Introducing Policy-Driven Governance and Service Level Failure Mitigation in Cloud Service Brokers: Challenges Ahead

Konstantinos Bratanis[1,2] and Dimitrios Kourtesis[1,2]

[1] South-East European Research Centre,
International Faculty, The University of Sheffield,
24 Proxenou Koromila Street, Thessaloniki, 54622, Greece
{kobratanis,dkourtesis}@seerc.org
[2] Department of Computer Science, The University of Sheffield,
Regent Court 211 Portobello Street, Sheffield, S1 4DP, United Kingdom
{k.bratanis,d.kourtesis}@dcs.shef.ac.uk

Abstract. Cloud service brokerage represents a novel operational model in the scope of cloud computing. A cloud broker acts as an intermediary between a service provider and a service consumer with the goal of adding as much value as possible to the service being provisioned and consumed. Continuous quality assurance is a type of brokerage capability having high value to both providers and consumers of cloud services. At the same time, it can be among the most challenging kinds of capability for cloud service brokers to realise. In this paper we focus on two specific themes within this scope. We present a motivating scenario and outline key research challenges associated with introducing policy-driven governance and service level failure mitigation capabilities in brokers.

Keywords: Cloud computing, cloud service brokerage, continuous quality assurance, policy-driven governance, service level failure mitigation.

1 Introduction

With the increasing adoption of cloud computing the enterprise IT environment is progressively transformed into a matrix of interwoven infrastructure, platform and application services, delivered from diverse providers. As the number of providers grows and the requirements of consumers become more complex, the need for entities to assume a role of intermediation between providers and consumers is becoming stronger. Cloud service intermediation is increasingly recognised as an indispensable component of the cloud computing value chain.

Examples of existing cloud service intermediation offerings include services helping enterprises to find and compare cloud services (e.g. marketplaces/stores), to develop and customise services (e.g. application platform as a service offerings), to integrate services (e.g. integration platform as a service), to monitor and manage services, and many more. Despite differences with respect to the capabilities such

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 177–191, 2014.

cloud service intermediaries offer, or how these capabilities are combined, they have one thing in common: making it easier, safer and more productive for cloud computing adopters to navigate, integrate, consume, extend and maintain cloud services. According to Gartner, this is precisely the value proposition of a 'Cloud Services Brokerage', a term coined in 2010 to refer to the emerging role of brokers in the context of cloud computing [1].

In the future, enterprises will require brokerage capabilities that are much more sophisticated than what is on offer by cloud service intermediaries today. Continuous quality assurance of cloud services is one such type of brokerage capability; foreseen to be most valuable for service consumers, but at the same time rather challenging for future brokers to implement.

In this paper we are briefly introducing the concept of cloud service brokerage (CSB) and motivating the need for continuous quality assurance as an important intermediation capability of future enterprise cloud service brokers. We focus our attention on two specific forms of continuous quality assurance intermediation: (i) policy-driven cloud service governance and (ii) service level failure mitigation for cloud services. For each area we present key challenges and provide an overview of related work.

2 Cloud Service Brokerage

As an enterprise comes to rely on an increasing number of externally-sourced cloud services, it becomes more difficult for the enterprise to keep track of when and how these third-party services evolve. Service evolution may be the result of change that is intentional – such as when the provider makes changes to a service's terms of provision, changes to its implementation, or changes to its deployment environment, but also unintentional – such as when the provider suffers an unexpected failure or variation in service performance.

Because of the complexity inherent in consuming multiple services from different cloud service providers, it becomes increasingly more difficult for the service consumer to appreciate all the different kinds of impact that a change to a service can have. A change to a service may mean that the service is no longer conformant to the internal policies of the consumer or to regulations that the consumer is required to observe, or more generally, that the service no longer fulfils the consumer's objectives and needs to be replaced.

Cloud service brokerage represents a new type of service and emerging business model in the space of cloud computing which is aimed at helping enterprises to address such challenges and to mitigate the risks that ensue from the complexity in large-scale cloud service usage [1]. In an analogy to the way other kinds of intermediaries operate within different areas of traditional commerce, a cloud service broker is an entity that works on behalf of a consumer of cloud services to intermediate and to add value to the services being consumed.

Much of the enabling technology that is needed to support different cloud service brokerage capabilities is certainly not new. Recent years have seen a proliferation of many relevant proprietary and open source tools that can provide building blocks for the implementation of such services, such as tools for monitoring and managing

applications and virtual infrastructures, or tools for integrating heterogeneous processes and applications. Companies such as SpotCloud[1], Vordel[2], Rightscale[3], JitterBit[4], or SnapLogic[5], who have already created offerings based on such enabling technologies, can be considered early examples of cloud service brokerages.

The kinds of intermediation capability offered by most of today's cloud service brokers relate to cloud service discovery, integration, customisation, or aggregation [2]. But as cloud service consumption grows and quality assurance becomes more of a problem to cloud service users, intermediation capabilities for continuous quality assurance of cloud services will become more and more prevalent.

