Self-adaptation Challenges for Cloud-based Applications: A Control Theoretic Perspective

Soodeh Farokhi
Faculty of Informatics
Vienna University of Tech.
soodeh.farokhi@tuwien.ac.at

Pooyan Jamshidi
Dept. of Computing,
Imperial College London
p.jamshidi@imperial.ac.uk

Ivona Brandic
Faculty of Informatics
Vienna University of Tech.
ivona.brandic@tuwien.ac.at

Erik Elmroth
Dept. of Computing Science
Umeå University
erik.elmroth@cs.umu.se

ABSTRACT

Software applications accessible over the web are typically subject to very dynamic workloads. Since cloud elasticity provides the ability to adjust the deployed environment on the fly, modern software systems tend to target cloud as a fertile deployment environment. However, relying only on native elasticity features of cloud service providers is not sufficient for modern applications. This is because current features rely on users' knowledge for configuring the performance parameters of the elasticity mechanism and in general users cannot optimally determine such sensitive parameters. In order to overcome this user dependency, using approaches from autonomic computing is shown to be appropriate. Control theory proposes a systematic way to design feedback control loops to handle unpredictable changes at runtime for software applications. Although there are still substantial challenges to effectively utilize feedback control in self-adaptation of software systems, software engineering and control theory communities have made recent progress to consolidate their differences by identifying challenges that can be addressed cooperatively. This paper is in the same vein, but in a narrower domain given that cloud computing is a sub-domain of software engineering. It aims to highlight the challenges in the self-adaptation process of cloud-based applications in the perspective of control engineers.

Categories and Subject Descriptors

H.4 [**information systems applications**]: miscellaneous; c.2.4 [**computer-communication networks**]: distributed systems—*distributed applications*

General Terms

theory, performance

Keywords

cloud computing, control theory, adaptive software, elasticity, quality of service (QoS).

1. INTRODUCTION

Web applications have very dynamic workloads generated by variable numbers of users and they face sudden peaks in the case of unexpected events. Therefore, dynamic resource allocation is necessary not only to avoid application performance degradation but also to avoid under-utilized resources. Elasticity, as a main selling point of cloud, is the ability to rapidly adjust the allocated resource capacity for an application according to the time-varying workload in order to meet the quality of service (QoS) requirements. However, relying only on elasticity features provided by cloud providers does not seem sufficient to ensure the sensitive performance requirements of modern applications. The main reason is that providing smooth elasticity is intrinsically hard and native elasticity requires critical performance configurations to be performed by users. However, in practice this is often insufficient and results in sub-optimal scaling decisions which may negatively affect robustness, performance and cost-efficiently, while also incurring unwanted oscillations in resource allocations. Therefore, cloud-based applications while utilizing the flexibility features of cloud environments, e.g. cloud elasticity, should be entangled with an intelligent software, i.e., autonomic managers, that continuously monitors application behaviour and automatically adjusts resource allocations to meet predefined performance targets. Among other possible solutions for realizing such autonomic managers, feedback control is promising for handling unpredictable runtime changes for software applications [30]. Although automated resource provisioning is among the primary application areas in which control theory is currently applied [18], cloud environments introduce new challenges making control theoretical solutions even more relevant.

There are some research on self-adaptive solutions for software systems which take software engineering point of view. Cheng et al. [6] identify critical challenges for the systematic software engineering of self-adaptive systems divided by modelling dimensions, requirements, engineering, and assurances. As a continuous trend De Lemos et al. [4] bring up research challenges when developing, deploying and managing self-adaptive software systems. They focus on four essential topics of self-adaptation: design space for self-adaptive solutions, software engineering processes for self-adaptive systems, from centralized to decentralized control, and practical run-time verification for self-adaptive systems.

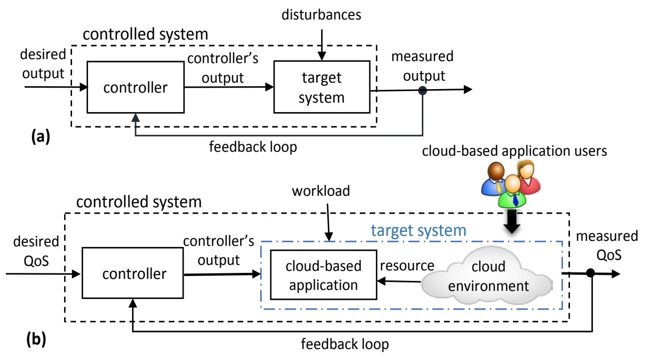

Figure 1: (a) A standard feedback control; (b) A realization of self-adaptive cloud-based applications via feedback control.

