014).
58. Rimal, B.P., Jukan, A., Katsaros, D., Goeleven, Y.: Architectural requirements for cloud computing systems: an enterprise cloud approach. Journal of Grid Computing 9(1), 3–26 (2011).
59. Ryan, J.: Rethinking the esb: building a secure bus with an soa gateway. Network Security 2012(1), 14–17 (2012).
60. Serrano, D., Bouchenak, S., Kouki, Y., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., Sens, P.: Towards qos-oriented sla guarantees for online cloud services. In: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on. pp. 50–57. IEEE (2013).
61. Sousa, F.R., Machado, J.C.: Towards elastic multi-tenant database replication with quality of service. In: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing. pp. 168–175. IEEE Computer Society (2012).
62. Takabi, H., Joshi, J.B., Ahn, G.J.: Security and privacy challenges in cloud computing environments. IEEE Security & Privacy (6), 24–31 (2010).
63. Tari, Z., Yi, X., Premarathne, U.S., Bertok, P., Khalil, I.: Security and privacy in cloud computing: Vision, trends, and challenges. Cloud Computing, IEEE 2(2), 30–38 (2015).
64. Villegas, D., Bobroff, N., Rodero, I., Delgado, J., Liu, Y., Devarakonda, A., Fong, L., Sadjadi, S.M., Parashar, M.: Cloud federation in a layered service model. Journal of Computer and System Sciences 78(5), 1330–1344 (2012).
65. Wind, S., Schro¨dl, H.: Requirements engineering for cloud computing: a comparison framework. In: International Conference on Web Information Systems Engineering. pp. 404–415. Springer (2010).
66. Wu, L., Garg, S.K., Buyya, R.: Sla-based admission control for a software-as-a-service provider in cloud computing environments. Journal of Computer and System Sciences 78(5), 1280–1299 (2012).
67. Wu, X., Liu, M., Dou, W., Gao, L., Yu, S.: A scalable and automatic mechanism for resource allocation in self-organizing cloud. Peer-to-Peer Networking and Applications 9(1), 28–41 (2016).

68. Xu, Z., Mei, L., Liu, Y., Hu, C., Chen, L.: Semantic enhanced cloud environment for surveillance data management using video structural description. Computing 98(1–2), 35–54 (2016).
69. Zardari, S., Bahsoon, R.: Cloud adoption: a goal-oriented requirements engineering approach. In: Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing. pp. 29–35. ACM (2011).
70. Zhang, Y., Zhou, Y.: Transparent computing: spatio-temporal extension on von neumann architecture for cloud services. Tsinghua Science and Technology 18(1), 10–21 (2013).