Intermediation for continuous quality assurance of cloud services represents an open research topic which, to the best of our knowledge, is only now receiving attention by research communities working on related fields. At the time of this writing the theoretical and pragmatic challenges of introducing continuous quality assurance functions in brokers of cloud services remain largely unexplored.

3 Continuous Quality Assurance Intermediation Example

In this section we present an abstract usage scenario that exemplifies two new forms of continuous quality assurance intermediation: policy-driven governance and service level failure mitigation for cloud services.

We assume a setting where a cloud service broker operates an online platform, though which it offers continuous quality assurance intermediation for cloud services. The broker's customers are enterprises that make extensive use of third-party cloud services and prefer to outsource their continuous quality assurance functions to a specialised and trusted third-party entity – the broker. The broker allows service consumers to exercise fine grained control over the cloud services they rely on. This is achieved by allowing consumers to express their objectives about how cloud services should be delivered to them, in the form of policies. The brokerage platform then undertakes to ensure that the objectives in the policies are met. Service consumption objectives may relate to the pricing characteristics of a cloud service, its security features, its availability guarantees, and many other service attributes. The brokerage platform is capable of monitoring service delivery on a continuous basis, detecting violations of consumer policies, and proposing mitigation measures.

Providers of cloud services who are interested in making their services available to the customers that the broker is serving need to onboard their services to the brokerage platform. For this to be done, providers need to create descriptions of their cloud services. The broker maintains its own vocabulary for service description that providers have to use. This vocabulary can be understood as a kind of reference model for cloud service attributes that allows a service description to be reconcilable

[1] http://www.spotcloud.com/
[2] http://www.vordel.com/
[3] http://www.rightscale.com
[4] http://www.jitterbit.com/
[5] http://www.snaplogic.com/

with descriptions of other cloud services as well as with consumers' policies. The broker offers the same description vocabulary to consumers, in order for them to create their own custom policies about the services they consume through the broker. Once a cloud service is accepted for onboarding into the brokerage platform it is continuously monitored for quality assurance purposes. Intentional or unintentional changes to the service or to its associated descriptive artefacts will be detected and evaluated as soon as they occur, allowing the consumer to be notified early and to take appropriate mitigation measures to continue meeting their service consumption objectives.

Below we provide a step-by-step walkthrough of a usage scenario where the three roles (broker, provider and consumer) are interacting. To improve readability we have decomposed the scenario into four phases. The interactions between roles in each phase are illustrated with the help of a respective BPMN diagram.

Phase 1: Service Onboarding

1. The provider creates a description of the service to be onboarded based on the broker's vocabulary and submits the description to the broker. The description references a wide range of service characteristics, including the service's pricing, security features, and reliability guarantees.
2. The broker checks if the service description includes all necessary description elements. The provider is not obliged to describe a service with respect to all of the attributes listed in the broker's vocabulary but some service attributes are required. Service availability over a defined period of time is one such mandatory description element[6]. For the service in question the provider commits to availability of 99.92% on a monthly basis.
3. The broker determines that the service description includes all of the required description elements, onboards the service, and starts to monitor the service and its associated description artefacts for changes.

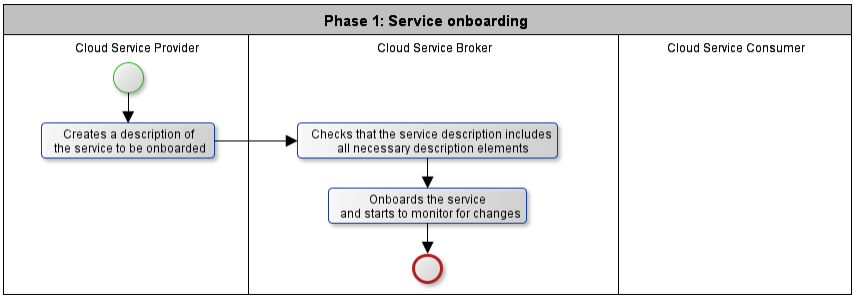

Fig. 1. The flow of service onboarding activities

[6] For example, the availability guarantees that Amazon offers for its EC2 and EBS services is a Monthly Uptime Percentage of at least 99.95% (as of June 2013). In the event that Amazon does not meet this commitment, consumers receive service credit as compensation.

Phase 2: Service Selection

4. The consumer discovers the service in the broker's service directory and subscribes to use it.
5. The consumer creates a policy governing how the service should be delivered. Based on the broker's vocabulary, the policy states that monthly service availability should be no less than 99.90%[7] and the time it takes for the service to recover from a failure should be no more than 30 minutes per outage. This is the recovery time objective (RTO) policy of the consumer.
6. The consumer submits the policy to the broker so that the latter can monitor the compliance of the selected service to the consumer's policy on a continuous basis.
7. The broker evaluates the consumer's policy against the description of the selected service and determines that the service description is conformant to the policy. The broker takes no further action.