Moreover, there is a new trend of applying the control theory in software systems, taking the point of view of control engineering. Filieri et al. [9] highlight potential solutions towards application of control theory in construction of adaptive software. They focus on adaptation of a specific class of models and control to achieve reliability properties. Maggio et al. [23] study different self-adaptation strategies and discuss that adaptive and model predictive control systems outperform other approaches in performance aspect on a priori unknown situations. Very recently, *GI-Dagstuhl seminar* [2] gathered two communities of software engineering and control theory to develop their cooperation for devising new modeling strategies to empower software engineers with theoretical and practical skills of control engineers and bring control to the core of adaptation. As a result of the last seminar, in [11] they present a general control design process for software systems which enables automatic (i) analysis and (ii) synthesis of a controller that is guaranteed to have the desired properties and behavior. However, the research regarding the application of control theory to enable self-adaptation in software engineering, despite its recent progress, is still in a very early stage [9].

Considering cloud computing as a sub-domain of software engineering, there are also some research which focus more on cloud scenarios. Patikirikorala et al. [26] investigate the benefits and limitations of applying feedback controllers in cloud computing platforms, and to this aim, they briefly highlight a few system design requirements. Kihl et al. [21] propose a new research area called *Cloud Control* as a control theoretic approach to a range of cloud management problems. They discuss major challenges for resource-optimized cloud data centers. Their idea lead to establishing a series of scientific meeting called *Control Cloud Workshop* [1] aim to foster research in the area of cloud computing and control theory. Current existing research on combining cloud computing and control theory have predominantly the perspective of cloud provider, hence their focus is more on controlling the cloud data centers. Whereas we look at the challenges from the cloud application's point of view. In particular, as discussed, there are only a few research attempts [26, 21, 1] that address the challenges of self-adaptive cloud-based applications, so there are still open challenges which have not been thoroughly investigated.

By combining cloud computing, modern software systems, and control theory, the ultimate objective is to turn cloud-based applications into self-adaptive systems which are performance sensitive, robust, flexible, resource-aware and cost-efficient. The aim of this paper is along the lines of *Cloud Control* research area proposed in [21], but with a more pronounced application perspective. We bring up a range of important research challenges as a research agenda on the way of realizing self-adaptation for cloud-based applications.

2. SELF-ADAPTIVE CLOUD APPLICATIONS

The key aspect of an elastic software is its capability to autonomic ally adapt at runtime (i.e., self-adapt) in response to changes in the operating conditions, such as fluctuations in workload, by automatically stretching and shrinking the resources. Cloud-based elastic software are the most common realization of elastic software. This category of software systems exploits the ability of cloud environments to acquire and release resources while servicing end users. For instance, when the usage of the system increases, the allocated resources may saturate and in order to avoid degradation of QoS, the elastic system allocates more resources to rectify the situation. Once the incoming workload diminishes and the allocated resources become under-utilized, the elastic system consolidates the load on a portion of resources and releases unused resources to reduce the costs. In this interpretation, elasticity is a feature or a means to avoid under or over-provisioning and allows elastic software to service end users with acceptable QoS while minimizing the operational costs. In the cloud citation, two elasticity strategies are defined. Horizontal elasticity is the ability to provision or release VMs which host the application, while vertical elasticity is adjusting the capacity (e.g., memory or cpu) of active individual VMs hosting the application to cope with runtime changes on demand.

In the context of control theory, a standard feedback control loop roughly looks as shown in Figure 1a. The system which is being controlled is labeled *target system* and the combination of the controller and the *target system* is labeled *controlled system*. It has a *desired output* that needs to be achieved by tuning a *controller's output* which can be one or more configurable parameters of the *target system*. The controller periodically adjusts the value of the *controller's output* (often named *control knob*) at runtime in such a way to ensure continuous satisfaction of the *desired output* despite *disturbances* in the *target system* [9]. The *disturbances* are what can affect the *measured output*, but they are not controllable [30]. The main goals of the controller is to minimize the effect of the *disturbances* on the behavior of the *target system*.

