Continuous Quality Assurance and Optimisation in Cloud-Based Virtual Enterprises

Simeon Veloudis, Iraklis Paraskakis, Andreas Friesen, Yiannis Verginadis, Ioannis Patiniotakis, Alessandro Rossini

Continuous Quality Assurance and Optimisation in Cloud-Based Virtual Enterprises

Simeon Veloudis[1], Iraklis Paraskakis[1], Andreas Friesen[2], Yiannis Verginadis[3], Ioannis Patiniotakis[3] and Alessandro Rossini[4]

[1] South East European Research Centre (SEERC), International Faculty of the University of Sheffield, CITY College, 24 Proxenou Koromila St, 54622, Thessaloniki, Greece
{sveloudis, iparaskakis}@seerc.org
[2] SAP AG, Vincenz-Priessnitz-Strasse 1, Karlsruhe, 76131, Germany
andreas.friesen@sap.com
[3] Institute of Communications and Computer Systems, National Technical University of Athens, Zografou, Athens, 15780, Greece
{jverg, ipatini}@mail.ntua.gr
[4] Department of Networked Systems and Services, SINTEF, P.O. Box 124 Blindern, 0314 Oslo, Norway
alessandro.rossini@sintef.no

Abstract. With the rise of cloud computing, enterprises increasingly rely for their daily operations on heterogeneous externally-sourced cloud services that span different levels of capability. Their IT environment is thus progressively transformed into an ecosystem of intertwined infrastructure, platform, and application services. To effectively manage the ensuing complexity, enterprises are anticipated to increasingly rely on cloud service brokerage (CSB). This work presents a conceptual architecture for a framework which provides solutions with respect to the quality assurance and optimisation dimensions of CSB in the context of virtual enterprises. The framework revolves around three general themes, namely *governance and quality control*, *failure prevention and recovery*, and *optimisation*.

Keywords: virtual enterprises, cloud computing, cloud service brokerage, reference architecture, governance, quality control, failure prevention and recovery, optimisation

1 Introduction

The increasing adoption of cloud computing is altering the way in which IT resources have traditionally been managed and consumed, bringing about significant advantages for enterprises in terms of cost, flexibility and business agility [1,2]. The cloud computing paradigm crucially involves the use of computing resources that are remotely delivered over the Internet as a service, and which are entrusted by users with data, software, and computation. Cloud computing has evolved out of Grid computing [2,3] as a result of a shift in focus from an infrastructure aiming to deliver storage and compute

adfa, p. 593, 2011.

resources, to an economy-based computing paradigm offering a wide range of abstract resources and services [3]. Such a shift is anticipated to impact the manner in which businesses and organisations share skills and core competencies within a distributed collaborative network.

More specifically, activities that take place within the confines of a dynamic multi-institutional virtual enterprise (VE) may involve heterogeneous externally-sourced cloud services that span different clouds and capability levels (IaaS, PaaS, and SaaS [4]). As an example, consider the following scenario (adapted from [5]). An industrial consortium formed to develop a feasibility study for a next-generation supersonic aircraft undertakes a highly accurate multidisciplinary simulation of the entire aircraft. This simulation integrates software components offered as a service by different consortium participants (SaaS offerings). Each component may be operating on a participant's proprietary infrastructure or, alternatively, on infrastructure provisioned as a cloud service (IaaS offering). At the same time, the simulation requires the development of new specialised software components. To this end, the consortium is provisioned the necessary software platform for developing these applications as a cloud service (PaaS offering).

Evidently, the IT environment of such enterprises is progressively transformed into an ecosystem of intertwined infrastructure, platform, and application services, typically delivered by a multitude of diverse service providers. As the number of externally-sourced services proliferates, it becomes increasingly difficult to keep track of when and how services evolve over time, either through intentional changes initiated by service providers, or through unintentional changes, such as variations in service performance and availability. Moreover, it becomes increasingly difficult to accurately predict the potential repercussions that such an evolution has with respect to a service's compliance to policies and regulations, and its conformance to service level agreements (SLAs). In addition, the proliferation of cloud services that offer similar functionality under comparable terms of provision, despite the obvious benefits, is adding to the overall complexity as the onus of discovering suitable alternatives inevitably falls on the service consumer.