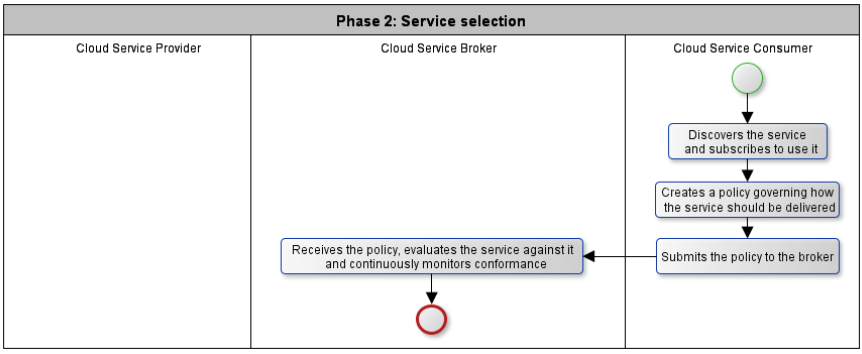

Fig. 2. The flow of policy conformance evaluation activities during service selection

Phase 3: Change to the Service's Terms of Provision

8. The provider updates the service description, changing the service availability commitment to 99.95%.
9. The broker detects the change in the service description artefact and carries out a conformance check to determine whether or not this creates a conflict with the consumer's policy. The change is not found to raise any conformance issues because the new availability commitment (99.95%) is higher than the consumer's monthly availability objective (99.90%). The broker takes no further action.[8]

[7] In a 24x7 setting, 99.90% availability translates to 43 min and 12 sec of downtime per month.

[8] In case the change to the service's terms of provision gave rise to a violation of the consumer's policy (e.g. if the new availability commitment was 99.85%) the broker would have alerted the consumer to take action. This could mean substituting the service or lowering the consumer's expectations in their RTO policy.

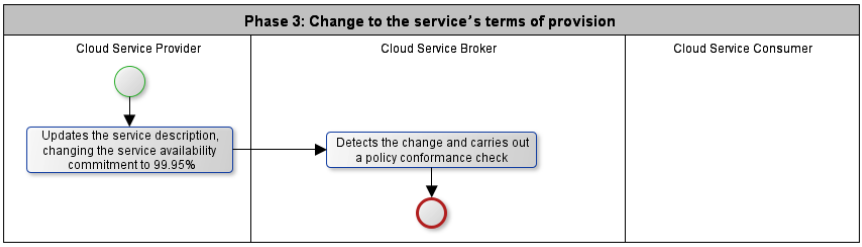

Fig. 3. The flow of policy conformance evaluation activities upon a change to the service caused by updating its terms of provision

Phase 4: Change to the Service's Availability

10. The service provider experiences an unexpected failure that causes service outage.
11. The broker detects a change in service availability (downtime).
12. The broker alerts both the consumer and the provider about the failure.
13. The broker starts a timer to track the service's downtime. At the same time, it proactively attempts to identify alternative services to potentially serve as substitutes for the failing service.
14. The broker predicts that the time it will take for the service to resume operation (time to recovery, or TTR) is 12 minutes. This is shorter than the objective of 30 minutes per outage as specified in the consumer's RTO policy. The broker concludes that the service is likely to recover within a time period that is tolerable for the consumer, and takes no further action.
15. The 12 minute period lapses and the service is still down, which means that the broker's predicted TTR was optimistic. The broker concludes that it is very likely that the provider will not be able to meet the consumer's RTO of 30 minutes[9], thus causing a major disruption to business continuity for the consumer.
16. The broker alerts the consumer to obtain approval to proactively substitute the service with one of the identified candidates before downtime exceeds the consumer's tolerable threshold. At the same time, it also alerts the provider.
17. The consumer approves the proactive substitution of the service. The broker has thus helped the consumer to mitigate the impact from a potential service level failure due to a sustained service outage and the consequent violation of the consumer's RTO. In doing so, the broker has helped the consumer to meet their objectives with respect to business continuity.

[9] In case the predicted TTR value was greater than the RTO specified in the consumer's policy (e.g. if the broker predicted a TTR of 31 minutes), the broker would have immediately alerted the consumer to obtain approval for proactive service substitution.