Self-adaptive cloud-based software application can be realized via a feedback control loop architecture (named MAPE loop consisting monitoring, analysis, planning, and execution, in self-adaptive software terminology [20]). Figure 1b illustrates a reference architecture, where a controller supervises a software application. The *target system* is an application deployed in a cloud environment (named cloud-based application). The *desired output* is one or more *desired QoS* attributes, such as application throughput or latency, which are monitored preodicly as the *measured output*. The number of user requests (workload) for a cloud-based application is considered as the *disturbances* which are unpredictably varying at runtime. Since the controller cannot control the workload, it should apply corrective actions and change the cloud environment in a way to meet the *desired QoS*.

More specifically, the controller implements a logic that adjusts the resources consumed by the application to accom-

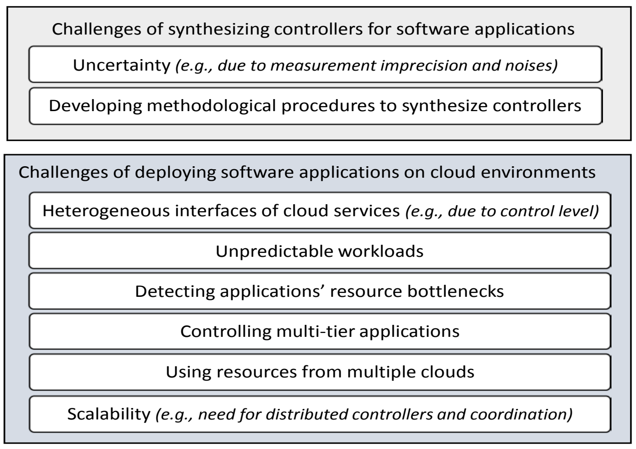
Challenges of synthesizing controllers for software applications
Uncertainty (e.g., due to measurement imprecision and noises)
Developing methodological procedures to synthesize controllers
Challenges of deploying software applications on cloud environments
Heterogeneous interfaces of cloud services (e.g., due to control level)
Unpredictable workloads
Detecting applications' resource bottlenecks
Controlling multi-tier applications
Using resources from multiple clouds
Scalability (e.g., need for distributed controllers and coordination)

Figure 2: Summary of self-adaptation challenges for cloud-based applications.

modate user requests. The controller monitors the operational condition of the cloud-based application. Based on the *controllers' output*, the controller instantiates new virtual machines (VM) or terminates existing ones, or adjusts the capacity of individual active VMs hosting the application to cope with runtime changes on demand. Application end users access the *target system* through its public end point. Finally, the cloud provider calculates the total usage and bills the application owners for the cost of their applications deployed and running on the cloud environment. In order to design and maintain an effective feedback control loop for cloud-based applications, there are still several challenges remaining which will be discussed in the next section.

To exemplify the challenges, consider a multi-tenant web application, a bulletin board application[1] that enables users (tenants) to browse and submit stories or comments. Some stories may attract a huge number of visitors in a short period of time. Therefore, as the nature of this web application may include sudden bursts or occasionally daily or monthly peaks, the application must be able to quickly adjust the deployment infrastructure. To achieve a desired level of performance requirements, this application can be implemented as a cloud-based application and employed a self-adaptive solution. In other words, the application owner decides to adopt a controller in order to be able to satisfy the performance requirements of the application users in spite of dynamic workload at runtime.

3. RESEARCH CHALLENGES

In this section we scratch the surface by exploring the most important challenges of making a cloud-based application self-adaptive and briefly propose some hints to pave the way for this potentially valuable research direction. In order to have a structure, we take into account the following definition of cloud-based application. According to NIST [24], a cloud-based application is a piece of software system that is deployed over a shared pool of configurable and virtual resources (e.g. cpu, memory) that can be auto-scaled. As shown in Figure 2, we start with the relevant challenges for designing a controller for a software application, then extend them by bring up challenges when deploying the application on the cloud and finally more long-term challenges will be presented that arise due to the cloud computing trend and hence are future requirements for cloud-based applications. Throughout this section, consider the scenario in which the owner of multi-tenant web application as a cloud-based application aims to use or design a controller in order to be able to satisfy the performance requirements of the users in spite of dynamic workload at runtime.