In order to effectively deal with this complexity, future enterprises are anticipated to increasingly rely on cloud service brokerage (CSB) [6]. Reflecting frequently-cited views by analysts such as Gartner [7], Forrester [8], and NIST [6], CSB capabilities may be categorised along the following dimensions: (i) Service Discovery, (ii) Service Integration, (iii) Service Aggregation, (iv) Service Customisation, (v) Service Quality Assurance, and (vi) Service Optimisation.

This work reports on the conceptual architecture of a framework which provides solutions with respect to the latter two dimensions of brokerage capability. In particular, we envisage the development of a brokerage framework[1] which provides mechanisms that are organised around the following general themes: (i) *governance and quality control*; (ii) *failure prevention and recovery*, and (iii) *optimisation*. The 1st theme is primarily concerned with checking the compliance of services with declaratively pre-

[1] This framework is being developed as part of the EU-funded project 'Broker@Cloud'.

specified policies concerning the technical, business, and legal aspects of service delivery. It is also concerned with testing services for conformance with their expected behaviour, and with continuously monitoring their operation for conformance to SLAs. The 2nd theme is concerned with the reactive and proactive detection of cloud service failures, and the selection of suitable adaptation strategies to prevent, or recover, from failures. The 3rd theme is concerned with continuously identifying opportunities to optimise service consumption with respect to such goals as cost, quality, and functionality.

The rest of this paper is structured as follows. Section 2 presents a set of requirements for the proposed framework, and Section 3 proposes a reference architecture that meets these requirements; Section 4 discusses the various mechanisms which materialise this reference architecture. Finally, Section 5 presents related work, and Section 6 presents conclusions and future work.

2 Framework Requirements

We argue that the incorporation of mechanisms organised around the general themes outlined in Section 1 will assist cloud service brokers in providing assurances with respect to the reliability and optimality of services. To motivate the need for such mechanisms, this section revisits the example of Section 1 and identifies a number of relevant *functional capabilities* that these mechanisms materialise. Functional capabilities are associated with tasks or activities that transform inputs into outputs. They are high-level, possibly complex, services that are provided by a software system in order to fulfil a user need. In our work, functional capabilities express requirements upon the Broker@Cloud framework which pertain to the general themes of Section 1.

Suppose that a cloud platform (call it CloudX) hosts various services that may be potentially used by the industrial consortium (hereafter referred to as the 'VE'). Let a service provider offer a new application service (call it *fuelConsum_A*) which supports collaborative creation of simulations for the aircraft's fuel consumption.

2.1 Capabilities of the Governance and Quality Control Mechanisms

Before *fuelConsum_A* can be offered to the VE for consumption, a number of onboarding criteria may need to be evaluated to ensure compliance with the VE's business policies (e.g. pricing, availability, response time, deployment infrastructure etc.). This implies a *policy evaluation* capability (hereafter referred to as C1) which must be supported by the platform's governance and quality control mechanisms. Moreover, other checks may verify – for example, through the provision of an automated *functional testing* capability (referred to as C2) – that the programmatic interfaces of *fuelConsum_A* adhere to the specifications of the CloudX platform.

In addition, if *fuelConsum_A* is successfully onboarded and starts being consumed by the VE, its performance must be continuously evaluated with respect to the corresponding SLA. This clearly implies a *monitoring* capability (C3) that must be supported by the platform's governance and quality control mechanisms. The capabilities C1, C2, and C3 are summarised in Table 1.