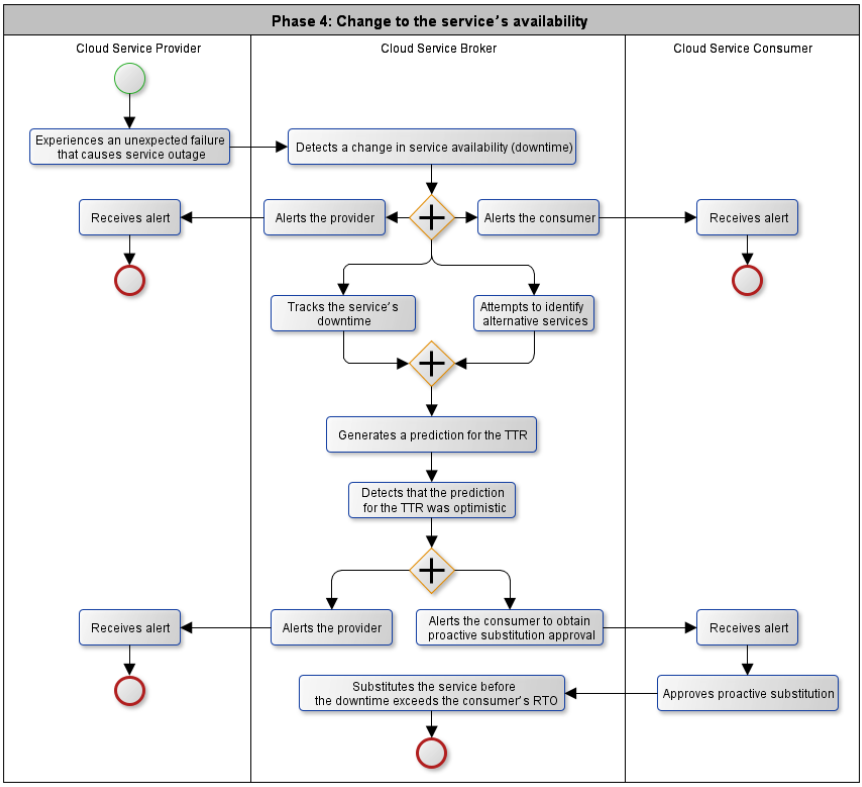

Fig. 4. The flow of service level failure mitigation activities upon a change to the service caused by an unexpected outage

For more example scenarios of continuous quality assurance intermediation we refer the interested reader to [3].

4 Challenges for Continuous Quality Assurance Intermediation

In this section we discuss the challenges associated with introducing capabilities for policy-driven service governance and service level failure mitigation in cloud service brokers. A research roadmap that considers other intermediation capabilities in the broader scope of continuous quality assurance and optimisation can be found in [4].

4.1 Challenges of Policy-Driven Governance for Cloud Services

Because of the fact that cloud service brokers intermediate between a consumer of cloud services and multiple service providers, they are uniquely positioned to address the need of the consumer to exercise as much control as possible over the external services on which it relies. This can be understood as a problem of cloud service

governance. In this context we take governance to mean the enforcement of policies to manage the lifecycle of a cloud service as seen from the consumer's perspective, as well as to apply quality control over the service and its associated artefacts. These two concerns map onto two complementary forms of policy-driven governance: process governance and artefact governance.

Process governance refers to defining and enforcing policies to ensure that cloud services are selected, tested, used and retired in a structured and disciplined manner, with explicit conditions for transitioning from one service lifecycle phase to the next. Artefact governance, on the other hand, refers to defining and enforcing policies to ensure that artefacts associated with cloud services conform to certain technical or business constraints.

Some key challenges in the scope of supporting policy-driven governance inside a cloud service brokerage platform include:

- *How to achieve adequate separation of concerns in the design of the brokerage platform's governance support system?* In designing a governance support system we need to take into account the fact that there are three main roles at play in a policy-driven governance setting: (1) the role of providing the policies, (2) the role of providing data about the resources which are governed by the policies, and (3) the role of evaluating the governed resource data against the policies. In the scenario of Section 3 these roles are assumed by the service consumer, the service provider and the service broker, respectively. Each role has a different primary concern (i.e. maintaining governance policies, maintaining data about governed resources, and maintaining mechanisms to evaluate policies). The governance support system should be designed so as to facilitate all of them. As Baker puts it, "if we are attempting to separate concern A from concern B, then we are seeking a design that provides that variations in A do not induce or require a corresponding change in B (and usually, the converse)" [5]. For instance, in our context, this means that if the service consumer changes the way that a governance policy is represented this should not induce a change in how cloud service providers represent governed resource data, or how the broker evaluates service provider data against service consumer policies. The entities assuming the three roles should be allowed to evolve independently of each other.
- *How to effectively represent governance policies and governed resource data?* The means of policy representation determine the ease with which policies can be: (1) analysed in a systematic way for validation/troubleshooting and policy evaluation, (2) shared with other stakeholders in a cloud service ecosystem, (3) exchanged between different software systems, (4) cross-referenced to other policies so as to keep track of relationships between policies at different hierarchical levels and be able to know which policy needs to change when some other policy changes, (5) cross-referenced to classes of governed resources so as to keep track of which policies are relevant to each kind of governed resource. Likewise, the means of representing governed resource data determine the ease with which the data can be: (1) analysed in a systematic way for validation/troubleshooting and policy evaluation, (2) shared with other stakeholders in a cloud service ecosystem, (3) exchanged between different software systems, (4) cross-referenced to other governed resource

data so as to keep track of dependencies between cloud services and between cloud service artefacts, (5) cross-referenced to policies so as to keep track of which policies are applicable to each governed resource. Notably, the way in which this challenge is met in the design of a governance support system is highly relevant to how adequately the first challenge above can be addressed (separation of concerns).