Uncertainty. Designing auto-scaling mechanisms pose complexity challenges because of uncertainty [12, 22, 19] that is likely to be present in every facet of elasticity reasoning. For instance, users often find it difficult to accurately describe elasticity policies, or knowledge used for elasticity reasoning may not be accurate. Moreover, in order to make decisions about corrective control actions, monitoring tools (or sensors in general) provide input data for auto-scaling decision making. These measurements are usually not free of noise and contains random and persistent disturbances that can affect the clarity of a given property, especially in cloud environments. They may also contain irrelevant or meaningless data. This will affect elastic systems in a way that they are not be able to replicate a given measurement consistently throughout the control period. These are some of the potential sources of uncertainty in elastic systems. These sources, if not explicitly taken into account in elasticity reasoning, impact on runtime scaling decisions, often resulting in unreliability. Theoretically, elasticity should accommodate even unexpected changes in capacity, adding resources when needed and reducing them during periods of low demand, but the decisions to adjust capacity must be made automatically and accurately to be cost effective. If elasticity decisions are made without considering uncertainty, then available resources may not be sufficient or cost-effective at a certain point in time. Several approaches are used in practice to cope with uncertainty, e.g., in software engineering [14] or self-adaptive software [8]. However, as discussed in [19, 22], uncertainty in the context of dynamic resource provisioning for cloud-based application [12] is still unclear.

Developing methodological procedures to synthesize controllers. A Cloud is not a deployment environment to which existing software solutions can be transferred easily. Instead, it offers novel characteristics not existing in traditional deployment environments like seemingly endless resource pool [15]. Therefore, the advantages of using cloud as a deployment environment for a software systems is leveraging such characteristics. For instance, cloud elasticity can provide consistent performance while minimizing resource costs for application owners. Horizontal and Vertical elasticity, as two possible elasticity strategies in cloud, have their own pros and cons to be adopted as *control knobs* and should be used in accordance to the application requirements at runtime. From the cloud provider's perspective, the details of the applications that they host are basically black-box and not visible. This makes it difficult to accurately devise optimal set of corrective actions (i.e., adopting a proper auto-scaling controller at runtime or defining auto-scaling thresholds). Thus, the burden of such tasks falls on application owner as a cloud user [13], which do not have deep knowledge about the application workloads, cloud environment characteristics, and performance modelling. To address this challenge, the control community can provide certain generic methodological solutions to facilitate the design of controllers for software systems and consequently cloud-based applications. A solution in which the applica-

[1]e.g., `http://slashdot.org`

tion owner is only required to define a desired level of QoS attributes and put the decision making responsibility on a smart controller at runtime. As a recent and promising research work, Filieri et al. [10, 11] propose a generic and yet practical methodology to synthesis controllers for software systems. The main benefit of their methodology is to reduce the need for a strong mathematical background as a software engineer to devise ad-hoc control solutions. Based on this, having chosen a *target system*, one only needs to indicate a *controller's output* which can change the behavior of the *target system* as well as specifying a *desired output* to be achieved by the controller.

Some common issues that should be addressed while designing and maintaining elasticity controllers are as follows: (i) determining when a resource is insufficient; (ii) quantifying requirements according to application environment; (iii) identifying when and how much of resource can be added or removed without degrading the application performance; (iv) finding a safe adjustment granularity at runtime as the reaction of the application deployed on cloud for the applied *controller's outputs* is not deterministic.

Heterogeneous interfaces of cloud services. A cloud application can be deployed either in an infrastructure-as-a-Service (IaaS) or platform-as-a-service (PaaS). These two delivery models provide different levels of control on the environment which hosts the application such as the interface for monitoring and the interface for a reactive *control knobs*. For instance, while the amount of resources (e.g., memory or cpu) can be adjusted at runtime using an IaaS, such levels of control on resources are not yet possible for a PaaS. As a result, from a control perspective, applying certain control actions or monitoring some QoS attributes might not be possible in some cloud environments, so the interface between could-based applications and cloud services can be a challenging point. Therefore, both interfaces must be designed cooperatively by taking into account the control level of the deployed environment to meet application scaling requirements while efficiently supportable by the cloud provider.