2.2 Capabilities of the Optimisation Mechanism

Suppose now that, after onboarding *fuelConsum_A*, CloudX's optimisation mechanism identifies an optimisation opportunity with respect to a similar, albeit more expensive, service that the VE is currently consuming (call the latter *fuelConsum_B*). This clearly implies an *optimisation analysis* capability (C4 – see Table 1) which must be supported by the optimisation mechanism. An optimisation opportunity can only be identified relative to a set of consumer-expressed preferences regarding service consumption (e.g. service cost, reputation, etc.); it follows that the optimisation mechanism must also possess the capability to perform consumer *preference analysis* (hereafter referred to as C5). Upon identification of the optimisation opportunity, CloudX may recommend appropriate *optimisation actions*, e.g. the renegotiation of the terms of provision of *fuelConsum_B*, or its replacement by *fuelConsum_A*. This clearly implies that the optimisation mechanism must also possess an *optimisation recommendation* capability (C6).

2.3 Capabilities of the Failure Prevention and Recovery Mechanism

Suppose now that during the consumption of *fuelConsum_B* by the VE, an SLA violation is detected. CloudX's failure recovery mechanism yields appropriate actions aiming at alleviating the effects of the violation, such as the substitution of *fuelConsum_B* with *fuelConsum_A*. This clearly implies a *failure analysis* capability (C7 – see Table 1). In addition, CloudX's failure prevention mechanism *proactively* generates suitable adaptation plans if certain key service attributes (such as, for instance, availability and response time) are deteriorating; these plans aim at averting an imminent failure. For this to be possible, the failure prevention mechanism must possess the capability of incorporating appropriate *failure prevention* rules (C8).

Table 1. Functional Capabilities[2]

Id	Name	Description
C1	Policy Evaluation	Assesses a service's conformance with Broker policies.
C2	Functional testing	Assesses a service's conformance with its specification.
C3	Monitoring	Collects and aggregates service performance data.
C4	Optimisation analysis	Analyses optimisation opportunities.
C5	Preference analysis	Handles and exploits consumer preferences expressed as precise or imprecise criteria.
C6	Optimisation recommendation	Reasons about alternative optimisation actions and recommends the best alternative.
C7	Failure analysis	Generates appropriate failure recovery or prevention actions.
C8	Failure prevention	Expresses rules for proactively identifying service failures.

[2] This is not the full set of functional capabilities identified for the Broker@Cloud framework, but a suitable subset discerned for the purposes of this paper; the full set can be found in [10].

2.4 Platform-neutral Descriptions of Cloud Services

The aforementioned mechanisms must clearly be able to interact with each other, as well as with the underlying cloud service delivery platform. To this end, an additional general theme is discerned, namely *platform-neutral description of cloud services*, which is concerned with the development of platform-agnostic methods for the declarative description of the information required by these mechanisms. For reasons of space, the *declarative* capabilities[3] associated with this theme are not considered here; the interested reader is referred to [9] for a specification of such capabilities.

3 Framework Reference Architecture

Our aim is to develop a framework which will be adoptable by cloud service intermediaries in order to equip their platforms with the functional capabilities of Table 1. This section presents a reference architecture for such a framework. The architecture is kept generic and minimal in order to increase its adoptability. The reference processes that comprise this architecture may be refined (adapted, extended, modified, etc.) by the adopting cloud service intermediaries in order to align them with the concrete architectural requirements of their platforms, providing that: (i) the capabilities that these processes entail are treated as black boxes; (ii) any interdependencies between these capabilities are dealt with by the adopting platform. These reference processes are categorised below relative to the main phases of a service's lifecycle, namely *Service Onboarding*, *Service Operation*, and *Service Evolution*. This allows a clear association between the capabilities of Table 1 and the particular phases of a service's lifecycle in which these capabilities are offered.