4.2 Challenges of Service Level Failure Mitigation for Cloud Services

The service levels delivered by cloud services cannot be assumed to be static. Service performance may degrade over time, and infrastructure failures (i.e. failures of servers or network equipment) may result in outages. When service performance does not meet the service level objectives(s) that a consumer has specified this is called a service level failure. This means that a temporary failure to the service delivery infrastructure of a provider may not necessarily give rise to a service level failure for a particular consumer. Consumers need to monitor their service level objectives and to mitigate the impact of service level failures when these occur. This creates scope for a cloud service broker to offer highly valuable services as an intermediary.

The goal of service level failure mitigation is to assist the consumer to avoid or minimise the impact from a potential failure of the service to meet the consumer's targets. Mitigation of service level failures requires processes for monitoring various metrics about the operation of a cloud service and generating predictions about how the values of those metrics will evolve. Based on the predictions, the likelihood of the service to continue meeting the consumer's service level objectives can be assessed. If there is indication of an impending service level failure the broker can issue early warnings to consumers allowing them to take proactive mitigation measures.

Key challenges in the scope of developing intermediation mechanisms for service level failure mitigation for cloud services include the following:

- *Which metrics are relevant and useful for consumers and which of those should the broker collect data for?* The cloud service broker has to support metrics that are useful to consumers, by performing data collection for those metrics and predicting impending failures of the service providers to meet the consumer's objectives. For example, some useful metrics include availability, throughput, completion time, response time, mean downtime and others. However, there is no uniform standard for metrics used across cloud service providers, which makes it more challenging for the broker to monitor the same metric in different providers.
- *How to implement scalable monitoring for different types of metrics from a large number of cloud services, without overwhelming the cloud service broker, or introducing additional overhead to the cloud service?* The broker has to monitor metrics for a continuously increasing number of cloud services and analyse them in near real-time to identify impending failures of the provider to meet the consumers' objectives. The broker has to offer affordable means for service providers to publish data about the service for monitoring, because most service providers do not have appropriate interfaces in place. In addition, the broker has to minimise the overhead introduced to the cloud services as a result of the monitoring activity.

- *Which is the most appropriate prediction technique that the cloud service broker can employ for identifying impeding service level failures with acceptable accuracy?* The broker has to analyse the monitored data about the metrics by applying appropriate prediction techniques for determining future values of metrics, preferably in near real-time. The broker has to deal with the use and maintenance of different prediction models, depending on the type of the monitored data, and instantiate those models for making predictions in massive scale. The broker has to generate predictions for several metrics which concern many cloud services and several objectives of a large number of consumers who use those services.

5 Related Work

5.1 Related Work on Policy-Driven Cloud Service Governance

A look at different cloud service intermediaries, such as cloud application platform providers, reveals different approaches and tools for policy-driven governance over processes and artefacts. Development and deployment of cloud services on the Intuit Partner Platform[10] proceeds through four phases, each of which is called 'a line of development'. The phases are called development, quality assurance, staging, and publishing. Similarly, on Heroku[11], add-ons (i.e. third-party services) advance through the phases of development, alpha, private beta, beta, and general availability. In Force.com[12] the majority of quality checks on cloud service artefacts are associated with a particular phase towards the end of the development and deployment process, referred to as 'security review' – though the scope of the review carried out is actually much broader than security.

The industrial state of the art in policy-based governance follows closely on the evolution of tools supporting different aspects of service governance, such as artefact cataloguing and storage, service lifecycle management, dependency tracking, and policy enforcement. Those tools are typically integrated within some kind of registry and repository system [6], [7]. Vendors of today's governance registry and repository systems support different means by which policies can be encoded and enforced [8], [9], [10]. As shown by a recent survey of methods for policy management in contemporary open source registry and repository systems [11], a major weakness in the state of the art is the lack of proper separation of concerns with regard to defining rules for governance and acting upon them. Policy definition and policy enforcement are entangled in the implementation of a single software component – the policy evaluation engine. For the most part, the rules that a policy comprises are encoded in an imperative manner, in the same programming language that the registry and repository system has been implemented, and as part of the same code that checks the data for violations. This can be shown to create many negative side effects.

[10] https://developer.intuit.com/
[11] https://www.heroku.com/
[12] http://www.force.com/

To avoid the problems stemming from insufficient separation of concerns, recent works in the field of policy-based systems management stress the importance of designing software applications such that business rules are kept separate from the core program logic. It is best for such rules to be captured though policy-specification languages and to be consulted at run-time when user activity dictates to do so [12].

Several works motivated by similar objectives have focused on the enhancement of existing policy languages and tools with ontology-based methods of representation and processing. The most prominent early works along this line were KAoS [13], Ponder [14], and Rei [15]. Other, more recent works in a similar direction are those by Kolovski et al. [16] and Kolovski and Parsia [17].