Unpredictable workloads. Typically, a variety of different application types can face different workloads, or even for a certain cloud-based application, different users usually have different usage patterns [3]. Self-adaption of such applications is realized by using controllers that dynamically tune the amount of allocated resources. The change of the resources should be accordance to the changing workload (i.e., *disturbances* in control lexicon) at runtime. Such changes are sometimes very sudden and unpredictable with sporadic runtime peaks. Since controlling the workload is unrealistic, classification and using workload analyzing tools can improve workload predictions and then synthesizing controllers to target them more effectively. This knowledge is also beneficial at runtime for dynamically adopting a number of controllers to cope with various situations. Ali-Eldin et al. [3] address this research challenge by propose a workload analysis and classification tool to analyzes workloads and assign them to the most suitable elasticity controllers. In a more generic view, in order to have an effective adaptive solutions for cloud-based applications, selection of a controller among a set of synthesized controllers based on runtime situations (e.g., workload) is inevitable. For instance, during the runtime different vertical elasticity controllers (adjusting different cloud resources, e.g., memory or cpu) or horizontal elasticity controller can be adopted for a cloud-based application. Therefore, investigating on solutions in which dynamic switching among various controllers are doable at runtime is a valuable research direction.

Detecting applications' resource bottlenecks. The host cloud environment should be able to well provide resources which are critical for the application at runtime. However, in spite of the importance of identifying the nature of application and its resource bottleneck before deployment on cloud environments, application owners do not pay attention to this issue while choosing a cloud environment. Without enough knowledge of what is the application bottleneck, designing corrective actions (*control knob*) is impossible. Different resources can be the main reasons of performance degradation for an application at runtime. For instance, an application can be cpu-intensive, memory-intensive, IO-intensive, or a combination of them. An auto-scaling controller should be able to adjust the allocation amount of such resources. To this aim, (i) bottleneck detection should be applied on an application before synthesizing the controller; (ii) application should be deployed on a cloud environments which can provide elasticity and control permission on the detected resources. Although, utilizing methods such as the "trigger-less black-box bottleneck detector" presented in [29] or familiarity with the potential cloud-based application categories [25] are possible solutions, software engineering community can still provide clearer guidelines or more effective tools to facilitate this process for cloud community.

Controlling multi-tier applications. The pervasive and popular architectural patterns for a cloud-based application is the 3-tier pattern [15]. It comprises *presentation tier* (representing user interface), *business tier* (featuring the main business logic), and *data tier* (managing the persistent data). Realizing self-adaptation of a multi-tier application deployed in cloud environment arises new challenges and acquire research attention. In a multi-tier application, every tier can be the main reason of performance degradation in a specific period of time; therefore, a possible solution can be adopting separated controllers for each tier and then use coordination methods such as message passing techniques among these tiers to make them isolate and avoid cascading effects. Each controller can pass the monitored data of its own tier as part of the input for controllers at other tiers.

Using resources from multiple clouds. The traditional approach of using a single cloud as the only deployment environment for an application has several limitations in terms of QoS (e.g., availability, delay), vendor lock-in, unoptimized renting cost for worldwide users [27]. Therefore, using resources from multiple clouds has envisioned as a future trend for cloud community. In such a model, dependent tiers of a single application can be distributivity deployed across multiple cloud environments. Therefore, the previous introduced challenges pose more complexity and enlightening solutions for interoperability and distributed controllers is getting necessary. As a recent work, Copil et al. [5] propose control mechanisms to address the elasticity of a multiple clouds deployment model.

Scalability. On one hand, software applications tend to be more large-scale and distributed; therefore, cloud environments are the most suitable environment to host such distributed applications. On the other hand, centralized control of a large-scale distributed system is seldom feasible. A solution which proposes a hierarchical control and leverages distributed controllers seems practical. However, this causes a problem of co-existence and possible inconsistencies and interferences between controllers. Hence, coordination is recognized as an important challenge, not completely solved by existing research [7, 17, 28], and requiring special attention [16].

4. CONCLUSION

Although feedback control is a powerful approach to construct any adaptive software system, in this paper our focus, as a member of cloud community, was on self-adaptive cloud-based applications. We highlighted the potential research challenges in the self-adaptation process of cloud applications in the perspective of control engineers. We first discussed the research challenges which cause due to the nature of software applications and then we extended them considering cloud services as a deployment environment for such applications. Along with the discussion of these challenges, we briefly proposed possible solutions. The listed challenges are not meant to be comprehensive but rather aim to help early researchers to find appropriate topics worth investigating on both communities, or for those who have little knowledge of the other community. There are still some open issues which were not covered in this paper such as the need of model-based controllers for design-time training and runtime model tuning, or the lack of benchmark application for comparing different controllers. We hope the paper can attract the attentions of cloud, software and control engineering communities to address these challenges which can potentially be the important research directions and barriers in the achieving self-adaptive cloud-based applications.