3.1 Service Onboarding Reference Processes

A service enters the Service Onboarding phase either when it is initially submitted, or when it is upgraded to a fresh version. The primary focus within this phase is the `Service Assessment` process. This process certifies that a new service, or an existing upgraded one, complies with the relevant onboarding business policies. It comprises two processes, namely *Policy Compliance Evaluation* and *Testing* (see Fig. 1[4]).

The former process checks that the service description is compliant with the relevant CSB business policies; it is thus associated with capability C1 of Table 1. If a service fails to comply, it is deemed unsuitable for deployment and the boundary escalation event *Compliance Failed* occurs. This event captures an inner escalation event in the

[3] Declarative capabilities are specifications of data models or structures.

[4] We use BPMN diagrams to illustrate reference processes and their corresponding mechanisms (see Section 4). BPMN has been chosen because it is intuitive, whilst its modelling power is, according to [11], analogous to that of other comparable modelling languages such as UML 2.0 Activity Diagrams. For reasons of space, and to enhance clarity, the BPMN diagrams have been simplified; their full versions can be accessed in [10]; a BPMN tutorial can be found in [12].

Policy Compliance Evaluation process (see Fig. 5); it triggers, in turn, the *Certification Failed* event which signifies the termination of the onboarding process. The latter process functionally tests the service in order to ensure that its description constitutes an accurate reflection of its behaviour; it is associated with capability C2. The event *Testing Failed* is analogous to the *Compliance Failed* event.

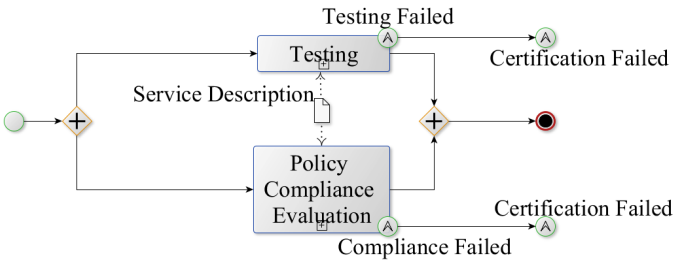

Fig. 1. Service Assessment reference process

3.2 Service Operation Processes

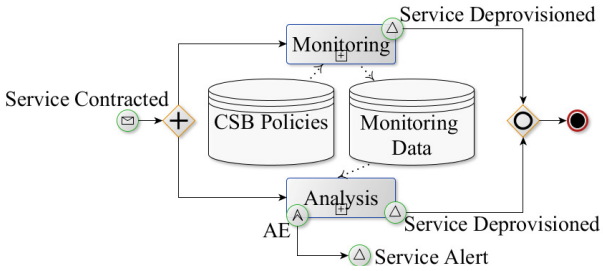

Fig. 2. Service Technical Management reference process

A service enters the Service Operation phase as soon as the first consumer subscribes to it. This phase comprises the *Service Technical Management* process which ensures the abidance of the service's behaviour by the corresponding SLA. It comprises two parallel processes, namely *Monitoring* and *Analysis*. The former process continuously checks whether a set of metrics that characterise the service's behaviour adheres to the SLA; which exact metrics are monitored is determined by the relevant policy in the *CSB Policies* store (see Fig. 2); monitored values are stored in the *Monitoring Data* store. The signal event *Service Deprovisioned* signifies that the service is no longer consumable and suspends monitoring. The *Monitoring* process is associated with capability C3 of Table 1.

The latter process is associated with capability C7. It analyses the monitoring data in order to identify potential behavioural patterns indicative of a service failure. In such a case, the boundary event *AE* (stands for "Analysis Event") occurs which captures a corresponding inner escalation event. This triggers the *Service Alert* event which invokes the *Failure Prevention and Recovery Mechanism* (see Section 3.3).

3.3 Service Evolution Processes

The Service Evolution phase spans throughout the entire service lifecycle. Two main processes are discerned here: *Optimisation Management* and *Failure Prevention and Recovery Management*.

Optimisation Management. This process (see Fig. 3) offers optimisation recommendations to service consumers; Table 2 outlines its constituent processes.