Several recent research efforts and industrial pilot projects have been turning their attention to the benefits that the application of ontology-based modelling and reasoning can have with respect to different aspects of software engineering [18]. As Bergman points out [19], many of the benefits which are generally obtained by ontology-centric approaches to the development of information systems are attributed to the fact that the locus of effort is shifted from software development and maintenance to the creation and modification of knowledge structures. Uschold cites six important benefits which result from the increased level of abstraction and the use of formal structures and methods in ontology-driven information systems: reduced conceptual gap; increased automation; reduced development times; increased reliability; increased agility/flexibility; decreased maintenance costs [20].

The above benefits of ontology-centric approaches to information systems engineering, in combination with the Semantic Web standards and tools currently available [21] appear to provide a promising foundation for addressing the shortcomings of policy management in contemporary governance support systems [11]. Results from efforts with similar objectives are already being reported in the wider context of policy-driven systems management, such as the work by IBM on the transformation of sources of management data into Linked Data providers to allow for uniform logic-based queries over heterogeneous systems in a network [22].

5.2 Related Work on Service Level Failure Mitigation for Cloud Services

Over the past decade there has been an increasing interest in incorporating self-managing capabilities in software systems, motivated by high complexity involved in the everyday administration, as well as the detection, diagnosis, and resolution of failures in software systems. The research fields of Autonomic Computing [23] and Self-Adaptive Systems [24] continue to demonstrate fundamental advances towards understanding the challenges associated with the aforementioned research directions. Several approaches have been proposed for partially addressing the detection of failures, the diagnosis process to identify the cause of failure and, the automation of adaptation actions, as a solution to recovering from failures.

Recent research works on cloud service monitoring [25], [26] focus on the infrastructure level, and do not consider cloud services related to the platform and the application levels. The monitoring techniques found in the field of service-oriented computing can provide a useful inspiration for addressing those two levels.

Research in self-adaptive service-based systems outlined in Papazoglou et al. [27] can serve as the foundation for exploring novel mechanisms for service level failure

mitigation in the context of cloud service brokerage, since cloud services, ranging from programmatically-accessible web APIs to complex software applications delivered as a service, present characteristics similar to those of services in the service-oriented architecture. Therefore, the research literature focusing on self-adaptive service-based systems is considered highly relevant.

Monitoring is well studied in self-adaptive service-based systems. Monitoring approaches follow either a push mode, where events or data are sent to a monitoring component, or a pull mode, where a monitoring component queries the subject of monitoring. Such approaches range from verification of service behaviour [28] and evaluating rules for detecting SLA violations [29] to dependency analysis for identifying causes offending some KPIs [30] and complex event processing for detecting situations based on the correlation of basic events [31], [32].

There exist approaches that attempt to proactively prevent failures from occurring by systematically testing services to uncover failures and deviations of quality of service from what is expected. Existing approaches for testing service-based systems mostly focus on testing during design-time, which is similar to testing of traditional software systems [33]. Others, like PROSA [34], exploit online testing [35] at run-time in order to proactively trigger adaptation. The focus of these approaches is to prevent QoS degradation of the service-based system.

A more relevant work is the PREvent framework [36], [37], which integrates event-based monitoring, prediction of SLA violations using machine learning, and runtime prevention of such violations at the provider-side by triggering adaptation actions in service compositions. There are also few other similar works [38], [39] concerned with prediction that make use of historical data regarding the execution of a business process to predict the performance of process instances.

The goal of the aforementioned works is to help the provider to prevent failures from occurring at the provider-side, whereas in our work we aim at helping a service consumer to avoid the impact of the provider's failure to meet the consumer's objectives. Furthermore, these approaches come from a different domain than cloud computing and they do not consider the intermediary's perspective, which aims at adding value to service consumers. Nevertheless, they offer inspiration for investigating prediction techniques for mitigation of service level failure realised by a cloud service broker. To the best of our knowledge, our work is the first that addresses service level failure mitigation in the context of cloud service brokerage.

6 Conclusion and Future Work

In this paper we have examined two facets of continuous quality assurance intermediation that we expect to become increasingly important in the scope of cloud service brokerage: policy-driven governance for cloud service and service level failure mitigation for cloud services. We presented an example scenario which demonstrates the utility of such types of continuous quality assurance intermediation capability. We attempted to introduce some basic concepts and key challenges, and to provide a glimpse of related work that one can build upon to develop solutions.

Our future work relative to policy-driven governance will focus on evaluating a new approach to the design of governance support systems that addresses the

challenges discussed in section 4.1. This new approach overcomes many of the limitations in existing governance support systems and is natively suitable for use in a cloud service brokerage context. The solution builds on Linked Data principles and Semantic Web technologies [40] and comprises four major components: 1) a cloud service governance ontology serving as common vocabulary for describing governance policies and governed resources; 2) a methodology for encoding governance policies that facilitates better knowledge management about governance operations and enables automated semantic analysis of governance policies; 3) mechanisms to automatically generate semantic descriptions of governed resources by means of transformation from their native representation into Linked Data; and 4) a generic and reusable infrastructure to automatically evaluate descriptions of governed resources against applicable policies. Past research by Kourtesis et al. [11], [41] will serve as baseline to this work.