Acknowledgment

This work was partially supported by the doctoral college "Adaptive Distributed Systems", HALEY project, and the Vienna Science and Technology Fund (WWTF) through grant PROSEED.

5. REFERENCES

[1] Cloud Control Workshop Series, ONLINE: http://cloudresearch.org/workshops, 2015.

[2] Control Theory meets Software Engineering Seminar, ONLINE: http://www.martinamaggio.com/dagstuhl, 2014,.

[3] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl. Workload classification for efficient autoscaling of cloud resources. Technical report, 2013.

[4] B. H. Cheng, R. De Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer, 2009.

[5] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar. On controlling cloud services elasticity in heterogeneous clouds. 2014.

[6] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[7] F. A. de Oliveira, T. Ledoux, and R. Sharrock. A framework for the coordination of multiple autonomic managers in cloud environments. In *SASO*, pages 179–188, 2013.

[8] N. Esfahani and S. Malek. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 214–238. Springer, 2013.

[9] A. Filieri, C. Ghezzi, A. Leva, and M. Maggio. Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements. In *26th International Conference on Automated Software Engineering*, pages 283–292, 2011.

[10] A. Filieri, H. Hoffmann, and M. Maggio. Automated Design of Self-Adaptive Software with Control-Theoretical Formal Guarantees. In *ICSE*, 2014.

[11] A. Filieri, Maggio, et al. Software engineering meets control theory. In *SEAMS*, 2015.

[12] A. Gambi, G. Toffetti, and M. Pezzè. Assurance of self-adaptive controllers for the cloud. In *Assurances for Self-Adaptive Systems*, pages 311–339. Springer, 2013.

[13] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang. Adaptive, Model-driven Autoscaling for Cloud Applications. In *ICAC*, pages 57–64, 2014.

[14] D. Garlan. Software engineering in an uncertain world. In *FSE/SDP workshop on Future of software engineering research*, pages 125–128. ACM, 2010.

[15] N. Grozev and R. Buyya. Multi-cloud provisioning and load distribution for three-tier applications. *TAAS*, 9(3):13, 2014.

[16] S. Gueye, N. De Palma, and E. Rutten. Component-based autonomic managers for coordination control. In *Coordination Models and Languages*, pages 75–89, 2013.

[17] S. Gueye, N. De Palma, E. Rutten, A. Tchana, and N. Berthier. Coordinating self-sizing and self-repair managers for multi-tier systems. *Future Generation Computer Systems*, 35:14–26, 2014.

[18] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback control of computing systems*. John Wiley & Sons, 2004.

[19] P. Jamshidi, A. Ahmad, and C. Pahl. Autonomic resource provisioning for cloud-based software. In *SEAMS*, pages 95–104, 2014.

[20] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[21] M. Kihl, E. Elmroth, J. Tordsson, K.-E. Årzén, and A. Robertsson. The challenge of cloud control. In *Feedback Computing*, 2013.

[22] Q. Lu, X. Xu, L. Zhu, L. Bass, Z. Li, S. Sakr, P. L. Bannerman, and A. Liu. Incorporating uncertainty into in-cloud application deployment decisions for availability. In *CLOUD*, pages 454–461. IEEE, 2013.

[23] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva. Decision making in autonomic computing systems: comparison of approaches and techniques. In *ICAC*, pages 201–204, 2011.

[24] P. Mell and T. Grance. The NIST definition of cloud computing. 2011.

[25] A. Milenkoski, A. Iosup, S. Kounev, K. Sachs, P. Rygielski, J. Ding, W. Cirne, and F. Rosenberg. Cloud Usage Patterns: A Formalism for Description of Cloud Usage Scenarios. *CoRR*, 2014.

[26] T. Patikirikorala and A. Colman. Feedback controllers in the cloud. In *APSEC*, 2010.

[27] D. Petcu. Multi-Cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM, 2013.

[28] M. Sedaghat, F. Hernandez, and E. Elmroth. Unifying cloud management: Towards overall governance of business level objectives. In *CCGrid*, pages 591–597, 2011.

[29] H. Wu, A. N. Tantawi, and T. Yu. A Self-Optimizing Workload Management Solution for Cloud Applications. In *ICWS*, pages 483–490, 2013.

[30] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and P. Padala. What does control theory bring to systems research? *SIGOPS Operating Systems*, 43(1):62–69, 2009.