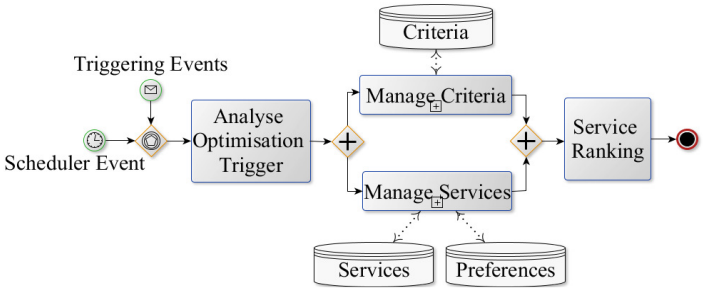

Fig. 3. Optimisation Management reference process

Table 2. Optimisation Management processes

Process	Description
Analyse Optim. Trigger	Strives to identify optimisation opportunities and thus it is directly associated with capability C4 of Table 1. It is initiated periodically (i.e. by the scheduler event of Fig. 3) or by a set of *triggering events* which are assumed to occur when a service is onboarded, deprecated, or when its contract changes; it may also be initiated when a consumer's preferences (regarding the consumption of a service) change.
Manage Criteria	Allows service consumers to express their preferences regarding service consumption with respect to a predetermined set of relevant service attributes or criteria. Preferences are expressed either as precise crisp values (for quantitative criteria, e.g. service response time), or as imprecise fuzzy values and linguistic expressions (for inherently qualitative criteria, e.g. service reputation). This process is directly associated with capability C5.
Manage Services	For each consumed service, collects consumer opinions relative to the imprecise criteria that characterise the service[5]. It then assigns an appropriate characterisation to each such criterion based on the majority of opinions. This ensures that per-service imprecise characterisations are consistent with current consumer opinions. This process is associated with capability C4.
Service Ranking	Discerns services that could be recommended to a consumer for replacing the currently consumed service. To this end, services within the same *functional*

[5] Precise criteria can be measured objectively and are not amenable to subjective opinions.

	category[6] as the one being consumed are ranked according to the current precise and imprecise criteria characterisations, and the consumer preferences. This process is associated with capabilities C4 and C6.

Failure Prevention and Recovery Management. This process recommends appropriate actions for recovering from a service failure, or for averting an impending one; it comprises the processes outlined in Table 3 and depicted in Fig. 4.

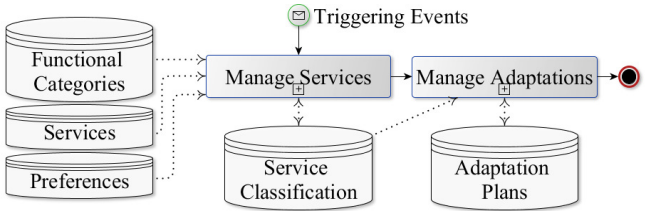

Fig. 4. Failure Prevention and Recovery Management reference process

Table 3. Failure Prevention and Recovery Management processes

Process	Description
Manage Services	Classifies services under appropriate functional categories (e.g. *fuelConsum_A* and *fuelConsum_B* are classified under the category *fuelConsumSimulations*). These categories form the basis of recommending appropriate alternative services in case a service fails or is about to fail. This process is initiated by the same *triggering events* as the ones that initiate the *Analyse Optim. Trigger* process of Table 2; it is associated with capability C7.
Manage Adapt.	Discerns alternative services that could be recommended to a consumer if, for a particular service, the *Service Alert* event occurs (see Section 3.2). It thus ranks services according to their current precise and imprecise criteria characterisations, and the expressed consumer preferences. This process is too associated with capability C7.

4 Framework Mechanisms

This section elaborates on the mechanisms implementing the reference processes of Section 3. Table 4 outlines a number of requirements on these mechanisms. For reasons of space, we confine ourselves to the *Policy Compliance Evaluation* process (see Fig. 5); mechanisms for the rest of the reference processes can be found in [10].