In the context of service level failure mitigation, our future work will focus on addressing the three challenges mentioned in section 4.2 through the development of a scalable approach for monitoring and prediction of metrics comprising three major components: 1) a set of metrics that are relevant to cloud service consumers; 2) a scalable software architecture for data collection based on the convergence of the "push" and "pull" communication paradigms and the use of Linked Data principles; 3) a study of failure prediction approaches using machine learning techniques appropriate for generating predictions for different kinds of metrics. We will use as baseline previous work on engineering of monitoring architectures for service-based systems [42], measuring many kinds of non-functional and functional properties of services [43] and the different metrics across the cloud stack [44].

References

1. Benoit, J., Lheureux, D., Plummer, C.: Cloud Services Brokerages: The Dawn of the Next Intermediation Age. Gartner (2010)
2. Verginadis, Y., Patiniotakis, I., Mentzas, G., Kourtesis, D., Bratanis, K., Friesen, A., Simons, A.J.H., Kiran, M., Horn, G., Rossini, A., Schwichtenberg, A., Gouvas, P.: D2.1 State of the art and research baseline. Broker@Cloud Project deliverable (2013)
3. Kourtesis, D., Bratanis, K., Friesen, A., Simons, A.J.H., Kiran, M., Verginadis, Y., Rossini, A., Schwichtenberg, A., Gouvas, P.: D2.3 Requirements Analysis Report. Broker@Cloud Project deliverable (2013)
4. Bratanis, K., Kourtesis, D., Paraskakis, I., Verginadis, Y., Mentzas, G., Simons, A.J.H., Friesen, A., Braun, S.: A Research Roadmap for Bringing Continuous Quality Assurance and Optimization to Enterprise Cloud Service Brokers. In: Proc. of eChallenges 2013, Dublin, Ireland (2013)
5. Baker, M.: The Lost Art of Separating Concerns.InfoQ (2006)
6. Marks, E.A.: Service-Oriented Architecture Governance for the Services Driven Enterprise. John Wiley & Sons, Hoboken (2008)
7. Zhang, L., Zhou, Q.: CCOA: Cloud Computing Open Architecture. In: Proc. 2009 IEEE Int. Conf. on Web Services (ICWS 2009), pp. 607–616. IEEE (2009)
8. WSO2 Governance Registry, http://wso2.com/products/governance-registry/

9. IBM WebSphere Service Registry and Repository, http://www.ibm.com/software/integration/wsrr/
10. Oracle Service Registry, http://www.oracle.com/us/products/middleware/soa/service-registry/overview/index.html
11. Kourtesis, D.: Towards an ontology-driven governance framework for cloud application platforms, Dept. Comp. Sci. Univ. Sheffield, UK. Tech. Rep. CS-11-11 (2011)
12. Fisler, K., Krishnamurthi, S., Dougherty, D.J.: Embracing policy engineering. In: Proc. of the FSE/SDP Workshop on Future of Software Engineering Research. ACM (2010)
13. Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate, A., Dalton, J., Aitken, S.: KAoS Policy Management for Semantic Web Services. In: IEEE Intelligent Systems, vol. 19(4), pp. 32–41. IEEE (2004)
14. Damianou, N., Dulay, N., Lupu, E.C., Sloman, M.: The Ponder Policy Specification Language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)
15. Kagal, L., Finin, T., Johshi, A.: A Policy Language for a Pervasive Computing Environment. In: Proc. of the 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks, pp. 63–74. IEEE, Washington, DC (2003)
16. Kolovski, V., Parsia, B., Katz, Y., Hendler, J.: Representing Web Service Policies in OWL-DL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 461–475. Springer, Heidelberg (2005)
17. Kolovski, V., Parsia, B.: WS-Policy and beyond: application of OWL defaults to Web service policies. In: Proc. of the 2nd Int. Semantic Web Policy Workshop, USA (2006)
18. Gasevic, D., Kaviani, N., Milanovic, M.: Ontologies and Software Engineering. In: Handbook on Ontologies, 2nd edn., pp. 593–615. Springer, Heidelberg (2009)
19. Bergman, M.: Ontology-Driven Apps Using Generic Applications, AI3 blog (2011)
20. Uschold, M.: Ontology-Driven Information Systems: Past, Present and Future. In: Eschenbach, C., Gruninger, M. (eds.) Proc. of the 5th Int. Conf. on Formal Ontology in Information Systems (FOIS 2008), pp. 3–18. IOS Press, The Netherlands (2008)
21. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & HallCRC (2009)
22. Feridun, M., Tanner, A.: Using linked data for systems management. In: Proc. of the IEEE Network Operations and Management Symposium, NOMS (2010)
23. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing—degrees, models, and applications. ACM Computing Surveys 40(3), 1–28 (2008)
24. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems 4, 1–42 (2009)
25. Bertolino, A., Calabro, A., Lonetti, F., Sabetta, A.: GLIMPSE: a generic and flexible monitoring infrastructure. In: Proceedings of the 13th European Workshop on Dependable Computing, New York, NY, USA, pp. 73–78 (2011)
26. Romano, L., De Mari, D., Jerzak, Z., Fetzer, C.: A Novel Approach to QoS Monitoring in the Cloud. In: Proceedings of the First International Conference on Data Compression, Communications and Processing (CCP), pp. 45–51 (2011)
27. 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, Berlin (2010)
28. Dranidis, D., Ramollari, E., Kourtesis, D.: Run-time Verification of Behavioural Conformance for Conversational Web Services. In: Proceedings of the 7th IEEE European Conference on Web Services, ECOWS (2009)

29. OriolHilari, M., Marco Gomez, J., Franch, X., Ameller, D.: Monitoring Adaptable SOA Systems using SALMon. In: Proceedings of the 1st Workshop on Monitoring, Adaptation and Beyond (MONA+), pp. 19–28 (2008)
30. Wetzstein, B., Leitner, P., Rosenberg, F., Dustdar, S., Leymann, F.: Identifying influential factors of business process performance using dependency analysis. Enterp. Inf. Syst. 5(1), 79–98 (2011)
31. Baresi, L., Caporuscio, M., Ghezzi, C., Guinea, S.: Model-Driven Management of Services. In: Proceedings of the 2010 IEEE 8th European Conference on Web Services (ECOWS), pp. 147–154 (2010)
32. Hermosillo, G., Seinturier, L., Duchien, L.: Using Complex Event Processing for Dynamic Business Process Adaptation. In: Proc. of the IEEE SCC 2010, pp. 466 (2010)
33. Gehlert, A., Metzger, A., Karastoyanova, D., Kazhamiakin, R., Pohl, K., Leymann, F., Pistore, M.: Integrating Perfective and Corrective Adaptation of Service-based Applications. In: Dustdar, S., Li, F. (eds.) Service Engineering: European Research Results Book, pp. 137–169. Springer (2011)
34. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A framework for proactive self-adaptation of service-based applications based on online testing. In: Mähönen, P., Pohl, K., Priol, T. (eds.) ServiceWave 2008. LNCS, vol. 5377, pp. 122–133. Springer, Heidelberg (2008)
35. Dranidis, D., Metzger, A., Kourtesis, D.: Enabling Proactive Adaptation through Just-in-Time Testing of Conversational Services. In: Di Nitto, E., Yahyapour, R. (eds.) ServiceWave 2010. LNCS, vol. 6481, pp. 63–75. Springer, Heidelberg (2010)
36. Leitner, P., Wetzstein, B., Rosenberg, F., Michlmayr, A., Dustdar, S., Leymann, F.: Runtime prediction of service level agreement violations for composite services. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 176–186. Springer, Heidelberg (2010)
37. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: Monitoring, Prediction and Prevention of SLA Violations in Composite Services. In: Proceedings of the 2010 IEEE International Conference on Web Services (ICWS 2010), pp. 369–376. IEEE (2010)
38. Sahai, A., Machiraju, V., Sayal, M., Van Moorsel, A., Casati, F.: Automated SLA monitoring for web services. In: Feridun, M., Kropf, P.G., Babin, G. (eds.) DSOM 2002. LNCS, vol. 2506, pp. 28–41. Springer, Heidelberg (2002)
39. Zeng, L., Lingenfelder, C., Lei, H., Chang, H.: Event-Driven Quality of Service Prediction. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 147–161. Springer, Heidelberg (2008)
40. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. In: Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1(1), pp. 1–136. Morgan and Claypool (2011)
41. Kourtesis, D., Paraskakis, I., Simons, A.J.H.: Policy-driven governance in cloud application platforms: an ontology-based approach. In: Proc. 4th. Int. Workshop on Ontology-Driven Information Systems Engineering (2012)
42. Bratanis, K.: Towards engineering multi-layer monitoring and adaptation of service-based applications, Dept. Comp. Sci. Univ. Sheffield, UK. Tech. Rep. CS-12-04 (2012)
43. Bratanis, K., Dranidis, D., Simons, A.J.H.: An Extensible Architecture for Run-time Monitoring of Conversational Web Services. In: Proc. of the 3rd International Workshop on Monitoring, Adaptation and Beyond / ECOWS 2010, pp. 9–16. ACM (2010)
44. Bratanis, K., Dranidis, D., Simons, A.J.H.: SLAs for cross-layer adaptation and monitoring of service-based applications: A case study. In: Proc. of the International Workshop on Quality Assurance for Service-Based Applications, p. 28. ACM (2011)