[6] According to their function, services may be categorised into functional categories – e.g. 'map services', 'calendar services', 'fuel consumption simulation services', etc. A functional category thus characterises the *kind* of a service.

Table 4. Architectural requirements on Broker@Cloud mechanisms

Availability	For each identified capability, there must be at least one mechanism specification and an associated mechanism implementation.
Dependency Handling	A mechanism assumed to operate as part of a collection of mechanisms must ensure interoperability with the other mechanisms in the collection.
Flexibility	A mechanism may be replaceable not only by another mechanism but also by an ecosystem of mechanisms which implement the same capabilities as the mechanism under replacement.
Replaceability	Methods must be provided for developing new mechanisms, or new implementations of existing mechanisms.

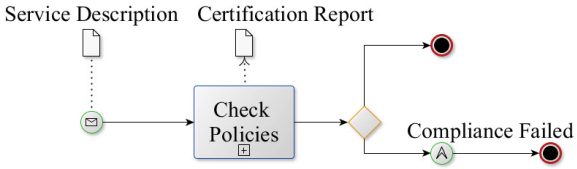

Fig. 5. Internal view of the Policy Compliance Evaluation process

The mechanism implementing the *Policy Compliance Evaluation* process determines whether a service description (SD) complies with the relevant CSB policy. To this end, it entails a number of processes which determine the compliance of key service-level attributes. These processes are intended to implement a general framework of policy-compliance checks, one which is potentially applicable to a wide range of services. Fig. 6 depicts an internal view of such a policy-checking process; Table 5 provides brief descriptions of the activities entailed by such a process.

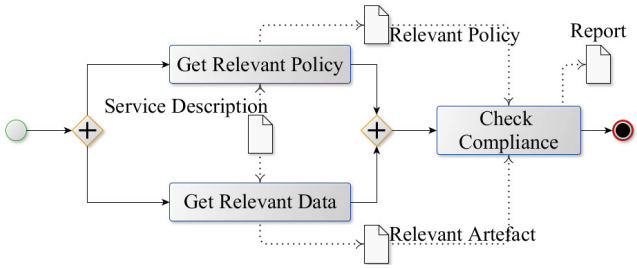

Fig. 6. Internal view of a policy-checking process

Table 5. Policy-checking process

Name	Description
Get Relevant Policy	Determines the applicable policy to the SD.
Get Relevant Data	Extracts from the SD those artefacts that are to be checked.
Check Compliance	Checks compliance of the SD artefacts with the policy; it outputs the outcome in the *Report* data object.

5 Related Work

To the best of our knowledge, there are no works addressing the quality assurance and optimisation dimensions of CSB in the context of VEs. [4] recognises the need for frameworks that guide the creation, execution, and management of services in cloud-based VEs; it does not, however, address the quality assurance and optimisation aspects of such frameworks. The rest of this section outlines work related to the three general themes of our conceptual architecture, namely c*loud service governance and quality control*, *failure prevention and recovery*, and *optimisation*.

Cloud service governance refers to policy-based management of cloud services with emphasis on quality assurance [13]. Current practice [14,15] focuses on the use of registry and repository systems combined with purpose-built software to check the conformance of services with relevant policies [13,14,16]. A major weakness in these works is failure to achieve appropriate separation of concerns between defining governance policies and evaluating data against these policies. This has a number of negative repercussions such as lack of portability and lack of explicit representation of policy interrelations. Turning to service certification, functional testing methods are largely interface-based [17,18]. Some attempts to specify complete behaviour have been suggested using (i) graph transformation rules [19], (ii) WSDL augmented with UML state machines [20], and (iii) SAWSDL augmented with pre- and post-conditions [21]. An extensive survey of the state-of-the-art in cloud service governance and quality control can be found in [9].

The field of adaptive service-based systems (SBSs) [22] investigates techniques for monitoring and adapting SBSs. SBSs share similar characteristics with cloud service ecosystems, hence techniques from the former domain can be readily adopted in our work. An extensive survey of the state-of-the-art in adaptive SBSs is provided in [9]. This survey identifies challenges relevant to cloud service failure prevention and recovery that are not addressed in the literature, including metrics for identifying failures, and scalable prediction techniques for identifying impending failures.

Cloud service optimisation has been primarily investigated from a cloud provider's perspective [23,24], considering consumer satisfaction as a constraint rather than as an optimisation goal. From a cloud consumer perspective, Han et al. [25] propose a service recommender framework for assisting optimisation decisions based solely on IaaS considerations; they fail to consider the dynamically changing conditions that typically characterise cloud service ecosystems. In [26], a service optimizer is proposed to manage dynamic SLAs at the IaaS layer. These works focus exclusively on quantitative metrics which may fail to accurately reflect the relative ranking among services for optimisation purposes [27].

6 Conclusions and Future Work

Unencumbered by the physical constraints of data centres and hardware platforms, cloud computing is anticipated to give rise to VEs that tie together services from vast networks of providers, enabling the creation of a wide range of new, more competitive,

and more agile products and solutions [28]. For this shift to be sustainable however, enterprises are expected to increasingly rely on cloud service brokerage (CSB). This work has presented a conceptual architecture for a framework which provides solutions with respect to the quality assurance and optimisation dimensions of CSB in the context of VEs. Such a framework revolves around the general themes outlined in Section 1, whilst it offers a range of reference processes which are refineable to the level of functional capabilities. The framework is open to the specification of new capabilities, and to the specification of new reference processes. Moreover, it allows adopters to only select individual capabilities providing that these are independently consumable.

In the future, we shall further elaborate and concretise the mechanisms of the proposed CSB framework. We are currently in the process of: (i) developing RDFS-based declarative models in Linked USDL [29] that will enable the interoperable exchange of service and consumer-preference descriptions, and the evaluation of these descriptions against pertinent CSB policies; (ii) developing APIs and prototypes for the CSB mechanisms of Section 3.

Acknowledgements. This research is funded by the EU 7th Framework Programme under the Broker@Cloud project (www.broker-cloud.eu), grant agreement n°328392.

7 References

1. Cloud: What an Enterprise Must Know, Cisco White Paper (2011)
2. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: Towards a cloud definition. SIGCOMM Comput. Commun. Rev., 39 (1), 50--55 (2008)
3. Foster, I., Zhao, Y., Raicu, I. Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. IEEE Grid Computing Workshop 2008. IEEE, pp.1--10, (2008)
4. Cretu, L.G.: Cloud-based Virtual Organization Engineering. Informatica Economică., 16 (1), 98--109 (2012)
5. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organisations., Int. J. High Perform. Comput. Appl., 15 (3), 200--222 (2001)
6. Cloud Computing Reference Architecture. Technical report, NIST (2011)
7. Plummer, D.C., Lheureux, B.J., Cantara, M., Bova,T.: Cloud Services Brokerage is Dominated by Three Primary Roles. Technical report, Gartner (2011)
8. Cloud Brokers Will Reshape The Cloud - Getting Ready For The Future Cloud Business Models. Technical report, Forrester, 2012
9. Broker@Cloud project Deliverable 20.3, http://www.broker-cloud.eu/documents
10. Broker@Cloud project Deliverable 30.1, http://www.broker-cloud.eu/documents
11. Peixoto, D., Batista, V., Atayde, A., Borges, E., Resende, R., Pádua, C.: A Comparison of BPMN and UML 2.0 Activity Diagrams. In: VII Simposio Brasileiro de Qualidade de Software, Florianopolis (2008)
12. Business Process Model and Notation (BPMN), http://www.omg.org/spec/BPMN/2.0/
13. Kourtesis, D., Parakakis, I., Simons, A.J.H.: Policy-driven governance in cloud application platforms: an ontology-based approach. In: 4th. Int. Workshop on Ontology-Driven Information Systems Engineering (2012)
14. Marks, E.A.: Service-Oriented Architecture Governance for the Services Driven Enterprise. John Wiley & Sons (2008)

15. Zhang, L.J., Zhou, Q.: CCOA: cloud computing open architecture. In: IEEE International Conference on Web Services, pp. 607--616. IEEE Press, New York (2009)
16. Kourtesis, D., Paraskakis, I.: A registry and repository system supporting cloud application platform governance. In: 9th Int. Conf. on Service Oriented Computing. LNCS, vol. 7221, pp. 255--256, Springer, Berlin/Heidelberg (2011).
17. Bai, X., Dong, W., Tsai, W., Chen, Y.: WSDL-based automatic test case generation for web services testing. In: Proc. IEEE Int. Workshop Service-Oriented System Eng., IEEE, pp. 215--220, IEEE Comp. Soc., Washington (2005)
18. Chakrabarti, S. Kumar, P.: Test-the-REST: an approach to testing RESTful web services. ComputationWorld 2009: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns. IEEE, pp. 302--308, IEEE Comp. Soc., Washington (2009)
19. Heckel, R., Mariani, L.: Automatic conformance testing of web services, In: Cerioli, M. (ed.) Fundamental Approaches to Software Eng. 2005. LNCS, vol. 3442, pp. 34–48, Springer, Heidelberg (2005)
20. Bertolino, A., Frantzen, I., Polini, A., Tretmans, J.: Audition of web services for testing conformance to open specified protocols. In: Reussner, R., Stafford, J.A., Szypersky, C. (eds.) Architecting Systems with Trustworthy Components, LNCS, vol. 3938, pp. 1--25 Springer, Heidelberg (2006)
21. Tsai, W.T, Paul, R., Wang, Y., Fan C., Wang, D.: Extending WSDL to facilitate web services testing. In: Proc. 7th IEEE Int. Symp. on High Assurance Systems Engineering, pp. 171-172 (2002)
22. Papazoglou, M., Pohl, K., Parkin, M. Metzger, A. (Eds): Service research challenges and solutions for the future internet: S-cube - towards engineering, managing and adapting service-based systems. Springer-Verlag, Berlin/Heidelberg (2010)
23. Moon, H.J., Chi, Y., Hacigumus, H.: SLA-aware profit optimization in cloud services via resource scheduling. In: Proc. 6th World Congress on Services, IEEE, pp. 152—153, IEEE Comp. Soc. (2010)
24. Li, J.Z., Woodside, M., Chinneck, J., Litoiu, M.: CloudOpt: multi-goal optimization of application deployments across a cloud. In: Proc. 7th Int. Conf. on Network and Services Management, pp. 162—170, International Federation for Information Processing, Laxenburg Austria (2011)
25. Han, S.M, Hassan, M.M, Yoon, C.W., Huh, E.N.: Efficient service recommendation system for cloud computing market. Grid an Distributed Computing, Communications in Computer and Information Science, vol 63, pp 117-124, Springer, Heidelberg (2009)
26. Lawrence, A., Djemame, K., Waldrich, O., Ziegler, W., Zsigri, C.: Using service level agreements for optimising cloud infrastructure services. Towards a Service-Based Internet, ServiceWave, LNCS, vol 6569, pp. 38—49, Springer, Berlin Heidelberg (2010)
27. Doyle, J., Thomason, R.H.: Background to qualitative decision theory. AI Magazine, vol. 20, no.2, p. 55--68, Association for the Advancement of Artificial Intelligence (1999)
28. Cloud computing issues and impacts, Ernst & Young, White Paper (2010)
29. Linked USDL, http://www.linked-usdl.org/